SKRIPSI

PERANCANGAN SISTEM INFORMASI PENYEWAAN GUEST HOUSE DI BANDAR LAMPUNG BERBASIS ANDROID

Diajukan sebagai salah satu syarat untuk mencapai gelar

SARJANA KOMPUTER

Pada Program Studi Sistem Informasi
IIB Darmajaya Bandar Lampung

Disusun Oleh :

M RIDHO TRI PUTRA

1611058003P

JURUSAN SISTEM INFORMASI
FAKULTAS ILMU KOMPUTER
INSTITUT INFORMATICS AND BUSINESS DARMAJAYA
BANDAR LAMPUNG

2019

PERNYATAAN ORISINILITAS PENELITIAN

PERNYATAAN

Saya yang bertanda tangan di bawah ini, menyatakan bahwa Skripsi yang diajukan ini adalah hasil karya saya sendiri, tidak terdapat karya yang pernah diajukan untuk memperoleh gelar kesarjanaan di suatu perguruan tinggi atau karya yang pernah ditulis atau diterbitkan orang lain kecuali yang secara tertulis diacu dalam naskah ini disebutkan dalam daftar pustaka. Karya ini adalah milik saya dan pertanggungjawaban sepenuhnya berada dipundak saya.

Bandar Lampung, 20 September 2019

METERAI TEMPEL 19DEBAHF334751738 6000 ENAM RIBU RUPIAH

M RIDHO TRI PUTRA
1611058003P

PERSETUJUAN

Judul Skripsi	:	**PERANCANGAN SISTEM INFORMASI PENYEWAAN GUEST HOUSE DI BANDAR LAMPUNG BERBASIS ANDROID**
Nama Mahasiswa	:	**M. RIDHO TRI PUTRA**
No. Pokok Mahasiswa	:	**1611058003P**
Program Studi	:	**S1 Sistem Informasi**

Telah diperiksa dan disetujui untuk diajukan dan dipertahankan dalam sidang Tugas Penutup Studi guna memperoleh gelar Sarjana Ilmu Komputer pada Program Studi Sistem Informasi IIB Darmajaya.

Disetujui oleh :

Dosen Pembimbing	Ketua Program Studi, Sistem Informasi
Bobby Bachry, S.Kom., M.M.S.I	**Nurjoko, S.Kom., M.T.I**
NIK 12740212	NIK 00440702

PENGESAHAN

Telah diuji dan dipertahankan di depan tim penguji Skripsi
Program Studi Sistem Komputer Institut Informatika dan Bisnis Darmajaya
Bandar Lampung dan dinyatakn diterima untuk
memenuhi syarat guna memperoleh Gelar
Sarjana

Mengesahkan

1. Tim Penguji — Tanda Tangan

Ketua : TM. Zaini, S.Kom., M.Kom .....

Anggota : Halimah, S.Kom., M.T.I .....

2. Dekan Fakultas Ilmu Komputer

Zainir Jamal, S.T., M.Eng
NIK. 00590203

Tanggal Lulus Ujian Skripsi : 20 September 2019

HALAMAN PERSEMBAHAN

Puji dan syukur penulis panjatkan kehadirat Allah SWT karena atas rahmat dan karunia-Nya, Skripsi yang berjudul " Perancangan Sistem Informasi Penyewaan Guest House di Bandar Lampung Berbasis Android " ini dapat diselesaikan dengan lancar . skripsi ini ku persembahkan kepada :

1. Ayahanda tercinta Sarhudin yang telah memberikan saya semangat tanpa henti dan membawa saya sampai ke jenjang perkuliahan.
2. Ibuku tercinta Erhayati terima kasih atas semua yang telah atas semua yang telah engkau berikan, kasih sayang yang begitu besar serta arahan dan motivasi dalam segala hal, tak lepas doamu untuk keberasilan anakmu.
3. Kakak -Kakaku ku Eriyansa Perdana Putra dan M Nugraha Dwi Putra yang selalu memberikan saya semangat serta dukungan untuk menyelesaikan tugas akhir (skripsi).
4. Adik Ku M Rafli Saputra yang selalu memberikan doa serta dukungan.
5. Sahabat – sahabatku dan teman – teman yang tak bias ku sebutkan satu persatu yang telah memberikan semangat, kritikan yang membangun serta dukungan dalam menyusun karya ini.
6. Seluruh keluarga besarku yang selama ini mendukungku selamaku menuntut ilmu diperguruan tinggi IIB Darmajaya.
7. Seluruh dosen-dosen IIB Darmajaya terima kasih semua, khusunya dosen-dosen Program Studi Sistem Komputer dan Teknik Komputer.
8. Almamater tercinta ku IIB DARMAJAYA.

MOTTO

“ Kesuksesan bukan dilihat dari hasilnya,
Tapi dilihat dari prosesnya.
Karena hasil direkayasa dan dibeli.
Sedangkan proses selalu jujur menggambarkan siapa diri kita sebenarnya**”**

“M Ridho Tri Putra”

PERANCANGAN SISTEM INFORMASI PENYEWAAN GUEST HOUSE BANDAR LAMPUNG BERBASIS ANDROID

Oleh
M Ridho Tri Putra
(1611058003P)

ABSTRAK

Guest house merupakan sejenis fasilitas baik milik perorangan atau perusahaan yang diperuntukkan khusus bagi tamu hendak menginap. *Guest house* dapat berupa rumah pribadi yang dikonversi untuk kepentingan tamu dan lebih dikenal sebagai penginapan ala *backpacker*. Tidak adanya detail informasi *guest house* dapat menyulitkan dan memperlambat para *backpaper* dalam melakukan perjalanan baik wisata maupun perjalanan lainnya di daerah yang dikunjungi. Maka dari itu diperlukan suatu sistem yang dapat mempermudah *backpaper* dalam mendapatkan informasi *guest house* secara detail.

Sistem yang dibangun berbasiskan Android yang dapat diakses melalui *smartphone* dengan sistem operasi Android. Sistem ini dibuat hanya dapat dijalankan pada versi Android 5.0 hingga versi terbaru saat ini. Editor yang digunakan untuk pembuatan aplikasi ini adalah Android Studio dengan menggunakan MySQL sebagai basis datanya. Di dalam sistem juga terdapat SIG dengan pemanfaatan Google Map.

Adanya sistem ini dapat menampilkan informasi guest house secara detail seperti harga, lokasi, masa menginap, fasilitas dan informasi lainnya sehingga memberikan kemudahan bagi para *backpaper* atau masyarakat lainnya yang hendak mencari tempat tinggal sementara selain hotel. Sistem ini juga dilengkapi dengan *chat* yang memudahkan pemilik dan penyewa dalam berkomunikasi. Adanya pembayaran secara *online* (pembayaran via transfer). Adanya pembayaran via transfer juga dapat memudahkan penyewa dalam membayar tempat tersebut tanpa harus menunggu kedatangan pemilik.

Kata Kunci : Guest House, Android, dan MySQL

DESIGN OF ANDROID BASED GUEST HOUSE RENT INFORMATION SYSTEM IN BANDAR LAMPUNG

By:
M Ridho Tri Putra
(1611058003P)

ABSTRACT

A guest house is a kind of facility either owned by an individual or a company specifically for guests who want to stay overnight. Guest house can be in the form of a private home converted for the benefit of guests and better known as a backpacker-style inn. Lack of detailed guest house information can complicate and slow down the backpackers in doing travel both tours and other trips in the area visited. Therefore, it need a system that can simplify back paper in getting detailed guest house information. The system was built based on Android which was accessed through smartphones with the android operating system. This system was only made run on Android version 5.0 up to the latest version at this time. Editor who used for making this application was Android Studio with using MySQL as its database. In the system there was also a GIS by using Google Map. The existence of this system can display detailed guest house information such as price, location, stay, amenities and other information, so it make the users easy for the backpackers or other people who want to find a temporary residence other than a hotel. This system is also equipped with chat that makes it easy for owners and tenants to communicate. There is payment online (payment via transfer). Payment via transfer can also make it easier for tenants to pay for the place without having to wait for the owner's arrival.

Keywords: Guest House, Android, and MySQL

KATA PENGANTAR

Puji syukur penulis panjatkan kehadirat Allah SWT karena atas rahmat dan karunia-Nya, sehingga saya dapat menyelesaikan suatu tugas dari jurusan dalam bentuk skripsi yang menjadi kewajiban bagi setiap mahasiswa Jurusan Sistem Informasi IIB Darmajaya Bandar Lampung dengan judul ***"PERANCANGAN SISTEM INFORMASI PENYEWAAN GUEST HOUSE DI BANDAR LAMPUNG BERBASIS ANDROID"*** Skripsi ini disusun untuk memenuhi salah satu syarat untuk mencapai gelar Sarjana Komputer (S.KOM) Sistem Informasi, Institut Informatika dan Bisnis Darmajaya.

Dengan selesainya skripsi ini, saya mengucapkan terima kasih yang setulus-tulusnya kepada :

1. Bapak Ir. Firmansyah YA., MBA., M. Sc., Selaku Rektor IIB Darmajaya Bandar Lampung.
2. Bapak Dr. RZ. Abdul Aziz, ST.,MT, selaku Wakil Rektor Bidang Akademik IIB DARMAJAYA.
3. Bapak Ronny Nazar, SE,M.M, selaku Wakil Rektor Bidang Administrasi Umum dan Keuangan IIB DARMAJAYA Bandar Lampung.
4. Bapak Muprihan Thaib, S.Sos, Selaku Wakil Rektor Bidang Kemahasiswaan dan Sumber Daya IIB DARMAJAYA Bandar Lampung.
5. Bapak Prof . Ir Zulkarnain Lubis, M.S., Ph.D., Selaku Wakil rector bidang HKPIO dan ICT IIB DARMAJAYA Bandar Lampung.
6. Bapak Nurjoko, S.Kom., M.T.I Selaku Ketua Program Studi Sistem Informasi, terimakasih atas waktu dan saran yang telah bapak berikan kepada saya.
7. Bapak Hendra Kurniawan, S.Kom., M.T.I selaku Sekretaris Program Jurusan Sistem Informasi . terimakasih atas waktu dan saran yang telah bapak berikan kepada saya.
8. Bapak Bobby Bachry, S.Kom., M.M.S.I selaku dosen pembimbing yang telah memberikan koreksi dan Saranya.

9. Bapak dan Ibu ku tercinta yang selalu setia mendukungku baik dalam material maupun spiritual selama penyusunan skripsi ini.
10. Bapak dan Ibu Dosen Pengajar terutama jurusan Sistem Informasi yang telah membagi ilmu dan pengetahuan mereka yang bermanfaat kepada penyusunan dalam pembelajaran.
11. Kakak ku M Eriyansa Perdana Putra yang selalu memberikan saya semangat serta dukungan untuk menyelesaikan tugas akhir (skripsi).
12. Adiku M Rafli Saputra , yang selalu memberikan doa serta dukungan;
13. Rekan – rekan seperjuangan jurusan Sistem Informasi IIB Darmajaya serta mahasiswa/mahasiswi angkatan 2015, yang telah memotivasi dan membantu demi terwujudnya skripsi ini.
14. Terimakasih untuk seluruh keluarga besar atas dukungannya.
15. Seluruh pihak yang membantu dalam kelancaran dan kesuksesan Skripsi penulis.

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

BAB I
PENDAHULUAN

1.1 Latar Belakang

Lampung merupakan Provinsi paling selatan di Pulau Sumatera, Indonesia, dengan Ibukota terletak di Bandar Lampung. Provinsi ini memilki 2 Kota (Kota Bandar Lampung dan Kota Metro) dan 15 Kabupaten. Sebagai pintu gerbang untuk masuk ke Pulau Sumatera, Provinsi ini memiliki pertumbuhan ekonomi yang stabil yang didukung dari sektor industri, pariwisata, maupun pendidikan. Sektor-sektor tersebut membawa keuntungan tersendiri bagi masyarakat yang tinggal di dekat kawasan sektor dengan membuka usaha bisnis tempat hunian sementara (kos atau kontrakan).

Bandar Lampung adalah pusat kota dari Provinsi Lampung. Di Bandar Lampung banyak ditemui Perguruan Tinggi baik negeri maupun swasta, rumah sakit, wisata kuliner maupun alam, dan masih banyak lainnya. Oleh karena itu, penginapan sangat penting bagi para pekerja, masyarakat, atau wisatawan dari luar daerah tentunya yang dibutuhkan bukan penginapan berkelas hotel. Indekos atau kos atau kontrakan merupakan sebuah jasa yang menawarkan sebuah kamar atau tempat untuk ditinggali dengan sejumlah pembayaran dalam setiap periode tertentu. Sebagai Provinsi yang memilki 2 Kota (Kota Bandar Lampung dan Kota Metro) dan 15 Kabupaten, tentu *guest house* seperti kos atau kontrakan banyak tersebar di seluruh wilayah Lampung. Akan tetapi, penyebaran kos atau kontrakan tidak diiringi oleh informasi yang dapat dengan mudah diketahui oleh mayarakat khususnya kalangan *backpacker*, pelajar/mahasiswa dan para pekerja. Mereka mencari informasi mengenai kos atau kontrakan dengan cara datang langsung ke beberapa titik lokasi yang dekat dengan lokasi pariwisata, tempat kerja atau sekolah/perguruan tinggi sampai ditemukannya tempat yang pas dari segi harga sewa, lama penyewaan maupun fasilitasnya. Pencarian dengan cara tersebut memakan biaya yang cukup tinggi dari segi transportasi dan makan, terlebih pencarian yang dilakukan memakan waktu lama yang lebih dari 1 hari.

Di era tekologi yang telah canggih dengan adanya *smartphone*, maka teknologi ini dapat dimanfaakan sebagai solusi alternatif sarana pemasaran para pelaku usaha kos atau kontrakan. Menurut Wahyu (2016) dalam penelitiannya mengenai sistem informasi rumah kos berbasis android di wilayah AUB Surakarta, meyimpulkan bahwa sistem ini dapat menyajikan informasi mengenai rumah kos dan wadah bagi pemlik kos untuk mempromosikan usahanya. Menurut Rosadi (2016), aplikasi sistem infromasi pencarian tempat kos Kota Bandung berbasis Android dapat memberikan kemudahan dalam melakukan pencarian tanpa datang ke lokasi tersebut. Oleh karena itu, dari beberapa penelitian yang dilakukan sebelumnya dan dari beberapa permasalahan mengenai pencarian tempat hunian sementara (kos atau kontrakan, maka pada penelitian ini dilakukanlah penelitian mengenai sistem informasi kos atau kontrakan dengan judul "**Perancangan Sistem Informasi Penyewaan *Guest House* Lampung Berbasis Android**". Penelitian ini diharapkan agar dapat mempermudah masyarakat khususnya *backpacker*, pelajar/mahasiswa, dan para pekerja dalam menemukan tempat tinggal sementara sesuai dengan kebutuhan mereka mulai dari segi harga sewa, fasilitas, serta kebutuhan akan lama penyewaan, dan sebagai media promosi para pelaku usaha *guest house*.

1.2 Perumusan Masalah

Dari permasalahan yang telah diuraikan pada latar belakang, maka pada penelitian ini dapat dirumuskan masalah "Bagaimana merancang dan membangun suatu sistem informasi penyewaan *guest house* Lampung berbasis Android?"

1.3 Ruang Lingkup Penelitian

Penelitian dilakukan di beberapa tempat kos/kontrakan di wilayah bandar Lampung, dengan batasan penelitian sebagai berikut :

a. Penelitian hanya dilakukan di wilayah Bandar Lampung tepatnya di daerah Labuhan Ratu.
b. Sistem yang dibangun menyajikan informasi penyewaan *guest house* seperti kos atau kontrakan yang ada di wilayah Bandar Lampung.

c. Biaya sewa dapat dilakukan dengan cara transfer dengan cara mengunggah foto bukti pembayaran.
d. Sistem yang dibangun berbasis Android dengan rekomendasi minimum sistem operasi adalah versi 5.0 (Lollipop).
e. *Database* yang digunakan dalam membangun sistem ini adalah MySQL.

1.4 Tujuan Penelitian

Tujuan dari penelitian ini adalah membangun sebuah sebuah sistem informasi penyewaan *guest house* (kos atau kontrakan) yang dapat membantu memberikan informasi berupa lokasi, harga, ukuran ruangan, dan fasilitas kepada masyarakat khususnya kalangan *backpacker,* pelajar/mahasiswa, dan para pekerja.

1.5 Manfaat Penelitian

Manfaat dari penelitian ini adalah sebagai berikut :

a. Memudahkan masyarakat khususnya kalangan *backpacker,* pelajar/mahasiswa, dan para pekerja guna mendapatkan informasi mengenai penyewaan *guest house* yang tersebar di wilayah Lampung.
b. Sebaga sarana alternatif media promosi para pelaku usaha *guest house* guna memperoleh keuntungan.

1.6 Sistematika Penulisan

Secara garis besar penelitian ini terdiri dari 5 (lima) bab dengan sistematika penulisan sebagai berikut :

BAB I Pendahuluan

Pada bab ini diuraikan latar belakang masalah, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, dan sistematika penulisan.

