IMPLEMENTASI *FIREBASE* DALAM PENGEMBANGAN *PLATFORM* SEWA SARANA OLAHRAGA BERBASIS *ANDROID*

SKRIPSI

0702163058

TEGUH KURNIAWAN

PROGRAM STUDI SISTEM INFORMASI

FAKULTAS SAINS DAN TEKNOLOGI

UNIVERSITAS ISLAM NEGERI SUMATERA UTARA

MEDAN

2021

IMPLEMENTASI *FIREBASE* DALAM PENGEMBANGAN *PLATFORM* SEWA SARANA OLAHRAGA BERBASIS *ANDROID*

SKRIPSI

Diajukan untuk memenuhi syarat mencapai gelar sarjana

0702163058

TEGUH KURNIAWAN

PROGRAM STUDI SISTEM INFORMASI

FAKULTAS SAINS DAN TEKNOLOGI

UNIVERSITAS ISLAM NEGERI SUMATERA UTARA

MEDAN

2021

PERSETUJUAN SKRIPSI

Hal : Surat Persetujuan Skripsi

Lampiran : -

Kepada Yth:
Dekan Fakultas Sains dan Teknologi
UIN Sumatera Utara
Medan

Assalamu'alaikum Wr, Wb.
Setelah membaca, meneliti, memberikan petunjuk dan mengkoreksi serta mengadakan perbaikan, maka kami selaku pembimbing berpendapat bahwa skripsi saudara :

Nama Lengkap	: Teguh Kurniawan
Nomor Induk Mahasiswa	: 0702163058
Program Studi	: Sistem Informasi
Judul Skripsi	: Implementasi Firebase Dalam Pengembangan Platform Sewa Sarana Olahraga Berbasis Android

Dengan ini kami menilai skripsi tersebut dapat disetujui untuk dapat segera di*munaqasyah*kan. Atas perhatiannya kami ucapkan terimakasih.

Medan, 07 Januari 2021
Jumadil Awal 1442 H

Komisi Pembimbing

Pembimbing I

(Samsudin, S.T, M.Kom)
NIP. 197612272011011002

Pembimbing II

(Triase, S.T, M.Kom)
NIB.1100000122

SURAT PERNYATAAN KEASLIAN SKRIPSI

Saya yang bertanda tangan dibawah ini :

Nama	: Teguh Kurniawan
Nomor Induk Mahasiswa	: 0702163058
Program Studi	: Sistem Informasi
Judul Skripsi	: Implementasi Firebase Dalam Pengembangan Platform Sewa Sarana Olahraga Berbasis Android

Menyatakan bahwa skripsi ini adalah hasil karya saya sendiri, kecuali beberapa kutipan dan ringkasan yang masing-masing disebutkan sumbernya. Apabila dikemudian hari ditemukan plagiat dalam skripsi ini maka saya bersedia menerima sanksi pencabutan gelar akademik yang saya peroleh dari sanksi lainnya sesuai dengan peraturan yang berlaku.

Medan,07 Januari 2021

Teguh Kurniawan

NIM.0702163058

KEMENTRIAN AGAMA REBUPLIK INDONESIA
UNIVERSITAS ISLAM NEGERI SUMATERA UTARA
FAKULTAS SAINS DAN TEKNOLOGI
Jl.IAIN No.1 Medan, Kode Pos 20235
Telp. (061) 6615683-6622925. Fax. (061) 6615683
Url: www.saintek.uinsu.ac.id , Email:

PENGESAHAN SKRIPSI
Nomor: B.065/ST/ST.V.2/PP.01.1/04/2021

Judul	: Implementasi *Firebase* Dalam Pengembangan *Platrform* Sewa Sarana Olahraga Berbasis Android
Nama	: Teguh Kurniawan
Nomor Induk Mahasiswa	: 0702163058
Program Studi	: Sistem Informasi
Fakultas	: Sains dan Teknologi

Telah dipertahankan di hadapan Dewan Penguji Skripsi Jurusan Sistem Informasi Fakultas Sains dan Teknologi UIN Sumatera Utara dan dinyatakan **LULUS**

Pada hari/tanggal	: Senin, 01 Maret 2021
Tempat	: Aplikasi Zoom Meeting

Tim Ujian Munaqasyah
Ketua

Samsudin, S.T, M.Kom
NIP. 197612272011011002

Dewan Penguji,

Penguji I

Suendri, M.Kom
NIP. 198712082015031006

Penguji II

Raissa Amanda Putri, M.TI
NIP. 198907102018012002

Penguji III

Samsudin, S.T, M.Kom
NIP. 197612272011011002

Penguji IV

Triase, S.T, M.Kom
NIB. 11000000122

Mengesahkan
Dekan Fakultas Sains dan Teknologi
UIN Sumatera Utara Medan

Dr. Mhd.Syahnan, MA
NIP. 196609051991031

UNTUK AYAH, IBU, DAN ORANG-ORANG BAIK

Dengan mengucapkan Alhamdulillah, saya sangat bersyukur kepada Allah SWT karena atas rahmat dan izinnya saya bisa menyelesaikan penelitian skripsi ini hingga akhir. Kepada kedua orangtua saya, ayah saya Asril dan ibu saya Yanti dengan kasih sayang mereka yang telah membersarkan, mendidik, dan memenuhi segala kebutuhan saya, terimakasih yang tidak terhingga untuk ayah dan ibu, lalu terimakasi juga kepada saudara perempuan saya Tiara yang telah menyemangati dan selalu memberikan do'a agar pendidikan saya lancar dan tidak ada hambatan dan untuk seluruh keluarga besar dari pihak ayah dan ibu terimakasih atas dukungan dan do'a-nya.

Terimakasih kepada teman satu kontrakan dari zaman saya yang masih polos hingga sekarang yang mungkin sudah tidak terlalu polos lagi, sang jidat Ahmad Syarif, Bagus Aji Pratama, Rusdin Halamoan, Silaen Rahmad Aulia, Mbah Muhammad Ihsan, terimakasih sudah menemani masa-masa suram saya diperantauan, semoga nanti kelak kita semua menjadi orang sukses sehingga tidak ada lagi hari-hari penuh mie instan, terimakasih juga untuk teman tebengan awal semester Muhammad Syarif Hidayatullah, dan terimakasih atas semangat-semangat nya kaum perempuan-perempuan yang pernah singgah tapi tak sungguh semoga salah satu dari kalian bisa menjadi jodoh saya jika Allah mengizinkan, terimakasih juga kepada *Bringus Coffee* atas dua tiga cangkir kopi gratis dan tempat berteduh dari panas dan hujan, semoga suatu saat bisa sebesar perusahaan *Alphabet.*

Terakhir untuk orang-orang yang tidak pernah saya temui di dunia nyata tapi memberikan dedikasi yang berguna di dunia maya, terimakasih untuk abang dan kakak *programmer* yang telah membagikan ilmunya lewat artikel di *internet***,** para *youtuber programmer* yang sudah memberikan penjalasan panjang, jelas, padat, orang-orang baik hati di *stackoverflow*, tanpa kalian semua kodingan saya hanya *Hello World* semata.

KATA PENGANTAR

Alhamdulillahrabill'alamin segala puji syukur atas kehadirat Allah SWT berkat rahmat dan hidayahnya penulis dapat menyelesaikan skripsi ini yang berjudul Implementasi *Firebase* Dalam Pengembangan *Platform* Sewa Sarana Olahraga Berbasis Android.

Dalam penyelesaian karya ilmiah ini tentunya tidak lepas dari kekuarangan, baik dari aspek kualitas ataupun kuantitas yang disajikan pada materi penelitian. Maka dari itu penulis memohon kritik dan saran yang membangun kepada pembaca agar penulis dapat memerbaiki kekurangannya dimasa yang akan datang.

Dalam penyusunan karya ilmiah ini juga penulis berterima kasih kepada semua pihak yang telah membantu dalam kelancaran penyusunan dan penyelesaian karya ilmiah ini, antara lain :

1. Bapak Prof. Dr Syahrin Harahap, MA selaku rektor Universitas Islam Negeri Sumatera Utara Medan.
2. Bapak Dr. Mhd. Syahnan, MA selaku Dekan Fakultas Sains dan Teknologi Universitas Islam Negeri Sumatera Utara Medan.
3. Bapak Samsudin, S.T, M.Kom selaku Ketua Program Studi S1 Sistem Informasi Universitas Islam Negeri Sumatera Utara Medan dan selaku Dosen Pembimbing I penulis.
4. Bapak Suendri, M.Kom selaku Sekretaris Program Studi S1 Sistem Informasi Universitas Islam Negeri Sumatera Utara Medan.
5. Ibu Triase, S.T, M.Kom selaku Dosen Pembimbing II penulis.
6. Pak M.Dedi Irawan, M.Kom selaku Dosen Pendamping Penulis.
7. Pak Ilka Zufria, M.Kom selaku Dosen Pembimbing Akademik Penulis.
8. Bapak dan Ibu Dosen Program Studi S1 Sistem Informasi Universitas Islam Negeri Sumatera Utara Medan yang telah memberikan ilmu dan pengalaman serta masukan dalam penyusunan karya ilmiah ini.

Semoga Allah memberikan balasan yang baik atas semua jasa dan kebaikan yang diberikan penulis yang InsyaAllah dapat dijadikan amal jariyah dan ilmu yang bermanfaat serta semoga tulisan karya ilmiah skripsi ini bisa dapat memberikan manfaat yang dapat dikembangkan dimasa yang akan datang.

Medan, 16 Januari 2021

Penyusun,

Teguh Kurniawan

NIM.0702163058

IMPLEMENTASI *FIREBASE* DALAM PENGEMBANGAN *PLATFORM* SEWA SARANA OLAHRAGA BERBASIS *ANDROID*

ABSTRAK

Salah satu sektor bisnis yang minim inovasi *digital* adalah olahraga, seperi jasa sewa sarana olahraga yang memiliki masalah seperti cara penyewaan yang masih konvensional, pemasaran yang tidak meluas, rekapitulasi keuangan secara manual atas transaksi yang berjalan sehingga menyebabkan kesalahan perhitungan. Penelitian ini menggunakan metode kualitatif dalam pengumpulan data yang dibutuhkan yang bertujuan dalam pengembangan sebuah *platform* berbentuk aplikasi berbasis *android* dalam memecahkan masalah-masalah diatas. Menggunakan metode pengambangan sistem *Rapid Application Development* dalam *platform* ini dilakukan sebuah integrasi dengan layanan *firebase*, seperti *firebase authentication, firebase realtime database, firebase cloud messaging*. Untuk melakukan integrasi antara aplikasi *android* dan *firebase* dapat dilakukan dengan menggunakan SDK yang disediakan *firebase*, serta mendaftarkan *key* aplikasi ke *firebase. Platform* ini mampu memfasilitasi bagi penyedia jasa sewa sarana olahraga (futsal, voli, basket, bulu tangkis) dan pelanggannya dalam melakukan transaksi secara cepat, *online* dan *realtime*, pemasaran yang luas, dan pengelolaan administrasi yang terkomputasi dan sistematis dengan implementasi *firebase*.

Kata Kunci : *Android, Firebase, Platform Sewa Sarana Olahraga*

IMPLEMENTATION OF FIREBASE IN DEVELOPING ANDROID APPS FOR THE PLATFORM RENT SPORT FACILITIES

ABSTRACT

One of the business sectors that lacks digital innovation is sports, such as sports facilities rental services that have problems such as conventional rental methods, marketing that is not widespread, manual financial recapitulation of ongoing transactions, which causes calculation errors. This research uses a qualitative method of collecting the necessary data which aims at developing a platform in the form of an Android-based application in solving the above problems. Using the Rapid Application Development system development method on this platform, an integration is carried out with Firebase services, such as Firebase authentication, firebase realtime database, firebase cloud messaging. To integrate between Android and firebase applications, you can use the SDK provided by Firebase, and register the application key with Firebase. This platform is able to facilitate sports facility rental service providers (futsal, volleyball, basketball, badminton) and their customers in making transactions quickly, online and in real time, extensive marketing, and computed and systematic administrative management with the implementation of firebase.

Keywords: Android, Firebase, Rent sport facilities platform

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1. Latar Belakang

Seiring berkembangnya waktu hingga saat ini dunia memasuki era revolusi industri 4.0. Banyak teknologi baru lahir dari ide-ide inovatif yang semakin memudahkan kegiatan dan kebutuhan manusia dalam kehidupan sehari-hari. Teknologi tersebut dimanfaatkan pada berbagai bidang seperti kesehatan, pendidikan, sosial, budaya, dan juga *enterpreneur*. Sebuah cabang keilmuan tentang bagaimana mengkolaborasikan sebuah kemajuan tekonologi dengan *enterpreneur* disebut dengan *Information System Enterprise*, pada era revolusi industri 4.0 ini sudah banyak pada kompetitor dibidang bisnis mencapai tujuan bisnisnya dengan berlandaskan hal-hal yang ada pada konsep *Information System Enterprise*.

Salah satu *enterpreneur* dibidang olahraga ialah jasa penyewaan sarana olahraga seperti lapangan olahraga, banyak masyarakat di Kota Medan menyediakan sebuah lahan atau tempat untuk disewa oleh orang-orang yang ingin memakai lapangan dan melakukan kegiatan olahraga yang dicintainya. Akan tetapi pemasaran mereka sukar kali terbatas, karena letak yang tidak strategis sehingga calon penyewa tidak tahu, dan juga tarif harga yang tidak mengikuti persaingan bisnis dibidang ini sehingga calon penyewa lebih memilih tempat yang murah. Disisi lain para penyewa juga sering kali mendapatkan masalah dalam proses penyewaan lapangan, sehingga menyita waktu dan tenaga hanya untuk sebuah proses reservasi. Seperti calon penyewa harus datang kelokasi hanya untuk melakukan sebuah proses reservasi, dan lagi sering calon penyewa merasa kecewa ketika sudah sampai dilokasi akan tetapi jadwal akan yang diinginkan sudah direservasi orang terlebih dahulu. Permasalahan lain yang dialami oleh calon penyewa ialah minimnya informasi terhadap penyedia jasa sarana olahraga, khususnya dikota Medan, masih banyak warga sekitar yang minim informasi dimana saja terdapat penyedia jasa layanan sewa sarana olahraga ini yang dekat dengan sekitar mereka.

Penelitian yang berhubungan dengan reservasi lapangan sudah pernah dilakukan sebelumnya adapun peneilitian terdahulu terkait permasalahan yang dipaparkan diatas adalah pada tulisan karya ilmiah "Rancang Bangun Aplikasi Penyewaan Lapangan Futsal Berbasis *Android*" (Ratnasari, Hadi, & Budiarto, 2018) dalam karya ilmiah tersebut dipaparkan hanya sebatas lapangan futsal, sistem hanya sampai reservasi, menggunakan *web service* dalam penanganan data. Kemudian pada penelitian karya ilmiah yang berjudul "Sistem Informasi Reservasi Lapangan Futsal Pada Futsal Corner Menggunakan Metode Waterfall" dijelaskan bahwa sistem yang dibangun masih belum bersifar *real-time*, dibangun berbasis dekstop, hanya terfokus pada satu lapangan futsal (Swastika & Khasanah, 2017).

Berdasarkan permasalahan yang dipaparkan peneliti tertarik untuk memberikan sebuah inovasi dibidang *enterprenur* olahraga ini untuk mengembangkan sebuah aplikasi yang memudahkan para calon penyewa dalam mendapatkan informasi penyedia jasa layanan sarana olahraga pada empat lapangan olahraga yaitu lapangan futsal, voli, basket, dan juga badminton, melakukan proses reservasi terjadwal dengan mudah dengan menggunakan teknologi *cloud service Firebase*, kemudian juga membantu para penyedia jasa layanan sarana olahraga dalam memasarkan jasanya pada publik secara digitalisasi dan mengikuti *trend* persaingan dunia bisnis. Adapun judul penilitian ini ialah **"Implementasi *Firebase* Dalam Pengembangan *Platform* Sewa Sarana Olahraga Berbasis *Android*".** Diharapkan dengan pengembangan aplikasi ini bisa membantu para *enterprenur* di era digitalisasi ini.

1.2. Rumusan Masalah

1. Bagaimana rancangan *platform* yang baik untuk membangun sebuah aplikasi yang memenuhi kebutuhan pengguna ?
2. Bagaimana membangun teknologi *Firebase* pada *platform* sewa sarana berbasis *android* ?

1.3. Tujuan Penelitian

1. Merancang *platform* yang baik dengan konsep *real-time* dan mampu memberikan dampak nilai *enterpreneur* yang lebih baik dam efektif kepada penyedia jasa layanan sewa lapangan dan memudahkan penyewa dalam memakai jasa ini.
2. Membangun *platform* sewa sarana olahraga dengan memakai teknologi *Firebase auth, Firebase Realtime Database, Firebase cloud messanging* agar sistem yang dibangun efektif dan efesien.

1.4. Manfaat Penelitian

1. Bagi Universitas

a. Universitas dapat meningkatkan kualitas lulusannya.
b. Mengetahui kemampuan mahasiswa dalam menguasai teori yang di-peroleh selama masa pendidikan.
c. Sebagai bahan rujukan kepada pembaca atau peneliti selanjutnya.

2. Bagi Penulis

a. Menambah pengalaman serta wawasan dalam membangun sebuah ap-likasi berbasis *android*.
b. Menerapkan ilmu akademis yang telah didapatkan selama masa perkuliahan.
c. Untuk memenuhi salah satu syarat kelulusan strata satu (S1) Program Studi Sistem Informasi Fakultas Sains dan Teknologi.
d. Sebagai suatu karya bagi penulis sehingga membantu jenjang karir penulis dalam membuat portofolio.

3. Bagi Masyarakat

a. Para Enterprenur penyedia jasa layanan sarana olahraga dapat me-masarkan jasanya lebih luas lagi sehingga memperbersar peluang bisnis.

b. Para calon penyewa mudah dalam mendapatkan informasi dimana saja lokasi terdekat dari mereka yang menyediakan jasa layanan sarana olahraga.
c. Para calon penyewa lebih mudah dalam melakukan reservasi lapangan dengan terjadwal dan lebih akurat.

1.5. Batasan Masalah

1. Pengembangan aplikasi menggunakan *android* studio versi 4.0.0 sebagai IDE.
2. Penggunaan aplikasi akan dilakukan pada lima kecamatan yaitu di Kecamatan Percut Sei Tuan Deli Serdang, Kecamatan Medan Perjuangan Kota Medan, Kecamatan Medan Tembung Kota Medan, Kecamatan Medan Timur Kota Medan, Kecamatan Medan Helvetia Kota Medan.
3. Pada penelitian ini berfokus pada proses implementasi tiga layanan *firebase* yaitu *authentication*, *realtime database*, *cloud messageging* yang mana dalam hal ini dibanggun menggunakan bahasa pemrograman *Kotlin*
4. Sistem hanya dapat berjalan pada *android* minimal versi 5.0
5. Sarana olahraga yang disewa berupa lapangan futsal, badminton, basket, dan voli.
6. Sistem tidak terfokus pada hal keamanan aset informasi.
7. Pemilik lapangan hanya dapat mendaftar pada satu lapangan dan kategori usaha
8. Voucher digunakan hanya pada lapangan terkait sebagai media pembayaran
9. Sistem menggunakan dua aplikasi guna membatasi level administrator dan client.
10. Sistem mengatur jadwal buka layanan reservasi yaitu mulai dari pukul 08:00 WIB hingga pukul 23:00 WIB.
11. Sistem mampu melihat jadwal secara realtime pada tiga hari saja
12. Penyewa yang melakukan pembayaran via cash ditempat tidak dapat membatalkan pesanan / mengubah jadwal reservasi.

13. Menggunakan autentikasi *google account*, nomor *handphone*, dan email pada aplikasi client, dan hanya email saja pada sisi administrator.
14. Sistem diuji dengan skenario atau metode *black box.*
15. Sistem belum mampu melakukan integrasi dengan *maps* terkait alamat lapangan.

BAB II
TINJAUAN PUSTAKA

2.1. Implementasi

Definisi Implementasi ialah sebuah penerapan ide, konsep, kebijakan, atau inovasi yang dituangkan dalam bentuk sebuah tindakan praktis sehingga memberikan sebuah dampak berupa perubahan pengetahuan, keterampilan, ataupun-nilai dan sikap (Oemar Hamalik, 2007). Implementasi juga dapat diartikan sebagai suatu perbuatan atau pelaksanaan dari sebuah rencana yang disusun secara matang dan terperinci jauh sebelum pelaksanaan tersebut dilakukan. Implementasi melakukan penyesuaian antara sistem yang asli dengan sistem yang akan direkayasa. Tujuan dari dilakukannya implementasi sendiri adalah sebagai berikut

1. Membuat desain sistem selama melakukan penelitian / analisa pada sistem.
2. Menguji serta mendokumentasikan prosedur dan program yang dibutuhkan.
3. Menyelesaikan desain sistem yang sudah disetujui.
4. Memperhitungkan sistem yang sudah dibuat sesuai kebutuhan pengguna (Irawan & Herviana ,2019).

Berdasarkan hal diatas dapat disimpulkan bahwa implementasi merupakan sebuah aksi dari suatu ide seseorang terkait suatu masalah untuk memecahkan hal tersebut, dalam konteks pembuatan sistem implementasi dapat diartikan memanfaatkan teknologi yang ada agar sistem yang lama dapat menjadi lebih optimal dalam pelaksanaanya.

2.2. Sistem

Sistem dapat diartikan sebagai kumpulan orang yang saling berkerja sama dengan ketentuan aturan-aturan yang sistematis dan terstruktur untuk membentuk suatu kesatuan yang melaksanakan fungsi untuk mencapai tujuan (Elisabet & Rita, 2017). Sistem juga dapat diartika sebagai kumpulan-kumpulan dari beberapa kompnen dan subsistem yang saling terikat dan berkeja sama dalam mencapi sua-

tu tujuan (Rini Asmara, 2016). Sehingga sistem dapat diartikan suatu kumpulan yang saling terintegrasi serta sifatnya sistematis untuk mencapai tujuan tersebut.

2.3. Karakteristik Sistem

Model umum pada semua sistem adalah *input, process, output*. Sebuah sistem memiliki karakteristik atau sifat tertentu yang mencirikan bahwa hal tersebut bisa memenuhi kriteria sehingga disebut sebuah sistem. Adapun karakteristik yang dimaksud ialah :

1. Komponen sistem (*Components*)

   Suatu sistem bisa terdiri dari beberapa kumpulan komponen yang berkerja sama, untuk membentuk satu kesatuan. Komponen sistem yang dimaksud dapat berupa subsistem, yang mempunyai sifat dari sistem yang menjalankan fungsi tertentu yang mempengaruhi proses sistem secara menyeluruh.

2. Batasan Sistem (*Boundary*)

   Batasan sistem atau *Bundary* sistem adalah daerah yang membatasi antara sistem dan sistem yang lain/sistem lingkungan luarnya. Batasan sistem ini membuat suatu sistem dipandang sebagai sebuah kesatuan yang tidak dapat dipisahkan.

3. Lingkungan Luar Sistem (*Enviroment*)

   Bentuk lingkungan yang ada pada batasan sistem yang mempengaruhi operasi sistem tersebut disebut lingkungan luar sistem. Lingkungan luar harus tetap dijaga dan dipelihara (*maintenance*). Dan pada lingkungan luar sistem yang merugikan sistem tersebut harus diatasi.

4. Penghubung sistem (*Interface*)

   Jembatan yang menghubungkan antara sisetm dan subsistem lain disebut penghubung sistem atau *interface*. Penghubung ini membuat sumber-sumber daya mengalir dari satu subsistem ke subsistem lain.

5. Masukan sistem (*Input*)

   Energi yang dimasukkan pada sebuah sistem disebut memasukkan sistem, yang bisa berupa seperti pemeliharaan (*maintenance input*) dan

sinyal (*signal input*). Contohnya pada suatu unit sistem komputer, program adalah masukan sisitem yang digunakan untuk mengoprasikan komputernya dan data merupakan sinyal untuk diolah menjadi informasi.

6. Keluaran sistem (*Output*)

   Hasil energi yang diolah serta diklasifikasikan menjadi sebuah keluaran yang berguna (*output*). Keluaran ini merupakan bagian dari subsistem yang lain seperti sitem informasi yang bisa digunakan sebagai masukkan untuk pengambilan pada sebuah keputusan atau hal-hal lain yang menjadi *input* bagi subsistem lain.

7. Pengolah sistem (proses)

   Sebuah sistem bisa memiliki sebuah proses yang akan mengubah masukkan menjadi sebuah keluaran, seperti sistem pada akuntansi. Sistem ini akan mengolah data transaksi menjadi beberapa laporan yang dibutuhkan oleh pihak manajemen.

8. Sasaran sistem (*Objective*)

   Sebuah sistem haruslah memiliiki tujuan dan sasaran yang pasti dan bersifat *deterministic*. Jika sebuah sistem tidak memiliki sasaran/tujuan maka operasi sistem tidak akan berguna. Suatu sistem dikatakan berhasil bila mengenai sasaran atau tujuan yang direncanakan (Elisabet & Rita, 2017).

2.4. Informasi

Informasi adalah data yang telah diolah menjadi sebuah bentuk yang berarti bagi penerimannya dan bermanfaat dalam pengambilan keputusan saai ini atau saat mendatang. Selain itu, pendapat lain menjelaskan informasi merupakan sekumpulan fakta-fakta yang telah diolah menjadi berbentuk data, sehingga dapat menjadi lebih berguna dan dapat digunakan oleh siapa saja yang membutuhkan data-data tersebut sebagai pengetahuan ataupun dapat digunakan dalam pengambilan keputusan (Kadir, 2003). Infrormasi merupakan data yamg telah diorganisir sehingga memberikan arti dan nilai kepada penerimanya, yang mana sumber dari informasi dalah data. Data merupakan bentuk jamak dari bentuk tunggal atau *item*,

dan data merupakan kenyataan yang menggambarkan suatu kejadian nyata, dalam dunia bisnis kejadian nyata tersebut merupakan perubahan nilai yang disebut dengan transaksi. (Andalia & Setiawan, 2015).

2.3.1. Nilai Informasi

Nilai dari sebuah informasi ditentukan dari dua hal, yaitu manfaat dan biaya untuk mendapatkanya. Suatu informasi dikatakan bernilai apabila manfaat yang diperoleh lebih berharga dibandingkan dengan biaya untuk mendapatkanya. Keuntungan pada informasi umunya tidak bisa dinilai dengan uang,akan tetapi dapat di prediksi dari nilai efektifitasnya. Nilai informasi umumnya dapat dihubungkan dengan analisis *cost effectiveness* atau *cost benefit*. Nilai informasi didasarkan pada sepuluh sifat sebagai berikut :

1. Mudah diperoleh

   Sesuatu dikatakan informasi apabila memliki kemudahan dan kecepatan untuk memperoleh informasi tersebut. Kecepatanya dapat diukur, misalnya 1 menit vs 24 jam. Akan tetapi berapa nilai bagi pemakai informasi sulit untuk mengukur nilai tersebut.

2. Luas dan lengkap

   Sesuatu dikatakan informasi apabila menunjukan kelengkapan isi informasi. Hal ini tidak hanya mengenai bersarnya informasi yang didapatkan, akan tetapi juga mengenai *output* dari informasi tersebut. Sifat ini sangat kabur dan karena itu sulit untuk mengukurnya.

3. Ketilitian

   Sifat ini berkaitan pada tingkat kebebasan dari kesalahan *output* informasi. Pada besarnya data yang besar biasanya terdapat dua jenis kesalahan, yaitu kesalahan pencatatan dan kesalahan perhitungan.

4. Kecocokan

   Kecocokan yang dimaksud adalah seberapa baik keluaran informasi dalam hubunganya dengan permintaan para pemakai. Isi informasi harus ada hubunganya dengan masalah yang sedang dihadapi sedangkan semua keluaran yang lainya tidak berguna. Sifat ini sulit mengukurnya.

5. Ketepatan Waktu

   Sifat yang berkaitan dengan waktu yang sudah dilalui, waktu yang lebih pendek dari siklus untuk mendapatkan informasi. Dalam beberapa hal, ketepatan waktu dapat diukur. Contohnya seberapa banyak penjualan dapat ditingkatkan dengan menanggapi permintaan *customer* mengenai ketersediaan barang-barang inventaris.

6. Kejelasan

   Hal ini berkaitan pada tingkat kejelasan informasi. Informasi hendaknya terbebas dari istilah-istilah yang tidak jelas.

7. Keluwesan

   Hal ini berkaitan dengan apakah informasi tersebut dapat digunakan untuk membuat lebih dari satu keputusan, tetapi apakah juga dapat membuat beberapa keputusan. Sifat ini sulit mengukurnya, akan tetapi dalam beberapa hal dapat diukur dengan variabel nilai tertentu.

8. Dapat dibuktikan

   Hal ini berkaitan tentang sejauh apa informasi dapat diuji oleh beberapa pemakai hingga sampai didapatkan kesimpulan yang sama.

9. Tidak ada prasangka

   Hal ini berkaitan dengan ada atau tidaknya keinginan untuk mengubah informasi tersebut untuk mendapatkan kesimpulan yang telah diarahkan sebelumnya.

10. Dapat diukur

    Hal ini berkaitan dengan hakikat informasi yang dihasilkan oleh sistem informasi formal. Meskipun kabar angin, desas-desus, dugaan-dugaan klenik, dan lainya juga sering dianggap sebagai informasi, namun hal-hal tersebut bearada diluar lingkup pembahasan (Sutabri, 2004).

2.3.2. Kualitas Informasi

Sementara itu informasi memiliki kualitas yang tediri dari tiga hal, sebagai berikut :

1. Akurat (*Accurate*)

   Informasi haruslah tidak memliki kesalahan artinya memliki keberanan dan tidak biasa atau menyesatkan. Akurat juga berarti bahwa informasi harus jelas mencerminkan maksudnya dan terdiri dari fakta.

2. Tepat Pada Waktunya (*Timelines*)

   Informasi yang sampai pada si penerima tidak boleh terlambat. Di dalam pengambilan keputusan, informasi yang sudah using tidak lagi bernilai. Bila informasi dating terlambat sehingga pengambilan keputusan terlambat dilakukan, hal itu dapat berakibat fatal bagi perusahaan.

3. Relevan (*Relevance*)

   Informasi yang disampaikan harus mempunyai keterkaitan dengan masalah yang akan dibahas dengan informasi tersebut. Informasi harus bermanfaat bagi pemakainya. Disamping karakteristik, nilai informasi juga ikut menentukan kualitasnya. Nilai informasi (*value of information*) ditentukan oleh dua hal, yaitu manfaat dan biaya untuk mendapatkanya (Kusrini & Koniyo, 2007).

2.5. *Database*

Secara sederhana *Database* atau yang dalam bahasa indonesia-nya disebut basis data bisa dikatakan sebagai sebuah pengorganisasian data dengan bantuan komputer yang memungkinkan data bisa diakses dengan mudah dan cepat (Kadir, 2010). *Database* juga dapat diartikan kumpulan dari banyak *file* yang memliki kaitan antara satu *file* dengan *file* yang lain sehingga membentuk satu bangunan data yang dapat menginformasikan sebuah perusahaan ataupun instansi dalam batasan tertentu. Bila terdapat *file* yang tidak dipadukan atau dihubungkan dengan *file* yang lainnya berarti *file* tersebut bukanlah kelompok dari satu *Database*, *file* tersebut akan dapat membentuk satu *Database* sendiri (Kristanto, 2004). *Database* bisa didefinisikan sebagai suatu bentuk penyimpanan dari informasi terpusat agar informasi yang ada mudah dicari, dikelola serta digunakan kembali. *Database* biasanya terdiri dari sebuah baris dan kolom, baris pada *Database* disebut dengan *record* dan kolom pada *Database* disebut *field* (Priyanto, 2007).

2.4.1. *Database SQL*

Basis data relasional (*Database SQL)* adalah kumpulan dari *item-item* yang distrukturkan sebagai tabel yang terdeskripsi secara formal dimana data bisa di-akses dan disusun kembali tanpa harus mengorganisasikan kembali tabel-tabel tersebut. Basis data relasional mengorganisasikan data dari satu tabel ketabel lainnya (relasi) yang berisi kolom dan baris, dengan *key* yang unik untuk setiap barisnya (Silalahi, 2018), relasi berguna dalam menjaga kelompok data sebagai koleksi yang tetap dengan bantuan tabel data yang berisi informasi terstruktur, menghubungkan semua masukan dengan cara memberikan nilai ke atribut. Ter-dapat beberapa DBMS (*Database Management System*) relasional, contohnya *Or-acle, DB2, Sybase, MySQL, MS.SQL Server dan MS Access.*

2.4.2. *Database NoSQL (Not Only SQL)*

Perusahaan-perusahaan *Web 2.0* yang besar telah mengembangkan atau mengadopsi berbagai jenis basis data *NoSQL* untuk data mereka yang terus ber-tambah sesuai kebutuhan infrastruktur perusahaan-perusahaan tersebut, misalnya *Google (BigTable), LinkedIn (Voldemort), Facebook (Cassandra), Amazon (Dy-namo),* dan sebagainya. (Silalahi, 2018)

Penggunaan *NoSQL* membuat para pengembang aplikasi dapat mengem-bangkan tanpa harus mengkonversi struktur di memori ke struktur relasional. Ba-sis data *NoSQL* dirancang untuk mengatasi isu *Big Data* dengan memanfaatkan mesin-mesin yang terdistribusi. *Database NoSQL* juga sering disebut sebagai ba-sis data *cloud* ,dan dikembangkan dengan tujuan untuk memecahkan sebuah per-masalahan basis data relasional yang mengalami permasalahan performa dalam pemrosesan data yang tidak terstruktur dan terus bertambah secara eksponensial, seperti dokumen, *email*, multimedia atau media sosial. *NoSQL* menyimpan dan mengambil data dalam bentuk yang berbeda. Terdapat empat kategori basis data yang terdapat dalam *NoSQL*, sebagai berikut:

1. *Key value store*, merupakan penyimpanan data *NoSQL* yang paling se-derhana untuk digunakan dari sudut pandang API (*Application Pro-gramming Interface*). Penyimpanan data berupa BLOB (*Binary Large*

Object) mampu menyimpan begitu saja tanpa memperhatikan komponen data apa yang disimpan.

2. *Column-family*, Sebuah sistem *sparse matrix* yang menggunakan baris dan kolom sebagai kunci.
3. *Graph Databases*, data yang disimpan menggunakan struktur grafis dengan *nodes* (entitas), *properties* (informasi tentang entitas) dan *line* (hubungan antar entitas).
4. *Document stores*, merupakan mekanisme dalam menyimpan struktur data hierarki langsung ke basis data (Silalahi, 2018)

2.6. *Cloud Computing*

Cloud computing merupakan teknologi dimana informasi disimpan disuatu server yang dapat diakses oleg *client* melalui jaringan internet, dan denganmenggabungkan teknologi komuter dengan pengembangan internet sehingga menjadi insfrastruktur komplek yang abstrak dan tersembunyi (Cancer & Alim, 2016). *Cloud computing* telah berkembang berdampingan dengan perkembangan *internet* dan *web* contoh dari *cloud computing* adalah *Microsoft Azure, Amazon Web Service, Google Cloud Firebase.*

Sistem *cloud computing* dibagi menjadi dua bagian yaitu ujung depan dan ujung belakang yang saling terhubung satu sama lain melaui jaringan *internet*.Ujung depan merupakan sisi pengguna komputer,dalam hal ini adalah *client*, dan bagian belakang adalah *cloud* atau sistem itu sendiri berupa server, dan sistem penyimpanan data dari layanan komputasi. Beberapa keunggunlan yang bisa didapatkan dari cloud computing adalah sebagai berikut :

1. Bagi para pelaku yang bergerak pada industri IT, hal ini merupakan peluang yang besar karena dengan meningkatnya penggunaan layanan dan meningkatkan penggunaan internet.
2. Integrasi aplikasi diberbagai *device* (perangkat).
3. Memungkinkan pengembangan dan implementasi dengan cepat segingga meningkatkan produktifitas.

4. Minimalisir biaya infrastruktur dalam dunia industri IT. (Intan Mutia, 2016)

2.7. *Firebase*

Firebase merupakan penyedia layanan *cloud service* dengan *back-end* sebagai servis yang letak divisi perusahaannya berada di San Fransisco, California. *Firebase* membuat sejumlah produk atau layanan untuk pengembangan aplikasi *mobile* ataupun web. *Firebase* didirikan oleh Andrew Lee dan James Tamplin pada tahun 2011 dan meluncurkan *cloud Database* secara *realtime* di tahun 2012. Perusahaan ini diakusisi oleh *Google* pada Oktober 2014 (Sonita & Fardianitama, 2018)

Firebase adalah penyedia layanan *realtime database* dan *backend* sebagai layanan. Suatu aplikasi yang memungkinkan pengembang membuat API (*Application Programming Interface*) untuk disinkronisasikan untuk *client* yang berbeda-beda dan disimpan pada *cloud*-nya *Firebase*. *Firebase* memiliki banyak *library* yang memungkinkan untuk mengintegrasikan layanan ini dengan *Android, IOS, Javacript, Java, Objective-C dan Node.JS*. Beberapa kemampuan *Firebase* lainnya yang ditawarkan adalah sebagai berikut :

1. Responsif walaupun saat *offline*. *Database Realtime* dilengkapi SDK untuk menyimpan data ke *disk* lokal. Sehingga saat *offline* pengguna dapat melakukan akses data.
2. *Firebase* memiliki tingkat keamanan yang mumpuni, karena diakuisi langsung oleh *Google*.
3. *Firebase* memberikan layanan gratis kepada pengembang aplikasi. (Sonita & Fardianitama, 2018).

2.6.1. *Firebase Realtime Database*

Firebase Realtime Database merupakan layanan *database realtime* yang ditawarkan *Firebase*. Pengguna selaku pengembang *software* juga dapat menggunakan *Database* ini untuk mengamankan data menggunakan server *Firebase* dengan *rules*/aturan yang ada. Dalam situs resminya *Database* dari *Firebase*

disebut sebagai *cloud Realtime Database* yang menyimpan bentuk data dalam bentuk JSON (*Javascript Object Notation*), *Firebase* melakukan sinkronasi data otomatis terhadap aplikasi yang terhubung padanya. Dalam *Firebase Realtime Database, Database* kita dapat mengurutkan dan melakukan penyaringan data dengan *query* NoSQL. *Database* NoSQL sendiri terdiri dari empat ciri-ciri yaitu, kunci-nilai, berbasis dokumen, berbasis kolom dan berbasisis grafik. *Firebase* sendiri memiliki kemampuan untuk mengoptimalkan akses data dalam hitugan mikrodetik daripada menggunakan *web-service* (Sudiartha, Indrayana, & Suasnawa, 2018). Berikut adalah langkah-langkah penggunaan *Firebase Realtime Database* dalam pengembangan aplikasi android :

1. Langkah pertama dalam menggunakan *Firebase Realtime Database* ini adalah dengan melakukan sinkronisasi akun email (*Gmail*) kita dengan *Firebase*.
2. Kemudian mengenalkan SHA-1 (*Secure Hash Algorithm* – 1) aplikasi kita kepada *Firebase*, hal ini berguna sebagai identitas aplikasi dalam penggunaan *Firebase Realtime Database*. Contoh SHA-1 yang dimaksud adalah seperti kode berikut :

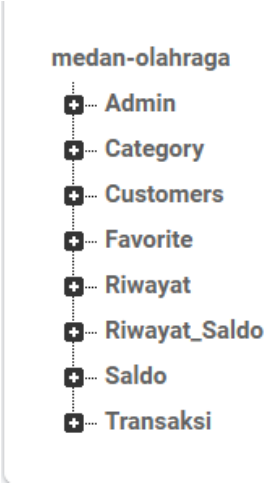
   ```
   SHA-1
   06:BA:DD:97:7A:36:E8:52:BC:BB:45:81:15:4E:C4:BD:23:45:AA:DB
   ```

3. Setelah mendaftarkan SHA-1 aplikasi tersebut, kemudian kita melakukan *depedency* pada aplikasi android kita dengan menambahkan baris kode yang disediakan oleh *Firebase*. Adapun kode *dependecy* yang dimaksudkan adalah sebegai berikut :

   ```
   com.google.Firebase:Firebase-realtime:20.0.0.
   ```

Sesuai penjelasan sebelumnya bahwa *Firebase Realtime Database* merupakan *Database* yang bersifat NoSQL. *Firebase Realtime Database* memiliki sistematika penulisan berbeda dengan *database* berbentuk relasi, penggunaan *docu-*

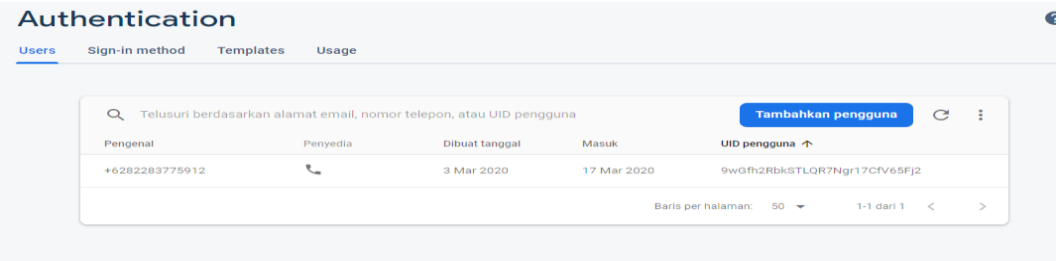
ment dan *column* digunakan sebagai pengganti tabel dan *field* dalam *database* relasi. Berikut adalah gambaran GUI dari *firebase Realtime Database.*



Gambar 2.1 Firebase Realtime Database
(sumber : console.firebase.google.com)

2.6.2. *Firebase Auth*

Layanan yang tidak kalah menarik dalam *Firebase* adalah *Firebase auth*, yang merupakan sebuah layanan yang memudahkan pengembang dalam membuat autentikasi untuk memanipulasi data email, nomor telepon, akun sosial media yang berada dalam aplikasi tersebut. Pada umumnya layanan ini dipakai dalam hal registrasi akun dan *login* pada sistem.



Gambar 2.2 Console Firebase Auth
(sumber : console.firebase.google.com)

Cara kerja *Firebase auth* adalah dengan mengetahui identitas pengguna kemudian menyimpan didalam *cloud*, dalam tahapannya pengguna wajib melakukan registrasi terlebih dahulu menggunakan email dan sandi atau autenti-

kasi dengan nomor *handphone* atau membuat sebuah data statis (untuk pengembangan sistem). Kemudian setelah pengguna melakukan registrasi pengguna bisa melakukan login dan logout dalam sistem tersebut (Ilhami, 2017). Berikut adalah langkah-langkah penggunaan *Firebase Auth* dalam pengembangan aplikasi android :

1. Langkah pertama dalam menggunakan *Firebase Realtime Database* ini adalah dengan melakukan sinkronisasi akun email (*Gmail*) kita dengan *Firebase*.
2. Kemudian mengenalkan SHA-1 (*Secure Hash Algorithm* – 1) aplikasi kita kepada *Firebase*, hal ini berguna sebagai identitas aplikasi dalam penggunaan *Firebase Realtime Database*. Contoh SHA-1 yang dimaksud adalah seperti kode berikut :

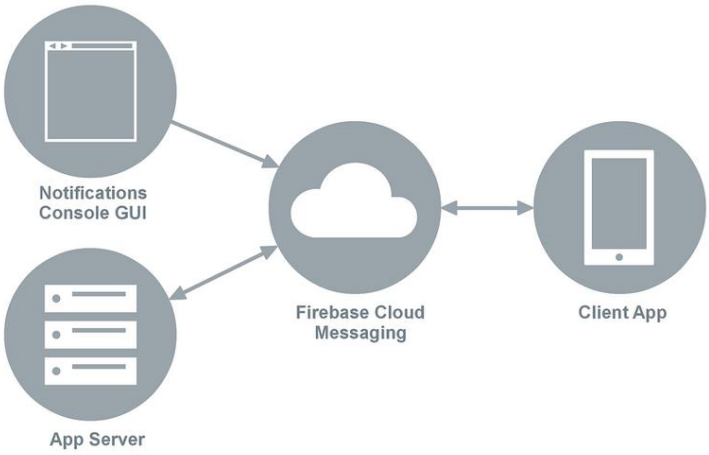
   ```
   SHA-1
   06:BA:DD:97:7A:36:E8:52:BC:BB:45:81:15:4E:C4:BD:23:45:AA:DB
   ```

3. Setelah mendaftarkan SHA-1 aplikasi tersebut, kemudian kita melakukan *depedency* pada aplikasi android kita dengan menambahkan baris kode yang disediakan oleh *Firebase*. Adapun kode *dependecy* yang dimaksudkan adalah sebegai berikut :

   ```
   com.google.firebase:firebase-auth:18.0.0
   ```

2.6.3. *Firebase Cloud Messanging*

Firebase cloud messanging (FCM) merupakan layanan *Firebase* yang menjawab permasalahan perpesanan lintas *platform* (*cross platform*) yang mampu melakukan *push notification* saat aplikasi sedang berjalan pada *background* belakang. FCM sangat membantu pengguna dalam menarik perhatian agar kembali menggunakan aplikasi, layanan FCM juga bersifat gratis (Ilhami, 2017).



Gambar 2.3 Skema Kerja Firebase Cloud Messanging
(sumber : imgs.developerpaper.com)

Pada gambar diatas, merupakan skema bagaimana FCM berkerja, sebagai *backend* FCM melakukan segala pengolahan data dalam *app server* yang dalam hal ini ditangani oleh *Google* dengan meyediakan layanan *interface* atau console dalam melakukan penanganan *notifications*, disisi *client*, *notifications* ini diterima pada aplikasi atau sistem yang didapatkan dari layanan *cloud* FCM. Dalam implementasinya FCM memiliki langkah seperti berikut untuk melakukan integrasi dengan aplikasi yang dibuat :

1. Langkah pertama dalam menggunakan *Firebase cloud messanging* ini adalah dengan melakukan sinkronisasi akun email (*Gmail*) kita dengan *Firebase*.
2. Kemudian mengenalkan SHA-1 (*Secure Hash Algorithm* – 1) aplikasi kita kepada *Firebase*, hal ini berguna sebagai identitas aplikasi dalam penggunaan *Firebase Realtime Database*. Contoh SHA-1 yang dimaksud adalah seperti kode berikut :

```
SHA-1
06:BA:DD:97:7A:36:E8:52:BC:BB:45:81:15:4E:C4:BD:23:45:AA:DB
```

3. Setelah mendaftarkan SHA-1 aplikasi tersebut, kemudian kita melakukan *depedency* pada aplikasi android kita dengan menambahkan baris kode yang disediakan oleh *Firebase*. Adapun kode *dependecy* yang dimaksudkan adalah sebegai berikut :

```
com.google.Firebase:Firebase-messanging:19.0.0.
```

2.8. *Platform*

Platform adalah sebuah struktur tata kelola yang mampu menentukan aktor siapa saja yang ikut berpatisipasi, peran aktor yang dimainkan, bagaimana cara setiap aktor berinteraksi, serta bagaimana menyelesaikan masalah dan untuk memfasilitasi hubungan, koordinasi dan kolobarasi (Rachma, 2018). Secara umum jenis platrorm terbagi menjadi dua jenis, yaitu *platform* perangkat keras (*Hardware*) dan *platform* perangkat lunak *(Software*) , *platform* perangkat keras terkait dengan keseluruhan sistem (perangkat) dan *interface*. Sedangkan *platform* perangkat lunak dikaitkan dengan aplikasi yang membuat antar pengguna saling berinteraksi.

Platform as a service (Paas) merupakan bagian dari *cloud computing,* berbentuk layanan yang menyediakan *hardware* ataupun *software* dan fungsinya memfasilitasi pengguna dalam memakai layananya. Berikut beberapa keuntungan dan manfaat dari *platform as a service* :

1. Menghemat biaya, pengguna tidak perlu memikirkan infrastruktur sumber daya yang digunakan dalam pengembangan aplikasi.
2. Efesien, tidak perlu malakukan instalasi dan konfigurasi lokal terhadap sumber daya penunjang.
3. Fleksibilitas, bila dibandingkan dengan cara manual.
4. Dapat diakses dimanapun dan kapanpun.
5. Membayar sesuai pemakaian (Cancer & Alim, 2016).

2.9. Sewa Menyewa

Sewa menyewa adalah kegiatan pemakaian atas suatu objek dengan membayar uang yang sudah disepakati, atau uang tersebut dibayarkan karena memakai sesuatu/meminjam sesuatu. Penghasilan yang diperoleh berhubungan dengan kesepakatan yang telah dilakukan sebelumnya untuk memberikan hak menggunakan harta selain tanah, suatu bangunan selama jangka waktu tertentu baik dalam perjanjian tertulis ataupun tidak tertulis sehingga harta tersebut hanya dapat digunakan oleh penerima hak selama jangka waktu yang telah disepakati sebelumnya (Tutik, Islam, & Pustaka, 2008). Dalam Pasal 1548 Kitab Undang-Undang Perdata menyatakan bahwa Sewa menyewa adalah suatu perjanjian, dengan nama pihak yang satu mengikatkan dirinya untuk memberikan kepada pihak lainnya kenikmatan dari suatu barang, selama waktu tertentu dan dengan pembayaran suatu harga, yang mana pihak belakangan itu disanggupi pembayarannya (Siti Chomsyah, 2017).

Sewa menyewa terbagi menjadi dua jenis, yaitu sewa terhadap benda dan sewa terhadap perkerjaan atau mengupah-upah.

1. Sewa terhadap benda atau sewa menyewa berlangsung atau manfaat yang berasal dari benda tertentu yang disebutkan ciri-cirinya.Contohnya seperti rumah, tanah, atau sarana seperti tempat (sarana) olahraga.
2. Sewa atas perkerjaan atau upah-mengupah menyewa (mengupah) untuk orang yang melakukan perkerjaan tertentu. Seperti jual beli jasa menjahit pakaian, membangun rumah, dan lain sebagainya (Sunarto, 2014).

2.10. Sarana Olahraga

Sarana olahraga adalah suatu bentuk permanen, baik itu ruangan maupun didalam. Sarana olahraga meliputi semua lapangan dan bangunan olahraga berserta perlengkapannya untuk melaksanakan program kegiatan olahraga. Kegiatan olahraga memerkukan ruang luas untuk bergerak, dan untuk memenuhi kebutuhan ini dibuat tempat olahraga dalam bentuk *indoor/outdoor* sehingga dapat menunjang pertumbuhan masyarakat dalam melaksanakan kegiatan olahraga yang

disukainya. Dalam penelitian ini terdapat empat lapangan sebagai sarana olahraga yang dijadikan tempat penelitian (Glady Sukma Perdana, 2017)

2.9.1. Sewa Menyewa Lapangan Olahraga Futsal

Futsal adalah sebuah permainan bola yang dimainkan dua regu, yang masing-masing regu berisikan lima orang. Tujuannya ialah memasukkan bola ke gawang lawan sama dengan sepakbola, dan cara bermain yang berfokus pada memanipulasi bola dengan kaki. Selain lima pemain utama, setiap regu juga diizinkan memiliki pemain cadangan. Tidak seperti permainan bola besar dalam ruangan lainnya, lapangan futsal dibatasi garis, bukan net atau papan. Futsal turut juga dikenali dengan berbagai nama lain. Istilah futsal adalah berlaku untuk internasional, Futsal dimainkan pada lapangan yang bersifat *indoor* ataupun *outdoor* (Glady Sukma Perdana, 2017).

