RANCANG BANGUN SISTEM INFORMASI GEOGRAFIS UNTUK PENCARIAN RUMAH KOS BERBASIS WEB

SKRIPSI

DIAWAN SALLEKARURUNG

H071181509

PROGRAM STUDI SISTEM INFORMASI

DEPARTEMEN MATEMATIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS HASANUDDIN

MAKASSAR

2024

RANCANG BANGUN SISTEM INFORMASI GEOGRAFIS UNTUK PENCARIAN RUMAH KOS BERBASIS *WEB*

SKRIPSI

Diajukan sebagai salah satu syarat memperoleh gelar Sarjana Sains pada Program Studi Sistem Informasi Departemen Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Hasanuddin

DIAWAN SALLEKARURUNG

H071181509

PROGRAM STUDI SISTEM INFORMASI

DEPARTEMEN MATEMATIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS HASANUDDIN

MAKASSAR

2024

PERNYATAAN KEASLIAN

Yang bertanda tangan di bawah ini:

Nama	: Diawan Sallekarurung
NIM	: H071181509
Program Studi	: Sistem Informasi
Jenjang	: S1

Menyatakan bahwa dengan ini bahwa karya tulis saya dengan judul:

Rancang bangun sistem informasi geografis untuk pencarian rumah kos berbasis *web*

Adalah karya tulisan saya sendiri dan bukan merupakan hasil plagiat dan belum pernah dipublikasikan dalam bentuk apapun.

Apabila dikemudian hari terbukti atau dapat dibuktikan bahwa sebagian atau keseluruhan skripsi ini hasil karya orang lain, maka saya bersedia menerima sanksi atau perbuatan tersebut.

Makassar, 25 Januari 2024

Menyatakan,

Diawan Sallekarurung

H071181509

RANCANG BANGUN SISTEM INFORMASI GEOGRAFIS UNTUK PENCARIAN RUMAH KOS BERBASIS *WEB*

Disusun dan diajukan oleh:

DIAWAN SALLEKARURUNG

H071181509

Telah dipertahankan di hadapan dewan penguji dan diterima sebagai bagian persyaratan yang diperlukan untuk memperoleh gelar Sarjana Komputer pada Program Studi Sistem Informasi Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Hasanuddin

Menyetujui,

Pembimbing Utama,	**Pembimbing Pertama,**
Dr. Hendra, S.Si., M.Kom.	**Muhammad Sadno, S.Si., M.Si.**
NIP. 197601022002121001	NIP. 199008162022043001

Ketua Program Studi,

Dr. Hendra, S.Si., M.Kom.
NIP. 197601022002121001

HALAMAN PENGESAHAN

Skripsi ini di ajukan oleh:

Nama	: Diawan Sallekarurung
NIM	: H071181509
Program Studi	: Sistem Informasi
Judul Skripsi	: Rancang Bangun Sistem Informasi Geografis Untuk Pencarian Rumah Kos Berbasis *Web*

Telah berhasil dipertahankan di hadapan Dewan Penguji dan diterima sebagai bagian persyaratan yang diperlukan untuk memperoleh gelar Sarjana Komputer pada Program Studi Sistem Informasi Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Hasanuddin.

DEWAN PENGUJI

		Tanda Tangan
Ketua	: Dr. Hendra, S.Si., M.Kom.	(.....)
Sekretaris	: Muhammad Sadno, S.Si., M.Si.	(.....)
Anggota	: A. Muh. Amil Siddik, S.Si.,M.Si.	(.....)
Anggota	: Riskawati, S.Si.,M.Si.	(.....)

Ditetapkan di : Makassar

Tanggal : 25 Januari 2024

KATA PENGANTAR

Segalan puji dan Syukur atas kehadirat Allah SWT atas berkat, rahmat, dan hidayah-Nya yang senantiasa dilimpahkan kepada penulis, sehingga bisa menyelesaikan skripsi dengan judul "Rancang Bangun Sistem Informasi Geografis Untuk Pencarian Rumah Kos Berbasis *Web*" sebagai syarat untuk menyelesaikan gelar Sarjana Sains pada Program Studi Sistem Informasi Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Hasanuddin.

Dalam menyusun skripsi ini banyak hambatan serta rintangan yang penulis hadapi namun pada akhirnya dapat dilalui berkat adanya bimbingan dan bantuan dari berbagai pihak baik secara moral maupun spiritual. Oleh karna itu, penulis ingin menyampaikan ucapan terimakasih yang tak terhingga kepada orang tua penulis, Ibu Kristina Rerung dan Kakak Wulan Sallekarurung serta seluruh keluarga yang selalu memberikan dukungan baik dari segi materi, moril, kasih sayang, doa dan nasihat yang tulus sebagai bekal kehidupan.

Penghargaan dan ucapan terimakasih dengan penuh ketulusan juga diucapkan kepada:

1. Rektor Universitas Hasanuddin Makassar **Prof. Dr. Ir. Jamaluddin Jompa, M. Sc** dan seluruh Wakil Rektor dalam lingkup Universitas Hasanuddin.
2. Bapak **Dr. Eng. Amiruddin, M. Si** selaku Dekan Fakultas Matematika dan Ilmu Pengetahun Alam Universitas Hasanuddin dan para Wakil Dekan serta seluruh staf yang telah memberikan bantuan selama penulis mengikuti pendidikan di FMIPA Universitas Hasanuddin.
3. Bapak **Prof. Dr. Nurdin, S.Si., M. Si** selaku Ketua Departemen Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Hasanuddin, penulis juga berterimakasih atas dedikasi dosen-dosen serta staf Departemen atas ilmu dan bantuan yang bermanfaat.
4. Bapak **Dr. Hendra, S.Si, M.Kom** selaku Ketua Program Studi Sistem Informasi Universitas Hasanuddin sekaligus dosen pembimbing utama yang telah menyediakan waktu, tenaga, dan pikiran untuk mengarahkan penulis dalam menyusun skripsi ini.

5. Bapak **Muhammad Sadno, S.Si., M.Si.** selaku dosen pembimbing pertama yang telah menyediakan waktu, tenaga dan pikiran dalam mengarahkan penulis dalam menyusun skripsi ini.
6. Bapak **A. Muh. Amil Siddik, S.Si.,M.Si** dan **Ibu Riskawati, S.Si., M.Si** selaku dosen penguji atas segala kritikan dan masukan yang membangun dalam penyusunan skripsi ini.
7. Teman-teman seperjuangan **Sistem Informasi 2018** yang turut membantu penulis dalam memberikan arahan dan dukungan serta selalu membersamai penulis dalam menyusun skripsi ini.
8. Kakak-kakak dan adik-adik Program Studi Sistem Informasi 2014, 2015, 2016, 2017, 2019, 2020, 2021.
9. Semua pihak yang telah memberikan bantuan dan dukungan kepada penulis yang tidak dapat di sebutkan satu persatu, terimakasih atas bantuannya.

Akhir kata, saya berharap Tuhan yang Maha Esa berkenan membalas segala kebaikan dari semua pihak yang telah membantu penulis. Semoga skripsi ini dapat memberikan manfaat bagi pengembangan ilmu.

Makassar, 25 Januari 2024
Menyatakan,

Diawan Sallekarurung

PERNYATAAN PERSETUJUAN PUBLIKASI TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIS

Sebagi civitas akademik Universitas Hasanuddin, saya yang bertanda tangan di bawah ini:

Nama	: Diawan Sallekarurung
NIM	: H071181509
Program Studi	: Sistem Informasi
Departemen	: Matematika
Fakultas	: Matematikan dan Imu Pengetahuan Alam
Jenis Karya	: Skripsi

Demi pengembangan ilmu pengetahun, menyetujui untuk memberikan kepada Universitas Hasanuddin **Hak Bebas Royalti Non Eksklusif (Non-exclusive Royalty-Free Right)** atas karya ilmiah saya yang berjudul:

"Rancang Bangun Sistem Informasi Geografis Untuk Pencarian Rumah Kos Berbasis *Web*"

Beserta perangkat yang ada (jika diperlukan). Terkait dengan hal diatas, maka pihak universitas berhak menyimpan, mengalih-media/format-kan, megelola dalam bentuk pangkalan data (*database*), merawat, dan mempublikasikan tugas akhir saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta.

Demikian pernyataan ini saya buat dengan sebenarnya.
Dibuat di Makassar pada tanggal 25 Januari 2024

Yang menyatakan,

(Diawan Sallekarurung)

ABSTRAK

Perkembangan teknologi yang pesat telah mengubah berbagai aspek kehidupan, termasuk cara kita mencari tempat tinggal, seperti rumah kos. Dalam konteks ini, proses pencarian rumah kos menjadi semakin kompleks seiring dengan pertumbuhan dan perkembangan kebutuhan masyarakat. Untuk mengatasi tantangan ini, dibutukan tempat yang menyediakan solusi inovatif berupa *website*, yang dapat mempermudah dan meningkatkan efisiensi dalam proses pencarian rumah sementara atau rumah kos. Oleh karena itu, Penelitian ini bertujuan untuk menyediakan informasi berbasis *website* kepada pengguna guna mempermudah proses pencarian rumah sementara atau rumah kos. Metode yang digunakan dalam penelitian ini adalah metode *waterfall*, yang mencakup analisis, perancangan, implementasi, dan pengujian. Pengembangan aplikasi dilakukan menggunakan *Framework Django* dan *ReactJs* serta *database* PostgreSQL. Hasil implementasi penelitian ini adalah sebuah aplikasi *website* yang dapat diakses secara *online*. Aplikasi ini tidak hanya menyediakan fitur pencarian rumah kos, tetapi juga memungkinkan pengguna untuk melakukan transaksi kamar kos, dan memungkinkan pemilik kos mengeloah rumah kosnya. Dengan adanya aplikasi ini, diharapkan masyarakat dapat dengan lebih efisien dan efektif mencari tempat tinggal sesuai dengan kebutuhan mereka.

Kata Kunci: Sistem Informasi, *React js, Web,* Rumah Kos

ABSTRACT

Rapid technological developments have changed various aspects of life, including the way we look for places to live, such as boarding houses. In this context, the process of searching for a boarding house becomes increasingly complex along with the growth and development of community needs. To overcome this challenge, we need a place that provides innovative solutions in the form of websites, which can simplify and increase efficiency in the process of searching for temporary housing or boarding houses. Therefore, this research aims to provide website-based information to users to simplify the process of searching for temporary housing or boarding houses. The method used in this research is the waterfall method, which includes analysis, design, implementation and testing. Application development was carried out using the Django and ReactJs frameworks and the PostgreSQL database. The result of implementing this research is a website application that can be accessed online. This application not only provides a boarding house search feature, but also allows users to carry out boarding room transactions, and allows boarding house owners to manage their boarding houses. With this application, it is hoped that people can more efficiently and effectively find a place to live according to their needs.

Keywords: Information Systems, React js, Web, Boarding House

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1 Latar belakang

Perkembangan teknologi telah menciptakan banyak inovasi yang bertujuan untuk memenuhi kebutuhan manusia. Hal ini telah menciptakan suatu pilihan penting dalam kehidupan kita, yaitu bahwa perkembangan teknologi sangat penting di era *modern* saat ini. Saat ini, dalam persaingan pasar yang sangat sengit, ketepatan informasi dan efisiensi waktu menjadi penentu keberhasilan dari strategi dan rencana yang disusun. Oleh karena itu, diperlukan solusi teknologi yang dapat mendukung proses pengambilan keputusan dengan efisien.

Tuntutan akan informasi terus mendorong pengembangan aplikasi yang dapat memenuhi kebutuhan masyarakat. Aplikasi tersebut memanipulasi dan mengolah informasi untuk memenuhi kebutuhan khusus di Indonesia. Beberapa penelitian telah dilakukan dengan memanfaatkan teknologi berbasis *web* untuk memudahkan masyarakat dalam mengakses informasi yaitu (Annugerah et al., 2016) untuk pemetaan toko, (Putra & Afri, 2020) informasi pariwisata, dan (Wahyutomo et al., 2015) dalam informasi persebaran kantor pos. Penelitian tersebut telah mampu memberikan sarana informasi yang bisa diakses dan digunakan oleh masyarakat.

Salah satu kebutuhan mendasar masyarakat adalah tempat tinggal dan dengan pertumbuhan populasi yang pesat terutama di Indonesia, permintaan akan tempat tinggal baik permanen maupun sementara terus meningkat. Namun, seringkali kurangnya informasi membuat masyarakat kesulitan dalam mencari tempat tinggal yang sesuai. Oleh karena itu, dibutuhkan suatu sistem yang dapat mempermudah masyarakat dalam mencari tempat tinggal terutama tempat tinggal sementara atau sering disebut "rumah kos". Aplikasi ini dapat menghemat waktu, tenaga, dan pikiran masyarakat dalam mencari informasi tentang rumah kos yang tersedia. Selain itu, aplikasi ini juga dapat mempermudah proses pencarian rumah kos yang sesuai dengan lokasi dan anggaran yang dimiliki oleh masyarakat. Beberapa penelitian telah membahas tentang pemanfaatan sistem informasi geografis dalam

penyediaan informasi rumah kos yaitu (Kosasih, 2015),(Agape Sianturi et al., 2018) dan (Wiatr-Kmieciak, 2016).

Dengan memanfaatkan teknologi informasi geografi, aplikasi pencarian rumah kos dapat memberikan informasi tentang lokasi rumah kos dengan sangat akurat dan detail. Pengguna dapat melihat peta dan melacak lokasi rumah kos yang mereka inginkan, serta melihat fasilitas dan layanan yang tersedia di sekitar *area* rumah kos tersebut. Sistem ini juga memudahkan bagi pemilik rumah kos untuk mempromosikan properti mereka dan memperluas jangkauan pasar. Pemilik rumah kos dapat memasukkan informasi tentang rumah kos mereka ke dalam aplikasi, seperti lokasi, fasilitas, harga sewa, dan foto. Informasi ini akan membantu mereka dalam mempromosikan rumah kos mereka dan menarik lebih banyak pelanggan.

Dalam konteks tersebut, penelitian ini membahas tentang sistem informasi geografis berbasis *web* yang dapat memudahkan masyarakat dalam mencari tempat tinggal sementara, seperti rumah kos. Tujuannya adalah mempermudah aktivitas masyarakat, terutama bagi pendatang baru dan mahasiswa yang seringkali kesulitan mencari tempat tinggal yang sesuai.

1.2 Rumusan Masalah

Berdasarkan uraian latar belakang, rumusan masalah pada penelitian ini sebagai berikut:

1. Bagaimana kebutuhan sistem informasi geografis pencarian kos berbasis *website*?
2. Bagaimana merancang dan membangun sistem informasi geografis pencarian rumah kos berbasis *web*?
3. Bagaimana efektivitas sistem informasi geografis pencarian rumah kos berbasis *web*?

1.3 Batasan Masalah

Batasan masalah dalam penelitian ini sebagai berikut :

1. Aplikasi berbasis *web* dirancang dan dibangun dengan menggunakan *Framework Django* dan *ReactJs*.
2. Perancangan *database* aplikasi berbasis *web* menggunakan *PostgreSQL*.
3. Aplikasi *website* dikembangkan menggunakan metode *waterfall*.

1.4 Tujuan Penelitian

Berdasarkan rumusan masalah, berikut tujuan dari penelitian:

1. Untuk menganalisis kebutuhan sistem informasi geografis pencarian kos berbasis *website*.
2. Untuk merancang dan membangun sistem informasi geografis pencarian rumah kos berbasis *web*.
3. Untuk menguji efektivitas sistem informasi geografis pencarian rumah Kos berbasis *web*.

BAB II
TINJAUAN PUSTAKA

2.1 Sistem Informasi

Sistem merupakan suatu jaringan kerja dari produser-produsernya yang saling berhubungan, berkumpul bersama-sama untuk melakukan suatu kegiatan atau untuk menyelesaikan suatu sasaran tertentu. Informasi adalah data yang telah diolah sehingga berguna bagi penerimanya, membantu pengambilan keputusan dan memberikan wawasan. Namun, informasi harus akurat dan dapat diandalkan. Jadi Sistem informasi adalah suatu rangkaian komponen yang saling berhubungan untuk mengumpulkan, menyimpan, mengolah, dan menyebarkan informasi dalam suatu organisasi. Sistem informasi dapat membantu perusahaan dalam memproses informasi secara efektif dan efisien, sehingga dapat meningkatkan produktivitas dan efisiensi bisnis.

Sistem Informasi terdiri dari enam komponen utama: *hardware*, *software*, data, prosedur, pemakai, dan jaringan. *Hardware* adalah peralatan fisik seperti komputer, printer, dan *scanner*. *Software* adalah program yang mengendalikan *hardware* dan melakukan tugas-tugas tertentu. Data adalah informasi yang diolah oleh sistem informasi. Prosedur adalah panduan yang menunjukkan bagaimana sistem informasi harus digunakan. Pemakai adalah orang yang menggunakan sistem informasi. Jaringan adalah jaringan komunikasi yang memungkinkan berbagai perangkat untuk terhubung dan berkomunikasi.

Ada beberapa jenis sistem informasi, termasuk sistem informasi manajemen (SIM), sistem informasi transaksi (SIT), sistem informasi besar (SIB), dan sistem informasi khusus (SIK). Masing-masing memiliki tujuan dan karakteristik yang berbeda. Sistem Informasi juga merupakan bagian dari teknologi informasi (TI), yang memainkan peran penting dalam membantu organisasi untuk mencapai tujuannya. Sistem Informasi membantu meningkatkan efisiensi dan efektivitas, membantu dalam pengambilan keputusan, dan membantu dalam mengatasi masalah bisnis dan operasional.

2.2 Sistem Informasi Geografis

Sistem Informasi Geografis (SIG) atau *Geographic Information System* (GIS) adalah suatu sistem informasi berbasis komputer yang digunakan untuk memasukkan, menyimpan, memanggil kembali, mengolah, menganalisis dan menghasilkan data geospasial untuk mendukung pengambilan keputusan dalam berbagai bidang seperti perencanaan dan pengelolaan penggunaan lahan, sumber daya alam, lingkungan, transportasi, fasilitas kota, dan pelayanan umum lainnya. Peta adalah representasi grafis dari dunia nyata atau wilayah tertentu yang mencakup informasi seperti jalan, sungai, bangunan, dan wilayah administratif. Untuk memberikan ketepatan lokasi pada peta, SIG menggunakan konsep koordinat geografis, yaitu *latitude* dan *longitude*.

Latitude dan *longitude* adalah koordinat geografis yang memberikan lokasi spesifik suatu objek atau data dalam sistem. *Latitude* mengukur jarak utara atau selatan suatu titik dari garis khatulistiwa, sedangkan *longitude* mengukur jarak timur atau barat suatu titik dari meridian utama. Dengan menggunakan kedua koordinat ini, SIG memungkinkan pemetaan digital dengan akurasi tinggi, memungkinkan pengguna untuk menentukan posisi yang tepat pada permukaan bumi. Dalam SIG, data geografis, seperti jalan, sungai, bangunan, dan wilayah administratif, dikombinasikan dengan informasi lain seperti demografi dan ekonomi untuk membentuk informasi komprehensif. *Latitude* dan *longitude*, sebagai koordinat geografis, memberikan lokasi spesifik dari setiap objek atau data dalam sistem memungkinkan pemetaan digital dengan akurasi tinggi. Dalam hal ini, SIG membantu ahli lingkungan, perencana wilayah, dan pembuat kebijakan dalam memahami pola dan tren data geografis, memungkinkan analisis seperti lalu lintas, lingkungan, dan pasar. Output SIG berupa peta atau grafik yang efektif untuk menyimpan, memvisualisasikan, dan menyajikan informasi geografis. Sebelum menggunakan SIG, penting untuk memahami teknologi dan metode yang digunakan dalam sistem ini.

Secara umum, definisi SIG (Sistem Informasi Geografis) adalah sebuah elemen yang terdiri dari perangkat keras, perangkat lunak, sumber daya manusia, dan data yang berkolaborasi dengan efektif untuk mengumpulkan, menyimpan,

memperbarui, mengelola, memanipulasi, mengintegrasikan, menganalisis, dan menampilkan data dalam konteks informasi berbasis geografis (Rosdania, 2015).

2.3 Rumah Kos

Rumah kos merupakan salah satu bentuk penginapan yang terdiri dari kamar atau apartemen yang disewakan untuk individu atau kelompok individu dalam jangka waktu tertentu. Biasanya, rumah kos menyediakan fasilitas umum seperti dapur, ruang tamu, dan kamar mandi yang digunakan bersama oleh para penghuni. Penginapan ini sangat populer di kota-kota besar harga properti tinggi dan jarak antara tempat kerja dan hunian cukup jauh. Rumah kos memberikan penghuni kesempatan untuk tinggal dengan biaya yang lebih rendah dibandingkan membeli atau menyewa apartemen penuh. Terlebih lagi, rumah kos seringkali menjadi pilihan bagi para mahasiswa, karena memberikan fleksibilitas dan kemudahan bagi mereka untuk tinggal dekat dengan kampus.

Cara penyewaan rumah kos bervariasi dari negara ke negara dan bahkan dari kota ke kota. Beberapa rumah kos memiliki pemilik yang mengelola penginapan, sementara yang lain dikelola oleh agen properti. Sebagai calon penyewa, ada beberapa hal yang perlu dipertimbangkan sebelum memilih rumah kos, seperti lokasi, harga, fasilitas, dan kondisi kamar. Selain itu, penting juga untuk membaca dan memahami perjanjian sewa sebelum menandatangani kontrak. Meskipun rumah kos dapat memberikan keuntungan bagi para penyewa, ada beberapa risiko yang perlu diwaspadai. Beberapa masalah yang sering dihadapi adalah masalah keamanan, kerusakan fasilitas umum, dan ketidaknyamanan akibat tinggal bersama dengan orang yang tidak dikenal. Oleh karena itu, penting untuk memilih rumah kos yang terpercaya dan memiliki reputasi yang baik di kalangan penyewa sebelum memutuskan untuk tinggal di sana.

Secara keseluruhan, rumah kos adalah salah satu pilihan penginapan yang praktis dan terjangkau, terutama bagi mereka yang membutuhkan fleksibilitas dan kemudahan dalam mencari tempat tinggal. Namun, sebelum memutuskan untuk tinggal di rumah kos, perlu dilakukan penilaian dan perencanaan yang baik untuk menghindari risiko dan masalah yang mungkin timbul.

2.4 Web

Sejarah *web* dimulai pada tahun 1989, ketika Tim Berners-Lee, seorang ilmuwan komputer di CERN, mengembangkan ide untuk membuat sistem informasi yang memungkinkan pemakaian dokumen dan informasi yang saling terkait melalui *internet*. Berners-Lee menamakan sistem ini "*World Wide Web*" atau WWW. Pada tahun 1991, Berners-Lee memperkenalkan teknologi dasar *web*, termasuk HTML (*Hypertext Markup Language*) sebagai bahasa *markup* untuk memformat halaman *web*, URL (*Uniform Resource Locator*) sebagai alamat halaman *web*, dan HTTP (*Hypertext Transfer Protocol*) sebagai protokol untuk mentransfer data melalui *internet*. Pada tahun 1993, *Mosaic*, peramban *web* pertama yang menampilkan grafik dan multimedia, dirilis dan membuka jalan bagi perkembangan peramban *web* seperti *Netscape Navigator* dan *Internet Explorer*. Sejak saat itu, *web* terus berkembang dan mempengaruhi bagaimana orang mengakses dan berbagi informasi. Pada awal 2000-an, perusahaan seperti *Google*, *Amazon*, dan *Facebook* mulai beroperasi dan membuat *web* menjadi bagian yang tidak terpisahkan dari kehidupan sehari-hari.

Situs *web* adalah sebuah platform informasi yang terdapat di *internet*. Selain berfungsi sebagai saluran penyebaran informasi, situs *web* juga dapat digunakan untuk mendirikan toko daring. Sebuah situs *web* merupakan koleksi halaman yang tergabung dalam suatu *domain* atau *subdomain*, yang terletak di *World Wide Web (WWW)* di *Internet*. Situs *web* mengandung informasi dalam bentuk teks, gambar, dan video yang dapat diakses melalui mesin pencari atau peramban *internet*. Setiap halaman *web* merupakan dokumen yang ditulis dalam format HTML (*Hyper Text Markup Language*) dan biasanya dapat diakses melalui protokol HTTP. Protokol ini mengirimkan informasi dari *server* situs *web* untuk ditampilkan kepada pengguna melalui peramban *web*. Publikasi-publikasi dari berbagai situs *web* dapat membentuk suatu jaringan informasi yang luas (Trimarsiah, 2017).

2.5 *Django*

Django adalah *web framework Python* yang diciptakan pada tahun 2003 di Lawrence, Kansas, dengan tujuan untuk membuat situs *web* dinamis yang sangat cepat. Pada tahun 2005, sumber kode *Django* dipublikasikan di lisensi BSD. *Django Software Foundation* didirikan pada tahun 2008 untuk mendukung dan memajukan *Django*, dan versi 1.0 dirilis beberapa bulan kemudian. *Django* menggunakan pola MVC dan menjadi pilihan utama banyak pengembang aplikasi *web*, termasuk organisasi besar seperti *Instagram*, Mozilla.org, dan Openstack.org.