BAB II Tinjauan Pustaka

Pada bab ini berisi teori-teori yang berkaitan dan mendukung penelitian serta penulisan skripsi ini yang akan dilakukan oleh penulis/peneliti.

BAB III Metodologi Penelitian

Dalam bab ini dijelaskan tentang metodologi penelitian penyelesaian masalah yang dijelaskan di perumusan masalah yang meliputi metode pengumpulan data, dan metode pengembangan sistem.

BAB IV Hasil Penelitian dan Pembahasan

Pada bab ini akan dibahas mengenai penjelasan implementasi dari aplikasi yang dibangun.

BAB V Simpulan dan Saran

Bab ini berisikan simpulan dari seluruh pembahasan dan saran yang diperlukan untuk perbaikan dimasa yang akan datang.

Daftar Pustaka

Lampiran

BAB II
TINJAUAN PUSTAKA

2.1 Sistem

Pada dasarnya, sistem adalah sekumpulan elemen yang saling terkait atau terpadu yang dimaksudkan untuk mencapai suatu tujuan. Sebagai gambaran, jika sebuah sistem terdapat elemen yang tidak memberikan manfaatdalam mencapai tujuan yang sama, maka elemen tersebut dapat dipastikan bukanlah bagian dari sistem (Abdul Kadir, 2014).

2.2 Informasi

McFadden, dan kawan-kawan mendefinisikan informasi sebagai data yang telah diproses sedemikian rupa sehingga meningkatkan pengetahuan seseorang yang menggunakan data tersebut. Shannon dan Weaver, dua orang insinyur listrik melakukan pendektan secara matematis untuk mendefinisikan informasi (Kroenke). Menurut mereka, informasi adalah jumlah ketidakpasian yang dikurangi ketika sebuah pesan diterima. Artinya, dengan adanya informasi, tingkat kepastian menjadi meningkat. Menurut Davis, informasi adalah data yang telah diolah menjadi sebuah bentuk yang berarti bagi penerimanya dan bermanfaat dalam pengambilan keputusan saat ini atau saat mendatang (Abdul Kadir, 2014).

2.3 Sistem Informasi

Sesungguhnya yang dimaksud sistem informasi tidak harus melibatkan komputer. Sistem informasi yang menggunakan komputer biasa disebut sistem informasi berbasis komputer (*Computer Based Information System* atau CBIS). Dalam praktik, istilah sistem informasi lebih sering dipakai tanpa berbasis komputer, walaupun dalam kenyataannya komputer merupakan bagian yang penting. Di buku ini, yang dimaksudkan dengan sistem iformasi adalah sistem informasi berbasis komputer. Ada beragam definisi sistem informasi, yaitu :

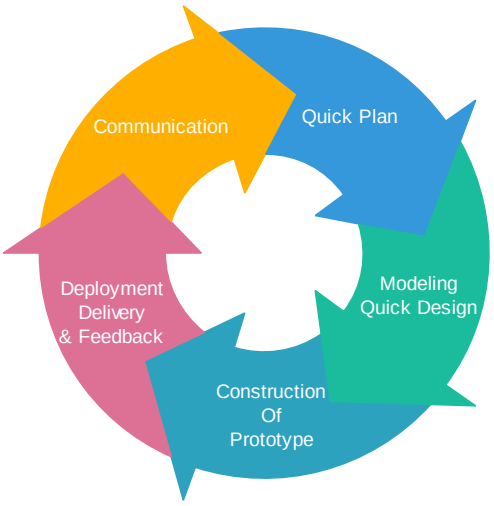
a. Alter, sistem informasi adalah kombinasi antar prosedur kerja, informasi, orang dan teknologi informasi yang diorganisasikan untuk mencapai tujuan dalam sebuah organisasi.

b. Bodnar dan Hopwoo, sistem informasi adalah kumpulan perangkat keras dan peragkat lunak yang dirancang untuk mentransformasikan data ke dalam bentuk informasi yang berguna.
c. Gelinas, Oram dan Wiggins, sistem informasi adalah suatu sistem buatan manusia yang secara umum terdiri atas sekumpulan komponen berbasis komputer dan manual yang dibuat untuk menghimpun, menyimpan dan mengelola data serta menyediakan informasi keluaran kepada para pemakai.
d. Hall, Sistem informasi adalah sebuah rangkaian prosedur formal, dimana data dikelompokkan, diproses menjadi informasi dan didistribusikan kepada para pemakai.
e. Turban, McLean dan Wetherbe, Sebuah sistem informasi mengumpulkan, memproses, mentimpan, menganalisis dan menyebarkan informasi untuk tujuam yang spesifik.
f. Wilkinson, Sistem informasi adalah kerangka kerja yang mengkoordinasikan sumber daya (manusia dan komputer) untuk mengubah masukan (*input*) menjadi keluaran (informasi) guna mencapai sasaran-sasaran perusahaan.

Berdasarkan berbagai definisi tersebut, dapat disimpulkan bahwa sistem informasi mencangkup sejumlah komponen (manusia, komputer, teknologi informasi dan prosedur kerja), ada sesuatu yang diproses (data menjadi informasi) dan dimaksudkan untuk mencapai suatu sasaran atau tujuan (Abdul Kadir, 2014).

2.4 Metode Pengembangan Sistem

Metode *prototype* suatu proses pembuatan *software* yang bersifat berulang dan dengan perencanaan yang cepat yang dimana terdapat umpan balik yang memungkinkan terjadinya perulangan dan perbaikan *software* sampai dengan *software* tersebut memenuhi kebutuhan dari sisi pengguna. Siklus atau ilustrasi dari metode prototipe dapat dilihat pada Gambar 2.1.



Gambar 2.1 Ilustrasi Model Prototipe (Sumber : Pressman, 2012)

Pembuatan prototipe dimulai dengan dilakukannya komunikasi antara tim pengembang perangkat lunak dengan para pelanggan. Tim pengembang perangkat lunak akan melakukan pertemuan-pertemuan dengan para stakeholder untuk mendefinisikan sasaran keseluruhan untuk perangkat lunak yang akan dikembangkan, mengidentifikasi spesifikasi kebutuhan apa pun yang saat ini diketahui, dan menggambarkan area-area dimana definisi lebih jauh pada iterasi selanjutnya merupakan keharusan. Iterasi pembuatan prototipe direncanakan dengan cepat dan pemodelan (dalam bentuk "rancangan cepat") dilakukan. Suatu rancangan cepat berfokus pada representasi semua aspek perangkat lunak yang akan dilihat oleh para pengguna akhir (misalnya rancangan antar muka pengguna (*user insterface)* atau format tampilan). Rabcangan cepat (*quick design*) akan memulai konstruksi pembuatan prototipe. Prototipe kemudian akan diserahkan kepada para stakholder dan kemudian mereka akan melakukan evaluasi-evaluasi tertentu terhadap prototipe yang telah dilakukan sebelumnya, kemudian akhirnya akan memberikan umpan-balik yang akan digunakan untuk memperhalus spesifikasi kebutuhan. Iterasi akan terjadi saat prototipe diperbaiki untuk memenuhi kebutuhan dari para stakeholder, sementara pada saat yang sama memungkinkan kita untuk lebih memahami kebutuhan apa yang akan dikerjakan pada saat iterasi selanjutnya.

Idealnya, prototipe bertindak sebagai mekanisme untuk mengidentifikasi spesifikasi-spesifikasi kebutuhan perangkat lunak. Jika suatu prototipe yang dapat digunakan akan dikembangkan, kita bisa menggunakan program yang sudah ada sebelumnya atau dengan menerapkan penggunaan perkakas yang sudah ada misalnya perkakas pembentuk laporan (*report generator*) atau aplikasi untuk melakukan perancangan antarmuka (*window manager*) yang memungkinkan program yang dapat digunakan dapat dibuat dengan mudah dan cepat (Pressman, 2012).

2.5 Metode Pengumpulan Data

Menurut Rosa dan Shalahuddin (2018), hal pertama yang dilakukan dalam analisis sistem adalah melakukan pengumpulan data. Ada beberapa teknik pengumpulan data yang sering dilakukan, yaitu :

a. Teknik Wawancara

   Pengumpulan data dengan menggunakan wawancara memiliki beberapa keuntungan, yaitu :

   1. Lebih mudah dalam menggali bagian sistem mana yang dianggap baik dan bagian mana yang dinaggap kurang baik.
   2. Jika ada bagian tertentu yang perlu digali lebih dalam, maka dapat menanyakannya langsung kepada narasumber.
   3. Dapat menggali kebutuhan user secara lebih bebas.
   4. *User* dapat mengungkapkan kebutuhannya secara lebih bebas.

b. Teknik Observasi

   Pengumpulan data dengan menggunakan observasi mempunyai keuntungan, yaitu :

   1. Ananlisis dapat melihat langsung bagaimana sistem lama berjalan.
   2. Mampu menghasilkan gambaran lebih baik jika dibanding dengan teknik lainnya.

c. Teknik Studi Pustaka

   Studi pustaka dilakukan untuk memperoleh data dan informasi dengan membaca berbagai bahan penulisan, karangan ilmiah serta sumber-sumber lain mengenai permasalahan yang berhubungan dengan penulisan.

2.6 Alat Pengembangan Sistem

Alat pengembangan sistem yang digunakan menggunakan pemodelan *Unified Modeling Language* (UML) *use case diagram*, *activity diagram*, dan *class diagram*.

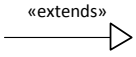
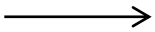
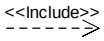
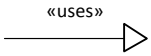
2.6.1 Use Case Diagram

Use case mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. Secara kasar, *use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi itu. Syarat penamaan pada *use case* adalah nama didefinisikan sesimpel mungkin dan dapat dipahami (Rosa dan Shalahuddin, 2018). Adapun simbol-simbol *use case diagram* dapat di lihat pada Tabel 2.1.

Tabel 2.1 Simbol *Use Case Diagram*

Keterangan	Simbol	Deskripsi
Use Case		Fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor; dinyatakan dengan menggunakan kata kerja diawal-awal frase nama *use case*
Aktor		Orang, proses atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar itu sendiri. Jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang
Asosiasi		Komunikasi antara aktor dan *use case* yang berpartisipasi pada *use case* atau *use case* memiliki interaksi dengan aktor.

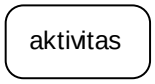
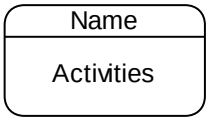
Tabel 2.1 (Lanjutan)

Keterangan	Simbol	Deskripsi
Ekstensi		Relasi *use case* tambahan ke sebuah *use case*, dimana *use case* yang ditambahkan dapat berdiri sendiri walau tanpa *use case* tambahan itu; mirip dengan prinsip *inheritance* pada pemograman berorientasi objek.
Generalisasi		Hubungan generalisasi dan spesialisasi (umum-khusus) antara dua buah *use case* dimana fungsi yang satu adalah fungsi yang lebih umum dari lainnya.
Menggunakan /*include/uses*	 	Ada dua sudut pandang yang cukup besar mengenai *include* di *use case* : a. *Include* berarti *use case* yang ditambahkan akan selalu dipanggil saat *use case* tambahan dijalankan b. *Include* berarti *use case* yang tambahan akan selalu melakukan pengecekan apakah *use case* yang ditambahkan telah dijalankan sebelum *use case* tambahan dijalankan.

2.6.2 Activiy Diagram

Diagram aktivitas atau *activity diagram* menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis. Perlu diperhatikan disini adalah bahwa diagram aktivitas menggambarkan aktivitas sistem bukan apa yang dilakukan aktor, jadi aktivitas yang dapta dilakukan oleh sistem (Rosa dan Shalahuddin, 2018). Simbol-simbol yang terdapat pada *activity diagram* adalah seperti pada Tabel 2.2.

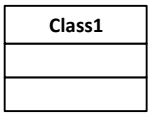
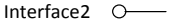
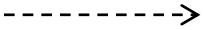
Tabel 2.2 Simbol Diagram Aktivitas

Keterangan	Simbol	Deskripsi
Status awal		Status awal aktivitas sistem, sebuah diagram aktivitas memiliki sebuah status awal.
Aktivitas		Aktivitas yang dilakukan sistem, aktivitas biasanya diawali dengan kata kerja.
Percabangan		Asosiasi percabangan dimana jika ada pilihan aktivitas lebih dari satu.
Penggabungan		Asosiasi penggabungan dimana lebih dari satu aktivitas digabungkan menjadi satu.
Swimlane		Memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi.
Status akhir		Status akhir yang dilakukan sistem, sebuah diagram aktivitas memiliki sebuah status akhir.

2.6.3 Class Diagram

Diagram kelas atau *class diagram* menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. kelas memiliki apa yang disebut atribut dan metode atau operasi. Atribut merupakan variabel-variabel yang dimiliki suatu kelas, sedangkan operasi atau metode adalah fungsi-fungsi yang dimiliki oleh suatu kelas (Rosa dan Shalahuddin, 2018). Simbol-simbol yang ada pada diagram kelas adalah seperti pada Tabel 2.3.

Tabel 2.3 Simbol *Class Diagram*

Simbol	Deskripsi
Kelas 	Kelas pada struktur sistem.
Natarmuka/*interface* 	Sama dengan konsep *interface* dalam pemograman berorientasi objek.
Asosiasi	Relasi antar kelas dalam makna umum, asosiasi biasanya juga disertai dengan *multiplicity*.
Asosiasi berarah	Relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan *multiplicity*.
Generalisasi	Relasi antar kelas dengan makna generalisasi-spesialisasi (umum-khusus).
Kebergantungan 	Relasi antar kelas dengan makna kebergantungan antar kelas.
Agregasi 	Relasi antar kelas dengan makna semua bagian (*whole-part*).

2.7 *Database Management System* (DBMS)

Basis data (*database*) adalah suatu pengorganisasian sekumpulan data yang saling terkait sehingga memudahkan aktivitas untuk memperoleh informasi. Basis data di maksudkan untuk mengatasi problem pada sistem yang memakai pendekatan berbasis berkas.

Untuk mengelola basis data diperlukan perangkat lunak yang disebut *Database Management System* (DBMS). DBMS adalah perangkat lunak sistem yang memungkinkan para pemakai membuat, memelihara, mengontrol, dan mengakses

basis data dengan cara yang praktis dan efisien. DBMS dapat digunakan untuk mengakomodasikan berbagai macam pemakai yang memiliki kebutuhan akses yang berbeda-beda. Tedapat beberapa elemen basis data, yaitu (Abdul Kadir, 2014):

a. *Database*

   Database atau basis data adalah kumpulan tabel yang mempunyai kaitan antara suatu tabel dengan tabel lainya sehingga membentuk suatu bangunan data.

b. Tabel

   Tabel adalah kumpulan *record-record* yang mempunyai panjang elemen yang sama dan atribut yang sama namun berbeda data valuenya.

c. Entitas

   Entitas adalah sekumpulan objek yang terdefinisikan yang mempunyai karakteristik sama dan bisa dibedakan satu dengan lainnya. Objek dapat berupa barang, orang, tempat atau suatu kejadian.

d. Atribut

   Atribut adalah deskripsi data yang bisa mengidentifikasi entitas yang membedakan entitas tersebut dengan entitas yang lain. Seluruh atribut harus cukup untuk menyatakan identitas objek atau dengan kata lain, kumpulan atribut dari setiap entitas dapat mengidentifikasi keunikan suatu individu.

e. *Data Value* (Nilai Data)

   Data value adalah data aktual atau informasi yang disimpan pada tiap data, elemen atau atribut. Atribut nama pegawai menunjukan tempat dimana informasi nama karyawan disimpan, nilai datanya misalnya adalah Anjang, Arif, Suryo dan lain-lain yang merupakan isi data nama pegawai tersebut.

f. *File*

   File adalah kumpulan record sejenis yang mempunyai panjang elemen yang sama, atribut yang sama namun berbeda nilai datanya.

g. *Record/Tuple*

   Kumpulan elemen-elemen yang saling berkaitan menginformasikan tentang suatu entitas secara lengkap. Satu record mewakili satu data atau informasi.

2.8 *Java*

Sebagai sebuah bahasa pemrograman, Java dapat membuat seluruh bentuk aplikasi, desktop, web dan lainnya, sebagaimana dibuat dengan menggunakan bahasa pemrograman konvensional yang lain.

Java adalah bahasa pemrograman yang berorientasi objek (OOP) dan dapat dijalankan pada berbagai platform sistem operasi. Perkembangan Java tidak hanya terfokus pada satu sistem operasi, tetapi dikembangkan untuk berbagai sistem operasi dan bersifat *open source* (Kadir, 2014).

2.9 Android

Android adalah sistem operasi dan *platform* pemograman yang dikembangkan oleh Google untuk ponsel cerdas dan perangkat seluler lainnya (seperti tablet). Android bisa berjalan di beberapa macam perangkat dari banyak produsen yang berbeda. Android menyertakan *kit development* perangkat lunak untuk penulisan kode asli dan perakitan modul perangkat lunak untuk membuat aplikasi bagi pengguna Android. Android Juga menyediakan pasar untuk mendistribusikan aplikasi. secara keseluruhan, Android menyatakan ekosistem untuk aplikasi seluler (*Google Developer Training Team*, 2016).