Pada beberapa daerah yang menjadi tempat penelitian penulis di Kota Medan banyak pengusaha yang menyewakan sarana tempat bermain (lapangan futsal) kepada masyarakat sekitar, proses penyewaannya dilakukan dengan melakuan perjanjian antara dua pihak yaitu pemilik lapangan/penjaga lapangan untuk memakai lapangan tersebut dalam waktu yang disepakati dan harga yang sudah tetap. Adapun rumus yang dipakai dalam penghitungan penyewaan sarana yang dipakai adalah sebagai berikut :

Total Bayar = Lama Peminjaman (Jam) X Harga Sewa

Dalam rekapitulasi pengusaha penyedia layanan sewa lapangan futsal ini melakukanya dalam sebulan sekali, untuk melihat penghasilan yang didapatkan.

2.9.2. Sewa Menyewa Lapangan Olahraga Basket

Permainan Olahraga Bola basket menjadi dari sekian banyak olahraga yang paling digemari oleh warga USA dan penduduk di belahan bumi lainnya, antara lain di Amerika Selatan, Eropa Selatan, Lithuania, dan juga di Indonesia. Banyak kompetisi bola basket yang diselenggarakan setiap tahun, seperti British *Basket-*

ball League (BBL) di Inggris, *National Basketball Association* (NBA) di Amerika, dan Indonesia *Basketball League* (IBL) di Indonesia.

Lapangan bola basket bentuknya persegi panjang dengan dua standar ukuran, yakni panjang 28,5 m X 15 m pada standar *National Basketball Association* dan panjang 26 m X 14 m pada standar Federasi Bola Basket Internasional. Tiga buah lingkaran yang terdapat dalam lapangan basket memiliki panjang jari-jari yaitu 1,80 meter (Glady Sukma Perdana, 2017).

Pada beberapa daerah yang menjadi tempat penelitian penulis di Kota Medan banyak pengusaha yang menyewakan sarana tempat bermain (lapangan basket) kepada masyarakat sekitar, proses penyewaannya dilakukan dengan melakuan perjanjian antara dua pihak yaitu pemilik lapangan/penjaga lapangan untuk memakai lapangan tersebut dalam waktu yang disepakati dan harga yang sudah tetap. Adapun rumus yang dipakai dalam penghitungan penyewaan sarana yang dipakai adalah sebagai berikut :

Total Bayar = Lama Peminjaman (Jam) X Harga Sewa

Dalam rekapitulasi pengusaha penyedia layanan sewa lapangan basket ini melakukanya dalam sebulan sekali, untuk melihat penghasilan yang didapatkan. Pengeluaran yang dilakukan oleh pengusaha biasanya untuk biaya perawatan fasilitas yang tersedia ditempat layanan tersebut, seperti *wi-fi*, bola, jaring *ring*, dan lain sebagainya.

2.9.3. Sewa Menyewa Lapangan Olahraga Voli

Permainan Bola voli adalah sebuah permainan olahraga yang dimainkan oleh dua regu yang berlawanan. Masing-masing regu mempunyai enam orang pemain. Terdapat pula variasi permainan bola voli pantai yang masing-masing regu hanya mempunyai dua orang pemain. Persatua Olahraga Bola Voli Internasional disebut dengan FIVB (*Federation Internationale de Volleyball*) sebagai induk organisasi internasional, sedangkan di Indonesia disebut PBVSI (Persatuan Bola Voli Seluruh Indonesia). Ukuran lapangan bola voli yang umumnya mem-

iliki luas 9 m x 18 m. Garis batas serang untuk pemain belakang berjarak 3 meter dari garis tengah (sejajar dengan jaring). Garis tepi lapangan adalah 5 meter. Ukuran tinggi net cabang putra 2,43 meter dan untuk net cabang putri memiliki tinggi 2,24 meter (Glady Sukma Perdana, 2017).

Sistematika pada sewa penyewaan lapangan voli juga sama dengan futsal dan basket, umumnya mereka melakukan penghitungan berdasarkan berapa lama lapangan tersebut dipakai, akan tetapi dalam hal ini beberapa penyedia layanan tidak menyediakan fasilitas seperti bola voli.

2.9.4. Sewa Menyewa Lapangan Olahraga Badminton

Lapangan bulu tangkis/badminton berbentuk persegi panjang. Garis-garis yang ada mempunyai ketebalan 40 mm dan harus berwarna kontras terhadap warna lapangan. Warna yang disarankan untuk garis adalah putih atau kuning. Permukaan lapangan disarankan terbuat dari kayu atau bahan sintetis yang lunak. Permukaan lapangan yang terbuat dari beton atau bahan sintetik yang keras sangat tidak dianjurkan karena dikhawatirkan membuat cedera pada pemain. Jaring setinggi 1,55 m berada tepat di tengah lapangan. Jaring harus berwarna gelap kecuali bibir jaring yang mempunyai ketebalan 75 mm harus berwarna putih (Glady Sukma Perdana, 2017)

Sistematika pada sewa penyewaan lapangan badminton juga sama dengan-hal diatas, umumnya mereka melakukan penghitungan berdasarkan berapa lama lapangan tersebut dipakai, akan tetapi dalam hal ini beberapa penyedia layanan tidak menyediakan fasilitas *Shuttlecock* dan raket.

2.11. *Android*

Android dalam istilah bahasa inggris ialah robot yang menyerupai manusia, *Android* merupakan sistem operasi yang dikembangkan untuk perangkat *mobile* berbasis *linux* sebagai kernel nya (Satyaputra & Aritonang E.M, 2016). Sistem operasi sendiri dapat diartikan sebagai jembatan penghubung antara *user* dan hardware (dalam hal ini *device*). Android merupakan sistem operasi yang bersifat

open source dan hingga kini tercatat sebagai salah satu sistem operasi yang banyak beroperasi di *device smartphone* dan *tablet.*

Sejarah pemngembangan android bermula pada tahun 2003 dimulai dengan berdirinya *Andoid Inc*, kemudian pada tahun 2005 *Android Inc* diakuisi oleh Google, lalu pada tahun 2007 dibentuk Open Handset Alliance (OHA), sebuah konsorium dari beberapa perusahaan, yaitu *Texas Instruments, Broadcom Corporation, Google, HTC, Intel, LG, Motorola, Nvida, Qualcomm, Samsung Electronics, Sprint Nextel, dan T-Mobile* dengan tujuan untuk mengembangkan standar terbuka untuk perangkat *mobile*. Pada tanggal 9 Desember 2008, diumumkan bahwa 14 orang anggota baru akan bergabung dengan proyek Android, yaitu *PacketVideo, ARM Holdings, Atheros Communictions, Asustek Computer Inc, Garmin Ltd, Softbank, Sony Ericsson, Toshiba Corp, Vodafone Group Plc* (Maiyana, 2018).

Gambar 2.4 Logo Sistem Operasi Android
(sumber : id.wikipedia.com)

2.10.1. Struktur Aplikasi *Android*

Struktur aplikasi android atau fundamental aplikasi ditulis dalam bahasa pemrograman Java, Kode java dikompilasi bersama *resource file* yang dibutuhkan oleh aplikasi. Prosesnya di *package* oleh *tools* yang dinamakan *aps tools* kedalam paket *Android*, sehingga menghasilkan *file* berekstensi **.apk** dimana *file* ini disebut dengan aplikasi dan dapatkan dijalan pada perangkat *mobile android*. Terdapat empat komponen pada aplikasi Android, yaitu :

1. *Activity*, merupakan komponen untuk menyajikan tampilan pemakai (*interface user*) kepada pengguna.
2. *Service*, merupakan komponen yang tidak memiliki tampilan pemakai (*user interface*), tetapi server berjalan secara *backgrounds.*
3. *Broadcast Receiver,* merupakan komponen yang berfungsi menerima dan bereaksi untuk menyampaikan notifikasi.
4. *Content Provider,* merupakan komponen yang membuat kumpulan aplikasi data secara spesifik, sehingga bisa digunakan oleh aplikasi lain (Supardi, 2017).

2.10.2. Versi *Android*

Banyak *Smartphone* dan *Tablet* yang menggunakan sistem operasi dengan versi yang berbeda. Semakin versi tinggi fiturnya, semakin canggih *smartphone* dan *Tablet* tersebut. Telepon pertama yang memakai sistem operasi Android adalah HTC Dream yang dirilis pada tanggal 22 Oktober 2008 (Supardi, 2017). Beberapa uraian versi android, yaitu :

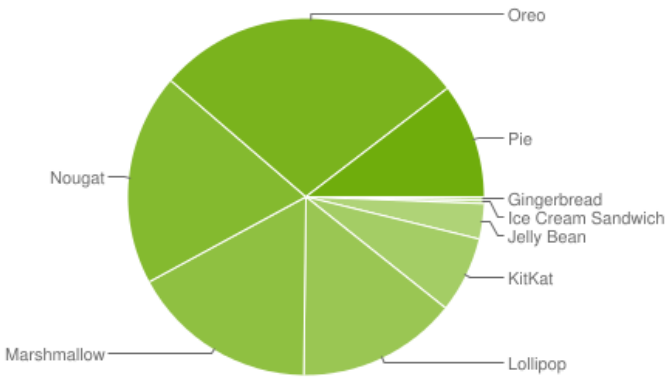
Tabel 2.1 Versi Android Hingga Tahun 2019
(sumber : developers.google.com)

No	Nomor Versi	Nama Versi	Tanggal Rilis
1	(Belum Memakai)	Android Beta	5 November 2007
2	1.0	Android 1.0	23 September 2007
3	1.1	Android 1.1	9 Februari 2009
4	1.5	Cupcake	27 April 2009
5	1.6	Donut	15 September 2009
6	2.0	Éclair	26 Oktober 2009
7	2.0.1	Éclair	3 Desember 2009
8	2.1	Éclair	12 Januari 2010
9	2.2	Froyo	20 Mei 2010
10	2.2.1	Froyo	18 Januari 2011
11	2.2.2	Froyo	22 Januari 2011
12	2.2.3	Froyo	21 November 2011

13	2.3	Gingerbread	6 Desember 2010
14	2.3.3	Gingerbread	9 Februari 2011
15	2.3.4	Gingerbread	28 April 2011
16	2.3.5	Gingerbread	25 Juli 2011
17	2.3.6	Gingerbread	2 September 2011
18	2.3.7	Gingerbread	21 September 2011
19	3.0	HoneyCumb	22 Februari 2011
20	3.1	HoneyCumb	10 Mei 2011
21	3.2.1	HoneyCumb	20 September 2011
22	3.2.2	HoneyCumb	30 Agustus 2011
23	3.2.4	HoneyCumb	Desember 2011
24	3.2.6	HoneyCumb	Februari 2012
25	4.0.1	ICS	18 Oktober 2011
26	4.0.2	ICS	28 November 2011
27	4.0.3	ICS	16 Desember 2011
28	4.0.4	ICS	29 Maret 2012
29	4.1	Jelly Bean	9 Juli 2012
30	4.4	KitKat	31 Oktober 2013
31	5.0	Lolipop	12 November 2014
32	5.1	Lolipop	25 Juni 2014
33	6.0	MarshMallow	5 Oktober 2015
34	7.0	Nougat	22 Agustus 2016
35	8.0	Oreo	21 Maret 2017
36	9.0	Pie	Agustus 2018
37	10	Q	September 2019

Berdasarkan analisis data yang dilakukan oleh *Google* selaku pengembang sistem operasi android hingga 7 Mei 2019, tercatat bahwa pengguna *android* versi lolipop kebawah sudah mulai ditinggalkan, saat ini pengguna *device android* lebih

banyak memakai versi 5.0 keatas, tentu saja alasannya ialah karena fitur yang ditawarkan oleh sistem operasi ini lebih canggih.



Gambar 2.5 Diagram Pengguna Android Tahun 2019
(sumber : developer.android.com)

2.10.3. Bahasa Pemrograman Kotlin

Kotlin adalah sebuah bahasa pemrograman yang berjalan diatas *Java Virtual Machine* dikembangkan oleh JetBrains, Kotlin merupakan bahasa pemrograman yang cocok untuk dikembangkan dalam android dan mendapatkan dukungan/rekomendasi resmi dari *Google* selaku pengembang resmi dari sistem operasi *android*. Kotlin juga dikembangkan dari bahasa pemrograman *Java* secara *behavior* gaya penulisan dikotlin tidaklah jauh berbeda dengan *Java* yaitu menggunakan konsep *Object Oriented Programming* (OOP). Sama seperti *Java* kotlin juga mampu membuat aplikasi berbasis dekstop, web, dan penanganan *backend* lainnya. (S. Doug, 2010)

Kotlin memilik kelebihan sebagai berikut, sehingga dikatakan lebih baik dari bahasa pemrograman *Java*, yaitu sebagai berikut :

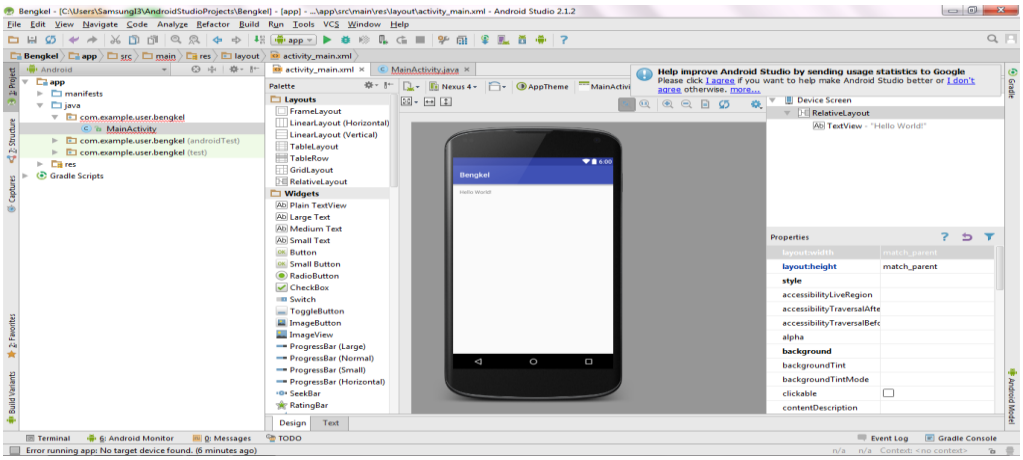
1. Memiliki kemampuan *Null Safety*, sehingga meminimalisir sebuah *error* yang disebut *NullPointerException.*
2. Penulisan kode yang lebih ringkas dibanding *Java*
3. Dukungan IDE dari *JetBrains* sebagai pengembang resmi. (R.K. Panchal dan A. K. Patel, 2017)

2.10.4. *Android Studio* IDE

Android Studio adalah sebuah IDE (*Integrated Development Environment*) untuk pengembangan aplikasi *android* yang di *support* oleh *google* dan *Intelij IDEA*, yang merupakan IDE bersifat *open source* atau gratis (Nazruddin, 2019). Peluncuran *Android Studio* diumumkan oleh *Google* pada 16 Mei 2013 di acara *Google I/O Conference*, dan sejak saat itulah *Android Studio* menggantikan *Eclipse* sebagai IDE resmi pengembangan aplikasi *android.*

Sama seperti pendahulunya yaitu *Eclipse*, pada *Android Studio* juga memiliki *Android Development Tools* (ADT) sebagai sebuah *Integrated Environment* yang kuat untuk membangun sebuah aplikasi *android* (Lengkong, Sinsuw, & Lumenta, 2015). *Android studio* memiliki beberapa fitur yang memudahkan pengguna dalam melakukan pengembangan sistem diantaranya adalah sebagai berikut :

1. *Project* berbasis *Gradle Build.*
2. *Refactory* dan pembenahan *bug* yang cepat
3. *Tools* baru bernama *lint* diklaim dapat melakukan *monitoring* kecepatan, kegunaan, serta kompetibelitas aplikasi dengan cepat
4. Mendukung *Proguard and App Signing* untuk keamanan.
5. Memiliki *GUI* aplikasi android lebih mudah.
6. Didukung oleh *Google Cloud Platform* untuk setiap aplikasi yang dikembangkan (Andi Juansyah, 2015)



Gambar 2.6 Workspace Android Studio
(sumber : forum-xda-developers.com)

2.12. *Unified Model Langague*

Unified Modeling Langague (UML) adalah sebuah bahasa yang berdasarkan grafik atau gambar untuk memvisualisasikan, menspesifikasikan, membangun dan pendokumentasian dari sebuah sistem pengembangan perengkat lunak berbasis *Object Oriented Programming*, UML sendiri juga memberikan standar penulisan sebuah sistem *blue print*, yang meliputi konsep bisnis proses, penulisan kelas-kelas dalam bahasa program yang spesifik, skema *Database*, dan komponen-komponen yang diperlukan dalam sistem *software* (Suendri,2018). *Unified Modeling Langague* (UML) adalah keluarga dari notasi grafis yang didukung oleh meta-model tunggal, dan membantu penelitian pada sistem pada *software*, khususnya yang dibangun dengan konsep pemrograman berorientasi objek atau lebih dikenal dengan OOP (Irawan & Herviana, 2019).

UML sekarang distandarisai oleh OMG (*Object Management Group*) dan segala perubahan serta revisi dari spesifikasi UML menjadi tanggung jawab OMG. Versi UML terakhir yang dikeluarkan pada Desember 2017 lalu dari OMG ialah UML 2.5.1. Dalam UML dikenal dua macam diagram utama yang menjadi model pembuatan UML yaitu *structure diagram* dan *behavior diagram* (Ibnu Akil, 2018). Model juga dapat dikatakan cara penyajian sudut pandang terhadap struktur atau fungsi sistem terkait.

Structure diagram menunjukan struktur statis dari sistem dan bagian dari abstraksi serta level implementasi yang berbeda dan bagaimana bagian-bagian tersebut saling berelasi satu sama lain. *Behavior diagram* menunjukkan tingkah laku dinamis dan objek-objek dalam sistem yang mana bisa dijelaskan sebagai sederet perubahan-perubahan dalam sistem sepanjang waktu (Ibnu Akil, 2018).

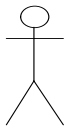
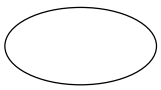
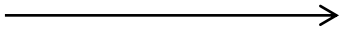
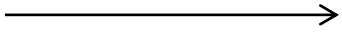
Berikut adalah diagram yang sering digunakan pengembang atau praktisi IT dalam menggunakan UML yang merupakan bagian dari dua model utama diagram UML , yaitu sebagai berikut :

1. *Use Case Diagram*

Sebuah *Use Case* adalah sebuah unit eksternal dari sistem (berupa antar muka) yang akan menerima perintah dari aktor berupa sebuah *event*. *Use case* ini terkait dengan implementasi didalamnya yang berupa urutan-urutan penyampaian

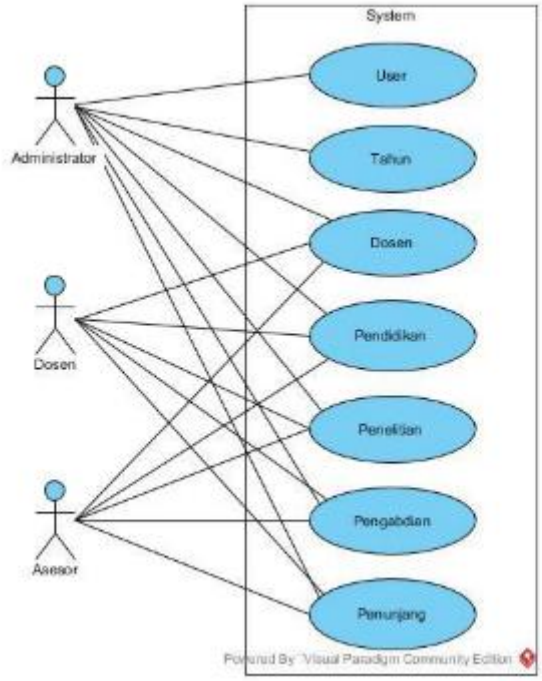
pesan antar objek-objek yang berkaitan (Ibnu Akil, 2018). *Use Case* juga dapat diartikan gambaran grafik (*Graphical*) dari beberapa aktor dan interkasi anatar aktor yang mempresentasikan sebuah sistem, *use case* menggambarkan siapa saja aktor atau pengguna yang melakukan prosedur dalam sistem serta fungsi-fungsi proses apa yang terlibat dalam transformasi pada sistem tersebut (Samsudin, 2019). *Use Case* diagram merupakan bagian dari *behavior diagram*. Berikut adalah tabel simbol-simbol yang digunakan dalam pembuatan diagram *Use Case* :

Tabel 2.2 Simbol Dalam Diagram Usecase
(Ade Handini, 2016)

Simbol	**Deskripsi**
	*Aktor*, merupakan orang atau proses yang berinteraksi dengan sistem dan berada diluar lingkungan sistem informasi.
	*Usecase*, merupakan fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan
	Asosiasi, komunikasi antar aktor dan *usecase* yang berpartisipasi pada *usecase.*
 << *extends* >>	*Extended*, relasi *usecase* tambahan kesebuah *usecase* dimana *usecase* tersebut dapat berdiri sendiri.
 << *include* >>	*Include*, relasi *usecase* dimana *usecase* yang ditambahkan memerlukan ini untuk menjakankan fungsi dan sebagai syarat dijalankannya *usecase* tersebut.

	*Generalisasi*, merupakan spesialisasi (umum-khusus) antara dua buah *use-case* dimana fungsi yang satu adalah fungsi yang lebih umum dari lainnya.

Dalam pembuatan *Use Case* harus dilakukan identifikasi kebutuhan seperti aktor-aktor yang berperan dalam sistem. Berikut adalah contoh diagram *Use Case* setelah identifikasi kebutuhan didapatkan :



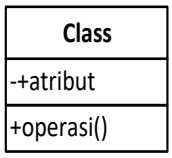
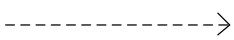
Gambar 2.7 Contoh Diagam Usecase
(Suendri, 2018)

Gambar di atas merupakan contoh dari penerapan simbol-simbol *use case* yang telah di jelaskan tadi digunakan dalam pembuatan diagram *use case*, adapun pada gambar di atas bisa di lihat aktor yang terlibat ada tiga, yaitu *administrator*, dosen dan asesor, yang masing-masing nya terhubung pada *use case*, yamh di-hubungkan melalui garis asosiasi.

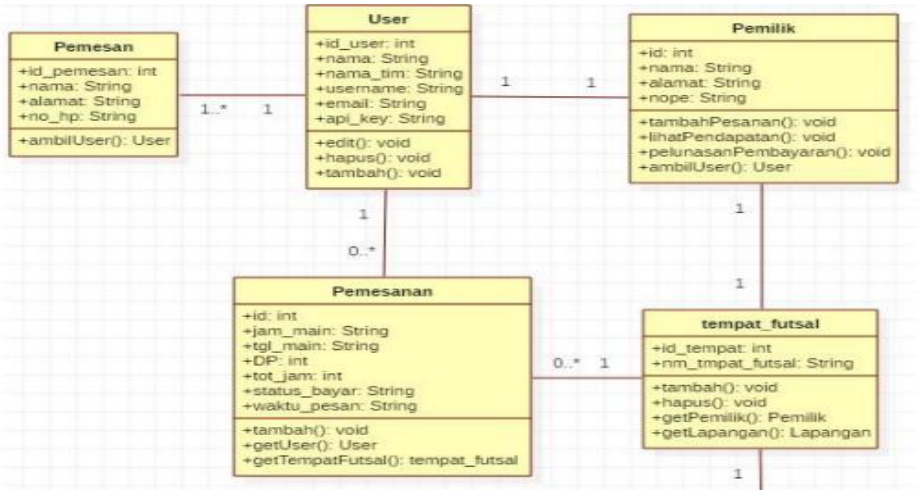
2. *Class Diagram*

Class Diagram merupakan salah satu dari diagram *Unified Model Language* atau yang disingkat dengtan UML, *class diagram* berfungsi untuk menggambarkan struktur statis dan hubungannya. Sebuah kelas diagram adalah diagram yang menunjukkan satu set kelas-kelas, antarmuka-antarmuka, dan hubungan-hubungannya dan *class diagram* merupakan bagian dari model *structure diagram* (Ibnu Akil, 2018). *Class Diagram* menampilkan eksitensi atau keberadaan dari *class-class* dan hubungan dalam desain logika dari sebuah sistem, semua proses yang dilakukan aktor terhadap aplikasi yang akan didefiniskan dengan menggunakan *class diagram* (Samsudin, 2019). Berikut adalah pernjelasan dari simbol yang digunakan dalam *class diagram* yang dimuat dalam bentuk tabel, yaitu adalah sebagai berikut :

Tabel 2.3 Simbol Pada Digram Class
(Ade Handini, 2016)

Simbol	Deskripsi
	*Class*, sebuah struktur sistem yang menjelaskan atribut dan operasi, hal ini sangat berguna nantinya dalam tahap implementasi pembuatan sistem.
	*Interface*, simbol yang mengartikan konsep OOP (*Object Oriented Programming*) dalam sistem.
	*Aggregation*, simbol yang menghubungkan antar kelas dengan makna untuk semua bagian. Jadi simbol ini dipakai jika kelas yang satu adalah semua bagian dari kelas yang lainnya.
	*Depedency*, simbol ini dipakai saat menunjukkan operasi pada suatu

	class yang menggunakan class yang lain.
→	*Generalisasi*, digunakan untuk menghubungkan antar kelas dengan arti umum-khusus.



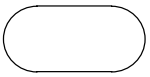
Gambar 2.8 Contoh Class Diagram
(sumber : Ratnasari, Hadi, & Budiarto, 2018)

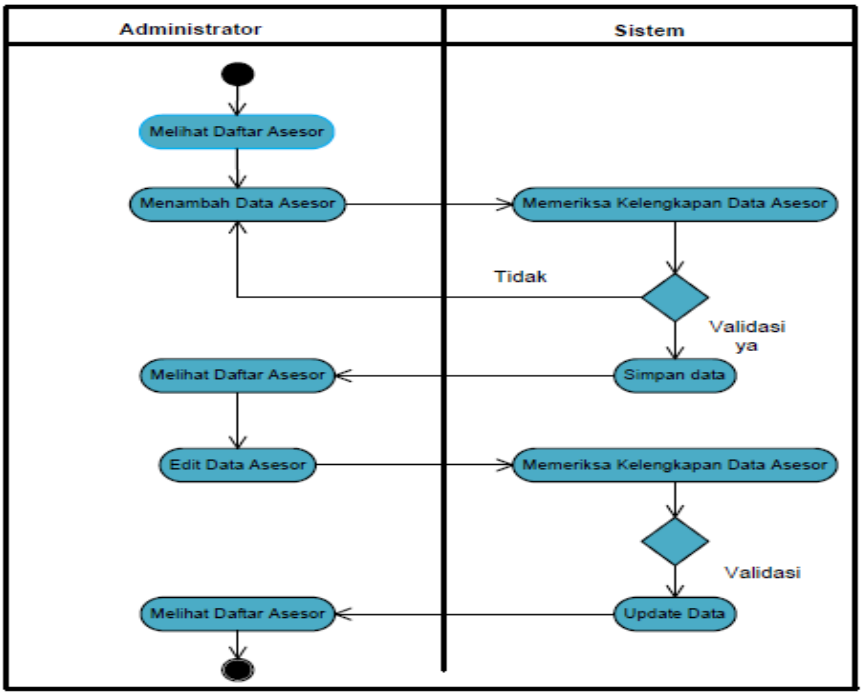
3. *Activity Diagram*

Activity Diagram adalah berfokus pada penggambaran proses logis dari komputasioanl sistem, mirip seperti *flowchart*, bisa jua dikatakan sebuah diagram yang menggambarkan arus kerjadan *businness process*. Kelebihan diagram ini dibandingkan *flowchart* adalah mendukung proses yang berjalan secara parallel (Ibnu Akil, 2018). Adapun simbol-simbol yang digunakan pada *Activity Diagram* adalah sebagai berikut :

Tabel 2.4 Simbol Pada Activity Diagram
(Ade Handani, 2016)

Simbol	**Deskripsi**
●	*Start*, untuk menyatakan awal suatu proses.

	*Stop*, untuk menyatakan akhir suatu proses
	*Decision*, digunakan untuk menyatakan kondisi dari suatu proses.
	*Action*, menyatakan aksi yang dilakukan dalam suatu arsitektur sistem.



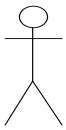
Gambar 2.9 Contoh Activity Diagram
(sumber : Samsudin, 2019)

4. *Sequence Diagram*

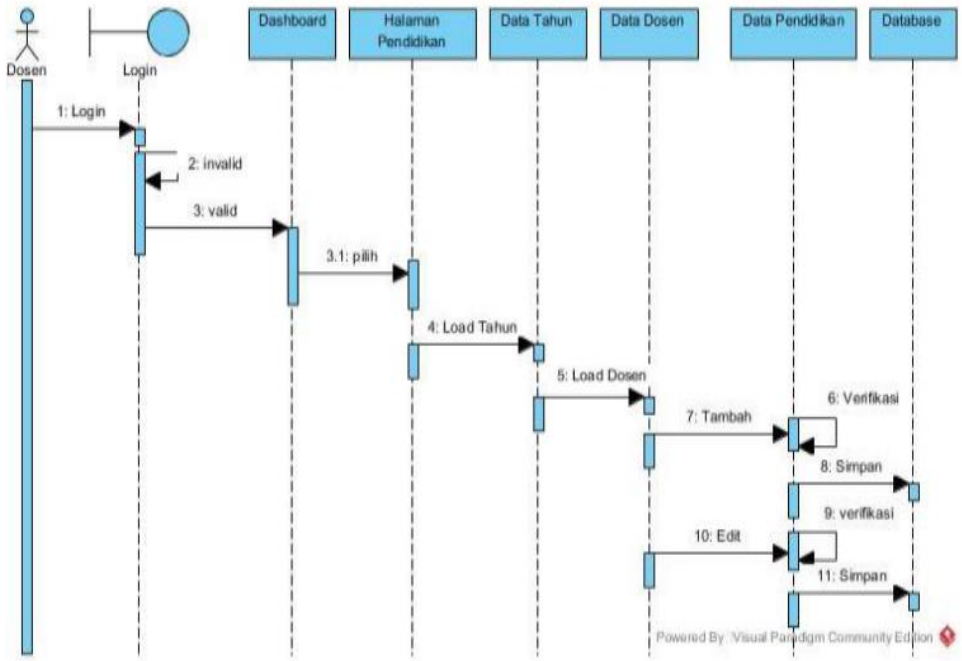
Sequence Diagram menggambarkan tingkah laku dari suatu skenario tunggal, diagram ini menunjukkan objek-objek yang terlihat dalam proses tersebut dan bagaimana urutan penyampaian pesan-pesan antara objek tersebut. Biasanya *sequence diagram* ditampilkan untuk menejelaskan suatu eksekusi *use case* (Ibnu Akil, 2018). *Sequence diagram* adalah gambaran tahap demi tahap, termasuk kronologi (urutan) perubahan secara logis yang seharusnya dilakukan untuk menghasilkan sesuatu sesuai dengan use case diagram (Irawan & Herviana, 2019).

Adapun simbol-simbol yang digunakan dalam membuat *sequence diagram* adalah sebagai berikut

Tabel 2.5 Simbol Pada Sequence Diagram
(*Ade Handini, 2016*)

Simbol	Deskripsi
	*Aktor,* orang atau pengguna yang berinterkasi dengan sistem.
	Lifeline, menginsialisasikan sebuah objek dalam basis waktu notasi untuk *lifeline* berbentuk garis putus-putus *vertical* yang ditarik sebuah objek.
	*Activation,* dinotasikan sebagai sebuah kotak segi empat yang digambar pada sebuah *lifeline*. Menginsialisasikan sebuah objek yang akan melakukan aksi.
	*Message,* digambarkan dengan anak panah *horizontal* antara *Activition Message* mengindikasikan komunikasi antara *object-object*.

Di bawah ini merupakan contoh gambaran dari bagaimana *sequential diagram* itu sendiri bisa di lihat sebagai aktor yang berperan adalah dosen, yang secara *sequential* dan sistematis melakukan proses demi proses, seperti *login* dan dari *login* menuju *dashboard* dan halaman pendidikan, lalu di lanjutkan data tahun, data dosen, dan data pendidikan, yang semuanya berkomunikasi dengan server atau sisi database, dimana dari database ini nantinya dikembalikan lagi dalam bentuk data kepada pengguna dalam hal ini ialah dosen sebagai *aktor*.



Gambar 2.10 Contoh Sequnce Diagram
(sumber : Suendri, 2018)

2.13. Penelitian Terdahulu

Penelitian terdahulu merupakan penelitian-penelitian yang telah dilakukan sebelumnya yang kemudian dijadikan sebagai acuan dalam melakukan penelitian. Tabel dibawah ini merupakan penelitian-penelitian terdahulu yang relevan dengan judul penelitian penulis.

Tabel 2.6 Data Penelitian Terdahulu

No	Peneliti	Judul	Tahun	Hasil Penelitian	Perbedaan Penelitian
1	Ratnasari, Dwi, Hayatulloh Firman Adi, dan Jian Budiarto	Rancang Bangun Aplikasi Penyewaan Lapangan Futsal Berbasis Android	2018	Penelitian inI berhasil membangun aplikasi penyewaan lapangan futsal berbasis android dan dalam implementasinya menggunakan *web service* menggunakan	Pada penelitian ini, penulis mengembangkan aplikasi penyewaan lapangan pada empat cabang olahraga yaitu, futsal, voli, basket, dan badminton, serta tidak

				bahasa pemrograman PHP dan MYSQL sebagai DBMS.	menggunakan *web service* melainkan menggunakan layanan *firebase* sebagai pengganti *web service* tersebut.
2	Swastika, Rino Herningtyas, dan Fata Nidaul Khasanah.	Sistem Informasi Reservasi Lapangan Futsal Corner Menggunakan Metode Waterfall.	2017	Penelitian ini berhasil membangun sebuah sistem informasi reservasi berbasis website menggunakan metode *waterfall* sebagai metode pengembangan sistemnya.	Pada penelitian yang dilakukan penulis, aplikasi di kembangkan menggunakan metode pengembangan sistem *Rapid Aplication Development* atau yang disingkat RAD.
3	Ilham Firman Maulana	Penerapan Firebase Realtime Database Pada Aplikasi E-Tilang Smartphone Berbasis Android	2020	Penelitian ini berhasil membagun aplikasi e-tilang berbasis android dengan melakukan integrasi terhadap realtime firebase guna mengurangi penggunaan kertas.	Pada penelitian penulis, jurnal ini menjadi rujukan bagaimana melakukan integrasi layanan firebase realtime database ini dengan platform sewa sarana olahraga yang akan dikembangkan.

BAB III
METODE PENELITIAN

3.1. Tempat dan Waktu Penelitian

Pada subbab kali ini akan membahas tempat penelitian serta waktu penelitian secara spesifik. Berikut adalah pemaparan dari subbab ini.

3.1.1. Tempat Penelitian

Dalam penelitian ini, penulis mengambil tempat penelitian kepada beberapa pengusaha penyedia jasa layanan sewa lapangan yang meliputi wilayah Kecamatan Percut Sei Tuan Deli Serdang, Kecamatan Medan Perjuangan Kota Medan, Kecamatan Medan Tembung Kota Medan, Kecamatan Medan Timur Kota Medan, Kecamatan Medan Helvetia Kota Medan. Beberapa lapangan yang dimaksud adalah sebagai berikut :

1. Kecamatan Percut Sei Tuan
   a. Laden Badminton Club (Lapangan Badminton)
   b. GOR Sumut (Lapangan Voli dan Basket)
2. Kecamatan Medan Tembung
   a. GOR Samudera (Lapangan Basket)
   b. Ido Futsal (Lapangan Futsal)
3. Kecamatan Medan Perjuangan
   a. Pelita Futsal (Lapangan Futsal)
4. Kecamatan Medan Timur
   a. ONE-A Badmiton (Lapangan Badminton)
5. Kecamatan Medan Helvetia
   a. Helvetia Volley (Lapangan Voli)

3.1.2. Waktu Penelitian

Waktu penelitian berguna untuk mengetahui batas waktu yang telah direncakanan dalam pembuatan sistem tersebut, penelitian ini dimulai dari Mei 2020 – Januari 2020 yang dirancang sebagai berikut :

Tabel 3.1 Waktu dan Jadwal Pelaksanaan Penelitian

Jadwal	Mei 2020				Juni 2020				Juli 2020				Januari 2021			
	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M
Identifikasi Masalah																
Pengumpulan Data																
Pengajuan Pro-posal																
Seminar Pro-posal																
Analisis Sistem																
Perancangan Sistem																
Pembuatan *Coding*																
Testing																

Adapun tahap penelitian yang akan dibutuhkan adalah sebagai berikut :

1. Identifikasi Masalah

   Dalam hal ini penulis melakukan observasi terlebih dahulu terhadap permasalahan yang ada ditengah masyarakat sekitar, atau isu yang bisa diangkat menjadi tema penelitian yang bisa diselesaikan dengan kemajuan teknologi.

2. Pengumpulan Data

   Agar masalah yang di identifikasi dapat dibutikan secara fakta, maka dilakukan tahap pengumpulan data secara kualitatif yaitu obersevasi, wawancara dengan melakuan riset penelitian ke tempat-tempat penyedia jasa layanan sewa sarana olahraga di kota Medan dan membaca studi pustaka terkait penelitian terdahulu yang memiliki persamaan konteks.

3. Pengajuan Proposal

   Tahap ini merupakan tahap selesai proses bimbingan pada dosen pembimbing I dan pembimbing II, dan proposal skripsi siap untuk diuji.

4. Seminar Proposal

   Seminar proposal diadakan agar melihat kesesuaian penelitian yang diangkat dengan melakukan presentasi judul terkait.

5. Analisis Sistem

   Setelah data terkumpul maka penulis melakukan analisa terhadap data yang didapatkan untuk membuat sebuah usulan sistem yang lebih baik dari sistem yang berjalan sebelumnya.

7. Perancangan Sistem

   Pada tahap ini penulis mulai melakukan perancangan dengan membuat alur sistem melalui diagram model (UML), perancangan *database*, dan juga perancangan *interface.*

8. Pembuatan *Coding*

   Tahap ini merupakan tahap dimana penulis melakukan pembuatan kode-kode program sehingga menjadi suatu aplikasi.

9. *Testing*

   Tahap ini melakukan pengujian terhadap aplikasi yang sudah jadi, tujuan dari testing adalah memastikan aplikasi berjalan sesuai apa yang diharapkan pada perancangan sistem. Dalam tahap *testing* ini penulis menggunakan pengujian *blackbox*, yang merupakan sebuah metode pengujian terhadap sistem tanpa memperhatikan *source code* dan lebih memperhatikan arsitektur fundamental dari sebuah sistem apakah sesuai yang diharapkan atau tidak (Cholifah, Yulianingsih, & Sagita, 2018).

3.2. Kebutuhan Sistem

Adapun kebutuhan sistem yang digunakan dalam penelitian bertujuan untuk perancangan dan pembuatan sistem.

3.2.1. Perangkat Keras

Perangkat keras yang digunakan/dibutuhkan dalam pengembangan sistem pada penelitian ini adalah sebagai berikut :

1. Spefikasi *Personal Computer* :

a. *Processor Intel ® Core™ i5*.
b. RAM 4 GB DDR3 *Memory*
c. SSD SATA 256 GB

2. Spefikasi *Smartphone Android* :
   a. *Processor Snapdragon Qualcomm MSM8917 Snapdragon 425*.
   b. RAM 2 GB
   c. ROM 16 GB

3.2.2. Perangkat Lunak

1. Sistem Operasi *Windows 7*
2. *Java Standart Edition 8.1*
3. *Android Studio IDE*
4. *Firebase Cloud Service*
5. *Android Version Nougat 7.1*
6. *Microsoft Visio 2010*
7. *Adobe Photoshop CC 2017*
8. *Vysor Emulator Android*
9. *SourceTree Version Control*

3.3. Cara Kerja

Cara kerja pada penelitian ini menggunakan metode pengumpulan data yang dilakukan secara kualitatif, metode penelitian kualitiatif adalah metode penelitian yang awal pengembangannya digunakan dalam bidang ilmu sosial guna mempelajari fenomena sosial budaya, metode ini berfokuskan pada tindakan, studi kasus serta etnografi, sumber data ini bisa berupa observasi, wawancara, dan studi pustaka (Ihwan Susila, 2015). Dan untuk metode pengembangan sistem menggunakan *Rapid Application Development* (RAD).

3.3.1. Metode Pengumpulan Data

Adapun metode pengumpulan data yang dilakukan pada penelitian ini dilakukan dengan tiga cara yaitu sebagai berikut :

1. Observasi

   Observasi merupakan pengamatan, observasi dilakukan secara sistemtatis yang dilakukan melalui penglihatan mata terhadap tempat/objek penelitian. Dalam hal ini penulis melakukan observasi pada lima kecamatan di kota Medan seperti yang sudah dipaparkan sebelumnya dengan cara melihat langsung kelapangan terkait tempat-tempat penyedia jasa layanan sewa sarana olahraga pada lima kecamatan tersebut, pengamatan yang dilihat antara lain adalah bentuk lapangan, lokasi yang strategis, fasilitas lapangan, serta pelayanan yang didapatkan, dan juga bagaimana sistem yang ada berjalan.

2. Wawancara

   Wawancara adalah sebuah kegiatan dialog yang dilakukan oleh dua individu, dialog tersebut bersifat tanya jawab. Dalam hal ini penulis selaku pewawancara, dan yang menjadi narasumber ialah para pengelola jasa layanan sewa sarana olahraga serta para *customer* yang biasa memakai jasa layanan tersebut. Pada lapangan futsal penulis melakukan wawancara kepada seorang narasumber bernama Dimas, beliau merupakan pengelola lapangan *Ido Futsal* di Kecamatan Medan Tembung, Kota Medan. Pada lapangan basket penulis melakukan wawancara kepada seorang narasumber bernama Jason yang merupakan pengelola lapangan basket di *Gor Samudera* di Kecamatan Medan Tembung, Kota Medan. Pada lapangan badminton penulis melakukan wawancara kepada seorang narasumber bernama Oktama Restu yang merupakan salah seorang pengelola dilapangan *Laden Club Badminton* dikecamatan Percut Sei Tuan, Deli Serdang. Dan terakhir untuk lapangan voli penulis melakukan wawancara dengan narasumber bernama Alfin selaku *member* di Helvetia Volley, kecamatan Medan Helvetia, Kota Medan. Secara umum isi wawancara yang diajukan oleh penulis hampir sama yaitu untuk mendapatkan informasi teknis bagaimana sistem berjalan saat ini, informasi harga yang ditawarkan dalam penyewaan, fasilitas apa saja yang diberikan, dan jadwal operasional.

3. Studi Pustaka

   Studi Pustaka dilakukan dengan mempelajari banyak kajian/penelitian terdahulu, baik itu berupa jurnal, skripsi, dan sebagainya. Serta juga dengan mempelajari buku-buku terkait permasalahan yang ingin di-tuntaskan dengan teknologi yang ingin diangkat. Adapun penelitian terdahulu yang dimaksud seperti Jurnal Ratnasari, Dwi, Hayatulloh Firman Hadi, and Jian Budiarto. "Rancang Bangun Aplikasi Penyewaan Lapangan Futsal Berbasis Android." *JUTI: Jurnal Ilmiah Teknologi Informasi* Volume 16 No.2, 2 Juli 2018. Jurnal Swastika, Rino Herningtyas and Fata Nidaul Khasanah. "Sistem Informasi Reservasi Lapangan Futsal Pada Futsal Corner Menggunakan Metode Waterfall." JURNAL MAHASISWA BINA INSANI, Vol.1, No.2, Februari 2017, 251 – 266 ISSN: 2528-6919 (Online) 251 1(2):251–66.

3.3.2. Jenis Data

Adapun data yang didapatkan pada pengumpulan yang telah dilakukan dibagi menjadi dua jenis data yaitu sebagai berikut :

1. Data Primer

   Data primer merupakan data yang dikumpulkan perorangan ataupun melalui ketempat penelitian tersebut, dengan melakukan wawancara atau observasi, pada penelitian ini penulis melakukan observasi dan wawancara kepada beberapa penyedia jasa layanan sewa sarana olahraga di empat jenis olahraga yang berbeda untuk memperoleh data yang dibutuhkan. Adapun data yang didapatkan berupa data harga lapangan, alur sistem berjalan saat ini, berapa banyak aktor yang terlibat dalam sistem, fasilitas apa saja yang didapatkan pelanggan, jadwal operasional lapangan.

2. Data Sekunder

   Data sekunder merupakan data yang dikumpukan melalui penelitian-penelitian terdahulu ataupun buku terkait tema penelitian. Data yang didapatkan ini bisa menjadi landasan penulis dalam membuat beberapa

pertanyaan saat wawancara atau objek apa saja yang akan diamati dalam melakukan observasi. Adapun penelitian terdahulu yang dimaksud seperti Jurnal Ratnasari, Dwi, Hayatulloh Firman Hadi, and Jian Budiarto. "Rancang Bangun Aplikasi Penyewaan Lapangan Futsal Berbasis Android." *JUTI: Jurnal Ilmiah Teknologi Informasi* Volume 16 No.2, 2 Juli 2018. Jurnal Swastika, Rino Herningtyas and Fata Nidaul Khasanah. "Sistem Informasi Reservasi Lapangan Futsal Pada Futsal Corner Menggunakan Metode Waterfall." JURNAL MAHASISWA BINA INSANI, Vol.1, No.2, Februari 2017, 251 – 266 ISSN: 2528-6919 (Online) 251 1(2):251–66.

3.3.3. Metode Pengembangan Sistem

Metode pengembangan sistem yang digunakan pada penelitian ini adalah *Rapid Application Development* (RAD), sebuah model proses pengembangan perangkat lunak yang menekan pada pengembangan daur hidup sistem yang singkat. RAD merupaka model gabungan dari beberapa teknik terstruktur yaitu *Prototyping* dan *Joint Application* untuk melakukan pengembagan sistem yang cepat. Singkatnya RAD ini merupakan model yang menakankan pada kecepatan pengembangan (Kosasi & Eka Yuliani, 2015). Tahapan RAD terdiri dari tiga tahap dimana tahap-tahap ini tersturktur dan saling bergantung, tahapannya adalah sebagai berikut :

1. *Requirements Planning* (Perencanaan Persyaratan)

   Tahap ini merupakan tahap dimana pengguna dan analisis bertemu untuk mengidetifikasi tujuan dari aplikasi atau sistem, dan pada tahap ini lebih berorientasi pada pemecahan masalah bisnis. Ditahap ini penulis melakukan beberapa hal berikut untuk melakukan *Requirements Planning*.

   a. Pengumpulan data dan syarat-syarat informasi yang akan digunakan untuk tahap berikutnya, data tersebut ialah berupa lokasi-lokasi penyewaan sarana olahraga di tempat kecamatan yang ada pada Kota Medan dan satu kecamatan di Kabupaten Deli Ser-

dang. Pengumpulan data ini dilakukan dengan observasi ketempat dan wawancara. Adapun data yang didapatkan penulis disini adalah data harga, jumlah fasilitas, jumlah lapangan, jumlah pelanggan dalam satu hari, dan lain sebagainya. Data tersebut didapatkan dari satu tempat lapangan yang mewakili masing-masing cabang olahraga.

b. Identifikasi sistem dilakukan dalam tahap ini dilakukan untuk mengembangkan sistem yang sudah ada. Identifikasi sistem dalam penelitian ini ialah identifikasi sistem berjalan/lama dan identifikasi sistem usulan. Adapun sistem lama/berjalan saat ini menggunakan cara konvensional atau manual dimana pelanggan datang ketempat lapangan untuk melakukan reservasi yang mana hal ini kadang memiliki beberapa kendala, seperti jadwal yang sudah tereservasi, ataupun minimnya informasi pelanggan ketika ingin memakai jasa layanan olahraga ditempat lain. Kemudian pembayaran dilakukan ditempat dan rekapitulasi keuangan yang ambigu juga menjadi masalah bagi pengelola lapangan, hal ini disebakan karena pengelola lapangan lupa mencatat pemasukan ataupun pengeluaran. Kemudian sistem usulan yang ingin diangkat adalah untuk memangkas birokrasi sistem tersebut dengan membuat reservasi secara online, pembayaran via online, informasi yang terdistribusi dengan baik, penjadwalan yang terstrukur dan rekapitulasi keuangan yang akurat.

2. *Design Workshop*

Tahap ini adalah fase desain dan menyempurnakan dengan menggunakan kelompok pendukukng keputusan sistem untuk membantu pengguna setuju pada sistem yang dibangun. Biasanya dilakukan dengan menunjukkan tampilan visual desain dan alur sistem kepada pengguna, dalam penelitian ini penulis membagi tahapan desain menjadi bagian sebagai berikut :

a. Desain Proses

Pada tahap desain proses ini penulis melakukan identifikasi aktor-aktor yang terlibat dalam sistem penyewaan sarana olahraga yang mana data tersebut didapatkan dari dari tahap sebelumnya. Menggunakan *Unified Model Langague* dengan diagram model yang digunakan ialah :

1) Membuat *Use Case*

Dalam diagram ini penulis menggambarkan keterkaitan antara sistem usulan dengan aktor tentang apa saja yang dapat dilakukan oleh aktor selaku *user* dalam alur sistem. Dalam *use case* ini terdapat 2 aktor yang menjadi *end user* yaitu pengelola lapangan dan pelanggan. Pengelola lapangan dapat melakukan registrasi ke sistem, melakukan *log-in,* melakukan pendaftaran lapangan, mengisi saldo voucher pelanggan, mengelola transaksi reservasi, mengelola rekapitulasi keuangan. Sedangkan pada pelanggan, dapat melakukan registrasi kesistem, melakukan *login*, melakukan reservasi sesuai jadwal yang diinginkan, melakukan *top-up* kepada pengelola.

2) Membuat *Class Diagram*

Setelah mengetahui aliran sistem selanjutnya dilakukan pembuatan diagram yang *Class Diagram* untuk mengetahui *class-class* yang nantinya akan dipakai dalam implementasi kode secara *Object Oriented Programming*. Adapun *class diagram* pada sistem ini terdapat 7 *class* yaitu *class* admin, *class* lapangan, *class* penyewa, *class* transaksi, *class* info jadwal, *class* favorit, *class* laporan.

