BAB III
METODOLOGI PENELITIAN

3.1 Subyek dan Obyek Penelitian

Berdasarkan latar belakang yang telah disusun, maka objek penelitian ini berupa *website* pemesanan *online* pendakian Gunung Slamet sedangkan subjeknya adalah *stakeholder Basecamp* Bambangan.

3.2 Alat dan Bahan Penelitian

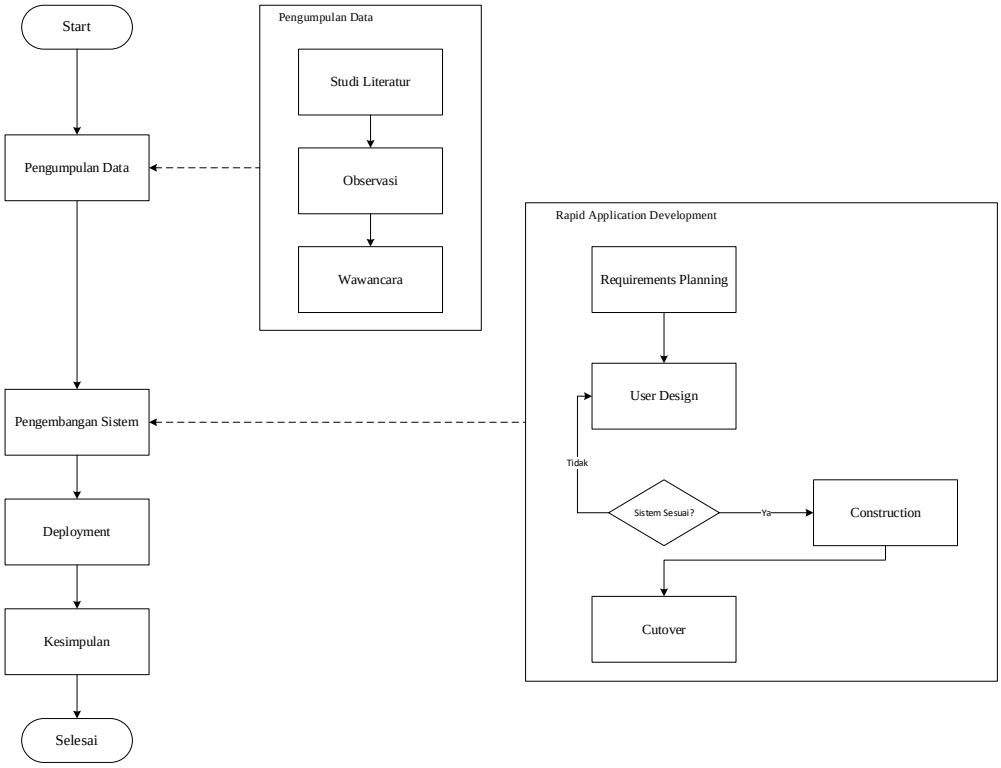
3.2.1. Alat

Alat yang digunakan adalah sebuah laptop dengan CPU: Intel® Core™ i5, Graphics: Nvidia Gtx 1050, RAM 8 GB, Disk: SSD 256 GB dan Harddisk 500 GB.

3.2.2. Bahan

Bahan yang digunakan berupa informasi mengenai kebutuhan *web development* yang berasal dari stakeholder.

3.3 Diagram Alir Penelitian/ Proses Penelitian



Gambar 3.1 Diagram Alir Penelitian

Gambar 3.1 adalah diagram alir atau *flowchart* dari penelitian yang akan dilakukan. Alur proses penelitian tersebut didasarkan pada *metode* penelitian yang penulis tentukan yaitu *Rapid Application Development*.

3.3.1. Pengumpulan Data

Proses ini merupakan pengumpulan data melalui studi literatur, observasi ke *Basecamp* Bambangan, dan wawancara dengan para *stakeholder*. Pengumpulan data ini akan dilakukan agar peneliti dapat memperoleh informasi secara tepat. Data yang dikumpulkan seperti data calon *user* dan fitur akan menjadi modal awal untuk mengidentifikasi objektif dari sistem yang akan dibuat. Berikut merupakan data yang didapatkan:

a. Calon pendaki kesulitan untuk memperoleh informasi kuota pendakian pada tanggal yang diinginkan.

b. Calon Pendaki merasa membuang waktu saat antri untuk melakukan pendaftaran di basecamp bambangan

c. Admin basecamp bambangan kesulitan untuk melakukan input data karena antrean yang panjang

3.3.2. Pengembangan Sistem (*Rapid Application Development*)

3.3.2.1. *Requirements Planning*

Pada tahap ini, peneliti akan mengidentifikasi seluruh data yang telah diperoleh sebelumnya. Dengan pengidentifikasian data dan didasarkan pada permasalahan yang terjadi tersebut, maka objektif sistem ini dapat ditentukan dengan tepat guna. Berikut merupakan kebutuhan kebutuhan pengguna berdasarkan data yang telah didapat:

a. Calon pendaki dapat mendapatkan informasi kuota pendakian berdasarkan tanggal.