Google menyediakan peningkatan versi bertahap utama untuk sistem operasi Android setiap enam hingga sembilan bulan, menggunakan nama bertema makanan. Adapun versi-versi sistem operasi Android adalah seperti pada Tabel 2.4.

Tabel 2.4 Versi Android

Nama Kode	**Nomor Versi**	**Tanggal Rilis Awal**	**API Level**
N/A	1.0	23 September 2008	1
N/A	1.1	9 Febuari 2009	2
Cupkace	1.5	27 April 2009	3
Donut	1.6	15 September 2009	4

Tabel 2.4 (Lanjutan)

Nama Kode	Nomor Versi	Tanggal Rilis Awal	API Level
Eclair	2.0-2.1	26 Oktober 2009	5-7
Froyo	2.2-2.2.3	20 Mei 2010	8
Gingerbread	2.3-2.3.7	6 Desember 2010	9-10
Honeycomb	3.0-3.2.6	22 Febuari 2011	11-13
Ice Cream Sandwitch	4.0-4.0.4	18 Oktober 2011	14-15
Jelly Bean	4.1-4.3.1	9 Juli 2012	16-18
KitKat	4.4-4.4.4	31 Oktober 2013	19-20
Lolipop	6.0-6.0.1	5 Oktober 2015	23
Nougat	7.0	22 Agustus 2016	24
Oreo	8.0	21 Agustus 2017	26
Pie	9.0	16 Agustus 2018	28

Untuk membantu mengembangkan aplikasi secara efisien, Google menawarkan Lingkungan *Development* Terintegrasi (IDE) Java Lengkap yang disebut Android Studio, dengan fitur lanjutan untuk pengembangan, *debug*, dan pemaketan aplikasi Android. Pendistribusian aplikasi Android dapat dilakukan dengan berbagai cara, yaitu melalui email, situs web, atau pasar aplikasi Google Play. Google Play adalah layanan distribusi digital yang dioperasikan dan dikembangkan oleh Google dan berfungsi sebagai toko aplikasi resmi untuk Android yang memungkinkan konsumen mengunduh dan menjelajah aplikasi yang dikembangkan dengan Android SDK dan dipublikasikan melalui Google.

2.10 Android Studio

Android Studio menyediakan alat untuk pengujian, dan mempublikasikan tahap proses development, serta lingkungan development terpadu untuk membuat aplikasi bagi semua perangkat Android. Lingkungan development menyertakan kode template dengan kode contoh untuk fitur aplikasi umum, alat pengujian dan kerangka kerja yang banyak, dan sistem pembangunan yang fleksibel (*Google Developer Training Team*, 2016).

2.11 MySQL

Menurut Solichin (2016), MySQL adalah sebuah perangkat lunak sistem manajemen basis data SQL (*Database Management System*) atau DBMS yang multithread, multi-user, dengan sekitar 6 juta instalasi di seluruh dunia. MySQL AB membuat MySQL tersedia sebagai perangkat lunak gratis di bawah lisensi GNU *General Public License* (GPL), tetapi mereka juga menjual dibawah lisensi komersial untuk kasus-kasus dimana penggunaannya tidak cocok dengan penggunaan GPL. Tidak seperti Apache yang merupakan *software* yang dikembangkan oleh komunitas umum, dan hak cipta untuk kode sumber dimiliki oleh penulisnya masing-masing, MySQL dimiliki dan disponsori oleh sebuah perusahaan komersial Swedia yaitu MySQL AB. MySQL AB memegang penuh hak cipta hampir atas semua kode sumbernya. Kedua orang Swedia dan satu orang Finlandia yang mendirikan MySQL AB adalah: David Axmark, Allan Larsson, dan Michael "Monty" Widenius. Beberapa kelebihan MySQL antara lain : *free* (bebas di *download*), stabil dan tangguh, fleksibel dengan berbagai pemrograman, *security* yang baik, dukungan dari banyak komunitas, kemudahan *management database*, mendukung transaksi dan perkembangan *software* yang cukup cepat.

2.12 Definisi *Guest House*

Guest House adalah sebuah tempat yang di bangun untuk penginapan , perbedaan yang mendasar dengan hotel adalah disini biasanya para pengelola menawarkan untuk sewa kamar harian dan bulanan (Maitra dan Adawiyah, 2017).

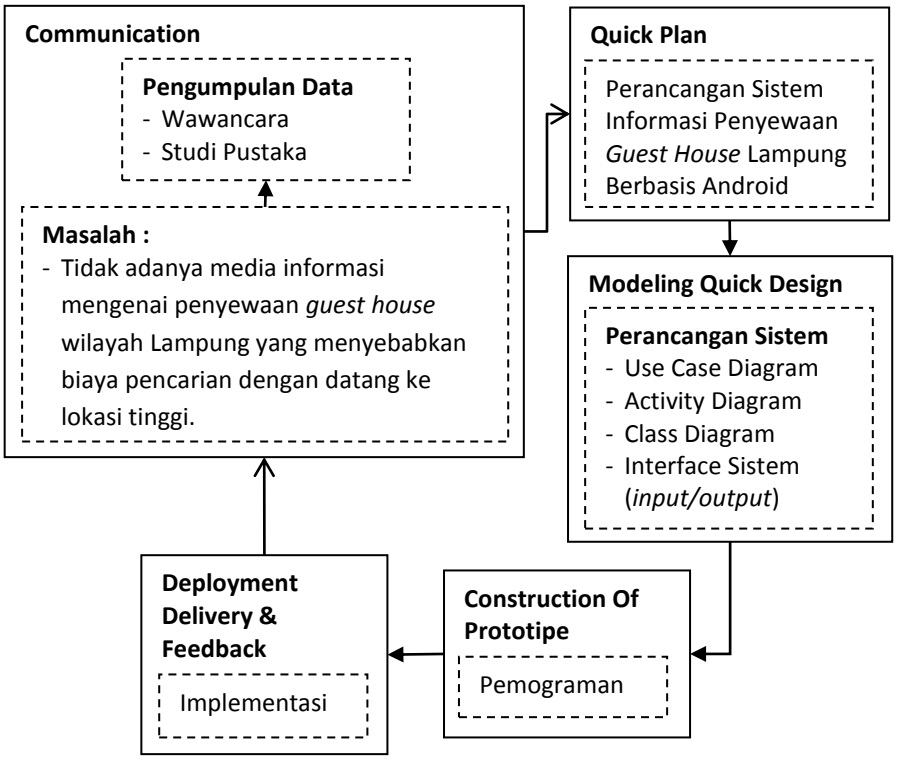
2.13 Referensi Jurnal Yang terkait dengan judul

Nama Peneliti	Judul	Tahun	Hasil Penelitian	Metode Penelitian
Rahmalia Syahputri Anni Setiani	Sistem Informasi Pemesanan Salon Online Berbasis Location Service	2019	1.Aplikasi PESONA dapat mempermudah transaksi pemesanan antara Salon dan pelanggannya. 2.Dengan adanya aplikasi ini maka salon dapat memperluas pemasaran dan Pelanggan menjadi lebih mudah mendapatkan informasi tentang jasa layanan yang di berikan tanpa harus datang ke Salon	*Protoype*
Hendra Kurniawan	Media Pembelajaran Mobile Learning Menggunakan Android	2017	Dengan menggunakan pembelajaran berbasis android mahasiswa diharapkan tidak bosan dalam kegiatan pembelajaran.	Rational Unified Process (RUP)
Melda Agarina Nurul Hikmah Afnil	Perancangan Sistem Informasi Berbasis Mobile Pada Restoran Lokal Di Bandar Lampung	2018	Perancangan Sistem Informasi Pada Restoran Lokal di Bandar Lampung berbasis mobile telah dibangun, sehingga proses booking menjadi lebih baik. b. Proses booking meja makan dan menu makanan menjadi lebih mudah	metodologi analisis dan desain terstruktur (structured system analysis and design)

BAB III
METODOLOGI PENELITIAN

3.1 Tahapan Penelitian

Sebelum memasuki tahap implementasi sistem, maka perlu diketahui tahapan-tahapan penelitian yang harus dilakukan terlebih dahulu sesuai dengan metode pengembangan sistem yang akan digunakan. Tahapan penelitian dilakukan menggunakan metode prototipe dengan 5 (lima) fase tahapan mulai dari tahap *communication* sampai ke tahap *deployment, delivery and feedback* adalah seperti pada Gambar 3.1.



Gambar 3.1 Alur Tahapan Penelitian

3.2 Metode Pengumpulan Data

Pengumpulan data yang digunakan dalam menyusun serta melengkapi data adalah dengan cara sebagai berikut :

a. Wawancara

Wawancara dilakukan dengan cara berkomunikasi dengan beberapa pemilik *guest house* (kos atau kontrakan) yang ada di area Bandar Lampung guna mendapatkan data yang diinginkan sebagai bahan untuk perancangan atau pembuatan sistem dalam penelitian ini.

b. Studi Pustaka

Studi pustaka dilakukan untuk memperoleh data dan informasi dengan membaca berbagai bahan penulisan karangan ilmiah serta sumber-sumber lain mengenai permasalahan yang berhubungan dengan penulisan.

3.3 Metode Pengembangan Sistem

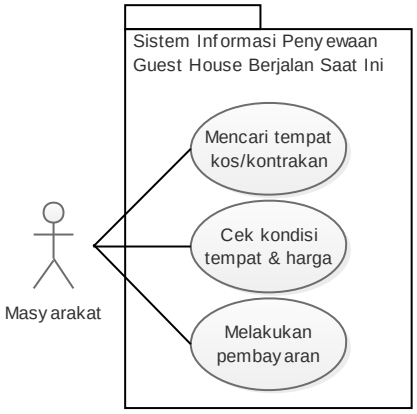
Metode pengembangan sistem yang digunakan dalam pembuatan sistem ini adalah prototipe yang terdiri dari 5 (lima) fase tahapan yaitu *communication, quick plan, modeling quick design, construction of prototype,* dan *deployment, delivery and feedback.* Adapun penjelasan dari tiap tahap tersebut dijelaskan pada sub pokok pembahasan di bawah ini.

3.3.1 Communication

Pada tahap ini, antara pelaku usaha *guest house* dengan peneliti berkomunikasi mengenai spesifikasi kebutuhan yang diinginkan. Peneliti melakukan pengumpulan data yang akan digunakan dalam pembuatan sistem yang dibutuhkan. Metode pengumpulan data yang digunakan guna mendapatkan data yang dibutuhkan adalah terdiri dari observasi dan wawancara seperti yang telah dijelaskan pada tahap pengumpulan data. Data yang diperoleh dari pengumpulan data tersebut kemudian dialasis. Adapun analisis yang didapat berupa cara mendapatkan tempat kos atau kontrakan sesuai dengan harga dan fasilitas yang diinginkan. Dari analisis sistem yang sedang berjalan saat ini, maka didapatlah analisis suatu permasalahan.

3.3.1.1 Analisa Sistem Berjalan

Analisis sistem yang sedang berjalan saat ini digambarkan dengan *use case diagram* seperti pada Gambar 3.2. Cara mendapatkan tempat kos atau kontrakan sesuai dengan harga dan fasilitas yang diinginkan dilakukan dengan datang langsung ke lokasi yang dikehendaki.



Gambar 3.2 Penyewaan *Guest House* Sistem Berjalan Saat Ini

3.3.1.2 Analisa Permasalahan

Dari pencarian penyewaan *guest house* sistem yang sedang berjalan saat ini, maka didapati suatu permasalahan dimana dengan tidak adanya media informasi penyewaan *guest house* menyebabkan tingginya biaya pencarian.

3.3.2 Quick Plan

Dari analisis permasalahan yang telah diuraikan sebelumnya, maka untuk mengurangi permasalahan yang terjadi ketika melakukan pencarian penyewaan *guest house* pada penelitian ini, peneliti mengusulkan suatu sistem informasi penyewaan *guest house* berbasis Android.

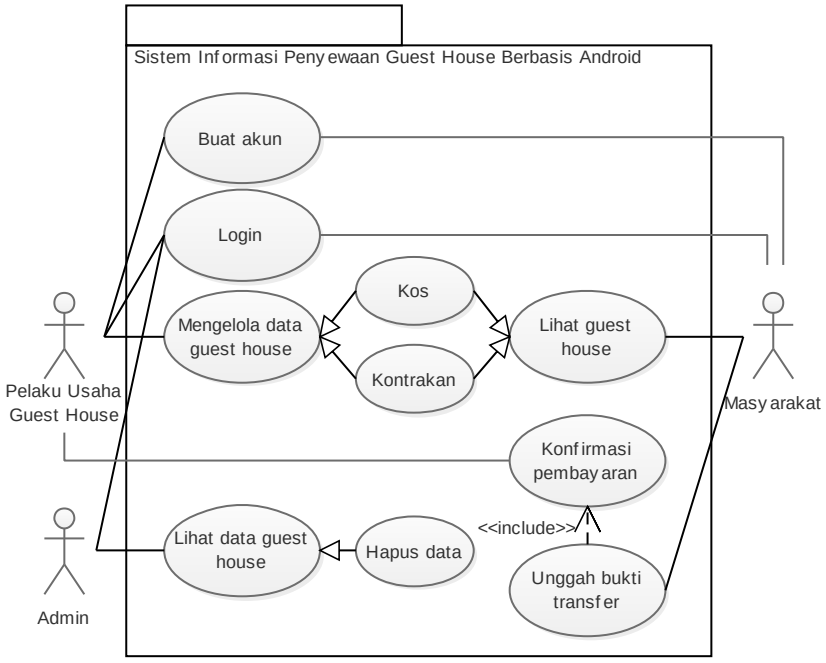
3.3.3 Modeling Quick Design

Setelah didapat ide mengenai sistem yang akan dibuat, maka pada tahap ini peneliti membuat perancangan/pemodelan sistem. Adapun perancangan sistem diusulkan terdiri dari perancangan *use case diagram, activity diagram, class*

diagram dan *interface* sistem. Perancangan ini dibuat menggunakan aplikasi *Edraw Max.*

3.3.3.1 Use Case Diagram

Use case diagram mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. Perancangan *use case diagram* sistem informasi penyewaan *guest house* yang diusulkan adalah seperti pada Gambar 3.3. Sistem yang diusulkan terdiri dari 3 (tiga) aktor, yaitu admin, para pelaku usaha *guest house*, dan masyarakat. Pelaku usaha dan masyarakat harus membuat akun terlebih dahulu untuk dapat login dan mengakses sistem. Akses sistem dari semua aktor berbeda-beda disesuaikan dengan kebutuhan.



Gambar 3.3 Perancangan *Use Case Diagram* Sistem Diusulkan

3.3.3.2 Activity Diagram

Activity diagram menggabarkan aliran kerja atau aktivitas sistem yang dibangun. Perancangan *activity diagram* sistem informasi penyewaan *guest house* berbasis Android yang diusulkan adalah sebagai berikut :

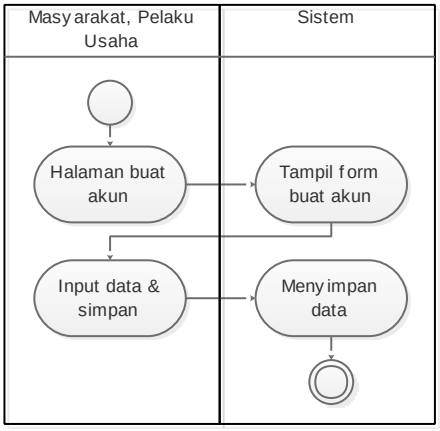
a. *Activity Diagram* Buat Akun

Activity diagram buat akun menggambarkan aktifitas pelaku usaha dan masyarakat dalam mendaftarkan diri dengan cara membuat akun. Tujuan membuat akun ini adalah untuk mendapatkan *login*. Perancangan *activity diagram* buat akun adalah seperti pada Gambar 3.4.

Nama *Use Case* : Buat Akun

Aktor : Pelaku Usaha dan Masyarakat

Tujuan : Mendapatkan akses login



Gambar 3.4 Perancangan *Activity Diagram* Buat Akun

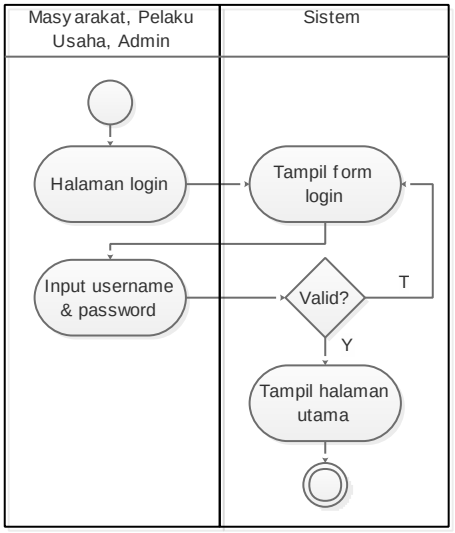
b. *Activity Diagram Login*

Activity diagram login menggambarkan aktifitas pelaku usaha, admin, dan masyarakat dalam melakukan login. Tujuan membuat *login* adalah untuk dapat mengakses sistem sesuai dengan hak akses masing-masing. Perancangan *activity diagram login* adalah seperti pada Gambar 3.5.