3) Membuat *Activity Diagram*

Dalam diagram ini berisikan aliran sistem usulan atau bisa dikatan sebagai proses bisnis dalam sistem. Terdapat dua model *activity diagram* yang akan di buat yaitu diagram

activity client dan *activity diagram administrator*. Pada *activity client* terdapat diantaranta diagram *activity register dan login, activity* favorit ,*activity* manajemen akun, *activity* reservasi, *activity* histori. Sedangkan pada *administrator* terdapat *activity login dan register*, manajemen keuangan, manajemen lapangan, manajemen saldo.

4) Membuat *Sequence Diagram*

Pada tahap ini penulis menggambarkan interaksi objek yang disusun dalam suatu urutan waktu suatu urutan waktu dan hubungan timbal balik terhadap sistem. Terdapat dua *sequence* diagram, yaitu *sequence* diagram pada aktor *customer*/pelanggan dan pada aktor pengelola lapangan.

b. Desain *Database*

1) Memenentukan potensial objek

Penulis membuat daftar potesial objek dengan cara menemukan objek yang penting berlandaskan *use case*. Dari diagram tersebut terdapat 6 object yaitu, pelanggan, pengelola lapangan, saldo voucher, informasi reservasi, data lapangan, data penjadwalan.

2) Membuat rancangan *Database*

Penulis membuat rancangan *database* dengan membuat dari nama *database* yang diberi nama medan_olahraga. Dalam *database* tersebut terdapat beberapa tabel antara lain ialah pelanggan, pengelola, voucher, inforeservasi, lapangan, penjadwalan.

a. Desain *Interface*

1) Rancangan Struktur Menu

Penulis mulai merancang struktur menu untuk dijadikan landasan pembuatan *Interface*. Adapun struktur menu pada sisi pengelola lapangan adalah, register, login, menu

mengelola lapangan ,menu mengelola jadwal, menu riwayat transaksi, menu *top-up*, menu rekapitulasi keuangan. Sedangkan pada sisi *end user* pelanggan adalah register, login, menu reservasi, menu *top-up*, menu riwayat transaksi, menu informasi lapangan tersedia.

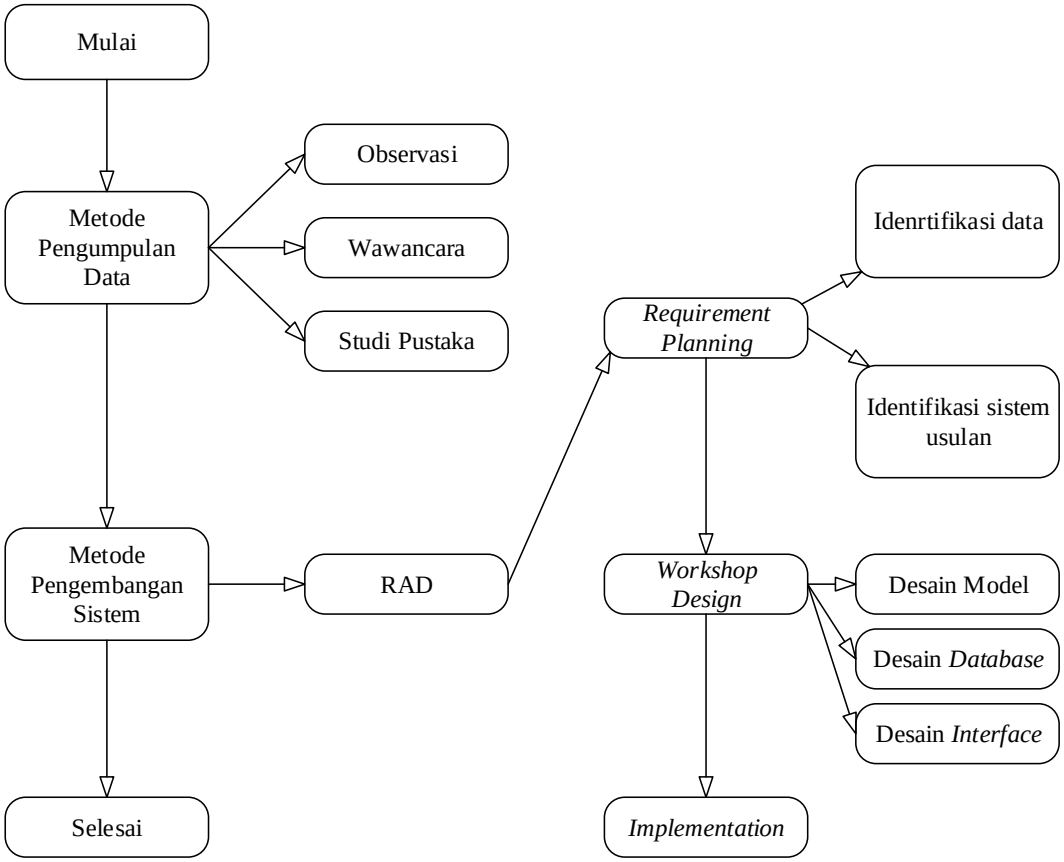
2) Rancangan *Interface*

Penulis merancang *Interface* sistem yang akan dibuat untuk menggambarkan tampilan sistem. Dalam hal ini penulis menggunakan *Microsoft Visio* 2013 untuk melakukan desain antarmuka sistem yang dapat dilihat pada bab berikutnya.

3. *Implementation*

Sistem yang baru dibangun, sistem baru atau parsial di uji dengan memperkenalkan kepada pengguna dalam hal ini menggunakan metode *blackbox* sebagai salah satu metode pengujian sistem dari sistem yang telah dikembangkan (*platform* sewa sarana olahraga berbasis *android*) dan sistem yang lama tidak perlu dijalalankan secara beriringan. Dan pada tahap ini jugalah melakukan pengkodean sistem berdasarkan tahapan pada model pengembangan sistem *Rapid Application Development* dimana tahapan-tahapan tersebut terdiri dari tiga tahap yaitu *Requirements Planning* dimana tahap ini merupakan tahap identifikasi kebutuhan dari masalah yang telah ditemukan dan pengumpulan data agar bisa diadposi kedalam sistem, *Workshop Design* yang merupakan tahap perancangan dan pemodelan dari bentuk sistem yang akan dikembangkan seperti desain model, desain antarmuka, dan desain struktur data dari basis data yang akan digiunakan dan disesuaikan dengan kebutuhan, lalu yang terakhir adalah tahap *implementation* dimana ini merupakan tahap implementasi atau penerapan dari *Firebase* (*firebase authentication. Firebase realtime database,* dan *firebase cloud messaging*).

3.4. Kerangka Berpikir



Gambar 3.1 Kerangka Berpikir

3.4.1 Deskripsi Kerangka Berpikir

Penyelesaian dalam penelitian ini memiliki beberapa tahapan proses, langkah awal dimulai dari metode pengumpulan data. Metode ini merupakan bagaimana penulis memperoleh data dalam penelitian ini yang dimulai dari tahap observasi pada tempat penelitian, dalam hal ini adalah beberapa tempat penyedia jasa layanan sewa sarana olahraga di empat kecamatan kota Medan dan satu kecamatan di kabupaten Deli Serdang. Dalam menunjang hasil yang optimal penulis juga melakukan wawancara kepada para pengelola lapangan dan

melakukan studi pustaka terkait penelitian yang diangkat dengan cara membaca dan mengumpulkan referensi dari berbagai karya ilmiah dan buku-buku terkait penelitian.

Selanjutnya dilakukan tahap metode pengembangan sistem, dimana dalam penelitian ini penulis menggunakan metode pengembangan sistem RAD, sebuah metode pengembangan sistem yang menekankan waktu pembuatan yang efesien dalam pengembangan sistem. RAD sendii memilki tiga tahapan yang harus dilakukan, yaitu *Requirements Planning, Workshop Design,* dan *Implementation.*

Dalam *Requirements Planning* merupakan kegiatan atau tahap dimana penulis mengidentifikasi kebutuhan pada sistem dilakukan dengan cara melakukan analisa data apa yang diperlukan, hal ini dilakukakan setelah sebelumnya penulis melakukan pengumpulan data dan memiliki ide usulan pembuatan sistem dengan melakukan implementasi *firebase* pada sistem yang akan dikembangkan. Lalu dilanjutkan dengan *Workshop Design* dimana pada tahap ini penulis melakukan mulai melakukan perancangan alur sistem atau model sistem, mempresentasikan data yang diperoleh dalam bentuk perancangan database, melakukan perancangan struktur menu utama, dan antarmuka sistem. Kemudian tahap terakhor dari RAD adalah *implemetation* , disinilah implementasi *firebase* diterapkan dan *coding* sistem dilakukan, setelah kedua hal ini dilakukan barulah *unit testing* dilakukan kepada pengguna.

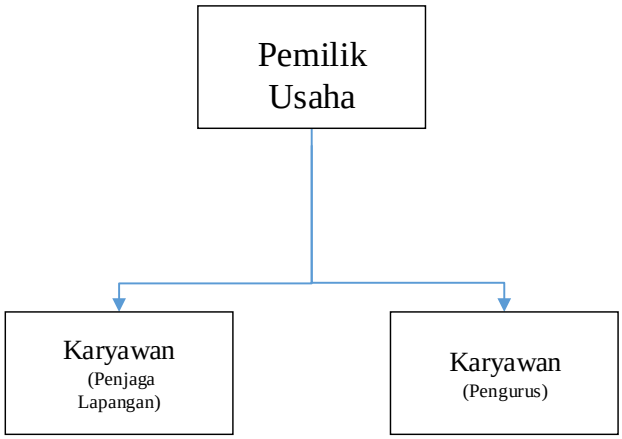
BAB IV
HASIL DAN PEMBAHASAN

4.1. *Requirements Planning*

Seperti yang sudah dibahas pada bab-bab sebelumnya dalam penelitian ini penulis melakukan penelitian pada lima kecamatan dan beberapa lapangan guna mendapatkan data yang dibutuhkan dalam pembuatan sistem atau aplikasi. Dalam melakukan observasi penulis mendapatkan informasi sebagai berikut :

4.1.1. Struktur Organisasi Dalam Bisnis Ini

Pada bisnis sewa-menyewa jasa lapangan ini penulis menyimpulkan bahwasanya struktur organisasi dalam bisnis ini tidaklah kompleks atau rumit. Strukturnya sangat sederhana, yaitu hanya diisi oleh pemilik usaha dan karyawannya, dalam hal ini karyawan bisa lebih dari satu tergantung tempat usaha tersebut. Dibawah ini merupakan gambaran struktur organisasi yang penulis katakan :



Gambar 4.1 *: Struktur Organisasi*

Skema bagan struktur organisasi diatas adalah struktur yang paling umum digunakan dalam bisnis ini, tapi ada juga suatu kasus dimana pemilik usaha tidak memiliki karyawan sama sekali, artinya pemilik usaha menjalankan kinerja dan tugas menjalankan usaha tersebut seorang diri.

4.1.2. *Job Description* dari struktur organisasi

Adapun *job description* daru struktur organisasi pada bisnis sewa-menyewa lapangan dapat dilihat pada tabel berikut ini :

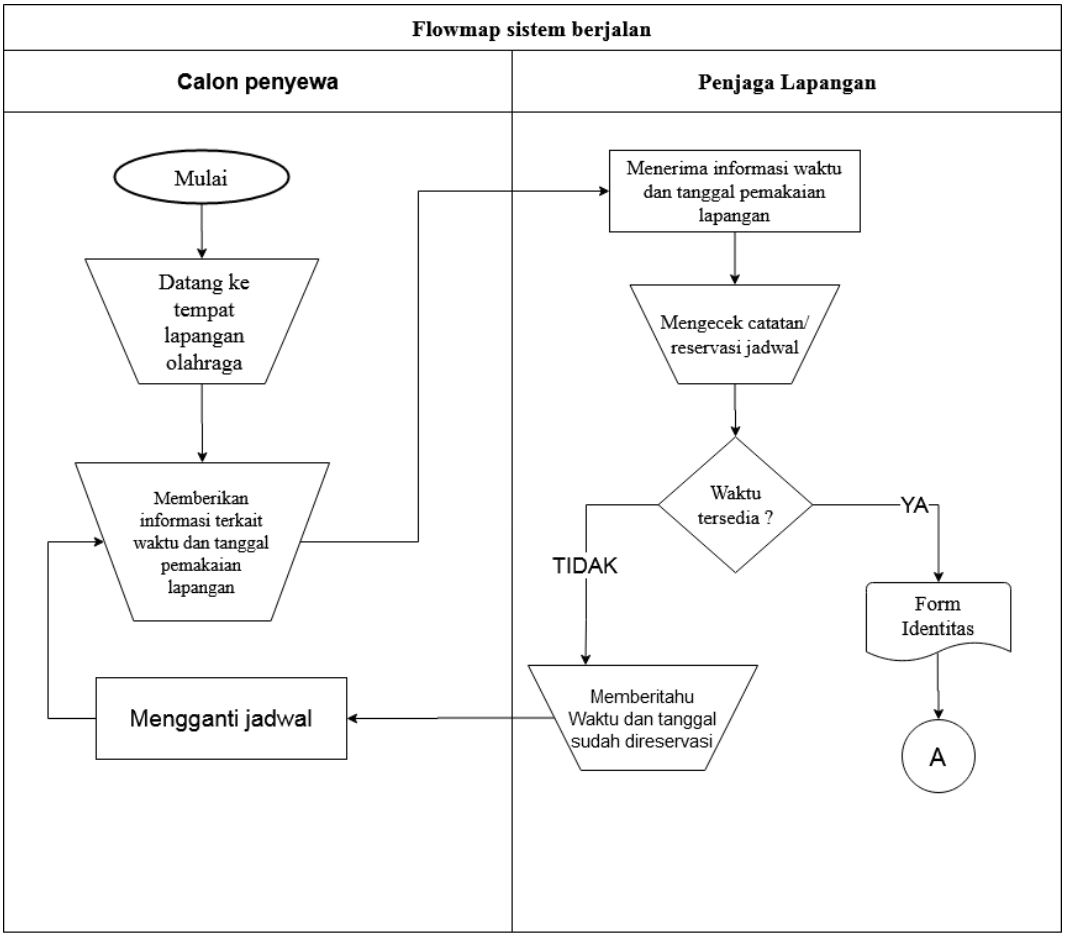
Tabel 4.1 *Job Description*

Jabatan	*Job Description*
1. Pemilik Lapangan	Jika pemilik lapangan memiliki karyawan, pemilik hanya bertugas untuk melihat rekapitulasi keuangan dan berperan dalam menyusun strategi pasar agar usahanya semakin dikenal oleh calon penyewa. Sebaliknya jika sang pemilik lapangan tidak memiliki karyawan kinerjanya akan mencangkup semua proses bisnis.
2. Penjaga Lapangan	Jabatan ini bertugas untuk menjaga lapangan selama waktuu yg ditentukan, bertugas sebagai resepsionis dalam bisnis ini apabila ada pelanggan yang ingin melakukan reservasi dan bermain dilapangan tersebut, tak jarang juga penjaga lapangan merangkap sebagai seorang yang dipercaya mencatat transaksi pemasukan dan pengeluaran dalam bisnis ini.
3. Lainnya	Jabatan ini bertugas secara kondisional, secara hirarki merupakan seorang karyawan.

Sedangkan melalui proses wawancara penulis mendapatkan informasi sebagai penunjang yang berguna dalam mengembangkan aplikasi, yaitu guna pembuatan kamus data yang akan dibahas pada sub-bab *data modeling*.

4.1.3. Analisis Sistem Berjalan

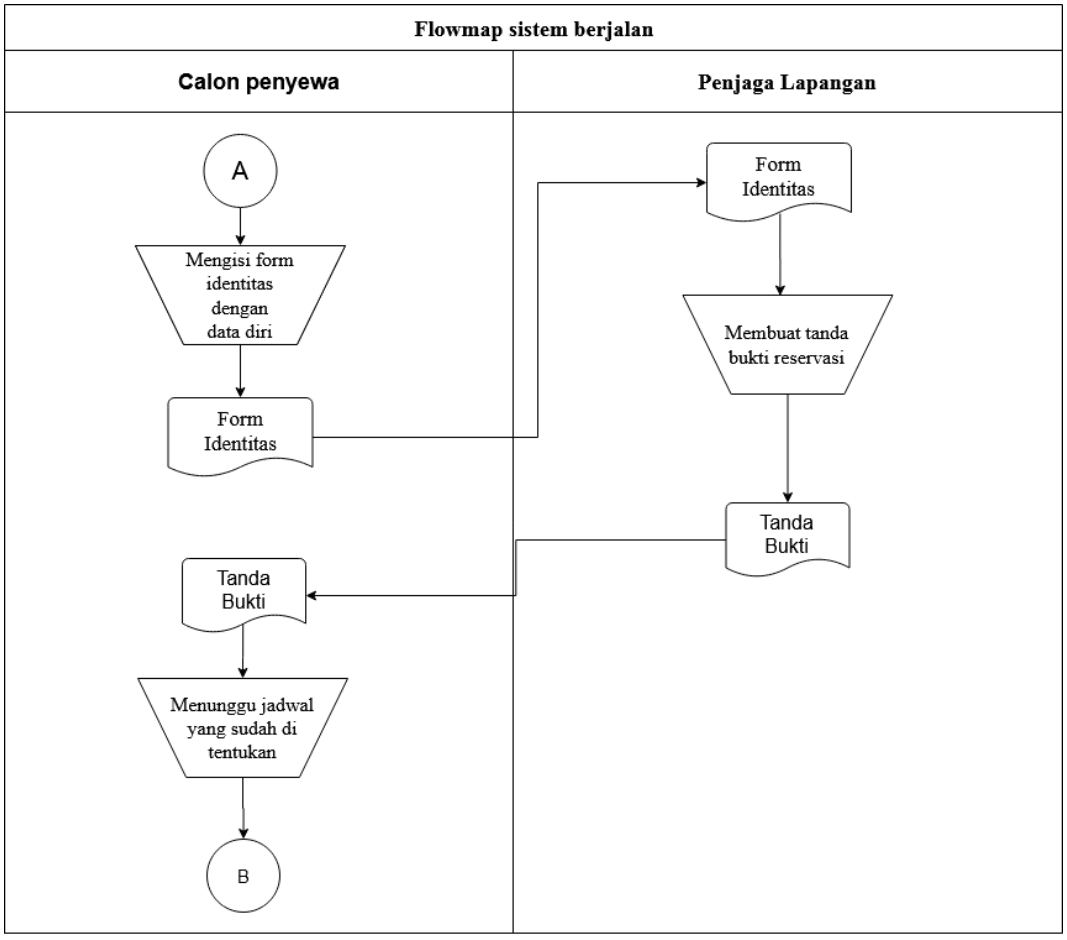
Pada sub-bab ini menjelasakan bagaimana *flow* atau alur dari sistem berjalan terhadap sewa sarana olahraga yang dilakukan secara konvensional, adapun alurnya dijelaskan pada diagram berikut :



Gambar 4.2 *Flowmap* sistem berjalan

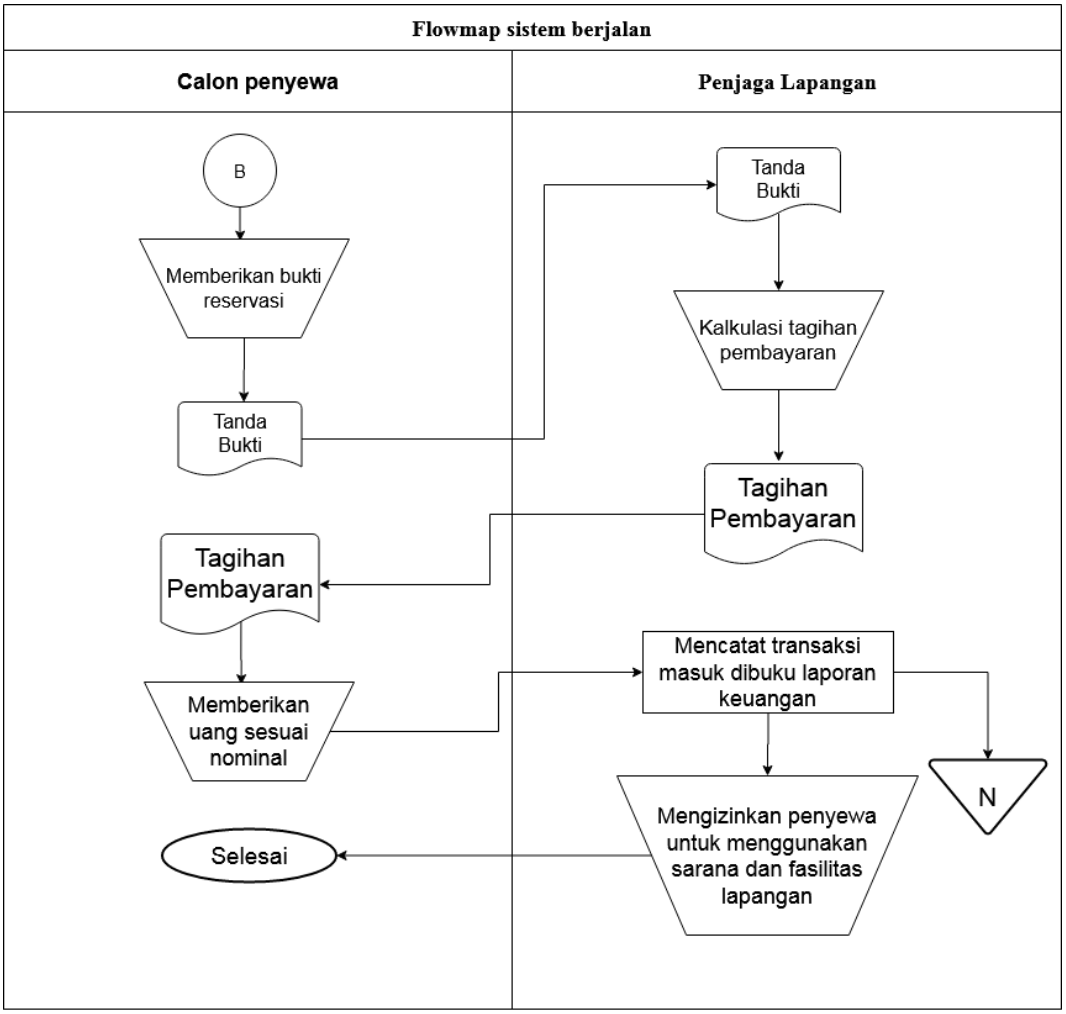
Flowmap diatas tersebut dimulai dari calon penyewa yang datang kelapangan olahraga guna melakukan reservasi lapangan tersebut, selanjutnya calon penyewa akan memberikan informasi kepada penjaga lapangan dan penjaga lapangan akan menyerap informasi tersebut lalu memprosesnya untuk dicek jadwal reservasi jadwal yang di inginkan calon penyewa, apabila waktu yang diminta tidak tersedia maka penjaga lapangan akan memberitahu kepada calon penyewa untuk mengganti jadwal dan calon penyewa akan melakukan proses penggantian

jadwal lalu mengulang memberikan informasi tersebut kepada penjaga lapangan, apabila waktu yang diminta tersedia maka penjaga lapagan akan menyiapkan form identitas.



Gambar 4.3 *Flowmap* sistem berjalan

Form identitas yang diserahkan dari penjaga lapangan akan mengisi *form* identitas dengan data diri calon penywa lalu *form* identitas tersebut kembali diserahkan kepada penjaga lapangan untuk dibuatkan tanda bukti reservasi yang mana tanda bukti reservasi ini akan diserahkan kepada calon penyewa. Selanjutnya calon penyewa akan menunggu jadwal yang sudah direservasi sesuai jadwal yang dipesan sebelumnya.

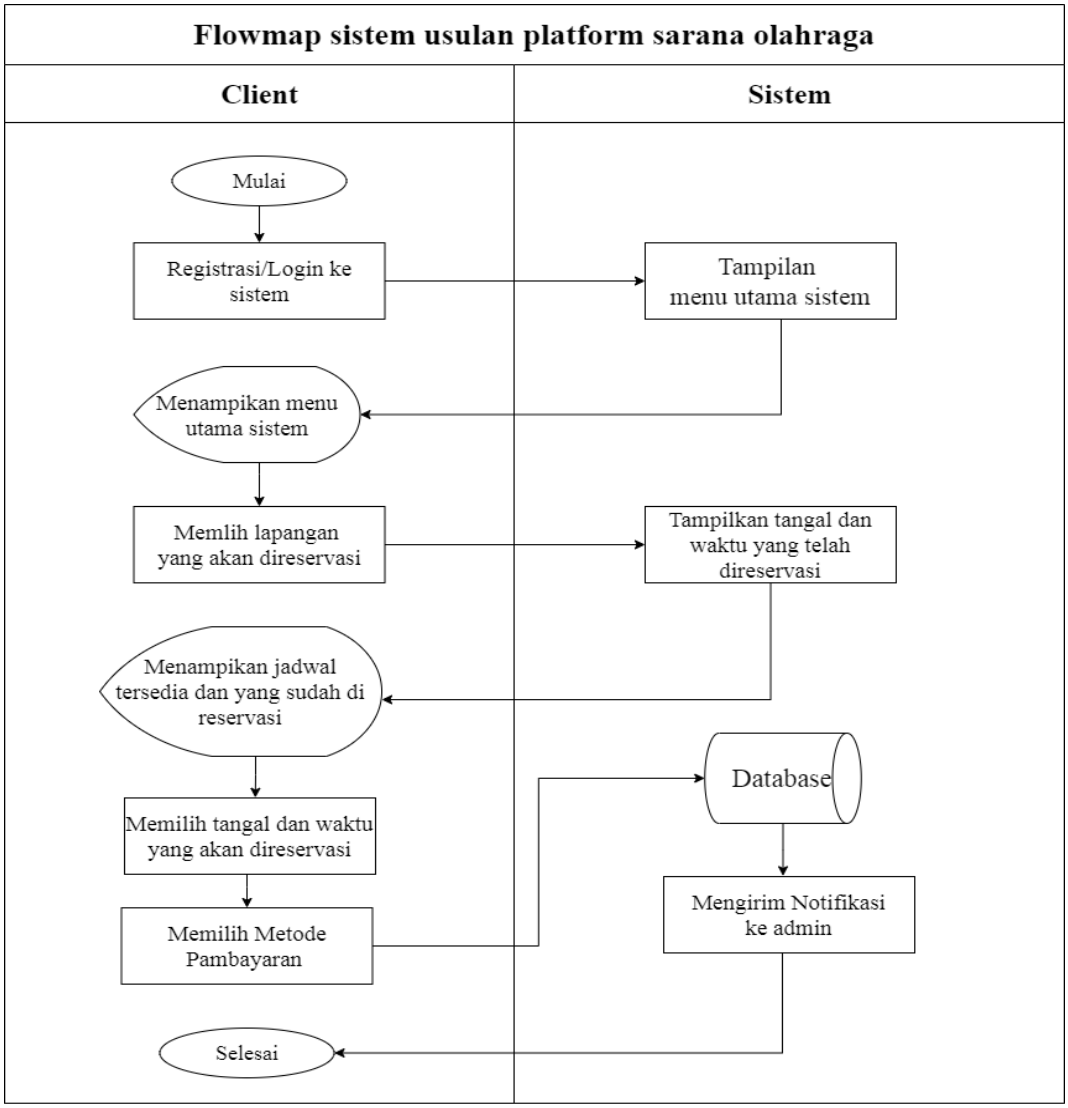


Gambar 4.4 *Flowmap* Sistem Berjalan

Setelah menunggu jadwal yang ditentukan, calon penyewa akan memberikan bukti reservasi kepada penjaga lapangan, kemudia penjaga lapangan akan melakukan kalkulasi tagihan pembayaran dan membuat dokumen tagihan pembayaran yang kemudian diserahkan kepada calon penyewa, lalu calon penyewa memberikan uang/membayar sesuai nominal dan mencatat transaksi pemasukan di laporan keuangan yang nantinya laporan ini akan di arsipkan, kemudian penjaga lapangan mengizinkan calon penyewa untuk memakai fasilitas sarana olahraga yang telah direservasi.

4.1.4. Analisis Sistem Usulan

Sistem usulan dari sistem yang berjalan pada penelitian ini adalah dengan mengembangkan sebuah aplikasi yang dapat berjalan di sistem operasi Android sebagai clientnya, dan memanfaatkan layanan *Firebase* dari *google*. Usulan pada sistem ini akan dibuat dua aplikasi yaitu untuk penyewa dan pemilik lapangan, alasan kenapa harus memakai dua aplikasi agar aplikasi tidak mengalami crash karena saat permintaan data antara admin dan penyewa serta menghindari pere-tasan data penting . Berikut adalah flowchart serta ilustrasi arsitektur sistem yang akan diusulkan :



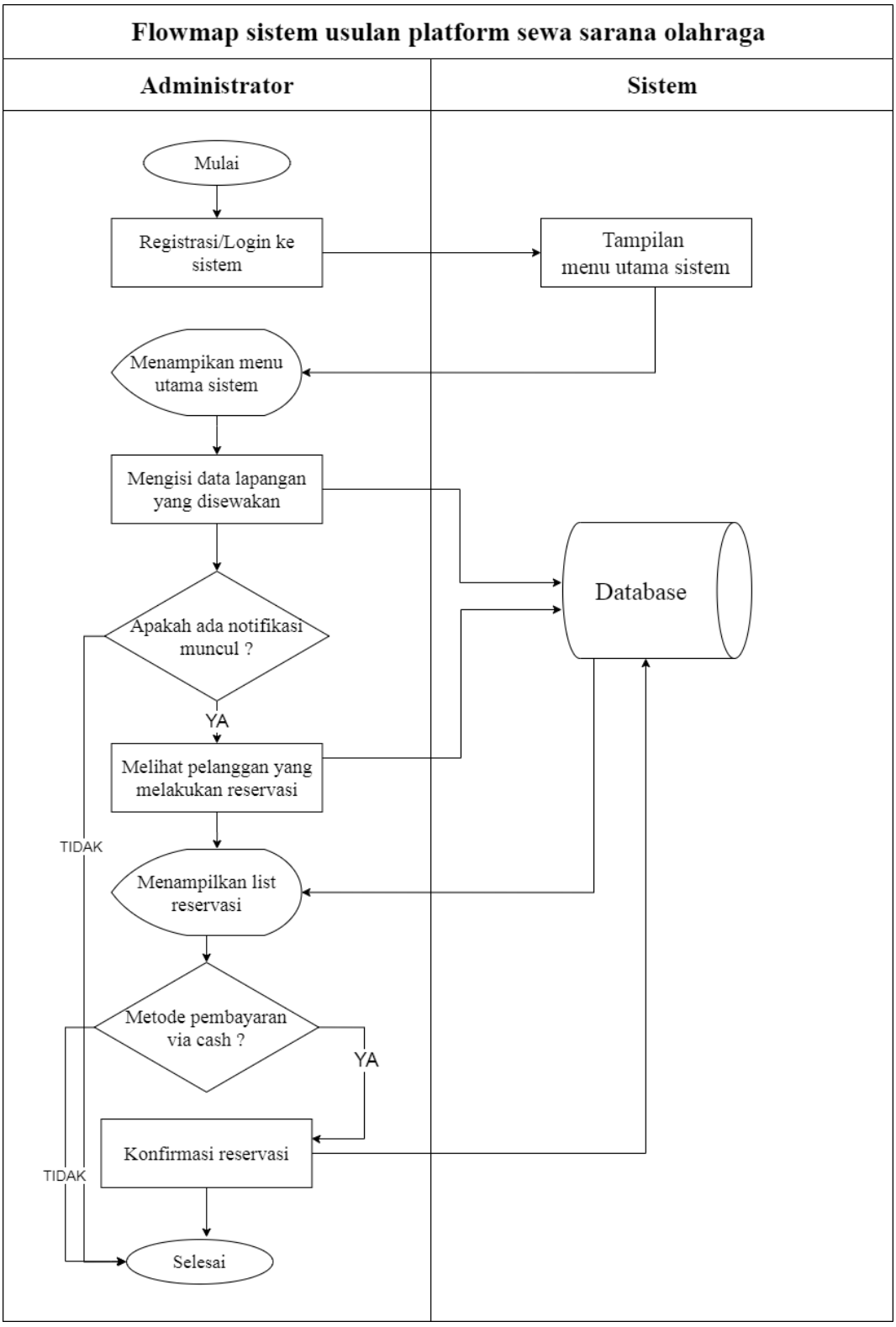
Gambar 4.5 *Flowmap* analisa sistem usulan pada sisi *client*

Pada sistem usulan gambar 4.5 merupakan usulan disisi client, beberapa proses manual dibuat ringkas dari analisa berjalan, adapun penjelasannya adalah sebagai berikut :

1. *Client* memulai membuka aplikasi
2. Lalu melakukan registrasi/*login* menggunakan akun yang telah ada ke sistem.
3. Sistem akan menampilkan menu utama kepada *client*.
4. *Client* akan memilih lapangan yang akan di reservasi.
5. Sistem akan menampilkan ke *interface* pengguna jadwal yang telah direservasi dan tersedia.
6. *Client* memilih tanggal dan waktu yang akan direservasi.
7. *Client* memilih metode pembayaran
8. Lalu sistem akan mengirim notifikasi ke admin
9. Selesai

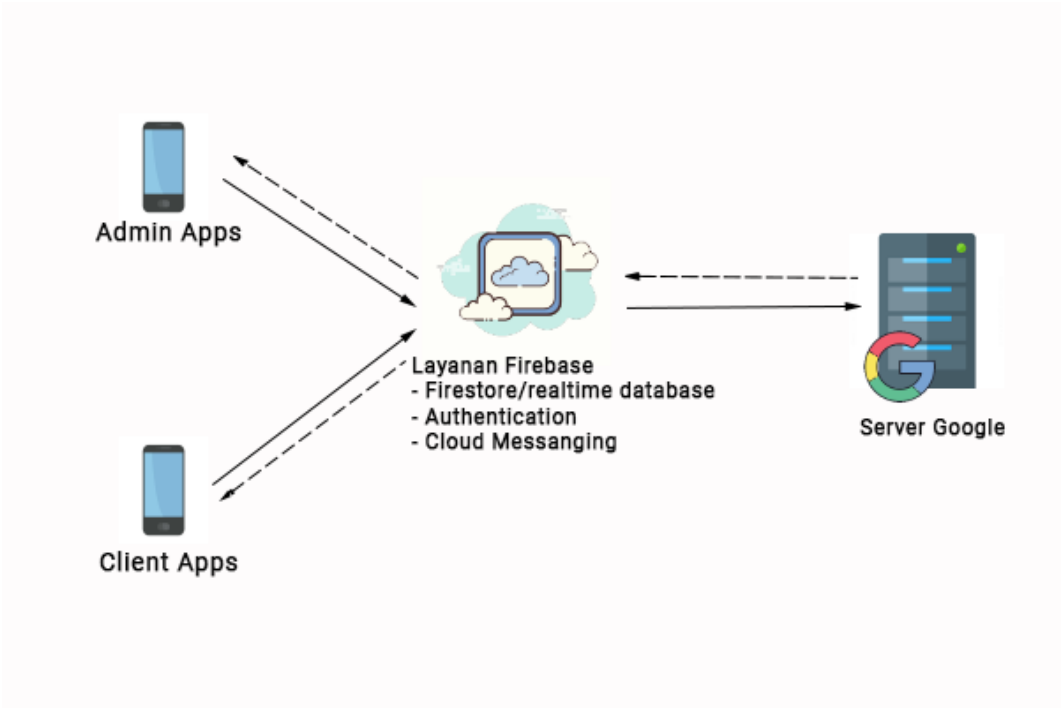
Selanjutnnya gambar berikut ini memaparkan tentang *flow* atau alir dari sistem usulan pada sisi *administrator*. Dimana penjelasannya adalah sebagai berikut :

1. *Administrator* (penjaga lapangan) memulai aplikasi
2. Selanjutnya melakukan registrasi akun/*login* menggunakan akun yang sudah ada.
3. Lalu sistem akan menampilkan menu utama.
4. Mengisi data terkait infomasi yang akan disewakan
5. Kondisi notifikasi muncul merupakan ada reservasi yang masuk dari *client*, apabila kondisi bernilai benar maka melakukan cek pelanggan yang melakukan reservasi jika bernilai salah maka alur selesai.
6. Dari poin nomor lima jika dilanjutkan maka disisi *interface* sistem akan menampilkan informasi list reservasi
7. Kemudian apabila data reservasi tersebut menggunakan metode bayar cash bisa melakukan konfirmasi pemesanan.
8. Selesai.



Gambar 4.6 *Flowmap* analisa sistem usulan pada sisi administrator

Selanjutnya adalah arsitektur dari bagaimana aplikasi akan dibangun, menggunakan dua aplikasi yang terkoneksi dengan layanan *firebase*, penjelasan secara detail bahasan ini akan dibahas disub-bab *implemention*.



Gambar 4.7: Arsitektur sistem usulan

Dari ilustrasi diatas dapat dilihat bahwa sistem ini melakukan *request* data ke layanan *firebase* dalam hal ini layanan yang dipakai ada tiga yaitu *database realtime, authentication,* dan *cloud messanging* dan dengan *google* sebagai infra-sturuktur servernya. Untuk melihat bagaimana fitur aplikasi atau alur dari sistem usulan ini dapat dilihat pada sub-bab berikutnya.

4.2. *Workshop Design*

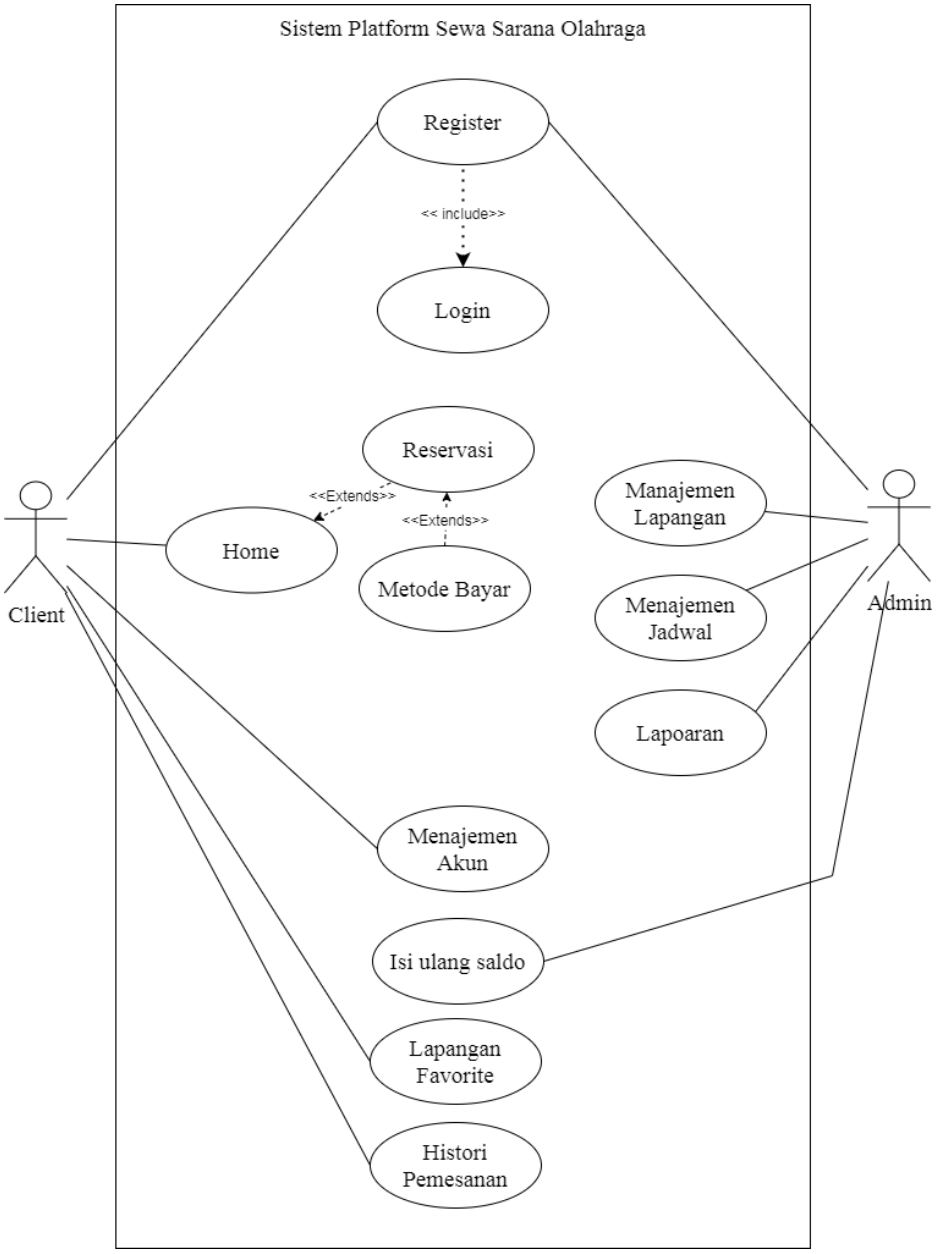
Pada subbab kali ini akan membahas tentang analisa yang lebih detail terkait alur sistem usulan yang telah dibahas pada subbab sebelumnya, subbab ini akan membahas tiga hal yaitu tentang analisa sistem dengan melalui diagram UML, analisa *database*, dan analisa *interface* yang akan dibuat.

4.2.1. *Design Model*

Pada perancangan diagram UML, penulis hanya membuat empat diagram model UML yaitu *use case, activity diagram, sequence diagram,* dan *class diagram.*

4.2.1.1. Use Case Diagram

Secara singkat *usecase* diagram memaparkan tentang hal-hal berupa aktifitas apa saja yang bisa dilakukan oleh aktor selaku *user* dalam sistem yang akan dibangun atau dikembangkan.



Gambar 4.8 : *Usecase Platform* Sewa Sarana Olahraga

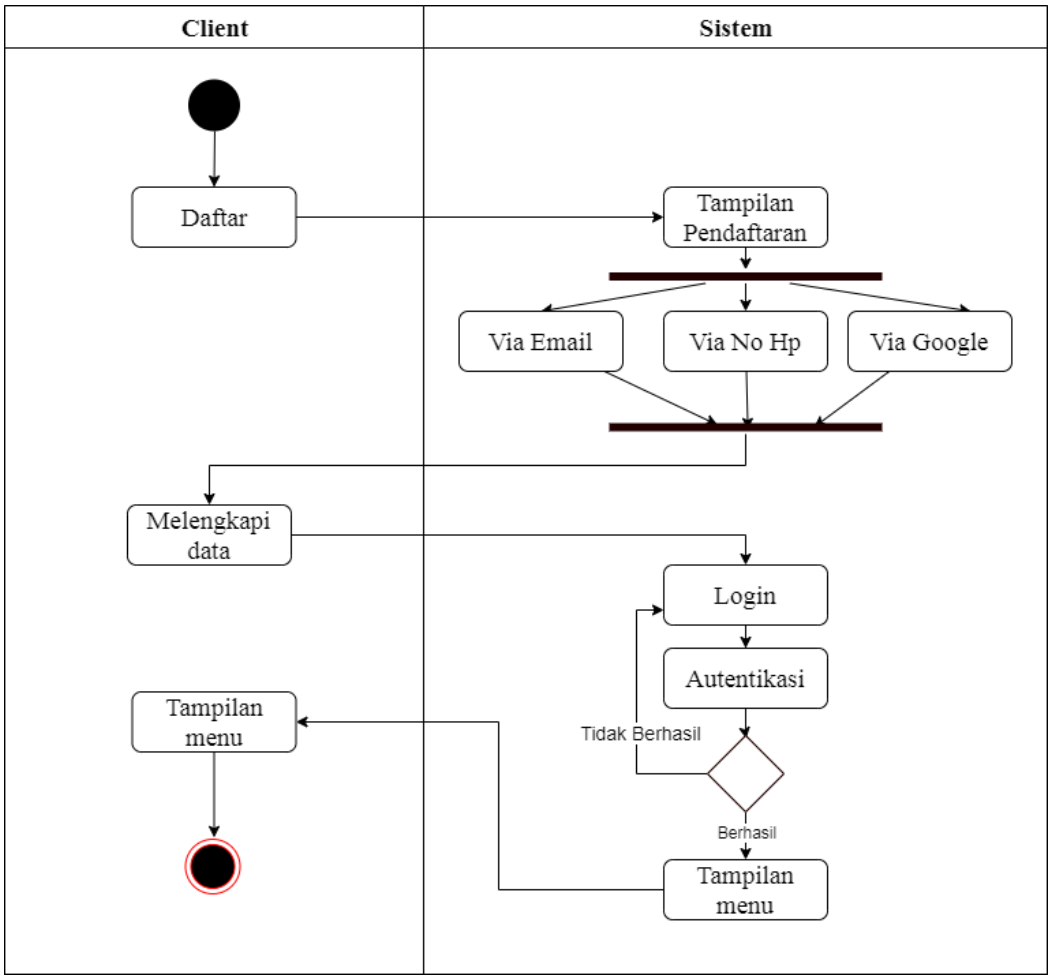
Pada gambar 4.8 dapat di lihat bahwa yang berperan sebagai aktor adalah calon penyewa (*client*) dan *adminsitrator*, dari diagram terlihat kedua aktor dapat melakukan *usecase* registrasi dan *login*. Aktor *client* dapat melakukan reservasi yang *extends* dengan metode pembayaran, melakukan manajemen akun, manajemen favorit lapangan, dan histori pemesanan. Sementara *administrator* dapat melakukan manajemen akun, menajemen jadwal, laporan.

4.2.1.2. Activity Diagram Client

Seperti yang sudah dibahas pada bab dua tentang *activity diagam* secara sederhana *activity diagram* adalah flow tentang aktifitas-aktifitas apa saja yang bisa dilakukan oleh user. *Activity* yang akan dipaparkan pada sub-bab ini yaitu *activity diagram* untuk sisi *client*.

1. *Activity* Diagram Registrasi dan *Login*

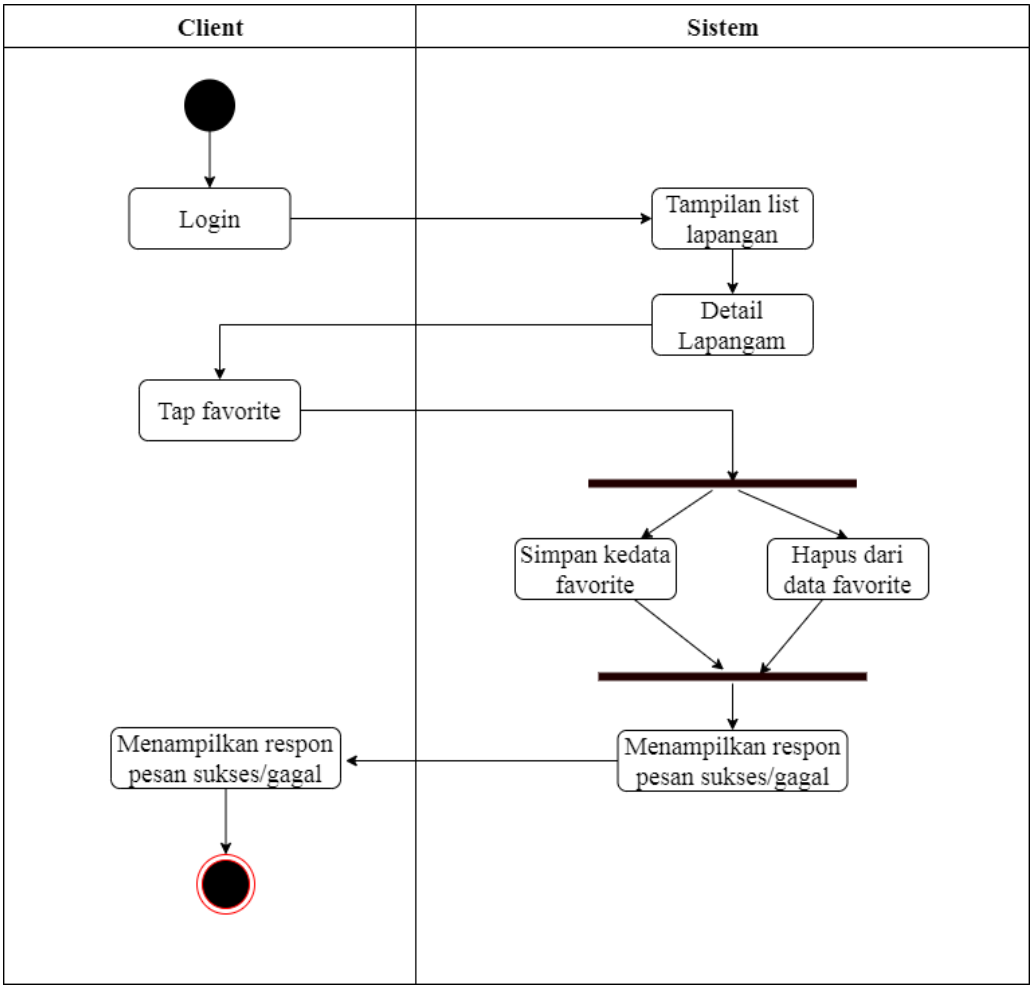
   Registrasi akun disisi *client* menggunakan tiga alternatif pilihan yang bisa atau dapat digunakan oleh calon penyewa selaku *user* untuk menggunakan *platform* sewa sarana olahraga ini. Tiga pilihan registrasi tersebut adalah autentikasi menggunaka email dimana *user* hanya perlu menautkan email mereka secara valid dan melakukan set *password* untuk akun mereka nantinya, autentikasi menggunakan nomor telepon dimana *user* hanya perlu melakukan kegiatan *input* nomor telepon dan menunggu kode *one time password* untuk di lakukan tahap autentikasi oleh *firebase*, autetikasi menggunakan *google account* sendiri *user* hanya perlu menautkan *google account* yang sudah terdaftar di perangkat *user* itu sendiri.. Kemudian login menggunakan alternattif pilihan yang sudah dipilih sebelumnya. Untuk lebih jelasnya dapat dilihat pada gambar beri-kut :



Gambar 4.9 *Activity Diagram* Registrasi di Sisi *Client*

Pada diatas user memulai aktifitas, lalu melakukan pendaftaran, sistem akan menampilkan opsi pendaftaran yang bisa dipilih oleh *user*, yaitu *email*, nomor telepon, atau *google account*, kemudia pengguna akan melengkapi data informasi data diri. Kemudian sistem akan melakukan *login* dan autentikasi, apabila kondisi autentikasi berhasil maka sistem akan menampilkan menu utama aplikasi apabila tidak maka akan mengulangi proses *login* dan autentikasi, hingga proses berhasil dan aktifitas pun selesai.