Django menggunakan pola *Model-View-Controller* (MVC) untuk menyediakan arsitektur yang koheren. Hingga tahun 2013, *Django* hanya kompatibel dengan *Python* versi 2.x. Namun, dengan rilis *Django* 1.5 pada 26 Februari 2013, dukungan untuk *Python* 3 dimulai. Saat ini, banyak organisasi besar seperti Instagram, *Mozilla.org*, dan *Openstack.org* menggunakan *Django* untuk membuat situs *web* mereka dan dalam perkembangannya hingga saat ini, *django* telah merilis versi 4.1.7 sebagai versi terbarunya.

Geo*Django* adalah modul *contrib* yang disertakan dalam *Django* dan mengubahnya menjadi kerangka *web* geografis. Fungsi yang ditambahkan ke dalam *Django* termasuk perluasan *Model* untuk mendukung penyimpanan dan pengambilan data geospasial, ORM untuk mendukung *query spasial*, sistem Template untuk menampilkan data geospasial menggunakan peta *OpenLayers Slippy*, Antarmuka *admin* untuk membuat dan mengedit data geospasial, serta Jarak dan *area* kalkulator untuk mengkonversi antara berbagai unit standar.

Berikut adalah peningkatan fungsi-fungsi terbaru yang telah dimasukkan ke dalam *Django* untuk mendukung data geospasial:

1. Model
    a. *Model Django* diperluas untuk menyimpan dan mengambil data geospasial.
    b. *Object Relational Mapping (ORM) Django* diperluas untuk mendukung kueri spasial.
    c. *Object relational mapper* secara otomatis mengonversi data geospasial dari basis data menjadi objek GEOS, memungkinkan metode kueri dan manipulasi data ini dengan cara yang canggih.

d. *Model* dapat mengimpor data dari sumber data vektor yang didukung oleh OGR ke *database GeoDjango*.

2. Template

   Sistem template *django* diperluas untuk memungkinkan tampilan data geospasial menggunakan peta *slippy openlayers* yang terintegrasi.

3. Antarmuka *Admin*

   Antarmuka *admin Django* diperluas untuk memfasilitasi pengguna dalam membuat dan mengedit data geospasial dengan menggunakan *OpenLayers*. Data vektor ditampilkan di atas peta dasar yang disediakan oleh *OpenStreetMap*.

4. Kalkulator Jarak dan *Area*

   Jarak dan *area* dapat diubah antara berbagai unit standar, seperti *millimeter*, *yards*, atau mil.

2.6 React Js

ReactJS, juga dikenal dengan nama *React* atau *ReactJS*, adalah pustaka *java script* bersifat *open source* digunakan untuk membangun antarmuka pengguna. Beberapa versi yang penting pada *react* antara lain v0.1 sebagai versi awal, v0.14 yang menambahkan dukungan untuk *stateless functional components* dan *context,* v16 yang menambahkan dukungan untuk *error boundaries* dan portal serta meningkatkan kinerja, dan v17 yang menambahkan pengembangan mode async rendering dan beberapa peningkatan API, dan versi terbaru dari *react* yaitu v18 yang menambahkan beberapa modifikasi pencapaian penting dan ekstra dan modifikasi yang diperlukan di wilayah penyedia sisi *server*.

ReactJS digunakan untuk menangani tampilan pada aplikasi berhalaman tunggal (*single page application*) dan pengembangan aplikasi *mobile* (Khuat 2018). Dengan *React JS*, pembuatan antarmuka pengguna interaktif dapat dilakukan dengan lebih mudah dan efisien. Anda dapat membuat tampilan sederhana untuk setiap *state* dalam aplikasi Anda, dan *React JS* akan secara otomatis memperbarui dan merender hanya komponen yang diperlukan ketika terjadi perubahan data. Hal ini memungkinkan aplikasi Anda untuk berjalan lebih cepat dan responsif, serta menghemat waktu dan usaha dalam pembuatan antarmuka pengguna yang kompleks. *React JS* juga memudahkan pengembangan aplikasi berbasis komponen,

sehingga Anda dapat memisahkan kode Anda ke dalam unit-unit yang terpisah dan mudah digunakan kembali. Dengan menggunakan teknologi ini, Anda dapat membangun aplikasi *web modern* yang menarik dan interaktif dengan lebih mudah dan efisien.

Beberapa Keunggulan *React Js*

1. Komponen *Reusable*

   React mempromosikan konsep pengembangan berbasis komponen, yang memungkinkan pengembang untuk membuat komponen UI yang dapat digunakan kembali. Ini mempermudah pengelolaan kode dan meningkatkan efisiensi pengembangan.

2. *Virtual* DOM

   React menggunakan *Virtual* DOM untuk meningkatkan kinerja. *Virtual* DOM adalah representasi virtual dari DOM yang ada di memori. Saat ada perubahan dalam aplikasi, *React* membuat perubahan terlebih dahulu di *Virtual* DOM, dan hanya memperbarui bagian-bagian yang berubah di DOM aktual. Ini membantu mengurangi waktu yang dibutuhkan untuk merender perubahan dan meningkatkan performa aplikasi.

3. JSX *(JavaScript* XML*):*

   JSX adalah ekstensi sintaksis *JavaScript* yang memungkinkan penulisan kode HTML di dalam *JavaScript*. Ini membuat kode lebih mudah dibaca dan memungkinkan penggunaan fitur-fitur *JavaScript modern* dalam pengembangan UI.

4. *Uni-directional Data Flow:*

   React mengikuti pola aliran data satu arah (uni-directional), yang berarti data mengalir dari satu arah saja. Ini mempermudah untuk melacak dan memahami aliran data dalam aplikasi.

5. *Ekosistem yang Kuat:*

   React memiliki ekosistem yang kaya dengan berbagai pustaka dan alat pendukung seperti *Redux* untuk manajemen keadaan, *React Router* untuk penanganan rute, dan banyak lagi. Ini membuatnya lebih mudah untuk membangun aplikasi web yang kompleks.

6. Mendukung SSR (*Server-Side Rendering*):

   React mendukung Server-Side Rendering, yang memungkinkan mengirimkan HTML yang telah dirender ke browser dari sisi server. Ini dapat meningkatkan kinerja dan pengalaman pengguna, terutama untuk aplikasi dengan tampilan halaman pertama yang cepat.

2.7 Leafletjs Library

Leaflet pertama kali diperkenalkan pada tahun 2011 dan mendukung berbagai *platform* seluler dan *desktop*, termasuk *HTML5* dan *CSS3*. Saat ini, versi terbaru *Leaflet* adalah 1.9.3. *Leaflet JS* merupakan perpustakaan *JavaScript open source* yang digunakan untuk membuat peta interaktif di situs *web*. Beberapa pengguna terkemuka *Leaflet* antara lain *FourSquare, Pinterest*, dan *Flickr*. Dengan *Leaflet*, pengembang tanpa pengalaman GIS dapat dengan mudah menampilkan peta web yang terstruktur dengan baik pada *server* publik, dengan opsi peta ubin yang dapat diperluas. *Leaflet* juga mampu memuat data fitur dari *file GeoJSON*, menata data tersebut, dan membuat lapisan interaktif seperti penanda dengan informasi tambahan yang muncul saat diklik. Vladimir Agafonkin, pengembang utama *Leaflet,* bergabung dengan *Mapbox* pada tahun 2013.

Leaflet adalah perpustakaan *JavaScript open source* pertama yang dirancang untuk membuat peta interaktif *mobile* dengan antarmuka yang ramah pengguna. Leaflet mencakup semua fitur yang diperlukan oleh pengembang atau pembuat peta berbasis *web* untuk membuat peta digital. Perpustakaan ini bekerja secara efisien di berbagai *platform mobile* dan *desktop*, dapat diintegrasikan dengan berbagai *plugin*, memiliki desain yang menarik, mudah digunakan, sederhana, dan memiliki sumber kode yang mudah dipahami (Cahyono, 2016). Peta digital ditampilkan menggunakan *Leaflet JavaScript* yang mendukung *file* berformat *GeoJSON* yang merupakan format data yang dapat menampung unsur-unsur geografis.

Leaflet sangat mendukung *platform mobile* dan *desktop, HTML5* dan *CSS3*, serta *OpenLayers* dan *Google Maps API* sebagai perpustakaan *JavaScript* yang umum digunakan untuk membangun aplikasi peta. Dengan *Leaflet*, pengembang tanpa latar belakang GIS dapat dengan mudah menampilkan peta interaktif berbasis *web* di *server*. *Leaflet* mampu menampilkan lapisan dari *file GeoJSON*,

memberikan gaya, dan membuat lapisan interaktif seperti menampilkan penanda dengan informasi tambahan yang muncul saat diklik.

Langkah-langkah sederhana dalam membuat *web* GIS sederhana dengan Leaflet melibatkan:

1. Memanggil perpustakaan *Leaflet*.
2. Menampilkan peta pada halaman *web*.
3. Menampilkan *file GeoJSON* pada peta *Leaflet.*

2.8 PostgreSql

PostgreSQL adalah sistem basis data yang terdistribusi secara bebas dengan lisensi BSD. Perangkat lunak ini adalah salah satu basis data paling populer saat ini, bersama dengan *MySQL* dan *Oracle*. *PostgreSQL* menyediakan fitur yang berguna untuk replikasi basis data, seperti DB *Mirro*r, PGPool, *Slony*, *PGCluster,* dan lain-lain.

PostgreSQL atau *Postgres* adalah salah satu *database* besar yang menawarkan skalabilitas, fleksibilitas, dan kinerja tinggi. *PostgreSQL* digunakan secara luas di berbagai *platform* dan didukung oleh banyak bahasa pemrograman. SQL di *PostgreSQL* berbeda dengan yang ditemukan pada DBMS umumnya. Perbedaan utama antara *PostgreSQL* dengan sistem relasional standar adalah arsitekturnya yang memungkinkan pengguna untuk menentukan *SQL* mereka sendiri, terutama dalam pembuatan *function* dan *stored procedure*. Hal ini dimungkinkan karena informasi yang disimpan oleh *PostgreSQL* tidak hanya tabel dan kolom, tetapi juga tipe, fungsi, metode akses, dan banyak lagi yang terkait dengan tabel dan kolom tersebut. Semua informasi ini tergabung dalam bentuk *class* yang dapat diubah oleh pengguna. Arsitektur ini dikenal dengan istilah *object-oriented.*

Keunggulan PostgreSQL dibandingkan dengan *database* lainnya (Utami, 2006) dapat dijelaskan sebagai berikut:

1. *PostgreSQL* menggunakan arsitektur multiproses (*forking*), yang menghasilkan tingkat stabilitas yang lebih tinggi.
2. Pada kondisi beban tinggi (jumlah koneksi simultan yang besar), kecepatan *PostgreSQL* sering kali mengungguli *MySQL* untuk kueri dengan klausa JOIN yang kompleks.

3. *PostgreSQL* menyertakan fitur berorientasi objek seperti pewarisan tabel dan tipe data, termasuk *array* yang dapat praktis untuk menyimpan sejumlah besar item data dalam satu rekaman.
4. *PostgreSQL* menyediakan hampir semua fitur *database* yang ditemukan dalam produk *database* komersial.
5. *PostgreSQL* memperkenalkan tipe data geometri (seperti titik, garis, lingkaran, poligon), yang bermanfaat untuk aplikasi ilmiah tertentu.

PostgreSQL memiliki keunikan yang membedakannya dari *database* gratis lainnya. Awalnya, *PostgreSQL* dimulai sebagai proyek bernama Postgres di *University of California*, Berkeley pada tahun 1986. Pada tahun 1989, versi 1 dari Postgres dirilis. Proyek Berkeley berakhir pada Postgres versi 4.2. Pada tahun 1994, Andrew Yu dan Jolly Chen menambahkan SQL ke Postgres dan menamakannya Postgres95, yang kemudian dirilis sebagai *open source*. Nama Postgres95 tidak digunakan lagi pada tahun 1996, dan digantikan oleh *PostgreSQL*, mencerminkan keterkaitan antara *PostgreSQL* dan *SQL*. Saat ini, versi terbaru *PostgreSQL* telah mencapai versi 15. *PostgreSQL* memiliki kemampuan untuk terhubung dengan berbagai bahasa pemrograman.

2.9 Metode *WaterFall*

Metode *waterfall* adalah sebuah *Model* proses pengembangan perangkat lunak yang bersifat liniar dan sekuensial. Metode ini terdiri dari lima tahap utama yaitu analisis kebutuhan, perancangan, implementasi, pengujian, dan pemeliharaan. Setiap tahapan harus selesai terlebih dahulu sebelum tahap berikutnya dimulai. Metode ini sangat terstruktur dan fokus pada perencanaan sebelum implementasi, sehingga dapat membantu mengurangi risiko dan memastikan kualitas produk yang dihasilkan. Namun, metode *waterfall* juga dapat menghambat fleksibilitas dan adaptabilitas dalam menghadapi perubahan kebutuhan atau perubahan lingkungan bisnis yang cepat.

Aktivitas-aktivitas yang terdapat dalam model *waterfall* dapat diuraikan sebagai berikut (Amrin, 2020):

1. Komunikasi (*Communication*):
    a. Analisis kebutuhan perangkat lunak dan tahap pengumpulan data.

b. Melibatkan pertemuan dengan pelanggan dan pengumpulan data tambahan dari berbagai sumber seperti jurnal, artikel, dan *internet*.

2. Perencanaan (*Planning*):
   a. Proses perencanaan merupakan kelanjutan dari analisis kebutuhan.
   b. Menghasilkan dokumen kebutuhan pengguna atau data terkait dengan keinginan pengguna dalam pembuatan perangkat lunak, termasuk rencana pelaksanaan.
3. Pemodelan (*Modeling*):
   a. Proses pemodelan mentranslasikan persyaratan kebutuhan menjadi desain perangkat lunak yang dapat diperkirakan sebelum melakukan pengkodean.
   b. Fokus pada perancangan struktur data, arsitektur perangkat lunak, representasi antarmuka, dan rincian prosedural (algoritma).
   c. Menghasilkan dokumen yang disebut kebutuhan perangkat lunak.
4. Konstruksi (*Construction*):
   a. Konstruksi melibatkan proses pembuatan kode.
   b. Pengkodean merupakan terjemahan desain ke dalam bahasa yang dapat dipahami oleh komputer.
   c. Program menerjemahkan transaksi yang diminta oleh pengguna.
   d. Tahapan ini merupakan tahap yang nyata dalam pengembangan perangkat lunak.
   e. Setelah pengkodean selesai, dilakukan pengujian terhadap sistem yang telah dibuat untuk menemukan dan memperbaiki kesalahan.
5. Implementasi (*Deployment*):
   a. Tahap ini dapat dianggap sebagai tahap akhir dalam pembuatan perangkat lunak atau sistem.
   b. Setelah melalui analisis, desain, dan pengkodean, sistem yang telah selesai akan digunakan oleh pengguna.
   c. Selanjutnya, perangkat lunak yang telah dibuat harus dipelihara secara berkala.berkala.

2.10 Black Box

Black box testing, atau yang juga dikenal sebagai *behavioral testing*, adalah sebuah teknik pengujian perangkat lunak yang dilakukan dengan mengamati masukan dan keluaran (*input* dan *output*) perangkat lunak tanpa mengetahui rincian atau struktur kode di dalamnya. Pengujian ini dilakukan sebagai salah satu tahapan pengujian akhir pada tahap pengembangan perangkat lunak, dengan tujuan untuk mengetahui apakah perangkat lunak dapat berfungsi dengan baik sesuai dengan spesifikasi dan harapan pengguna. Metode pengujian *blackbox* adalah salah satu metode yang sederhana karena hanya memerlukan penentuan batas bawah dan batas atas dari data yang diharapkan. Estimasi jumlah data uji dapat dihitung berdasarkan jumlah bidang data *entry* yang akan diuji, aturan entri yang harus dipenuhi, serta skenario batas atas dan batas bawah yang relevan (Colifah, 2018).

Pengujian *black box* adalah teknik pengujian perangkat lunak yang melihat sistem secara keseluruhan dari perspektif pengguna tanpa perlu memeriksa kode program. Beberapa keuntungan dari penggunaan *black box testing* adalah:

1. Penguji tidak memerlukan pengetahuan tentang bahasa pemrograman yang digunakan dalam pembuatan perangkat lunak, sehingga dapat dilakukan oleh orang yang tidak memiliki latar belakang teknis yang kuat.
2. Pengujian dilakukan dari sudut pandang pengguna, sehingga dapat menemukan inkonsistensi dalam perangkat lunak yang tidak terlihat dari perspektif pengembang.
3. Ketergantungan antara pengembang dan penguji dapat ditingkatkan melalui penggunaan *black box testing* karena keduanya harus saling bekerja sama untuk memastikan bahwa perangkat lunak berfungsi dengan baik dari perspektif pengguna.
4. Penguji tidak perlu memeriksa kode program untuk melakukan pengujian, sehingga waktu yang diperlukan untuk melakukan pengujian dapat dikurangi.
5. *Black box testing* memungkinkan pengguna dan pengembang untuk bekerja secara independen tanpa mengganggu proses kerja satu sama lain, sehingga pengembang dapat terus membangun sistem dan memperbaiki *bug* sedangkan penguji dapat terus melakukan pengujian secara independen.

2.11 Penelitian Terkait

Penelitian dilakukan oleh Abidin, dkk (2012) dalam penelitiannya yang berjudul "perancangan dan implementasi sistem informasi kos *online*" untuk wilayah kota Surabaya Selatan. Sistem ini membantu pengguna dalam mendapatkan informasi mengenai tempat dan kamar kos yang sesuai dengan pemesanan kamar lewat media SMS (*Short Message Service*), serta membantu pemilik kost dalam memasarkan rumah kostnya secara *online* (Abidin, 2012).

Penelitian yang dilakukan oleh Pratikto, dkk (2014) dalam penelitiannya yang berjudul "Sistem Pencarian Dan Pemesanan Rumah Kos Menggunakan Sistem Informasi Geografi (SIG)". Sistem ini membantu pendatang di Kota Yogyakarta dalam mencari dan memesan rumah kos yang sesuai dengan kebutuhan mereka. Selain itu, sistem ini juga membantu pemilik kos dalam memasarkan rumah kos secara *online*, menggunakan *framework CodeIgniter* dan *Google Maps* (Pratikto, 2014).

Penelitian yang dilakukan oleh Sianturi, dkk (2018) dalam penelitiannya yang berjudul "Aplikasi Pencarian dan Penyewaan Rumah Kos Berbasis *Web* dan *Android*". Sistem ini dapat membantu dalam melakukan pencarian dan penyewaan rumah kos yang diinginkan dan pemilik terbantu dalam melakukan promosi rumah kos (Sianturi, 2018).

BAB III
METODE PENELITIAN

3.1 Waktu dan Lokasi Penelitian

Pelaksanaan penelitian ini dimulai pada Februari 2023. Penelitian ini dilaksanakan di Laboratorium Rekayasa Perangkat Lunak, Program Studi Sistem Informasi, Departemen Matematika, Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Hasanuddin.

3.2 Tahapan Penelitian

Penelitian ini menggunakan metode pengembangan sistem *waterfall*. Metode ini terdiri dari beberapa tahapan yang saling terhubung, yaitu:

3.2.1 Analisis Kebutuhan Sistem

Penelitian pada tahap ini mengacu pada proses identifikasi, pemahaman, dan dokumentasi kebutuhan atau persyaratan yang harus dipenuhi oleh suatu sistem untuk mencapai tujuan tertentu atau memecahkan masalah tertentu. Proses ini melibatkan berbagai pihak, termasuk pemangku kepentingan (*stakeholders*) yang dapat mencakup pengguna akhir dan pihak terkait lainnya.

Analisis kebutuhan dapat dilakukan dengan melakukan pencarian informasi penting terkait dengan rumah kos, pencarian itu bisa berupa melakukan tanya jawab dengan pemilik kos, *survey* lokasi, mencari informasi di *internet* serta membaca penelitian yang sudah ada yang terkait dengan penelitian rumah kos. Tahapan ini juga dilakukan dengan mengumpulkan kebutuhan yang diperlukan oleh pengguna dan perangkat lunak yang digunakan untuk memenuhi kebutuhan tersebut. Pada tahapan ini, peneliti mengidentifikasi masalah dan mengumpulkan kebutuhan yang diperlukan untuk mencapai tujuan sistem.

3.2.2 Perancangan (*Design*)

Tahapan ini lebih menekankan pada desain sistem secara menyeluruh. Desain dilakukan untuk melanjutkan tahap sebelumnya dan menjadi landasan dalam pembuatan program.

1. Membuat *use case* dengan tujuan untuk memahami bagaimana setiap aktor berinteraksi atau menunjukkan hubungan dari masing-masing aktor.
2. Membuat *activity diagram* dengan tujuan memahami langkah-langkah yang diperlukan dalam membangun sebuah sistem secara terstruktur.
3. Membuat *entity relationship diagram* (ERD) untuk memahami hubungan antara basis data yang akan dibuat.
4. Membuat rancangan antarmuka dengan tujuan memberikan gambaran awal dalam tahapan pembuatan desain *website.*

3.2.3 Pembuatan Kode Program

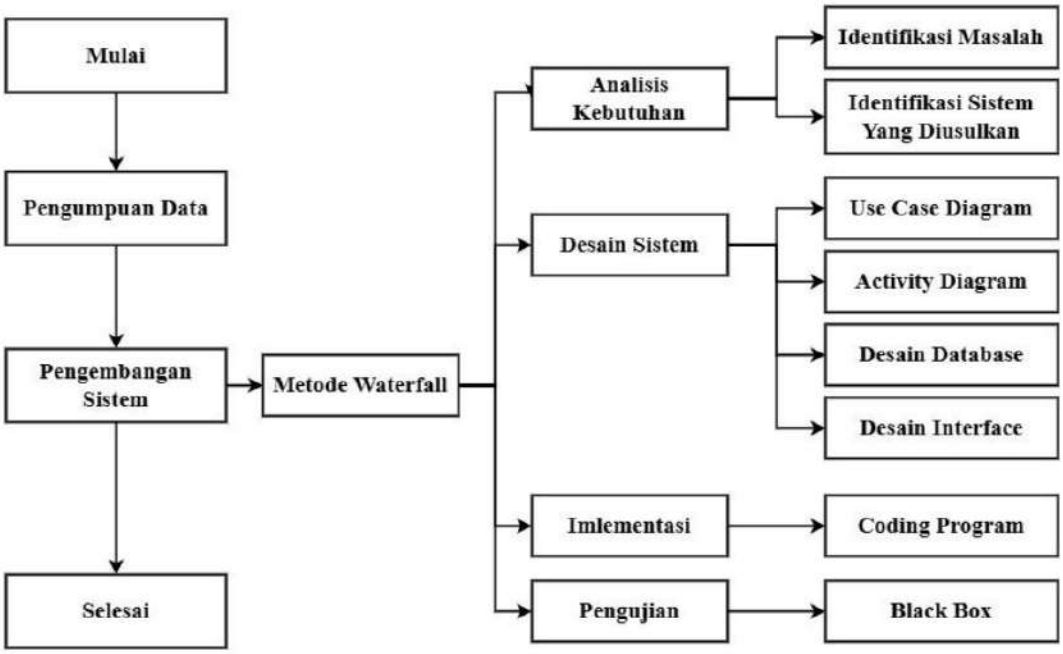
Tahap ini dilakukan dengan membuat kode program berdasarkan desain yang telah dibuat pada tahap desain. Pada tahap ini, penulis melakukan pengkodean dengan memanfaatkan *framework django* untuk *backend*, *ReactJs* untuk *frontend* dan *postgreSql* untuk *database*. Selama proses pembuatan kode program, penulis memastikan implementasi yang konsisten dengan spesifikasi desain dan menerapkan prinsip-prinsip pengembangan perangkat lunak yang baik. Selain itu, uji coba terus-menerus dilakukan untuk memastikan kehandalan dan fungsionalitas program yang sedang dikembangkan.

3.2.4 Pengujian

Pada tahap ini, dilakukan pengujian menggunakan metode *black box testing* untuk meminimalkan kesalahan dan memastikan keluaran yang dihasilkan sesuai dengan yang diinginkan. Pengujian ini bertujuan untuk mengetahui apakah sistem informasi pencarian rumah kos telah berjalan sesuai dengan perencanaan, mulai dari tahapan awal. Dengan menerapkan metode *black box testing*, pengujian dapat fokus pada fungsionalitas eksternal sistem tanpa memperhatikan implementasi internal, sehingga memastikan bahwa sistem memberikan respons yang sesuai dengan spesifikasi yang telah ditetapkan.

3.3 Alur Penelitian

Alur penelitian menjelaskan beberapa langkah yang dilakukan dalam pembangunan sistem secara terstruktur. Alur penelitian yang akan dilakukan sebagai berikut :



Gambar 3.1 Alur Penelitian

Gambar 3.1 memperlihatkan alur penelitian yang diawali dengan melakukan mengidentifikasi masalah yang akan terjadi dalam penelitian, dilanjutkan dengan pengumpulan data dan perancangan sistem, dan kemudian dilakukan implementasi dan pengujian sistem setelah itu dapat diperoleh kesimpulan dari penelitian yang dikerjakan.