Nama *Use Case* : *Login*

Aktor : Pelaku Usaha, Admin, dan Masyarakat

Tujuan : Dapat mengakses sistem



Gambar 3.5 Perancangan *Activity Diagram Login*

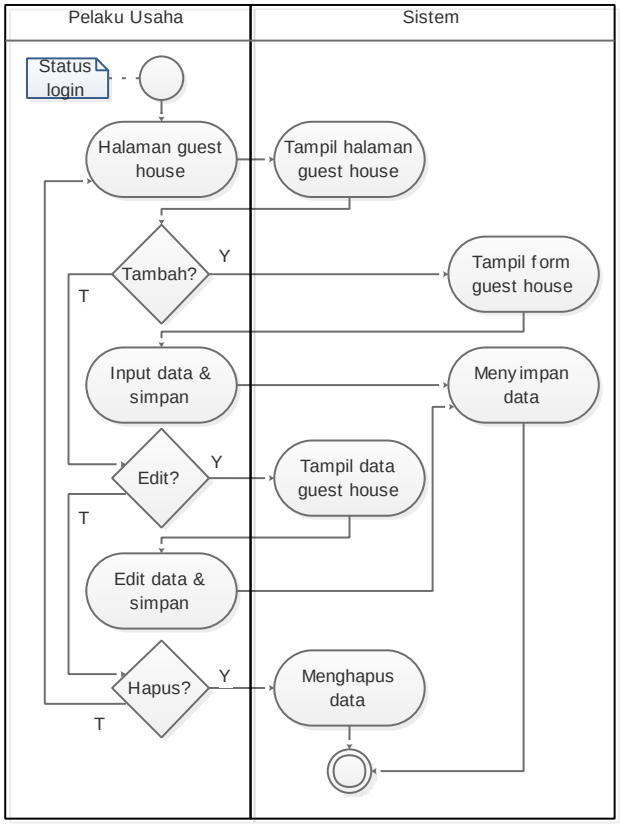
c. *Activity Diagram* Mengelola Data *Guest House*

Activity diagram mengelola data *guest house* menggambarkan aktifitas pelaku usaha dalam mengelola data *guest house* (kontrakan atau kos). Perancangan *activity diagram* mengelola data *guest house* adalah seperti pada Gambar 3.6.

Nama *Use Case* : Mengelola Data *Guest House*

Aktor : Pelaku Usaha

Tujuan : Mengolah data *guest house* (kontrakan atau kos)



Gambar 3.6 Perancangan *Activity Diagram* Mengelola Data *Guest House*

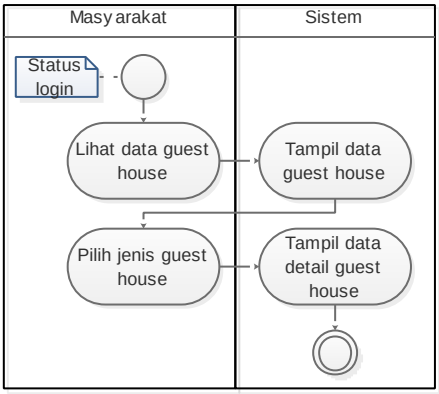
d. *Activity Diagram* Lihat *Guest House*

Activity diagram lihat *guest house* menggambarkan aktifitas masyarakat dalam melihat data *guest house* (kontrakan atau kos). Perancangan *activity diagram* lihat *guest house* adalah seperti pada Gambar 3.7.

Nama *Use Case* : Lihat *Guest House*

Aktor : Masyarakat

Tujuan : Melihat *guest house* (kontrakan atau kos) yan tersedia



Gambar 3.7 Perancangan *Activity Diagram* Lihat *Guest House*

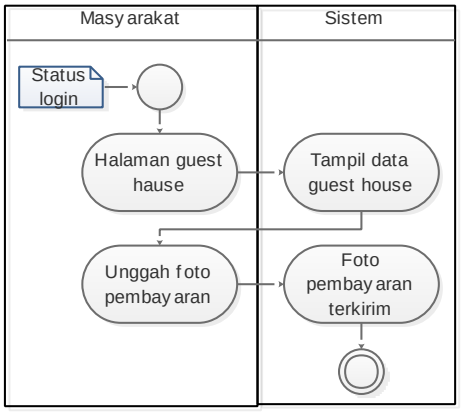
e. *Activity Diagram* Unggah Bukti Transfer

Activity diagram unggah bukti transfer menggambarkan aktifitas masyarakat dalam mengunggah bukti transfer. Perancangan *activity diagram* unggah bukti transfer adalah seperti pada Gambar 3.8.

Nama *Use Case* : Unggah Bukti Transfer

Aktor : Masyarakat

Tujuan : Mengunggah foto bukti transfer pembayaran



Gambar 3.8 Perancangan *Activiy Diagram* Unggah Bukti Transfer

f. *Activity Diagram* Konfirmasi Pembayaran

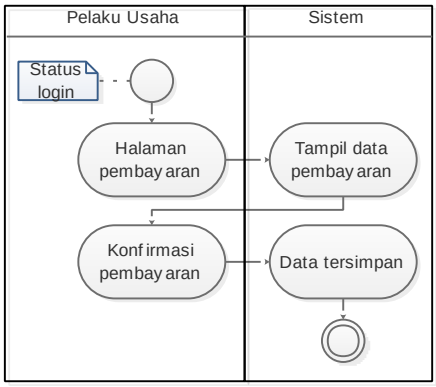
Activity diagram konfirmasi pembayaran menggambarkan aktifitas pelaku usaha dalam konfirmasi bukti transfer yang telah dilakukan oleh masyarakat

dalam melakukan pembayaran. Perancangan *activity diagram* konfirmasi pembayaran adalah seperti pada Gambar 3.9.

Nama *Use Case* : Konfirmasi Pembayaran

Aktor : Pelaku Usaha

Tujuan : Melakukan konfirmasi bukti transfer yang dilakukan oleh masyarakat sebelumnya



Gambar 3.9 Perancangan *Activity Diagram* Konfirmasi Pembayaran

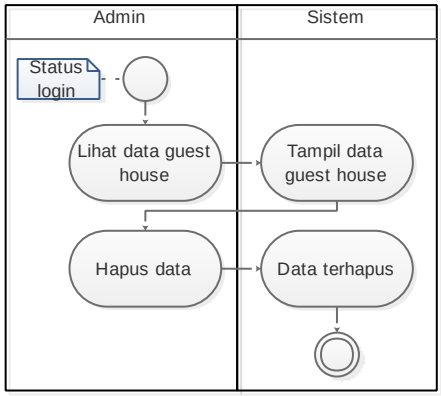
g. *Activity Diagram* Lihat Data *Guest House*

Activity diagram lihat data *guest house* menggambarkan aktifitas admin dalam mengolah data *guest house*. Admin disini juga dapat melakukan penghapusan *guest house*. Perancangan *activity diagram* lihat data *guest house* adalah seperti pada Gambar 3.10

Nama *Use Case* : Lihat Data *Guest House*

Aktor : Admin

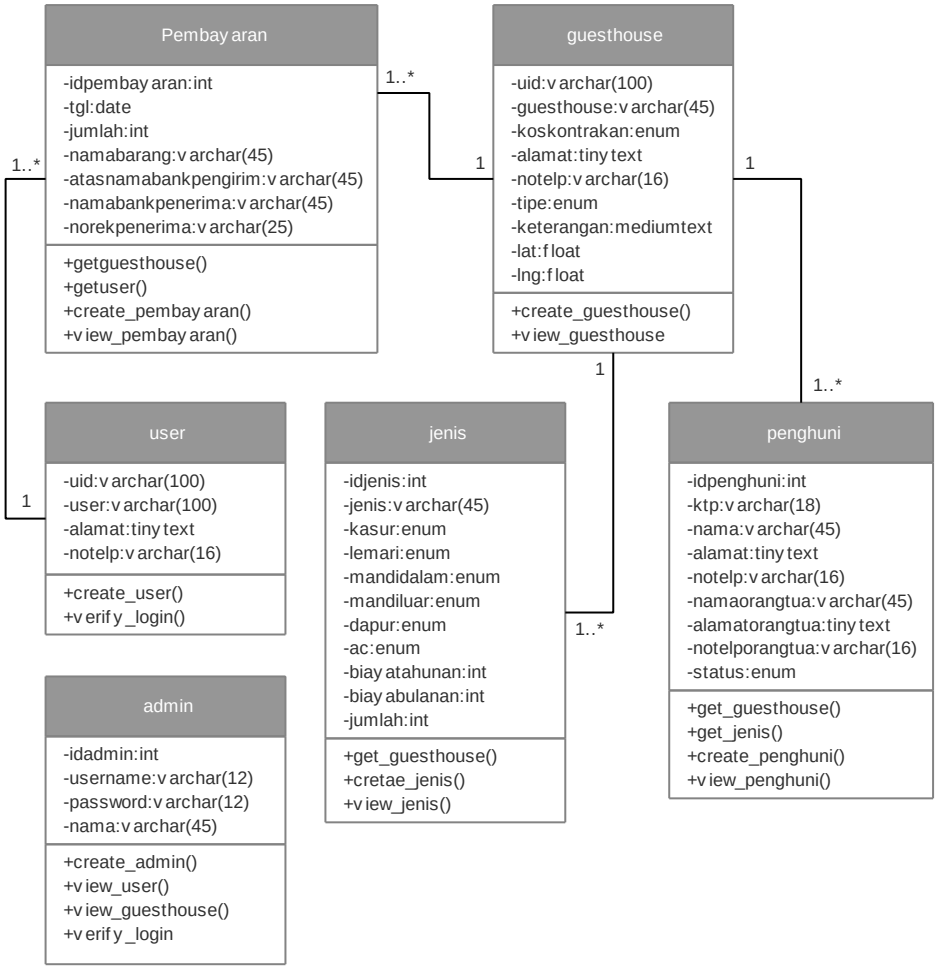
Tujuan : Melihat dan menghapus data *guest house*



Gambar 3.10Perancangan *Activity Diagram* Lihat Data *Guest House*

3.3.3.3 Class Diagram

Perancangan *class diagram* menggambarkan aktivitas sistem dalam penyimpanan data berdasarkan kelas-kelas data yang disinpan yang terkoneksi satu sama lain, sehingga dalam pemrosesan data yang dilakukan oleh sistem dapat dilakukan dengan baik. Adapun perancangan *class diagram* pada sistem yang diusulkan adalah seperti pada Gambar 3.11.



Gambar 3.11 Perancangan *Class Diagram*

3.3.3.4 *Interface* Sistem

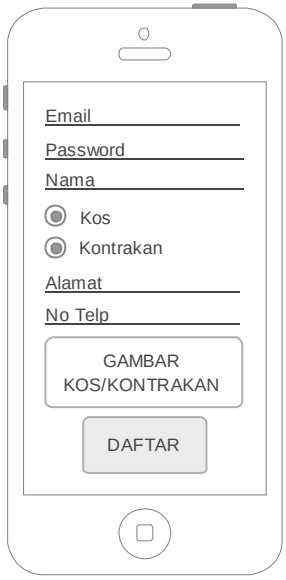
Interface sistem diusulkan terdiri dari 2 (dua) hak akses, yaitu *interface* sistem dengan hak akses masyarakat (pengguna), dan *interface* sistem hak akses pelaku usaha. Sistem yang akan dibangun yang dipergunakan oleh semua hak akses berbasiskan Android.

3.3.3.4.1 *Interface* Sistem Akses Pelaku Usaha

Perancangan *interface* sistem diusulkan dengan hak akses pelaku usaha adalah sebagai berikut :

a. *Interface* Daftar Akun

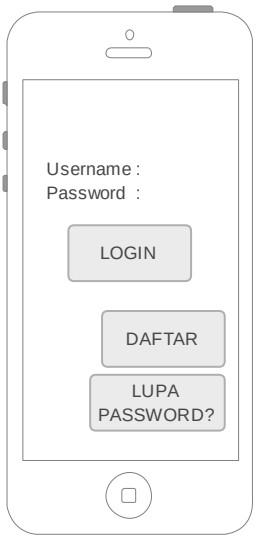
Perancangan daftar akun yang digunakan untuk hak akses pelaku usaha sebagai pemilik usaha kos/kontrakan adalah seperti pada Gambar 3.12.



Gambar 3.12 Perancangan *Interface* Daftar Akun Akses Pelaku Usaha

b. *Interface Login*

Setelah mendaftar akun, pelaku usaha dapat *login* pada halaman *login* dengan memasukkan *username* dan *password*. Perancangan *interface login* akses pelaku usaha adalah seperti pada Gambar 3.13.



Gambar 3.13 Perancangan *Interface Login* Akses Pelaku Usaha

c. *Interface* Halaman Utama

Peracangan halaman utama hak akses pelaku usaha setelah berhasil *login* adalah seperti pada Gambar 3.14.

Gambar 3.14 Perancangan *Interface* Halaman Utama Akses Pelaku Usaha

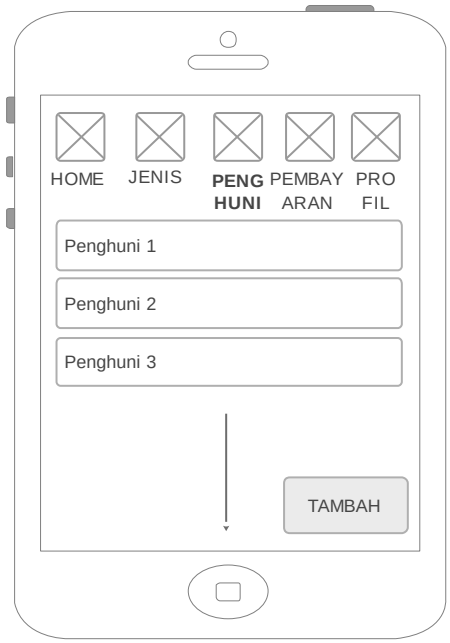
d. *Interface* Jenis

Perancangan *interface* jenis dalam arti jenis usaha (kos/kontrakan) yang dikelola oleh pelaku usaha adalah seperti pada Gambar 3.15.

Gambar 3.15 Perancangan *Interface* Halaman Jenis Akses Pelaku Usaha

e. *Interface* Penghuni

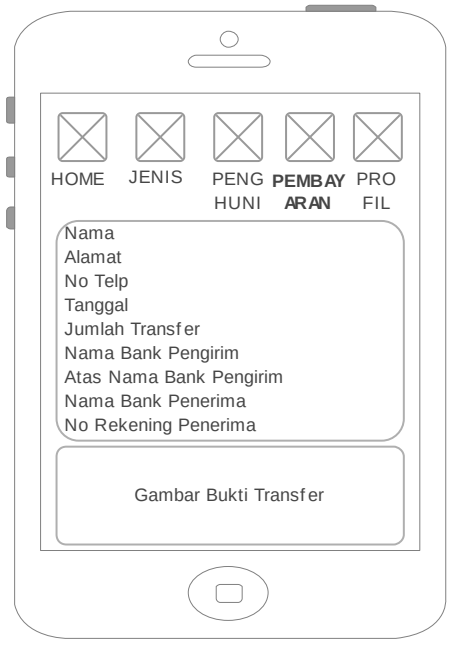
Perancangan *interface* penghuni dibuat untuk dipergunakan oleh pelaku usaha untuk mengolah data penghuni. Perancangan *interface* penghuni adalah seperti pada Gambar 3.16.



Gambar 3.16 Perancangan *Interface* Halaman Penghuni Akses Pelaku Usaha

f. *Interface* Pembayaran

Perancangan *interface* pembayaran dibuat untuk dipergunakan oleh pelaku usaha untuk mengolah data pembayaran yang dilakukan oleh masyarakat (pengguna). Perancangan *interface* pembayaran adalah seperti pada Gambar 3.17.



Gambar 3.17 Perancangan *Interface* Halaman Pembayaran Akses Pelaku Usaha

g. *Interface* Profil

Perancangan *interface* profil dibuat untuk dipergunakan oleh pelaku usaha untuk mengolah data profil pelaku usaha. Perancangan *interface* profil adalah seperti pada Gambar 3.18.

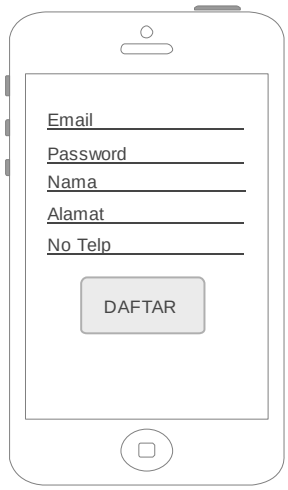
Gambar 3.18 Perancangan *Interface* Halaman Profil Akses Pelaku Usaha

3.3.3.4.2 *Interface* Sistem Akses Masyarakat (Pengguna)

Perancangan *interface* sistem diusulkan dengan hak akses masyarakat (pengguna) adalah sebagai berikut :

a. *Interface* Daftar Akun

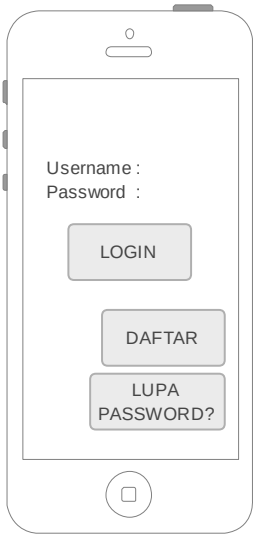
Perancangan daftar akun yang digunakan untuk hak akses masyarakat sebagai pengguna aplikasi atau sistem adalah seperti pada Gambar 3.19.