2. *Activity* Diagram *Favorite (client)*



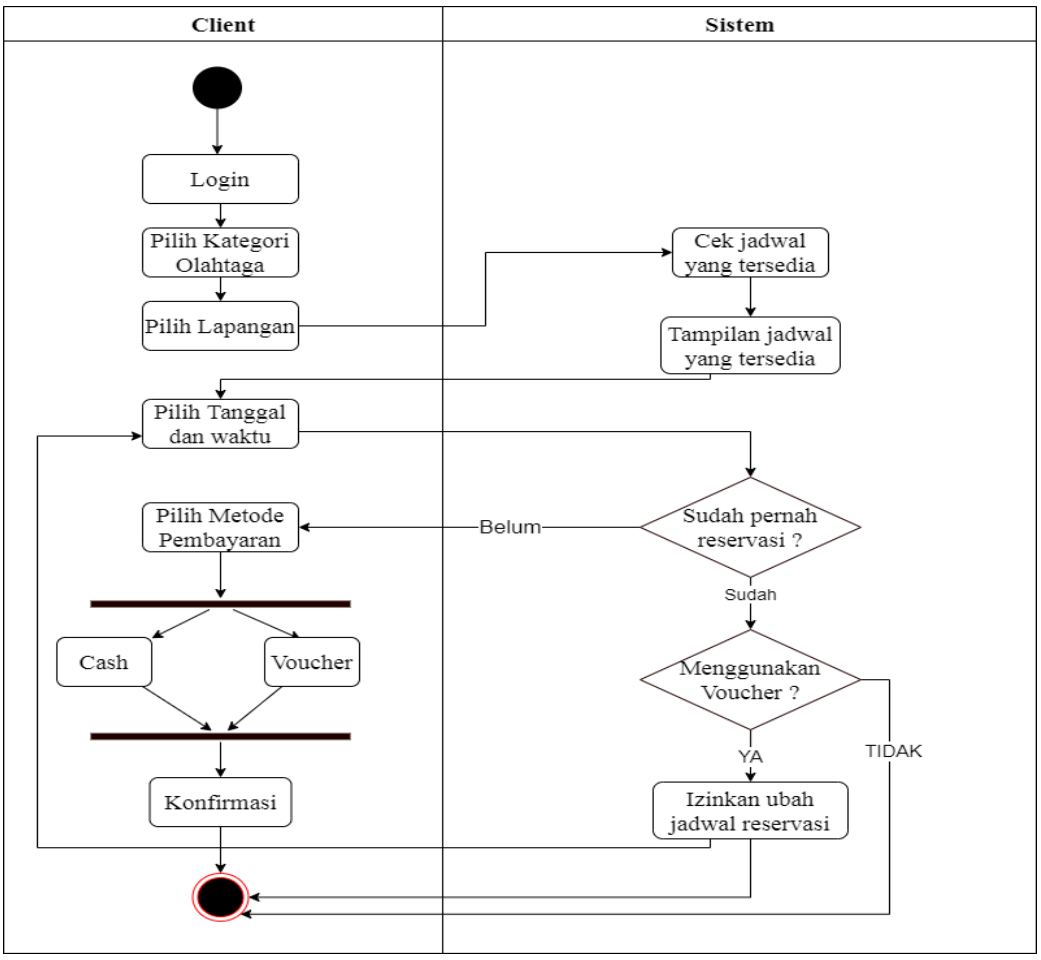
Gambar 4.10 : *Diagram Activity* Favorit

Pada gambar diatas merupakan *flow* dari *activity* diagram *favorite,* dimulai dari user melakukan *login* dan sistem akan menampilkan lapangan yang dipilih oleh *user*, kemudian dilanjutkan dengan user yang melakukan *tap* pada *icon* favorit dari anatarmuka halaman detail lapangan dimana ada *join fork* yang bisa dilakukan oleh pengguna yaitu bisa menyimpan data lapangan tersebut sebagai *list* favorit atau menghapus data lapangan tersebut dari *favorit* dan aktifitas selesai, lalu setelah itu sistem akan menampilkan sebuah respon berupa sebuah pesan sukses

ataupun gagal dari masing-masing aksi yang telah dilakukan user sebelumnya.

3. *Activity* Diagram Reservasi (*Client*)

   Diagram ini merupakan *flow* utama pada penelitian ini karena disisi ini yang menjadikan proses reservasi yang sebelumnya konvensional menjadi terkomputasi dan bersifat online. Berikut adalah *flow* dari reservasi yang sudah dilakukan secara komputasi :



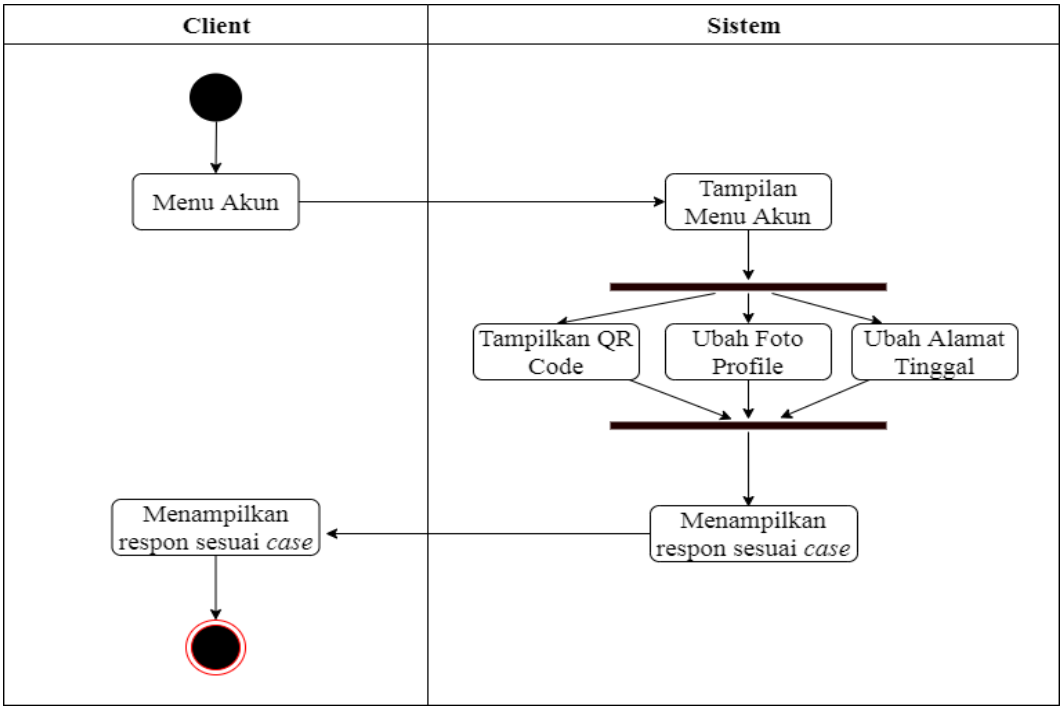
Gambar 4.11 *: Activity Diagram Reservasi*

Flow dimulai dari user memilih kategori olahraga yang disenangi user, selanjutnya user akan memilih dari salah satu daftar lapangan yang tersedia agar bisa direservasi. Kemudian sistem akan melakukan cek daftar jadwal dari lapangan yang telah dipilih oleh user sebelumnya dan men-

ampilkannya kepada user, lalu user akan memilih jadwal yang di inginkan, kemudian user akan melanjutkan kepilihan metode pembayaran akan tetapi sebelum kemetode pembayaran, sistem akan melakukan pengecekan lagi apakah user sudah pernah memesan pada lapangan tersebut, jika sudah pernah dan status pilihan metode pembayaran yang digunakan adalah voucher maka user bisa menggunakan fitur ganti jadwal reservasi Kemudian jika user belum pernah sama sekali melakukan transaksi sebelum hari yang diinginkan maka user bisa memilih metode bayar.

4. *Acitivity* Manajemen Akun

   Pada *activity* ini sebenarnya secara fungsionalitas sangatlah sederhana, seperti manajemen akun pada aplikasi-aplikasi umumnya yaitu *user* dapat melakukan perubahan gambar dan alamat kemudian melakukan *update* data.

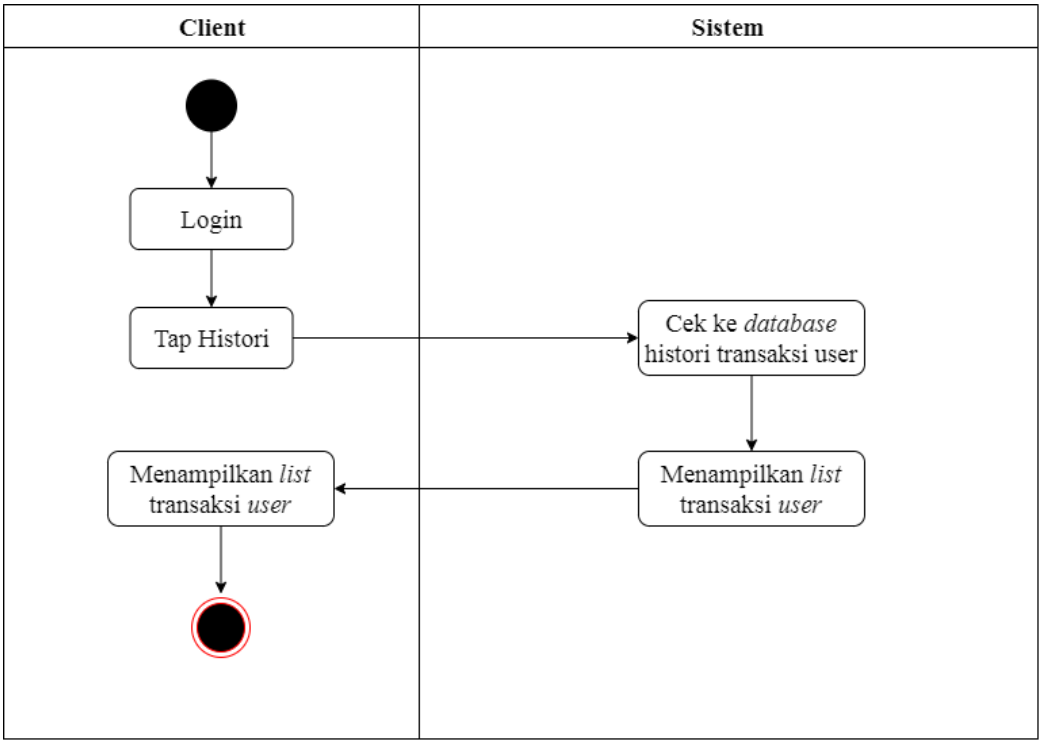


Gambar 4.12 *: Activity Manajemen Akun*

Pada gambar diatas penulis memaparkan lebih detail atas activity manajemen akun ini, yaitu *flow* mengubah gambar, mengubah alamat, dan menunjukkan *qr code*.

5. *Activity Diagram Histori*

   Pada *activity* ini menjelaskan *flow* bagaimana *user* memeriksa proses transaksi reservasi yang telah dilakukan sebelumnya oleh *user*, dimana *flow* dimulai dari *user* yang melakukan *login* dan melakukan *tap button* menu histori, dan di layar belakang sistem akan melakukan sebuah pengecekan kepada *database* dan lalu menampilkan sebuah *list* atas transaksi yang telah dilakukan *user* untuk ditampilkan dalam antarmuka yang bisa dilihat oleh calon penyewa.

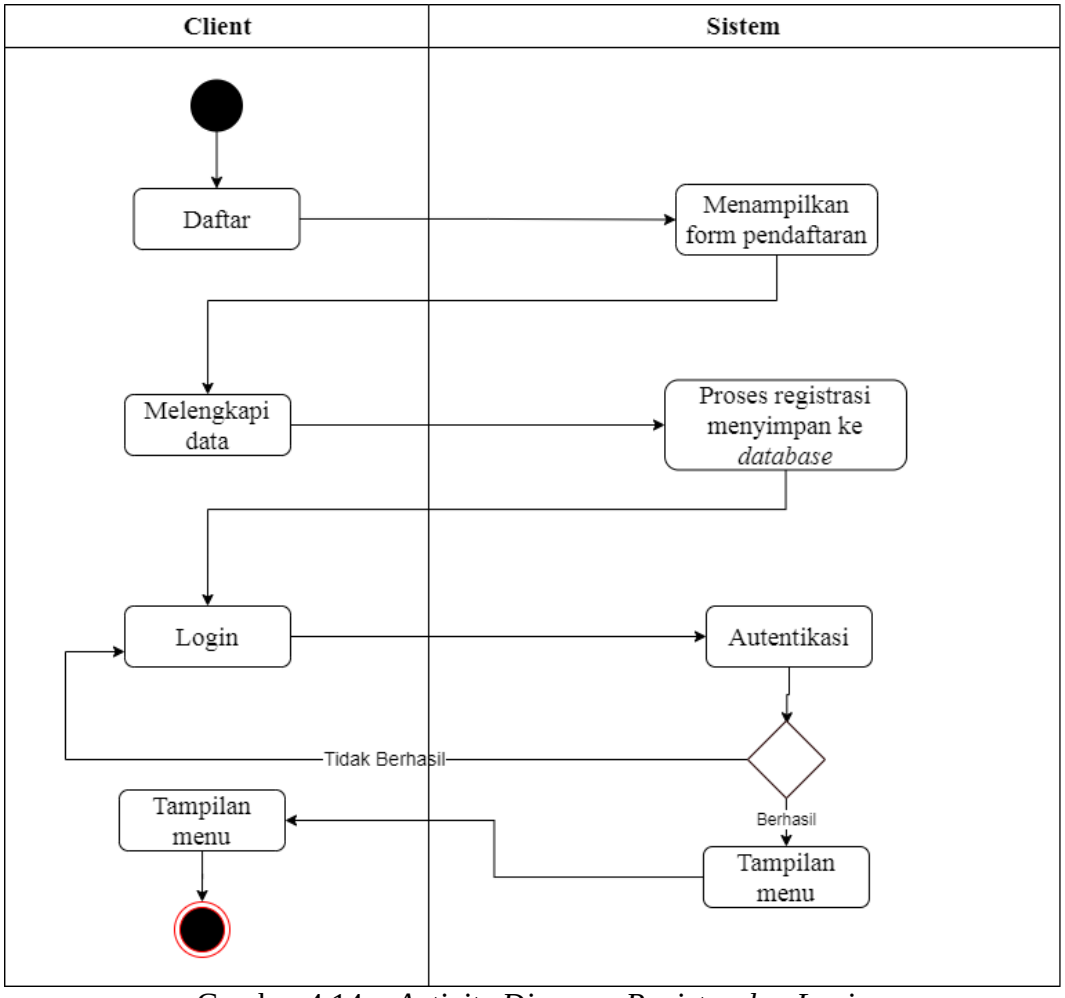


Gambar 4.13 : *Activity Diagram* Histori Pemesanan

4.2.1.3. *Activity Diagram Administrator*

Pada sub-bab ini akan memaparkan *activity diagram* dari sisi administrator atau pemilik jasa lapangan olahraga, yang mama diantaranya adalah *activity login* dan *register*, *activity* manajemen lapangan, *activity* manajemen jadwal, *activity* laporan, dan *activity* pengisian ulang saldo voucher.

1. *Activity* Diagram Register dan *Login (Administrator)*

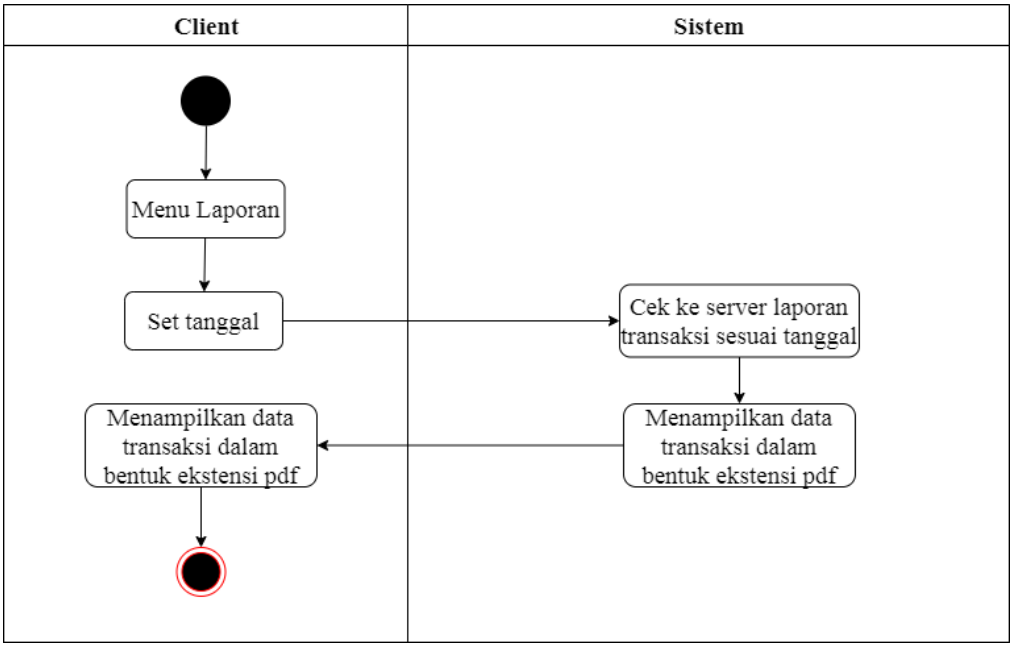


Gambar 4.14 : *Activity Diagram Register dan Login*

Proses daftar dan *login* pada sisi *administrator* di mulai dari pemilik jasa lapangan yang melakukan daftar di aplikasi dan melengkapi seluruh data yang dibutuhkan pada *form* yang di sediakan, lalu selanjutnya sistem akan melakuakan proses registrasi dengan menyimpan data yang telah di isi

oleh user ke dalam *database* dan melakukan *login* lalu di autentikasi oleh sistem dan apabila autentikasi berhasil sistem akan melakukan atau dapat menampilkan menu yang mana nantinya dari sistem akan dilemparkan kembali kepada antarmuka *user* dan apabila gagal *user* melakukan *login* lagi.

2. *Activity* Diagram Laporan Keuangan (*Administrator*)



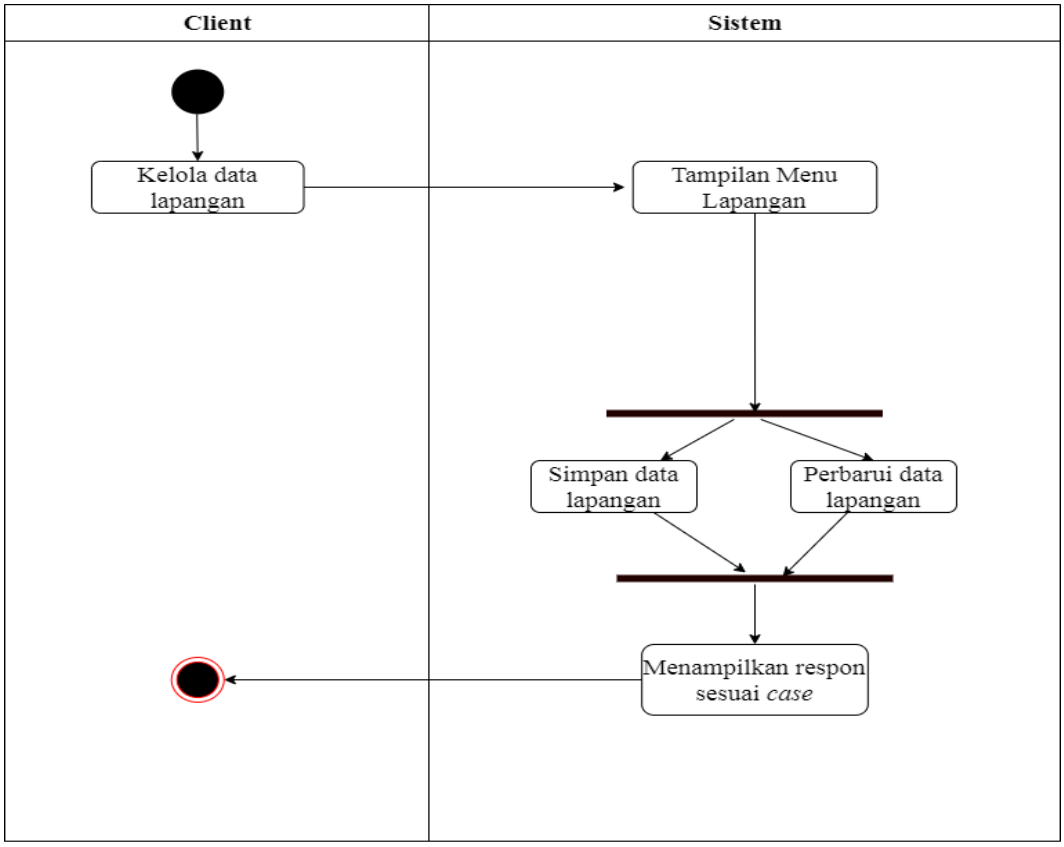
Gambar 4.15 *Diagram Activity* manajemen keuangan

Pada gambar diatas merupakan *flow* dari *activity* diagram aktifitas laporan keuangan yang bisa di akase pengelola lapangan, sederhanya *flow* ini merupakan alur dimana user selaku *admin* (pemilik lapangan) untuk dapat melihat jalannya sebuah bisnis selama ini dalam konteks perolehan transaksi selama ini, dan dari aktifitas ini juga *user* dapat melihat setiap tanggalnya atas transaksi yang sudah dilakukan atau yang pernah terjuadi. Kemudian jika user ingin membuat atau melakukan konversi laporan tersebut dalam bentuk dokumen dengan ekstensi *pdf *user* juga bisa melakukan hal tersebut, sehingga nantinya dokumen dalam bentuk ekstensi atau format *pdf* ini bisa dicetak dan data tersebut berbentuk *hardcopy*.

Caranya user bisa dengan masuk atau melakukan *tap* ke menu laporan lalu melakukan set tanggal awal dan tanggal akhir sesuai kebutuhan yang di inginkan pengelola lapanagan (*administrator*), setelah itu sistem akan melakukan cek ke server sesuai tanggal yang di minta, dan kemudian sistem akan menampilkan hasilnya yang akan ditampilkan pada antarmuka.

3. *Activity* Diagram Manajemen Lapangan (*Administrator*)

   Pada *activity* ini user dapat melakukan hal-hal seperti menambahkan data lapangan, mengubah data lapangan, umumnya ini merupakan proses yang wajib dimiliki pada sebuah sistem berbasis komputer.

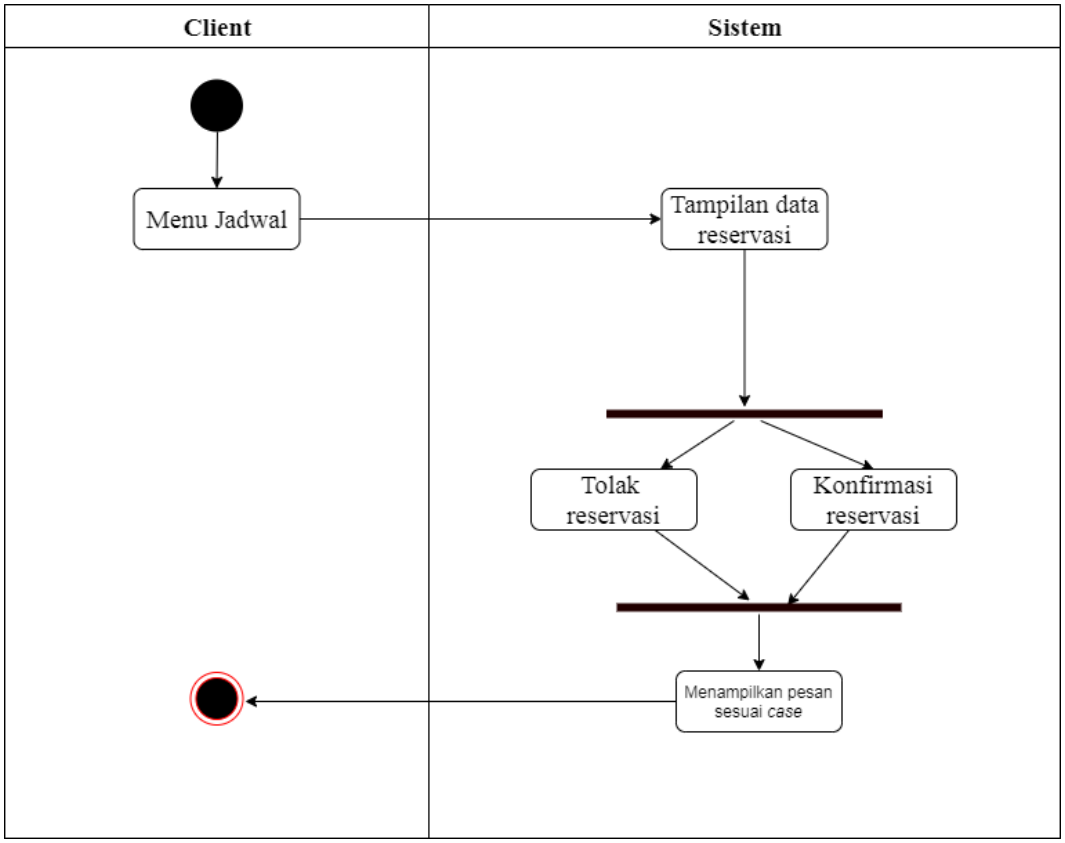


Gambar 4.16 : *Diagram activity* manajemen lapangan

4. *Activity* Diagram Manajemen Jadwal (*Administrator*)

   Pemesanan yang dilakukan dengan metode pembayaran *cash on delivery* dapat dibatalkan oleh pemilik lapangan apabila calon penyewa tidak datang tepat waktu dalam pembayaran, dan untuk hal itu disinilah maksud

dari *flow* manajemen jadwal dibuat, yaitu untuk menangani kasus seperti ini, *flow* dimulai dari *client* kemenud jadwal dan sistem akan menampilkan data reservasi yang masuk lalu ada *join fork* dimana client bisa menolak reservasi atau menkonfirmasi reservasi yang ada pada *list*, untuk lebih jelas silahkan lihat gambar berikut:

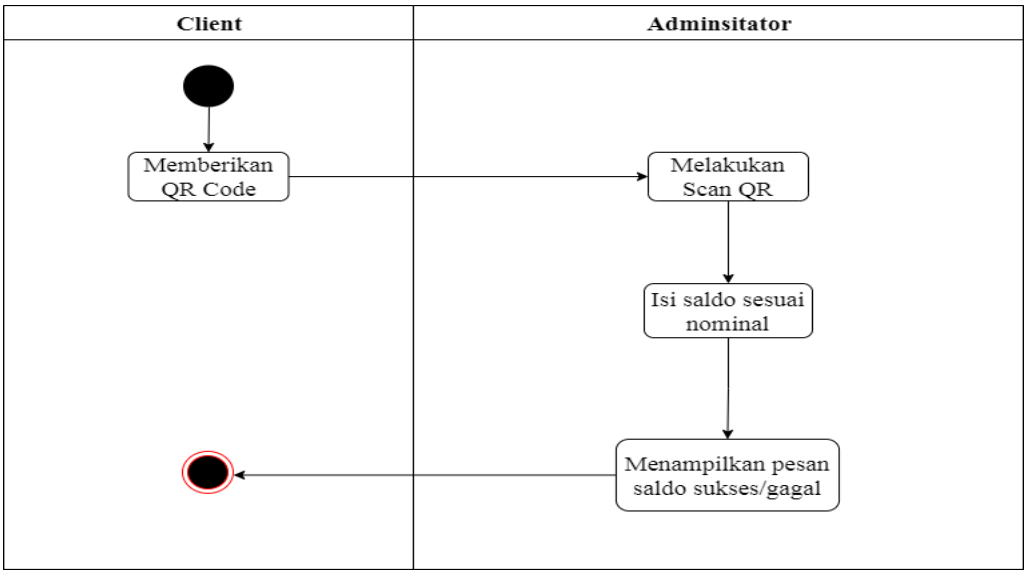


Gambar 4.17 : *Diagram activity* manajemen jadwal

5. *Activity* Diagram Mengisi Saldo Voucer (*Administrator*)

   Flow ini merupakan aktifitas bagaimana mengisi saldo voucher kepada user dalam bentuk digital, proses transaksi yang dilakukan ialah proses *barter* (tukar-menukar) yaitu dengan menukar uang tunai dengan voucher sejumlah nominal yang ada, sehingga voucher ini bisa digunakan saat proses reservasi dilakukan. Proses dimulai dari user menyerahkan ID pengguna yang digenerate dalam bentuk QR code nya kepada admin, ID digunakan sebagai data identitas unik yang dimiliki

setiap user, kemudian admin melakukan scan QR code dan menginput nominal saldo yang akan diisi, proses pengisian saldo selesai. Berikut adalah *activity* diagramnya :



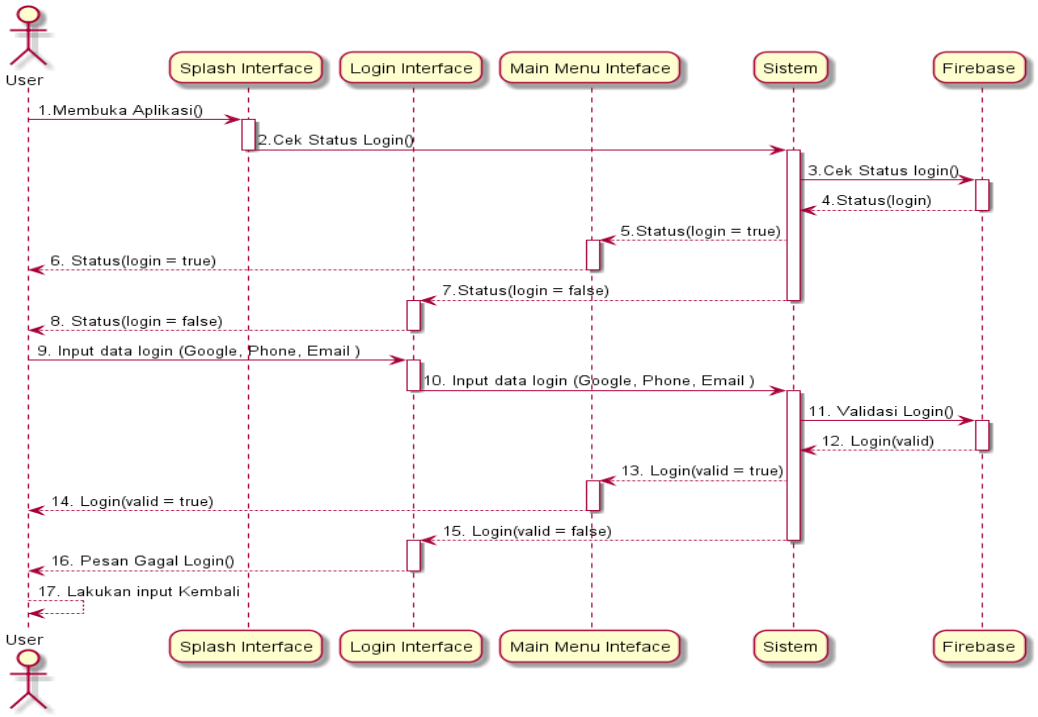
Gambar 4.18 *: Diagram Activity* Saldo

4.2.1.4. Sequential Diagram Client

Berikut adalah beberapa diagram *sequential* yang digunakan dalam pengembangan aplikasi ini :

1. *Sequential Diagram Login*

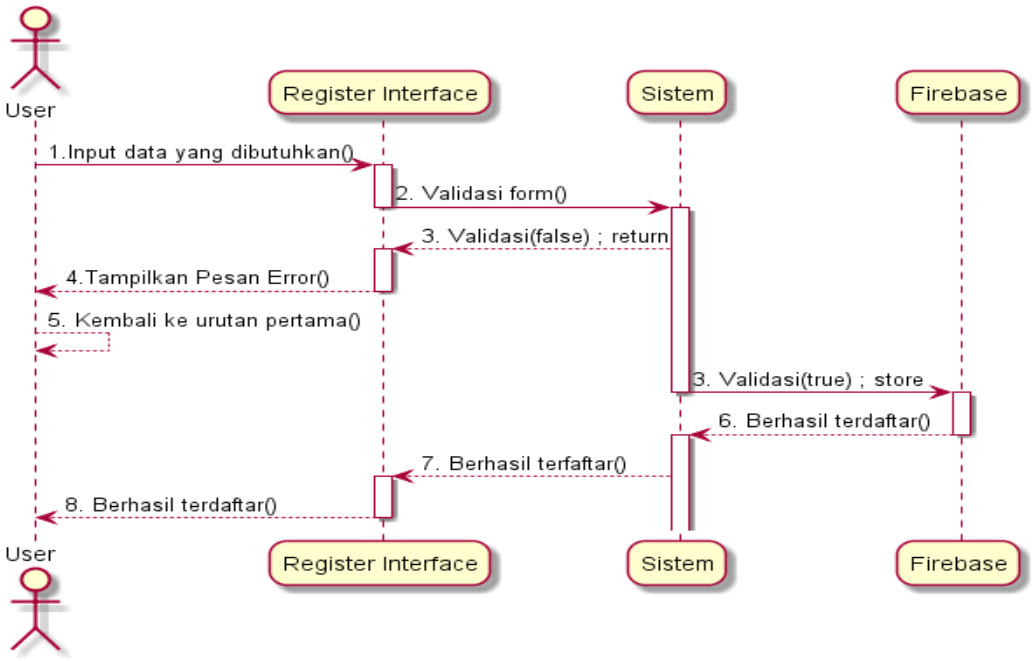
   Adalah sebuah alur diagram yang menunjukkan kegiatan atau flow pada diagram dari kegiatan atau *flow* aktifitas *login* dalam aplikasi *platform* sewa sarana olahraga berbasis android ini, sama seperti kegiatan login umumnyam aktifitas *sequential login* pada aplikasi inu juga dilakukan demikian yaitru secara sederhan dengan memasukkan data yang di per-lukan seperti *email* dan *password*, ataupun nomor telepon untuk melakukan autentikasi berupa pesan *one time password* atau yang biasa disingkat dengan OTP, dan juga *login* menggunakan *google account*.



Gambar 4.19 : Sequential diagram login

Sequential diagram diatas merupakan rangkaian pesan antar objek saat user melakukan login. Dimulai ketika user membuka aplikasi user akan disambut oleh sebuah tampilan *splash*, atau tampilan selamat datang dimana saat tampilan ini dibuka, dibalik layar aplikasi akan melakukan sebuah aktifitas pengecekan apakah user pernah masuk sebelumnya dalam aplikasi ini, proses ini bisa dinamakan dengan *session*. Dalam melakukan pengecekannya sistem melakukan *request* ke server (dalam hal ini dilakukan oleh *firebase*) dan server akan memberi *response* apakah user sudah pernah login atau belum. Jika *response* yang dikembalikan bernilai benar maka user bisa langsung menggunakan aplikasi (aplikasi menampilkan *interface* utama), dan jika *response* nya bersifat negasi maka user akan menuju *interface login*. Lalu user akan melakukan kegiatan *input*, melalui inputan ini sistem akan meminta *request* kembali *firebase* apakah data tersebut ada, dan jika ada maka user memiliki hak akses untuk menggunakan aplikasi dan jika tidak maka user mengulang kembali proses *inputan.*

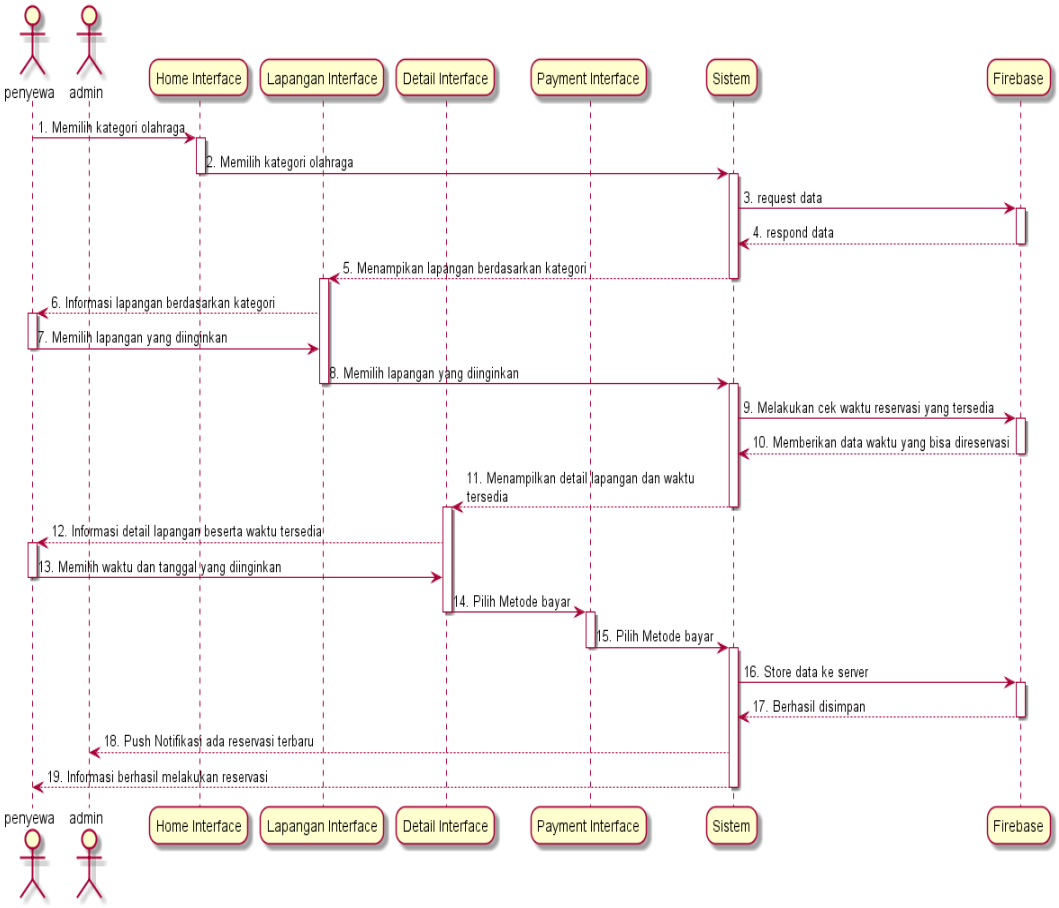
2. *Sequential Diagram Register*



Gambar 4.20 : Diagram sequential register

Gambar diatas mempresentasikan rangkaian pesan antar objek saat user melakukan proses *register* atau pendaftaran akun untuk menggunakan aplikasi tersebut. Diagram dimulai dari kegiatan *input* data diri kedalam *interface*/antar muka aplikasi kemudian dibelakang layar sistem akan melakukan validasi apakah user sudah benar-benar melengkapi *form* tersebut, jika validasi berjalan baik maka sistem akan meminta *request* data ke *firebase* dan memberi *response* bahwa data *user* telah berhasil terdaftar dalam *database*, akan tetapi jika validasi ditolakoleh sistem maka sistem akan memberi sebuah *return/callback* yang menginformasikan bahwa *user* harus melengkapi data *form* terlebih dahulu.

3. *Sequential Diagram* Reservasi Lapangan

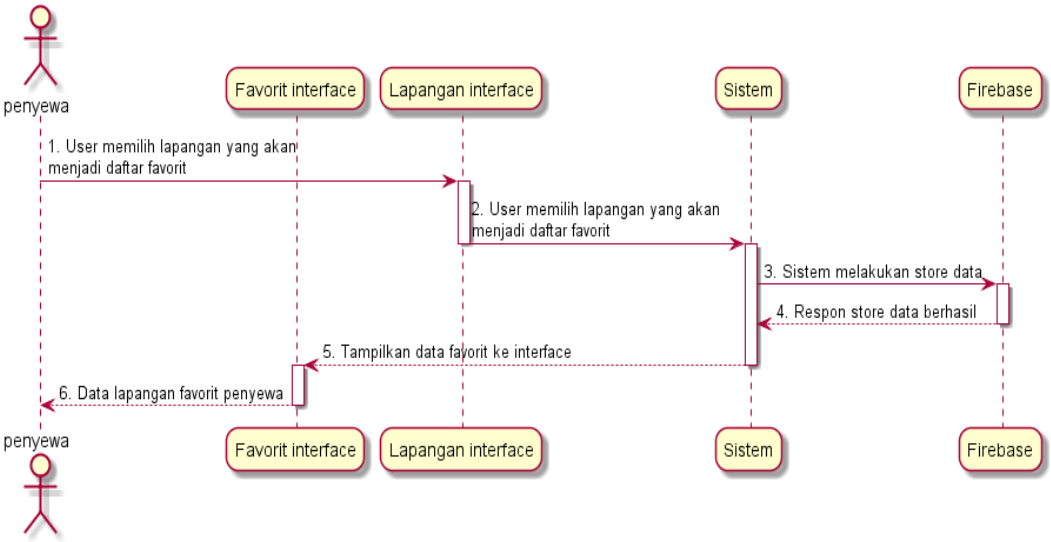


Gambar 4.21 : Sequential diagram reservasi

Pada *sequence* diagram ini menjelaskan skenario tentang bagaimana proses reservasi dalam sistem terjadi skenario pada proses transaksi sewa menyewa atau proses reservasi ini melibatkan dua aktor sebagai *end user* yaitu calon penyewa dan pengelola lapangan sebagai *administatrator*, awal skenario atau alir dari sistem ini bermula saat calon penyewa akan memilih kategori olahraga yang ingin mereka mainkan dan mencari lapangan yang menurut mereka sesuai standar yang mereka butuhkan , seperti yang telah dibahas sebelumnya bahwa pada penelitian ini akan mencakup empat kategori olahraga yaitu futsal, voli, bulu tangkis, dan basket. Setelah calon penyewa memilih kategori yang mereka inginkan maka dibelakang layar sistem akan melakukan sebuah permintaan data

lapangan berdasarkan kategori yang dipilih oleh calon penyewa sebelumnya dan jika permintaan berhasil maka akan menampilkan sebuah *list item* atau daftar yang sesuai permintaan yang diberikan dari sistem sebelumnya, yang kemudian data ini di infokan kedalam *interface* sistem, proses selanjutnya calon penyewa bisa memilih lapangan yang disukai nya dan yang mereka inginkan, dan untuk melakukan hal tersebut calon penyewa melihat informasi detail dari lapangan yang dirasa sesuai dengan kebutuhan dari calon penyewa tersebut, kemudian dibelakang layar sebuh permintaan kembali dilakukan oleh sistem kembali dilakukan kepada server untuk melakukan pengecekan ketersediaan data waktu pada semua tanggal yang ada di *database* dan menampilkannya kembali kepada penyewa agar penyewa bisa mendapatkan informasi waktu yang sudah tereservasi pada tanggal tertentu. Setelahnya penyewa dapat memilih waktu yang tersedia untuk melakukan reservasi lapangan, dan sistem kembali melakukan permintaan data yaitu dengan mengirim data reservasi ke server dan memberikan *push notification* ke admin bahwa ada informasi reservasi terbaru dari penyewa, dan skenario bagaimana proses transaksi sewa menyewa yang menggunakan cara *online* pun selesai sampai disini. Tentunya dari diagram di atas merupakan solusi yang telah dipikirkan sebelumnya dalam penelitian ini dalam memberikan solusi dari permasalahan transaksi sewa lapangan yang masih menggunakan cara konvensional, permasalahan seperti pelanggan atau calon penyewa dalam hal ini yang semulanya harus repot untuk datang ke lapangan atau tempat sewa sarana olahraga tersebut tidak perlu lagi datang ke lokasi karena sudah bisa menggunakan cara yang lebih mudah dan menghemat waktu, yaitu dengan menggunakan *platform* sewa sarana olahraga berbasis android ini sehingga pengalaman pengguna dalam melakuikan transaksi sewa sarana olahraga menjadi lebih menarik, cepat, efisien, efektif dan mudah tentunya.

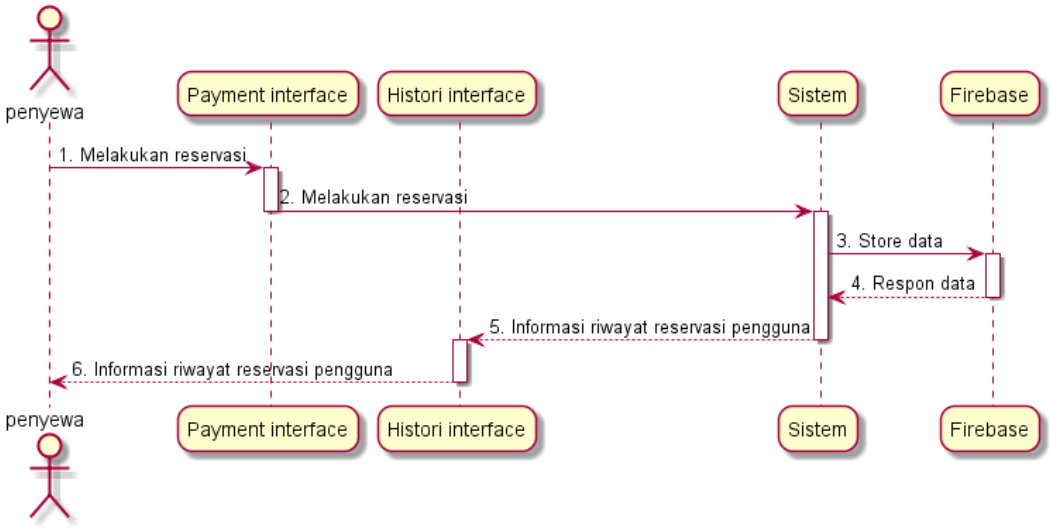
4. *Sequential Diagram Favorit*



Gambar 4.22 : Sequential Diagram Favorit

Diagram *sequential* favorit dimulai dari calon penyewa yang memlilih lapangan yang disukainya pada halaman lapangan, setelah itu sistem akan melakukan store data keserver ketika server berhasil merespon maka sistem akan mengembalikan bahwa lapangan yang dipilih user berhasil di-*store* kedalam database dan lapangan yang dipilih tadi ditampilkan kedalam *list* (daftar) pada antar muka favorit.

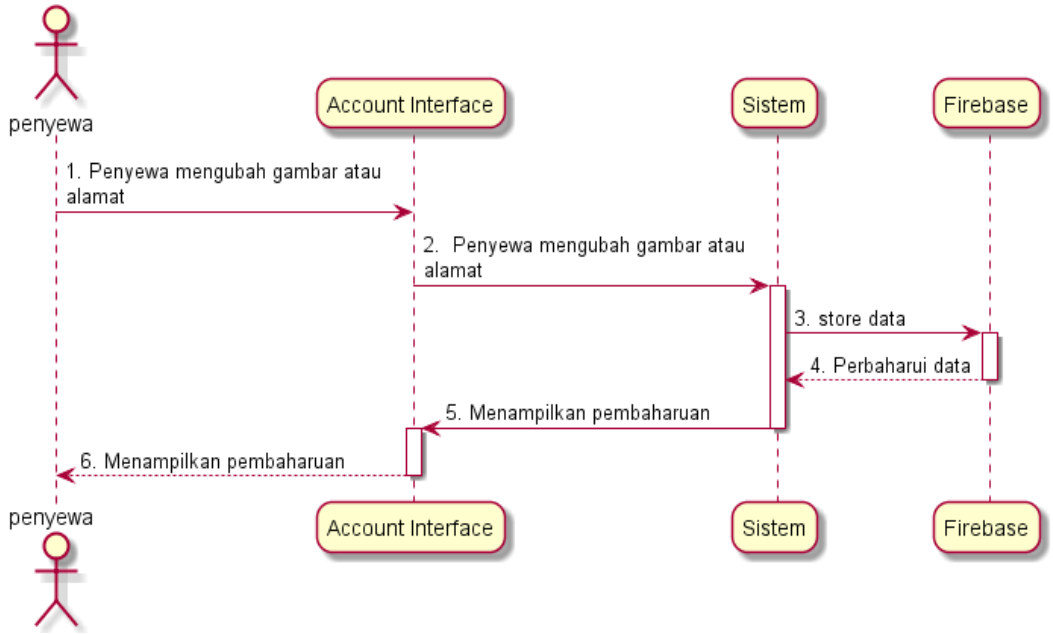
5. *Sequential Digram* Histori



Gambar 4.23 : Sequential diagram histori

Pada diagram diatas merupakan diagram yang menjelaskan proses komunikasi data dalam fitur histori pada pengembangan sistem ini, dimulai dari saat proses reservasi selesai dilakukan, dibalik layar saat sistem melakukan *store* data sistem sekaligus mengambil informasi terkait pencatatan transaksi yang pernah dilakukan penyewa, dan informasi yang didapatkan ini akan ditampilkan dihalaman histori *interface* yang bisa dilihat oleh calon penyewa.

6. *Sequential Diagram* Manajemen Akun *(Edit Profile)*

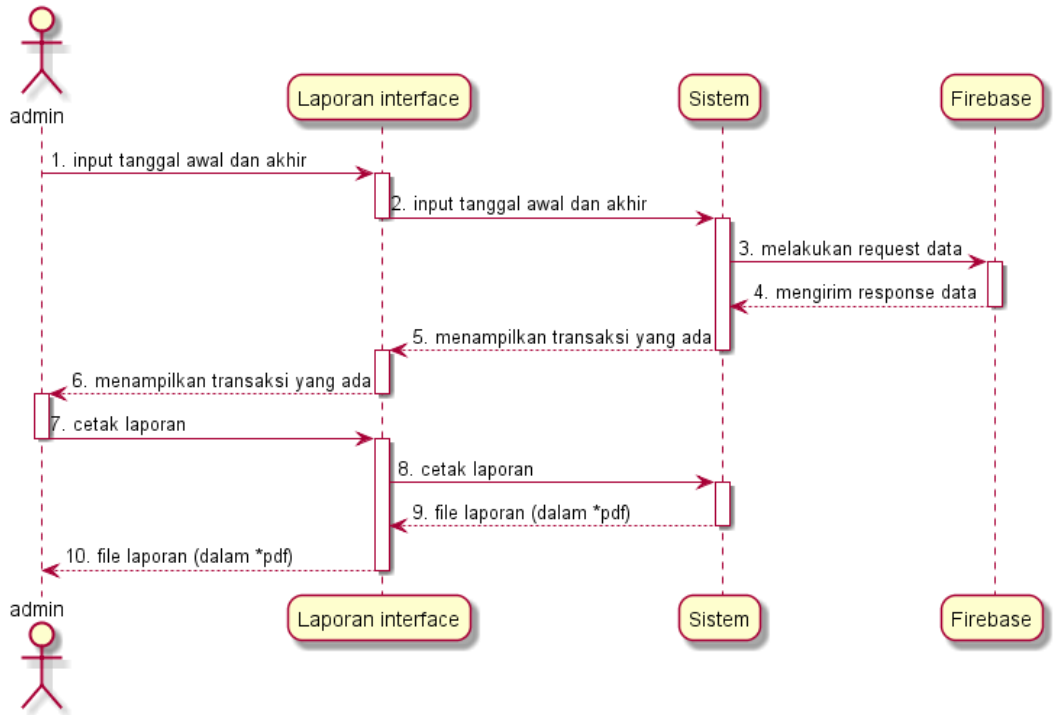


Gambar 4.24 : Sequential diagram manajemen akun

Diagram diatas menjelaskan proses *sequential* dari aktifitas user melakukan manajemen akun, yaitu dimulainya dengan penyewa melakukan ubah gambar atau alamat kemudian sistem akan melakukan *store* data dan firebase akan memberi informasi kesistem bahwa data sukses diperbaharui, lalu sistem akan menampilkan pembaharuan sesuai inputan yang dilakukan user.