3.4 Instrumen Penelitian

Adapun instrumen penelitian yang dilakukan dalam penelitian ini adalah :

1. Kebutuhan perangkat keras

   Spesifikasi perangkat keras sebagai alat pengembangan dan pengujian ialah laptop ASUS memiliki kapasitas:

   a) Intel(R) Core(TM) i3-6006U CPU @ 2.00GHz 1.99 GHz
   b) RAM 8GB

2. Kebutuhan perangkat lunak
   a) *Django*
   b) *Visual Studio code*
   c) *React Js*
   d) *PostgreSQL*
   e) *Web Browser* : *chrome, Mozilla Firefox*

3.5 Timeline Penelitian

Tahapan ini memperlihatkan rencana penyusunan aktivitas yang dilakukan selama proses pembuatan dan perancangan konsep penelitian serta periode waktu pelaksanaan seperti dalam **Tabel 3.1** berikut :

Tabel 3.1 Periode Pembuatan Penelitian

No	Kegiatan	Tahun 2023																													
		Februari 2023				Maret 2023				April 2023				Mei 2023				Juni 2023				Oktober 2023				November 2023				Desember 2023			
		1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
1	Pengumpulan dan Analisis Data																																
2	Rancangan Aplikasi																																
3	Pembuatan Aplikasi																																
4	Implementasi dan Pengujian																																

BAB IV
HASIL DAN PEMBAHASAN

4.1 Analisis Sistem

Analisis sistem adalah proses yang dilakukan untuk memahami, menguraikan, dan menganalisis sebuah sistem dengan tujuan mengidentifikasi masalah, kebutuhan, dan persyaratan yang terkait. Tujuan utama dari analisis sistem adalah untuk memperoleh pemahaman yang mendalam tentang bagaimana sistem beroperasi, bagaimana komponen-komponennya saling berinteraksi, serta mengidentifikasi *area* perbaikan atau pengembangan yang diperlukan.

Analisis sistem melibatkan berbagai langkah, metode, dan teknik untuk memahami sistem secara menyeluruh. Adapun analisis sistem yang dilakukan sebagai berikut:

4.1.1 Identifikasi Masalah

Setelah melakukan identifikasi masalah mengenai rancang bangun sistem informasi geografis untuk pencarian rumah kos berbasis *web*, dan mengidentifikasi beberapa kendala yang menjadi perhatian saat ini. Kendala-kendala tersebut antara lain:

1. Kurangnya aksesibilitas informasi, calon penghuni kos saat ini menghadapi kesulitan dalam mengakses informasi lengkap mengenai rumah kos yang tersedia. Informasi tersebut tersebar di berbagai *platform* dan sumber, yang membuat proses pencarian menjadi rumit dan memakan waktu.
2. Kurangnya integrasi data geografis, sistem yang ada saat ini belum efektif mengintegrasikan data geografis. Ini menghambat pengguna dalam mencari rumah kos berdasarkan lokasi yang diinginkan, sehingga menghambat efisiensi dalam pemilihan rumah kos yang sesuai dengan kebutuhan dan preferensi pengguna.
3. Kurangnya interaktivitas, pengguna tidak memiliki pengalaman interaktif yang memadai dalam melakukan pencarian rumah kos. Fitur interaktif, seperti peta interaktif, tampilan visual, dan filter yang canggih masih kurang tersedia,

sehingga pengguna kesulitan melakukan penelusuran yang lebih spesifik dan efisien.

4.1.2 Analisis Kebutuhan sistem

Analisis kebutuhan sistem digunakan dalam proses untuk mengidentifikasi, menggambarkan, dan menilai kebutuhan-kebutuhan yang harus dipenuhi oleh suatu sistem agar dapat mencapai tujuan atau memenuhi keperluan pengguna dengan efektif dan efisien. Rancang bangun sistem informasi geografis untuk pencarian rumah kos berbasis *web* membutukan kebutuhan sistem yang dapat membuat *user* dapat mengelolah dan menjalankan *website* pencarian rumah kos yang bisa dilihat pada Table 4.1.

Tabel 4.1 Analisis Kebutuhan Sistem

User	**Aktivitas**
Pengunjung	Pengunjung dapat melakukan pencarian rumah kos dan melihat informasi rumah kos berserta rute rumah kos.
Customer	*Customer* dapat melakukan semua yang dapat dilakukan pengunjung dan juga dapat melakukan transaksi kamar kos, *chat* pemilik kos, dan mereview rumah kos.
Pemilik Kos	Pemilik kos dapat mengelolah rumah kos dan kamar kos juga dapat menyetujui transaksi kamar kos.
Admin	*Admin* dapat mengelolah *user* dan dapat menambahkan *point of interest.*

Kebutuhan perangkat perlu dipertimbangkan dalam proses pembuatan *web* yang efektif. Beberapa kebutuhan yang harus dipertimbangkan antara lain:

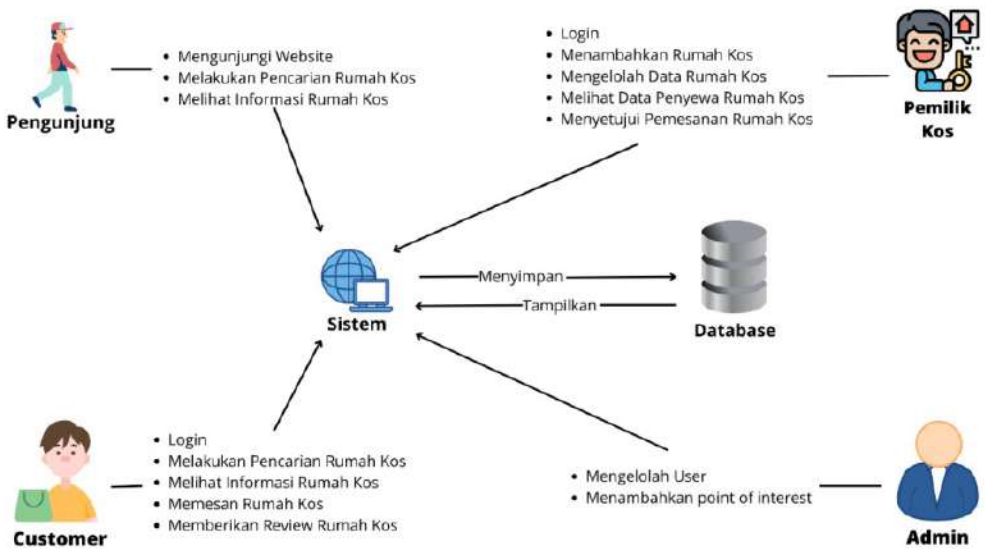
1. Kebutuhan perangkat keras, penelitian ini membutuhkan perangkat keras yang mampu mendukung operasional program dengan memenuhi spesifikasi minimum yang dibutuhkan. Ini mencakup perangkat keras yang dibutuhkan oleh pengembang dan pengguna akhir.

2. Kebutuhan perangkat lunak, penelitian ini menggunakan visual studio code sebagai *platform* pengembangan *web* dengan *framework Django* sebagai *backend*, dan *React JS* sebagai *frontend*. Penelitian ini juga menggunakan *PostgreSQL* untuk pengelolaan basis data.

Dengan memperhatikan kebutuhan-kebutuhan ini, kami dapat merancang dan mengembangkan sistem yang memenuhi harapan pengguna dan membantu mengatasi kendala-kendala yang ada dalam pencarian rumah kos berbasis geografis secara efektif.

4.1.3 Analisis Sistem yang Diusulkan

Analisis sistem yang diusulkan adalah proses yang dilakukan untuk menganalisis dan merencanakan sistem yang diusulkan sebagai solusi untuk mengatasi masalah atau memenuhi kebutuhan yang telah diidentifikasi. Analisis ini bertujuan untuk mengidentifikasi persyaratan sistem yang spesifik, merancang arsitektur dan komponen sistem yang sesuai, serta menyusun rencana implementasi dan pengembangan sistem yang diusulkan. Berikut sistem yang diusulkan untuk rancang bangun sistem informasi geografis nntuk pencarian rumah kos berbasis *web*, dapat dilihat pada Gambar 4.1



Gambar 4.1 Analisis Sistem yang Diusulkan

Analisis sistem yang diusulkan membantu dalam merancang solusi yang sesuai dan efektif untuk mengatasi masalah atau memenuhi kebutuhan yang telah diidentifikasi. Sistem yang diusulkan memiliki beberapa tugas yang dapat dilakukan pengguna dalam rancang bangun sistem informasi geografis Untuk pencarian rumah kos berbasis *web*, sebagai berikut:

a) Halaman utama memungkinkan pengunjung/pencari kos memiliki beberapa *menu* yang dapat diakses, seperti halaman *Home*, Pencarian Rumah Kos dll.
b) Pengunjung dapat melakukan pencarian rumah kos di *menu* pencarian rumah kos.
c) Halaman pencarian rumah kos, pengunjung dapat melihat beberapa informasi yang ditampilkan *website* seperti kolom pencarian, daftar rumah kos dan peta.
d) Pengunjung dapat memasukkan *input* di kolom pencarian untuk mencari rumah kos yang sesuai.
e) Pengunjung dapat melihat informasi rumah kos dengan lebih detail dengan menekan detail pada halaman pencarian rumah kos. Selanjutnya akan *system* akan menampilkan informasi detail tentang rumah kos beserta rute dari rumah kos, pada halaman ini juga akan menampilkan tempat strategis terdekat seperti universitas, rumah sakit dan tempat ibadah.
f) Memesan kamar dapat dilakukan dengan pengunjung/*customer* memilih *menu* pesan kamar pada halaman detail rumah kos. *Website* selanjutnya akan menampilkan halaman yang berisikan daftar kamar beserta informasinya, jika informasi kamar terdapat pesan 'belum dipesan' maka kamar tersebut dapat dipesan oleh *customer.*
g) Transaksi dapat dimulai dengan memilih *menu order* dan memasukkan *input* yang sesuai.
h) Sebelum melakukan proses transaksi pengunjung dapat menghubungi pemilik kos *via whatsapp* dengan memilih *menu whatsapp.*
i) Untuk melihat proses transaksi dapat melihat di *menu* 'Riwayat transaksi'. Dan jika terdapat pesan proses yang menandakan bahwa transaksi belum disetujui pemilik kos dan jika sukses maka proses transaksi telah setujui oleh pemilik kos.

j) Pengunjung yang memiliki rumah kos dapat mendaftar sebagai pemilik kos dan melakukan *login* dengan memasukkan nama *email* dan *password.*

k) Pemilik kos dapat memasukkan lebih dari satu rumah kos. Hal ini memungkinkan pemilik kos lebih mudah dalam mengelola rumah kos.

l) Dalam proses transaksi, pemilik kos dapat memeriksa dan menyetujui pemesanan kamar kos di *menu* pesan.

m) Sebelum menyetujui transaksi, pemilik kos dapat melakukan chat via whatsapp dalam memastikan dan mempermudah dalam menyetujui dan mengecek keaslian transaksi sebelum menyetujui transaksi.

n) Pemilik kos dapat melihat informasi kamar pada *menu* detail pada halaman data rumah kos.

o) Pemilik kos dapat melihat daftar penghuni kos di *menu* penghuni kos.

p) *Admin* dapat mengelola *we* dalam *website.*

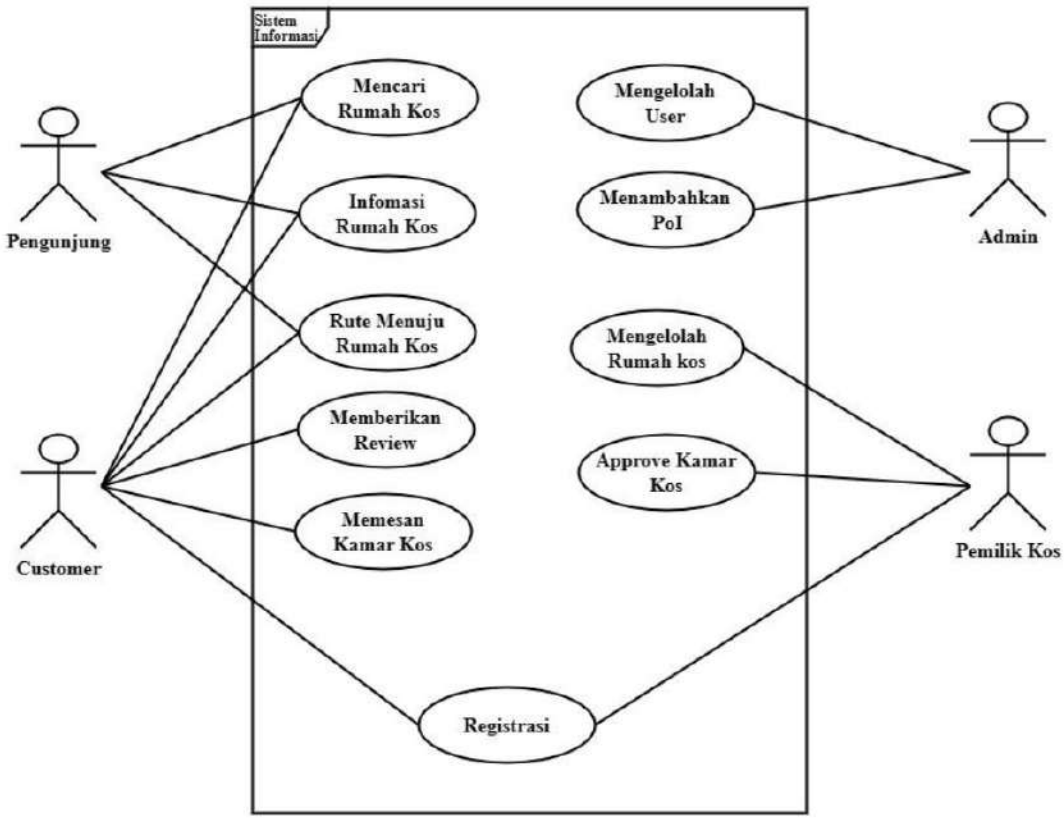
q) *Admin* dapat menambahkan *point of interest* dalam *website.*

4.2 Rancangan Sistem

Rancangan sistem adalah proses perencanaan dan perancangan sistem komputer atau aplikasi, yang dapat membantu memastikan bahwa sistem yang dikembangkan sesuai dengan kebutuhan pengguna, efisien, mudah dipelihara, dan dapat diandalkan. Adapun perancangan sistem melibatkan beberapa tahap sebagai berikut:

4.2.1 Use Case Diagram

Gambar 4.2 merupakan *use case* dari rancang bangun sistem informasi geografis untuk pencarian rumah kos berbasis *web* sebagai berikut:



Gambar 4.2 *Use Case* Pencarian Rumah Kos

Gambar 4.2 memperlihatkan bahwa pada aplikasi ini terdiri dari 4 pengguna yaitu *customer,* pengunjung, pemilik kos dan admin. Masing-masing dari pengguna ini, memiliki aksi yang berbeda. Misalnya yaitu pengunjung. Pengunjung dapat memasuki *website* tanpa *login* ataupun *registrasi* terlebih dahulu. Akan tetapi, pengunjung hanya memiliki sedikit pilihan yang dapat digunakan. Seperti, dapat

melakukan pencarian dan melihat sedikit informasi rumah kos. Berbeda dengan *customer* dan pemilik kos, yang memiliki pilihan yang lebih banyak dan diwajibkan untuk *login* dan *registrasi* terlebih dahulu.

Customer memiliki beberapa fitur yang dapat digunakan. Proses ini dimulai dengan mengakses *menu* pencarian. Tampilan pertama akan menampilkan beberapa *list* rumah sebagai tampilan awal. Customer dapat memasukkan beberapa kata kunci tertentu di *menu* pencarian, kemudian system akan menampilkan daftar pilihan rumah yang tersedia dan jika pilihan sudah ditentukan, maka akan ditampilkan informasi spesifik yang menggambarkan situasi rumah kos dan juga menunjukkan rute rumah kos.

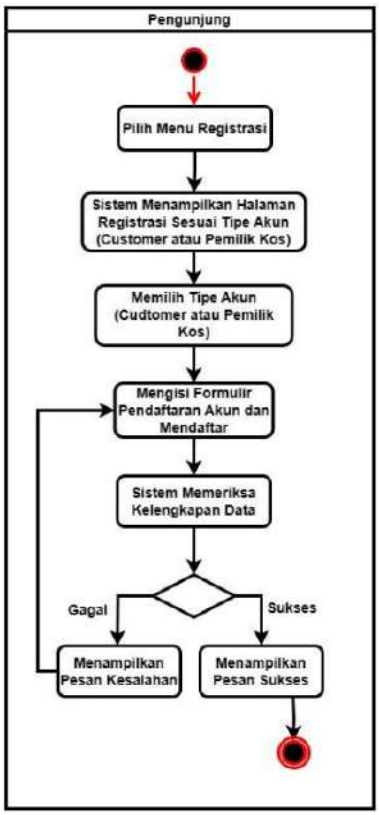
Pemilik kos dapat melakukan beberapa pilihan yang dapat memungkinkan pemilik kos dapat mengelola dengan baik kos yang dimilikinya. Seperti menambahkan, mengupdate, menghapus data kos yang tidak relevan lagi, menyetujui proses transaksi dll Pemilik kos selaku sebagai pemilik kos, dapat memiliki beberapa rumah kos yang dapat didaftarkan, hal ini dapat memudahkan pemilik kos dalam mengelola beberapa rumah kos yang dimiliki. Admin memiliki beberapa tugas yang dapat dilakukan seperti mengelah *user* dan menambahkan *point of interest* dalam *website*.

4.2.2 *Activity Diagram*

Activity diagram adalah jenis *diagram* yang digunakan untuk menggambarkan urutan aktivitas dalam suatu proses atau sistem. Berikut rancangan *activity diagram web* pencarian rumah kos:

1. *Activity Diagram* Registrasi

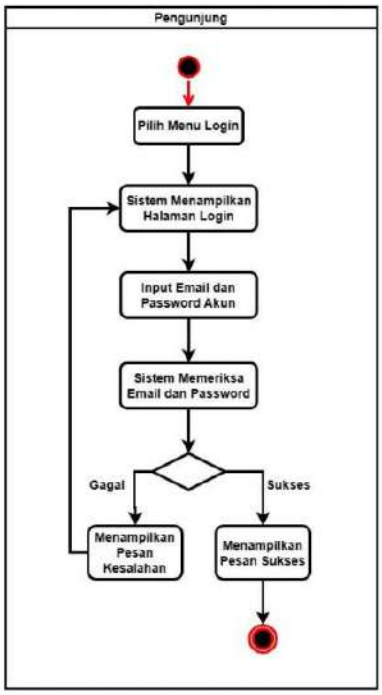
   Gambar 4.3 menunjukkan *activity diagram* untuk registrasi.



Gambar 4.3 *Activity Diagram* Registrasi

2. *Activity Diagram Login*

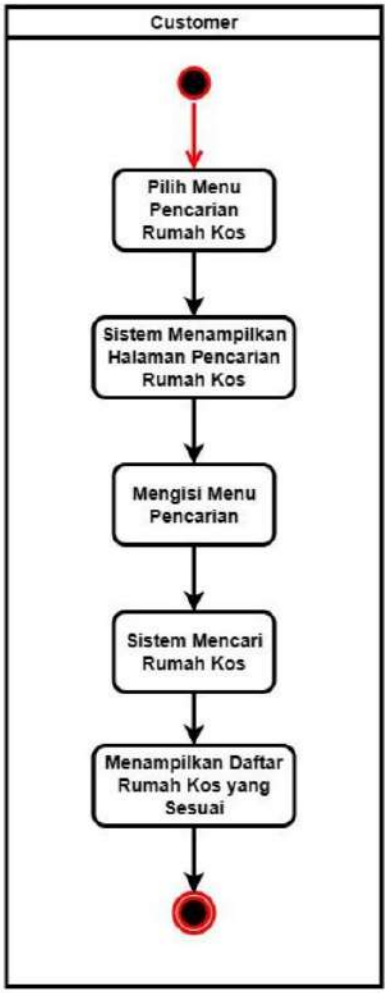
   Gambar 4.4 menunjukkan *activity diagram* untuk *login*. Proses ini menunjukkan aktivitas yang dapat dilakukan *customer* dan pemilik kos dalam memasuki *web* pencarian kos.



Gambar 4.4 *Activity Diagram Login*

3. *Activity Diagram* Pencarian Rumah Kos

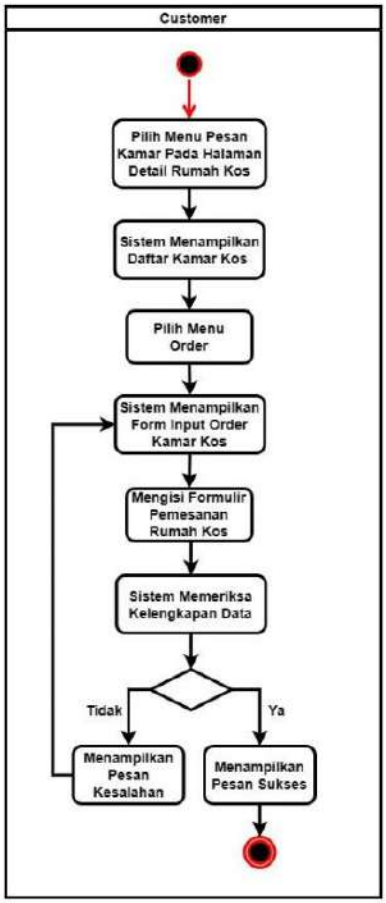
   Gambar 4.5 menunjukkan *activity diagram* untuk pencarian rumah kos. Proses ini menunjukkan aktivitas yang dapat dilakukan *customer* atau pengunjung untuk melakukan pencarian rumah kos.



Gambar 4.5 *Activity Diagram* Pencarian Rumah Kos

4. *Activity Diagram* Penyewaan Kamar Kos

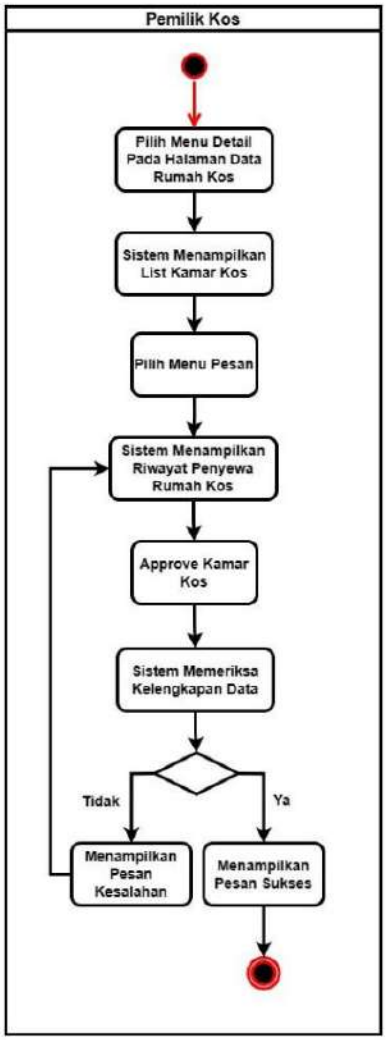
   Gambar 4.6 menunjukkan *activity diagram* untuk penyewaan kamar kos. Proses ini menunjukkan aktivitas yang dapat dilakukan *customer* untuk melakukan penyewaan kamar kos.



Gambar 4.6 *Activity Diagram* Penyewaan Rumah Kos

5. *Activity Diagram Approve* Sewa Kamar Kos

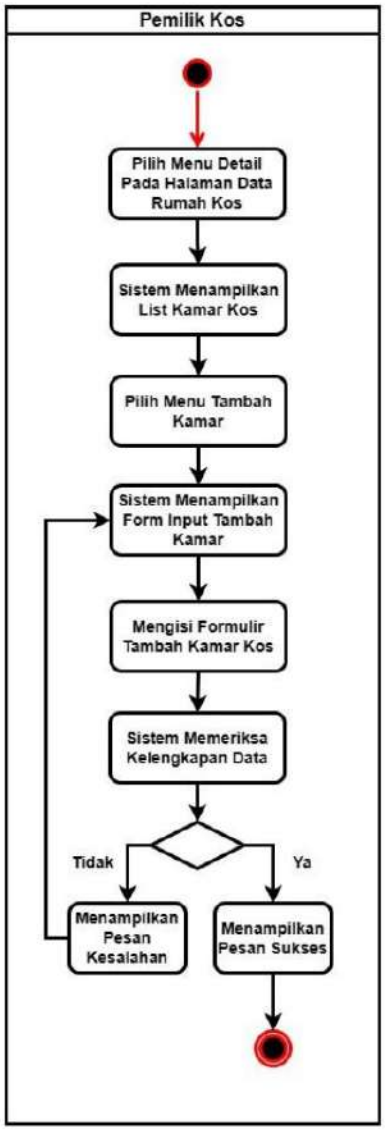
   Gambar 4.7 menunjukkan *activity diagram* untuk *approve* penyewaan kamar kos. Proses ini menunjukkan aktivitas yang dapat dilakukan pemilik kos untuk melakukan *approve* penyewaan kamar kos yang dilakukan *customer*.