Gambar 3.19 Perancangan *Interface* Halaman Daftar Akun Akses Masyarakat (Pengguna)

b. *Interface Login*

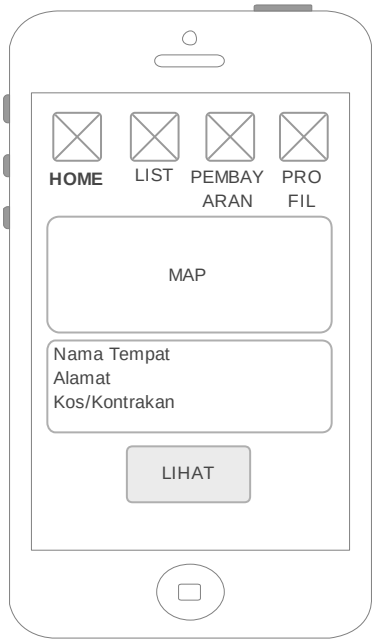
Setelah mendaftar akun, masyarakat (pengguna) dapat *login* pada halaman login dengan memasukkan *username* dan *password*. Perancangan *interface login* akses masyarakat (pengguna) adalah seperti pada Gambar 3.20



Gambar 3.20. Perancangan *Interface* Halaman *Login* Akses Masyarakat (Pengguna)

c. *Interface* Halaman Utama

Peracangan halaman utama hak akses masyarakat (pengguna) setelah berhasil *login* adalah seperti pada Gambar 3.21.



Gambar 3.21 Perancangan *Interface* Halaman Utama Akses Masyarakat (Pengguna)

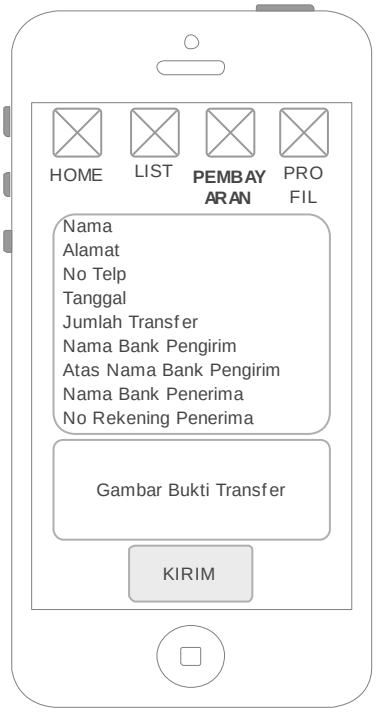
d. Interface List

Perancangan *interface* list dibuat untuk dipergunakan oleh masyarakat (pengguna) untuk melihat tempat kos/kontrakan sesuai dengan jenis fasilias yang diinginkan. Perancangan *interface list* adalah seperti pada Gambar 3.22.

Gambar 3.22 Perancangan *Interface* Halaman *List* Akses Masyarakat (Pengguna)

e. *Interface* Pembayaran

Perancangan *interface* pembayaran dibuat untuk dipergunakan oleh masyarakat (pengguna) untuk mengolah data pembayaran. Perancangan *interface* pembayaran adalah seperti pada Gambar 3.23.



Gambar 3.23 Perancangan *Interface* Halaman Pembayaran Akses Masyarakat (Pengguna)

f. *Interface* Profil

Perancangan *interface* profil dibuat untuk dipergunakan oleh masyarakat (pengguna) untuk mengolah data profil. Perancangan *interface* profil adalah seperti pada Gambar 3.24.

Gambar 3.24 Perancangan *Interface* Halaman Profil Akses Masyarakat (Pengguna)

3.3.4 Construction Of Prototipe

Setelah tahap pemodelan, maka peneliti mulai melakukan pengkodean program. Bahasa pemograman yang digunakan adalah *Java* menggunakan *tools* Android Studio dengan *database* MySQL.

3.3.5 *Deployment, Delivery & Feedback*

Tahap pengkodean program dibarengi oleh tahapan implementasi dan pengujian sistem. Sistem yang baru dapat digunakan oleh para pelaku usaha *guset house* sebagai wadah media promosi. Jika ada kekurangan atau penambahan kebutuhan sistem, maka para pelaku usaha *guest house* akan mengkomunikasikan kembali dengan peneliti.

BAB IV
HASIL PENELITIAN DAN PEMBAHASAN

4.1 Alat Pendukung Pembuatan Sistem

Alat pendukung pembuatan sistem informasi penyewaan *guest house* berbasis Android terdiri dari beberapa perangkat lunak dan perangkat keras. Adapun alat pendukung pembuatan sistem yaitu sebagai berikut :

4.1.1 Perangkat Lunak (*Software*)

Perangkat lunak yang dibutuhkan sistem informasi penyewaan *guest house* berbasis Android adalah sebagai berikut :

a. Sistem Operasi : *Windows* 10 64bit dan Android Versi 5.0 (Lollipop).
b. *Database* : *MySQL.*
c. *Tools* : *Android Studio, Edraw Max, Xampp (MySQL), Ms. Word* dan *Notepad.*

4.1.2 Perangkat Keras (*Hardware*)

Perangkat keras dengan rekomendasi minimum yang digunakan dalam pembuatan sistem informasi penyewaan *guest house* berbasis Android adalah sebagai berikut:

a. Komputer atau laptop, dengan spesifikasi minimal :
    1. *Processor Core i3.*
    2. *Hardisk* 500 GB.
    3. RAM 4 GB.
b. *Smartphone* Android.

4.2 Implementasi Sistem

Implementasi dari sistem yang telah dirancang sebelumnya adalah terdiri dari 3 (tiga) hak akses yaitu hak akses yaitu Pelaku Usaha, masyarakat (pengguna), dan admin. Adapun penjelasan implementasi sistem tiap akses adalah sebagai berikut.

4.2.1 Implementasi Sistem Akses Pelaku Usaha

Implementasi dari sistem yang telah dirancang sebelumnya dengan hak akses pelaku usaha adalah sebagai berikut :

a. Implementasi Halaman Daftar Akun

Implementasi halaman daftar akun yang digunakan oleh pelaku usaha sebagai pemilik usaha kos/kontrakan adalah seperti pada Gambar 4.1. Pelaku usaha harus mendaftar dengan membuat akun terlebih dahulu dengan mengisikan data sesuai dengan yang tertera pada halaman tersebut.

Gambar 4.1 Implementasi Halaman Daftar Akun Akses Pelaku Usaha

b. Implementasi Halaman *Login*

Setelah mendaftar akun, pelaku usaha dapat *login* pada halaman *login* dengan memasukkan *username* dan *password*. Di dalam halaman ini terdapat tombol "daftar", dan "lupa *password*". Implementasi halaman *login* akses pelaku usaha adalah seperti pada Gambar 4.2.

Gambar 4.2 Implementasi Halaman *Login* Akses Pelaku Usaha

c. Implementasi Halaman Utama Pelaku Usaha

Ketika berhasil login, sistem menampilkan halaman utama akses pelaku usaha seperti pada Gambar 4.3. Halaman ini terdapat beberapa *icon* tombol seperti "*home*", "jenis", "penghuni", "pembayaran", "profil", dan "*chat*". Halaman ini juga berisikan informasi mengenai jumlah penghuni, sisa kamar, map lokasi tempat (kos/kontrakan), dan informasi data tempat (kos/kontrakan).

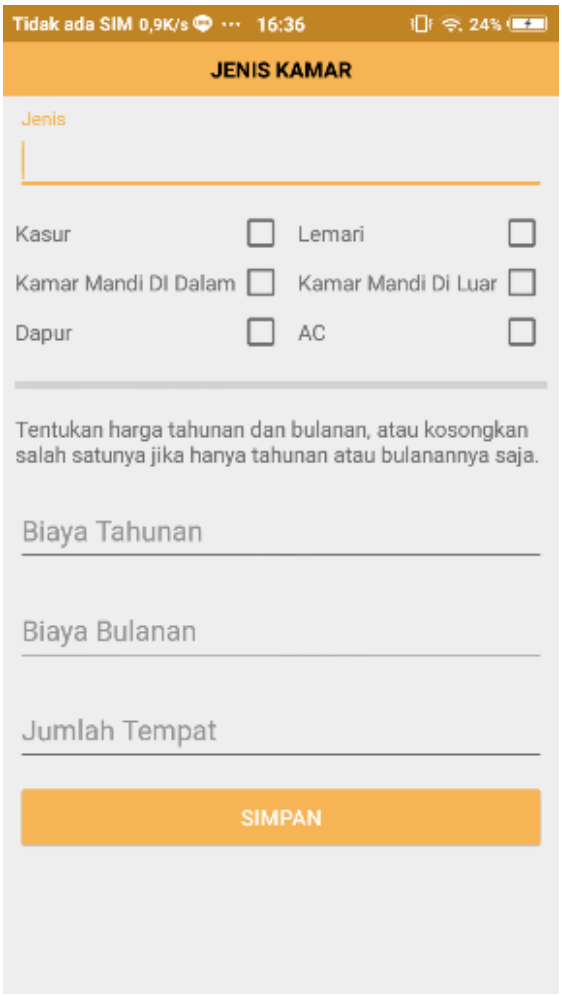
Gambar 4.3 Implementasi Halaman Utama Akses Pelaku Usaha

d. Implementasi Halaman Jenis

Halaman jenis digunakan oleh pelaku usaha dalam mengolah data jenis usaha (kos/kontrakan). Di dalam halaman ini terdapat informasi jenis usaha (kos/kontrakan), dan tombol “tambah” seperti pada Gambar 4.4.

Gambar 4.4 Implementasi Halaman Jenis Akses Pelaku Usaha

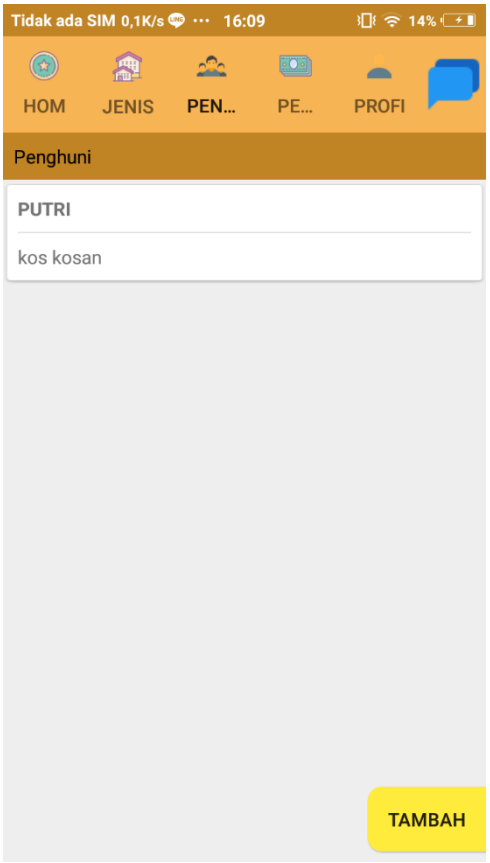
Ketika menekan tombol “tambah”, sistem menampilkan halaman *input* data jenis usaha seperti pada Gambar 4.5.



Gambar 4.5 Implementasi Halaman *Input* Data Jenis Usaha Akses Pelaku Usaha

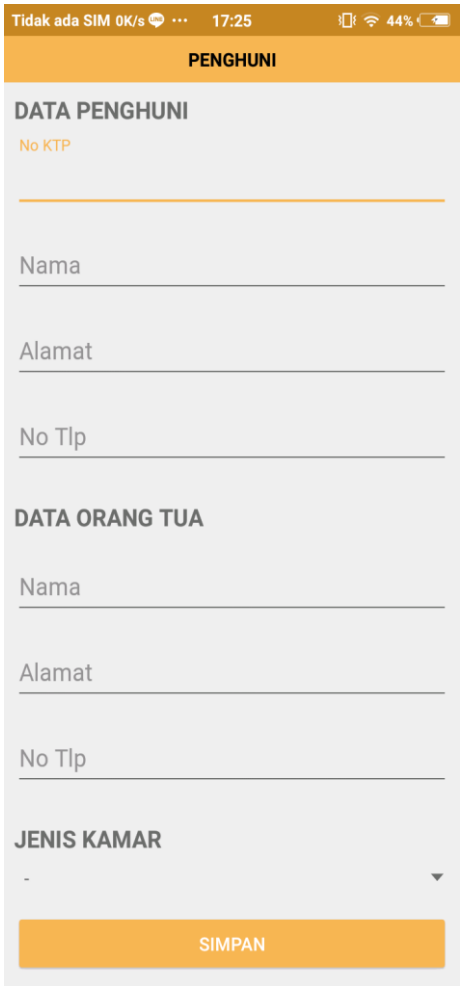
e. Implementasi Halaman Penghuni

Halaman penghuni dipergunakan oleh pelaku usaha untuk mengolah data penghuni. Implementasi halaman penghuni adalah seperti pada Gambar 4.6. Di dalam halaman ini terdapat informasi berupa *list* data penghuni, dan tombol "tambah".



Gambar 4.6 Implementasi Halaman Penghuni Akses Pelaku Usaha

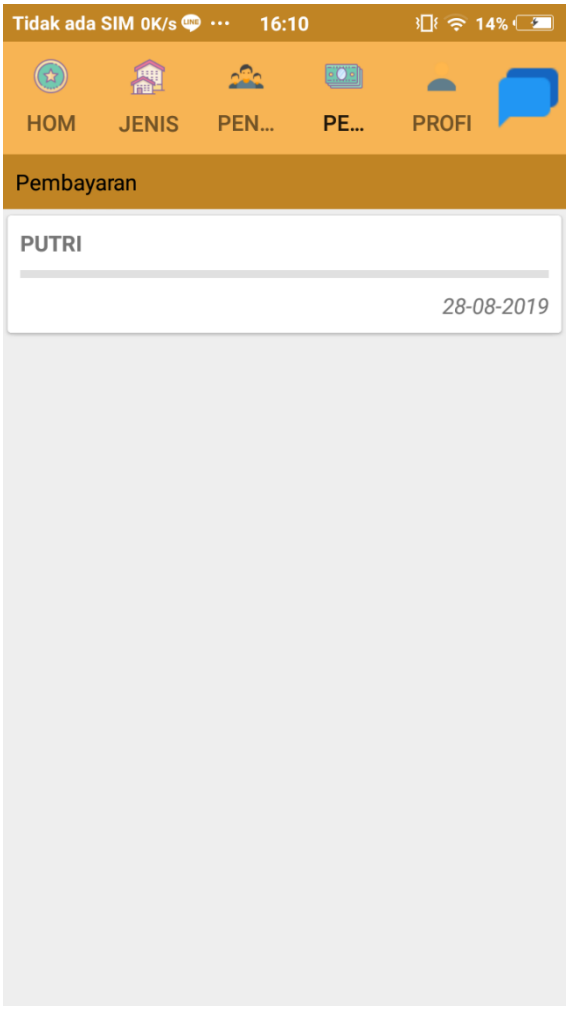
Ketika menekan tombol “tambah”, sistem menampilkan halaman *input* data penghuni seperti pada Gambar 4.7.



Gambar 4.7 Implementasi Halaman *Input* Data Penghuni Akses Pelaku Usaha

f. Implementasi Halaman Pembayaran

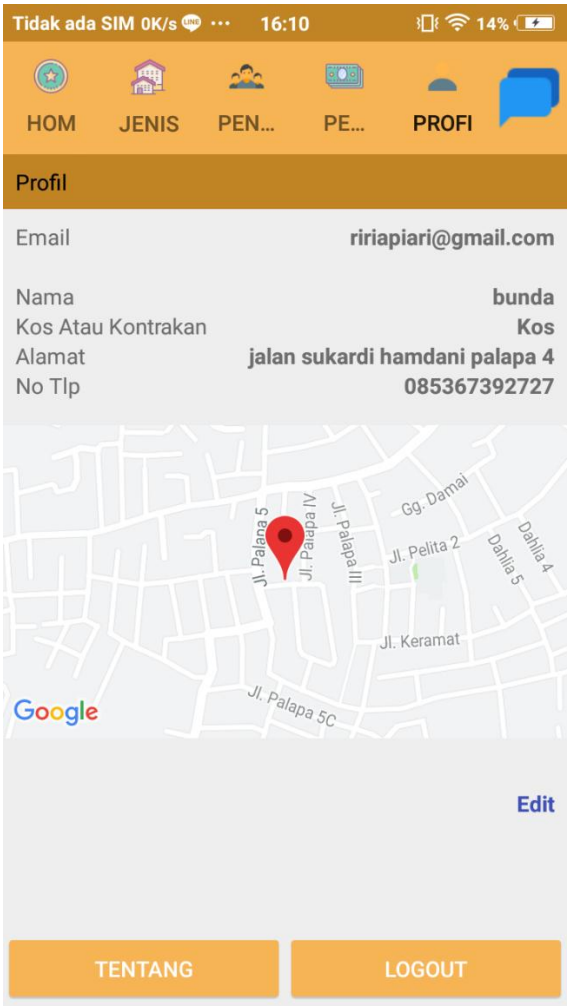
Implementasi halaman pembayaran dibuat untuk dipergunakan oleh pelaku usaha untuk melihat data pembayaran yang dilakukan oleh masyarakat (pengguna). Implementasi pembayaran akses pelaku usaha adalah seperti pada Gambar 4.8.