4.2.1.5. Sequential Diagram Administrator

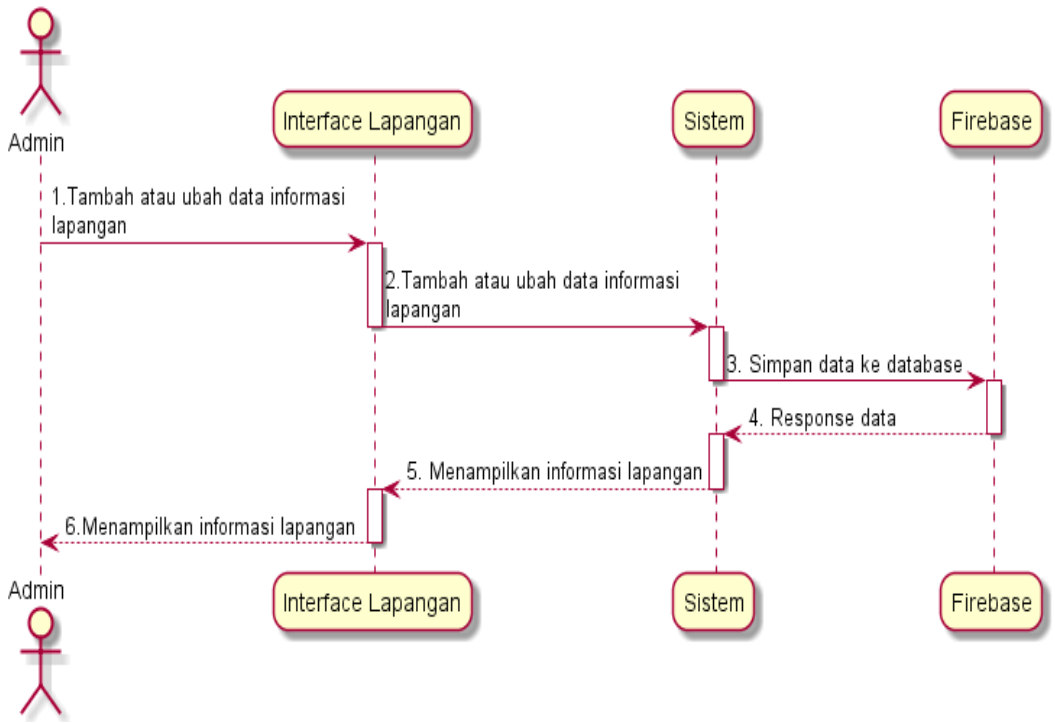
1. *Sequential Diagram* Manajemen Keuangan



Gambar 4.25 : Sequential manajemen keuangan

Pada *sequence* diagram diatas menjelaskan *sequential* atau skenari tentang bagaimana sebuah proses dari manajemen keuangan berjalan dan apa saja yang bisa dilakukan oleh admin dan sistem,proses awalnya dimulai dari kegiatan sebuah *input* berupa tanggal awal dan tanggal akhir oleh admin yang dimana nantinya disistem tanggal ini adalah berguna sebagai paramater untuk sistem melakukan permintaan ke server. baru setelah itu proses dibalik layar sistem akan melakukan request data dan mengirim response data kemudian menampilkan data tersebut (transaksi yang ada) diantar muka laporan pada sistem lalu informasi ini dilihat oleh admin, selain itu admin juga bisa melakukan kegiatan cetak laporan dan sistem akan mengolah data yang ada dalam bentuk file berekstensi *pdf.

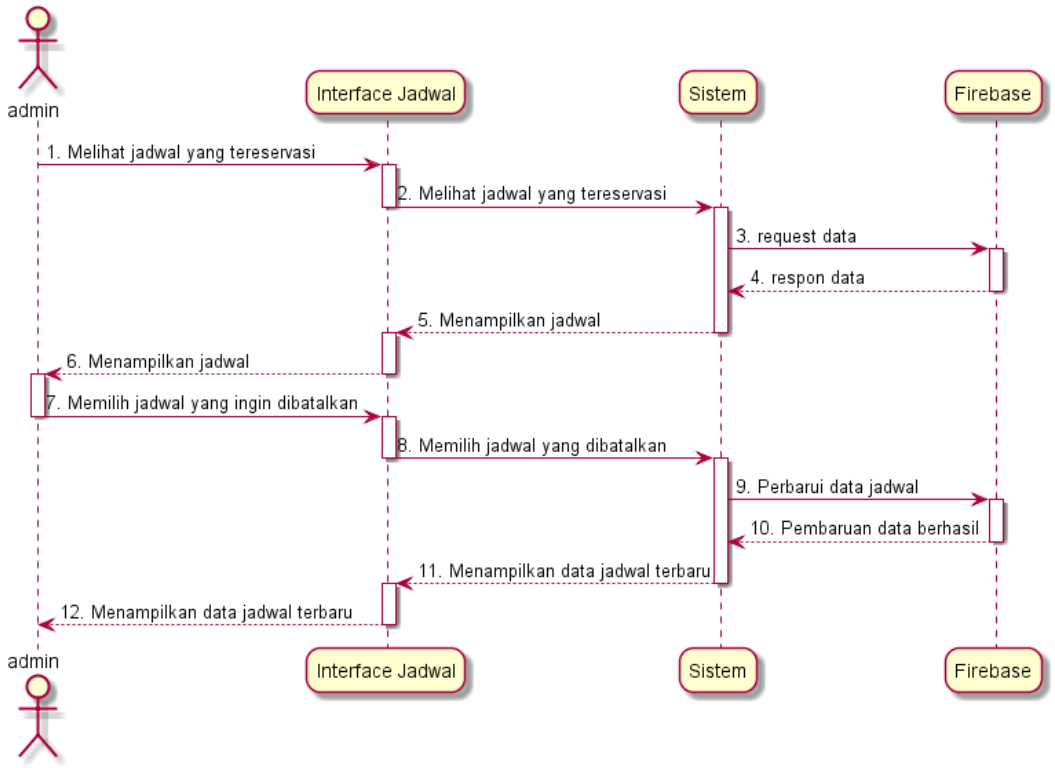
2. *Sequential Diagram* Manajemen Lapangan



Gambar 4.26 : Sequential manajemen lapangan

Bentuk *sequence* diagram diatas merupakan *sequential* diagam dari proses manajemen lapangan, dalam *platform* ini pengguna (pengelola lapangan) dapat melakukan perubahan informasi ataupun menambah informasi terkait lapangan yang akan disewakan dan di informasikan dalam *platform* sewa sarana olahraga, proses *sequence diagram* diatas bermula dari pengguna atau pengelola lapangan yang membuka antarmuka menu kelola lapangan dan kemudian mengisi field yang dibutuhkan dalam form yang telah disediakan oleh sistem *platform* sewa sarana olahraga, lalu seacara sistem akan melakukan *request* data atau permintaan data ke *firebase* dan server *firebase* akan memberikan sebuah *response* data sehingga sistem akan menampilkan informasi lapangan yang telah didapatkan dari *firebase* ke pengguna, dan proses manajamen lapangan pun berakhir sampai disini.

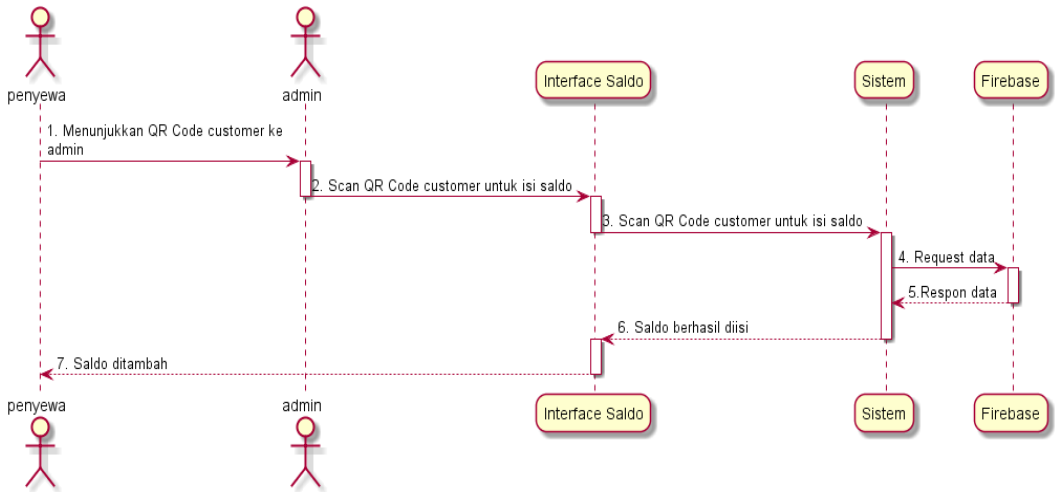
3. *Sequential Diagram* Manajemen Jadwal



Gambar 4.27 : Sequential Manajemen Jadwal

Manajemen jadwal dimulai dengan admin atau pengelola lapangan melakukan sebuah aksi untuk melihat jadwal yang tereservasi dalam antarmuka menu jadwal, kemudian secara sistem melakukan *request* data, dan memberikan respon kepada sistem berupa jadwal yang telah direservasi oleh penyewa (dalam kasus metode pembayaran *cash*), dan secara teknis jika penyewa tidak datang tepat waktu untuk melunasi reservasi atau membayar tagihan atas reservasi yang telah dibuat calon penyewa dalam sisem yang ada, maka admin bisa memiliki hak untuk bisa melakukan pembatalan dengan cara memilih jadwal yang ingin dibatalkan kemudian sistem akan memperbaharui data tersebut dalam *database* dan menampilkan data jadwal terbaru kepada admin. Itulah alur-alur atau kegiatan secara *sequential* dalam melakukan aktifitas kelola saldo pada *platform* sewa sarana olahraga berbasis android.

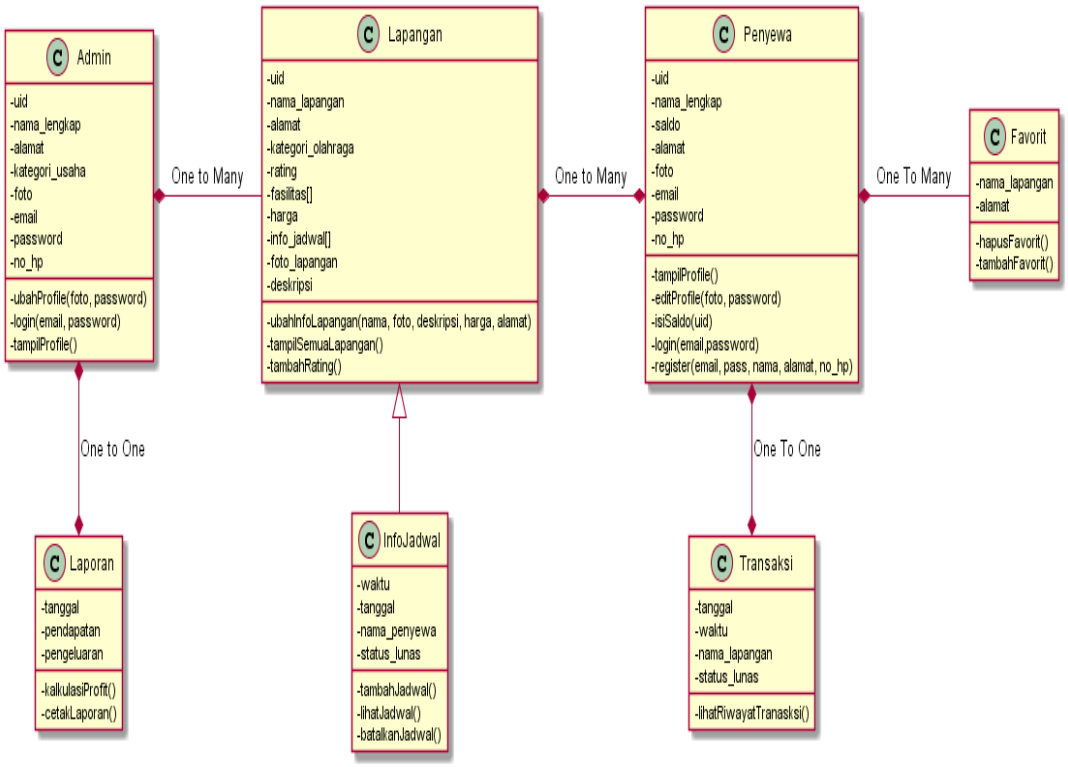
4. *Sequential Diagram* Manajemen Saldo (*Voucher*)



Gambar 4.28 : Sequential Diagram Voucher

Diagram diatas merupakan diagram *sequential* dari sebuah proses pengisian voucher kepada penyewa oleh admin selaku pengelola lapangang, dalam proses atau alurnya proses tersebut dimulai dari penyewa yang menunjukkan sebuah *qr code customer* yang didapatkan dari sistem atau aplikasi untuk calon pengguna, oleh sistem akan secara otomatis membuat sebuah *qr code* yang di *generate* berdasarkan *unique id* yang dimiliki oleh masing-masing calon penyewa yang sudah tefdaftar dalam *platform* sewa sarana olahraga, lalu calon penyewa akan menunjukkan *qr code* ini kepada admin selaku pengelola lapangan, lalu kemudian admin selaku pengelola lapangan yang berhak mengisis saldo vouche calon penyewa akan melakukan scan *qr code user* tersebut dan melakukan pengisian saldo sesuai nomial, lalu secara sistem akan melakukan *request data* dan menyimpan perubahan yang ada pada database yang kemudian ditampilkan pada antarmuka bahwa saldo berhasil ditambah.

4.2.1.6. *Class Diagram*



Gambar 4.29 : Class Diagram

Class diagram di atas menjelaskan hubungan antara objek-objek yang akan di implementasikan pada kode nantinya. Merujuk dari dua aktor dalam *usecase* yaitu Admin (pengelola lapangan) dan Penyewa, dapat di lihat bahwa admin dapat memiliki lebih dari satu lapangan dan admin hanya memiliki satu objek berupa kumpulan data laporan. Jika kita melihat *class* lapangan, dapat dilihat *class* info jadwal merupakan objek dari properti list info jadwal. Lalu pada aktor penyewa yang dalam *class diagram* dijadikan objek memiliki hubungan *one to many* ke objek favorit, yang artinya satu penyewa dapat memliki banyak data favorit atas lapangan-lapangan yang ada. Kemudian penyewa juga memliki hubungan satu ke satu pada objek transaksi dimana satu penyewa hanya bisa melakukan satu kali transaksi dalam satu waktu.

4.2.2. *Design Database*

Setelah melakukan *proses modelling* tahap selanjutnya adalah merancang dataase berdasarkan data dan informasi yang telah didapatkan, pada tahapan ini kita dapat menentukan entitas apa saja yang akan menjadi tabel nantinya dalam dbms kita. Pada database jenis NoSql (dalam hal ini *realtime firebase*) memiliki format JSON (*Javascript Object Notation*).

JSON (*JavaScript Object Notation*) adalah format pertukaran data yang sangat ringan serta lebih mudah dibaca dan ditulis oleh manusia, sehingga mudah untuk diterjemahkan dan dibuat (generate) oleh komputer. Dalam NoSql dikenal istilah *collection* dan *document*, istilah ini dalam database sql dapat disamakan dengan tabel dan baris. Untuk memudahkan pembacaan dalam penelitian ini penulis menggambarkan gambaran database yang akan dirancang dalam bentuk tabel lalu mengimplemantasikan rancangan tersebut dalam bentuk JSON.

4.2.2.1. *Collection Customers*

Tabel 4.2 Collection Customers

Key	***Type data***	**Deskripsi**
uid_customer	Object	Unique ID

Tabel 4.3 Child Dari Customers

Child	***Type data***	**Deskripsi**
alamat	String	Alamat domisi *user*
avatar	String	Foto *profile user*, yang mengarah pada url *firebase storage*
email	String	Alamat email user
emoney	Long	Saldo *user*
nama_lengkap	String	Nama lengkap *user*
password	String	Password *user*
nomor_hp	String	Nomor hp *user*
token	String	*Firebase cloud messanging* token

Representasi *collection* dan *document* dalam bentuk JSON

```
{
    "Customers" : {
        "uid_customer" : {
            "alamat" : "alamat user",
            "avatar"                                        :
"https://firebasestorage.googleapis.com/url....",
            "email" : "user@mail.com",
            "password" : "12345678",
            "emoney": 20,
            "nomor_hp" : "082298929999",
            "token" : "6lkGUkclk8578jnjYUhjcn"
        }
    }
}
```

4.2.2.2.*Collection Favorite*

Tabel 4.4 Collection Favorite

Key	***Type data***	**Deskripsi**
uid_customer	Object	Unique ID

Tabel 4.5 Child dari key uid_customer

Child	***Type data***	**Deskripsi**
uid_lapangan	Object	Key

Tabel 4.6 Child dari key uid_lapangan

Key	***Type data***	**Deskripsi**
alamat	String	Alamat lapangan olahraga
nama_lapangan	String	Nama lapangan yang menjadi favorit

Representasi dalam bentuk JSON

```
{
    "Favorite" : {
        "uid_customer" : {
            "uid_lapangan": {
                "alamat" : "alamat dari lapangan",
                "nama_lapangan" : "nama dari lapangan"
            }
        }
    }
}
```

4.2.2.3.*Collection Admin*

Tabel 4.7 Collection Admin

Key	***Type data***	**Deskripsi**
uid_admin	Object	Unique ID

Tabel 4.8 Child dari key uid_admin

Child	***Type data***	**Deskripsi**
alamat	String	Alamat domisi *user*
avatar	String	Foto *profile user*, yang mengarah pada url *fire-base storage*
email	String	Alamat email user
nama_lengkap	String	Nama lengkap *user*
password	String	Password *user*
nomor_hp	String	Nomor hp *user*
token	String	*Firebase cloud messanging* token
Kategori_usaha	String	Kategori usaha penyedia lapangan

Representasi kedua tabel diatas dalam bentuk JSON

```
{
    "uid_admin" : {
        "uid_customer" : {
            “alamat” : “alamat”,
            “email” : “email”
            “nama_lengkap” : “nama_lengkap”
            “kategori_usaha” : “kategori_usaha”
            “uid_lapangan” : “uid_lapangan”
            “token” : “token”
        }
```

4.2.2.4.*Collection Kategori*

Tabel 4.9 Collection Kategori

Key	***Type data***	**Deskripsi**
kategori	Object	Key

Tabel 4.10 Child dari key kategori

Child	***Type data***	**Deskripsi**
futsal	Object	Key
Volley	Object	Key
Basket	Object	Key
Badminton	Object	Key

Tabel 4.11 Child dari key masing-masing kategori

Child	***Type data***	**Deskripsi**
uid_lapangan	Object	Key
alamat	String	Alamat Lapangan
gambar	String	Gambar Lapangan
kategori	String	Kategori Lapangan
harga	Integer	Harga lapangan
nama_lapangan	String	Nama Lapangan
reservasi	Object	Key

Tabel 4.12 Child dari key reservasi

Child	***Type data***	**Deskripsi**
slot	Integer	Representasi waktu
tanggal	String	Representasi tanggal
uid_pelanggan	String	Unique id pelanggan

Representasi tabel diatas dalam bentuk JSON

```
"kategori" : {
    "futsal": {
        "uid_lapangan" : "",
        "nama_lapangan" : "",
        "alamat" : "",
        "gambar" : "",
        "kategori" : "",
        "harga" : "",
        "reservasi" :{
            "slot" : "",
            "tanggal": "",
            "uid_pelanggan":""
        }
    }
    "basket" : {}
    "volley" : {}
    "badminton" : {}
}
```

4.2.2.5.*Collection Saldo Voucher*

Tabel 4.13 Collection Saldo Voucher

Key	***Type data***	**Deskripsi**
saldo	Object	Key

Tabel 4.14 Child dari key saldo

Child	***Type data***	**Deskripsi**
uid_user	Object	Key

Tabel 4.15 Child dari key uid_user

Child	***Type data***	**Deskripsi**
uid_lapangan	Object	Key

Tabel 4.16 Child dari key uid_lapangan

Child	***Type data***	**Deskripsi**
saldo	Integer	Sisa jumlah voucher yg bisa dipakai

Reperesentasi dalam bentuk JSON dari Saldo

```
"saldo" : {
    "uid_customer" : {
        "uid_lapangan" : {
            "saldo" : 0
        }
    }
}
```

4.2.2.6.*Collection Transaksi*

Tabel 4.17 Collection Transaksi

Key	*Type data*	Deskripsi
Transaksi	Object	Key

4.2.3. Design Interface

Pada subbab ini menjelaskan tentang bagaimana antarmuka yang akan diimplementasikan dalam sistem, *desain interface* (antarmuka aplikasi) dibangun menggunakan *wireframe*, secara sederhana *wireframe* adalah kerangka atau coretan kasar untuk penataan item-item pada halaman website atau mobile sebelum proses desain sesungguhnya dimulai.

Dalam pengembangan aplikasi berbasis android ada beberapa komponen utama yang disebut *view* atau *widget* adapun beberapa komponen tersebut adalah sebagai berikut :

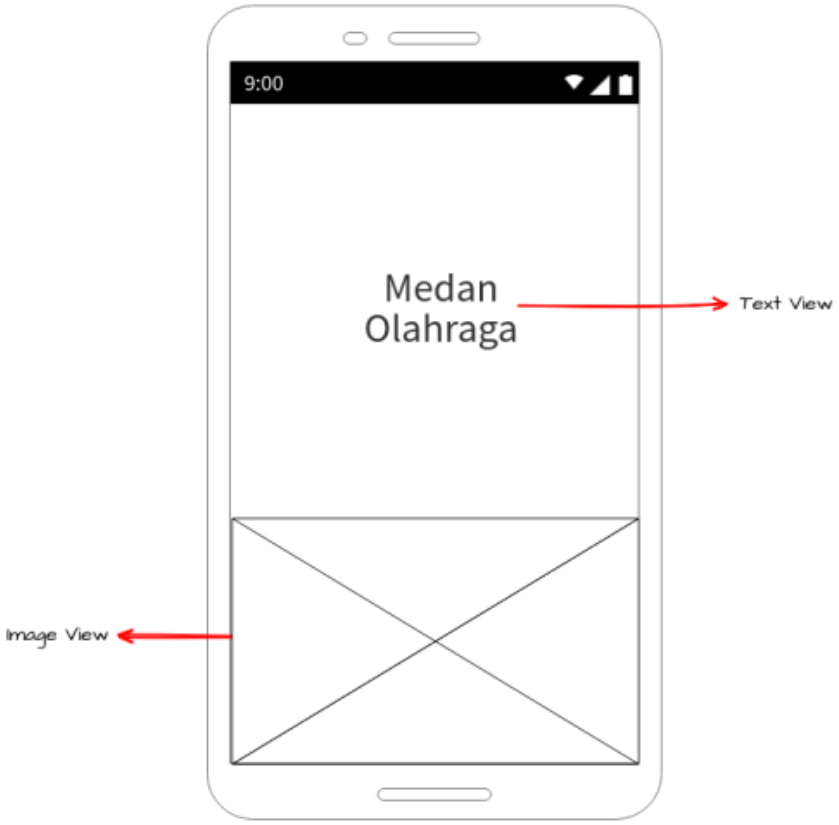
Tabel 4.18 Macam-Macam Widget Android

Nama Widget	Fungsi
Textview	Merupakan sebuah komponen yang menampilkan teks kepada pengguna, komponen ini cocok digunakan untuk menyampaikan informasi kepada pengguna dalam bentuk teks.
EditText	Merupakan sebuah komponen yang berfungsi sebagai wadah pengguna menginputkan suatu nilai/masukan yang dibutuhkan dalam sistem, contoh melakukan inputan data nama, tanggal lahir, dan sebagainya.
ImageView	Merupakan sebuah komponen yang

	mempresentasikan suatu gambar kepada pengguna, misalkan gambar avatar pengguna.
Button	Merupakan sebuah komponen aksi, dimana user bisa memberikan aksi atau perintah kepada sistem sesuai kasusnya.
RecyclerView	Adalah sebuah komponen yang tedapat pada android yg mempresentasikan perulangan data dalam bentuk list yang bisa di kustomasi.

Tabel diatas merupakan beberapa komponen-komponen view dalam SDK yang berada pada IDE *Android Stuidio* yang dapat dimanfaatkan dalam proses perangan dan pengembangan aplikasi berbasis *android*. Komponen-komponen yang disebutkan diatas dapat ditemukan dalam IDE *Android Studio* mulai dari versi pertama hingga versi terbarui kini, dalam penerapannya kompionen-komponen ini dapat digunakan dalam sebuah bahasa *markup xml*, sebuah bahasa yang umum digunakan dalam pengembangan aplikasi berbasis android, dan *default* dari *android studio* sendiripun menggunakan *markup xml languange* tersebut. Untuk menggunakan komponen-komponen *view* tersebut dalam *xml* bisa melalui *native code* ataupun secara mudah dengan menggunakan *drag and drop* pada menu *design* yang telah disediakan oleh *IDE Android Studio* itu sendiri. Komponen-komponen *view* tesebut juga bisa diubah dan dikustomisasi oleh pengembangan aplikasi seperti *edittext*, pengembang aplikasi bisa memberikan *styleing* berupa penambahan atribut seperti *background color, width, height, font color, font size, font style, hint*, dan bermacam atribut lainnya yang dapat di eksplorasi lebih lagi oleh pengembang aplikasi di dokumentasi resmi *android developer* pada situsnya. Kustomosisasi tadi juga tidak pada komponen *view edittext* saja tetapi juga berlaku untuk komponen *view* lainnya seperti *textview, imageview, recyclerview, button*.

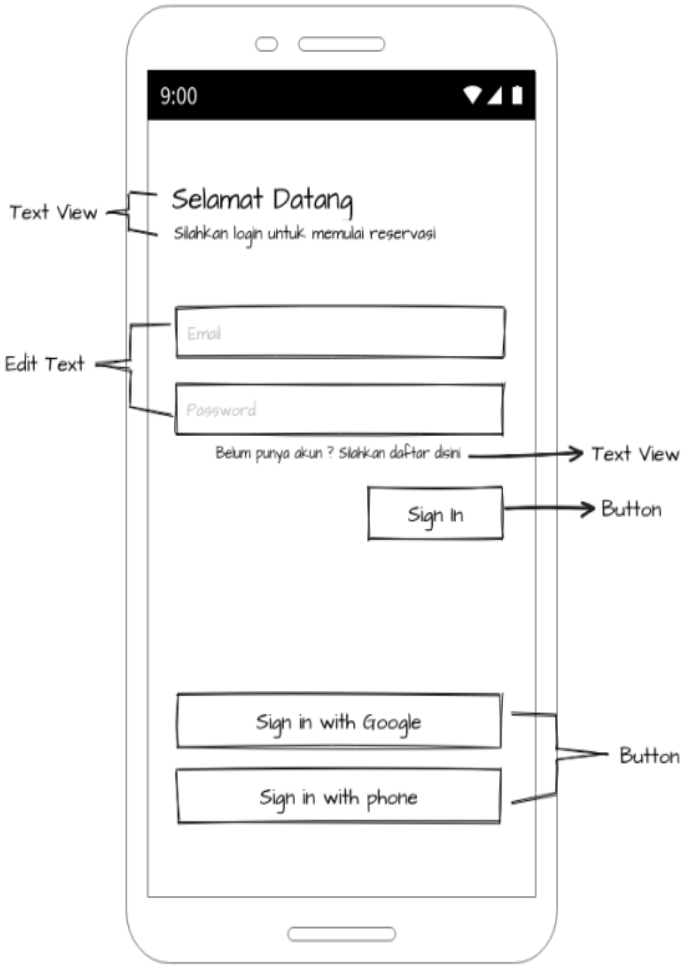
1. Tampilan *Wireframe Client*
   a) *Splash screen*



Gambar 4.30 Splash Screen Pada Aplikasi Client

Gambar di atas merupakan sebauh gambar rancangan antarmuka yang di sebut dengan *splash screen* di mana ini merupakan antarmuka atau *interface* yang di lihat user pertama kali sebelum masuk ke halaman *login* dan lainnya. Pada *Splash Screen* ini dapat di lihat dua komponen view yang tertera yaitu *Text View* dan *Image View*, antarmuka ini nant-inya hanya akan berjalan selama tiga detik yang di *handle* oleh layar belakang sistem, dan selama tiga detik ini pula beberapa aktifitas terkait kebutuhan sistem bisa dilakukan seperti melakukan atentikasi akun pengguna di balik layar, dan melakukan autentukasi *cloud messaging*.

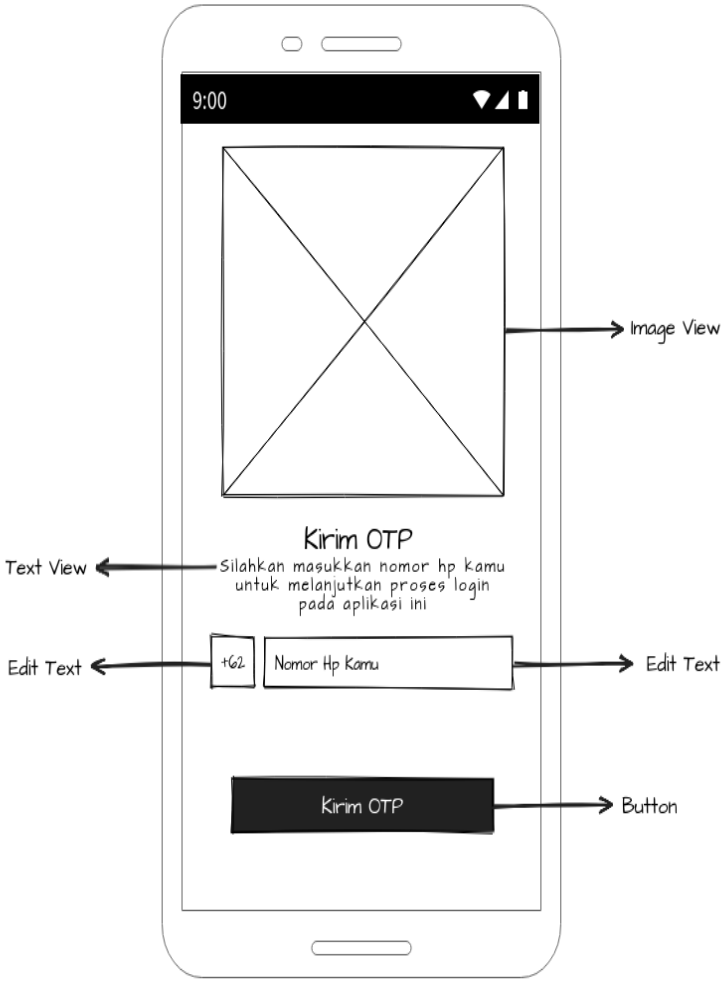
b) *Login* dengan *email*



Gambar 4.31 Tampilan Login Via Email

Pada gambar di atas merupakan sebuah gambar rancangan antarnuka di *platform* sewa sarana olahraga yang menjelaskan tentang sebuah aktifitas untuk melakukan kegiatan *login* menggunakan *email*, dimana pada rancangan antarmuka tersebut dapat di lihat komponen-komponen *view* dari *android* seperti *Textview, Edittext,* dan *Button*. Masing-masing komponen *view* ini memiliki peran-nya masing-masing untuk dapat melakukan dan membantu *user* dalam melakukan proses kegiatan *login* ke dalam *platform* sewa sarana olahraga.

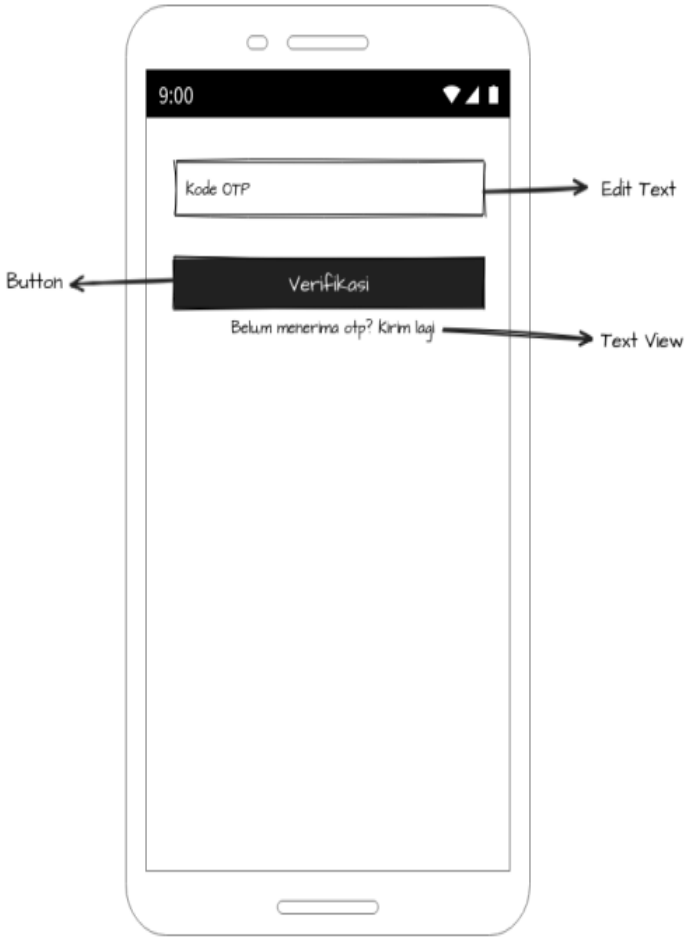
c) *Login* dengan nomor telepon



Gambar 4.32 Tampilan *Login* dengan Nomor Telepon

Pada gambar di atas merupakan sebuah gambar rancangan antarmuka pada *platform* sewa sarana olahraga, dimana antamuka tersebut memaparkan tentang *login* ke dalam sistem menggunakan nomor telepon, pada rancangan antarmuka tersebut dapat di lihat beberapa komponen *view* di antaranya adalah *Imangeview*, *Textview*, *Edittext*, dan *Button.* Dalam rancangan antarmuka ini *user* bisa melakukan *input* nomor telepon untuk melakukan proses autentikasi dengan cara mengirim kode OTP kepada server melalui *button* kirin OTP dan lalu kode OTP akan di kirimkan melalui layanan SMS.

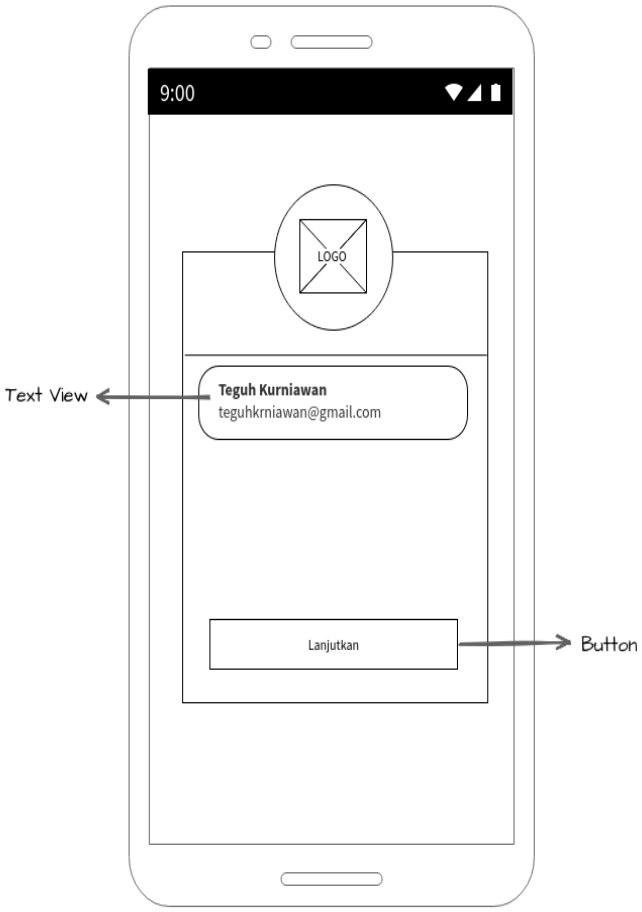
d) Verifikasi *One Time Password* (OTP)



Gambar 4.33 Tampilan Verifikasi Kode OTP

Pada gambar di atas merupakan sebuah gambar rancangan antarmuka pada *platform* sewa sarana olahraga, dimana antamuka tersebut memaparkan tentang verifikasi *login* ke dalam sistem menggunakan nomor telepon, di aman dalam rancangan ini sebuah aktifitas dari kode *One Time Password* sudah di kirimkan oleh server atau sistem melalui layanan SMS, dan rancangan ini berfungsi agar pengguna melakukan *inputan* dari kode tersebut dalam *field* atau *edittext* pada rancangan antarmuka tersebut. Pada gambar di atas rancangan tersebut terdiri dari komponen-kompoen *view* sebagai berikut yaitu *textview*, *button*, dan *edittext*.

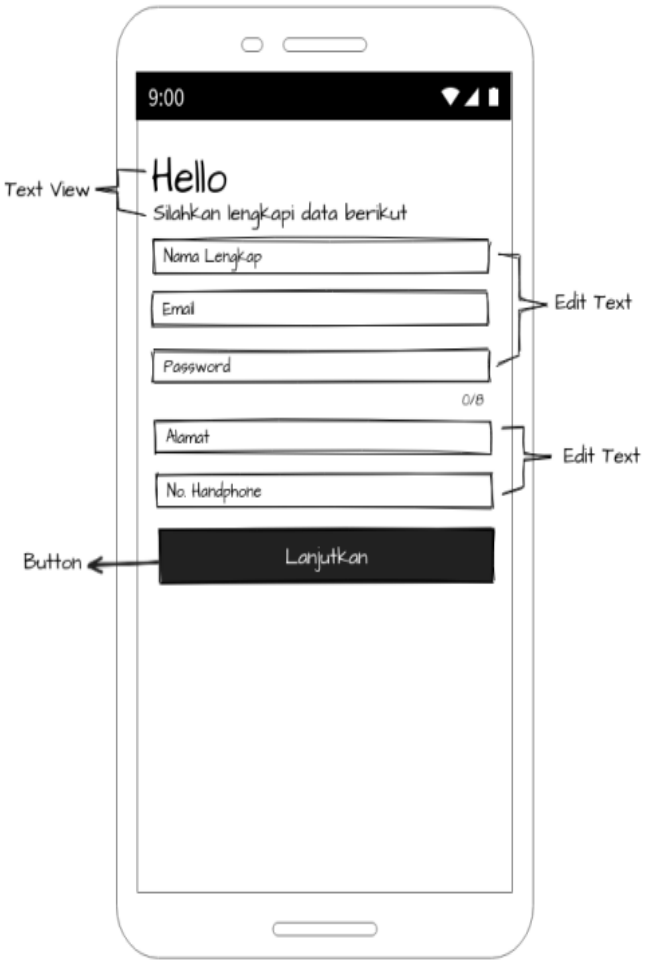
e) *Login* dengan akun google



Gambar 4.34 Tampilan *Login* menggunakan *Google Account*

Pada gambar di atas merupakan sebuah gambar rancangan antarmuka pada *platform* sewa sarana olahraga, dimana antamuka tersebut memaparkan tentang *login* ke dalam sistem menggunakan *Google Account* yang tertataut pada *smartphone* atau perangkat *user*, rancangan di atas bisa di bilang berbentuk *pop up* di mana dalam *pop up* tersebut terdapat bentuk daftar dari akun-akun *google account* yang sudah terdaftar dalam perangkat *user*, untuk melakukan proses autentikasi dengan cara ini *user* dapat melakukan tap pada akun yang ingin di gunakan dan melakukan *tap* pada *button* lanjutkan. Lalu pada rancangan ini terdapat komponen *view* seperti *textview* dan *button*.

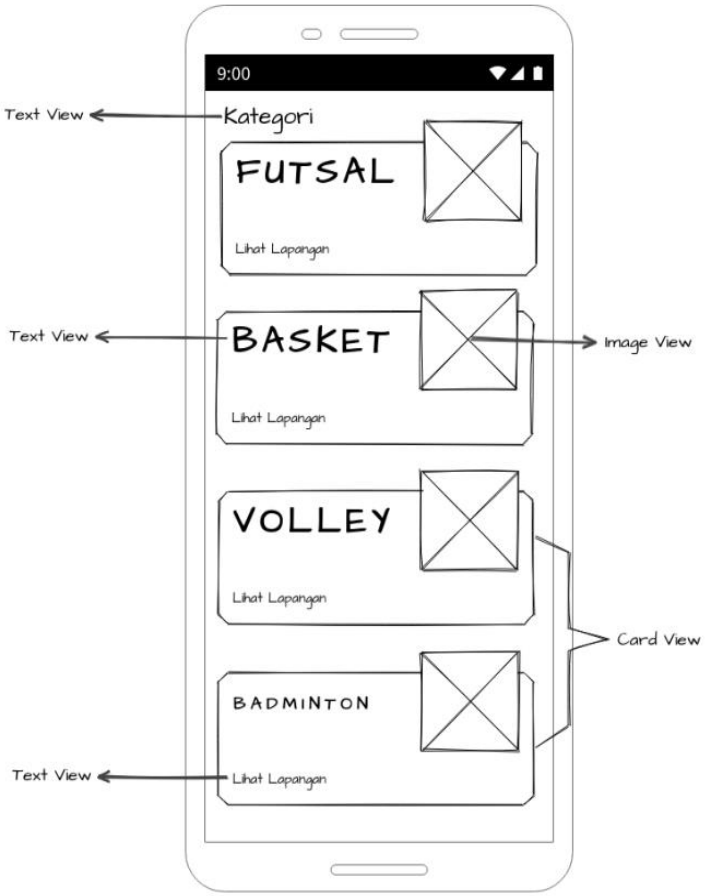
f) *Sign up* pengguna



Gambar 4.35 Tampilan *Sign up* pengguna

Pada gambar di atas merupakan sebuah gambar rancangan antarmuka pada *platform* sewa sarana olahraga, dimana antamuka tersebut memaparkan tentang pendaftaran ke dalam sistem *platform* sewa sarana olahraga, di mana dalam rancangan antarmuka ini *user* di haruskan untuk melengkapi data diri yang dibutuhkan dengan melakukan *inputan* data ke dalam *field-field* yang sudah di sediakan daam *form*. Adapun data diri yang terdapat dalam rancangan antarmuka ini adalah nama lengkap, *email*, *password*, alamat pengguna, dan nomor telepon. Adapun komponen view pada rancangan ini adalah *textview*, *edittext*, dan *button*.

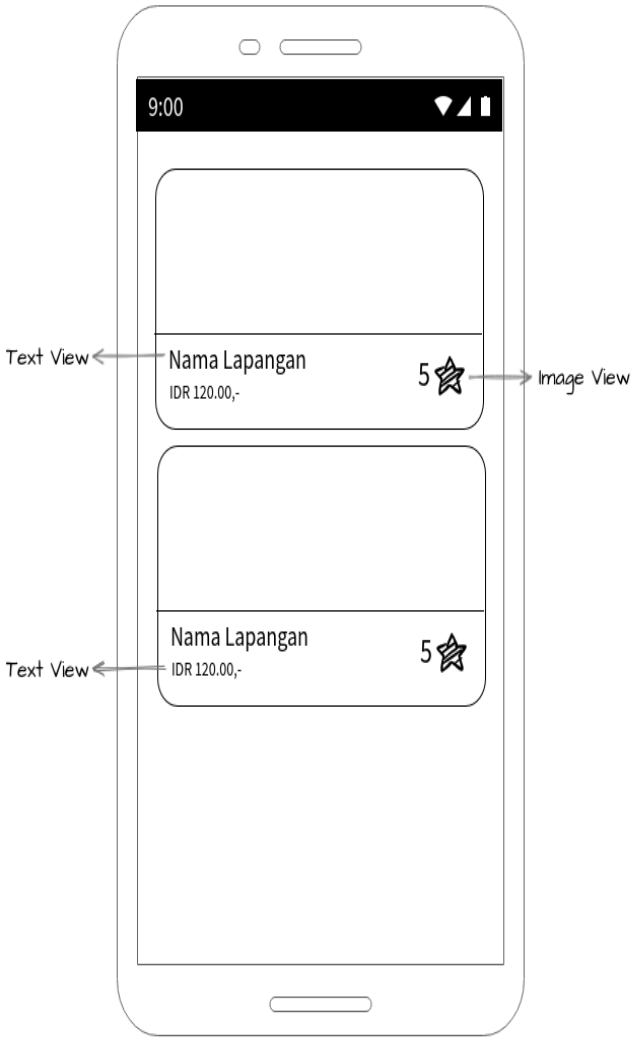
g) Halaman *Home*



Gambar 4.36 Tampilan *Home*

Pada gambar di atas merupakan sebuah gambar rancangan antarmuka pada *platform* sewa sarana olahraga, dimana antamuka tersebut memaparkan tentang antarmuka halaman utama atau biasa di sebut dengan *dashboard*. Dalam *dashboard* ini pengguna dapat melihat dan memilih kategori olahraga yang sarana olahraganya ingin mereka gunakan dan secara fisik *user* dapat melihat kategori-kategori olahraga ini dalam bentuk daftar. Dalam rancangan antarmuka menu *dashboard* atau halaman utama ini terdapat komponen view seperti *textview*, dan *cardview* yang disusun sedemikian rupa hingga berbentuk seperti yang ada pada gambar di atas.

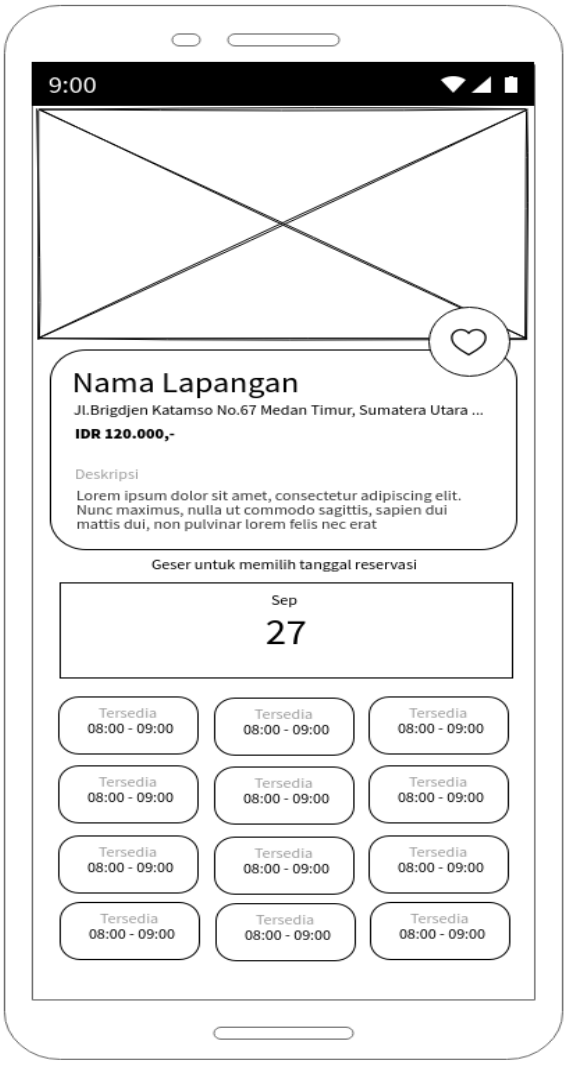
h) Halaman Daftar Lapangan



Gambar 4.37 Tampilan Halaman Daftar Lapangan

Pada gambar di atas merupakan sebuah gambar rancangan antarmuka pada *platform* sewa sarana olahraga, dimana antamuka tersebut memaparkan tentang daftar lapangan setelah *user* memilih kategori olahraga mana yang ingin digunakan dalam proses sewa menyewa tersebut, dalam rancangan dapat di lihat bahwa ada komponen-komponen *view* seperti *textview* yang mana *view* ini digunakan untuk menampilkan informasi seperti nama lapangan yang di sewakan oleh pihak pengelola lapangan dan menampikan informasi harga setiap jam par transaksi.

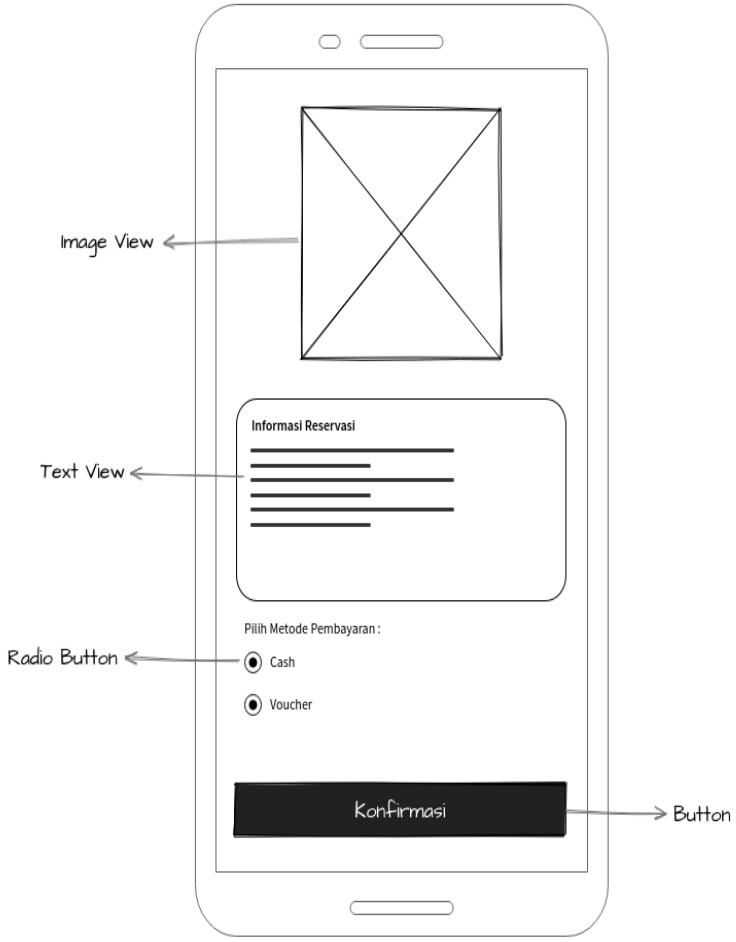
i) Halaman Detail Lapangan



Gambar 4.38 Tampilan Detail Lapangan

Pada gambar di atas merupakan sebuah gambar rancangan antarmuka pada *platform* sewa sarana olahraga, dimana antamuka tersebut memaparkan tentang detai lapangan setelah user memilih lapangan yang di inginkan dan dibutuhkan, pada antarmuka tersebut user bisa mengetahui infotmasi berupa foto dari lapangan yang akan di sewakan, informasi berupa nama lapangan, harga sewa lapangan, deskripsi lengkap tentang lapangan dan informasi jadwal yang tersedia, user juga dapat memilih waktu dan tanggal reservasi sesuai keinginan calon penyewa.