Gambar 4.7 *Activity Diagram Approve* Penyewaan Rumah Kos

6. *Activity Diagram* Menambahkan Kamar Kos

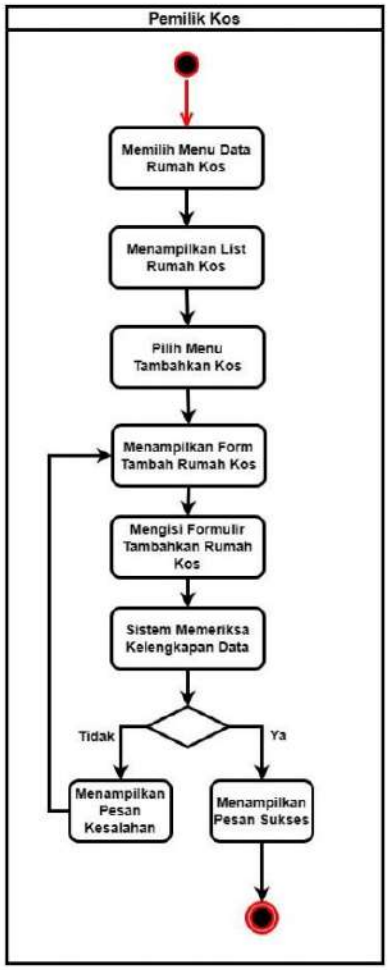
   Gambar 4.8 menunjukkan *activity diagram* untuk menambahkan kamar kos. Proses ini menunjukkan aktivitas yang dapat dilakukan pemilik kos untuk menambahkan kamar kos.



Gambar 4.8 *Activity Diagram* Menambahkan Kamar Kos

7. *Activity Diagram* Menambahkan Rumah Kos

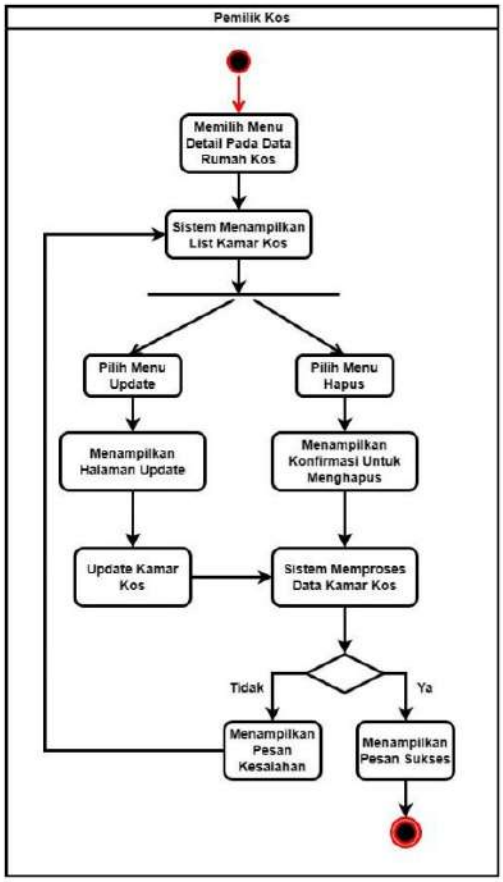
   Gambar 4.9 menunjukkan *activity diagram* untuk menambahkan rumah ros. Proses ini menunjukkan aktivitas yang dapat dilakukan pemilik kos untuk menambahkan rumah kos.



Gambar 4.9 *Activity Diagram* Menambahkan Rumah Kos

8. *Activity Diagram* Menghapus dan Mengupdate Kamar Kos

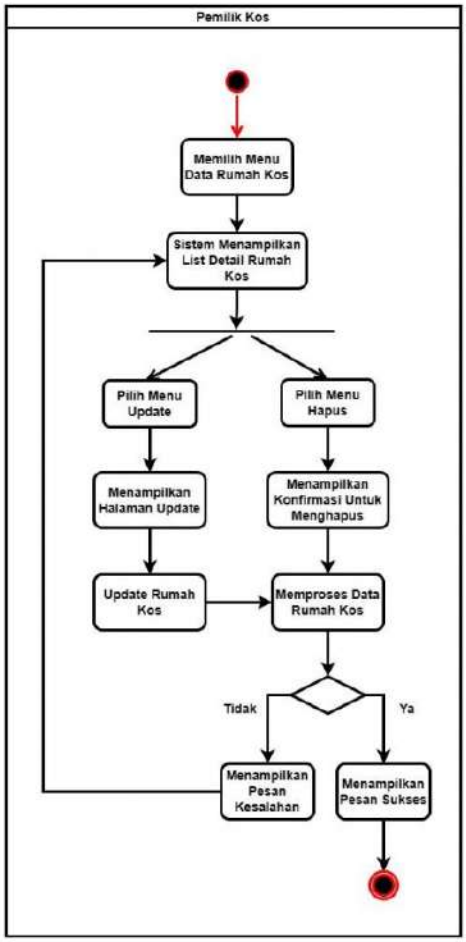
   Gambar 4.10 menunjukkan *activity diagram* untuk menghapus dan mengupdate kamar kos. Proses ini menunjukkan aktivitas yang dapat dilakukan pemilik kos untuk menghapus dan mengupdate kamar kos.



Gambar 4.10 *Activity Diagram* Menghapus dan *update* Kamar Kos

9. *Activity Diagram* Menghapus dan Mengupdate Rumah Kos

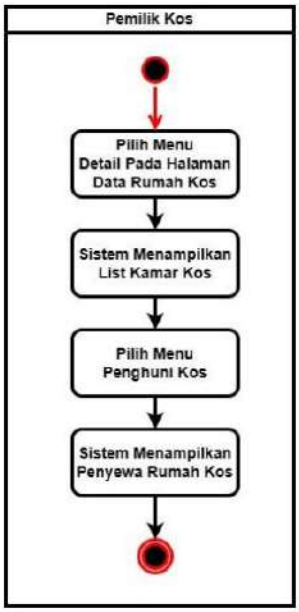
   Gambar 4.11 menunjukkan *activity diagram* untuk menghapus dan mengupdate rumah kos. Proses ini menunjukkan aktivitas yang dapat dilakukan pemilik kos untuk menghapus dan mengupdate rumah kos.



Gambar 4.11 *Activity Diagram* Menghapus dan *update* Rumah Kos

10. *Activity Diagram* Melihat Data Penyewa Rumah Kos

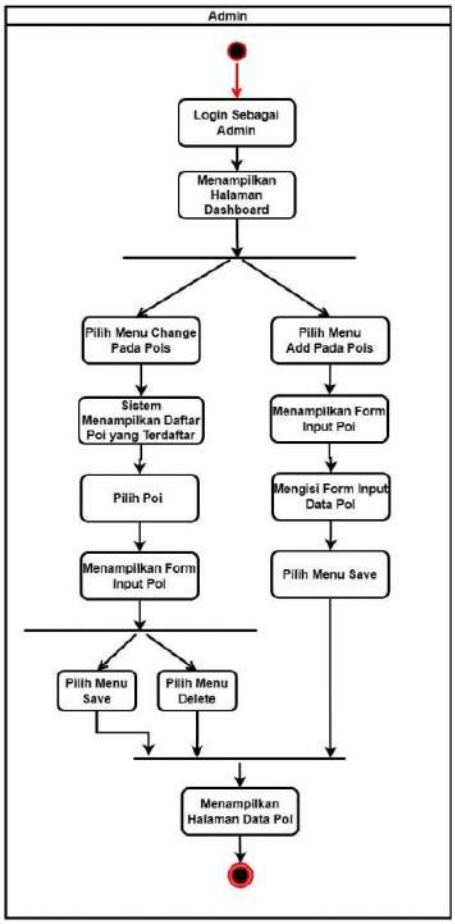
Gambar 4.12 menunjukkan *activity diagram* untuk melihat penyewa rumah kos. Proses ini menunjukkan aktivitas yang dapat dilakukan pemilik kos untuk melihat penyewa kamar kos.



Gambar 4.12 *Activity Diagram* Melihat Data Penyewa Rumah Kos

11. *Activity Diagram* Menambahkan *Point of Interest* (POI)

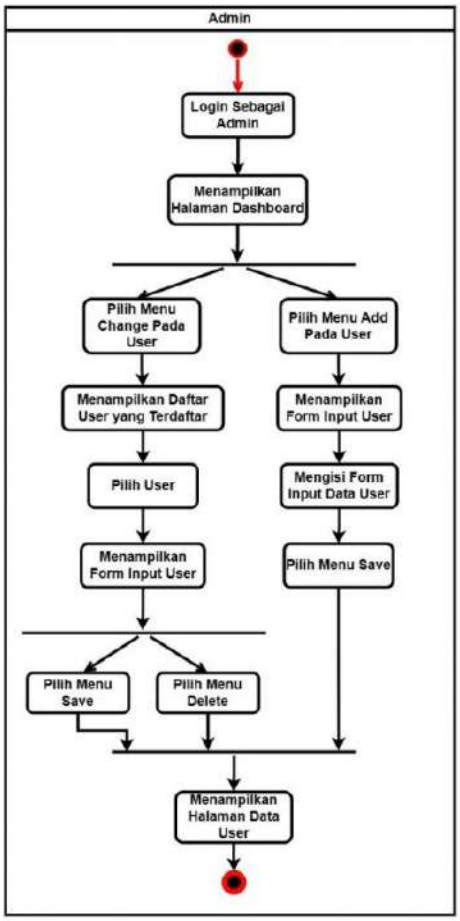
    Gambar 4.13 menunjukkan *activity diagram* untuk menambahkan *point of interest*. Proses ini menunjukkan aktivitas yang dapat dilakukan *admin* untuk menambahkan *point of interest*.



Gambar 4.13 *Activity Diagram* Menambahkan POI

12. *Activity Diagram Admin* Mengelola Akun

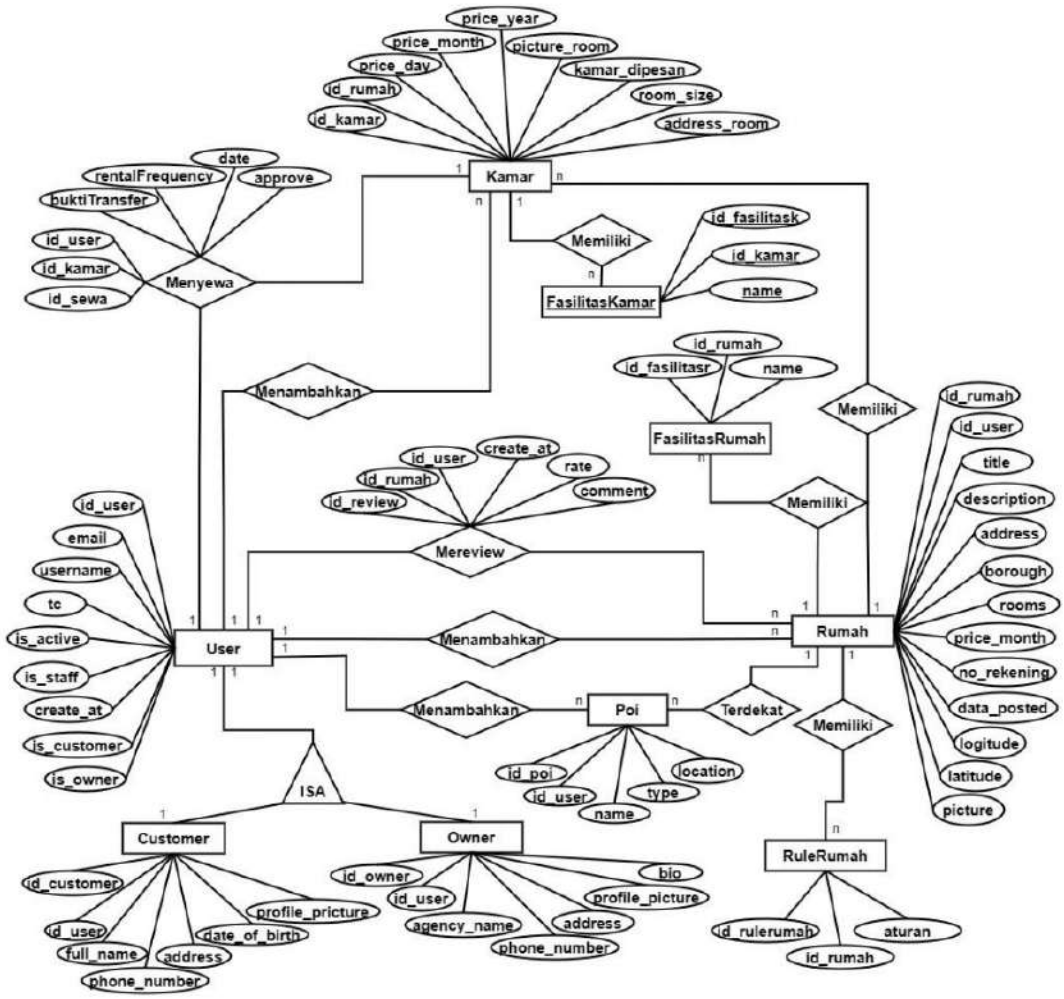
Gambar 4.14 menunjukkan *activity diagram* untuk *admin* dalam mengelola *user* dalam *website* rumah kos. Proses ini menunjukkan tugas *admin* dalam mengelola *user* dan beberapa fungsi lainnya.



Gambar 4.14 *Activity Diagram Admin* Mengelola *User*

4.2.3 Entity relationship *diagram* (ERD)

Entity Relationship Diagram (ERD) adalah *diagram* yang digunakan untuk menggambarkan hubungan antara entitas dalam suatu sistem atau basis data. ERD adalah alat visual yang membantu dalam merancang dan mengkomunikasikan struktur data, hubungan, dan atribut dalam suatu sistem. *Diagram* ini memberikan pemahaman yang jelas tentang bagaimana entitas berinteraksi satu sama lain dalam sistem, struktur data *entity relationship diagram* dilihat pada Gambar 4.15.



Gambar 4.15 *Entity Relationship Diagram*

4.2.4 Rancangan *Interface*

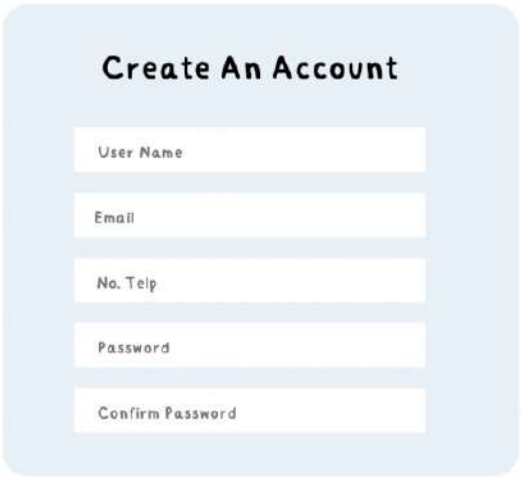
Rancangan antarmuka (*interface*) merujuk pada cara elemen-elemen suatu sistem berinteraksi atau berkomunikasi satu sama lain. Dalam konteks desain, rancangan antarmuka seringkali mengacu pada tata letak, struktur, dan interaksi antara elemen-elemen yang membentuk suatu produk atau sistem. Rancangan interface pada *website* sistem informasi geografis untuk pencarian rumah kos berbasis *web* sebagai berikut:

1. Halaman *Login* dan Registrasi

   Halaman ini menampilkan halaman *Login* dan Registrasi. Seperti yang terlihat dalam Gambar 4.16 terdapat halaman *login* yang diperlukan untuk masuk ke dalam halaman *website*. Jika pengunjung *web* sudah mempunyai akun, pengunjung dapat melakukan *login* pada halaman tersebut. Sementara itu, halaman registrasi dapat dilihat dalam Gambar 4.17 yang digunakan untuk mendaftarkan akun baru bagi yang pengunjung yang belum mempunyai akun.

   Website ini dapat dikunjungi tanpa melakukan registrasi atau *login* terlebih dahulu, akan tetapi pengunjung tidak akan dapat mengakses fitur-fitur khusus yang tersedia dalam *website*.

Gambar 4.16 Halaman *Menu Login*



Gambar 4.17 Halaman *Menu Registrasi*

2. Halaman Pencarian Rumah Kos

Halaman ini menampilkan rancangan tampilan untuk halaman pencarian rumah kos yang memungkinkan pengunjung atau *customer* untuk mencari rumah/kamar kos yang terdaftar di dalam *website*.

Gambar 4.18 Halaman Pencarian Rumah/Kamar kos

Gambar 4.18 menunjukkan rancangan tampilan halaman pencarian rumah kos. Halaman ini berisi *menu* pencarian dan daftar rumah serta dilengkapi dengan peta atau map yang dapat ditampilkan. Halaman ini dapat membantu

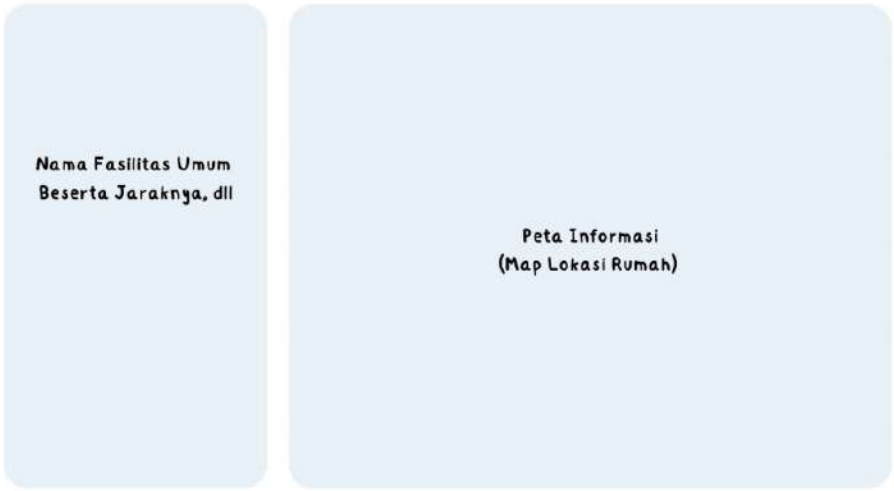
dalam proses pemilihan rumah kos, serta menunjukkan lokasi tepat rumah yang akan dituju dengan menggunakan *icon* tertentu pada peta.

3. Halaman Informasi Rumah Kos

Halaman ini menampilkan rancangan tampilan untuk halaman informasi rumah kos, yang memungkinkan pelanggan untuk melihat informasi detail tentang rumah dan kamar kos tertentu. Halaman ini dirancang untuk membantu pelanggan dalam memperoleh informasi yang mereka butuhkan sebelum memutuskan untuk menyewa rumah kos tersebut.

Gambar 4.19 Halaman Tampilan Informasi Rumah/Kamar Kos

Gambar 4.19 menunjukkan rancangan tampilan halaman informasi rumah/kamar kos. Halaman ini menampilkan informasi detail tentang rumah seperti harga, fasilitas, jumlah kamar kosong, deskripsi rumah, dan lain sebagainya. Selain itu, terdapat beberapa foto yang menunjukkan situasi asli dari rumah kos, seperti gambar rumah kos, kamar kos, parkiran, dan lain-lain. Informasi-informasi tersebut menjadi dasar bagi pengguna untuk memutuskan apakah rumah kos tersebut sesuai atau tidak.



Gambar 4.20 Tampilan Map Menuju Rumah Kos

Gambar 4.20 merupakan lanjutan dari rancangan tampilan informasi rumah kos. Tampilan ini menampilkan peta yang menunjukkan lokasi rumah kos dan rute perjalanan yang dapat diambil berdasarkan informasi yang ada. Selain itu, halaman ini juga menampilkan informasi mengenai nama dan fasilitas umum terdekat, seperti universitas, rumah ibadah, dan lain-lain. Halaman ini juga akan menampilkan nama fasilitas umum beserta jaraknya dari rumah kos. Dengan informasi tersebut, dapat membantu pelanggan dalam menentukan pilihan rumah kos yang tepat.

4. Halaman Pesan Kamar Kos

Halaman ini menampilkan rancangan tampilan untuk halaman pesan rumah kos, yang ditujukan untuk *customer* dalam memesan kamar kos.

Gambar 4.21 Tampilan Halaman Pesan Kamar Kos

Gambar 4.21 menunjukkan rancangan tampilan halaman pemesanan kamar kos. Halaman ini memungkinkan pelanggan untuk memesan kamar kos yang sesuai dengan kebutuhan mereka. Halaman ini menampilkan formulir yang harus diisi seperti nama, nomor telepon, durasi penyewaan, dan informasi lainnya yang relevan dengan pemesanan kamar kos. Pelanggan dapat mengisi formulir sesuai dengan kebutuhan mereka. Dengan adanya halaman ini, diharapkan dapat memudahkan pelanggan dalam memesan kamar kos secara *online*.

5. *Form* Halaman Menambahkan Rumah Kos

   Halaman ini menampilkan rancangan tampilan yang ditujukan untuk pemilik kos. Pemilik kos dapat menambahkan rumah kos dengan informasi tambahan yang sesuai dengan rumah kos yang akan ditambahkan.

Gambar 4.22 Halaman Menambahkan Rumah Kos

Gambar 4.22 merupakan *form* menambahkan rumah kos. Pemilik kos selaku yang mempunyai rumah kos dapat memasukkan rumah kosnya melalui halaman ini, dan dalam halaman ini terdapat *form* yang harus diisi dan sesuai dengan informasi tentang rumah kos yang akan dimasukkan. Informasi yang akan dimasukkan, haruslah relevan dengan keadaan rumah kos yang sekarang sehingga informasi yang dimasukkan akan menjadi tolak ukur bagi pengunjung untuk memilih rumah kos yang sesuai.

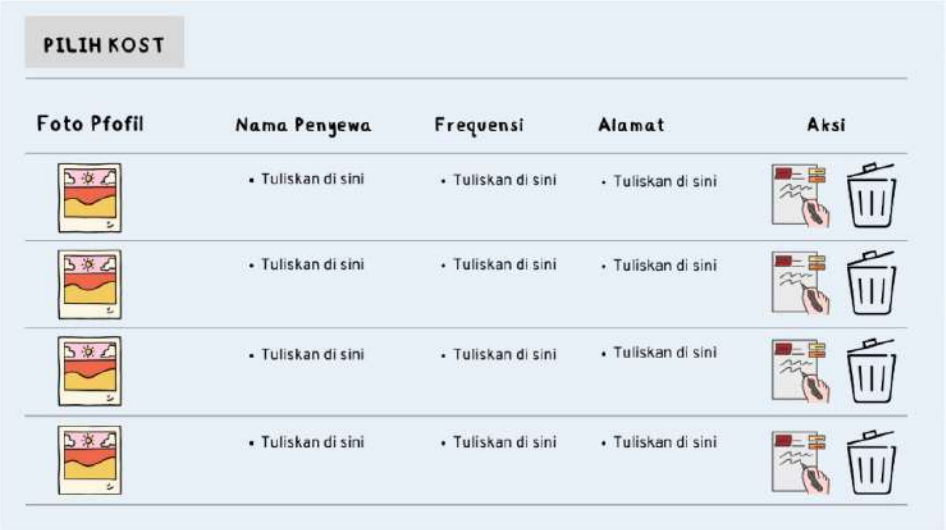
6. Halaman Data Rumah Kos

Halaman ini menampilkan rancangan halaman yang memungkinkan pemilik kos melakukan manajemen rumah kos, mulai dari menambahkan rumah kos, *update* rumah kos, menghapus rumah kos, dll.

Gambar 4.23 Tampilan Halaman Data Rumah Kos

Gambar 4.23 merupakan tampilan halaman data rumah kos. Halaman ini memungkinkan pemilik kos untuk menambahkan, *update*, dan menghapus rumah kos yang dimiliki oleh pemilik kos dan di halaman ini, juga akan menampilkan informasi dari rumah kos yang ditambahkan.

Gambar 4.24 Tampilan Laporan Data Kamar Kos



Gambar 4.25 Tampilan Halaman Data Kamar Kos

Gambar 4.24 menunjukkan tampilan pemilik kos yang dapat menyetujui atau mengkonfirmasi penyewa dari kamar kos. Calon penyewa terlebih dahulu melakukan pembayaran atau datang langsung ke rumah kos untuk melakukan pembayaran ditempat sebelum disetujui untuk menempati kamar kos. Gambar 4.25 menunjukkan daftar dari penyewa kamar kos yang sudah dikonfirmasi.

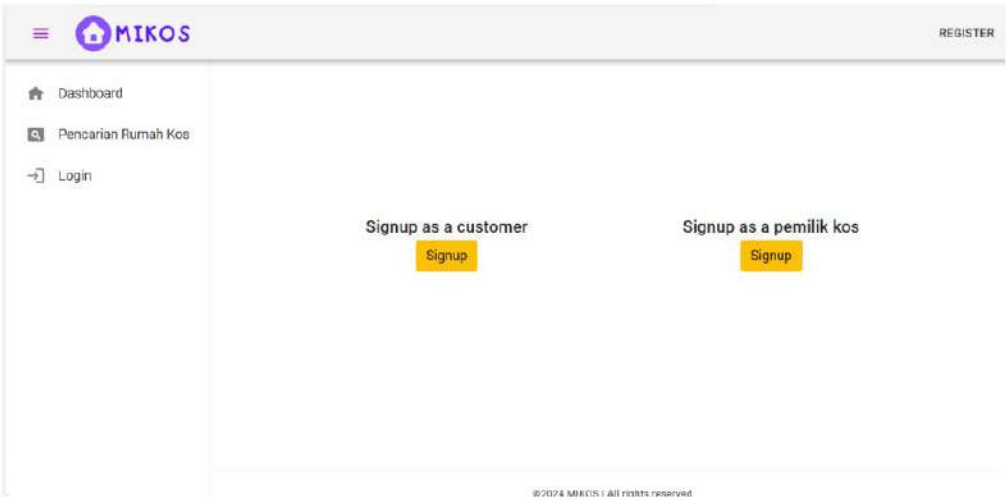
4.3 Implementasi Sistem

Berikut implementasi sistem dari sistem informasi geografis pencarian rumah kos berbasis *web*.