Gambar 4.8 Implementasi Halaman Pembayaran Akses Pelaku Usaha

g. Implementasi Halaman Profil

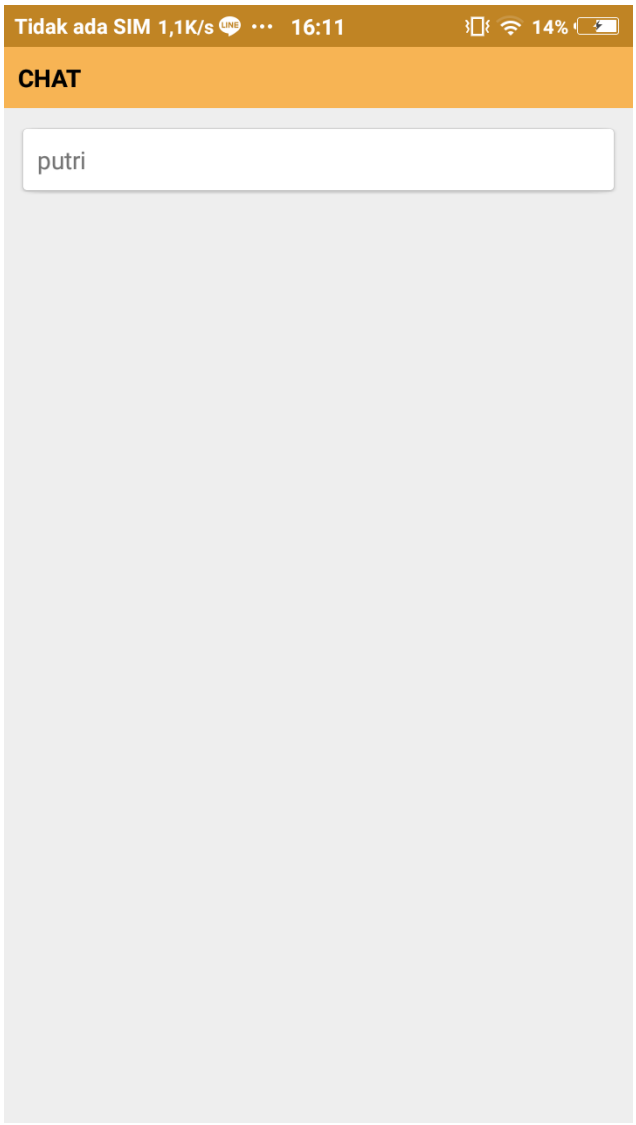
Implementasi halaman profil dipergunakan oleh pelaku usaha untuk mengolah data profil pelaku usaha. Di dalam halaman ini terdapat tombol "edit", "tentang", dan "*logout*" seperti pada Gambar 4.9.



Gambar 4.9 Implementasi Halaman Profil Akses Pelaku Usaha

h. Implementasi Halaman *Chat*

Halaman ini digunakan oleh pelaku usaha dalam berkomunikasi melalui pesan singkat (*chat*) kepada penghuni. Adapun implementasi halaman chat akses pelaku usaha adalah seperti pada Gambar 4.10.



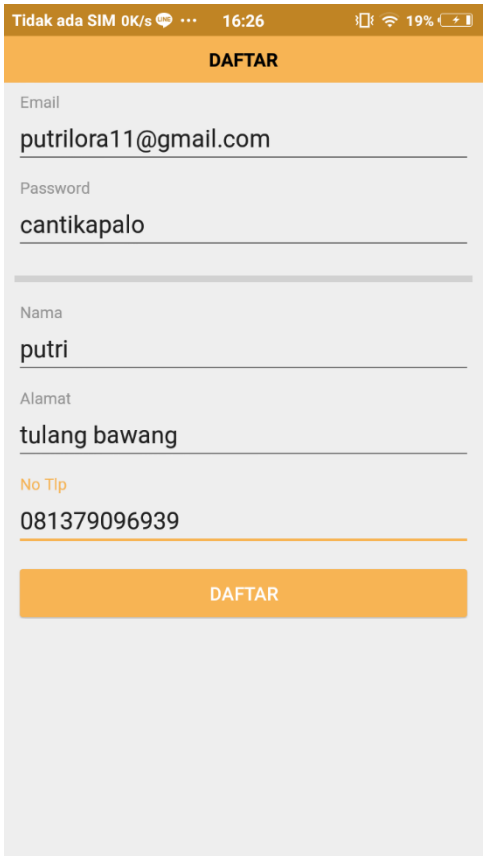
Gambar 4.10 Implementasi Halaman *Chat* Akses Pelaku Usaha

4.2.2 Implementasi Sistem Akses Masyarakat (Pengguna)

Implementasi dari sistem yang telah dirancang sebelumnya dengan hak akses masyarakat (pengguna) adalah sebagai berikut :

a. Implementasi Daftar Akun

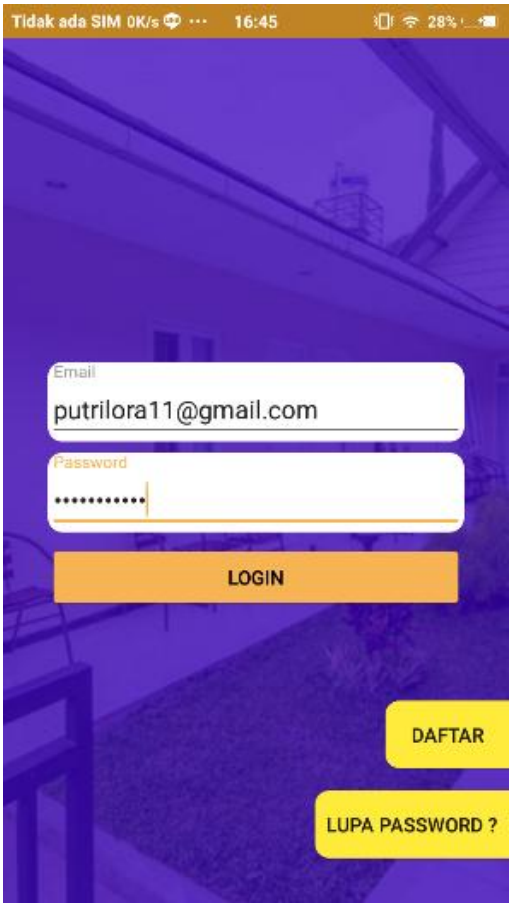
Halaman daftar akun digunakan oleh masyarakat dalam melakukan pendaftaran akun dengan tujuan agar dapat login dan mengakses sistem. Adapun implementasi dari halaman daftar akun akses masyarakat adalah seperti pada Gambar 4.11.



Gambar 4.11 Implementasi Halaman Daftar Akun Akses Masyarakat (Pengguna)

b. Implementasi Halaman *Login*

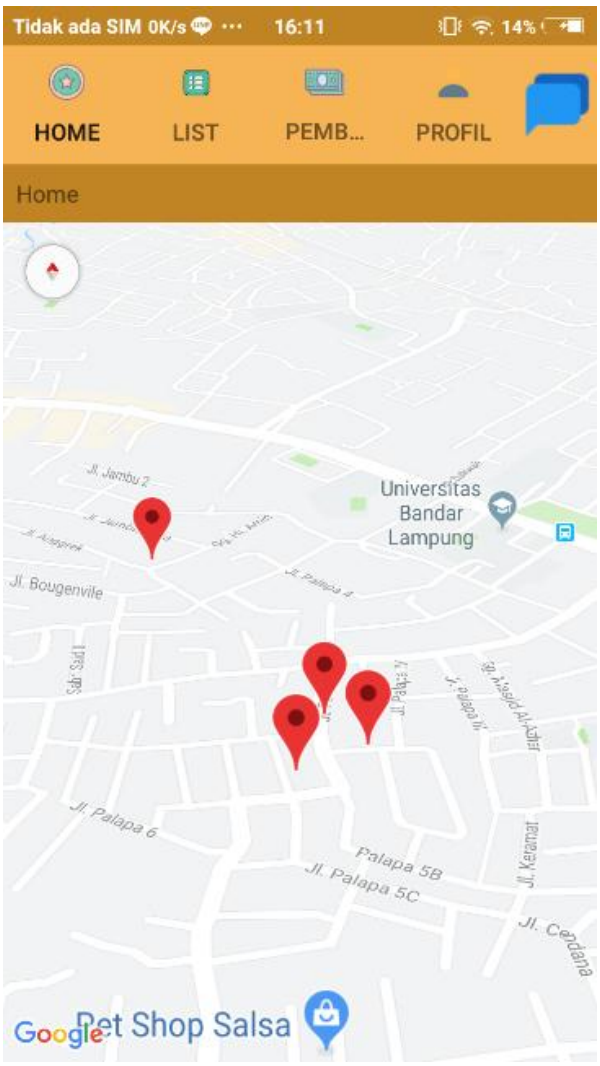
Setelah mendaftar akun, masyarakat (pengguna) dapat *login* pada halaman *login* dengan memasukkan *username* dan *password*. Di dalam halaman ini terdapat tombol "daftar", dan "lupa *password*". Implementasi halaman *login* akses masyarakat (pengguna) adalah seperti pada Gambar 4.12.



Gambar 4.12 Implementasi Halaman *Login* Akses Masyarakat (Pengguna)

c. Implementasi Halaman Utama

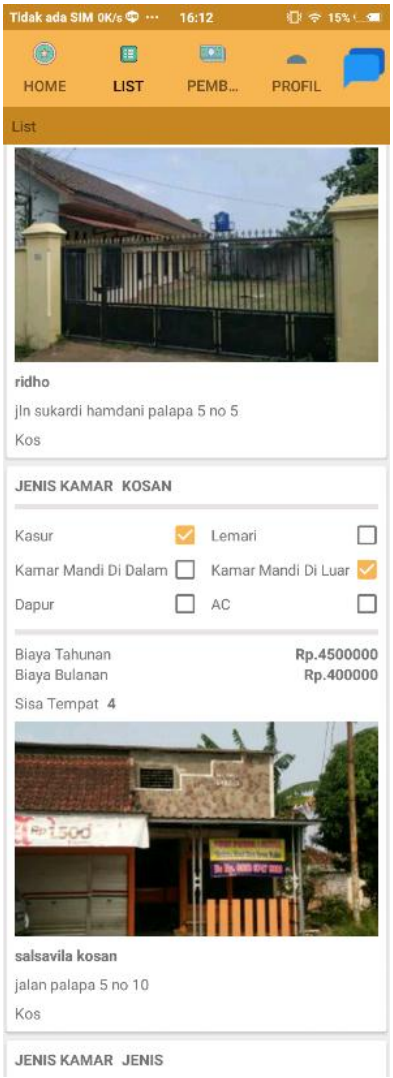
Ketika berhasil *login*, sistem menampilkan halaman utama akses masyarakat seperti pada Gambar 4.13. Di dalam halaman ini terdapat beberapa *icon* tombol menu, seperti "*home*", "list", "pembayaran", "profil", dan "*chat*". Halaman ini juga berisikan informasi mengenai titik lokasi *guest house* (kos/kontrakan).



Gambar 4.13 Implementasi Halaman Utama Akses Masyarakat (Pengguna)

d. Implementasi Halaman *List*

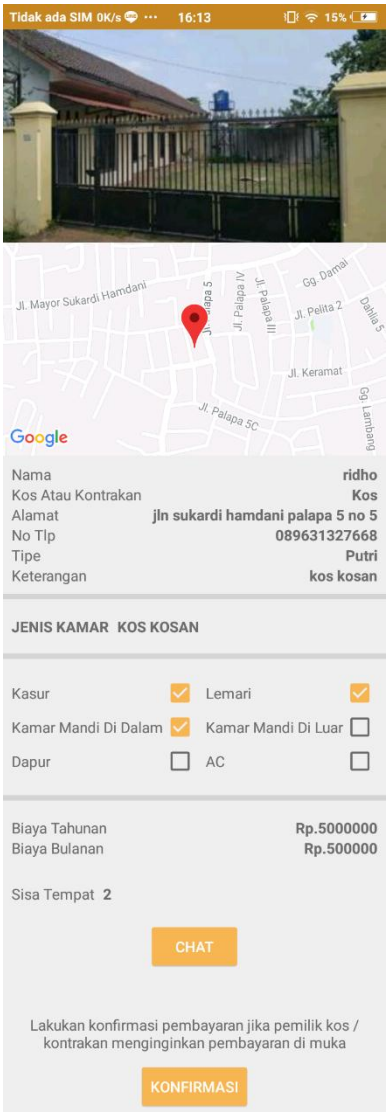
Halaman *list* dapat dipergunakan oleh masyarakat (pengguna) untuk melihat tempat kos/kontrakan sesuai dengan jenis fasilias yang diinginkan seperti pada Gambar 4.14.



Gambar 4.14 Implementasi Halaman *List* Akses Masyarakat (Pengguna)

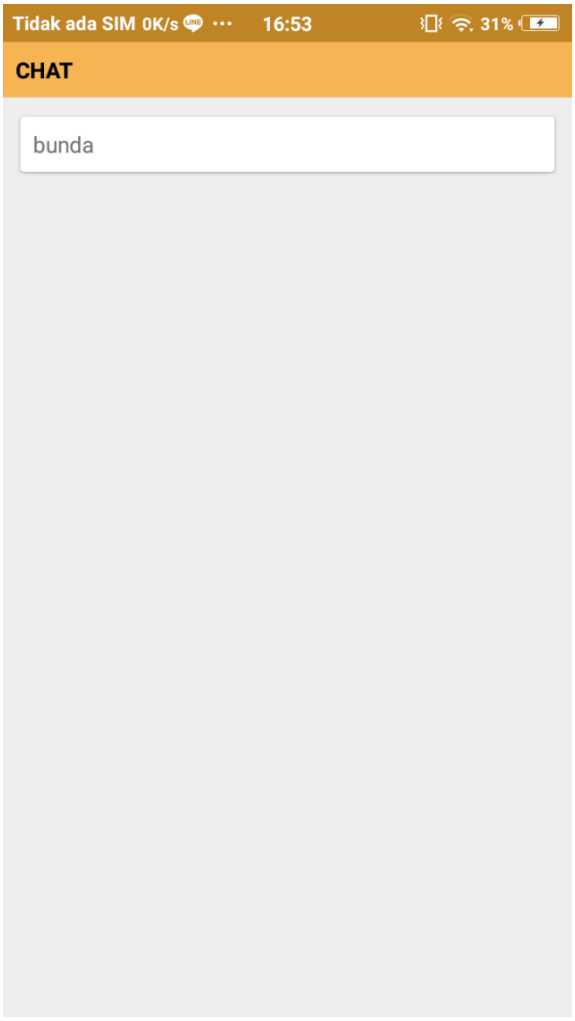
Ketika pengguna menekan salah satu *list* tempat kos/kontrakan, sistem menampilkan informasi detail mengenai tempat tersebut beserta lokasi dan

fasilitasnya seperti pada Gambar 4.15. Di dalam halaman ini juga terdapat tombol “chat”, dan “konfirmasi”.



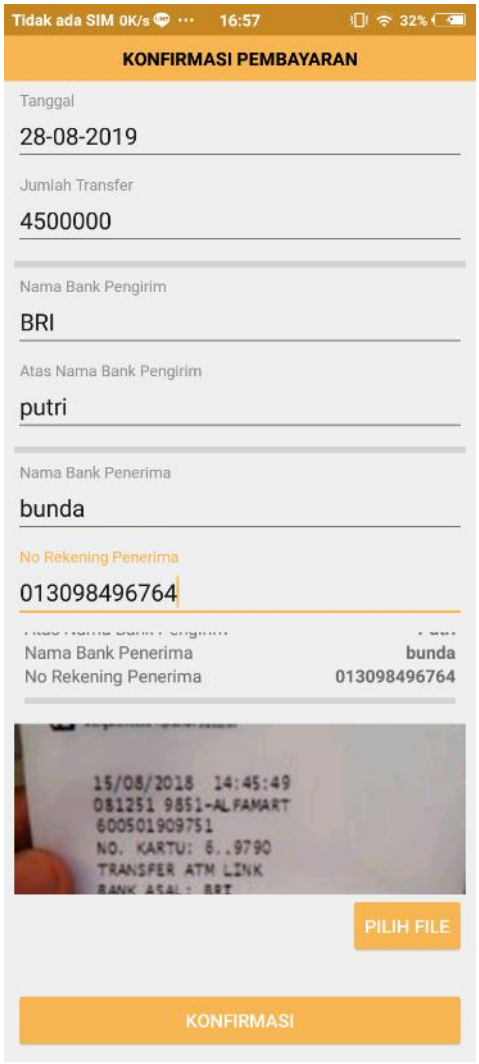
Gambar 4.15 Implementasi Halaman Informasi Detail Tempat Kos/Kontrakan Akses Masyarakat (Pengguna)

Ketika menekan tombol "*chat*" pada halaman ini, maka sistem menampilkan kolom *chat* seperti pada Gambar 4.16.