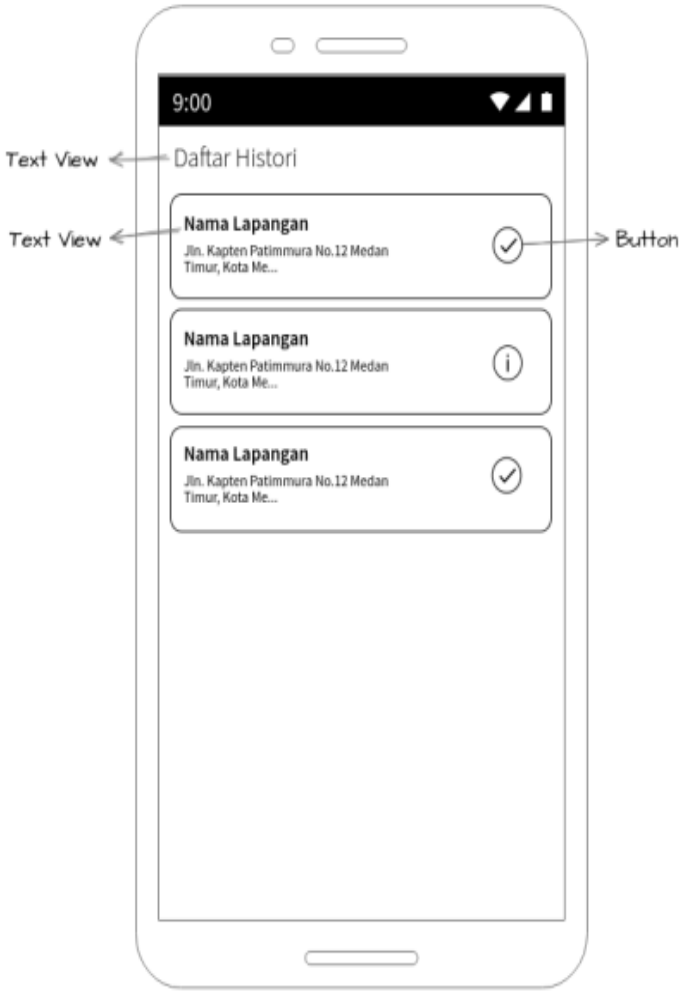
j) Halaman Konfirmasi



Gambar 4.39 Tampilan Halaman Konfirmasi

Pada gambar di atas merupakan sebuah gambar rancangan antarmuka pada *platform* sewa sarana olahraga, dimana antamuka tersebut memaparkan tentang konfirmasi lapangan atas transaksi sewa menyewa setelah *user* memilih jadwal yang tersedia sesuai kebutuha dari user selaku calon penyewa, dalam rancangan tersebut terdapat informasi berupa ringkasan terhadap data lapangan yang akan di sewa, dan pemilihan metode pembayaran yang terdiri dari metode pembayaran *cash* dan metode pembayaran voucher, baru setelah user memilih salah satu metode pembayaran tersebut barulah bisa di lakukan konfirmasi bahwasanya proses transaksi ini di setujui oleh user sebagai calon penyewa.

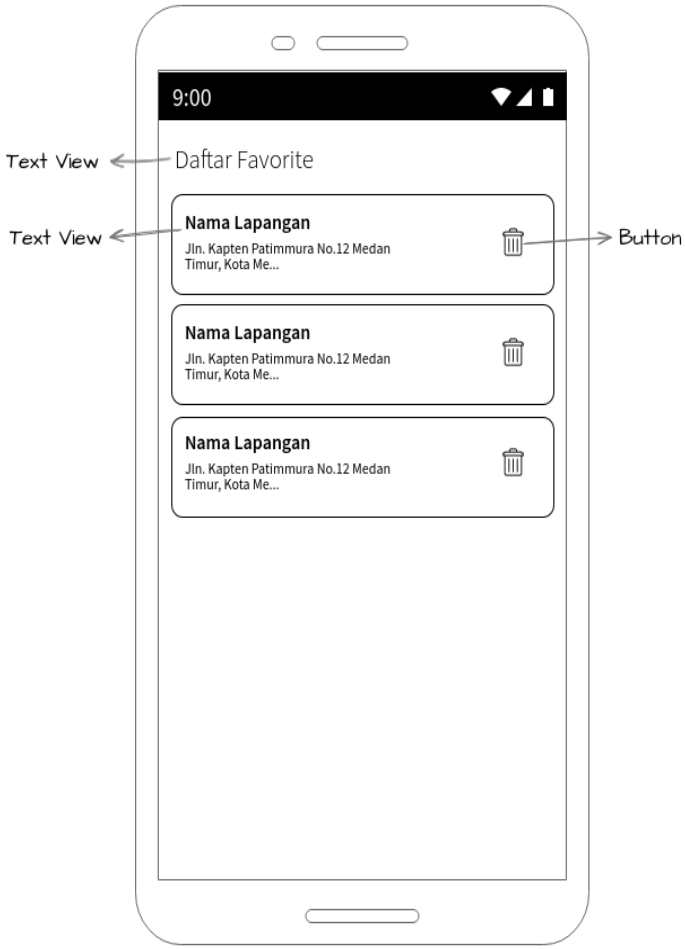
k) Halaman Histori



Gambar 4.40 Tampilan Halaman Histori

Pada gambar di atas merupakan sebuah gambar rancangan antarmuka pada *platform* sewa sarana olahraga, dimana antamuka tersebut memaparkan tentang histori atas transaksi yang pernah di lakukan oleh *user*, singkatnya secara sederhana ini adalah antarmuka berupa daftar yang berisikan informasi lapangan saja yang pernah di sewa oleh pengguna selaku penyewa, dalam daftar tersebut terdapat informasi berupa nama lapangan dan alamat lapangan dan juga status dari pemesanan yang di lakukan oleh pengguna selaku penyewa atas transaksi yang telah di lakukan sebelumnya.

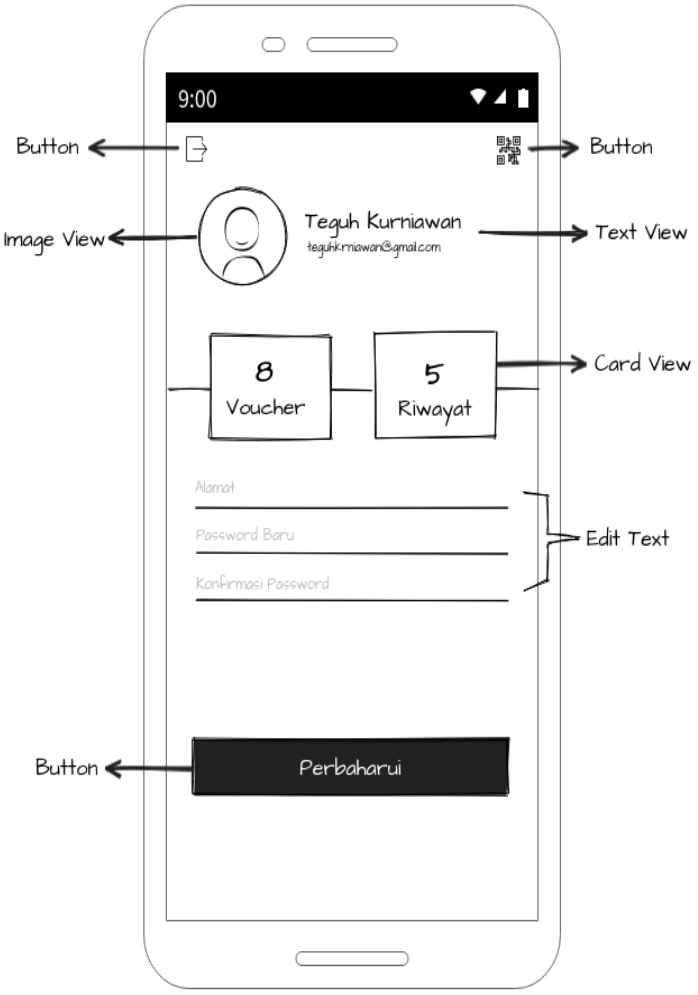
l) Halaman Favorite



Gambar 4.41 Halaman Daftar Favorite

Pada gambar di atas merupakan sebuah gambar rancangan antarmuka pada *platform* sewa sarana olahraga, dimana antamuka tersebut memaparkan tentang daftar favorit lapangan yang telah di tambahkan user dalam hal ini adalah calon penyewa pada halaman detail lapangan, dalam rancangan antarmuka ini terdapat sebuah daftar yang di mana masing-masing dari daftar tersebut berupa infomasi seperti nama lapangan, alamat lapangan, dan sebuah *button* berbentuk *icon trash* yang berfungsi untuk mengahapus item lapangan yang telah di favoritkan dari daftar favorit pengguna atau *user*.

m) Halaman Akun

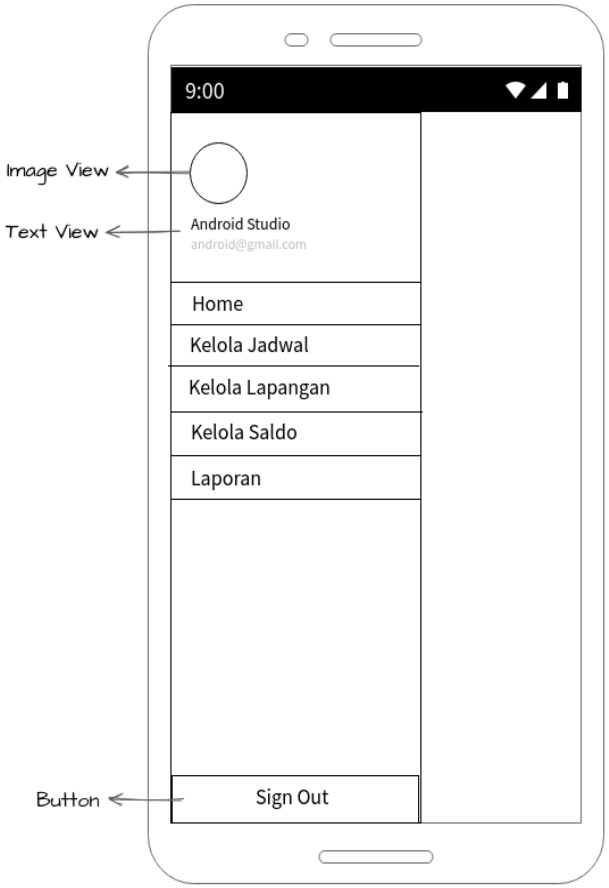


Gambar 4.42 Tampilan Account

Pada gambar di atas merupakan sebuah gambar rancangan antarmuka pada *platform* sewa sarana olahraga, dimana antamuka tersebut memaparkan tentang informasi akun pengguna, di mana dalam rancangan antarmuka ini user bisa melakukan *generate qr code*, mengubah data diri, informasi seperti nama dan proses *logout* apabila user melakukan aksi pada *button logout*, komponen view pada rancangan antarmuka ini ada *edittext*, *button*, *imageview*, dan juga *cardview*.

2. Tampilan Wireframe Admin

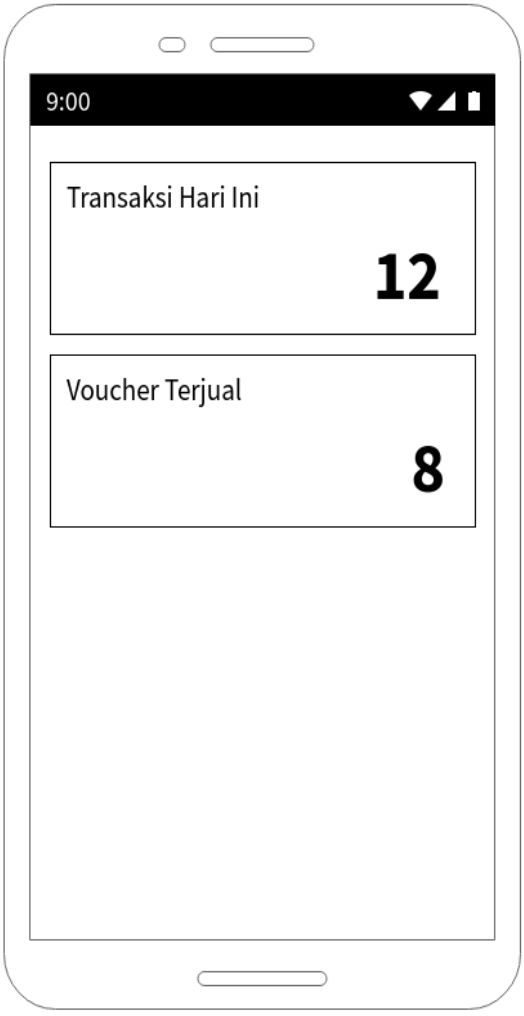
a) Halaman Menu



Gambar 4.43 Tampilan Menu Aplikasi Admin

Pada gambar di atas merupakan sebuah gambar rancangan antarmuka pada *platform* sewa sarana olahraga, dimana antamuka tersebut memaparkan tentang menu apa saja yang bisa di dapatkan oleh pengelola lapangan selaku *administrator*, menu-menu tersebut di antaranya adalah *home* atau halaman utama, kelola jadwal, kelola lapangan, kelola saldo, dan juga laporan, serta *sign out* yang berfungsi apabila pengguna ingin mengeluarkan akun tersebut dari aplikasi. Adapun komponen-komponen *view* yang bisa terdapat dalam aplikasi tersebut adalah *button*, *textview*, dan *imageview*.

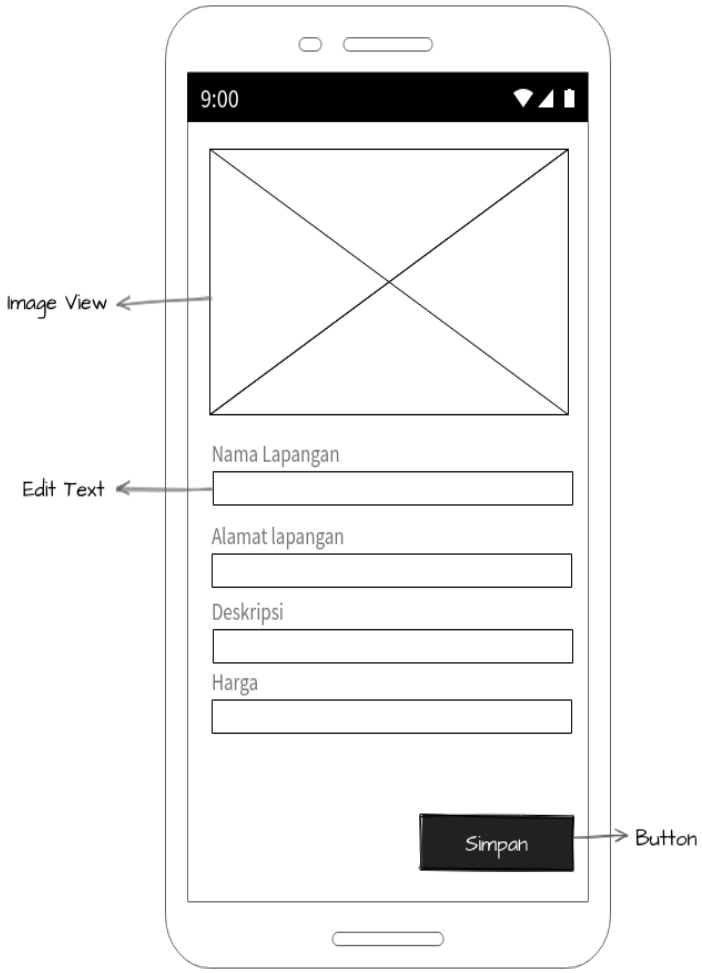
b) Halaman Home



Gambar 4.44 Tampilan Admin Home

Pada gambar di atas merupakan sebuah gambar rancangan antarmuka pada *platform* sewa sarana olahraga, dimana antamuka tersebut memaparkan tentang halaman home yang bisa di lihat oleh pengelola lapangan selaku *administrator*, dalam abtarmuka ini pengelola lapangan dapat melihat jumlah transaksi hari ini, dan juga jumlah voucher yang sudah terjual.

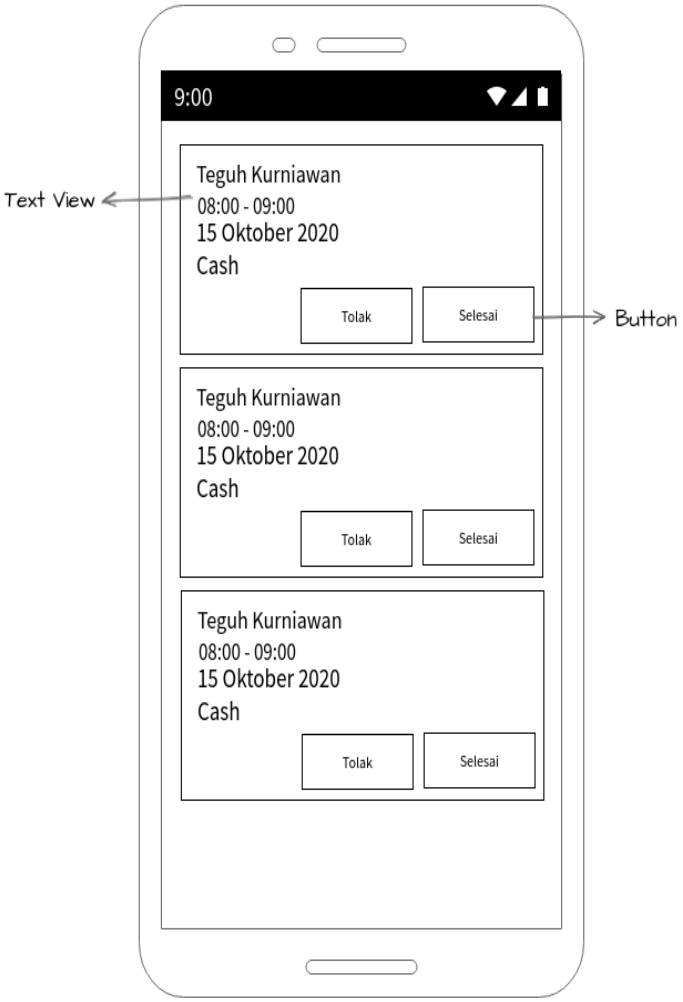
c) Halaman Kelola Lapangan



Gambar 4.45 Tampilan Kelola Lapangan

Pada gambar di atas merupakan sebuah gambar rancangan antarmuka pada *platform* sewa sarana olahraga, dimana antamuka tersebut memaparkan tentang pengelolaan lapangan yang akan disewakan oleh pengelola lapangan selaku *administrator*, dalam antarmuka tersebut user diwajibkan untuk melakukan kegiantan input atas data lapangan yang dibutuhkan oleh sistem yaitu seperti nama lapangan, alamat lapangan, harga lapangan, dan deskripsi lapangan. Adapun komponen-komponen *view* yang bisa terdapat dalam aplikasi tersebut adalah *button*, *textview*, dan *imageview*.

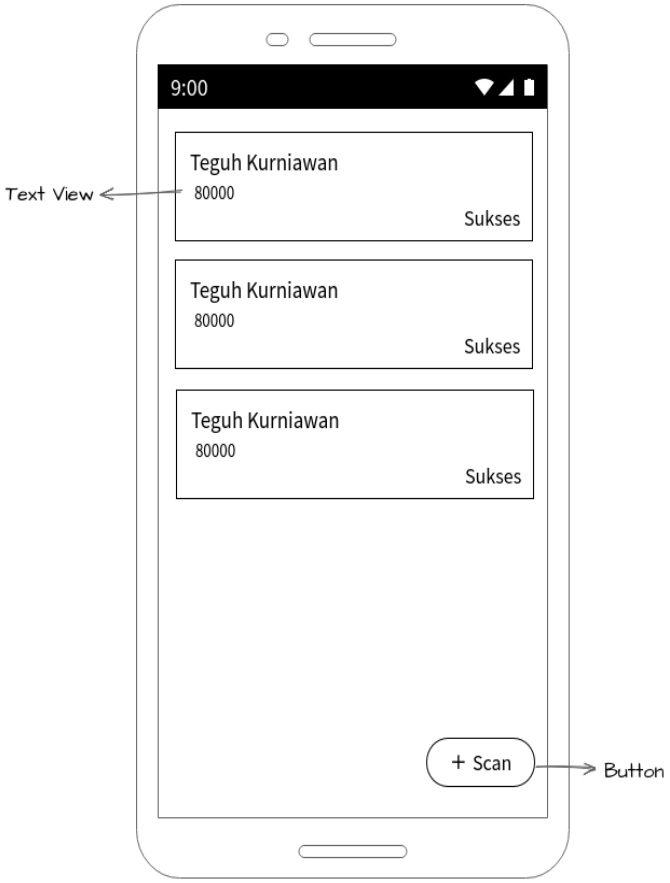
d) Halaman Kelola Jadwal



Gambar 4.46 Tampilan Kelola Jadwal

Pada gambar di atas merupakan sebuah gambar rancangan antarmuka pada *platform* sewa sarana olahraga, dimana antamuka tersebut memaparkan tentang pengelolaad transaksi yang sudah terjadi yang bisa di lakukan oleh pengelola lapangan selaku *administrator*, transaksi-transaksi tersebut berupa daftar yang masing-masing daftarnya adalah item yamh berisikan informasi transaksi. Adapun komponen-komponen *view* yang bisa terdapat dalam aplikasi tersebut adalah *button*, *textview*, dan *imageview*.

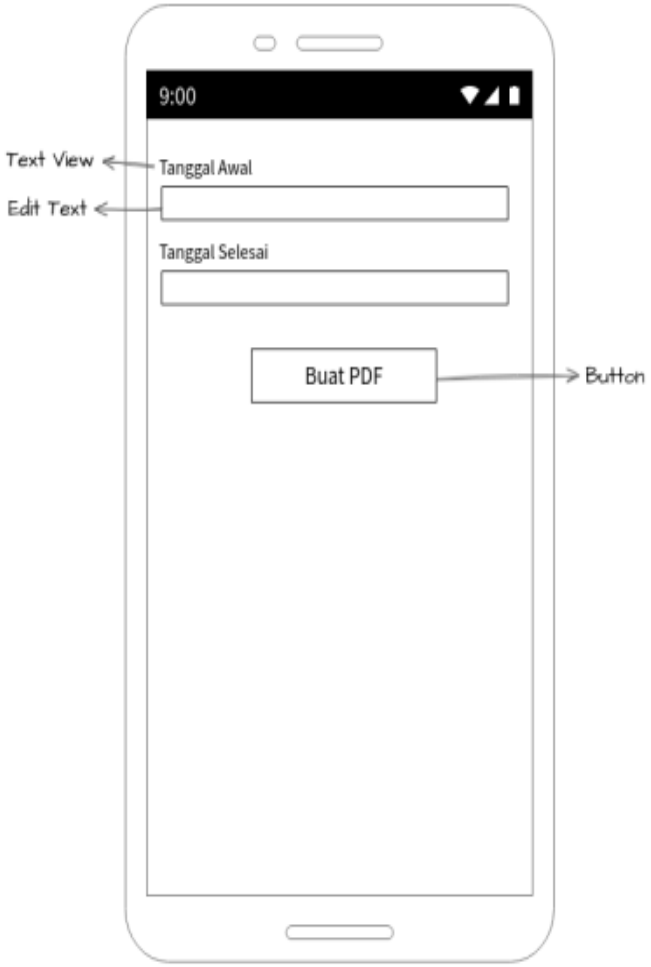
e) Halaman Saldo



Gambar 4.47 Tampilan kelola saldo

Pada gambar di atas merupakan sebuah gambar rancangan antarmuka pada *platform* sewa sarana olahraga, dimana antamuka tersebut memaparkan tentang pengelolaan saldo yang bisa di lakukan oleh pengelola lapangan selaku *administrator*, dalam antarmuka tersebut *user* bisa melihat daftar yang berupa item dari proses pengisian saldo yang berhasil dan dapat melakukan proses *scan qr code*. Adapun komponen-komponen *view* yang bisa terdapat dalam aplikasi tersebut adalah *button*, *textview*, dan *imageview*.

f) Halaman Laporan / Manajemen Keuangan



Gambar 4.48 Tampilan Manajemen Keuangan

Pada gambar di atas merupakan sebuah gambar rancangan antarmuka pada *platform* sewa sarana olahraga, dimana antamuka tersebut memaparkan tentang manajemen keuangan yang dapat dilakukan oleh pengelola lapangan selaku *administrator*, pada antarmuka ini dapat di lihat user bisa melakukan set tanggal awal dan akhir lalu memprosesnya hingga hasil dari laporan keuangan sesuai tanggal yang di inputkan oleh pengguna. Adapun komponen-komponen *view* yang terdapat dalam aplikasi tersebut atau rancangan di atas adalah *button*, *textview*, dan *imageview*.

4.3. *Implementation*

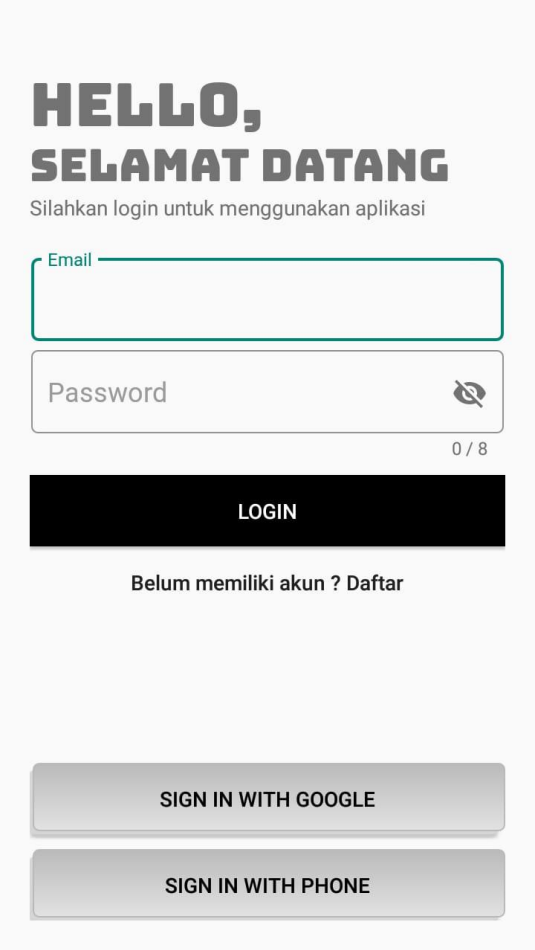
4.3.1. Implementasi rancangan antarmuka disisi *client*

1. Tampilan *splash screen*

Gambar 4.49 : Implementasi tampilan *splash screen*

Gambar di atas merupakan sebauh gambar implementasi antarmuka yang di sebut dengan *splash screen* di mana ini merupakan antarmuka atau *interface* yang di lihat user pertama kali sebelum masuk ke halaman *login* dan lainnya. Pada *Splash Screen* ini dapat di lihat dua komponen view yang tertera yaitu *Text View* dan *Image View*, antarmuka ini nantinya hanya akan berjalan selama tiga detik yang di *handle* oleh layar belakang sistem, dan selama tiga detik ini pula beberapa aktifitas terkait kebutuhan sistem bisa dilakukan seperti melakukan atentikasi akun pengguna di balik layar, dan melakukan autentukasi *cloud messaging*.

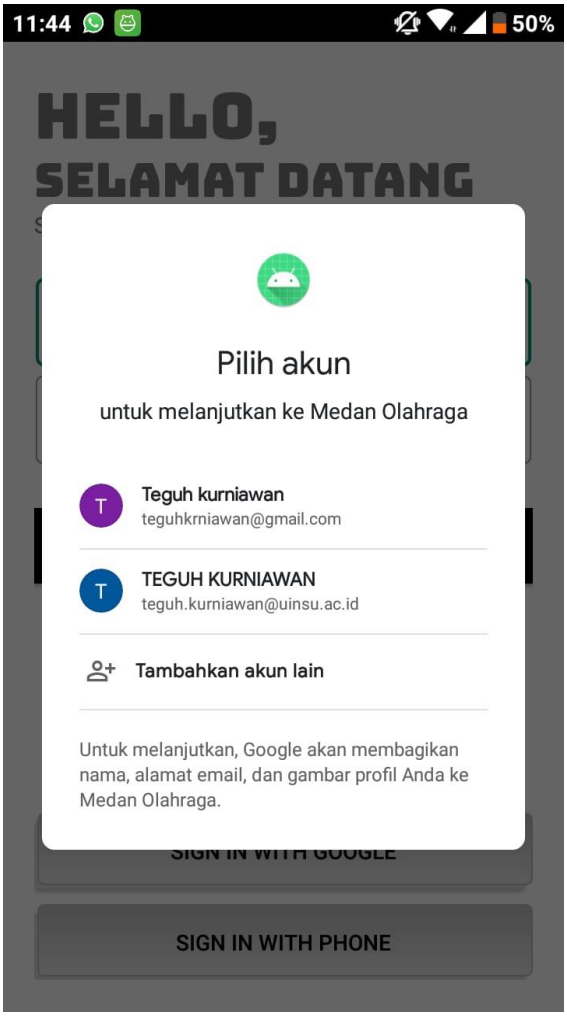
2. Tampilan halaman *login*



Gambar 4.50 Implementasi tampilan *login*

Pada gambar di atas merupakan sebuah gambar rancangan antarnuka di *platform* sewa sarana olahraga yang menjelaskan tentang sebuah aktifitas untuk melakukan kegiatan *login* dimana pada rancangan antarmuka tersebut dapat di lihat komponen-komponen *view* dari *android* seperti *Textview, Edittext,* dan *Button*. Masing-masing komponen *view* ini memiliki peran-nya masing-masing untuk dapat melakukan dan membantu *user* dalam melakukan proses kegiatan *login* ke dalam *platform* sewa sarana olahraga.

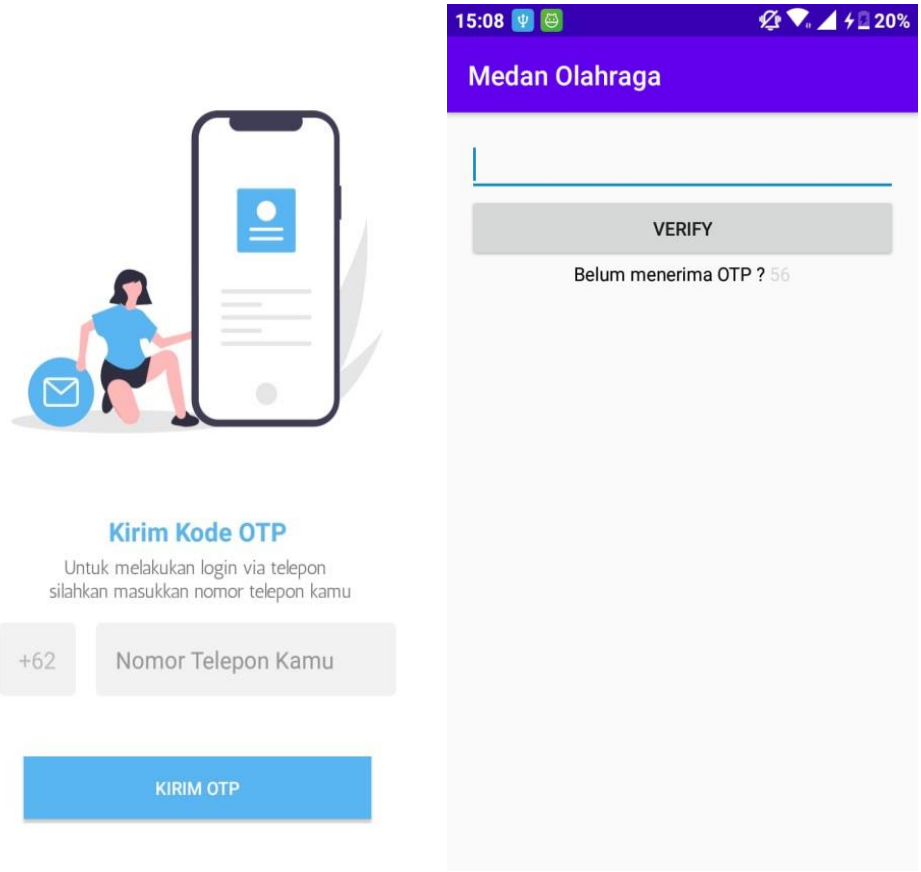
3. Tampilan halaman login dengan *google account*



Gambar 4.51 Implementasi *login* via *google account*

Pada gambar di atas merupakan sebuah gambar impelementasi antarmuka pada *platform* sewa sarana olahraga, dimana antamuka tersebut memaparkan tentang *login* ke dalam sistem menggunakan *Google Account* yang tertataut pada *smartphone* atau perangkat *user*, rancangan di atas bisa di bilang berbentuk *pop up* di mana dalam *pop up* tersebut terdapat bentuk daftar dari akun-akun *google account* yang sudah terdaftar dalam perangkat *user*, untuk melakukan proses autentikasi.

4. Tampilan *login* via nomor telepon



Gambar 4.52 *Input* nomor telepon dan verifikasi OTP

Pada gambar di atas merupakan sebuah gambar rancangan antarmuka pada *platform* sewa sarana olahraga, dimana antamuka tersebut memaparkan tentang verifikasi *login* ke dalam sistem menggunakan nomor telepon, di aman dalam rancangan ini sebuah aktifitas dari kode *One Time Password* sudah di kirimkan oleh server atau sistem melalui layanan SMS, dan rancangan ini berfungsi agar pengguna melakukan *inputan* dari kode tersebut dalam *field* atau *edittext* pada rancangan antarmuka tersebut. Kemudian pengguna dapat melakukan proses verifikasi dengan melakukan inpurtan kode otp tersebut ke dalam *field* yang telah di sediakan oleh sistem.

5. Tampilan *Home* (Beranda)

Gambar 4.53 Tampilan *home* pada aplikasi *client*

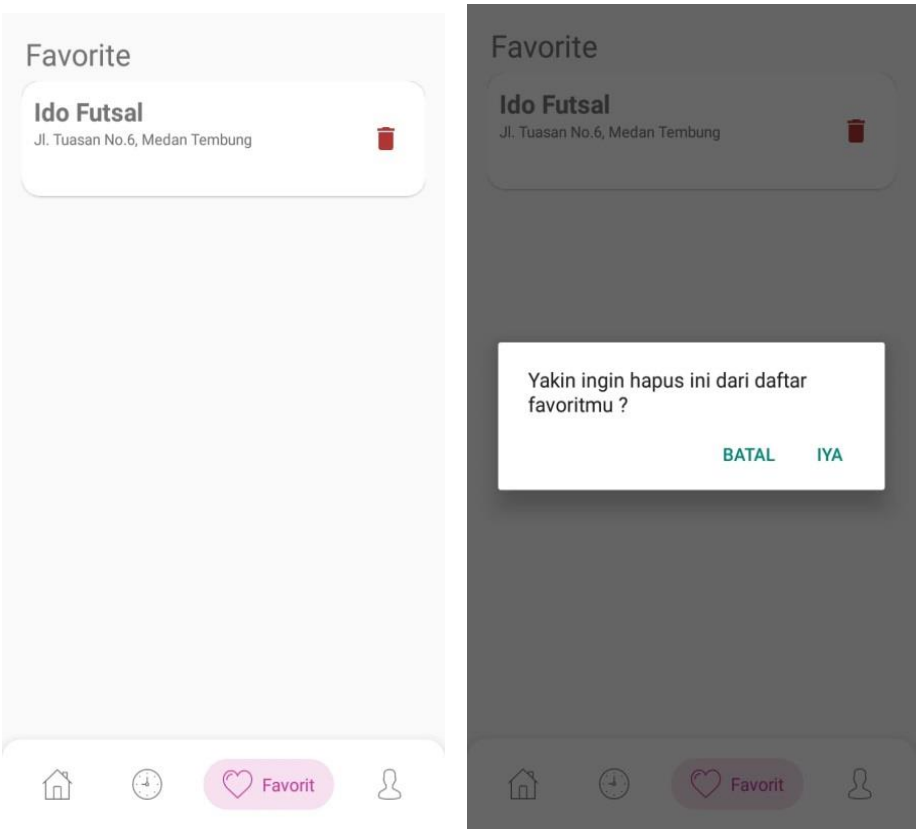
Pada gambar di atas merupakan sebuah gambar implementasi antarmuka pada *platform* sewa sarana olahraga, dimana antamuka tersebut memaparkan tentang antarmuka halaman utama atau biasa di sebut dengan *dashboard*. Dalam *dashboard* ini pengguna dapat melihat dan memilih kategori olahraga yang sarana olahraganya ingin mereka gunakan dan secara fisik *user* dapat melihat kategori-kategori olahraga ini dalam bentuk daftar. Dalam rancangan antarmuka menu *dashboard* atau halaman utama ini terdapat komponen view seperti *textview*, dan *cardview* yang disusun sedemikian rupa hingga berbentuk seperti yang ada pada gambar di atas.

6. Tampilan Histori

Gambar 4.54 Tampilan histori reservasi

Pada gambar di atas merupakan sebuah gambar implementasi antarmuka pada *platform* sewa sarana olahraga, dimana antamuka tersebut memaparkan tentang histori atas transaksi yang pernah di lakukan oleh *user*, singkatnya secara sederhana ini adalah antarmuka berupa daftar yang berisikan informasi lapangan saja yang pernah di sewa oleh pengguna selaku penyewa, dalam daftar tersebut terdapat informasi berupa nama lapangan dan alamat lapangan dan juga status dari pemesanan yang di lakukan oleh pengguna selaku penyewa atas transaksi yang telah di lakukan sebelumnya.

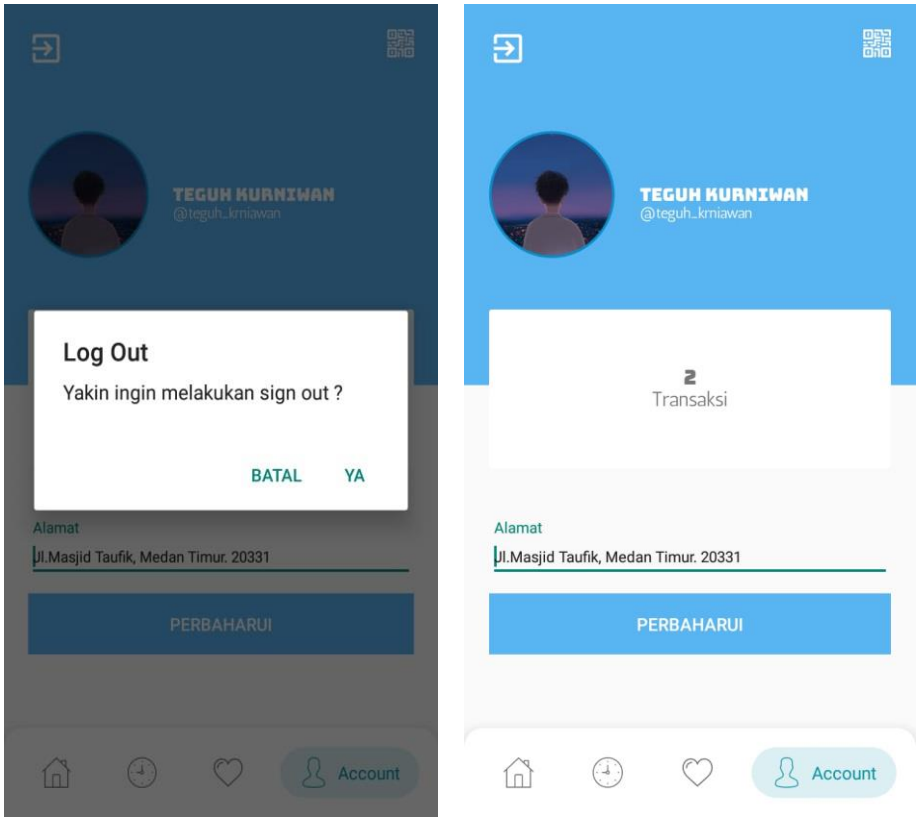
7. Tampilan *Favorite*



Gambar 4.55 Tampilan *favorite* dan dialog ketika hapus

Pada gambar di atas merupakan sebuah gambar impelentasi atau penerapan dari antarmuka pada *platform* sewa sarana olahraga, dimana antamuka tersebut memaparkan tentang daftar favorit lapangan yang telah di tambahkan user dalam hal ini adalah calon penyewa pada halaman detail lapangan, dalam rancangan antarmuka ini terdapat sebuah daftar yang di mana masing-masing dari daftar tersebut berupa infomasi seperti nama lapangan, alamat lapangan, dan sebuah *button* berbentuk *icon trash* yang berfungsi untuk mengahapus item lapangan yang telah di favoritkan dari daftar favorit pengguna atau *user*, dan terdapat alert dialog konfirmasi yang menanyakan apaka user yakin ingin menghapus item tersebut sebelum di eksekusi atau di hapus.

8. Tampilan *Account*



Gambar 4.56 Tampilan *account* dan *log out*

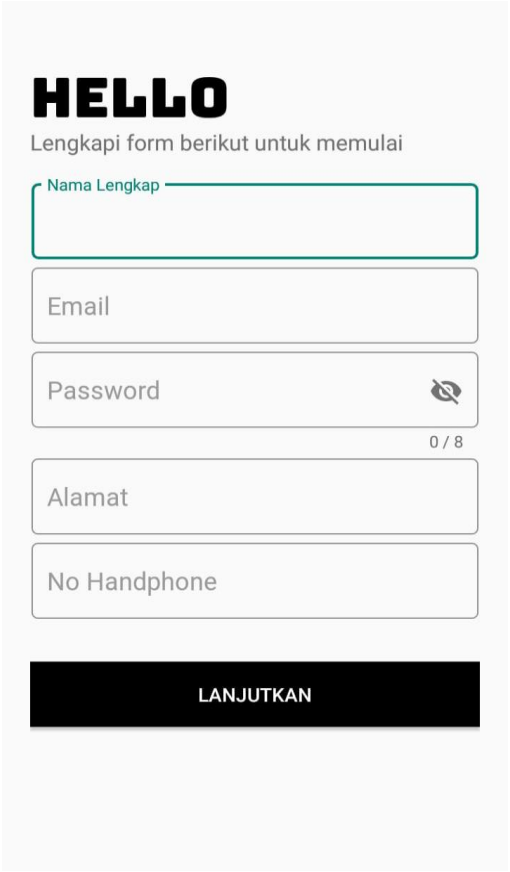
Pada gambar di atas merupakan sebuah gambar implementasu atau penerapan dari antarmuka pada *platform* sewa sarana olahraga, di mana antamuka tersebut memaparkan tentang informasi akun pengguna, di mana dalam rancangan antarmuka ini user bisa melakukan *generate qr code*, mengubah data diri, informasi seperti nama dan proses *logout* apabila user melakukan aksi pada *button logout*, komponen view pada rancangan antarmuka ini ada *edittext, button, imageview*, dan juga *cardview*. Dan dari gambar di atas juga terlihat ada pop up apa konfirmasi berupa alert apabila user ingin melakukan *log out*, alert tersebut meberikan peringatan apakah user yakin ingin melakukan *logout* atau tidak.

9. Tampilan QR Code

Gambar 4.57 Tampilan *qr code* user saat isi ulang voucher

Pada gambar di atas merupakan sebuah gambar implementasi antarmuka pada *platform* sewa sarana olahraga, dimana antamuka tersebut memaparkan tentang informasi akun pengguna, di mana dalam rancangan antarmuka ini user bisa melakukan *generate qr code* dan *qr code* ini digunakan oleh pengguna dalam hal ini calon penyewa untuk melakukan pengisian ulang saldo voucher di akun nya agar dapat menggunakan fitur metode bayar dengan pilihan metode bayar voucher komponen view pada rancangan antarmuka ini ada *edittext, button, imageview,* dan juga *cardview*.

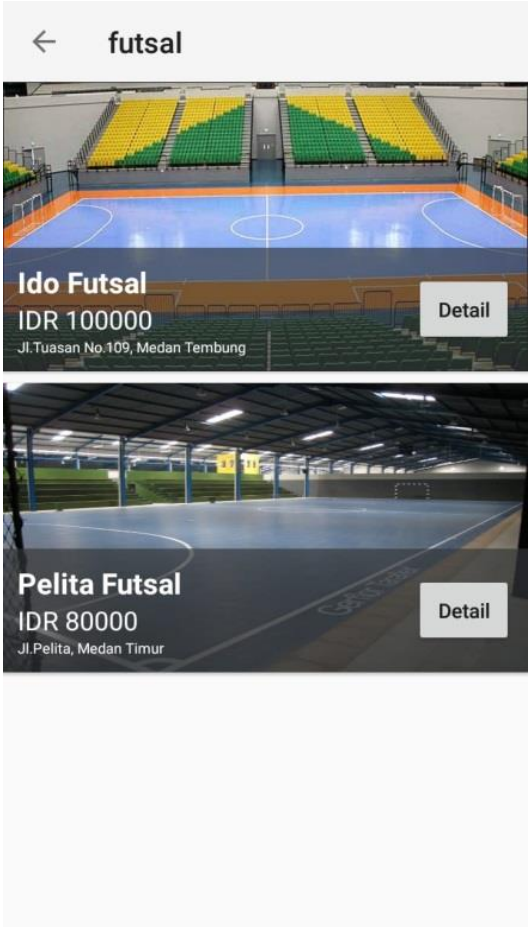
10. Tampilan *register account*



Gambar 4.58 Tampilan *register account*

Pada gambar di atas merupakan sebuah gambar implementasi antarmuka pada *platform* sewa sarana olahraga, dimana antamuka tersebut memaparkan tentang informasi akun pengguna di mana dalam rancangan antarmuka ini *user* di haruskan untuk melengkapi data diri yang dibutuhkan dengan melakukan *inputan* data ke dalam *field-field* yang sudah di sediakan daam *form*. Adapun data diri yang terdapat dalam rancangan antarmuka ini adalah nama lengkap, *email, password,* alamat pengguna, dan nomor telepon. Adapun komponen view pada rancangan ini adalah *textview*, *edittext*, dan *button*.

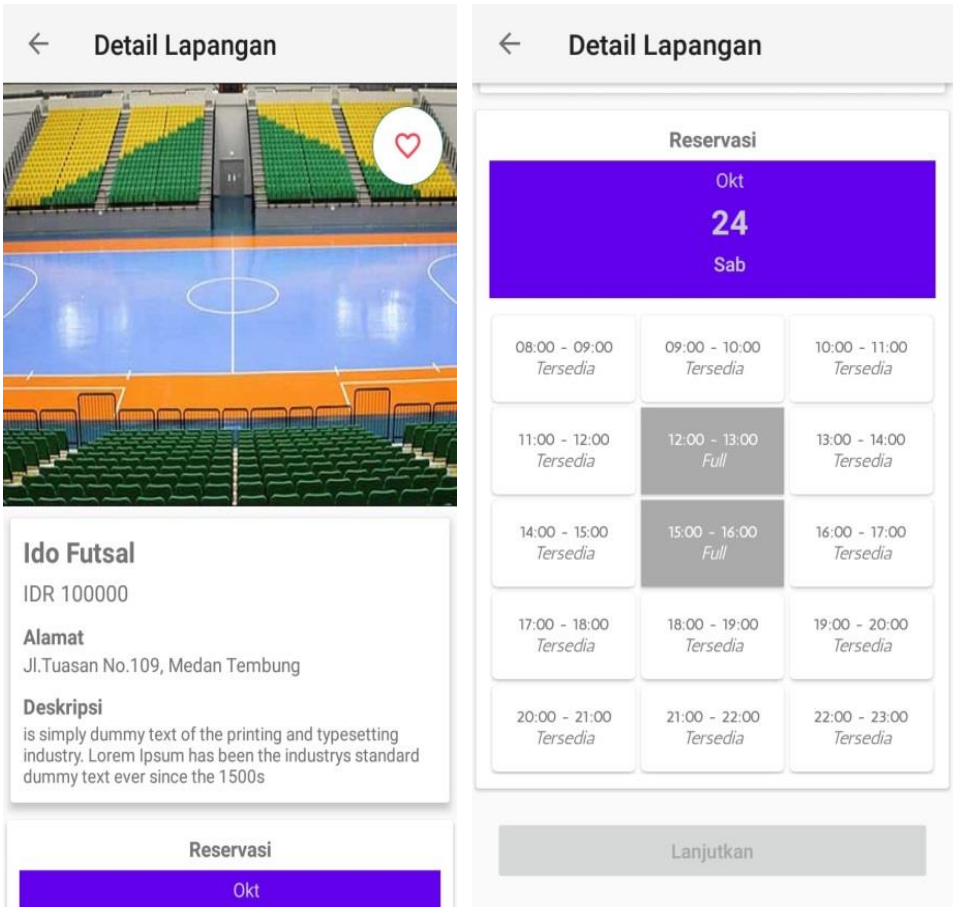
11. Tampilan *list* lapangan



Gambar 4.59 Tampilan *list* lapangan dari kategori yang dipilh

Pada gambar di atas merupakan sebuah gambar implementasi atau penerapan antarmuka pada *platform* sewa sarana olahraga, dimana antamuka tersebut memaparkan tentang daftar lapangan setelah *user* memilih kategori olahraga mana yang ingin digunakan dalam proses sewa menyewa tersebut, dalam rancangan dapat di lihat bahwa ada komponen-komponen *view* seperti *textview* yang mana *view* ini digunakan untuk menampilkan informasi seperti nama lapangan yang di sewakan oleh pihak pengelola lapangan dan menampikan informasi harga setiap jam par transaksi.