4.3.1 Authentication

1. Registrasi

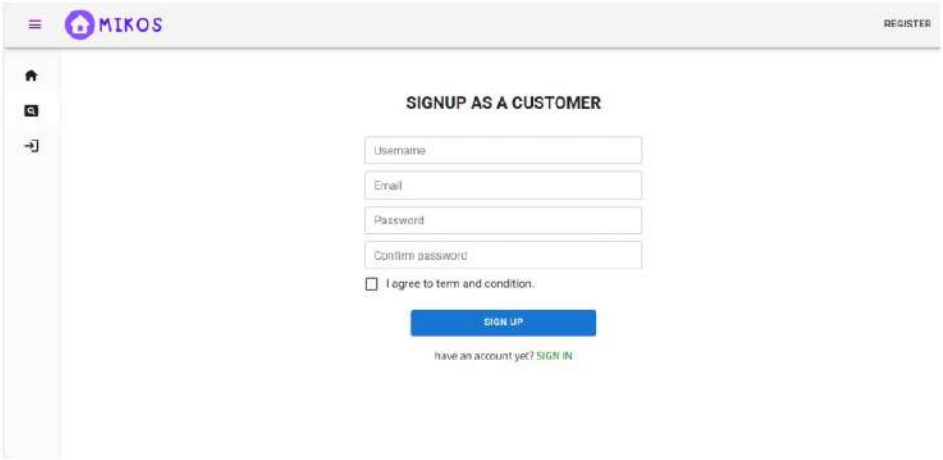
   Halaman registrasi dapat diakses oleh pengunjung yang ingin membuat akun *customer* atau pemilik kos.



Gambar 4.26 Halaman Registrasi

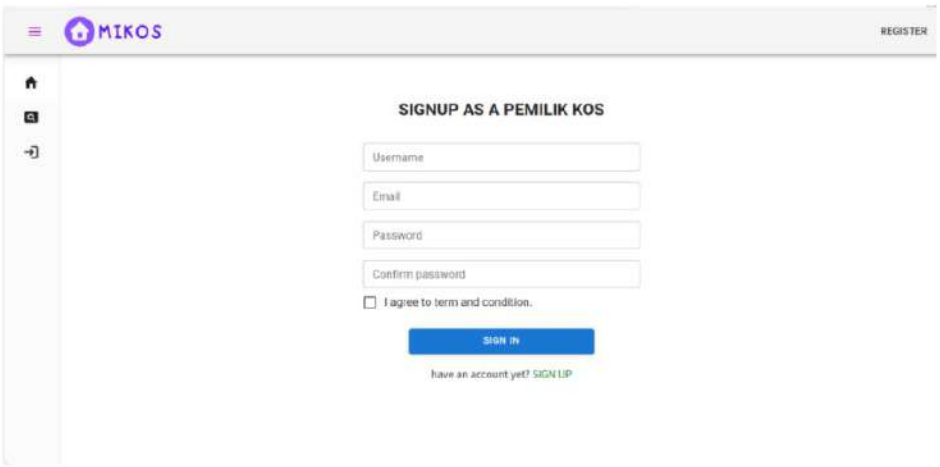
Gambar 4.26. menunjukkan dua pilihan untuk membuat jenis *user* yaitu *customer* dan pemilik kos.

a. Registrasi *customer*, halaman registrasi *customer* dapat diakses oleh pengunjung yang ingin membuat akun *customer*.



Gambar 4.27 Registrasi *Customer*

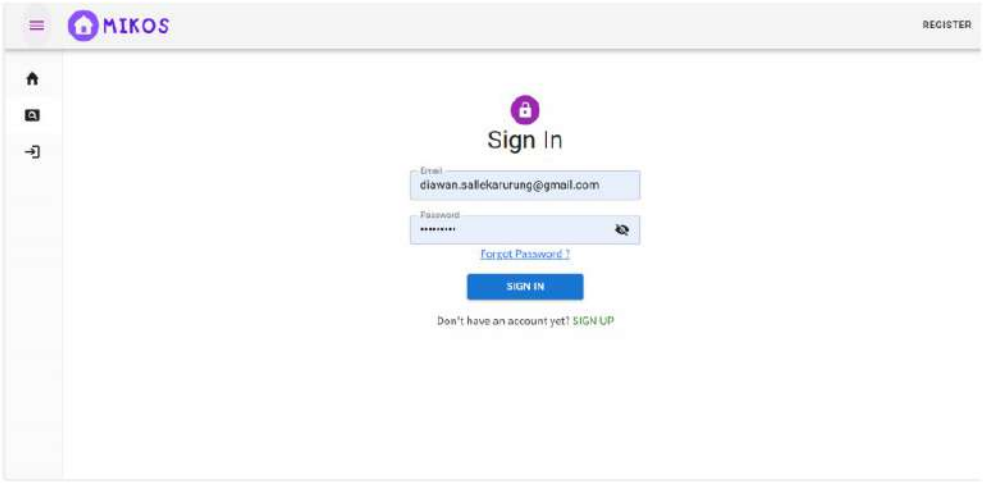
b. Registrasi pemilik kos, halaman registrasi pemilik kos dapat diakses oleh pengunjung yang ingin membuat akun pemilik kos.



Gambar 4.28 Registrasi Pemilik Kos

2. *Login*

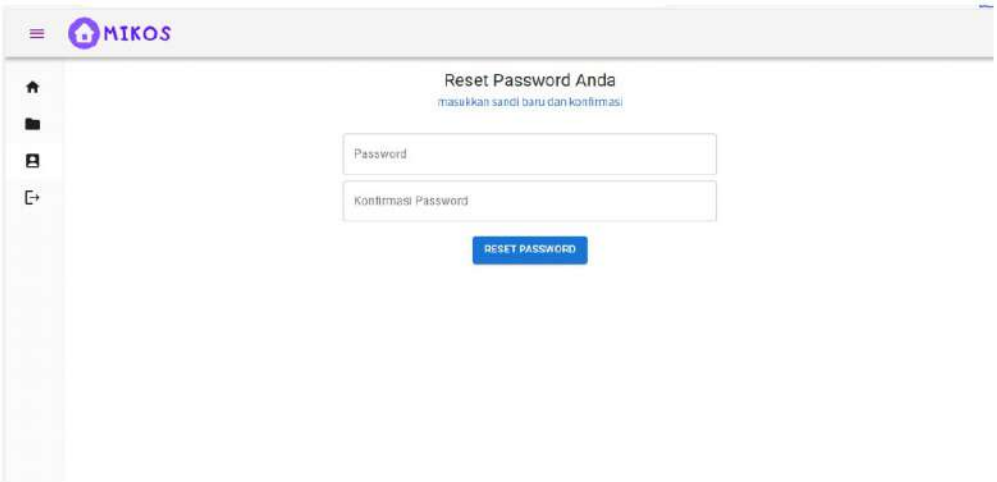
   Halaman *login* dapat diakses oleh pengunjung yang telah membuat akun *customer* atau pemilik kos.



Gambar 4.29 *Login*

3. *Forget Password*

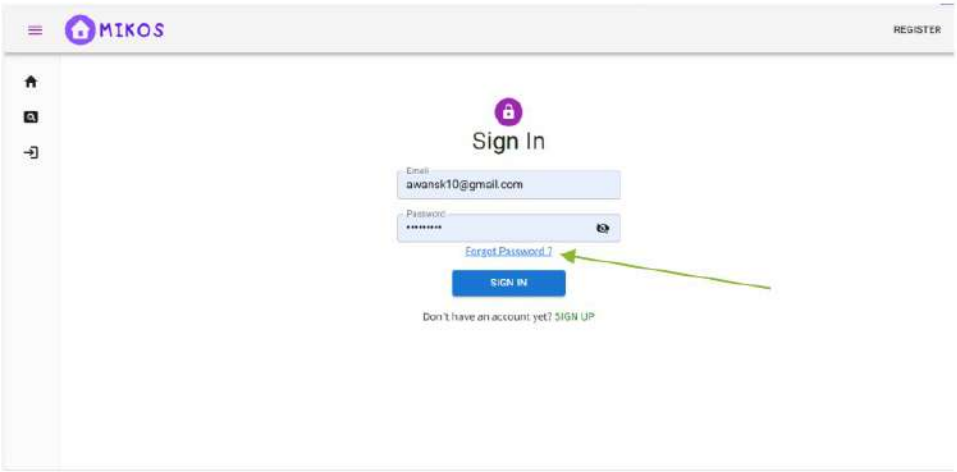
Halaman ini dapat diakses oleh *customer* dan pemilik kos yang ingin mengubah *password* akun. Halaman ini terdapat dua *form input* yang dapat diisi dengan *password* baru yang ingin diubah.



Gambar 4.30 Ubah *Password Account*

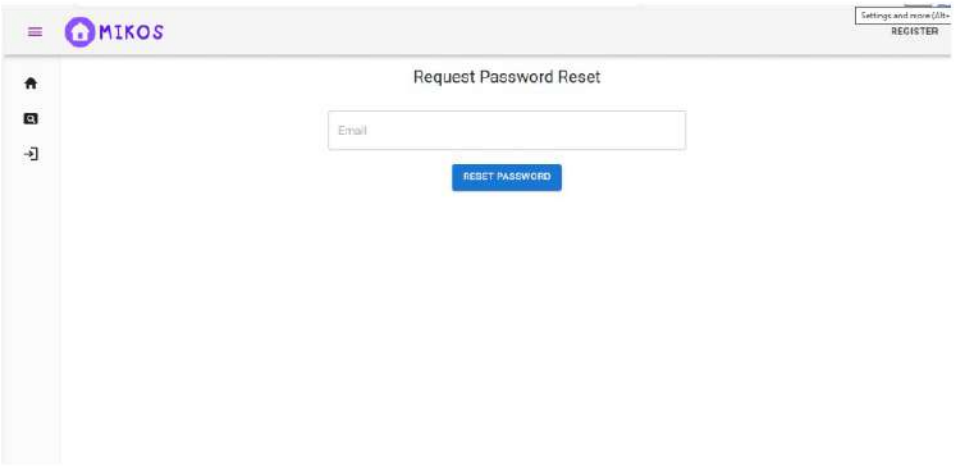
Ubah *password* dapat juga dilakukan, jika *customer* dan pemilik kos melupakan *password* akun yang terdaftar. berikut yang dapat dilakukan untuk mengubah *password* bagi pemilik akun yang melupakan passwordnya.

a. Pemilik akun dapat memilih *menu forgot password* pada halaman *login.*



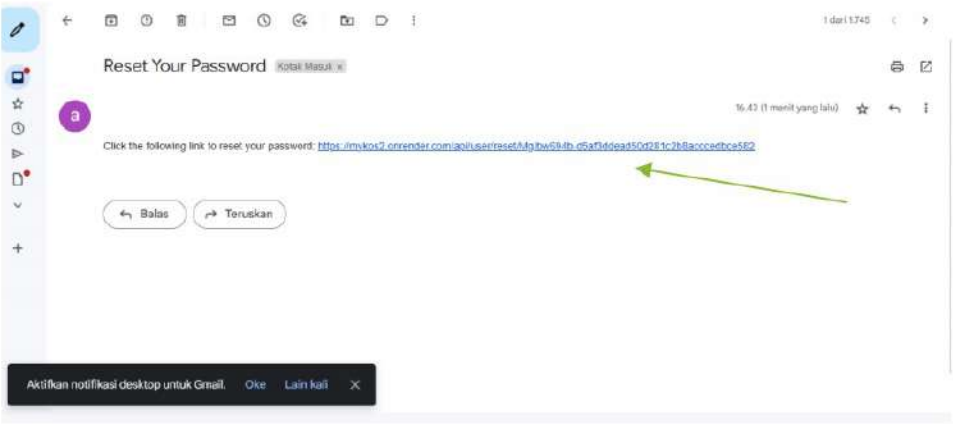
Gambar 4.31 *Menu Forgot Password*

b. Pemilik akun dapat mengisi *email* terdaftar yang akan diubah passwordnya.



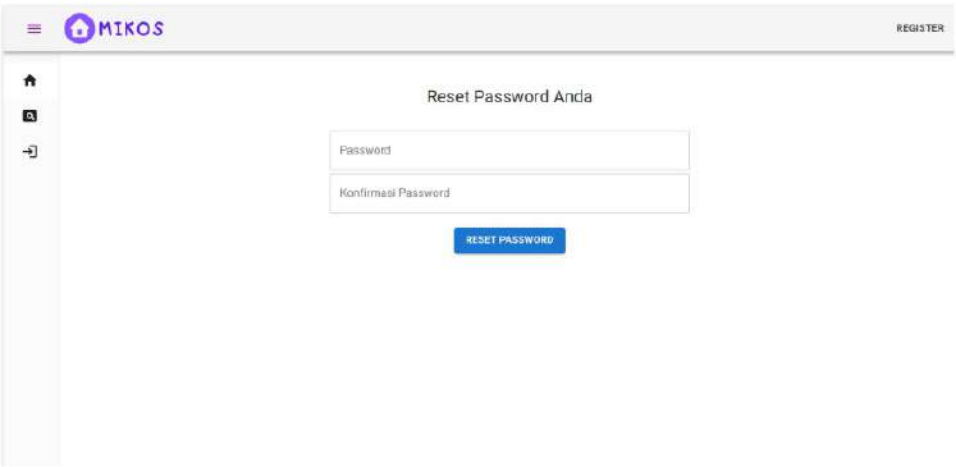
Gambar 4.32 *Request Password Reset*

c. Pemilik akun dapat membuka notifikasi *email* dan mengklik link untuk mereset *password* akun.



Gambar 4.33 Notifikasi *Email* ubah *Password*

d. Pemilik akun dapat mengisi dua *form input* yang dapat diisi dengan *password* baru yang ingin diubah.

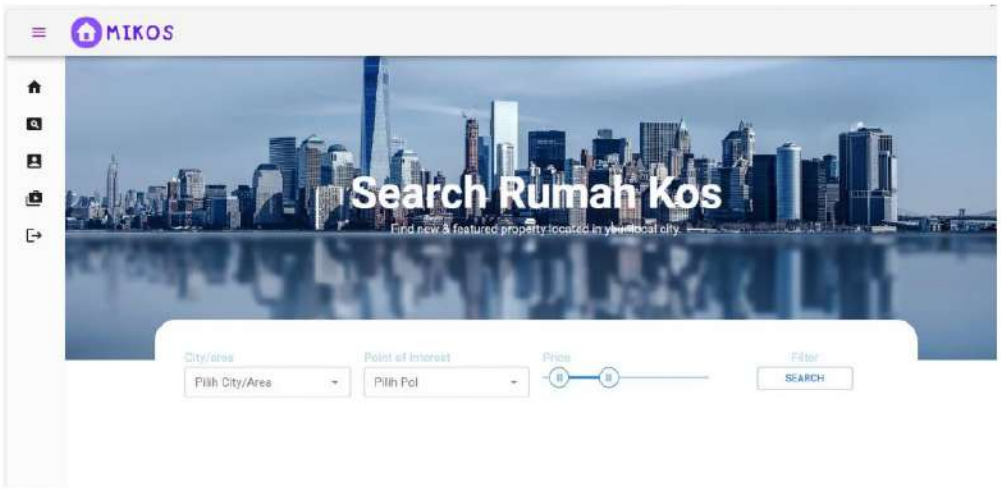


Gambar 4.34 Ubah *Password Account*

4.3.2 Pengunjung dan *Customer*

1. *Dashboard*

   Halaman *dashboard* untuk pengunjung dan *customer* saat *website* diakses.

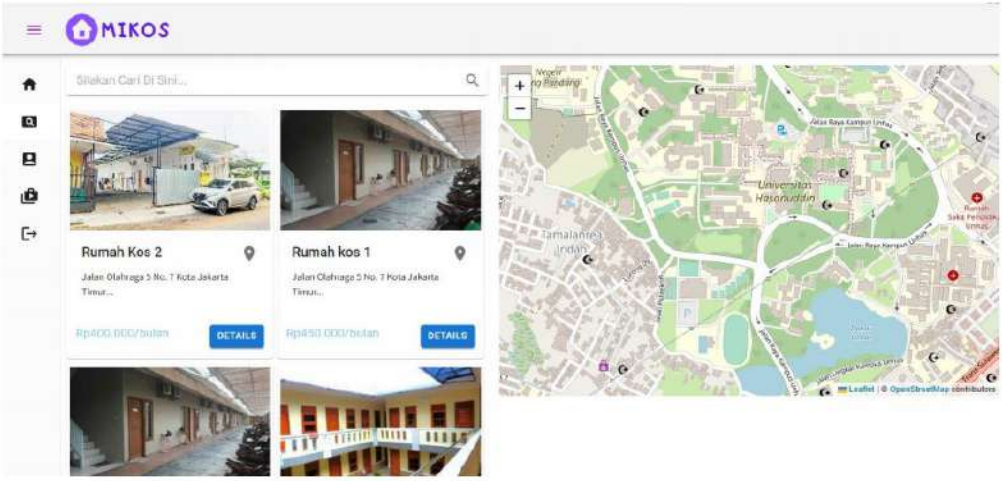


Gambar 4.35 *Dashboard*

Halaman ini menampilkan tampilan antarmuka *dashboard* yang akan tampil saat pertama kali *website* diakses. Halaman ini juga terdapat antarmuka pencarian yang dapat digunakan pengunjung untuk pencarian rumah kos yang sesuai.

2. Pencarian Rumah Kos

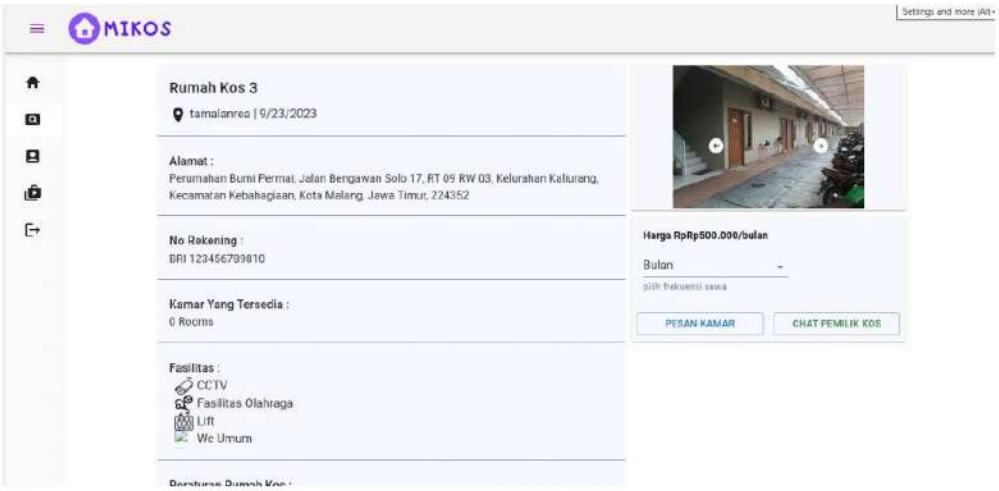
   Halaman pencarian rumah kos dapat diakses oleh pengunjung dan *customer*. Halaman ini menampilkan antarmuka pencarian rumah kos, daftar rumah kos dan peta yang menunjukkan lokasi rumah kos. Halaman ini memungkinkan pengunjung melakukan pencarian rumah kos dengan memasukkan *input* yang sesuai dengan kebutuhan.



Gambar 4.36 Pencarian Rumah Kos

3. Informasi Rumah Kos

Halaman detail rumah kos dapat diakses oleh pengunjung dan *customer*.



Gambar 4.37 Halaman Detail Rumah Kos

Halaman ini menampilkan informasi dari rumah kos yang menggambarkan kondisi dari rumah kos. Halaman ini terdapat detail informasi yang dapat ditampilkan, diantaranya sebagai berikut:

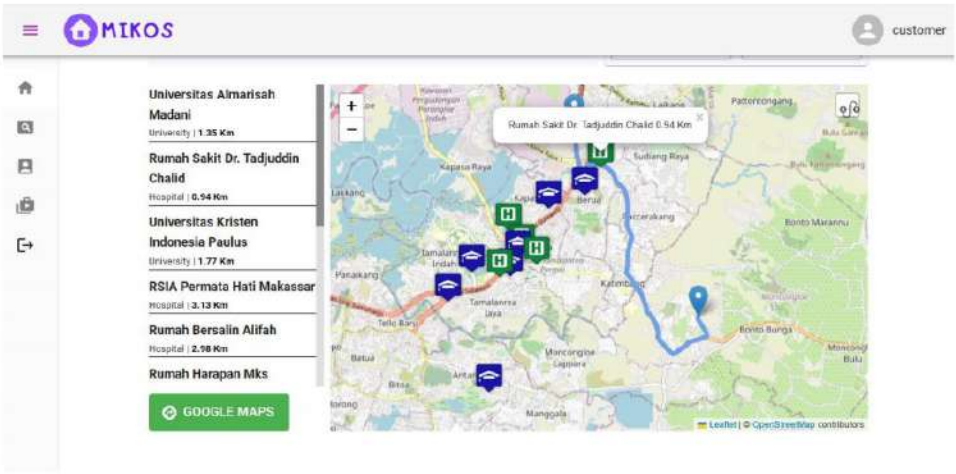
a. Nama rumah kos, informasi yang menampilkan nama dari rumah kos.
b. Harga, informasi yang menampilkan harga dari rumah kos.
c. Alamat, informasi yang menampil lokasi dari rumah kos yang terdaftar.
d. *Area*, informasi yang menampilkan *area* letak rumah kos yang terdaftar.
e. No. rekening, informasi yang menampilkan no. rekening yang digunakan dalam transaksi antara *customer* dan pemilik kos dalam proses penyewaan.

f. Fasilitas, informasi yang menampilkan beberapa fasilitas yang dimiliki rumah kos.
g. Peraturan, informasi yang menampilkan beberapa aturan yang dimiliki oleh rumah kos.
h. Kamar yang tersedia, informasi yang menampilkan jumlah kamar kosong dapat di sewa.
i. *Description*, informasi yang menampilkan deskripsi dari dari rumah kos yang terdaftar.
j. *Picture,* informasi tentang rumah kos dalam bentuk gambar.

Halaman ini juga terdapat beberapa fitur yang dapat digunakan pengunjung dam *customer*, diantaranya sebagai berikut:

a. Rute menuju rumah kos

Tampilan rute menuju rumah kos berada pada halaman detail rumah kos yang dapat diakses. Tampilan ini dapat diakses oleh pengunjung dan *customer.*

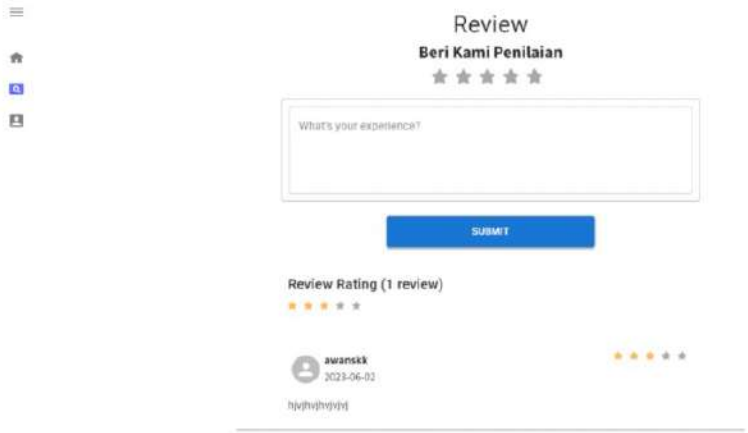


Gambar 4.38 Tampilan Rute Menuju Rumah Kos

Tampilan ini menampilkan informasi lokasi dari rumah kos yang terdaftar dan informasi rute menuju rumah kos. Tampilan ini juga menampilkan lokasi strategis terdekat dengan rumah kos seperti rumah sakit, universitas tempat ibadah dll.

b. Review

Tampilan *reviews* berada pada halaman detail rumah kos yang dapat diakses. Tampilan ini hanya dapat diakses oleh *customer*.