Gambar 4.16 Implementasi Halaman *Chat* Tempat Kos/Kontrakan Akses Masyarakat (Pengguna)

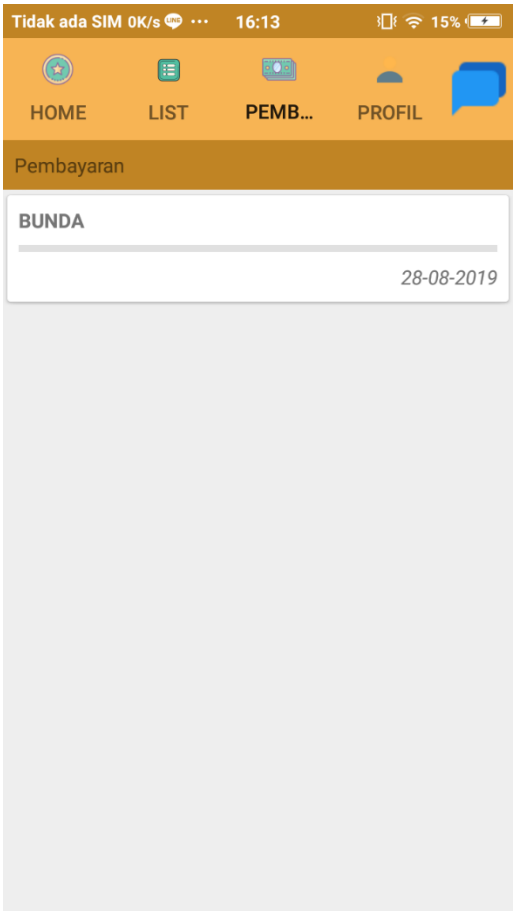
Ketika menekan tombol “konfirmasi” pada halaman ini, maka sistem menampilkan halaman *input* data konfirmasi pembayaran seperti pada Gambar 4.17.



Gambar 4.17 Implementasi Halaman Input Data Konfirmasi Pembayaran Akses Masyarakat (Pengguna)

e. Implementasi Pembayaran

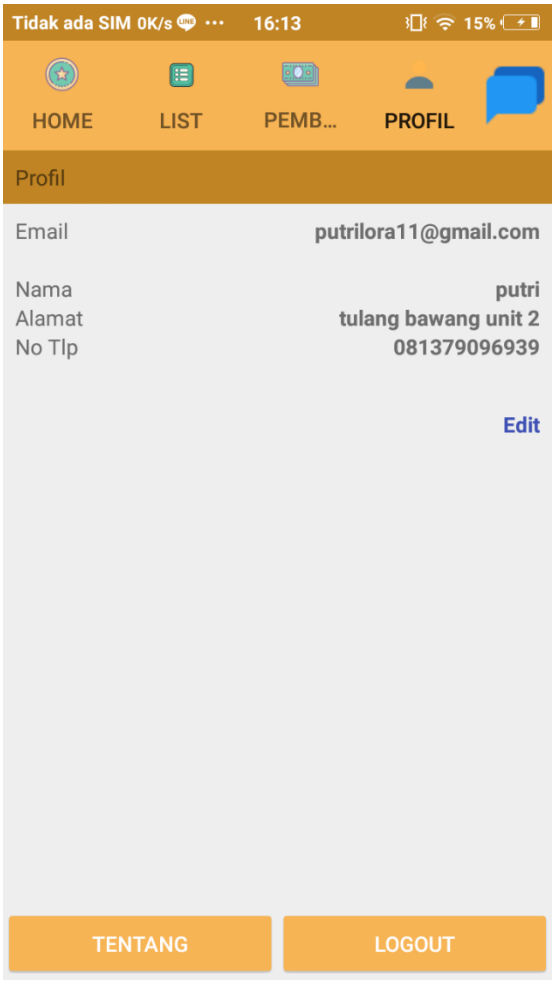
Implementasi halaman pembayaran dipergunakan oleh masyarakat (pengguna) untuk melihat data pembayaran. Implementasi halaman pembayaran akses masyarakat (pengguna) adalah seperti pada Gambar 4.18.



Gambar 4.18 Implementasi Halaman Pembayaran Akses Masyarakat (Pengguna)

f. Implementasi Halaman Profil

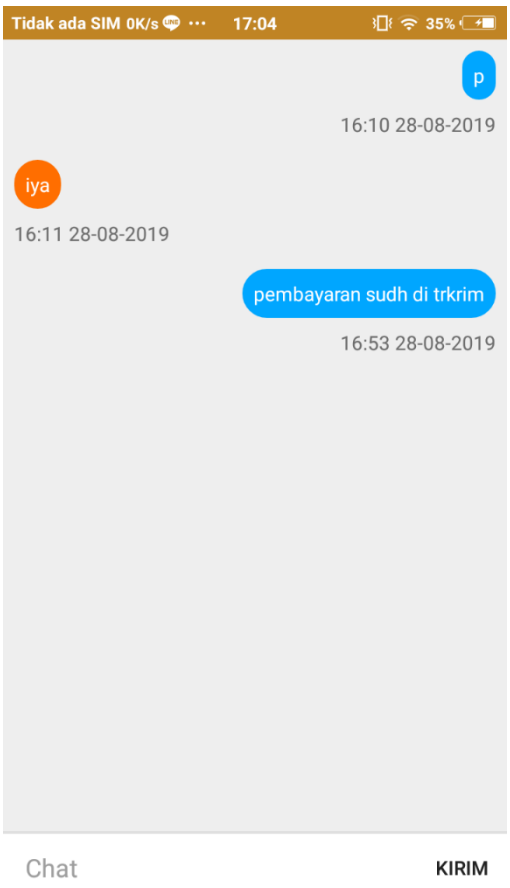
Implementasi halama profil dipergunakan oleh masyarakat (pengguna) untuk mengolah data profil. Di dalam halaman ini terdapat tombol “edit”, “tentang”, dan “*logout*” seperti pada Gambar 4.19.



Gambar 4.19 Implementasi Halaman Profil Akses Masyarakat (Pengguna)

g. Implementasi Halaman *Chat*

Halaman ini digunakan oleh masyarakat (pengguna) dalam berkomunikasi melalui pesan singkat (*chat*) kepada pelaku usaha. Adapun implementasi halaman *chat* akses masyarakat (pengguna) adalah seperti pada Gambar 4.20.



Gambar 4.20 Implementasi Halaman *Chat* Akses Masyarakat (Pengguna)

BAB V
SIMPULAN DAN SARAN

5.1 Simpulan

Adapun kesimpulan dari implementasi sistem informasi penyewaan *guest house* berbasis Android adalah sebagai berikut :

a. Sistem yang dibangun dapat memudahkan pelaku usaha dalam memberikan informasi mengenai jenis usaha (kos/kontrakan) yang dimilikinya.
b. Sistem ini dapat digunakan sebagai wadah informasi mengenai *guest house* oleh pelaku usaha karena di dalamnya terdapat lokasi tempat kos/kontrakan beserta harga dan fasilitasnya.
c. Sistem dapat memudahkan masyarakat (pengguna) dalam mendapatkan informasi mengenai *guset house* dan memudahkan dalam berkomunikasi dengan pemiliknya.

5.2 Saran

Adapun saran yang diajukan untuk penelitian yang akan dilakukan selanjutnya adalah :

a. Menambahkan ruang lingkup penelitian (Kabupaten)
b. Menambah perangkingan *guest house*.

DAFTAR PUSTAKA

A.S, Rosa., dan Shalahuddin, M. 2018. Rekayasa Perangkat Lunak Terstruktur dan Berorientasi Objek) Edisi Revisi. Informatika, Bandung.

Google Developer Training Team. 2016. *Android Developer Fundamentals Course-Learn to Develop Android Application-Concept Reference by Google Developer Training Team*.

Kadir, Abdul. 2014. Buku Pertama Belajar Pemograman Java Untuk Pemula. Mediakom, Yogyakarta.

Kadir, Abdul. 2014. Pengenalan Sistem Informasi Edisi Revisi. Andi Offset, Yogyakarta.

Maita, Idria., dan Adawiyah, Arabiatul. 2017. Sistem Informasi Reservasi Online Pada Guest House UIN Suska Riau Berbasiskan Web. Jurnal Ilmiah Rekayasa dan Manajemen Sistem Informasi, Vol. 3, No. 1, e-ISSN : 2502-8995.

Rosaldi, Dadi., dkk. 2016. Aplikasi Isstem Informasi Pencarian Tempat Kos Di Kota Bandung Berbasis Android. Jurnal *Computech* dan Bisnis, Vol. 10, No. 1, ISSN: 2442-4943.

S, Roger Pressman. 2012. Rekatyasa Perangkat Lunak. Andi, Yogyakarta.

Solichin, Achmad. 2016. Pemograman Web dengan PHP dan MySQL. *E-Book* diunduh dari https://www.researchgate.net/publication/236885805_Pemrograman_Web_dengan_PHP_dan_MySQL.

Wahyu, Kalis Herdianto., dan Cahyo, Ernes Nugroho. 2016. Sistem Informasi Rumah Kost Berbasis Android Di Wilayah AUB Surakarta. Jurnal Ilmiah *Go Infotech*, Volume. 22, No. 2, ISSN: 1693-590x.

Rahmalia Syahputri, Ani Setianni . 2019. Sistem Informasi Pemesanan Salon Online Berbasis Location Service. Jurnal Sinergitas Multidisiplin Ilmu Pengetahuan dan Teknologi, Volume. 2, No. 2, ISSN: 2622-0520.

Hendra Kurniawan . 2017. Media Pembelajaran Mobile Learning Menggunakan Android. Jurnal Sistem Informasi Dan Telematika, Volume. 8, No. 1, ISSN: 2087-2062.

Melda Agarina, Nurul Hikmah Afiril . 2018. Perancangan Sistem Informasi Berbasis Mobile Pada Restoran Lokal Di Bandar Lampung. Jurnal Sistem Informasi Dan Telematika, Volume. 8, No. 2, ISSN: 2088-5555

LAMPIRAN

Koding Desain Halaman Guest House

```
package com.flaniest.skripsi.guesthousebdlusr.chat;

import android.content.Context;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.view.inputmethod.InputMethodManager;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import com.flaniest.skripsi.guesthousebdlusr.R;
import com.flaniest.skripsi.guesthousebdlusr.svc.LoadingSvc;
import com.google.android.gms.tasks.OnFailureListener;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.firebase.Timestamp;
import com.google.firebase.firestore.DocumentChange;
import com.google.firebase.firestore.DocumentReference;
import com.google.firebase.firestore.EventListener;
import com.google.firebase.firestore.FirebaseFirestore;
import com.google.firebase.firestore.FirebaseFirestoreException;
import com.google.firebase.firestore.FirebaseFirestoreSettings;
import com.google.firebase.firestore.ListenerRegistration;
import com.google.firebase.firestore.QueryDocumentSnapshot;
import com.google.firebase.firestore.QuerySnapshot;

import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.HashMap;
import java.util.List;
import java.util.Locale;
import java.util.Map;
```

```
import java.util.TimeZone;

import javax.annotation.Nullable;

public class Chat extends AppCompatActivity {
    private String uidOwn, nameOwn;
    private String uidOth, nameOth;

    private RecyclerView rv;
    private RecyclerView.LayoutManager rvLm;
    private ChatAdp adp;
    private List<ChatMdl> mdls;

    private EditText etChat;
    private Button btnSend;

    private ChatDtc dtc;
    private FirebaseFirestore db;
    private ListenerRegistration listenerRegistration;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.chat);

        String sTitle = getString(R.string.strChat);
        setTitle(sTitle);

        uidOwn = getIntent().getStringExtra("uidOwn");
        nameOwn = getIntent().getStringExtra("nameOwn");
        uidOth = getIntent().getStringExtra("uidOth");
        nameOth = getIntent().getStringExtra("nameOth");

        sTitle = nameOth;
        setTitle(sTitle);

        rv = findViewById(R.id.rv);
        rvLm = new LinearLayoutManager(this);
        rv.setLayoutManager(rvLm);

        etChat = findViewById(R.id.etChat);
```

```
        btnSend = findViewById(R.id.btnSend);
        btnSend.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                vld();
            }
        });

        dt();
    }

    private void dt() {
        LoadingSvc.loadingShow(this);

        dtc = new ChatDtc(this, uidOwn);
        mdls = dtc.getChat(uidOth);

        adp = new ChatAdp(mdls);
        rv.setAdapter(adp);

        fb();

        LoadingSvc.loadingDismiss();
    }

    private void fb() {
        db = FirebaseFirestore.getInstance();

        FirebaseFirestoreSettings settings = new
FirebaseFirestoreSettings.Builder().setTimestampsInSnapshotsEnabled(true).build
();
        db.setFirestoreSettings(settings);

        listenerRegistration =
db.collection("chat").document(uidOwn).collection("messages")
                .orderBy("createAt")
                .addSnapshotListener(new EventListener<QuerySnapshot>() {
                    @Override
                    public void onEvent(@Nullable QuerySnapshot
queryDocumentSnapshots, @Nullable FirebaseFirestoreException e) {
                        if (e != null) {
                            Log.w("FirebaseFirestore", "Listen failed.", e);
```

```
                    return;
                }

                for (DocumentChange documentChange :
queryDocumentSnapshots.getDocumentChanges()) {
                    switch (documentChange.getType()) {
                        case ADDED:
                            QueryDocumentSnapshot document =
documentChange.getDocument();
                            if (document.getString("uid").equals(uidOth) ||
document.getString("uid").equals(uidOwn)) {
                                ChatMdl chatMdl = new ChatMdl();
                                chatMdl.createAt =
dtToUTC(document.getTimestamp("createAt").toDate());
                                chatMdl.content = document.getString("content");
                                chatMdl.owner = document.getString("owner");

                                long r = dtc.addChat(chatMdl, uidOth);
                                if (r > -1) {
                                    mdls.add(chatMdl);

                                    fbDel(document.getId());
                                }
                            }
                            break;
                    }
                }

                /*for (QueryDocumentSnapshot document :
queryDocumentSnapshots) {
                    if (document.getString("uid").equals(uidOth) ||
document.getString("uid").equals(uidOwn)) {
                        ChatMdl chatMdl = new ChatMdl();
                        chatMdl.createAt =
dtToUTC(document.getTimestamp("createAt").toDate());
                        chatMdl.content = document.getString("content");
                        chatMdl.owner = document.getString("owner");

                        long r = dtc.addChat(chatMdl, uidOth);
                        if (r > -1) {
                            mdls.add(chatMdl);
```

```
                        fbDel(document.getId());
                    }
                }
            }*/

            rv.scrollToPosition(mdls.size() -1);
        }
    });
}

private String dtToUTC(Date dt) {
    SimpleDateFormat simpleDateFormat = new SimpleDateFormat("yyyy-
MM-dd'T'HH:mm:ss.SSS'Z'", Locale.US);
    simpleDateFormat.setTimeZone(TimeZone.getTimeZone("UTC"));

    return simpleDateFormat.format(dt);
}

private void fbDel(String documentId) {

db.collection("chat").document(uidOwn).collection("messages").document(docu
mentId)
        .delete()
        .addOnSuccessListener(new OnSuccessListener<Void>() {
            @Override
            public void onSuccess(Void aVoid) {
                Log.d("FirebaseFirestore", "DocumentSnapshot successfully
deleted!");
            }
        })
        .addOnFailureListener(new OnFailureListener() {
            @Override
            public void onFailure(@NonNull Exception e) {
                Log.w("FirebaseFirestore", "Error deleting document", e);
            }
        });
}

private void vld() {
    View view = this.getCurrentFocus();
    if (view != null) {
        InputMethodManager inputMethodManager = (InputMethodManager)
```

```
getSystemService(Context.INPUT_METHOD_SERVICE);

inputMethodManager.hideSoftInputFromWindow(view.getWindowToken(),
InputMethodManager.RESULT_UNCHANGED_SHOWN);
        }

        boolean valid = true;

        if (etChat.getText().toString().trim().isEmpty()) {
            valid = false;
        }

        if (valid) {
            sendOwn();
        } else Toast.makeText(this, R.string.strCantEmpty,
Toast.LENGTH_LONG).show();
    }

    private void sendOwn() {
        LoadingSvc.loadingShow(this);

        final Timestamp createAt = new Timestamp(new Date());
        final String content = etChat.getText().toString();

        Map<String, Object> values = new HashMap<>();
        values.put("createAt", createAt);
        values.put("content", content);
        values.put("owner", "own");
        values.put("uid", uidOwn);
        values.put("name", nameOwn);

        db.collection("chat").document(uidOwn).collection("messages")
                .add(values)
                .addOnSuccessListener(new
OnSuccessListener<DocumentReference>() {
                    @Override
                    public void onSuccess(DocumentReference documentReference) {
                        sendOth(createAt, content);
                        etChat.setText(null);

                        Log.d("FirebaseFirestore", "DocumentSnapshot added" +
documentReference.getId());
```

```
                }
            })
            .addOnFailureListener(new OnFailureListener() {
                @Override
                public void onFailure(@NonNull Exception e) {
                    LoadingSvc.loadingDismiss();

                    Log.w("FirebaseFirestore", "Error adding document", e);
                }
            });
    }

    private void sendOth(final Timestamp createAt, final String content) {
        Map<String, Object> values = new HashMap<>();
        values.put("createAt", createAt);
        values.put("content", content);
        values.put("owner", "oth");
        values.put("uid", uidOwn);
        values.put("name", nameOwn);

        db.collection("chat").document(uidOth).collection("messages")
                .add(values)
                .addOnSuccessListener(new
OnSuccessListener<DocumentReference>() {
                    @Override
                    public void onSuccess(DocumentReference documentReference) {
                        LoadingSvc.loadingDismiss();

                        Log.d("FirebaseFirestore", "DocumentSnapshot added : " +
documentReference.getId());
                    }
                })
                .addOnFailureListener(new OnFailureListener() {
                    @Override
                    public void onFailure(@NonNull Exception e) {
                        LoadingSvc.loadingDismiss();

                        Log.w("FirebaseFirestore", "Error adding document", e);
                    }
                });
    }
```

```
    @Override
    protected void onDestroy() {
        super.onDestroy();

        listenerRegistration.remove();
    }
}
```