12. Tampilan detail lapangan



Gambar 4.60 Detail lapangan dan info jadwal *real time*

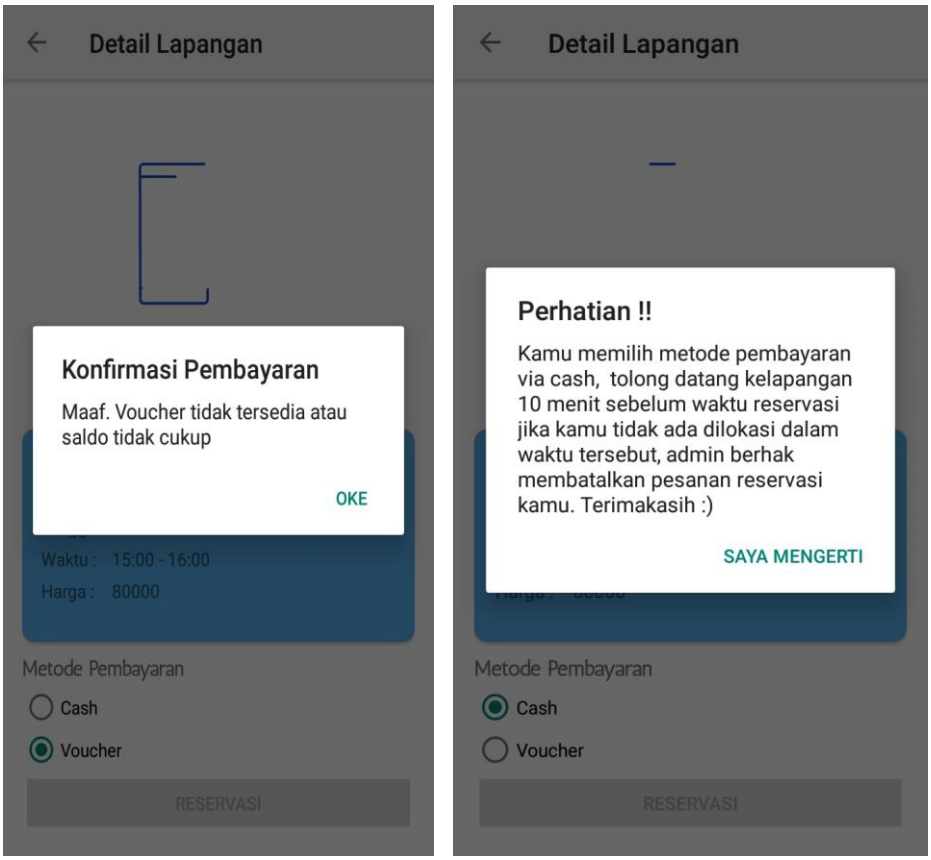
Pada gambar di atas merupakan sebuah gambar implementasi atau penerapan antarmuka pada *platform* sewa sarana olahraga, dimana antamuka tersebut memaparkan tentang detail lapangan setelah user memilih lapangan yang di inginkan dan dibutuhkan, pada antarmuka tersebut user bisa mengetahui infotmasi berupa foto dari lapangan yang akan di sewakan, informasi berupa nama lapangan, harga sewa lapangan, deskripsi lengkap tentang lapangan dan informasi jadwal yang tersedia, user juga dapat memilih waktu dan tanggal reservasi sesuai keinginan calon penyewa.

13. Tampilan konfirmasi pembayaran

Gambar 4.61 Tampilan konfirmasi pembayaran

Pada gambar di atas merupakan sebuah gambar implementasi atau peenrapan antarmuka pada *platform* sewa sarana olahraga, dimana antamuka tersebut memaparkan tentang konfirmasi lapangan atas transaksi sewa menyewa setelah *user* memilih jadwal yang tersedia sesuai kebutuha dari user selaku calon penyewa, dalam implementasi tersebut terdapat informasi berupa ringkasan terhadap data lapangan yang akan di sewa, dan pemilihan metode pembayaran yang terdiri dari metode pembayaran *cash* dan metode pembayaran voucher, baru setelah user memilih salah satu metode pembayaran tersebut barulah bisa di lakukan konfirmasi bahwasanya proses transaksi ini di setujui oleh user sebagai calon penyewa

14. Tampilan *dialog* metode pambayaran

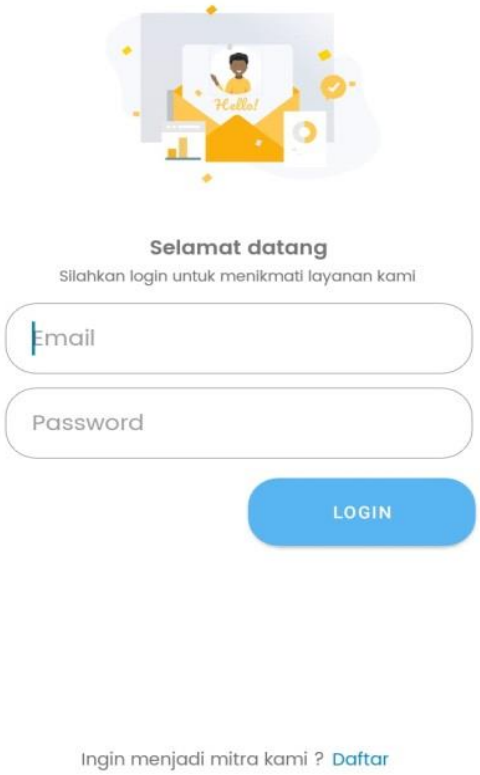


Gambar 4.62 *dialog* ketika tap *radio cash* atau *voucher*

Pada gambar di atas merupakan sebuah gambar implementasi antarmuka pada *platform* sewa sarana olahraga, dimana antamuka tersebut memaparkan tentang konfirmasi lapangan atas transaksi sewa menyewa setelah *user* memilih jadwal dan metode bayar yang tersedia sesuai kebutuhan dari user selaku calon penyewa, dalam implementasi tersebut terdapat akan muncu sebuah alert dialog setelah memilih metode *cash* dan metode pembayaran voucher, baru setelah user memilih salah satu metode pembayaran tersebut barulah bisa di lakukan konfirmasi bahwasanya proses transaksi ini di setujui oleh user sebagai calon penyewa

4.3.2. Implementasi rancangan antarmuka disisi *administrator*

1. Tampilan *login account*



Gambar 4.63 Tampilan *login* pada aplikasi di sisi *administrator*

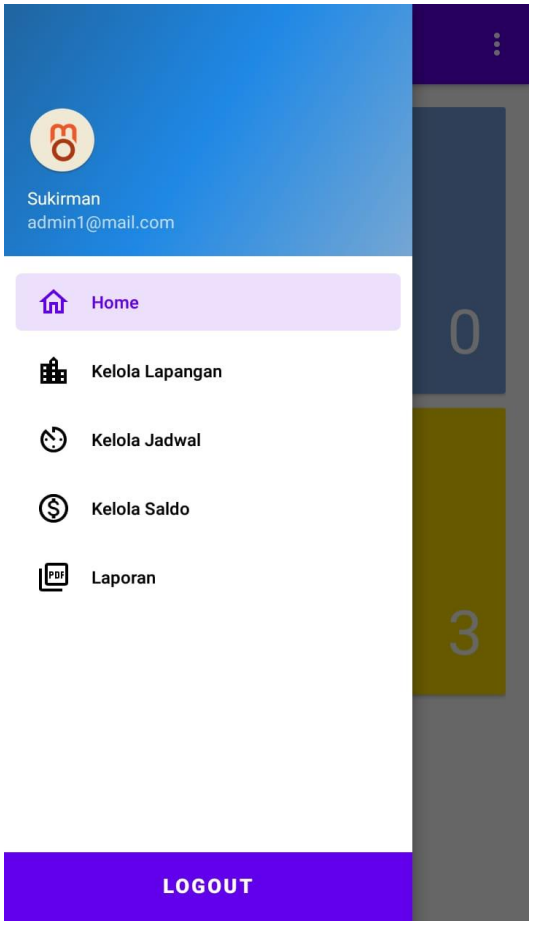
Pada gambar di atas merupakan sebuah gambar implementasi antarnuka di *platform* sewa sarana olahraga yang menjelaskan tentang sebuah aktifitas untuk melakukan kegiatan *login* dimana pada rancangan antarmuka tersebut dapat di lihat komponen-komponen *view* dari *android* seperti *Textview*, *Edittext*, dan *Button*. Masing-masing komponen *view* ini memiliki peran-nya masing-masing untuk dapat melakukan dan membantu *user* dalam melakukan proses kegiatan *login* ke dalam *platform* sewa sarana olahraga.

2. Tampilan daftar/*register account*

Gambar 4.64 Tampilan *register account*

Pada gambar di atas merupakan sebuah gambar implementasi antarmuka pada *platform* sewa sarana olahraga, dimana antamuka tersebut memaparkan tentang informasi akun pengguna di mana dalam rancangan antarmuka ini *user* di haruskan untuk melengkapi data diri yang dibutuhkan dengan melakukan *inputan* data ke dalam *field-field* yang sudah di sediakan daam *form*. Adapun data diri yang terdapat dalam rancangan antarmuka ini adalah nama lengkap, *email*, *password*, alamat pengguna, dan nomor telepon. Adapun komponen view pada rancangan ini adalah *textview*, *edittext*, dan *button*.

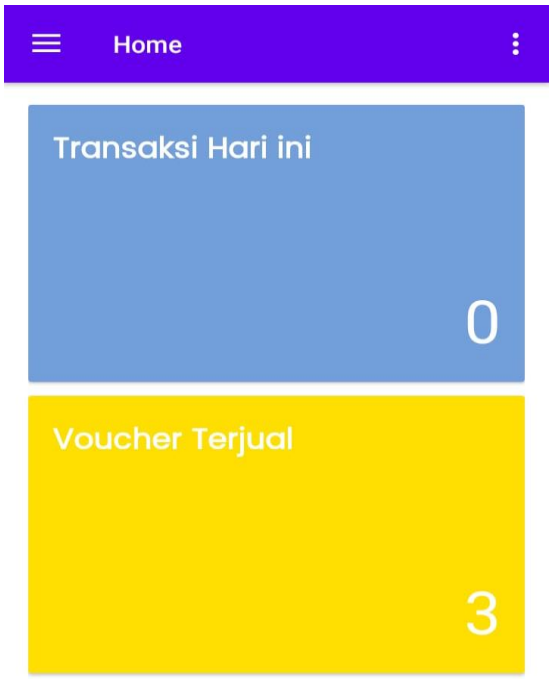
3. Tampilan navigasi menu *administrator*



Gambar 4.65 Navigasi menu

Pada gambar di atas merupakan sebuah gambar implementasi antarmuka pada *platform* sewa sarana olahraga, dimana antamuka tersebut memaparkan tentang menu apa saja yang bisa di dapatkan oleh pengelola lapangan selaku *administrator*, menu-menu tersebut di antaranya adalah *home* atau halaman utama, kelola jadwal, kelola lapangan, kelola saldo, dan juga laporan, serta *sign out* yang berfungsi apabila pengguna ingin mengeluarkan akun tersebut dari aplikasi. Adapun komponen-komponen *view* yang bisa terdapat dalam aplikasi tersebut adalah *button*, *textview*, dan *imageview*.

4. Tampilan *home administrator*



Gambar 4.66 Tampilan home administrator

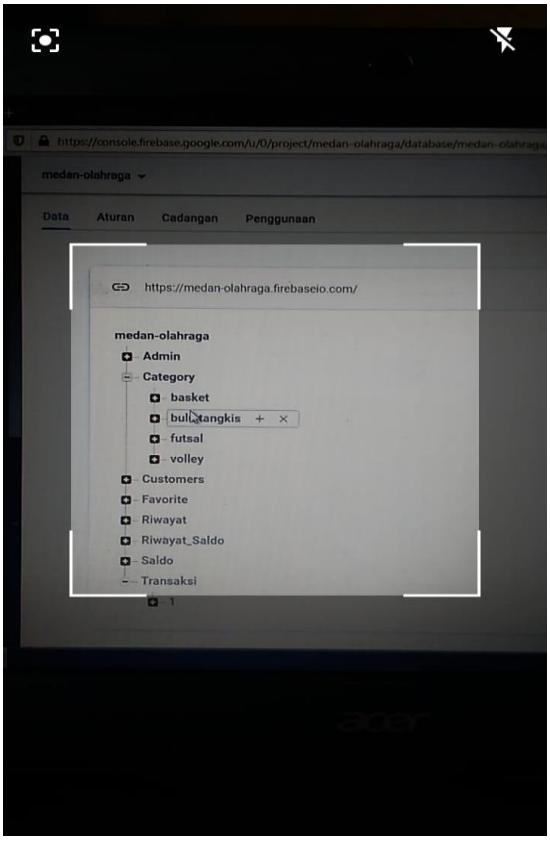
Pada gambar di atas merupakan sebuah gambar implementasi antarmuka pada *platform* sewa sarana olahraga, dimana antamuka tersebut memaparkan tentang antarmuka halaman utama atau biasa di sebut dengan *dashboard*. Dalam *dashboard* ini pengguna dapat melihat dan memilih kategori olahraga yang sarana olahraganya ingin mereka gunakan dan secara fisik *user* dapat melihat kategori-kategori olahraga ini dalam bentuk daftar. Dalam rancangan antarmuka menu *dashboard* atau halaman utama ini terdapat komponen view seperti *textview*, dan *cardview* yang disusun sedemikian rupa hingga berbentuk seperti yang ada pada gambar di atas.

5. Tampilan kelola saldo *administrator*

Gambar 4.67 Tampilan informasi saldo di menu kelola saldo

Pada gambar di atas merupakan sebuah gambar implemetasi antarmuka pada *platform* sewa sarana olahraga, dimana antamuka tersebut memaparkan tentang pengelolaan saldo yang bisa di lakukan oleh pengelola lapangan selaku *administrator*, dalam antarmuka tersebut *user* bisa melihat daftar yang berupa item dari proses pengisian saldo yang berhasil dan dapat melakukan proses *scan qr code*. Adapun komponen-komponen *view* yang bisa terdapat dalam aplikasi tersebut adalah *button*, *textview,* dan *imageview*.

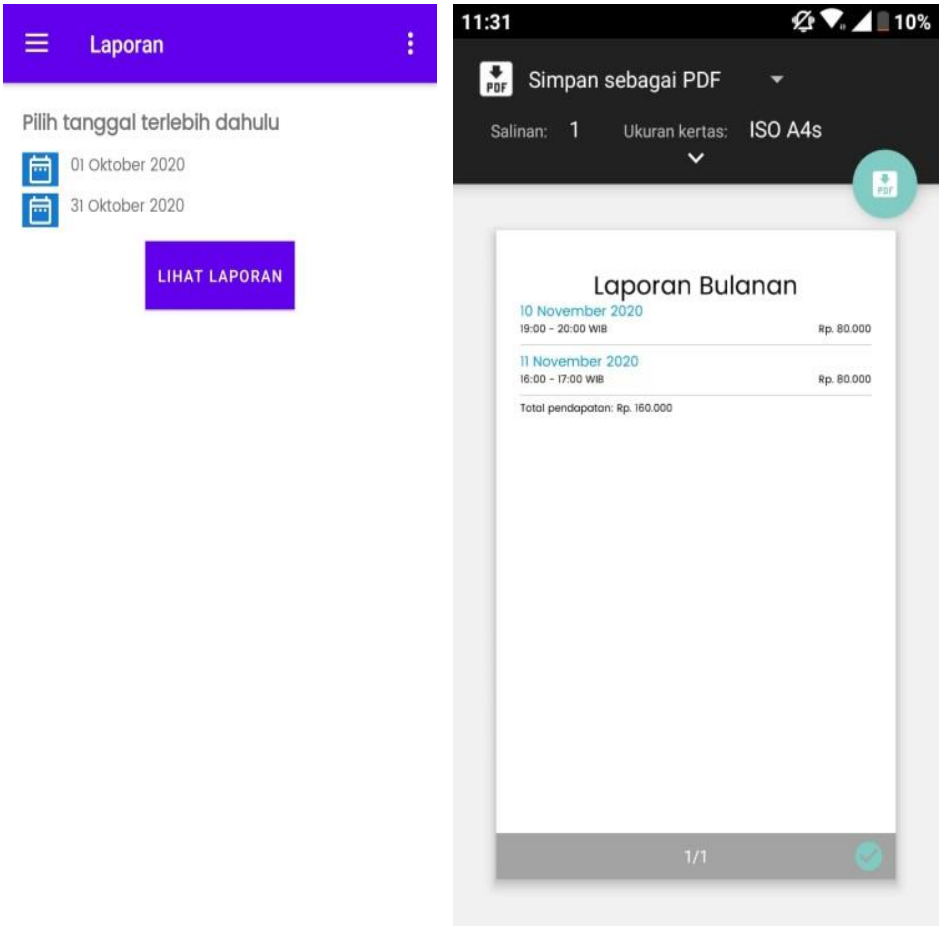
6. Tampilan *scan QR CODE administrator*



Gambar 4.68 Tampilan scan qr code pelanggan untuk isi ulang voucher

Pada gambar di atas merupakan sebuah gambar implementasi antarmuka pada *platform* sewa sarana olahraga, dimana antamuka tersebut memaparkan tentang pengelolaan saldo yang bisa di lakukan oleh pengelola lapangan selaku *administrator*, dalam antarmuka tersebut *user* bisa melihat daftar yang berupa item dari proses pengisian saldo yang berhasil dan dapat melakukan proses *scan qr code*. Adapun komponen-komponen *view* yang bisa terdapat dalam aplikasi tersebut adalah *button*, *textview,* dan *imageview*.

7. Tampilan laporan *administrator*



Gambar 4.69 Tampilan laporan penghasilan bulanan

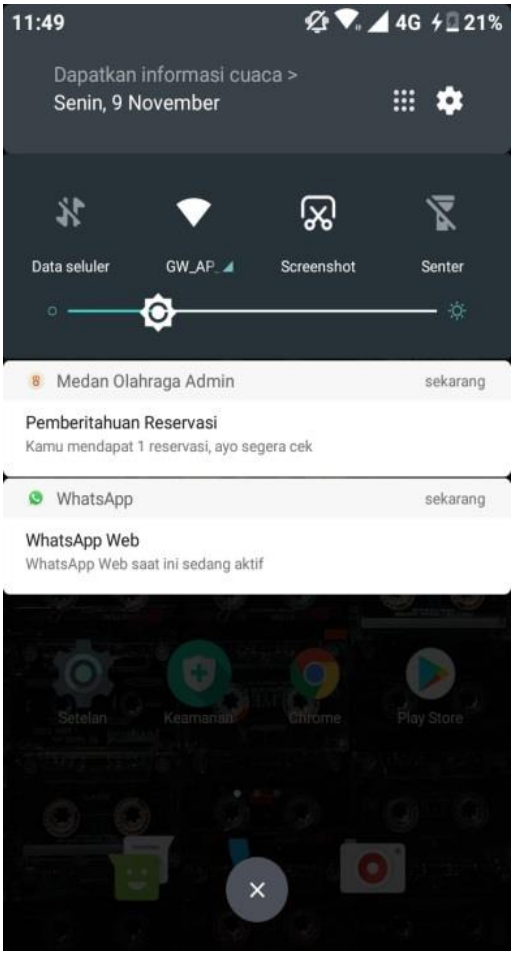
Pada gambar di atas merupakan sebuah gambar imlementasi antarmuka pada *platform* sewa sarana olahraga, dimana antamuka tersebut memaparkan tentang pengelolaan keuangan yang bisa di lakukan oleh pengelola lapangan selaku *administrator*, dalam antarmuka tersebut *user* bisa melakukn set tanggal untuk melakukan pengecekan keuangan. Adapun komponen-komponen *view* yang bisa terdapat dalam aplikasi tersebut adalah *button, textview*.

8. Tampilan Pengelolaan Jadwal

Gambar 4.70 Tampilan pengelolaan jadwal

Pada gambar di atas merupakan sebuah gambar implementasi antarmuka pada *platform* sewa sarana olahraga, dimana antamuka tersebut memaparkan tentang pengelolaad transaksi yang sudah terjadi yang bisa di lakukan oleh pengelola lapangan selaku *administrator*, transaksi-transaksi tersebut berupa daftar yang masing-masing daftarnya adalah item yamh berisikan informasi transaksi. Adapun komponen-komponen *view* yang bisa terdapat dalam aplikasi tersebut adalah *button*, *textview*, dan *imageview*.

9. Tampilan notifikasi



Gambar 4.71 Tampilan notifikasi ketika ada reservasi

Pada gambar di atas merupakan sebuah gambar implementasi antarmuka pada *platform* sewa sarana olahraga, dimana antamuka tersebut memaparkan tentang notifikasi apabila ada pesanan masuk yang bisa di lihat oleh pengelola lapangan selaku *administrator*, transaksi-transaksi tersebutakan masuk secara realtime ke perangkat pengguna sebagai notifikasi. Adapun komponen-komponen *view* yang bisa terdapat dalam aplikasi tersebut adalah *textview*.

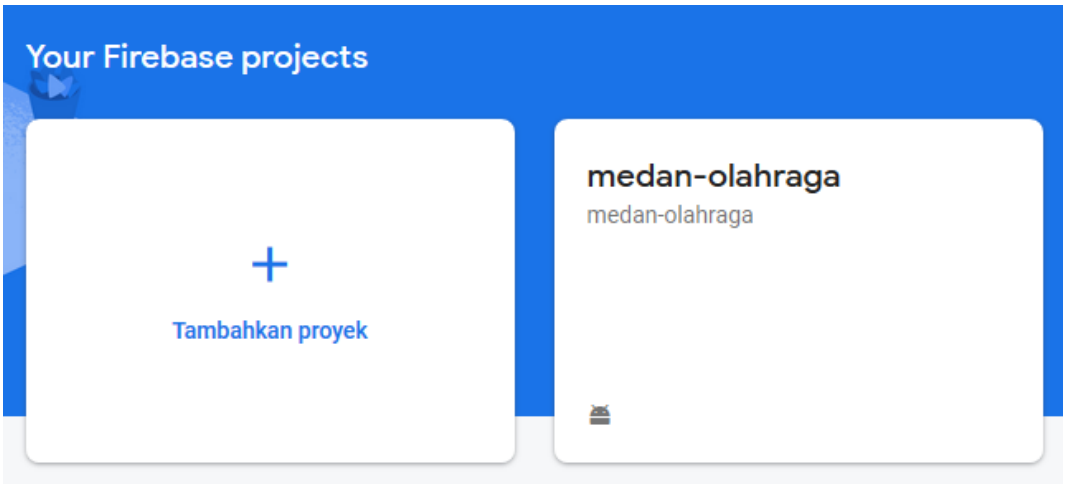
4.3.3. Implementasi *Firebase Auth*

Pada sub-bab ini akan dibahas tentang bagaimana menerapkan salah satu layanan *firebase cloud* di aplikasi *android secara native*. *Firebase* menyediakan *resource authentication* cukup banyak, mengambil data API dari perusahaan-perusahaaan besar didunia, seperti *google*, *facebook, github, twitter, microsoft* dan lain sebagainya. Dalam pengembangan aplikasi tentu hal ini akan memudahkan seorang *software developer* dalam menangani kasus autentiksi dari bermacam-macam *resource* perusahaan besar didunia. Misalkan saja ketika ingin membangun sebuah aplikasi terintergasi dengan akun google, tentu dalam arsitektur pegembangannya aplikasi ini membutuhkan API dari google yang mana aplikasi sudah didaftarkan terlebih dahulu, tidak masalah apabila hanya satu *resource* mungkin pemngembang belum kesulitan, tapi bayangkan jika aplikasi ini membutuhkan lebih dari satu *resource,* maka akan menjadi hal yang merepotkan untuk mendapatkan dan mendaftarkan aplikasi kita dimasing-masing *resource* tersebut.

Selain kasus diatas *autentikasi* juga bisa dilakukan dengan cara *one time password* (OTP) melalui *sms gateway* yang dikirimkan pada nomor telepon. Umumnya *sms gateway* untuk *one time password* dibuat menggunakan *websocket* dan dilempar sebuah *web service*. Hal ini juga akan memakan waktu apabila membangun sebuah *push notification* dengan *websocket* dalam pengembangan aplikasi tersebut.

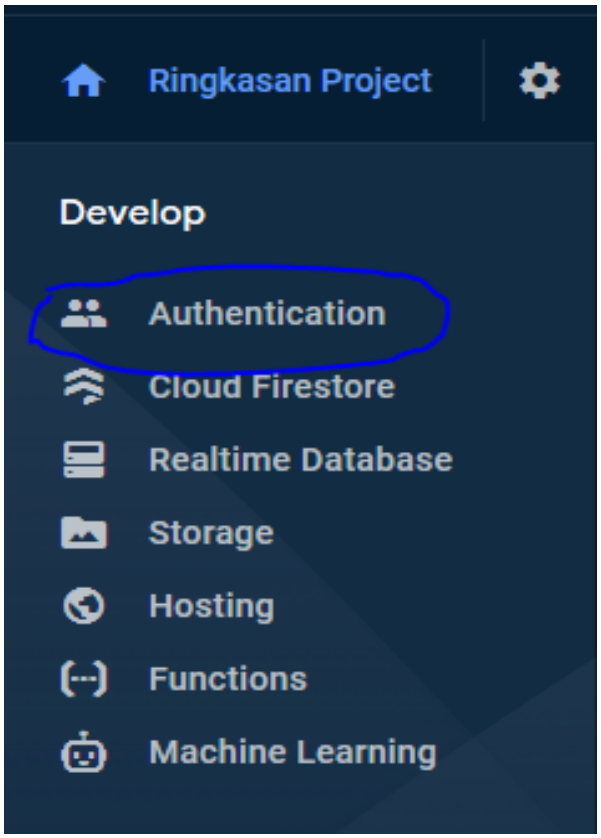
Layanan *firebase authetication* akan memudahkan proses-proses yang sudah dipaparkan diatas, tanpa memikirkan arsitektur sistem yang rumit tersebut. Yaitu dengan mengintegrasikan aplikasi android native kita dengan layanan *firebase auth*. Berikut ini adalah pemaparan secara lengkap bagaimana proses layanan *firebase authentication* ini dibuat.

1. Pendaftaran *google account* ke layanan *firebase*. Setelah akun terdaftar artinya *firebase* akan memberikan akses kepada *pengembang* (*software developer*) bahwa bisa membuat satu *project environment* untuk aplikasinya. Dalam kasus kali ini *project firebase* di integrasikan dengan aplikasi *android*.



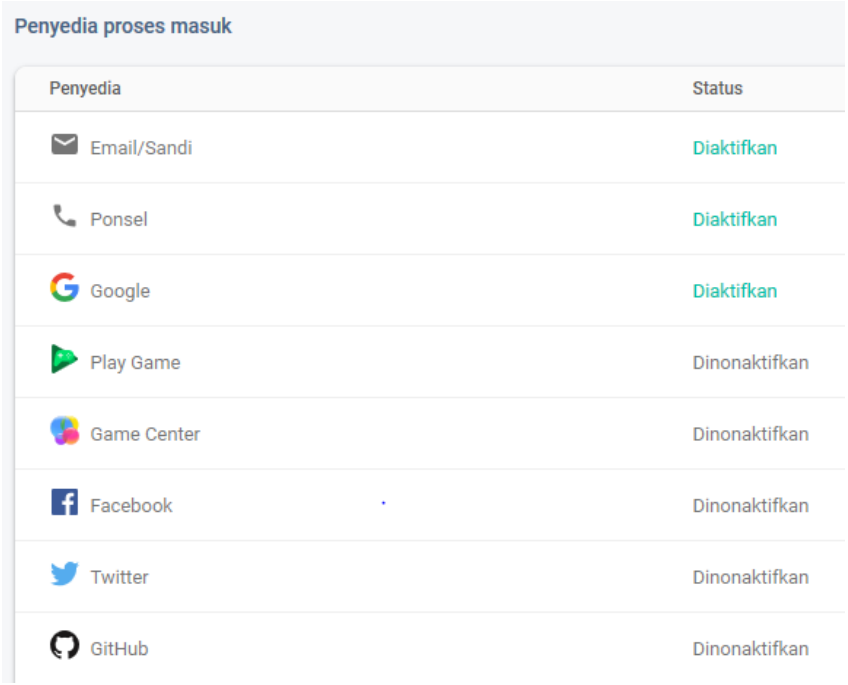
Gambar 4.72 Halaman *project enviromnet* dari *firebase*

Kemudian dapat diklik dari project yang telah dibuat tadi dan akan menampilkan menu seperti gambar berikut :



Gambar 4.73 Menu dari *project* layanan *firebase*

2. Mengaktifkan layanan autentikasi yang akan dipilih, dalam penelitian ini autentikasi yang akan dipakai adalah *email*, nomor telepon dan *google account*



Gambar 4.74 Layanan penyedia autentikasi *firebase*

3. Setelah layanan yang dihendaki sudah diaktifkan, barulah dapat melakukan integrasi dengan aplikasi android, menggunakan *depedency java* melalui gradle. *Firebase* menyediakan SDK (*software development kit*) sehingga lebih dalam implementasinya *firebase* menyediakan *built in* yang bisa di *import* oleh sistem. Agar SDK ini dapat digunakan, integrasikan aplikasi android dengan library *firebase authentication* berikut :

```
Implementation'com.google.firebase:firebase-auth:17.0.0'
```

Setelah melakukan *sync gradle* pada *project apps*, akan ada *built in* yang mewakili dari masing-masing proses autentikasi, seperti yang dikatakan sebe-

lumnya bahwa autentikasi dalam penelitian ini ada tiga *resource* yaitu *google account, email,* dan nomor telepon seluluer. Dalam implementasi 3 layanan ini punya penerapan yang berbeda. Untuk lebih jelasnya bisa dilihat dari pemaparan dibawah ini.

4.3.3.1. Autentikasi menggunakan email

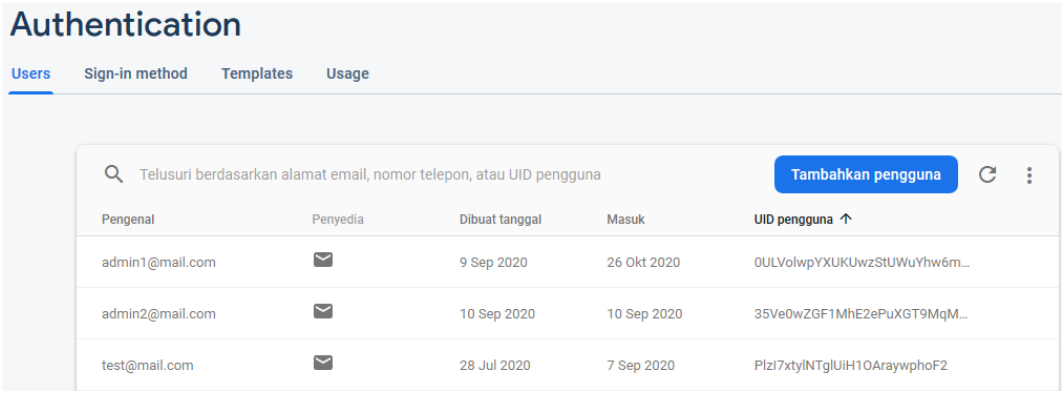
Untuk menggunakan autentikasi email *firebase,* didalam file *code* nya wajib melakukan sebuah *instance object* dari class FirebaseAuth.

```
private lateinit var mAuth: FirebaseAuth

// Insialisasi objek dari FirebaseAuth
mAuth = FirebaseAuth.getInstance();
```

Setelah melakukan *instance object,* menggunakan *built in* dari *sdk firebase* barulah bisa menggunakan *function* dari class FirebaseAuth untuk melakukan *registrasi* dan *login* akun menggunakan email.

```
mAuth.createUserWithEmailAndPassword(email, password)
            .addOnCompleteListener {task ->
                if (task.isSuccessful){
                    // Lakukan suatu proses logika
                } else {
                    Log.e("REGISTER", "storeToDatabase: "
+task.result.toString() )
                }
            }
```

Sintaks diatas adalah penerapan *function* dari *sdk firebase* yang dipakai untuk menerapkan pendaftaran akun ke layanan *firebase authentication* menggunakan *email* disertai *password,* **createUserWithEmailAndPassword()** membutukan dua parameter yaitu email dan password, yang mana nantinya email dan password ini dibutuhkan sebagai nilai/*value* yang akan didaftarkan dalam *firebase authentication,* sehingga pada *dashboard firebase* akan seperti berikut :



Gambar 4.75 Email yang sudah terdaftar di *firebase authentication*

Selanjutnya setelah email berhasil didaftarkan pada *firebase authentication* aplikasi berarti sudah bisa di integrasikan dengan *function login* dari *sdk firebase* dan implementasinya adalah sebagai berikut

```
mAuth.signInWithEmailAndPassword(email, password)
       .addOnCompleteListener { task ->
                if (task.isSuccessful){
                        Toast.makeText(this, "Selamat datang",
Toast.LENGTH_SHORT).show()
                    } else {
                        Toast.makeText(this, "Username atau
password salah", Toast.LENGTH_SHORT).show()
                    }
                }
```

Sintaks diatas merupakan sintaks dimana proses *login* dilakukan, yaitu dengan menerapkan *function* **signInWithEmailAndPassword()** dengan parameter yang dibutuhkan adalah email dan password yang telah didaftarkan sebelumnya.

Dapat dilihat dari kedua *function* yaitu **createUserWithAndEmail()** dan **siginInWithEmailAndPassword()** terdapat sebuah *function* dari *abstract class* yaitu **addCompleteListener** yang mana *function* ini memiliki sebuah nilai *callback* pada variabel *task* yang bisa digunakan untuk melakukan suatu pengecekan kondisi apakah berhasil atau tidak dalam melakukan pendaftaran akun menggunakan *email* dan *login*.

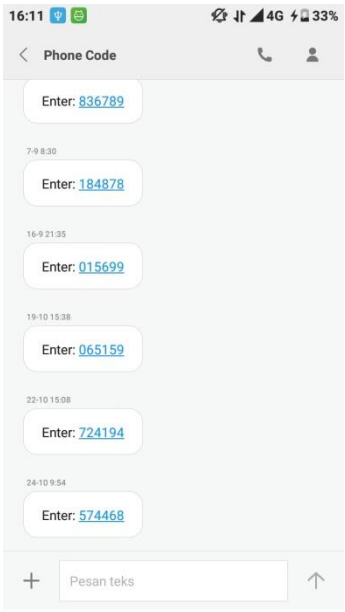
4.3.3.2. Autentikasi Dengan Nomor Telepon

Ini adalah salah satu fitur yang cukup efisien dari layanan firebase, tanpa mengeluarkan biaya dan minimalisir waktu untuk membangun infrasturktur sms gateway yang digunakan dalam melakukan keamanan login melalui one time password, dengan menggunakan firebase authentication hal ini dengan mudah dapat ditangani. Cara melakukan implementasinya sama dengan email, yaitu dengan menggunkan built in dari firebase authentication.

Data yang dibutuhkan dalam menggunakan layanan autentikasi dengan nomor telepon ini adalah kode negara telepon dan nomor telepon pengguna itu sendiri. Indonesia memliki kode nomor telepon negera +62 dan kode ini akan digabungkan dengan nomor telepon pengguna. Untuk melakukan proses pengiriman one time password melalui sms gateway dibutuhkan instace object dari class PhoneAuthProvider yang merupakan sebuah class built in dari bawaan SDK Firebase.

```
PhoneAuthProvider.getInstance().verifyPhoneNumber(
    phoneNumber, // nomor hp disertai kode +62
      60, // durasi aktif OTP
      TimeUnit.SECONDS, // waktu durasi dalam detik
      this, // context
      phoneAuthCallback // callback
)
```

Ketika kode tersebut dieksekusi maka akan otomatis mengirimkan kode OTP melalui sms ke nomor telepon kita melalui layanan SMS yang akan di terima oleh perangkat kita. SMS ini dikirimkan oleh server google melalui firebase cloud messaging langsung, dan tentu saja pesannya tidak bisa diubah karena sudah merupakan kebijakan pihak google agar tidak terjadi tindak kejahatan yang memanfaatkan kode OTP ini dam ini merupakan kebijakan google demi menjaga keamanan penggunaan firebase authentication menggunakan metode sign in with number phone, berikut adalah gambaran contoh dari kode one time password yang dikirimkan pleh pihak google melalui firebase autentication adalah sebagai berikut :



Gambar 4.76 Kode OTP yang dikirimkan *google*

kemudian setelah mendapatkan kode OTP lewat sms dari nomor telepon yang didaftarkan, umumnya kote OTP yang diberikan ini akan diverifikasi secara sistem agar autentikasi berjalan secara sukses. Dan untuk melakukan verifikasi tersebut dapat mengguakan sintaks atau *built in sdk firebase* berikut

```
val credential = PhoneAuthProvid-
er.getCredential(storedVerificationId!!, edtOtp.text.toString())
signInWithPhoneAuthCredential(credential)

private fun signInWithPhoneAuthCredential(credential:
PhoneAuthCredential) {
        firebaseAuth.signInWithCredential(credential)
            .addOnCompleteListener(this) { task ->
                if (task.isSuccessful) {
                    Toast.makeText(this, "Sukses",
Toast.LENGTH_SHORT).show()
                } else {
                    Toast.makeText(this, "Gagal",
Toast.LENGTH_SHORT).show()
                    }
                }
            }
```

Pada sintaks di atas terdapat *function* **getCredential(string, string)** yang mana function ini akan melakukan task untuk melakukan pengecekan apakah kode OTP yang di inputkan pengguna dalam sistem sama atau tidak dengan kode OTP yang dikirimkan, parameter yang dibutuhkan pada function ini adalah kredential berupa nilai unik serta nomor telepon pengguna, function ini nantinya juga akan mengembalikan sebuah nilai yang digunakan pada saat *sign in/login*. Setelah medapatkan nilai *credential* yang ditampung pada *variabel crendetial* lakukan proses *sign-in autentikasi* dengan melemparkan *value credential* sebagai argumen dalam *function built-in* **signInWithCrendential.**

Selanjutnya kadang kala terjadi gangguan pada server google walaupun hal itu memliki peluang kecil untuk terjadi sehingga bisa saja kode OTP yang dikirimkan via SMS tersebut tidak masuk, hal ini tentunya bisa diatasi dengan cara meminta kembali kepada *server google* agar di kirimkan kembali kode OTP tersebut, cara implementasinya adalah sebagai berikut

```
phoneAuthCallback = object : PhoneAuthProvid-
er.OnVerificationStateChangedCallbacks() {

            override fun onVerificationCompleted(credential: PhoneAuthCreden-
tial) {
                Log.i("INFO", "onVerificationCompleted:$credential")
                //signInWithPhoneAuthCredential(credential)
            }

            override fun onVerificationFailed(e: FirebaseException) {
                // This callback is invoked in an invalid request for verifi-
cation is made,
                // for instance if the the phone number format is not valid.
                Log.w("Warning", "onVerificationFailed", e)

                if (e is FirebaseAuthInvalidCredentialsException) {
                    // Invalid request
                    // ...
                } else if (e is FirebaseTooManyRequestsException) {
                    // The SMS quota for the project has been exceeded
                    // ...
                }

            }

            override fun onCodeSent(verificationId: String, token:
PhoneAuthProvider.ForceResendingToken) {
                Log.i("INFO", "onCodeSent:$verificationId")

                storedVerificationId = verificationId
                resendToken = token
```

Sintaks diatas adalah objek yang bersifat *callback* yang mengimplementasikan *member* atau *function* dari class *interface*, untuk melakukan pengulangan *resend* atau mengirim kembali kode OTP dari server *google* ke nomor telepon yang sudah didaftarkan pengguna.

4.3.3.3. Autentikasi Menggunakan *Google Account*

Autentikasi selanjutnya pada *platform* sewa sarana olahraga ini adalah menggunakan *google account*. Pengguna android pasti memiki *account google*, sehingga ketika suatu aplikasi memiliki integrasi dengan *google account* dalam proses autentikasi hal ini akan menjadikan pengguna lebih mudah dalam proses login atau masalah hak akses dalam penggunaan aplikasi. Karena *firebase* merupakan salah satu produk *google* tentu saja firebase menyediakan kemudahan ini lewat layanan *firebase authentication* mereka. Ketika menggunakan layanan ini terlebih dahulu melakukan integrasi dengan libraru UI *authentication firebase*, secara sederhana library ini menyediakan desain antarmuka yang siap dipakai, jadi dalam pengembangan tidak perlu membuat *layouting* sendiri, hasilnya kurang lebih seperti pada gambar 4.24

```
implementation 'com.firebaseui:firebase-ui-auth:6.2.0'
```

Setelah melakukan *sync gradle* dapat digunakan *built in class* yang ada pada SDK *firebase-ui-auth*, dalam implemetasinya cukup melakukan *intent with result* ke activity yang telah disediakan dalam *built in* seperti berikut :

```
// insialisasikan layout login
val authMethodPickerLayout = AuthMethodPickerLay-
out.Builder(R.layout.activity_login)
            .setGoogleButtonId(R.id.btn_google).build()

// lakukan intent ke UI auth dengan membawa data
        startActivityForResult(AuthUI.getInstance()
            .createSignInIntentBuilder()

.setAuthMethodPickerLayout(authMethodPickerLayout)
            .setTheme(R.style.SplashTheme)
            .setAvailableProviders(providers)
            .setIsSmartLockEnabled(false)
            .build(), LOGIN_REQUEST_CODE)
```

Sintaks diatas menjelaskan terkait erat dengan bagaimana caranya mengimplemtasikan *login/autentikasi* menggunakan *google account* seperti gambar 4.24 dengan menuliskan *function startActivityResult* dan mengisi dua parameter dimana parameter pertama argumennya adalah sebagai berikut :

1. *Instance* dari *built in* antarmuka bawaan SDK *firebase UI auth*
2. parameter untuk mengambil nilai custom layout yang dirancang sendiri, seperti berikut :

```
val authMethodPickerLayout = AuthMethodPickerLay-
out.Builder(R.layout.activity_login)

       .setGoogleButtonId(R.id.btn_google).build()
```

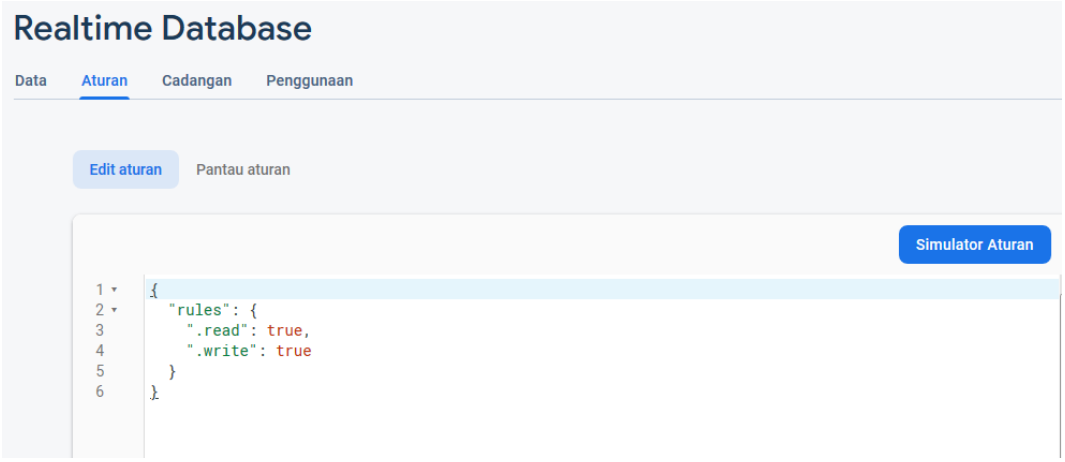
3. tema dari layout yang digunakan
4. penyedia provider yang ingin dipakai dalam hal ini adalah *google account* yang di insialisaikan sebagai berikut
5. merupakan sebuah *function* guna menyimpan *email* dan *password* agar mudah digunakan kembali saat menggunakan *login* via *google account*
6. .Build dalam *java/kotlin* digunakan untuk membentuk serangkain *function-function*

Untuk paramater kedua dari class **startActivityResulut()** ialah kode unik yang telah di insialisasikan sebelumnya, yang artinya function ini berkeja dibalik layar dan hasilnya akan dipanggil melalui sebuah function bernama onActivityResult yang dapat di overiding dalam class, dan di function inilah dapat dilakukan manipulasi berupa data-data autentication via goofle account lebih tepatnya menggunakan class intent, yang merupakan class bawaan android yang berguna atau berfungsi dalam membawa, mengirim, ataupun menerima data antar activity, ataupun antar fragment. Untuk lebih jelasnya penggunaan kedua function ini dapat dilihat dalam dokumentasi resmi android developer yang membahas tentang dua function ini secara rinci.

4.3.4. Implementasi *Firebase Realtime Database*

Selanjutnya pada sub-bab ini akan dibahas tentang bagaimana melakukan implementasi *firebase realtime database* yang gunanya melakukan *handling* penyimpanan dan manipulasi data dalam bentuk *database*. Yang harus dilakukan terlebih dahulu untuk menerapkan *realtime database* ini adalah konfigurasi *rules* pada *realtime database*.

Rules sendiri merupakan aturan keamanan dalam penggunaan *realtime database* terkait hak akses siapa saja yang dapat melakukan proses *read* dan *write* data. Dalam peneltian *rules* yang digunakan adalah membenarkan seluruh akses akan *write* dan *read* pada *firebase realtime database*, seperti gambar berikut :

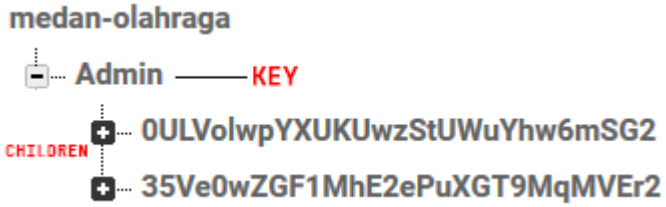


Gambar 4.77 aturan *realtime database platform* sewa sarana olahraga

Penjelasan dari rules diatas bahwa dalam proses manipulasi *database*, aplikasi yang terintegrasi dengan *realtime database* tersebut selalu di izinkan untuk proses *write* dan *read*. Selanjutnya agar memudahkkan dalam implementasi pada *dashboard* aplikasi *firebase* bisa mengisi struktur dokumen mengikuti pembahasan pada sub-bab *design database.*

Sama seperti implementasi *firebase authentication* dalam penerapan *firebase database realtime* juga menggunakan *built in sdk* yang telah disediakan oleh pihak google melalui *library* yang sudah di intergrasikan via *gradle*.

Dalam penelitian ini, pengembangan aplikasi menggunakan beberapa *function built in* yang disediakan dalam proses manipulasi database, implementasi pertama yang akan dibahas adalah terkait bagaimana memperoleh atau mengambil data dari *realtime firebase database*. Jika dilihat dari gambar dibawah merupakan struktur pada *firebase realtime database* yang mana sifat dari jenis database nya adalah *NoSQL*. Setelah yang dipaparkan pada pembahasan di bab dua terkait *NoSql* bahwa struktur dari database ini tidak saling berelasi melainkan konsepnya menggunakan dokumen dan isinya. Dalam *firebase realtime database* istilah dokumen dan isi ini menggunakan format JSON sehingga akan lebih mudah jika dibaca mengggunakan kata *key* dan *value* akan tetapi dalam *built in firebase database* pengunaan kata *value* disebut *child* sehingga dalam tulisan penelitian ini istilah ***key*** **dan** ***value*** diganti menjadi ***key*** **dan** ***child.*** Untuk lebih jelasnya dapat dilihat pada gambar dibawah ini :



Gambar 4.78 Penjelasan struktur dari *key* dan *children*

Dalam menangani proses *get data*, *firebase SDK* telah menyediakan *function* yang dapat digunakan sesuai kebutuhan, *function* yang dimaksud adalah *addSingleValueEventListener()* dan *addValueEventListener()*, secara garis besar kedua *function* ini memiliki tujuan yang sama yaitu itu untuk memperoleh data yang diminta kepada server. Akan tetapi kedua *function* ini memiliki cara penerapan yang berbeda dari proses pengambilan data nya. Kemudian untuk menangani proses update data dengan menggunakan updateChilderen() dan menghapus data menggunakan removeValue().

Berikut adalah pembahasan lebih lanjut bagaimana proses dan implementasi dari function yang telah dijelaskan dalam pengembangan platform sewa sarana olahraga ini.

4.3.4.1. *Add Single Listener Value Event*

Pada dasarnya konsep memanggil data di android menggunakan listener dan callback, secara sederhana konsep listener dapat dikatakan suatu event dalam proses logic yang kita tidak tahu user kapan akan melakukan proses event tersebut, dan callback adalah tempat penampung atau nilai kembalian dari event yang sudah diminta tersebut. Penggunaan *addSingleValueEventListener* yaitu hanya untuk mendapatkan atau melakukan permintaan data ke server sekali saja dan tidak muat lagi apabila ada pembaharuan dalam aplikasi jika data diserver berubah dan lalu jenis response yang dihasilkan dipanggil dalam callback jika user membutuhkan. Imlementasinya adalah sebagai berikut :

```
val adminRef = FirebaseDatabase.getInstance()
            .getReference(Commons.REFERENCE_ADMIN)
            .child(firebaseAuth.currentUser!!.uid)
        adminRef.addListenerForSingleValueEvent(object : Val-
ueEventListener {
            override fun onCancelled(p0: DatabaseError) {
                Log.e("SPLASH", "fetch data admin: " +p0.message )
            }

            override fun onDataChange(snapshot: DataSnapshot) {
                // jika path ada
                if (snapshot.exists()){
                    val adminModel = snap-
shot.getValue(AdminModel::class.java)
                    displayMainActivity(adminModel!!)
                } else {
                    Toast.makeText(this@SplashActivity, "Data kamu
tidak ditemukan diserver", Toast.LENGTH_SHORT).show()
                }
            }

        })
```

Dalam sintaks diatas seperti biasa wajib melakukan instance atas class yang akan digunakan, kemudian mengarahkan pada path *database* yang telah dibuat. getReference() akan menampung nilai *key* sedangan child() akan menampung nilai *children.*

Lalu barulah menggunakan *addSingleValueEventListener* dalam proses pengambilan datanya, yang mana dalam function ini akan mengasilkan callback berupa onCanceled dan OnDataChange. Kedua function ini adalah hasil imple-

mentasi dari interface ValueEventListener, onCanceled akan menangani callback apabila terjadi gangguan saat meminta request data, artinya permintaan belum sempat masuk ke server, contoh kasus seperti koneksi jaringan internet yang terputus, konfigurasi firebase yang tidak benar, dan sebagainya. Sedankan onDataChange akan menangani response dari *request* yang diminta kepada server. Dalam hal ini ditampung didalam paramater snapshot dimana semua value yang kita minta akan ditampung disana.