Gambar 4.39 Tampilan *Review*

Tampilan ini menampilkan fitur yang dapat digunakan *customer* untuk memberikan beberapa *review* tentang rumah kos, dan dengan adanya *review* ini dapat membantu *customer* baru dalam menilai keadaan rumah kos.

c. Pesan kamar

Menu pesan kamar berada pada halaman detail rumah kos, dan hanya dapat diakses oleh *customer*. *Menu* ini dapat digunakan *customer* untuk mendapatkan akses menuju halaman pesan kamar yang dapat dilihat pada Gambar 4.40.

d. Chat via whatsapp

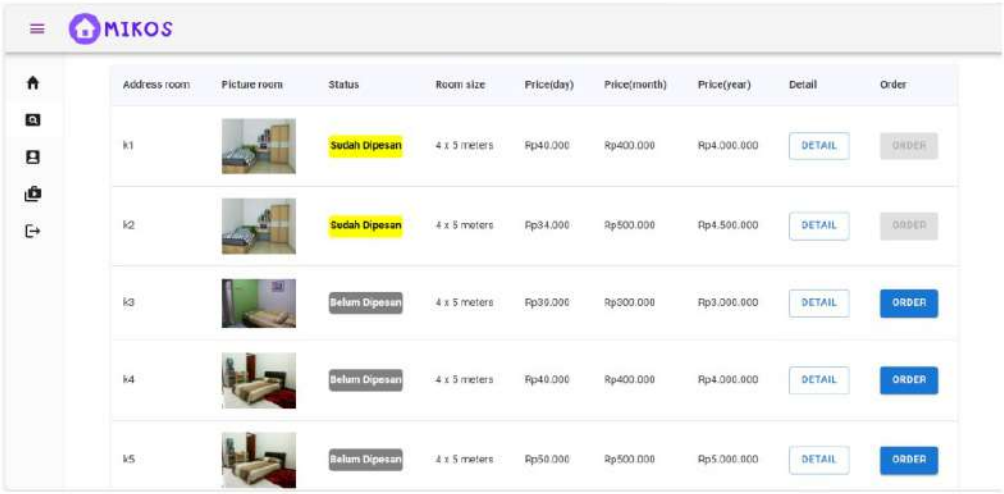
Menu chat via whatsapp berada pada halaman detail rumah kos dan hanya dapat diakses oleh *customer*. *Menu* ini dapat digunakan *customer* untuk dapat menghubungi pemilik kos *via whatsapp* dalam proses penyewaan kamar kos dll.

e. Google maps

Menu pintasan *google maps* pada halaman detail rumah kos yang dapat diakses pengunjung dan *customer*. *Menu* ini dapat digunakan *customer* untuk mendapatkan lokasi di *google maps* jika peta sistem bermasalah.

4. Pesan Kamar Kos

Halaman pesan kamar hanya dapat diakses oleh *customer*.



Gambar 4.40 Halaman Pesan Kamar

Halaman ini menampilkan daftar kamar kos beserta informasi yang mendukung. Halaman ini terdapat *menu order* yang dapat digunakan jika *customer* mau melakukan transaksi. Halaman ini terdapat detail informasi yang dapat ditampilkan, diantaranya sebagai berikut:

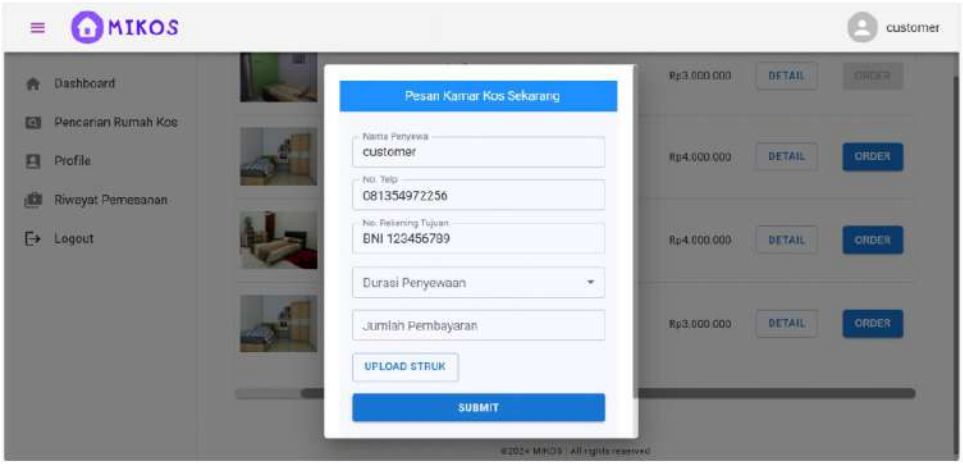
a. *Address room*, informasi yang menampilkan alamat dari kamar kos yang ada di rumah kos
b. *Picture room*, informasi yang menampilkan gambar dari kamar kos.
c. Status, informasi yang menampilkan status dari kamar kos. Jika kamar kos berstatus belum dipesan, maka kamar kos tersebut dapat dipesan dan jika kamar kos berstatus sudah dipesan, maka kamar kos tersebut tidak dapat dipesan.
d. *Room size*, informasi yang menampilkan ukuran dari kamar kos.
e. *Price*, informasi yang menampilkan harga yang dimiliki oleh kamar kos.
f. Detail, *menu* yang dapat memungkinkan pengunjung dapat melihat informasi detail kamar kos. *Menu* ini dapat terhubung dengan halaman detail kamar kos pada Gambar 4.41. halaman detail kamar kos dapat diakses oleh *customer*.

Gambar 4.41 Detail Kamar Kos

Gambar 4.41. menampilkan informasi dari kamar kos yang menggambarkan kondisi dari kamar kos. Halaman ini terdapat detail informasi yang dapat ditampilkan, diantaranya sebagai berikut:

- Alamat kamar, informasi yang menampilkan Alamat dari kamar kos
- Ukuran kamar, informasi yang menampilkan ukuran dari kamar kos.
- Fasilitas kamar, informasi yang menampilkan fasilitas yang dimiliki kamar kos.
- Aturan Kamar, informasi yang menampilkan aturan yang dimiliki kamar kos.

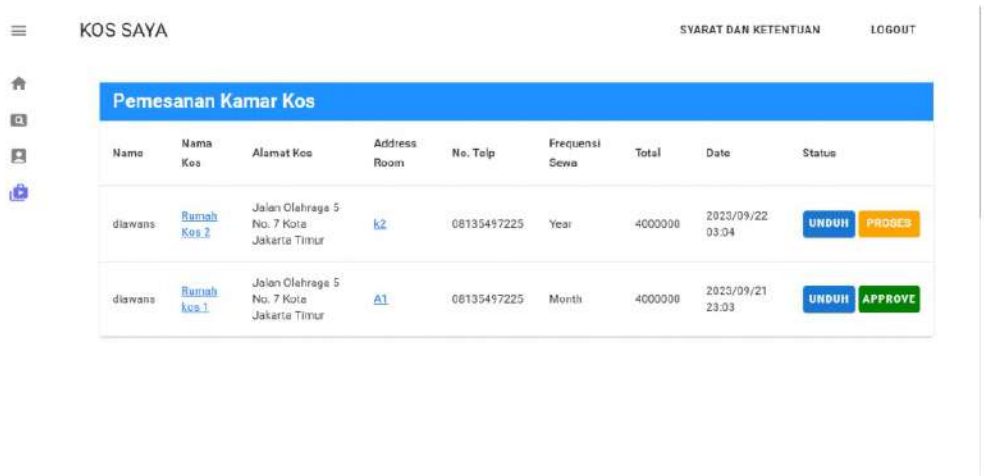
g. *Order*, *menu* yang dapat memungkinkan *customer* dapat memesan kamar kos yang dimiliki oleh rumah kos. *Menu* dapat terhubung dengan halaman *form order* yang dapat terlihat pada Gambar 4.42.



Gambar 4.42 *Form Order* Kamar Kos

5. Riwayat Transaksi Kamar Kos

Halaman riwayat transaksi kamar kos dapat diakses oleh *customer* yang telah melakukan transaksi.



Gambar 4.43 Halaman Riwayat Pemesanan Kamar Kos

Halaman ini menampilkan calon penghuni kos yang masih dalam tahap transaksi atau sudah melakukan transaksi kamar kos. Halaman ini terdapat kolom informasi yang dapat diisi, diantaranya sebagai berikut:

a. Nama, menampilkan nama informasi dari *customer* calon pemesan kamar kos.
b. Nama kos, menampilkan informasi nama dari rumah kos.
c. Alamat kos, menampilkan informasi dari Alamat rumah kos yang dipesan.
d. *Address room*, menampilkan informasi dari Alamat kamar yang dimiliki rumah kos.
e. No.telp, menampilkan informasi no telepon dari *customer*.

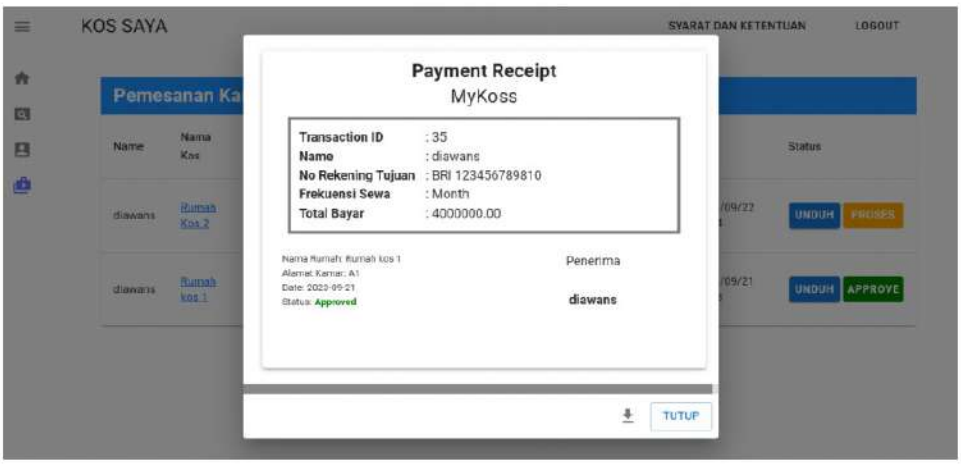
f. Frekuensi sewa, menampilkan informasi berapa lama dari waktu pesan kamar kos oleh *customer*.

g. Total, menampilkan informasi dari total pembayaran transaksi sewa kamar kos.

h. *Date,* menampilkan informasi waktu dari transaksi sewa kamar kos.

i. Status, menampilkan status dari proses transaksi sewa kamar kos, jika terdapat pesan 'proses' maka transaksi kamar kos belum disetujui oleh pemilik kos dan jika terdapat pesan '*approve*' maka proses transaksi telah disetujui oleh pemilik kos.

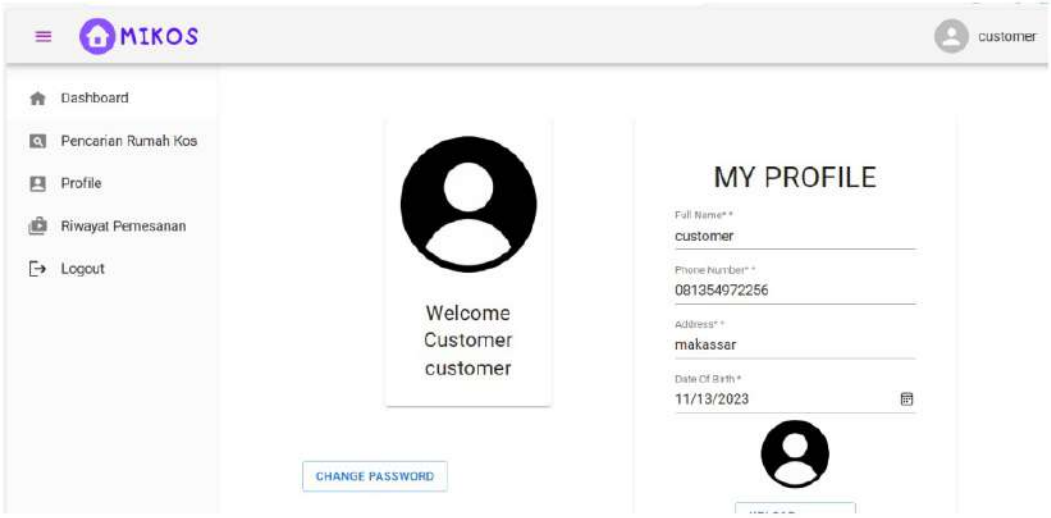
j. Unduh, menampilkan *menu* yang memungkinkan *customer* untuk mengunduh bukti proses transaksi sewa kamar kos.



Gambar 4.44 Bukti Transaksi Sewa Kamar Kos

6. *Profile Customer*

Halaman profil dapat diakses oleh *customer* yang telah melakukan registrasi dan *login* terlebih dahulu. Halaman ini menampilkan *form* profile *customer* yang dapat diisi untuk memperbarui isi dari informasi *profile customer* yang sudah terdaftar sebelumnya.

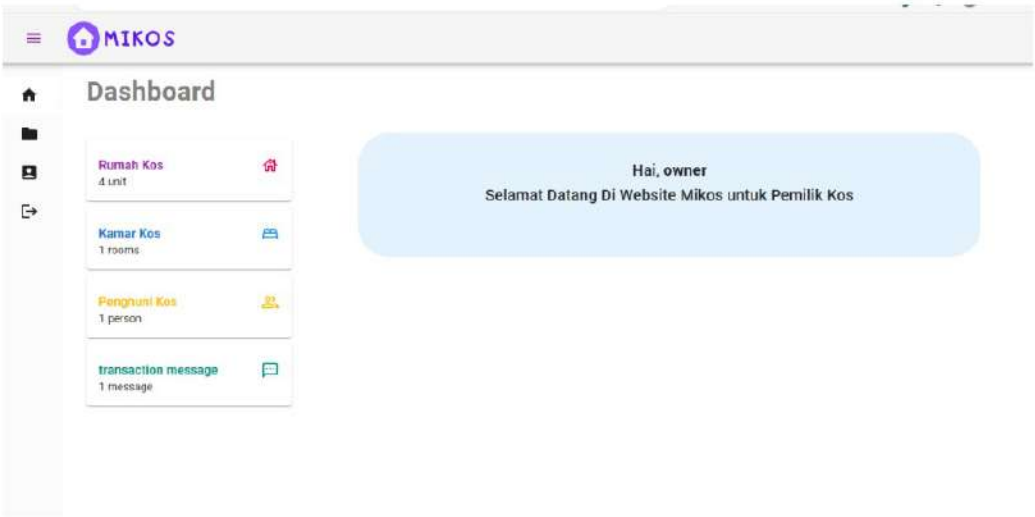


Gambar 4.45 *Profile Customer*

4.3.3 Pemilik Kos

1. Dashboard

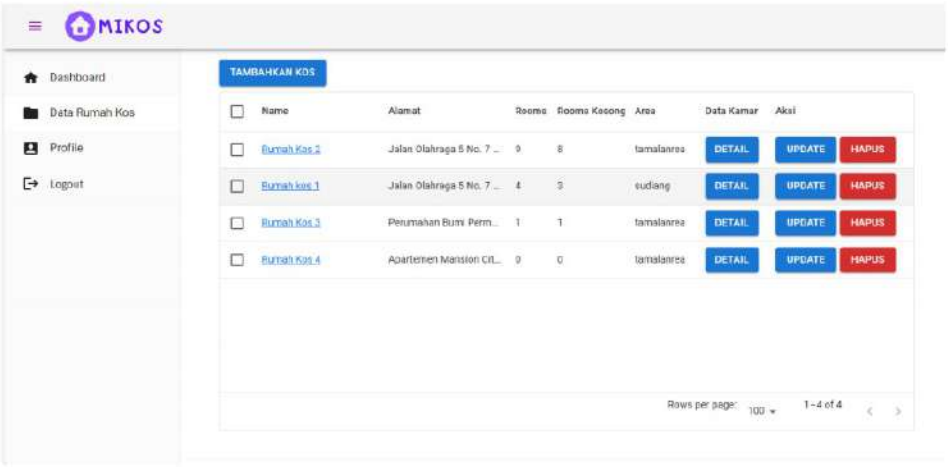
Halaman dashboard untuk pemilik kos. Halaman ini menampilkan halaman utama yang dimiliki pemilik kos setelah *login*.



Gambar 4.46 *Dashboard* Pemilik Kos

2. Data Rumah Kos

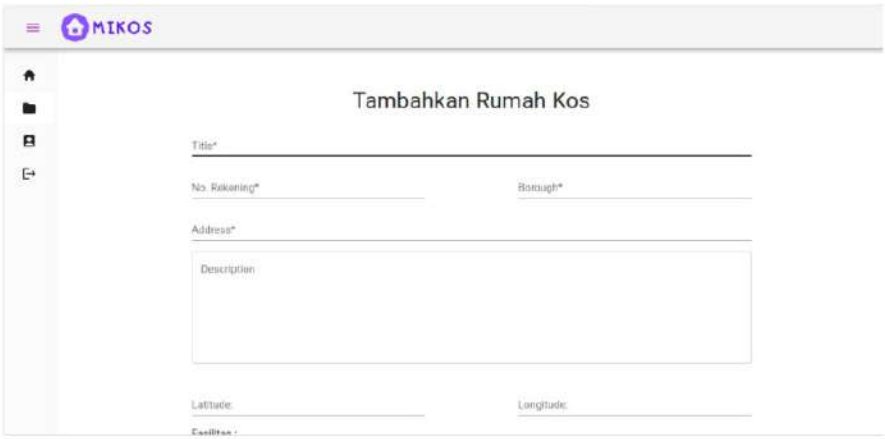
Halaman data rumah kos dapat diakses pemilik kos. Halaman ini menampilkan daftar rumah kos yang dimiliki pemilik kos.



Gambar 4.47 Data Rumah Kos

Halaman ini terdapat beberapa *menu* yang dapat digunakan pemilik kos dalam mengelola rumah kos. diantaranya sebagai berikut:

a. Tambahkan kos, *menu* ini dapat digunakan pemilik kos untuk menambahkan data rumah kos. *Menu* ini menampilkan *form* untuk menambahkan rumah kos yang dapat diakses oleh pemilik kos.



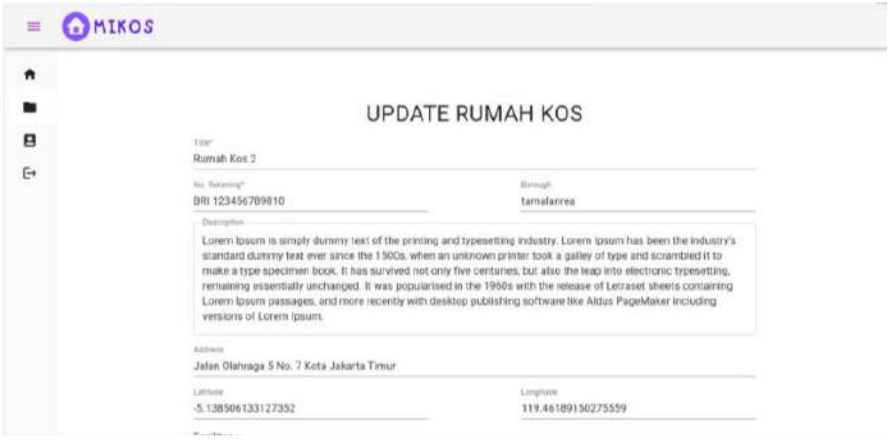
Gambar 4.48 *Form* Tambah Rumah Kos

Halaman ini menampilkan *form* tambahkan rumah kos yang dapat diisi untuk menambahkan rumah kos. Halaman ini terdapat kolom informasi yang dapat diisi, diantaranya sebagai berikut:

- Nama rumah kos, kolom untuk memasukkan nama dari rumah kos.

- Harga, kolom untuk memasukkan harga dari rumah kos, kolom harga memiliki tiga *type* yaitu harga per hari, harga per bulan dan harga per tahun.
- Alamat, kolom untuk memasukkan lokasi dari rumah kos yang terdaftar.
- *Borough*, kolom untuk memasukkan lokasi *area* dari rumah kos.
- No. rekening, kolom untuk memasukkan no rekening yang digunakan dalam transaksi antara *customer* dan pemilik kos dalam proses penyewaan.
- Fasilitas, kolom untuk memasukkan beberapa fasilitas yang dimiliki rumah kos.
- Aturan, kolom untuk memasukkan beberapa aturan yang dimiliki rumah kos.
- *Description*, kolom untuk memasukkan deskripsi/penjelasan singkat dari dari rumah kos yang terdaftar.
- *Latitude*, kolom untuk memasukkan *latitude* dari lokasi rumah kos.
- *Longitude*, kolom untuk memasukkan *longitude* dari lokasi rumah kos.
- *Picture*, *menu* yang digunakan untuk *upload* gambar dari rumah kos.

b. *Update*, *menu* ini dapat digunakan pemilik kos untuk memperbarui data rumah kos. *Menu* ini menampilkan *form update* rumah kos yang dapat diisi untuk memperbarui isi dari informasi rumah kos yang sudah terdaftar sebelumnya.



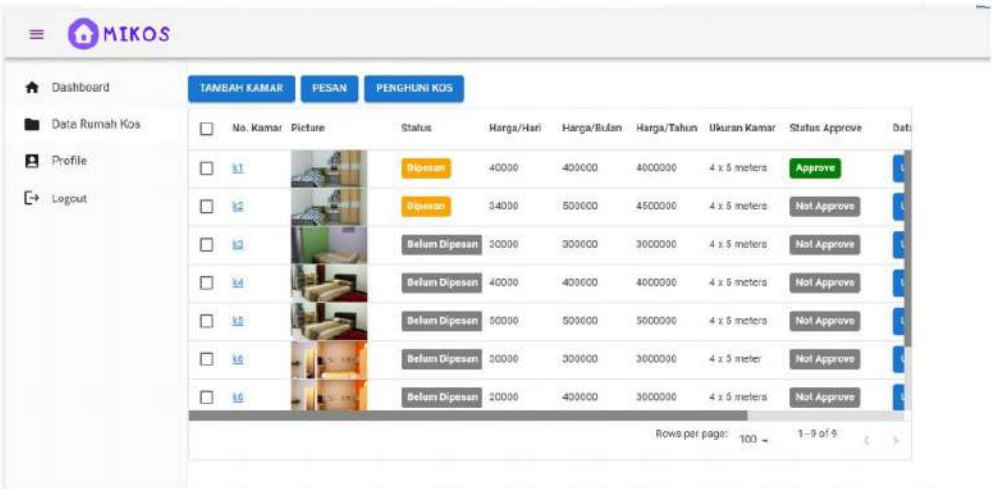
Gambar 4.49 *Form Update* Rumah Kos

c. Detail, *menu* ini dapat digunakan pemilik kos untuk melihat daftar kamar yang dimiliki rumah kos. *Menu* ini menampilkan halaman detail rumah kos untuk pemilik kos dapat diakses pemilik kos.

d. Hapus, *menu* ini dapat digunakan pemilik kos untuk menghapus data rumah kos.

3. Data Kamar Kos

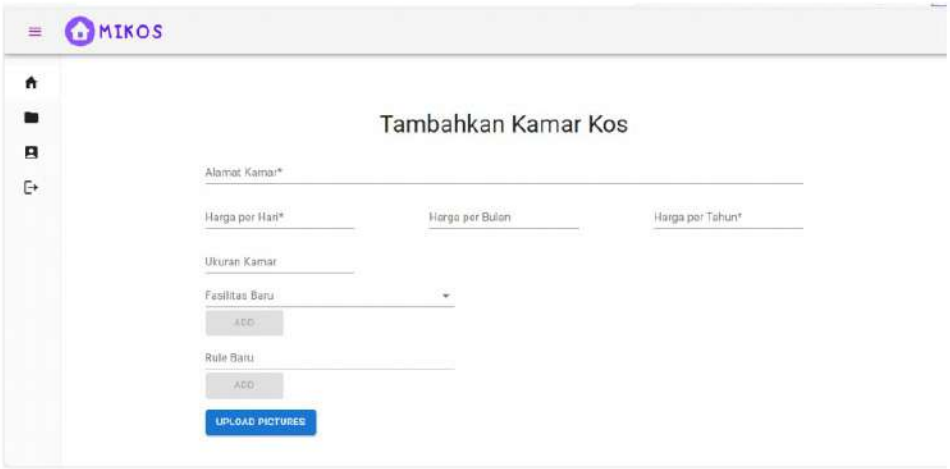
Halaman data kamar kos dapat diakses oleh pemilik kos. Halaman ini menampilkan daftar kamar kos yang ada dalam rumah kos, beberapa informasi kamar kos dan beberapa *menu* yang dapat digunakan pemilik kos dalam mengelola kamar dari rumah kos.



Gambar 4.50 Data Kamar Kos

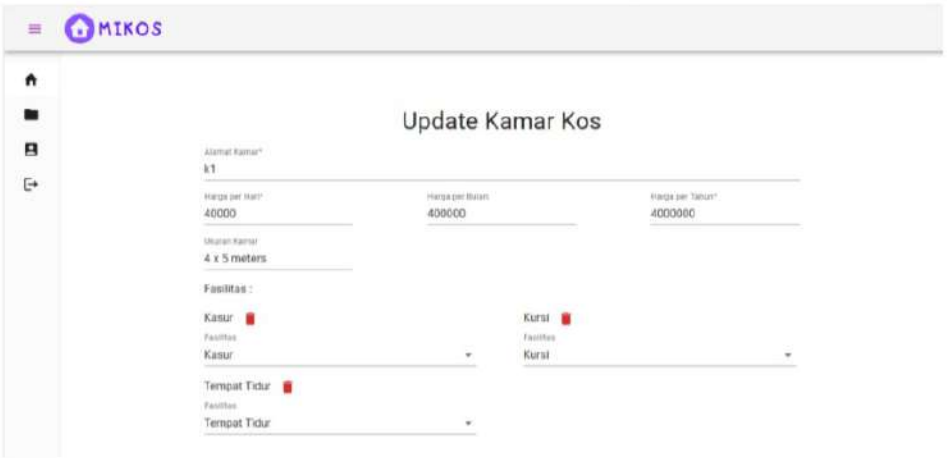
Halaman ini terdapat beberapa *menu* yang dapat digunakan pemilik kos dalam mengelola kamar kos. diantaranya sebagai berikut:

a. Tambahkan kamar, *menu* ini dapat digunakan pemilik kos untuk menambahkan data kamar kos. *Menu* ini menampilkan *form input* yang dapat diisi oleh pemilik kos dalam mengelola kamar dari rumah kos.