Koding sistem yang dibuat

```
package com.flaniest.skripsi.guesthousebdlusr;

import android.content.Intent;
import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.widget.Toast;

import com.android.volley.Response;
import com.android.volley.VolleyError;
import com.enwif.lib.vllskr.RqsDtc;
import com.flaniest.skripsi.guesthousebdlusr.cnn.Cnn;
import com.flaniest.skripsi.guesthousebdlusr.mdl.UserMdl;
import com.flaniest.skripsi.guesthousebdlusr.sct.Home;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.auth.GetTokenResult;

import org.json.JSONException;
import org.json.JSONObject;

import java.util.HashMap;
import java.util.Map;

public class Main extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        FirebaseUser firebaseUser = FirebaseAuth.getInstance().getCurrentUser();
        if (firebaseUser != null) {
            firebaseUser.getIdToken(true)
                    .addOnCompleteListener(this, new OnCompleteListener<GetTokenResult>()
{
                        @Override
                        public void onComplete(@NonNull Task<GetTokenResult> task) {
                            if (task.isSuccessful()) {
                                signin(task.getResult().getToken());
                            } else {
```

```
                    Toast.makeText(Main.this, R.string.strLoadDataFail,
Toast.LENGTH_SHORT).show();

                    goSignIn();

                    Log.w("Frb", "GetIdToken Failed", task.getException());
                }
            }
        });
    } else goSignIn();
  }

  private void signin(String token) {
    Map<String, String> params = new HashMap<>();
    params.put("token", token);

    RqsDtc.StrRqsPOSTDtc(this, Cnn.url + "user/login", params, new
Response.Listener<String>() {
      @Override
      public void onResponse(String response) {
        if (!response.equals("invalid") && !response.equals("err") &&
!response.isEmpty()) {
          try {
            JSONObject jsonObject = new JSONObject(response);

            UserMdl mdl = new UserMdl();
            mdl.uid = jsonObject.getString("uid");
            mdl.user = jsonObject.getString("user");
            mdl.alamat = jsonObject.getString("alamat");
            mdl.notlp = jsonObject.getString("notlp");

            goHome(mdl);
          } catch (JSONException e) {
            e.printStackTrace();
          }
        } else {
          Toast.makeText(Main.this, R.string.strLoadDataFail,
Toast.LENGTH_LONG).show();

          goSignIn();
        }
      }
    }, new Response.ErrorListener() {
      @Override
      public void onErrorResponse(VolleyError error) {
        Toast.makeText(Main.this, R.string.strLoadDataFail,
```

```
Toast.LENGTH_LONG).show();

                Log.e("Rqs", error.toString());
            }
        });
    }

    private void goHome(UserMdl mdl) {
        Intent intent = new Intent(this, Home.class);
        intent.putExtra("mdl", mdl);
        intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TASK |
Intent.FLAG_ACTIVITY_NEW_TASK);
        startActivity(intent);
    }

    private void goSignIn() {
        Intent intent = new Intent(this, SignIn.class);
        intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TASK |
Intent.FLAG_ACTIVITY_NEW_TASK);
        startActivity(intent);
    }
}
```

Koding Java Admin

```
package com.flaniest.skripsi.guesthousebdladm.sct;

import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;
import androidx.swiperefreshlayout.widget.SwipeRefreshLayout;

import android.os.Bundle;
import android.util.Base64;
import android.util.Log;
import android.widget.Toast;

import com.android.volley.Response;
import com.android.volley.VolleyError;
import com.enwif.lib.vllskr.RqsDtc;
import com.flaniest.skripsi.guesthousebdladm.R;
import com.flaniest.skripsi.guesthousebdladm.adp.HomeAdp;
import com.flaniest.skripsi.guesthousebdladm.cnn.Cnn;
import com.flaniest.skripsi.guesthousebdladm.mdl.GuestHouseMdl;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import java.util.ArrayList;
import java.util.List;

public class Home extends AppCompatActivity {
    private RecyclerView rv;
    private RecyclerView.LayoutManager rvLm;
    private SwipeRefreshLayout srl;
    private HomeAdp adp;
    private List<GuestHouseMdl> mdls;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.home);

        rv = findViewById(R.id.rv);
```

```
        rvLm = new LinearLayoutManager(this);
        rv.setLayoutManager(rvLm);
        srl = findViewById(R.id.srl);
        srl.setOnRefreshListener(new SwipeRefreshLayout.OnRefreshListener() {
            @Override
            public void onRefresh() {
                dt();
            }
        });

        dt();
    }

    private void dt() {
        mdls = new ArrayList<>();
        adp = new HomeAdp(mdls);
        rv.setAdapter(adp);

        srl.setRefreshing(true);

        RqsDtc.StrRqsGETDtc(this, Cnn.url + "guesthouse", new
Response.Listener<String>() {
            @Override
            public void onResponse(String response) {
                if (!response.equals("invalid") && !response.equals("err")) {
                    try {
                        JSONArray jsonArray = new JSONArray(response);

                        for (int i = 0; i < jsonArray.length(); i++) {
                            JSONObject jsonObject = jsonArray.getJSONObject(i);

                            GuestHouseMdl mdl = new GuestHouseMdl();
                            mdl.uid = jsonObject.getString("uid");
                            mdl.guesthouse = jsonObject.getString("guesthouse");
                            mdl.koskontrakan = jsonObject.getString("koskontrakan");
                            mdl.alamat = jsonObject.getString("alamat");
                            mdl.notlp = jsonObject.getString("notlp");
                            mdl.tipe = jsonObject.getString("tipe");
                            if (!jsonObject.isNull("keterangan")) mdl.keterangan =
jsonObject.getString("keterangan");
                            mdl.lat = jsonObject.getString("lat");
                            mdl.lng = jsonObject.getString("lng");
```

```
                mdls.add(mdl);
            }

            for (int i = 0; i < mdls.size(); i++) {
                getImages(i);
            }
        } catch (JSONException e) {
            Toast.makeText(Home.this, R.string.strProgressFail,
Toast.LENGTH_SHORT).show();

            e.printStackTrace();
        }
    } else Toast.makeText(Home.this, R.string.strLoadDataFail,
Toast.LENGTH_SHORT).show();

    adp.notifyDataSetChanged();
    srl.setRefreshing(false);
  }
}, new Response.ErrorListener() {
  @Override
  public void onErrorResponse(VolleyError error) {
    srl.setRefreshing(false);

    Toast.makeText(Home.this, R.string.strLoadDataFail,
Toast.LENGTH_SHORT).show();

    Log.e("Rqs", error.toString());
  }
});
}

private void getImages(final int i) {
  RqsDtc.StrRqsGETDtc(this, Cnn.url + "img/guesthouse/thumbnails/" +
mdls.get(i).uid, new Response.Listener<String>() {
    @Override
    public void onResponse(String response) {
      if (i < mdls.size()) {
        if (!response.equals("invalid") && !response.equals("err"))
          mdls.get(i).imgByte = Base64.decode(response, 0);
        else {
          mdls.get(i).imgByte = null;
```

```
                }

                adp.notifyItemChanged(i);
            }
        }
    }, new Response.ErrorListener() {
        @Override
        public void onErrorResponse(VolleyError error) {
            Log.e("Rqs", error.toString());
        }
    });
  }
}
```

```
package com.flaniest.skripsi.guesthousebdladm;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.view.inputmethod.InputMethodManager;
import android.widget.Button;
import android.widget.Toast;

import com.android.volley.Response;
import com.android.volley.VolleyError;
import com.enwif.lib.vllskr.RqsDtc;
import com.flaniest.skripsi.guesthousebdladm.cnn.Cnn;
import com.flaniest.skripsi.guesthousebdladm.mdl.AdminMdl;
import com.flaniest.skripsi.guesthousebdladm.sct.Home;
import com.flaniest.skripsi.guesthousebdladm.svc.LoadingSvc;
import com.google.android.material.textfield.TextInputEditText;
import com.google.android.material.textfield.TextInputLayout;

import org.json.JSONException;
import org.json.JSONObject;

import java.util.HashMap;
import java.util.Map;

public class SignIn extends AppCompatActivity {
    private TextInputLayout tilUsername, tilPassword;
    private TextInputEditText tietUsername, tietPassword;
    private Button btnSignIn;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.sign_in);

        tilUsername = findViewById(R.id.tilUsername);
        tietUsername = findViewById(R.id.tietUsername);
```

```
        tilPassword = findViewById(R.id.tilPassword);
        tietPassword = findViewById(R.id.tietPassword);

        btnSignIn = findViewById(R.id.btnSignIn);
        btnSignIn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                vld();
            }
        });
    }

    private void vld() {
        View view = this.getCurrentFocus();
        if (view != null) {
            InputMethodManager inputMethodManager = (InputMethodManager)
getSystemService(Context.INPUT_METHOD_SERVICE);

inputMethodManager.hideSoftInputFromWindow(view.getWindowToken(),
InputMethodManager.RESULT_UNCHANGED_SHOWN);
        }

        boolean valid = true;

        if (tietUsername.getText().toString().trim().isEmpty()) {
            tilUsername.setError(getString(R.string.strCantEmpty));
            valid = false;
        }

        if (tietPassword.getText().toString().trim().isEmpty()) {
            tilPassword.setError(getString(R.string.strCantEmpty));
            valid = false;
        }

        if (valid) {
            signin();
        } else Toast.makeText(this, R.string.strCantEmpty,
Toast.LENGTH_SHORT).show();
    }

    private void signin() {
        LoadingSvc.loadingShow(this);
```

```
        Map<String, String> params = new HashMap<>();
        params.put("username", tietUsername.getText().toString());
        params.put("password", tietPassword.getText().toString());

        RqsDtc.StrRqsPOSTDtc(this, Cnn.url + "admin/login", params, new
Response.Listener<String>() {
            @Override
            public void onResponse(String response) {
                if (!response.equals("invalid") && !response.equals("err") &&
!response.isEmpty()) {
                    try {
                        JSONObject jsonObject = new JSONObject(response);

                        AdminMdl mdl = new AdminMdl();
                        mdl.idadmin = jsonObject.getString("idadmin");
                        mdl.username = jsonObject.getString("username");
                        mdl.password = jsonObject.getString("password");
                        mdl.nama = jsonObject.getString("nama");

                        SharedPreferences sharedPreferences =
getSharedPreferences("lgn", MODE_PRIVATE);
                        SharedPreferences.Editor editor = sharedPreferences.edit();

                        editor.putString("idadmin", mdl.idadmin);
                        editor.putString("username", mdl.username);
                        editor.putString("password", mdl.password);
                        editor.putString("nama", mdl.nama);
                        editor.apply();

                        goHome(mdl);
                    } catch (JSONException e) {
                        e.printStackTrace();
                    }
                } else Toast.makeText(SignIn.this, R.string.strSignInFail,
Toast.LENGTH_LONG).show();

                LoadingSvc.loadingDismiss();
            }
        }, new Response.ErrorListener() {
            @Override
            public void onErrorResponse(VolleyError error) {
```

```
                LoadingSvc.loadingDismiss();

                Toast.makeText(SignIn.this, R.string.strSignInFail,
Toast.LENGTH_LONG).show();

                Log.e("Rqs", error.toString());
            }
        });
    }

    private void goHome(AdminMdl mdl) {
        Intent intent = new Intent(this, Home.class);
        intent.putExtra("mdl", mdl);
        intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TASK |
Intent.FLAG_ACTIVITY_NEW_TASK);
        startActivity(intent);
    }
}
```

```
package com.flaniest.skripsi.guesthousebdladm.adp;

import android.content.Context;
import android.content.DialogInterface;
import android.graphics.BitmapFactory;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AlertDialog;
import androidx.cardview.widget.CardView;
import androidx.recyclerview.widget.RecyclerView;

import com.android.volley.Response;
import com.android.volley.VolleyError;
import com.enwif.lib.vllskr.RqsDtc;
import com.flaniest.skripsi.guesthousebdladm.R;
import com.flaniest.skripsi.guesthousebdladm.cnn.Cnn;
import com.flaniest.skripsi.guesthousebdladm.mdl.GuestHouseMdl;
import com.flaniest.skripsi.guesthousebdladm.svc.LoadingSvc;

import java.util.HashMap;
import java.util.List;
import java.util.Map;

public class HomeAdp extends RecyclerView.Adapter<HomeAdp.VH> {
    private List<GuestHouseMdl> mdls;
    private Context context;

    public HomeAdp(List<GuestHouseMdl> mdls) {
        this.mdls = mdls;
    }

    class VH extends RecyclerView.ViewHolder {
        private CardView cv;
        private ImageView iv;
        private TextView tvGuestHouse, tvAlamat, tvNoTlp;
```

```
        public VH(@NonNull View itemView) {
            super(itemView);

            cv = itemView.findViewById(R.id.cv);
            iv = itemView.findViewById(R.id.iv);
            tvGuestHouse = itemView.findViewById(R.id.tvGuestHouse);
            tvAlamat = itemView.findViewById(R.id.tvAlamat);
            tvNoTlp = itemView.findViewById(R.id.tvNoTlp);

            cv.setOnLongClickListener(new View.OnLongClickListener() {
                @Override
                public boolean onLongClick(View view) {
                    chs(getAdapterPosition());

                    return false;
                }
            });
        }
    }

    @NonNull
    @Override
    public VH onCreateViewHolder(@NonNull ViewGroup parent, int viewType)
{
        context = parent.getContext();
        View view =
LayoutInflater.from(parent.getContext()).inflate(R.layout.home_i, parent, false);
        return new VH(view);
    }

    @Override
    public void onBindViewHolder(@NonNull VH holder, int position) {
        GuestHouseMdl mdl = mdls.get(position);

        if (mdl.imgByte != null)
holder.iv.setImageBitmap(BitmapFactory.decodeByteArray(mdl.imgByte, 0,
mdl.imgByte.length));
        else holder.iv.setImageResource(R.drawable.noimg);

        holder.tvGuestHouse.setText(mdl.guesthouse);
        holder.tvAlamat.setText(mdl.alamat);
```

```
        holder.tvNoTlp.setText(mdl.notlp);
    }

    @Override
    public int getItemCount() {
        return mdls.size();
    }

    private void chs(final int pos) {
        AlertDialog.Builder builder = new AlertDialog.Builder(context);

        builder.setTitle(R.string.strMenu)
                .setItems(new CharSequence[]{
                        context.getString(R.string.strDelete),
                        context.getString(R.string.strCancel)
                },
                new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                        switch (which) {
                            case 0:
                                chsDel(pos);
                                break;

                            case 1:
                                break;
                        }
                    }
                });

        builder.create().show();
    }

    private void chsDel(final int pos) {
        AlertDialog.Builder builder = new AlertDialog.Builder(context);
        builder.setTitle(R.string.strDeleteQst)
                .setPositiveButton(R.string.strYes, new
DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialogInterface, int i) {
                        del(pos);
                    }
```

```
            })
            .setNegativeButton(R.string.strNo, null);
        builder.show();
    }

    private void del(final int pos) {
        LoadingSvc.loadingShow(context);

        Map<String, String> params = new HashMap<>();

        RqsDtc.StrRqsDELDtc(context, Cnn.url + "guesthouse/" +
mdls.get(pos).uid, params, new Response.Listener<String>() {
            @Override
            public void onResponse(String response) {
                if (!response.equals("invalid") && !response.equals("err")) {
                    Toast.makeText(context, R.string.strDeletingSuccess,
Toast.LENGTH_SHORT).show();

                    mdls.remove(pos);
                    notifyDataSetChanged();
                } else Toast.makeText(context, R.string.strDeletingFail,
Toast.LENGTH_SHORT).show();

                LoadingSvc.loadingDismiss();
            }
        }, new Response.ErrorListener() {
            @Override
            public void onErrorResponse(VolleyError error) {
                LoadingSvc.loadingDismiss();

                Toast.makeText(context, R.string.strDeletingFail,
Toast.LENGTH_SHORT).show();

                Log.e("Rqs", error.toString());
            }
        });
    }
}
```