4.3.4.2. *Add Value Event Listener*

Berbeda dari addSingleValueEventListener, function ini akan selalu melakukan trigger terhadap proses perubahan yang terjadi didalam database. Akan tetapi memiliki konsep implementasi yang sama dengan addSingleListenerValueEvent yaitu menghasilkan dua buah callback onCanceled dan onDataChange. Implementasinya adalah sebagai berikut :

```
transaksiRef.addValueEventListener(object : ValueEventListener{
            override fun onCancelled(p0: DatabaseError) {

            }

            override fun onDataChange(snapshot: DataSnapshot) {
                listHistory.clear()
                for (item in snapshot.children){
                    //Log.i("riwayat", "${item.value}")
                    val model =
item.getValue(TransaksiItem::class.java)
                    listHistory.add(model!!)
                }
                listHistoryLiveData.postValue(listHistory)
            }
        })
```

4.3.4.3. *Update Children*

Dalam implementasi *firebase realtime database* penggunaan updateChilderen dipakai ketika hendak melakukan proses perubahan data kedalam database yang telah dibuat. Tentunya mengarah ke *path* yang disesuaikan, dengan landasannya adalah *key* dan *child* dari struktur JSON. Berikut adalah contoh sintaks penerapan/implementasi kodenya

```
val adminData = HashMap<String,Any>()
adminData["uid"] = firebaseAuth.currentUser!!.uid
adminData["nama_lengkap"] = namaLengkap
adminData["alamat"] = alamat
adminData["no_handphone"] = noHp
adminData["email"] = email
adminData["password"] = password
adminData["token"] = Commons.token

adminRef.updateChildren(adminData)
    .addOnCompleteListener { task->
        if (task.isSuccessful){
            Toast.makeText(this, "Berhasil terdaftar",
Toast.LENGTH_SHORT).show()
           val intent = Intent(this, LoginActivity::class.java)
           startActivity(intent)
           finish()
        }
    }
```

Paramater updateChildren() membutuhkan collection hasmap dalam mepresentasikan data apa saja yang akan dimasukkan kedalam path tujuan data firebase, contoh diatas adalah penerapan saat ingin memasukkan data identitas pemilik usaha lapangan kedalam *database realtime*, yaitu pertama dengan melakukan deklarasi collection hashmap dan memberikan assignment berupa nilai yang dibutuhkan, baru setelahnya di jadikan argument dalam updateChildren.

4.3.4.4. *Remove Value*

Terakhir adalah implementasi pada *realtime database firebase* terkait penghapusan *key* atau *children* pada suatu path atau bisa di bilang ini merupakan function yang berguna untuk menghapus data, *built in* ini akan menghapus secara pernamen nilai pada *children* dan *key*, jadi dalam implementasinya disarankan untuk berhati-hati dalam melakukan trigger ke path yang dituju karena sidat dari realtime database sendiri tidak bisa mengembalikan data atau melakukan restore apabila ada data tertentu yang sudah di hapus sehingga sangat berbahaya atau fatal apabila ada kesalahan, karena jika salah dalam menuju path yang dituju bisa menyebabkan data pada *key parent* juga terhapus. Berikut adalah contoh implementasi ataupun penerapan dari function *removeValue*.

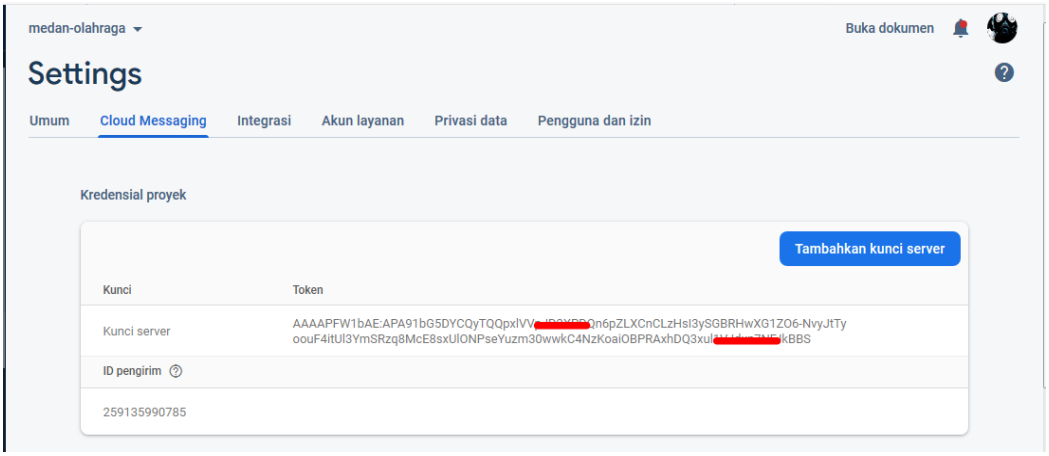
```
val favoriteRef = FirebaseDatabase.getInstance()
    .getReference("Favorite")
    .child(Commons.currentCustomer!!.uid)
    .child(uid)
favoriteRef.removeValue()
    .addOnCompleteListener { task ->

        if(task.isSuccessful){
           FirebaseDatabase.getInstance()
               .getReference("Favorite")
               .child(Commons.currentCustomer!!.uid)
               .addListenerForSingleValueEvent(object : Val-
ueEventListener{
                   override fun onCancelled(p0: DatabaseError) {

                   }

                   override fun onDataChange(snapshot: DataSnap-
shot) {
                       val templist = ArrayList<Favorite>()
                       if (snapshot.exists()){
                           for (item in snapshot.children){
                               val model =
item.getValue(Favorite::class.java)
                               templist.add(model!!)
                           }
                           callback.onCallback(templist)
                       } else {

callback.onCallback(ArrayList<Favorite>())
                       }
                   }

               })
        }
    }
```

Penggunaan *remove value* diawali dengan deklarasi path *firebase database realtime* yang akan mengarahkan ke path firebase itu sendiri, kemudian path tersebut ditampung pada sebuah variabel yang mana variabel ini akan dipanggil function *removeValue* dan function ini pun akan mengeksekusi path yang akan dihapus, dan mengembalikan dua callback function yaitu onCanceled dan onDataChange.

4.3.5. Implementasi *Firebase Cloud Messanging*

Pada pengembangan platform sewa sarana olahraga ini firebase cloud messanging digunakan untuk melakukan handle notifikasi pada aplikasi admin, aplikasi akan melakukan trigger terhadap data transaksi yang masuk. Jadi ketika customer melakukan pemesanan/reservasi lapangan akan masuk notifikasi ke pemilik bahwa ada pesanan baru.

Untuk mengakses *firebase cloud messanging* dibutuhkan sebuah API *key* agar device atau perangkat pengguna bisa berkomunikasi dengan server, untuk mendapatkan API key tersebut bisa didapatkan dari dashboard firebase seperti gambar dibawah ini :

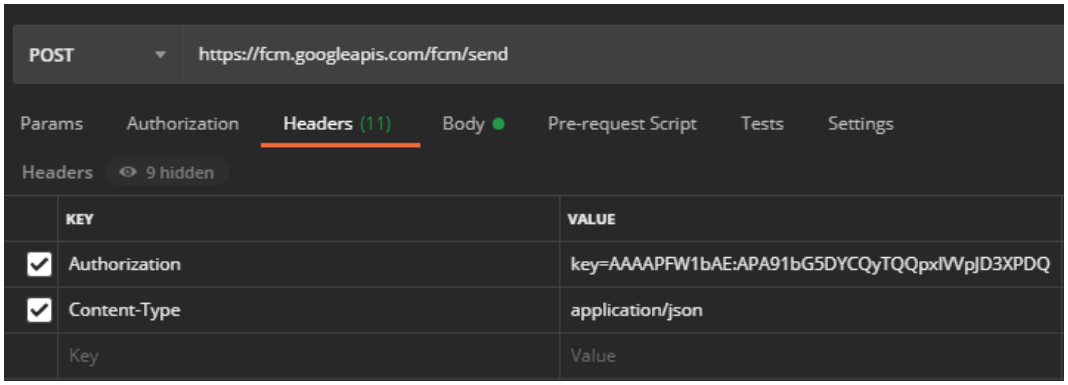


Gambar 4.79 Kunci server pada *cloud messanging*

Setelah mendapatkan kunci server, kunci ini dapat diuji menggunakan aplikasi penguji dari sisi *http clienti* seperti *postman* agar memberikan kepastian dalam pengembangan aplikasi. Untuk menguji *firebase cloud messanging* ini, bisa menembak request ke endpoint berikut

```
https://fcm.googleapis.com/fcm/send
```

dimana *header* nya adalah *Authorization* dengan *value "key = kunci key*" dan *content-type* berupa *application/json*, dan *http-verbs* adalah POST seperti gambar berikut :



Gambar 4.80 Pengaturan endpoint fcm menggunakan postman

```
{
    "to" : "dCL3S4WHArM:APA91bEyEe8l9spfAlGW0yfNKs-3SJPsDTJzg-
smb-
mE95dMHUQjBqNXaXCQRW9Usahmw5bsZibMeRAmK7zWhllrUBZvvURBwmZepP_B9RJWKousfMBwgr6oSj
LAnyEnn5g32oSUXUvbT",
    "data" : {
        "title" : "Pemberitahuan Reservasi",
        "message" : "Kamu mendapat 1 reservasi, ayo segera cek"
    }
}
```

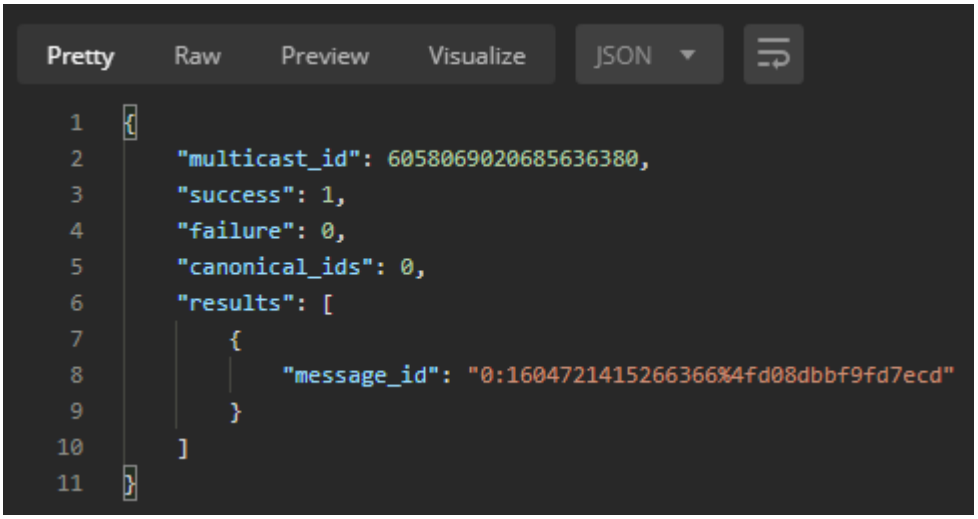
Gambar 4.81 Uji endpoint fcm menggunakan postman

Pada bagian body request data yang diminta ke server dapat dilihat seperti sintaks diatas, dimana request yang dimaksud adalah sebagai berikut :

Tabel 4.19 Penjelasan *request* data ke server

Key	Value
to	Tujuan token yang akan dikirimkan pesan broadcast, token tersebut di generate secara otomatis.
data	Objek dari request yang diberikan terkait judul dan pesan yang ingin di broadcast.
title	Judul pesan yang ingin di broadcast
message	Isi pesan yang ingin di broadcast

Setelah melakukan pengujian via postman, dari endpoint tersebut akan menampil-kan response, adapun response yang dimaksud adalah sebagai berikut :



Gambar 4.83 response sukses dari FCM

Setelah menguji endpoint FCM melalui postman artinya tidak ada masalah dalam API key yang digunakan, barulah implementasi dari FCM bisa dilakukan pada aplikasi android (dalam hal ini platform sewa sarana olahraga), sepeti yang dibahas diawal subbab ini, FCM digunakan untuk melakukan handle notifikasi, implementasi nya dilakukan dengan beberapa step berikut :

1. Mendapatkan token yang digenerate secara otomatis, yaitu dengan melakukan *instance* dari class *FirebaseInstance*, generate token ini dilakukan pada sisi administrator. Adapun sintaks nya adalah sebagai berikut

```
FirebaseIn-
stanceId.getInstance().instanceId.addOnCompleteListener(OnCompleteListen
er { task ->
    if (task.isSuccessful){
        // set nilai token
        Commons.token = task.result!!.token
        Log.d(TAG, "Token : " +task.result?.token)
    } else {
        Log.i("MAIN", "Gagal ambil token")
    }
})
```

Sintaks diatas menjelaskan tentang bagaimana suatu token digenerate yaiti dengan melakukan instance dan memanggil object instanceId yang mana kemudian listener nya menghasilkan return value dari task, dan didalam blok lambda nya, dapat dilakukan suatu pengecekan, apabila task token

berhasil di generate maka bisa melihat value *task.result.token* yang valuenya adalah *token fcm*.

2. Kemudian disisi client menggunakan bantuan *http client* seperti libarary retrofit, aplikasi dapat mengirimkan request layaknya postman kepada endpoint FCM yang telah disebutkan diatas.
3. Lalu disisi administrator, dapat dibuat satu class service yang akan selalu melakukan *trigger* dan berjalan di *background process* untuk melihat apakah ada permbaruan data pada server. Sintaks nya dapat dilihat sebagai berikut :

```
class FirebaseService : FirebaseMessagingService(){

  companion object {
     val TAG = FirebaseService::class.java.simpleName
  }

 private val channelID = "mychannel"

  override fun onMessageReceived(remoteMessage: RemoteMessage) {
     super.onMessageReceived(remoteMessage)

     Log.d(TAG, "onMessageReceived: " +remoteMessage.data["title"])

    val intent = Intent(this, SplashActivity::class.java)
   val notificationManager = getSystem-
Service(Context.NOTIFICATION_SERVICE) as NotificationManager
     val notificationID = Random.nextInt()

    // Jika hp user lebih besar dari OS oreo
     if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O){
        createNotification(notificationManager)
     }

     intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP)
    val pendingIntent = PendingIntent.getActivity(this, 0, intent, FLAG_ON
E_SHOT)
     val notifBuild = NotificationCompat.Builder(this, chan-nelID)
```

```
            .setPriority(NotificationCompat.PRIORITY_HIGH)
            .setDefaults(NotificationCompat.DEFAULT_VIBRATE)
            .setContentTitle(remoteMessage.data["title"])
            .setContentText(remoteMessage.data["message"])
            .setSmallIcon(R.drawable.ic_outline_av_timer)
            .setAutoCancel(true)
            .setContentIntent(pendingIntent)
            .build()

        notificationManager.notify(notificationID, notifBuild)

    }
    @RequiresApi(Build.VERSION_CODES.O)
    private fun createNotification(notificationManager: Notifica-
tionManager) {
        val channelName = "channel_name"
        val channel = NotificationChannel(channelID, channelName, IMPORTA
NCE_HIGH)
            .apply {
                description = "channel description"
                enableLights(true)
                lightColor = Color.GREEN
            }
        notificationManager.createNotificationChannel(channel)
    }
}
```

Sintaks diatas merupakan class *service*, yang mewarisi class *FirebaseCloud-Messanging* yang mana class ini akan mengimplementasikan *function onMessa-geReceived* yang mana paramater-nya adalah *remotemessage*, pada paramater ini ada data *title* dan *message* yang telah dikirimkan dari *request* keserver. Selanutnya dalam blok function ini dibuat sebuah notification yang akan melakukan handle title dan message yang telah didapatkan.

4.3.6. *Testing Blackbox*

Testing atau tahap pengujian merupakan fase atau tahap setelah implemetasi koding dilakukan, gunanya adalah untuk menguji apakah sistem yang dibangun sesuai atau memenuhi eskpetasi dari perancangan sistem yang telah dilakukan sebelumnya. Pada pengujian *blackbox* sebuah produk/sistem diuji dengan memperhatikan masing-masing fungsi, menu, dan tampilan. Berikut adalah kerangka atau rancangan tabel pengujian dari *platform* sewa sarana olahraga :

4.3.6.1 Pengujian *Blackbox* Pada Aplikasi Client

1. *Interface* Pendaftaran (Aplikasi *client*)

Tabel 4.20 Pengujian *Blackbox* Pada Interface Pendaftaran

No	Test case	Hasil yang diharapkan	Hasil yang didapatkan	Keterangan
1	Pengujian pendaftaran menggunakan *firebase authentication* layanan ***email***	*User* terdafar dengan *email* dan bisa melakukan *login.*	Sistem melakukan *insert* data ke *authentication* dan antarmuka menuju halaman utama.	Berhasil
2	Pengujian pendaftaran menggunakan *firebase authentiction* layanan **nomor telepon**	*User* terdafar dengan nomor telepon dan bisa melengkapi form data identitas lalu menuju halaman utama	Sistem melakukan *insert* data ke *authentication* dan sistem menampilkan form pengisian data, selanjutnya setelah melakukan tap pada button lanjutkan sistem menuju halaman menu utama.	Berhasil
3	Pengujian pendaftaran menggunakan *firebase authentication* layanan ***google account***	*User* terdafar dengan *google account* dan bisa melakukan *login*.	Sistem melakukan *insert* data ke *authentication* dan sudah bisa digunakan untuk login.	Berhasil

2. *Interface Login* (Aplikasi Client)

Tabel 4.21 Pengujian *Blackbox* Pada *Interface Login*

No	Test case	Hasil yang diharapkan	Hasil yang didapatkan	Keterangan
1	Pengujian *login* menggunakan *firebase authentication* layanan ***email***	*User* akan menuju halaman atau *activity home*.	Sistem mengeluarkan *toast to login* dan menuju *activity home*	Berhasil
2	Pengujian *login* menggunakan *firebase authentiction* layanan **nomor telepon**	*User login* dan mendapatkan kode OTP untuk verifikasi login, dan ketika submit OTP berhasil diverifikasi menju *activity home*.	Sistem mendapatkan kode OTP dan berhasil diverifikasi ketika submit lalu menuju *activity home*.	Berhasil
3	Pengujian *login* menggunakan *firebase authentication* layanan ***google account***	*User login* menggunakan akun *google* tertaut pada *device* user dan menuju *activity home*.	Sistem berhasil mengautentikasi dan menuju *activity home*.	Berhasil

3. *Interface* Menu *Home* (Aplikasi *Client*)

Tabel 4.22 Pengujian *Blackbox* Pada *Interface Home*

No	Test case	Hasil yang diharapkan	Hasil yang didapatkan	Keterangan
1	Menguji *button* lapanagan futsal	Menuju menu list lapangan futsal yang tersedia di platform	Sistem menampilkan menu llist lapangann futsal	Berhasil
2	Menguji *button* lapanagan basket	Menuju menu list lapangan basket yang tersedia di platform	Sistem menampilkan menu list basket	Berhasil
3	Menguji *button* lapangan voli	Menuju menu list lapangan voli yang tersedia di platform	Sisrem menampilkan menu list volli	Berhasil
4	Menguji *button* lapangan badminton	Menuju menu list lapangan badminton yang tersedia di platform	Sistem menampilkan menu list lapangan badminton	Berhasil

4. *Interface* Reservasi Lapangan (*Client*)

Tabel 4.23 Pengujian *Blackbox* Pada *Interface* Reservasi Lapangan

No	Test case	Hasil yang diharapkan	Hasil yang didapatkan	Keterangan
1	Memilih salah satu lapangan dari kategori olahraga yang disukai dengan mengklik *button* detail	Setelah memilih kategori lapangan akan menampilkan list lapangan sesuai kategori yang dipilih, dan ketika *button* detail diklik akan menampilkan detail lapangan berserta informasi jadwal yang tersedia.	Sistem menampilkan list lapangan sesuai kategori dan informasi detail lapangan ketika diklik *button* detail, dan menampilkan informasi jadwal yang tersedia.	Berhasil.
2	Menguji *button* favorit (icon hati).	Apabila *button* favorit di *tap* maka *icon* hati akan berubah menjadi berwarna merah, dan data akan tersimpan dalam *list* favorit. Sebaliknya apabila *icon* hati berwarna merah dan ketika di *tap* akan membuat *icon* hati menjadi tidak berwarna dan menghapus lapangan dari *list* favorit.	Sistem berhasil melakukan perubahan pada *icon* favorit setiap kali di *tap.* Dan melakukan penyimpanan ataupun hapus data setiap *icon* favorit di *tap*.	Berhasil
3	Menguji mengubah tanggal dan mengecek apakah jadwal yang di informasi *real time* sesuai tanggal didalam database	Ketika tanggal diubah informasi di item jadwal otomatis akan berubah sesuai jadwal pada tanggal yang dipilih.	Sistem menampilkan informasi jadwal yang tereservasi dan tidak sesuai tanggal yang dipilih.	Berhasil
4	Menguji *item* jadwal dengan melakukan *tap*	*Item* jadwal bisa dipilih lebih dari 1 item (lebih dari 1 jam) dan *item* berwarna abu-abu tidak bisa ditap	Sistem mampu memberikan *experience* bisa memilih item lebih dari satu kepada pengguna dan *item* yang berwarna abu-abu tidak	Berhasil

			meresponse ketika di *tap*	
5	Menguji *button* lanjutkan	Ketika di *tap* akan menuju tampilan konfirmasi dengan menampilkan informasi nama lapangan, kategori olahraga, tanggal reservasi, lama bermain dalam jam, jam berapa saja yang direservasi, dan total harga bayar dan pemilihan metode pembayaran	Sistem menampilkan informasi nama lapangan yang dipilih, kategori lapangan, lama permainan, jam berapa saja yang akan disewa, tanggal dan total bayar serta bisa memilih metode bayar.	Berhasil

5. *Interface* Konfirmasi Reservasi (*Client*)

Tabel 4.24 Pengujian *Blackbox* Pada *Interface* Konfirmasi Reservasi

No	**Test case**	**Hasil yang diharapkan**	**Hasil yang didapatkan**	**Keterangan**
1	Menguji *radio button cash*	Akan muncul sebuah *dialog alert*	Muncul sebuah *dialog alert*	Berhasil
2	Menguji *radio button voucher*	Akan muncul sebuah alert dialog dengan sisa *saldo voucher* tersebut	Muncul sebuah *alert dialog* dengan sisa saldo	Berhasil
2	Apabila *user* sudah melakukan reservasi di satu lapangan pada hari yang sama dan menggunakan metode pembayaran *cash*	Sistem akan menolak *user* untuk mengganti jadwal	Sistem menolak *user* untuk menggganti jadwal	Berhasil
3	Apabila *user* sudah melakukan reservasi disatu lapangan pada hari yang sama dan menggunakan metode pambayaran *voucher*	Sistem akan mengizinkan pergantian jadwal (waktu dan tanggal) dan memperbarui *database*	Sistem mengizinkan pergantian jadwal (waktu dan tanggal) dan memperbarui *database*	Berhasil

4	Melakukan *reset* hari setelah hari pemesanan hari ini.	Pengguna yang memakai via *cash* dilapangan yang sama boleh melakukan reservasi lagi	Sistem melakukan reset otomatis pada pengguna yang membayar via *cash* dihari berikutnya	Berhasil
5	Menguji reservasi-lapangan dengan metode *cash*	Data akan masuk ke *database* slot dan transaksi	Data masuk ke slot *database children slot* dan transaksi	Berhasil
6	Menguji reservasi-lapangan dengan metode *voucher*	Data akan masuk ke *database* slot dan transaksi	Data masuk ke slot *database children slot* dan transaksi	Berhasil

6. *Interface* Histori (*Client*)

Tabel 4.25 Pengujian *Blackbox* Pada *Interface* Histori

No	**Test case**	**Hasil yang diharapkan**	**Hasil yang didapatkan**	**Keterangan**
1	Menampilkan *list* transaksi reservasi *user*	Sistem akan menampilkan *list* transaksi reservasi *user* dengan tanggal dan jam berapa saja yang direservasi	Sitem menampilkan *list* transaksi reservasi *user* dengan tanggal dan jam berapa saja yang direservasi	Berhasil

7. *Interface* Favorit (*Client*)

Tabel 4 26 Pengujian *Blackbox* Pada *Interface* Favorit

No	**Test case**	**Hasil yang diharapkan**	**Hasil yang didapatkan**	**Keterangan**
1	Menampilkan *list* favorit lapangan *user*	Sistem menampilkan *list* favorit lapangan *user*	Berhasil menampilkan *list* favorit lapangan *user*	Berhasil
2	*Button* hapus untuk menghapus *item* dari *list* favorir	Sistem menghapus item	Berhasil menghapus item	Berhasil

8. *Interface Account* (*Client*)

Tabel 4.27 Pengujian *Blackbox* Pada *Interface Account*

No	Test case	Hasil yang diharapkan	Hasil yang didapatkan	Keterangan
1	*Button Logout*	Sistem menampil-kan alert dialog konfirmasi *logout*	Berhasil menam-pilkan konfirmasi *logout*	Berhasil
2	*Button QR Code*	Sistem menam-pilkan *qr code*	Menampilkan *qr code*	Berhasil
3	Mengubah alamat dan foto ptofil	Memperbarui alamat atau foto profil di*database*	Memperbarui alamat atau foto profil di *dataabse*	Berhasil

4.3.6.2 Pengujian *Blackbox* Pada Aplikasi Admin

1. *Interface Login* (*Admin*)

Tabel 4.28 Pengujian *Blackbox* Pada *Interface Login*

No	Test case	Hasil yang diharapkan	Hasil yang didapatkan	Keterangan
1	*Button Login*	Setelah memasuk-kan *email* dan *password* sistem masuk kehalaman utama	Sistem masuk ke halaman utama .	Berhasil

2. *Interface* Registrasi (*Admin*)

Tabel 4.29 Pengujian *Blackbox* Pada *Interface* Registrasi

No	Test case	Hasil yang diharapkan	Hasil yang didapatkan	Keterangan
1	*Button* Daftar	Setelah mlengkapi data, dan tap *button* daftar, sistem akan masuk kehalaman *login*	Sistem berhasil mendaftar dan menyimpan data ke *database* dan menuju halaman login	Berhasil

3. *Interface Home* (*Admin*)

Tabel 4.30 Pengujian *Blackbox* Pada *Interface Home*

No	Test case	Hasil yang diharapkan	Hasil yang didapatkan	Keterangan
1	Menampilkan jumlah transaksi hari ini dan jumlah transaksi saldo selama ini	Menampilkan jumlah transaksi sesuai data pada *database*	Sistem Menmpilkan jumlah transasi sesuai data yang ada pada *database*	Berhasil

4. *Interface* Kelola Lapangan (*Admin*)

Tabel 4.31 Pengujian *Blackbox* Pada *Interface* Kelola Lapangan

No	Test case	Hasil yang diharapkan	Hasil yang didapatkan	Keterangan
1	*Button* Tambah Lapangan	Dapat menambahkan lapangan sesuai data yang di *inputkan* dan masuk ke dalam database.	Sistem menambahkan data lapangan sesuai *inputan*	Berhasil
2	*Button* perbarui data lapangan	Dapat memperbarui data lapangan berupa gambar data informasi lainnya	Sistem memperbarui data lapangan berserta data pada *database*	Berhasil

5. *Interface* Kelola Jadwal (*Admin*)

Tabel 4.32 Pengujian *Blackbox* Pada *Interface* Kelola Jadwal

No	Test case	Hasil yang diharapkan	Hasil yang didapatkan	Keterangan
1	*Button* Konfirmasi untuk metode pembayaran cash	Mengubah status transaksi menjadi selesai dan pelanggan dapat melakukan reservasi kembali	Sistem melakukan *update* untuk status transaksi dan pelanggan dapat melakukan reservasi kembali	Berhasil
2	*Button* Konfirmasi untuk metode pembayaran voucher	Mengubah status transaksi menjadi selesai dan pelanggan dapat melakukan reservasi kembali	Mengubah status transaksi menjadi selesai dan pelanggan dapat melakukan reservasi kembali	Berhasil

3	*Button* Tolak untuk metode pembayaran *cash*	Pelanggan tidak bisa melakukan reservasi pada hari yang sama	Sistem membuat pelanggan tidak bisa melakukan reservasi pada hari yang sama	Berhasil

6. *Interface* Kelola Saldo (*Admin*)

Tabel 4.33 Pengujian *Blackbox* Pada *Interface* Kelola Saldo

No	Test case	Hasil yang diharapkan	Hasil yang didapatkan	Keterangan
1	*list user* yang pernah melakukan transaksi saldo	Menampilkan *list user* yang pernah melakukan transaksi saldo	Sistem menampilkan *list user* yang pernah melakukan transaksi saldo	Berhasil
2	*Button* scan *qr code*	Menampilkan tampilan scan *qr code*	Sistem menampilkan tampilan scan *qr code*	Berhasil
3	Pengisian saldo	Ketika scan dilakukan maka akan melakukan trigger ke user dan saldo otomatis ditambahkan	Sistem membaca qr code dengan baik dan saldo ditambahkan	Berhasil

7. *Interface* Laporan (*Admin*)

Tabel 4.34 Pengujian *Blackbox* Pada *Interface* Laporan

No	Test case	Hasil yang diharapkan	Hasil yang didapatkan	Keterangan
1	Button tanggal awal	Menampilkan dialog tanggal	Sistem berhasil menampilkan dialog tanggal	Berhasil
2	*Button* tanggak akhir	Menampilkan dialog tanggal	Sistem menampilkan dialog tanggal	Berhasil
3	Lihat laporan	Laporan digenerate dalam bentuk PDF	Sistem melakukan generate data dalam bentuk dokumen pdf	Berhasil

BAB V
PENUTUP

5.1 Kesimpulan

Berdasarkan penelitian yang telah dilakukan oleh penulis, maka diperoleh beberapa kesimpulan berikut, diantaranya :

1. Telah dikembangkannya sebuah *platform* yang mampu memfasilitasi antara pelanggan dan penyedia jasa sewa sarana olahraga untuk melakukan transaksi secara *online* dan *realtime*, sebuah platform yang mampu melakukan proses pembuatan rekapitulasi keuangan yang sistematis, dan *platform* yang mampu memberikan pemasaran yang lebih luas dan tepat sasaran.
2. Dalam melakukan implementasi *firebase* pada aplikasi *android* dapat dilakukan dengan mendaftarakan *key* aplikasi tersebut ke *firebase*, dan menggunakan SDK yang disediakan *firebase* untuk menggunakan layanan *authentication, realtime database,* dan *cloud messanging* kedalam pengembangan *platform* sarana olahraga.

5.2 Saran

Saran dari penulis agar penelitian ini jauh lebih bisa berkembang, maka dalam penelitian selanjutnya dapat memperhatikan beberapa hal berikut :

1. Peneliti selanjutnya diharapkan dapat mengintegrasikan sistem dengan layanan *maps* untuk memudahkan pengguna dalam mencari alamat lapangan sesuai titik kordinat.
2. Layanan *firebase* tidak hanya sebatas tiga hal saja yaitu *authentication, realtime database,* dan *cloud messaging.* Untuk pengembangan selanjutnya diharapkan menggunakan layanan firebase lainnya, ataupun membandingkan layanan *firebase* dengan lainnya seperti *Amazon Web Service*.

DAFTAR PUSTAKA

Akli, Ibnu. (2018). *Referensi Dan Panduan UML 2.4 Singkat Tepat Jelas.* Surabaya : Garuda Mas Sejatera.

Ameldi, Roni dan Tengku Khairil Ahsyar. (2018). Sistem Informasi Lapangan Futsal Berbasis Android Pada Lapangan Futsal. *Jurnal Ilmiah Rekayasa dan Manajemen Sistem Informasi*, ISSN : 2502-8995

Andi Juansyah. (2015). Pembangunan Aplikasi Child Tracker Berbasis Assisted – Global Positioning System (A-GPS) Dengan Platform Android. *Jurnal Ilmiah Komputer Dan Informatika (KOMPUTA), 1*(1), 1–8. Retrieved from elib.unikom.ac.id/download.php?id=300375

Andalia, F., & Setiawan, E. B. (2015). Pengembangan Sistem Informasi Pengolahan Data Pencari Kerja Pada Dinas Sosial Dan Tenaga Kerja Kota Padang. *Komputa : Jurnal Ilmiah Komputer Dan Informatika*, 4(2), 93–97. https://doi.org/10.34010/komputa.v4i2.2431

Asmara, R. (2016). Sistem Informasi Pengolahan Data Penanggulangan Bencana Pada Kantor Badan Penanggulangan Bencana Daerah (BPBD) Kabupaten Padang Pariaman. *Jurnal J-Click*, *3*, 80–91.

Cancer, Y., & Alim, Z. (2016). Platform As a Service (PaaS) Sebagai Layanan Sistem Operasi Cloud Computing. *Jurnal TIMES*, Vol. V No(6), 381–384. https://doi.org/10.1007/s12599-011-0183-3

Cholifah, Wahyu Nur, Yulianingsih Yulianingsih, and Sri Melati Sagita. (2018). "Pengujian Black Box Testing Pada Aplikasi Action & Strategy Berbasis Android Dengan Teknologi Phonegap." *STRING (Satuan Tulisan Riset Dan Inovasi Teknologi)* 3(2):206.

Ems, Tim. 2015. *Belajar Bahasa Pemrograman Java Dari Nol*. Jakarta : Elex Media Komputindo

Glady Sukma Perdana. (2017). *Sistem Informasi Geografis Tempat Olahraga di Provinsi Daerah Istimewa Yogyakarta Berbasis Web*. 1–5.

Hamalik, Oemar. (2007). *Dasar-dasar Pengembangan Kurikulum*. Bandung : Remaja Rosdakarya.

Handini, Ade. (2016). Pemodelan UML Sistem Infrormasi Monitoring Penjualan Dan Stok Barang (Studi Kasus: Distro ZheZha Pontianak). *Jurnal Khatulistiwa Informatika*, 4(2), 107-116.

Ihwan Susila. (2015). Pendekatan Kualitatif Untuk Riset Pemasaran Dan Pengukuran Kinerja Bisnis. *Benefit Jurnal Manajemen Dan Bisnis* 19(1):12–23.

Ilhami, M. (2017*).* Pengenalan Google Firebase Untuk Hybrid Mobile Apps Berbasis Cordova. *Jurnal IT CIDA, 3*(124), 16–29.

Irawan, M. D., & Herviana, H. (2019). Implementasi Logika Fuzzy Dalam Menentukan Jurusan Bagi Siswa Baru Sekolah Menengah Kejuruan (Smk) Negeri 1 Air Putih. *Jurnal Teknologi Informasi,* 2(2), 129. https://doi.org/10.36294/jurti.v2i2.427

Juansyah, Andi. (2015). Pembangunan Aplikasi Child Tracker Berbasis Assisted – Global Positioning System (A-GPS) Dengan Platform Android. *Jurnal Ilmiah Komputer dan Informatika (KOMPUTA),* ISSN : 2089-9033

Kadir, Abdul. (2010). *Dasar Aplikasi Database MySQL-Delphi.* Yogyakarta : Andi.

Kristanto, Harianto. (2004). *Konsep dan Perancangan Database.* Yogyakarta : Andi.

Lengkong, H. N., Sinsuw, A. A. E., & Lumenta, A. S. M. (2015). Perancangan Penunjuk Rute Pada Kendaraan Pribadi Menggunakan Aplikasi Mobile GIS Berbasis Android Yang Terintegrasi Pada Google Maps. *E-Journal Teknik Elektro Dan Komputer*, *4*(2), 18–25.

Maiyana, Efmi. (2018). Pemanfaatan Android Dalam Perancangan Aplikasi Kumpulan Doa, ISSN : 2459-9549

Mutia, Intan. (2016). Pemanfaatan Komputasi Awan (Cloud Computing) Bagi Pembelajaran Mahasiswa Perguruan Tinggi. *Jurnal String* . 1(1), 1–9.

Ratnasari, D., Hadi, H. F., & Budiarto, J. (2018). Rancang Bangun Aplikasi Penyewaan Lapangan Futsal Berbasis Android. *JUTI*, *16*(2), 144. https://doi.org/10.12962/j24068535.v16i2.a738

Rachma, Nuzulia dan Arif Wibisono. (2018). Analisis Pemetaan Model Bisnis Platform Online Property Di Indonesia Dengan Menggunakan Platform Design Tools. *Jurnal Teknik ITS*. ISSN :2337-3520

R. K. Panchal dan A. K. Patel, (2017). A comparative study: Java Vs Kotlin Programming in Android, *International Journal of Innovative Trends in Engineering & Research,* vol. 2, no. 9

Safaat, Nazruddin. (2019). *Rancang Bangun Aplikasi Android dan Web.* Bandung. Informatika

Satyaputra, A & Aritonang, E, M. (2016). *Let's Build Your Android Apps With Android Studio*. Jakarta. Elex Media Komputindo.

Samsudin. (2019). Optimalisasi Penerimaan Remunerasi Dosen Menggunakan Metode Rule Base Reasoning. *Kumpulan Jurnal Ilmu Komputer (KLIK)*, 06(3), 224–240.

S. Doug. (2010). *High Performance Android Apps*, Sebastopol: O'Reilly Media, Inc,

Sonita, A., & Fardianitama, R. F. (2018). Aplikasi E-Order Menggunakan Firebase dan Algoritme Knuth Morris Pratt Berbasis Android. Pseudocode, *5*(2), 38–45. https://doi.org/10.33369/pseudocode.5.2.38-45

Sutabri, Tata. (2012). *Konsep Sistem Informasi.*Yokyakarta : ANDI

Suendri. (2018). Implementasi Diagram UML (Unified Modelling Language) Pada Perancangan Sistem Informasi Remunerasi Dosen Dengan Database Oracle (Studi Kasus: UIN Sumatera Utara Medan*). Jurnal Ilmu Komputer Dan* Informatika, 3(1), 1–9.

Silalahi, M. (2018). Perbandingan Performansi Database Mongodb Dan Mysql Dalam Aplikasi File Multimedia Berbasis Web. *Computer Based Information System Journal*, *6*(1), 63. https://doi.org/10.33884/cbis.v6i1.574

Siti Chomsyah (2017). Tinjauan Yuridis Perjanjian Sewa Menyewa Bangunan Toko Dalam Bentuk Tidak Tertulis. *Jurnal Advokasi*, 197-208

Sunarto (2014). Sewa Menyewa Mobil Rental ditinjau dari Ekonomi Islam (Studi Kasus di Kecamatan Sario Kota Manado. *Jurnal Ilmiah Al-Syri'ah.* 12(1)

Tutik, T. T., Islam, P. E., & Pustaka, L. (2008). Idri dan Titik Triwulandari Tutik, Prinsip-Prinsip Ekonomi Islam (Jakarta: Lintas Pustaka, 2008), h. 1.

Yunaeti Anggraini, Elisabet dan Rita irviani. (2017). *Pengantar Sistem Informasi.* Yokyakarta : ANDI

LAMPIRAN I

Source Code Program :

LoginActivity.kt

```
class LoginActivity : AppCompatActivity() {

  private lateinit var providers: List<AuthUI.IdpConfig>
  private lateinit var firebaseAuth: FirebaseAuth

  private var email: String = ""
  private var password: String = ""

  companion object {
    const val LOGIN_REQUEST_CODE = 7171
  }

  override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_login)

    // insialiasasi firebase auth
    firebaseAuth = FirebaseAuth.getInstance()

    // isi value provider (mau login via apa)
    providers = Arrays.asList(
      AuthUI.IdpConfig.GoogleBuilder().build()
    )

    // insialisasi button
    val btnLogin = findViewById<Button>(R.id.btn_login)
    val btnPhone = findViewById<Button>(R.id.btn_phone)
    val btnDaftar = findViewById<Button>(R.id.btn_daftar)
    val edtEmail = findViewById<TextInputLayout>(R.id.edt_email)
    val edtPassword = findViewById<TextInputLayout>(R.id.edt_password)

    btnLogin.setOnClickListener{

      email = edtEmail.editText!!.text.toString()
      password = edtPassword.editText!!.text.toString()

      checkUserFromDb(email, password)
    }

    btnDaftar.setOnClickListener{
      val intent = Intent(this@LoginActivity, SignUpActivity::class.java)
      startActivity(intent)
    }

    btnPhone.setOnClickListener {
      val intent = Intent(this@LoginActivity, SendsmsActivity::class.java)
      startActivity(intent)
    }

    btn_google.setOnClickListener{
      // insialiasai auth UI Google
      showGoogleLayout()
    }
  }

  private fun checkUserFromDb(email: String, password: String) {
```

```
        fire-
baseAuth.signInWithEmailAndPass
word(email,password)
            .addOnCompleteListener{
task ->
                if (task.isSuccessful){
                    Toast.makeText(this,
"Succes to login",
Toast.LENGTH_SHORT).show()
                } else {
                    Log.e("ERROR",
"checkUserFromDb:
"+task.exception!!.message )
                }
            }
    }


    private fun showGoogleLayout() {
        // insialisasikan layout login
        val authMethodPickerLayout =
AuthMethodPickerLay-
out.Builder(R.layout.activity_login)

.setGoogleButtonId(R.id.btn_google)
.build()

        // lakukan intentke UI auth
dengan membawa data
        startActivityForRe-
sult(AuthUI.getInstance()
            .createSignInIntentBuilder()

.setAuthMethodPickerLayout(authM
ethodPickerLayout)

.setTheme(R.style.SplashTheme)

.setAvailableProviders(providers)

.setIsSmartLockEnabled(false)
            .build(),           LOG-
IN_REQUEST_CODE)

    }

    override fun onActivi-
tyResult(requestCode: Int, re-
sultCode: Int, data: Intent?) {
        su-
per.onActivityResult(requestCode,
resultCode, data)
    }

    override fun onBackPressed() {
        finish()
    }
```

ConfirmFragment.kt

```
class ConfirmFragment : Fragment()
{

    // Binding xml variable
    private var txtKategori :
TextView? = null
    private var txtVanue: TextView? =
null
    private var txtTanggal: TextView?
= null
    private var txtWaktu: TextView?
= null
    private var txtHarga: TextView? =
null
    private var txtJam:TextView? =
null
    private var radioGroup: Radi-
oGroup? = null
    private var btnKonfirmasi: But-
ton? = null

    private var maxKey: Long = 0
    private var sisaSaldo: Long? = null
    private var viaVoucher: Boolean =
false

    private lateinit var confirmView-
Model: ConfirmViewModel
    private var currentSaldo = 0L

    override fun onCreateView(
        inflater: LayoutInflater, con-
tainer: ViewGroup?,
        savedInstanceState: Bundle?
```

```
    ): View? {
        confirmViewModel = ViewModelProviders.of(this).get(ConfirmViewModel::class.java)
        val root = inflater.inflate(R.layout.fragment_confirm, container, false)
        initView(root)
        return root
    }

    @SuppressLint("SetTextI18n")
    private fun initView(root: View) {
        txtKategori = root.findViewById(R.id.confirm_txt_kategori)
        txtVanue = root.findViewById(R.id.confirm_txt_vanue)
        txtTanggal = root.findViewById(R.id.confirm_txt_tanggal)
        txtWaktu = root.findViewById(R.id.confirm_txt_waktu)
        txtHarga = root.findViewById(R.id.confirm_txt_harga)
        txtJam = root.findViewById(R.id.confirm_txt_jam)
        radioGroup = root.findViewById(R.id.confirm_radioGroup)
        btnKonfirmasi = root.findViewById(R.id.btn_konfirmasi)

        //ubah format tanggak ke dd/MM/yyyy
        val dateSelected: String = SimpleDateFormat("dd/MM/yyyy", Locale.getDefault()).format(Commons.bookingDate.time)

        txtKategori!!.text = Commons.selectedCategory
        txtVanue!!.text = Commons.courtSelected!!.name_vanue
        txtWaktu!!.text = Commons.convertSlotText(Commons.newSlotPosition ?: "none")
        txtTanggal!!.text = dateSelected
        txtJam?.text = Commons.listSlotSelected.size.toString() + " Jam"

        val totalHarga = Commons.courtSelected!!.price * Commons.listSlotSelected.size
        txtHarga!!.text = "IDR $totalHarga"

    }

    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {

        // cek saldo
        confirmViewModel.checkSaldo()

        confirmViewModel.getSaldoLiveData().observe(viewLifecycleOwner, androidx.lifecycle.Observer {
            sisaSaldo = it.saldo
        })

        /// cek transaksi
        confirmViewModel.checkTransaksi()

        // livedata
        confirmViewModel.getTransaksiData().observe(viewLifecycleOwner, androidx.lifecycle.Observer {result ->

            val tanggal = result.tanggal
```

```
        val metodeBayar = result.metode_bayar
        val durasi = result.durasi

        // TODO: Jika tanggalnya ada dan metode bayarnya voucher
        if (tanggal.isNotEmpty() && metodeBayar == "voucher"){

            // TODO : Alert Dialog konfirmasi perubahan tanggal
            val alertDialogBuilder = AlertDialog.Builder(context)
            alertDialogBuilder.setCancelable(false)
            alertDialogBuilder.setTitle("Konfirmasi Perubahan")
            alertDialogBuilder.setMessage("Yakin ingin memperbaharui tanggal dan waktu reservasi ?")
            alertDialogBuilder.setPositiveButton("Yakin"){ dialogInterface, i ->

                confirmViewModel.updateTransaksi()

                // coba intent dengan eventbus
                EventBus.getDefault().postSticky(ChangeJadwal(true))
                val intent = Intent(requireActivity(), HomeActivity::class.java)
                intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TASK)
                startActivity(intent)
                requireActivity().finish()
            }
            alertDialogBuilder.setNegativeButton("Batal"){ dialogInterface, i ->
                dialogInterface.dismiss()
                label_metode_bayar.visibility = View.GONE
                radioGroup!!.visibility = View.GONE
                btnKonfirmasi!!.visibility = View.GONE
            }

            // TODO: jika item slot yang dipilih gak sama dengan durasi
            if (Commons.listSlotSelected.size != durasi){
                Toast.makeText(context, "Tidak boleh memilih slot lebih atau kurang dari $durasi jam ", Toast.LENGTH_LONG).show()
                label_metode_bayar.visibility = View.GONE
                radioGroup!!.visibility = View.GONE
                btnKonfirmasi!!.visibility = View.GONE
            } else {
                alertDialogBuilder.create()
                alertDialogBuilder.show()
            }

            // TODO : jika metode bayarnya cash
        } else if (tanggal.isNotEmpty() && metodeBayar == "cash"){


            // TODO: Buat alert dialog
            val alertDialogBuilder = AlertDialog.Builder(context)
            alertDialogBuilder.setCancelable(false)
```

```
            alertDialogBuilder.setTitle("Tidak bisa mengubah jadwal")
            alertDialogBuilder.setMessage("Tidak dapat mengubah jadwal untuk pemesanan dengan metode bayar cash")
            alertDialogBuilder.setPositiveButton("Oke"){ dialogInterface, i ->
                // coba intent dengan eventbus
                EventBus.getDefault().postSticky(ChangeJadwal(true))
                Commons.tanggalSlotINfo= ""
                val intent = Intent(requireActivity(), HomeActivity::class.java)
                intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TASK)
                startActivity(intent)
                requireActivity().finish()
            }
            alertDialogBuilder.create()
            alertDialogBuilder.show()
        }


    })

    // ambil nilai saldo via viewmodel
    confirmViewModel.getSaldoLiveData().observe(viewLifecycleOwner, androidx.lifecycle.Observer {
        currentSaldo = it.saldo
    })

    // state
    confirmViewModel.getStateSaldo().observe(viewLifecycleOwner, androidx.lifecycle.Observer {
        handleStateUI(it)
    })

    // insialisasi alert dialog
    val alertDialog = AlertDialog.Builder(context)

    radioGroup!!.setOnCheckedChangeListener(RadioGroup.OnCheckedChangeListener{ radioGroup, checkedId ->
        val radio: RadioButton = view.findViewById(checkedId)

        // kalau radio button yg di cek sama dengan nilai view yg dibandingkan
        if (radio == view.findViewById(R.id.confirm_radioCash)){
            // tampilkan alert dialog
            alertDialog.setTitle(getString(R.string.perhatian))
            alertDialog.setMessage(getString(R.string.pesan_cash))
            alertDialog.setPositiveButton("Saya Mengerti"){ dialogInterface, i ->
                btnKonfirmasi!!.isEnabled = true
                btnKonfirmasi!!.setOnClickListener{
                    confirmViewModel.tambahTransaksi("cash", null)
                    EventBus.getDefault().postSticky(ChangeJadwal(true))

                    // push notificatoon
                    val title = "Pesanan Baru"
```

```
                val message = "Ada yg booking lapangan kamu loh!"

                PushNotification(
                    NotificationData(
                        title, message
                    ),
                    Commons.courtSelected!!.token
                ).also {
                    sendNotification(it)
                }

                val intent = Intent(requireActivity(), HomeActivity::class.java)
                intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TASK)
                startActivity(intent)
                requireActivity().finish()
            }
        }

        val dialog = alertDialog.create()
        dialog.show()

    } else if (radio == view.findViewById(R.id.confirm_radioMoney)){
        if (currentSaldo < Commons.courtSelected!!.price){
            configDialog(2)
        } else {
            configDialog(1)
        }
    }
})

btnKonfirmasi!!.setOnClickListener {
    confirmViewModel.tambahTransaksi("voucher", sisaSaldo!!.toInt())
    EventBus.getDefault().postSticky(ChangeJadwal(true))

    // push notificatoon
    val title = "Pesanan Baru"
    val message = "Ada yg booking lapangan kamu loh!"

    PushNotification(
        NotificationData(
            title, message
        ),
        Commons.courtSelected!!.token
    ).also {
        sendNotification(it)
    }

    val intent = Intent(requireActivity(), HomeActivity::class.java)
    intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TASK)
    startActivity(intent)
    requireActivity().finish()
  }

}

private fun sendNotification(notification: PushNotification) = CoroutineScope(Dispatchers.IO).launch {
    try {
        val response = RetrofitInstance.api.postNotification(notification)
        if(response.isSuccessful) {
            Log.d(TAG, "Response: ${Gson().toJson(response)}")
        } else {
            Log.e(TAG, response.errorBody().toString())
        }
```

```
        } catch(e: Exception) {
            Log.e(TAG, e.toString())
        }
    }


    private fun handleStateUI(it: SaldoState?) {

    }

    fun configDialog(show: Int?){

        val alertDialogBuilder = AlertDialog.Builder(context)

        if (show == 1){
            alertDialogBuilder.setTitle("Konfirmasi Pembayaran")
            alertDialogBuilder.setMessage("Saldo voucher kamu untuk lapangan ini Rp.$currentSaldo. Gunakan voucher ?")
            alertDialogBuilder.setPositiveButton("Iya"){ dialogInterface, i ->
                btnKonfirmasi!!.isEnabled = true
            }
            alertDialogBuilder.setCancelable(false)
            alertDialogBuilder.create()
            alertDialogBuilder.show()
        } else {
            alertDialogBuilder.setTitle("Konfirmasi Pembayaran")
            alertDialogBuilder.setMessage("Maaf tidak bisa melanjutkan transaksi. Saldo kamu sisa Rp.$currentSaldo")
            alertDialogBuilder.setPositiveButton("Oke"){dialogInterface, i ->
                dialogInterface.dismiss()
                btnKonfirmasi!!.isEnabled = false
            }
            alertDialogBuilder.setCancelable(false)
            alertDialogBuilder.create()
            alertDialogBuilder.show()
        }
    }
```