Gambar 4.51 *Form* Tambah Kamar Kos

b. *Update*, *menu* ini dapat digunakan pemilik kos untuk memperbarui data kamar kos. *Menu* ini menampilkan *form input* informasi kamar kos yang dapat diisi oleh pemilik kos dalam mengelolah kamar dari rumah kos.

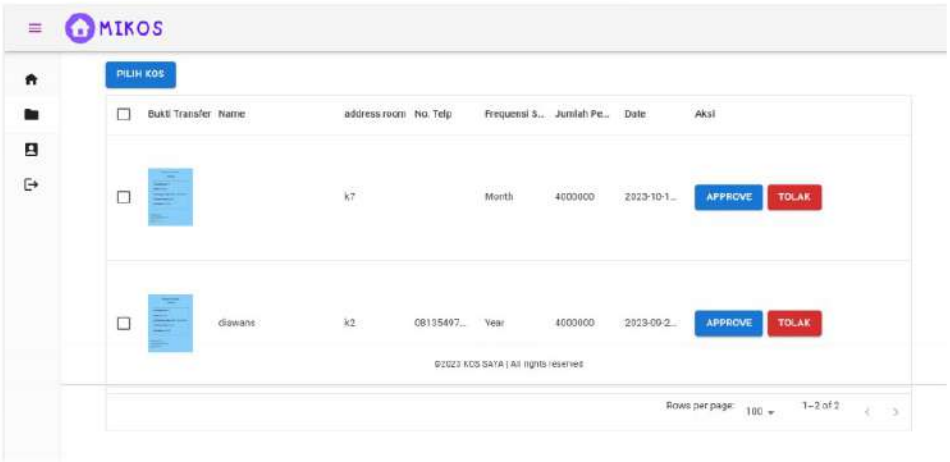


Gambar 4.52 *Form Update* Kamar Kos

c. Hapus, *menu* ini dapat digunakan pemilik kos untuk menghapus kamar kos.

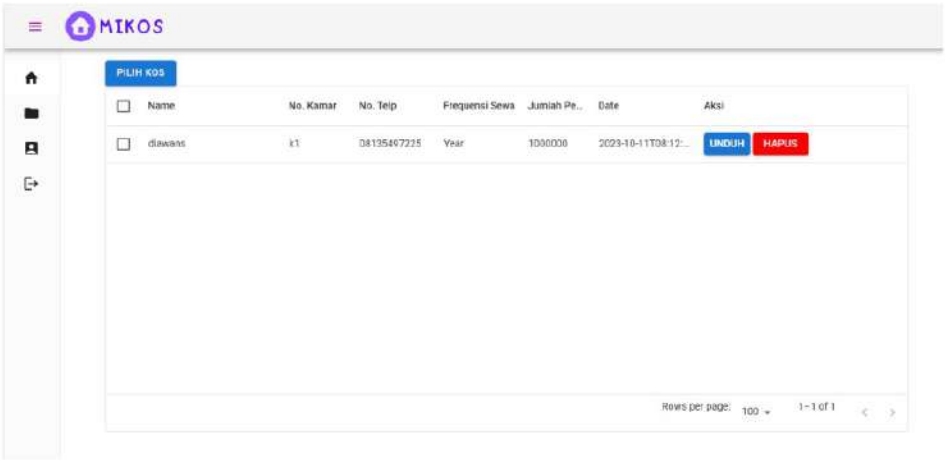
d. Pesan, *menu* ini dapat digunakan pemilik kos, untuk menampilkan pesan transaksi kamar kos yang dapat di *approve* atau ditolak. *Menu* ini, memungkinkan pemilik kos menyetujui transaksi yang dilakukan oleh *customer*, jika semua syarat dalam proses pemesanan telah sesuai dengan

persetujuan pemilik kos. Pemilik kos juga dapat menghapus pesan pemesanan tersebut jika syarat pemesanan yang dilakukan *customer* tidak sesuai.



Gambar 4.53 *Approve* Transaksi

e. Penghuni kos, *menu* ini dapat digunakan pemilik kos untuk menampilkan daftar penghuni rumah kos. *Menu* ini menampilkan penghuni kos yang telah disetujui oleh pemilik kos dengan syarat telah menyelesaikan pembayaran dll. Halaman ini juga terdapat *menu* unduh, yang dapat digunakan pemilik kos untuk melihat bukti transaksi *customer*.

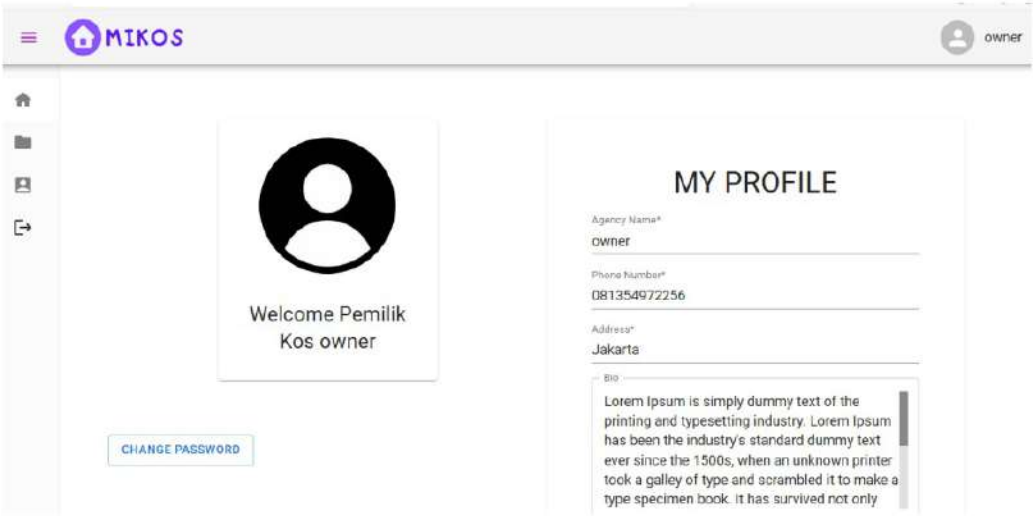


Gambar 4.54 Daftar Penghuni Kos

f. *Cancel*, *menu* ini dapat digunakan pemilik kos untuk menghapus transaksi yang terkait dengan kamar kos.

4. *Profile* Pemilik Kos

Halaman *profile* pemilik kos dapat diakses pemilik kos. Halaman ini menampilkan *form profile* pemilik kos yang dapat diisi untuk memperbarui isi dari informasi *profile* pemilik kos yang sudah terdaftar sebelumnya.

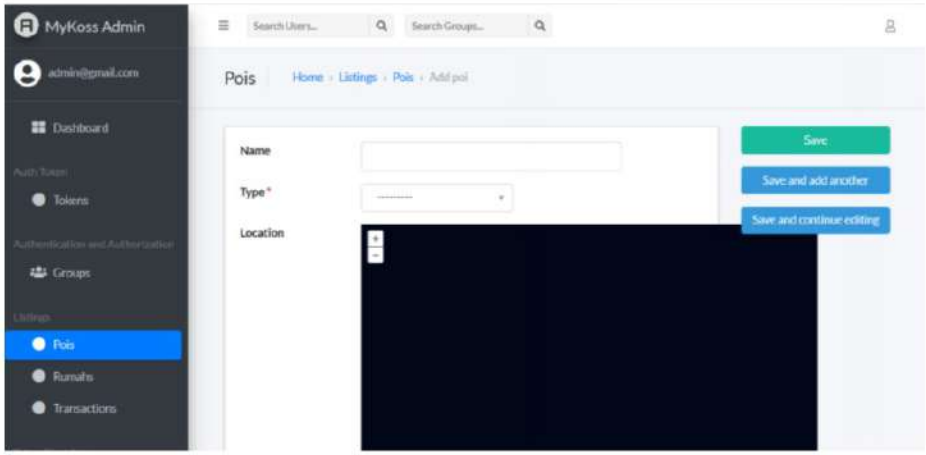


Gambar 4.55 *Profile* Untuk Pemilik Kos

4.3.4 Admin

1. Menambahkan *Point of Interest*

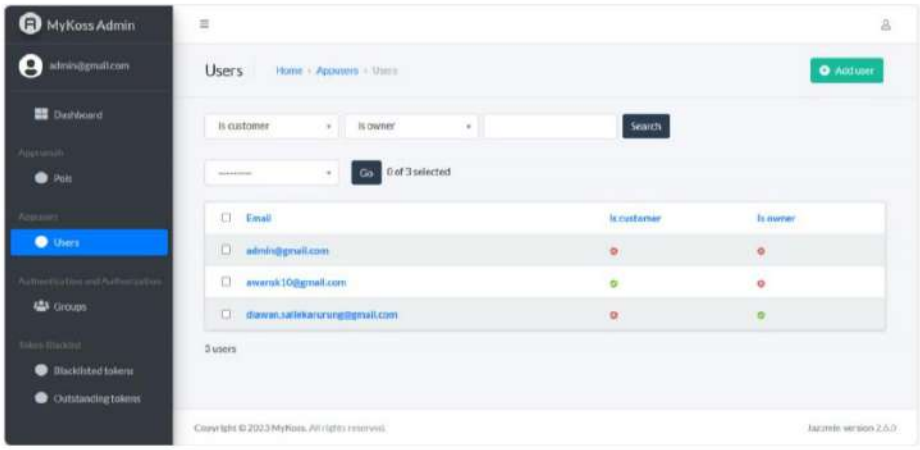
   Halaman ini dapat diakses oleh *admin* untuk menambahkan *point of interest* yang berdekatan dengan rumah kos. Penambahan *point of interest* bertujuan untuk memberikan informasi tambahan dalam pencarian rumah kos, yang akan memberikan informasi tentang lokasi-lokasi strategis yang berdekatan dengan rumah kos, seperti universitas, rumah sakit, dan tempat ibadah.



Gambar 4.56 *Form Input Point of Interest*

2. Mengelolah *User*

   Halaman ini dapat diakses oleh *admin* untuk mengelola *user* yang terdaftar dalam *website* pencarian rumah kos.



Gambar 4.57 Tampilan Daftar *User*

4.4 Pengujian Sistem

Pengujian dengan metode *black box testing* dilakukan dengan menjalankan sistem dan melihat output-nya yang memastikan bahwa system yang dikembangkan telah sesuai dengan rancangan system. adapun hasil pengujian sebagai berikut:

1. Pengujian *Menu Login*

   Pengujian *menu login* dilakukan dengan mengisi *form log in* yang disediakan system untuk customer atau pemilik kos ketika mengakses halaman *website*. Adapun hasil pengujian pada *menu login* dapat dilihat pada Tabel 4.2

Tabel 4.2 Hasil Pengujian *Menu Login*

Kasus Pengujian	**Hasil yang Diharapkan**	**Hasil Pengujian**	**Kesimpulan**
Mengisi *email* dan *password* yang benar	Sistem berhasil menemukan *email* dan *password,* dan selanjutnya *system* menampilkan halaman utama setelah *login*	Sesuai	Berhasil
Mengisi *email* dan *password* yang salah	Sistem tidak berhasil menemukan *email* dan *password,* lalu muncul pesan "Invalid *email* or *password*"	Sesuai	Berhasil
Mengosongkan *email* dan *password*	Sistem akan menampilkan pesan kesalahan "*Email is required!*" atau "*Password is required*"	Sesuai	Berhasil
Mengisi *email* dan *password* yang tidak sesuai aturan yang berlaku	Sistem akan menampilkan pesan kesalahan "*Email doesn't match the required pattern*" atau "*Password doesn't match the required pattern*"	Sesuai	Berhasil

2. Pengujian *Menu* Registrasi

Pengujian *menu* registrasi dilakukan dengan mengisi *form* registrasi yang disediakan system untuk pengguna yang akan membuat akun customer atau pemilik kos. Adapun hasil pengujian pada *menu* registrasi dapat dilihat pada Tabel 4.3

Tabel 4.3 Hasil Pengujian *Menu* Registrasi

Kasus Pengujian	**Hasil yang Diharapkan**	**Hasil Pengujian**	**Kesimpulan**
Mengisi semua *form* yang disediakan dengan benar	Sistem membuat akun *customer* atau pemilik kos, selanjutnya sistem secara otomatis *login* dan menampilkan halaman utama.	Sesuai	Berhasil
Menginput *email* yang tidak valid atau *password* terlalu pendek	Tidak dapat melanjutkan pendaftaran dan sistem menampilkan pesan kesalahan *"Email doesn't match the required pattern"* atau *"Password doesn't match the required pattern"*	Sesuai	Berhasil
Mengosongkan *form* lalu klik sign up	Tidak dapat melanjutkan pendaftaran dan sistem akan menampilkan pesan "(nama kolom) is required"	Sesuai	Berhasil
Mengisi *email* atau *password* yang sudah terdaftar	Tidak dapat melanjutkan pendaftaran dan *System* akan menampilkan pesan kesalahan "*user* with this *email* already exists."	Sesuai	Berhasil

Mengosongkan kotak penyetujuan *term and condition* yang berlaku	Tidak dapat melanjutkan pendaftaran dan *system* akan menampilkan pesan kesalahan *"You must agree to the terms and conditions to proceed with registration."*	Sesuai	Berhasil

3. Pengujian *menu forget password*

Pengujian *menu forget password* dilakukan dengan mengisi form ubah password yang disediakan system untuk pengguna yang ingin mengubah password ataupun lupa password. Adapun hasil pengujian pada *menu forget password* dapat dilihat pada Tabel 4.4

Tabel 4.4 Hasil Pengujian *Menu Forget Password*

Kasus Pengujian	**Hasil yang Diharapkan**	**Hasil Pengujian**	**Kesimpulan**
Pengguna mengakses *menu Forgot Password?* Dalam *menu login*	*System* akan menampilkan halaman request *password* reset	Sesuai	Berhasil
Mengisi *email* sesuai akun terdaftar untuk konfirmasi reset sandi	*System* akan mengirim pesan konfirmasi reset sandi ke *email* yang dimasukkan. Selanjutnya pengguna dapat mengakses link untuk mereset sandi akun	Sesuai	Berhasil
Mengisi *password* baru dan *confirm password* dengan benar	*System* akan menampilkan pesan *"Password reset successful"*	Sesuai	Berhasil

Mengosongkan kolom *password* atau *confirm password*	*System* akan menampilkan pesan "*(nama kolom) is required*"	Sesuai	Berhasil
Mengisi *password* dan *confirm password* tidak sesuai atau *password* tidak sama dengan *confirm password*	*System* akan menampilkan pesan "*Password and Confirm Password doesn't match*"	Sesuai	Berhasil
Pengguna dapat mengakses *change password* Dalam *menu profile* saat pengguna ingin mengubah kata sandi	*System* akan menampilkan halaman *change password*	Sesuai	Berhasil

4. Pengujian Pencarian Rumah Kos

Pengujian pencarian rumah kos dilakukan dengan pengunjung dan customer mengakses beberapa fitur dan melakukan pencarian rumah kos. Adapun hasil pengujian dapat dilihat pada Tabel 4.5

Tabel 4.5 Hasil Pengujian Pencarian Rumah Kos

Kasus Pengujian	Hasil yang Diharapkan	Hasil Pengujian	Kesimpulan
Mengakses *menu dashboard* pada *sidebar*	*System* akan menampilkan halaman *dashboard* dan antarmuka kolom pencarian rumah kos	Sesuai	Berhasil
Melakukan pencarian rumah kos pada *dashboard* berdasarkan *area*, *point of interest* dan harga pada halaman *dashboard.*	*System* akan menampilkan hasil dari pencarian rumah kos berdasarkan *area*, *point of interest* dan harga dari rumah kos.	Sesuai	Berhasil
Mengakses *menu* pencarian rumah pada *sidebar.*	*System* akan menampilkan halaman pencarian rumah kos beserta peta dari rumah kos	Sesuai	Berhasil
Mengakses kolom pencarian pada halaman pencarian rumah dan melihat letak rumah kos.	*System* akan menampilkan hasil dari pencarian rumah kos dan melihat letak dari rumah kos	Sesuai	Berhasil

5. Pengujian Transaksi kamar kos

   Pengujian transaksi kamar kos dilakukan dengan *customer* mengakses beberapa fitur yang memungkinkan *customer* melakukan transaksi untuk memesan kamar kos. Adapun hasil pengujian transaksi kamar kos dapat dilihat pada Tabel 4.6

Tabel 4.6 Hasil Pengujian Transaksi Rumah Kos

Kasus Pengujian	**Hasil yang Diharapkan**	**Hasil Pengujian**	**Kesimpulan**
Mengakses Halaman Pesan Kamar lewat *menu* pesan kamar pada halaman detail rumah kos	*System* akan menampilkan halaman pesan kamar yang berisikan daftar kamar yang dimiliki rumah kos.	Sesuai	Berhasil
Mengakses halaman detail kamar lewat *menu* detail dihalaman pesan kamar.	*System* akan menampilkan halaman detail kamar kos yang berisikan detail dari kamar kos.	Sesuai	Berhasil
Mengakses *menu order* untuk memulai proses transaksi.	*System* akan menampilkan halaman *order*, selanjutnya *customer* dapat mengisi *form* inputan yang sesuai dengan informasi yang diminta.	Sesuai	Berhasil
Menghubungi pemilik kos dengan mengakses *menu chat via whatsapp*.	*System* akan menghubungkan ke aplikasi *whatsapp customer* dan pemilik kos sesuai dengan nomor yang terdaftar.	Sesuai	Berhasil

Mengakses *menu* Riwayat pemesanan pada *sidebar.*	*System* akan menampilkan halaman Riwayat pemesanan, dalam halaman ini menampilkan sebuah proses interaksi dari transaksi rumah kos antara *customer* dengan pemilik kos.	Sesuai	Berhasil

6. Pengujian Pengelolaan Rumah Kos

Pengujian pengelolaan rumah kos dilakukan dengan pemilik kos mengakses beberapa fitur yang memungkinkan pemilik kos dapat mengelola rumah kos. Adapun hasil pengujian pada *menu* pemilik kos dapat dilihat pada Tabel 4.7.

Tabel 4.7 Hasil Pengujian *Menu* Pengelolaan Rumah Kos

Kasus Pengujian	**Hasil yang Diharapkan**	**Hasil Pengujian**	**Kesimpulan**
Mengakses *menu dashboard* pada *sidebar*	*System* menampilkan halaman *dashboard* dan beberapa informasi mengenai rumah kos	Sesuai	Berhasil
Mengakses *menu* data rumah kos	*System* menampilkan halaman data rumah kos serta informasi rumah kos	Sesuai	Berhasil
Mengakses *menu* tambahkan kos pada halaman data rumah kos	*System* akan menampilkan *form* data untuk menambahkan rumah kos. Jika selesai, selanjutnya *system* akan menampilkan halaman data rumah kos	Sesuai	Berhasil
Menekan link nama dari rumah kos	*System* akan menampilkan detail informasi rumah kos untuk pemilik kos	Sesuai	Berhasil

Menekan *menu* update pada halaman data rumah kos	*System* akan menampilkan *form* data dari rumah kos. Selanjutnya pemilik kos dapat mengubah data sesuai kebutuhan dalam *form* tersebut	Sesuai	Berhasil
Menekan *menu* hapus pada halaman data rumah kos	*System* akan menghapus data rumah kos, selanjutnya *system* akan menampilkan halaman data rumah kos	Sesuai	Berhasil
Menekan menu detail pada halaman data rumah kos.	System akan menampilkan halaman data kamar kos serta informasi kamar kos.	Sesuai	Berhasil
Mengakses *menu* tambahkan kamar pada halaman data kamar kos.	*System* akan menampilkan *form* data untuk menambahkan kamar kos. Jika selesai, selanjutnya *system* akan menampilkan halaman data kamar kos.	Sesuai	Berhasil
Menekan *menu update* pada *list* kamar kos	*System* akan menampilkan *form* data dari kamar kos. Selanjutnya pemilik kos dapat mengubah data sesuai kebutuhan dalam *form* tersebut	Sesuai	Berhasil
Menekan *menu* hapus pada *list* kamar kos	*System* akan menghapus data rumah kos, selanjutnya *system* akan menampilkan halaman data kamar kos	Sesuai	Berhasil

Menekan *menu cancel* pada *list* data kamar kos	*System* akan menghapus data transaksi yang terkait dengan kamar kos, selanjutnya *system* akan menampilkan halaman data kamar kos	Sesuai	Berhasil
Mengakses *menu* penghuni kos pada halaman data kamar kos	*System* akan menampilkan halaman penghuni kos yang suda melakukan transaksi dan disetujui oleh pemilik kos	Sesuai	Berhasil
Mengakses *menu* pesan pada halaman data kamar kos	*System* akan menampilkan halaman pesan kamar kos. Halaman memungkinkan pemilik kos menyetujui proses transaksi jika semua syarat terpenuhi	Sesuai	Berhasil

7. Pengujian *Admin*

Pengujian menu *admin* dilakukan dengan mengakses beberapa fitur yang memungkinkan *admin* dapat mengelola *user* dalam *website*. Adapun hasil pengujian pada *menu* pemilik kos dapat dilihat pada Tabel 4.8.

Tabel 4.8 Hasil Pengujian *Menu Admin*

Kasus Pengujian	**Hasil yang Diharapkan**	**Hasil Pengujian**	**Kesimpulan**
Mengakses *menu user* pada *admin page*	Menampilkan halaman dari *menu user* pada *admin page*	Sesuai	Berhasil
Menambahkan *user* lewat *menu add* pada halaman *user*	Menampilkan *form input* untuk menambahkan *user*	Sesuai	Berhasil

Mengakses *menu update* untuk memperbarui data *user*	Menampilkan *form input user* yang dapat digunakan untuk memperbarui data *user*	Sesuai	Berhasil
Mengakses menu hapus untuk menghapus *user*	System akan menghapus data *user*	Sesuai	Berhasil
Mengakses *menu* pois pada *admin page*	Menampilkan halaman dari *menu* poi pada admin page	Sesuai	Berhasil
Menambahkan *point of interest* lewat menu add pada halaman *user*	Menampilkan *form input* untuk menambahkan *point of interest*	Sesuai	Berhasil
Mengakses *menu update* untuk memperbarui data *point of interest*	Menampilkan *form input user* yang dapat digunakan untuk memperbarui data *point of interest*	Sesuai	Berhasil
Mengakses *menu* hapus untuk menghapus *point of interest*	System akan menghapus data *point of interest*	Sesuai	Berhasil

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan hasil dan pembahasan dari penelitian yang dilakukan dapat disimpulkan:

1. Analisis kebutuhan sistem dengan judul sistem informasi geografis untuk pencarian rumah kos berbasis *web* berupa kebutuhan perangkat keras, kebutuhan perangkat lunak serta kebutuhan sistem pada aplikasi *website* dan rancangan sistem terdiri dari *use case diagram*, *activity diagram*, *entity relationship diagram*, dan rancangan *interface*.
2. Rancangan sistem informasi geografis untuk pencarian rumah kos berbasis *web* diimplementasikan ke dalam kode dengan penggunakan *framework django* sebagai *backend* dan *react js* sebagai *frontend*. Implementasi pada aplikasi terdiri dari beberapa fitur yaitu fitur untuk pengunjung dapat mencari rumah kos, melihat informasi rumah kos dan melihat rute rumah kos. *Customer* dapat melakukan semua yang dapat dilakukan pengunjung dan melakukan transaksi kamar kos dll. dan pemilik kos dapat menambahkan rumah kos dan mengelola rumah kos dan dll. serta *admin* dapat mengelola *user* dalam *website*.
3. Hasil pengujian efektivitas pada aplikasi *website* dilakukan dengan menggunakan metode *black box testing*. Pengujian ini mencoba seluruh fitur yang ada pada *website*. Hasil pengujian fitur pada *website* tersebut berjalan sesuai dengan fungsinya.

5.2 Saran

Dalam melakukan penelitian ini, penulis mengusulkan beberapa saran untuk penelitian selanjutnya guna dalam pengembangan sistem lebih lanjut sebagai berikut:

1. Diharapkan penelitian selanjutnya dapat mengembangkan *website* ini dalam bentuk aplikasi *mobile*.
2. Diharapkan *system* pencarian dalam penelitian ini dikembangkan lebih efisien dan lebih memudahkan penggunaan dalam mencari rumah kos.

3. Pada penelitian ini *script* yang digunakan belum cukup efisien, sedangkan dalam pengembangan sebuah aplikasi *website script* harus dibuat serapih mungkin guna meningkatkan efisiensi dari kode yang dibuat. Oleh karena itu diharapkan kepada peneliti selanjutnya untuk dapat menggunakan *script* yang lebih efisien dari segi kompleksitas serta ukuran kode.

DAFTAR PUSTAKA

Abidin, B., Prasetyaningrum ,I., & Karlita, T. (2012). Sistem Informasi Rumah Kos Online Berbasis Web dan Messaging, Skripsi, Fakultas Teknik, Institut Teknologi Sepuluh November, Surabaya.

Agape Sianturi, J., Piarsa, I. N., & Adi Purnawan, I. K. (2018). Aplikasi Pencarian dan Penyewaan Rumah Kost Berbasis Web dan Android. Jurnal Ilmiah Merpati (Menara Penelitian Akademika Teknologi Informasi), 6(3), 192. https://doi.org/10.24843/jim.2018.v06.i03.p06

Amrin, Larasati, M.D., & Satriadi, I. (2020). *Model Waterfall* Untuk Pengembangan Sistem Informasi Pengolahan Nilai Pada SMP Kartika XI-3 Jakarta Timur, 4(1). 135-140.

Annugerah, A., Astuti, I. F., & Kridalaksana, A. H. (2016). Sistem Informasi Geografis Berbasis Web Pemetaan Lokasi Toko Oleh-Oleh Khas Samarinda. Informatika Mulawarman : Jurnal Ilmiah Ilmu Komputer, 11(2), 43. https://doi.org/10.30872/jim.v11i2.213

Cahyono, A. B., & Dj, W. N. M. (2016). Perancangan Sistem Informasi Geografis Zona Nilai Tanah Berbasis Web Menggunakan *Leaflet JavaScript Library* (Studi Kasus: Kecamatan Kenjeran, Kecamatan Gubeng, Kecamatan Tambak Sari dan Kecamatan Bulak, Kota Surabaya, Jawa Timur). *Jurnal Teknik ITS*, 5(2).

Colifah, W.N., Yulianingsi, & Sagita, S.M. (2018). Pengujian *Black box testing* Pada Aplikasi Action & Strategy Berbasis Android Dengan Teknologi Phonegap, 3(2). 206-210.

Khuat, Tung. (2018). *Developing a Frontend Application Using ReactJS and Redux.*

Kosasih, S. (2015). Sistem Informasi Geografis Pemetaan Tempat Kost Berbasis Web. CSRID (Computer Science Research and Its Development Journal), 6(3), 171. https://doi.org/10.22303/csrid.6.3.2014.171-181

Pratikto, H.S., Suraya, & Sutanta, E. (2014). Sistem Pencarian dan Pemesanan Rumah Kos Menggunakan Sistem Informasi Geografi (SIG), *Jurnal Script*, 1(2). 41-48.

Rosdania., Agus, F., & K, A.H. (2015). Sistem Informasi Geografi Batas Wilayah Kampus Universitas Mulawarman Menggunakan Google Maps Api, 10(1), 38-46.

Trimarsiah, Y., & Arafat, M. (2017). Analisis Dan Perancangan *Website* Sebagai Sarana Informasi Pada Lembaga Bahasa Kewirausahaan Dan Komputer Akmi Baturaja, 19(1). 1-10.

Utami, E., & Raharjo, S. (2006). *RDBMS Dengan PostgreSQL Di GNU/Linux*. Yogyakarta: Andi.

Wiatr-Kmieciak, M. (2016). „Просто Наступила Зима" – Повторительный Урок Для Раздела 1 (Времена Года) Учебника „Вот И Мы 2". Studia Rossica Posnaniensia, 40(1), 285–292. https://doi.org/10.14746/strp.2015.40.1.28

LAMPIRAN

Peta.jsx

```
import React from "react";
import { Icon } from "leaflet";
import { Grid, Typography, Box, Button } from "@mui/material";
import wellknown from "wellknown";
import { MapContainer, TileLayer, Marker, Popup } from "react-leaflet";
import AssistantDirectionIcon from "@mui/icons-material/AssistantDirection";
import TheMapComponent from "./TheMapComponent";
import stadiumIconPng from "../data/Mapicons/stadium.png";
import hospitalIconPng from "../data/Mapicons/hospital.png";
import universityIconPng from "../data/Mapicons/university.png";
function Peta(props) {
  const { datapeta } = props;
  const stadiumIcon = new Icon({
    iconUrl: stadiumIconPng,
    iconSize: [40, 40],
  });
  const hospitalIcon = new Icon({
    iconUrl: hospitalIconPng,
    iconSize: [40, 40],
  });
  const universityIcon = new Icon({
    iconUrl: universityIconPng,
    iconSize: [40, 40],
  });
  const GoogleMapsShortcut = () => {
    const url =
`https://www.google.com/maps/search/?api=1&query=${datapeta.latitude},${data
peta.longitude}`;
```

```
  window.open(url, "_blank");
};
return (
 <>
  <Grid
   item
   container
   style={{ marginTop: "1rem" }}
   spacing={1}
   justifyContent="space-between"
  >
   <Grid item lg={3} md={3} sm={12} xs={12}>
    <Box style={{ height: "24rem", overflow: "auto" }}>
     {datapeta.listing_poi.map((poi) => {
      function DegreeToRadian(coordinate) {
       return (coordinate * Math.PI) / 180;
      }
      function CalculateDistance() {
       const lat1 = DegreeToRadian(datapeta.latitude);
       const lon1 = DegreeToRadian(datapeta.longitude);
       const coordinates = wellknown(poi.location).coordinates;
       if (coordinates && coordinates.length === 2) {
        const lat2 = DegreeToRadian(coordinates[0]);
        const lon2 = DegreeToRadian(coordinates[1]);
        const centralAngle = Math.acos(
         Math.sin(lat1) * Math.sin(lat2) +
          Math.cos(lat1) * Math.cos(lat2) * Math.cos(lon2 - lon1)
        );
        const R = 6371;
```

```
const distance = R * centralAngle;
        return distance.toFixed(2);
      } else {
        return "Invalid Coordinates";
      }
    }
    return (
      <div
        key={poi.id}
        style={{
          marginBottom: "0.5rem",
          borderBottom: "1px solid black",
        }}
      >
        <Typography style={{ fontSize: "14px", fontWeight: "bold" }}>
          {poi.name}
        </Typography>
        <Typography variant="subtitle1" style={{ fontSize: "12px" }}>
          {poi.type} |{" "}
          <span style={{ fontWeight: "bolder", color: "black" }}>
            {CalculateDistance()} Km
          </span>
        </Typography>
      </div>
    );
  })}
</Box>
<Button
  onClick={GoogleMapsShortcut}
  style={{
```

```
            backgroundColor: "#4CAF50",
            color: "white",
            fontSize: "16px",
            padding: "10px 20px",
            marginTop: "0.5rem",
            borderRadius: "5px",
            cursor: "pointer",
          }}
          startIcon={<AssistantDirectionIcon />}
        >
          google maps
        </Button>
      </Grid>
      <Grid item lg={9} md={9} sm={12} xs={12} style={{ height: "28rem" }}>
        <MapContainer
          center={[datapeta.latitude, datapeta.longitude]}
          zoom={16}
          scrollWheelZoom={true}
        >
          <TileLayer
            attribution='&copy; <a
href="https://www.openstreetmap.org/copyright">OpenStreetMap</a>
contributors'
            url="https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png"
          />
          {datapeta.listing_poi.map((poi) => {
            function PoiIcon() {
```

```
if (poi.type === "Stadium") {
            return stadiumIcon;
           } else if (poi.type === "Hospital") {
            return hospitalIcon;
           } else if (poi.type === "University") {
            return universityIcon;
           }
          }
          function DegreeToRadian(coordinate) {
           return (coordinate * Math.PI) / 180;
          }
          function CalculateDistance() {
           const lat1 = DegreeToRadian(datapeta.latitude);
           const lon1 = DegreeToRadian(datapeta.longitude);
           const coordinates = wellknown(poi.location).coordinates;
           if (coordinates && coordinates.length === 2) {
            const lat2 = DegreeToRadian(coordinates[0]);
            const lon2 = DegreeToRadian(coordinates[1]);
            const centralAngle = Math.acos(
             Math.sin(lat1) * Math.sin(lat2) +
              Math.cos(lat1) * Math.cos(lat2) * Math.cos(lon2 - lon1)
            );
            const R = 6371;
            const distance = R * centralAngle;
            return distance.toFixed(2);

        } else {
```

```
return "Invalid Coordinates";
            }
          }
          const coordinates = wellknown(poi.location).coordinates;
          return (
            <Marker
              key={poi.id}
              position={[coordinates[0], coordinates[1]]}
              icon={PoiIcon()}
            >
              <Popup>
                {poi.name} {CalculateDistance()} Km
              </Popup>
            </Marker>
          );
        })}
        <TheMapComponent listingInfo={datapeta} />
      </MapContainer>
    </Grid>
  </Grid>
 </>
 );
}
export default Peta;
```

ListingList.jsx

```
import React, { useState, useEffect } from "react";
import Axios from "axios";
import {
  Paper,
  InputBase,
  IconButton,
  Grid,
  Card,
  CardMedia,
  CardHeader,
  CardContent,
  Typography,
  Button,
  Box,
  CardActions,
} from "@mui/material";
import SearchIcon from "@mui/icons-material/Search";
import RoomIcon from "@mui/icons-material/Room";
import { useNavigate } from "react-router-dom";
import { makeStyles } from "@mui/styles";
const useStyles = makeStyles({
  cardStyle: {
    overflow: "hidden",
    border: "4px solid white",
    width: "98%",
    position: "relative",
    height: "21rem",
    marginBottom: "0.5rem",
  },
```

```
  pictureStyle: {
    height: "10rem",
    cursor: "pointer",
  },
  priceLabel: {
    fontSize: "12px",
    fontWeight: "bold",
    color: "skyblue",
    height: "8px",
  },
});
const ListingsList = ({ onFlyToMap }) => {
  const navigate = useNavigate();
  const classes = useStyles();
  const [allListings, setAllListings] = useState([]);
  const [searchTerm, setSearchTerm] = useState("");
  const [searchResults, setSearchResults] = useState([]);
  useEffect(() => {
    const source = Axios.CancelToken.source();
    async function GetAllListings() {
      try {
        const response = await Axios.get(
          "https://mikos03.onrender.com/api/listings/",
          { cancelToken: source.token }
        );
        setAllListings(response.data);
        setSearchResults(response.data);
      } catch (error) {}
    }
    GetAllListings();
```

```
return () => {
    source.cancel();
  };
 }, []);
 function handleSearch(event) {
  event.preventDefault();
  setSearchResults(allListings);
 }
 function fuzzySearch(needle, haystack) {
  const words = needle.toLowerCase().split(" ");
  return words.every((word) => {
   return haystack.toLowerCase().includes(word);
  });
 }
 const filteredListings = searchResults.filter((listing) => {
  return (
   fuzzySearch(searchTerm, listing.title) ||
   fuzzySearch(searchTerm, listing.address) ||
   fuzzySearch(searchTerm, listing.borough)
  );
 });
 const handleFlyTo = (listing) => {
  onFlyToMap(listing);
 };
 return (
  <>
   <Grid item xs={12} padding="0.5rem">
    <Paper
     component="form"
     sx={{
```

```
        display: "flex",
        alignItems: "center",
        paddingLeft: "0.8rem",
      }}
      onSubmit={handleSearch}
    >
      <InputBase
        style={{ height: "100%" }}
        id="outlined-basic"
        placeholder="Silakan Cari Di Sini..."
        fullWidth
        value={searchTerm}
        onChange={(e) => setSearchTerm(e.target.value)}
      />
      <IconButton type="button" aria-label="search">
        <SearchIcon />
      </IconButton>
    </Paper>
  </Grid>
  <Grid container padding="0.5rem" spacing={0}>
    {filteredListings.slice(0, 10).map((listing) => {
      return (
        <Grid item key={listing.id} xs={12} sm={6} md={12} lg={6}>
          <Card key={listing.id} className={classes.cardStyle}>
            <CardMedia
              className={classes.pictureStyle}
              component="img"
              height="160"
              image={listing.picture1}
              alt={listing.title}
```

```
onClick={() => navigate(`/listings/${listing.id}`)}
          />
          <CardHeader
           title={
            <Typography
             gutterBottom
             variant="h6"
             style={{ fontSize: "16px" }}
            >
             {listing.title.substring(0, 35)}
            </Typography>
           }
           action={
            <IconButton
             aria-label="settings"
             onClick={() => handleFlyTo(listing)}
            >
             <RoomIcon />
            </IconButton>
           }
          />
          <CardContent style={{ marginTop: "-30px" }}>
           <Typography
            gutterBottom
            variant="body4"
            component="div"
            style={{ fontSize: "11px" }}
           >
            {listing.address.substring(0, 90)}...
           </Typography>
```

```
</CardContent>
    <CardActions
     style={{
      position: "absolute",
      bottom: "0",
      width: "100%",
      backgroundColor: "white",
      zIndex: 99,
     }}
    >
     <Box
      width="100%"
      style={{
       display: "flex",
       justifyContent: "space-between",
       alignItems: "center",
       marginBottom: "2px",
      }}
     >
      <Typography
       className={classes.priceLabel}
       component="span"
       zIndex={99}
       style={{ fontSize: "14px" }}
      >
       {listing.price_month
        ? `Rp${listing.price_month.toLocaleString(
           "id-ID"
          )}/bulan`
        : "Harga tidak tersedia"}
```

```
    </Typography>
                <Button
                  variant="contained"
                  color="primary"
                  size="small"
                  onClick={() => navigate(`/listings/${listing.id}`)}
                >
                  Details
                </Button>
              </Box>
            </CardActions>
          </Card>
        </Grid>
      );
    })}
  </Grid>
 </>
 );
};
export default ListingsList;
```

Order.jsx

```
import React, { useState, useEffect, useCallback } from "react";
import { useNavigate, useParams } from "react-router-dom";
import Axios from "axios";
import { useSelector } from "react-redux";
import { useImmerReducer } from "use-immer";
import Swal from "sweetalert2";
// MUI
import {
  Grid,
  Typography,
  Box,
  TextField,
  Button,
  MenuItem,
  Snackbar,
  Dialog,
  DialogContent,
  DialogTitle,
  IconButton,
} from "@mui/material";
import { makeStyles } from "@mui/styles";
import { styled } from "@mui/material/styles";
import CloseIcon from "@mui/icons-material/Close";
const useStyles = makeStyles({
  formContainer: {
    width: "80%",
    marginLeft: "auto",
    marginRight: "auto",
    border: "5px solid lightWhite",
```

```
padding: "1rem",
  },
});
const BootstrapDialog = styled(Dialog)(({ theme }) => ({
  "& .MuiDialogContent-root": {
    padding: theme.spacing(2),
  },
  "& .MuiDialogActions-root": {
    padding: theme.spacing(1),
  },
}));
function Order(props) {
  const { id, rumah } = props;
  const classes = useStyles();
  const navigate = useNavigate();
  const [modalOpen, setModalOpen] = useState(false);
  const isAuthenticated = useSelector((state) => state.auth.isAuthenticated);
  const isCustomer = useSelector((state) => state.auth.isCustomer);
  const userId = useSelector((state) => state.auth.userId);
  const customerId = useSelector((state) => state.auth.customerId);
  const isOwner = useSelector((state) => state.auth.isOwner);
  const params = useParams();
  useEffect(() => {
    if (!isAuthenticated) {
      navigate("/login");
    }
  }, [isAuthenticated, navigate]);
  useEffect(() => {
    if (!isCustomer) {
      navigate("/login");
```

```
  }
 }, [isCustomer, navigate]);
 useEffect(() => {
  if (isOwner) {
   navigate("/owner/home");
  }
 }, [isOwner, navigate]);
 const initialState = {
  fullNameValue: "",
  phoneNumberValue: "",
  noRekeningValue: "",
  buktiTransferValue: "",
  nominalValue: "",
  uploadedPicture: [],
  openSnack: false,
 };
 const [state, dispatch] = useImmerReducer(ReducerFuction, initialState);
 function ReducerFuction(draft, action) {
  switch (action.type) {
   case "catchUserOrderInfo":
    draft[action.name] = action.value;
    break;
   case "catchOrderInfo":
    draft.buktiTransferValue = action.profileOrder.buktiTransfer;
    break;
   case "catchUploadedPicture":
    draft.uploadedPicture = action.pictureChosen;
    break;
   case "catchProfilePictureChange":
    draft.buktiTransferValue = action.buktiTransferChosen;
```

```
break;
    case "catchUserProfileInfo":
      draft.userProfile.fullName = action.profileObject.full_name;
      draft.userProfile.phoneNumber = action.profileObject.phone_number;
      draft.userProfile.noRekeningValue = action.profileObject.no_rekening;
      break;
    case "openTheSnack":
      draft.openSnack = true;
      break;
    case "sendRequest":
      draft.sendRequest = true;
      break;
    case "requestSent":
      draft.sendRequest = false;
      break;
    default:
      return draft;
  }
}
useEffect(() => {
  if (state.uploadedPicture[0]) {
    dispatch({
      type: "catchProfilePictureChange",
      buktiTransferChosen: state.uploadedPicture[0],
    });
  }
}, [state.uploadedPicture, dispatch]);
const handleChange = useCallback(
  (e) => {
    const { name, value } = e.target;
```

```
dispatch({ type: "catchUserOrderInfo", name, value });
  },
  [dispatch]
);
const handleSubmit = useCallback(
  async (e) => {
    e.preventDefault();
    dispatch({ type: "sendRequest" });
    const formData = new FormData();
    formData.append("fullName", state.fullNameValue);
    formData.append("phoneNumber", state.phoneNumberValue);
    formData.append("rentalFrequency", state.rentalFrequencyValue);
    formData.append("nominal", state.nominalValue);
    formData.append("buktiTransfer", state.buktiTransferValue);
    formData.append("kamar", id);
    formData.append("customer", customerId);
    formData.append("user", userId);
    Swal.fire({
      title: "Are you sure?",
      text: "Are you sure you want to order this room?",
      icon: "warning",
      showCancelButton: true,
      confirmButtonColor: "#3085d6",
      cancelButtonColor: "#d33",
      confirmButtonText: "Yes, ordered it!",
    }).then(async (result) => {
      if (result.isConfirmed) {
        try {
          const response = await Axios.post(
            `https://mikos03.onrender.com/api/transaction/create`,
```

```
          formData
        );
        await updateKamar(params.id, true);
        Swal.fire("Ordered!", "Your ordered has been success.", "success");
        dispatch({ type: "openTheSnack" });
       } catch (error) {
        dispatch({ type: "requestSent" });
       }
     }
   });
  },
  [
   dispatch,
   state.fullNameValue,
   state.phoneNumberValue,
   state.nominalValue,
   state.buktiTransferValue,
   state.rentalFrequencyValue,
   params.id,
   customerId,
   userId,
  ]
);
useEffect(() => {
 async function GetProfileInfo() {
  try {
   const response = await Axios.get(
    `https://mikos03.onrender.com/api/profiles/customer/${userId}/`
   );
   dispatch({
```

```
        type: "catchUserOrderInfo",
        name: "fullNameValue",
        value: response.data.full_name,
      });
      dispatch({
        type: "catchUserOrderInfo",
        name: "phoneNumberValue",
        value: response.data.phone_number,
      });
    } catch (e) {}
  }
  GetProfileInfo();
}, [userId, dispatch]);
useEffect(() => {
  async function GetRumahInfo() {
    try {
      const response = await Axios.get(
        `https://mikos03.onrender.com/api/listings/${rumah}/`
      );
      dispatch({
        type: "catchUserOrderInfo",
        name: "noRekeningValue",
        value: response.data.no_rekening,
      });
    } catch (e) {}
  }
  GetRumahInfo();
}, [params.rumah, dispatch]);
async function updateKamar(id, newValue) {
  try {
```

```
  await Axios.patch(
    `https://mikos03.onrender.com/api/kamar/${id}/update/`,
    {
     barang_dipesan: newValue,
    }
  );
 } catch (error) {}
}
function SubmitButtonDisplay() {
 if (
  isAuthenticated &&
  state.fullNameValue !== null &&
  state.fullNameValue !== "" &&
  state.phoneNumberValue !== null &&
  state.phoneNumberValue !== ""
 ) {
  return (
   <Button
    variant="contained"
    fullWidth
    type="submit"
    className={classes.loginBtn}
    disabled={state.disabledBtn}
   >
    SUBMIT
   </Button>
  );
 } else if (
  isAuthenticated &&
  (state.fullNameValue === null ||
```

```
    state.fullNameValue === "" ||
      state.phoneNumberValue === null ||
      state.phoneNumberValue === "")
  ) {
   return (
    <Button
     variant="outlined"
     fullWidth
     className={classes.loginBtn}
     onClick={() => navigate("/profileCustomer")}
    >
     COMPLETE YOUR PROFILE TO ORDER KAMAR
    </Button>
   );
  } else if (!isAuthenticated) {
   return (
    <Button
     variant="outlined"
     fullWidth
     onClick={() => navigate("/login")}
     className={classes.loginBtn}
    >
     SIGN IN TO ADD A PROPERTY
    </Button>
   );
  }
 }
 const handleOpenClick = async () => {
  try {
   setModalOpen(true);
```

```
  } catch (error) {
    console.error(error);
  }
};
const handleCloseModal = () => {
  setModalOpen(false);
};
useEffect(() => {
  if (state.openSnack) {
    setTimeout(() => {
      navigate(`/riwayatTransaksi`);
    }, 1500);
  }
}, [state.openSnack, navigate]);
return (
  <>
    <Button variant="contained" onClick={handleOpenClick}>
      Order
    </Button>
    <BootstrapDialog
      open={modalOpen}
      onClose={handleCloseModal}
      style={{ zIndex: 1000, marginTop: "3rem" }}
    >
      <DialogTitle sx={{ m: 0, p: 2 }} id="customized-dialog-title">
        Pesan Kamar
      </DialogTitle>
      <IconButton
        aria-label="close"
        onClick={handleCloseModal}
```

```
    sx={{
      position: "absolute",
      right: 8,
      top: 8,
      color: (theme) => theme.palette.grey[500],
    }}
  >
    <CloseIcon />
  </IconButton>
  <DialogContent dividers id="modal-content" style={{ overflow: "auto" }}>
    <Box
      width="100%"
      margin="auto"
      border="4px solid white"
    >
      <Box
        height="40px"
        backgroundColor="#1E90FF"
        justifyContent="center"
        alignItems="center"
        display="flex"
      >
        <Typography style={{ marginLeft: "10px", color: "white" }}>
          Pesan Kamar Kos Sekarang
        </Typography>
      </Box>
      <Box height="550px" backgroundColor="#F8F8FF">
        <div className={classes.formContainer}>
          <form onSubmit={handleSubmit}>
            <Grid item container style={{ marginTop: "1rem" }}>
```

```
<TextField
  id="username"
  label="Nama Penyewa"
  variant="outlined"
  fullWidth
  size="small"
  value={state.fullNameValue}
  onChange={handleChange}
  name="fullNameValue"
 />
</Grid>
<Grid item container style={{ marginTop: "1rem" }}>
 <TextField
  id="username"
  label="No. Telp"
  variant="outlined"
  fullWidth
  size="small"
  value={state.phoneNumberValue}
  onChange={handleChange}
  name="phoneNumberValue"
 />
</Grid>
<Grid item container style={{ marginTop: "1rem" }}>
 <TextField
  id="no_rekening"
  label="No. Rekening Tujuan"
  variant="outlined"
  fullWidth
  size="small"
```

```
          value={state.noRekeningValue}
            onChange={handleChange}
            name="noRekeningValue"
           />
          </Grid>
          <Grid margin="auto" marginTop="1rem">
           <TextField
            id="number"
            label="Durasi Penyewaan"
            variant="outlined"
            fullWidth
            size="small"
            value={state.rentalFrequencyValue}
            onChange={handleChange}
            name="rentalFrequencyValue"
            select
           >
            <MenuItem value="Year">Year</MenuItem>
            <MenuItem value="Month">Month</MenuItem>
            <MenuItem value="Day">Day</MenuItem>
           </TextField>
          </Grid>
          <Grid item container style={{ marginTop: "1rem" }}>
           <TextField
            id="nominal pembayaran"
            label="Jumlah Pembayaran"
            variant="outlined"
            fullWidth
            size="small"
            value={state.nominalValue}
```

```
 onChange={handleChange}
   name="nominalValue"
 />
</Grid>
<Grid
 item
 container
 xs={6}
 style={{
   marginTop: "1rem",
 }}
>
 <Button
   variant="outlined"
   component="label"
   style={{ textAlign: "center" }}
 >
   upload struk
   <input
     type="file"
     accept="image/png, image/gif, image/jpeg"
     hidden
     onChange={(e) =>
       dispatch({
         type: "catchUploadedPicture",
         pictureChosen: e.target.files,
       })
     }
   />
 </Button>
```

```
          <Typography style={{ marginTop: "1rem" }}>
            {state.buktiTransferValue ? (
              <p>{state.buktiTransferValue.name}</p>
            ) : (
              ""
            )}
          </Typography>
        </Grid>
        <Grid style={{ marginTop: "1rem" }}>
          {SubmitButtonDisplay()}
        </Grid>
        <Grid item container style={{ marginTop: "1rem" }}>
          <Typography variant="small">
            Sudah memesan?{" "}
            <span
              onClick={() => navigate(`/riwayatTransaksi`)}
              style={{ color: "#2F80ED", cursor: "pointer" }}
            >
              Lihat transaksi kamu disini
            </span>
          </Typography>
        </Grid>
      </form>
      <Snackbar
        open={state.openSnack}
        message="You have successfully Order Rumah Kos!"
        anchorOrigin={{
          vertical: "bottom",
          horizontal: "center",
        }}
```

```
        />
            </div>
          </Box>
        </Box>
      </DialogContent>
    </BootstrapDialog>
  </>
 );
}
export default Order;
```