b. Calon Pendaki dapat melakukan pemesanan secar online

c. Calon Pendaki dapat melakuka pengecekan kode bookin yang didapat

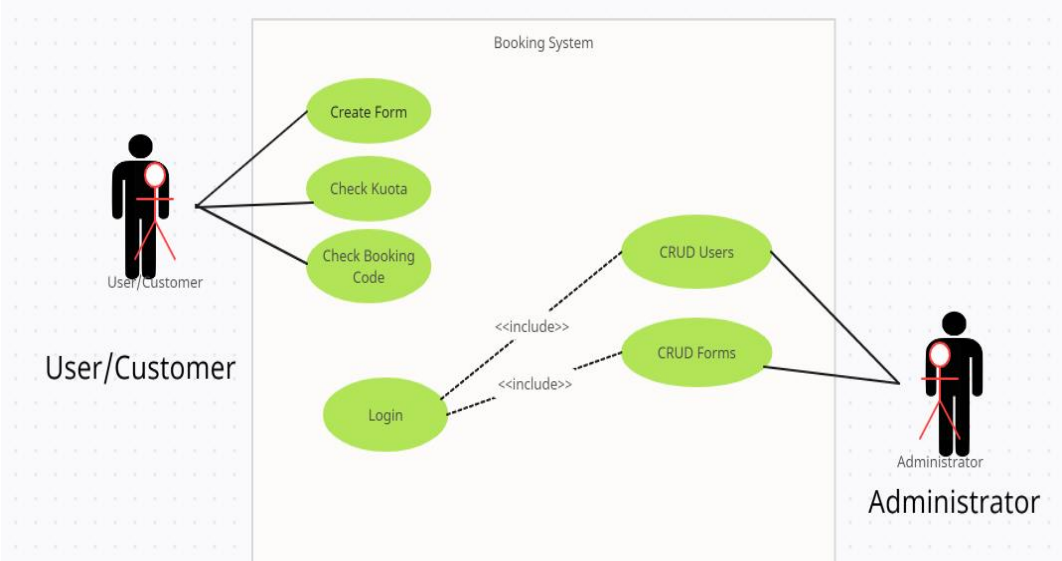
d. Admin dapat melakukan melakukan Create, Read, Update dan Delete pada data pendaki

e. Admin dapat melakukan Create, Read, Update dan Delete pada data login Admin

3.3.2.2. *User Design*

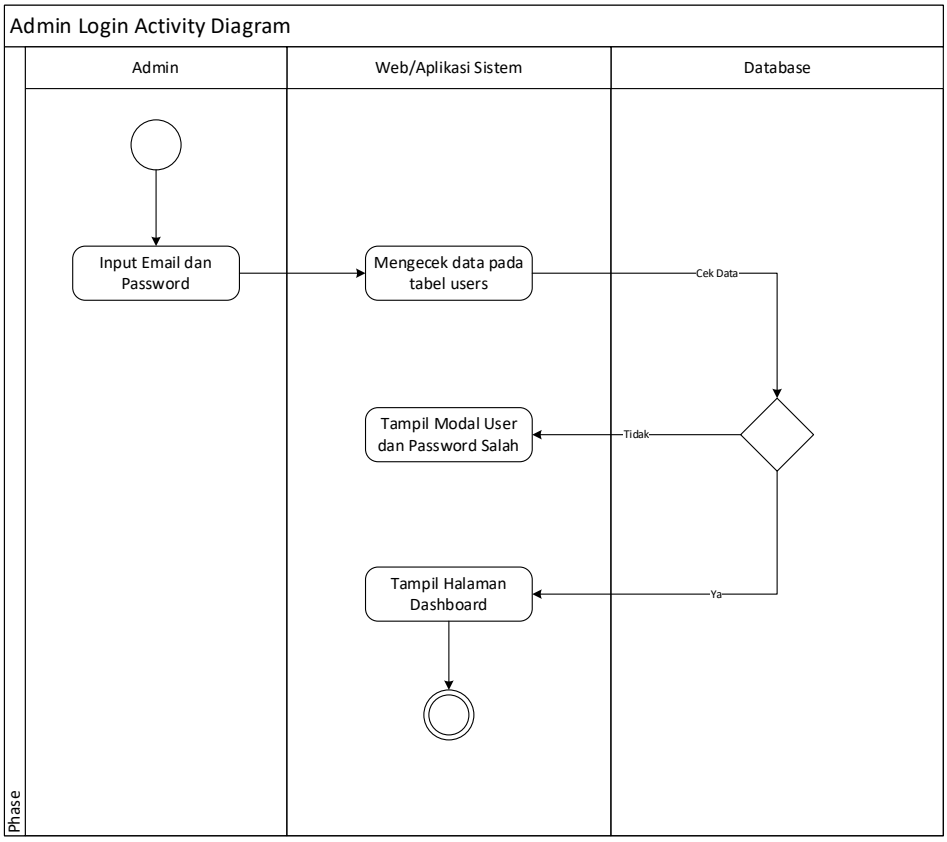
Tahap ini merupakan *prototype development*. Semua yang terjadi dalam prosesnya, didasarkan pada objektif sistem yang telah

diidentifikasi sebelumnya melalui data yang telah diolah. Pemodelan dalam perancangan ini menggunakan *Unified Modelling Language* (UML) meliputi *use case diagram*, *activity diagram*, *sequence diagram*, entity relationship diagram kemudian *interface design (Mockup).*



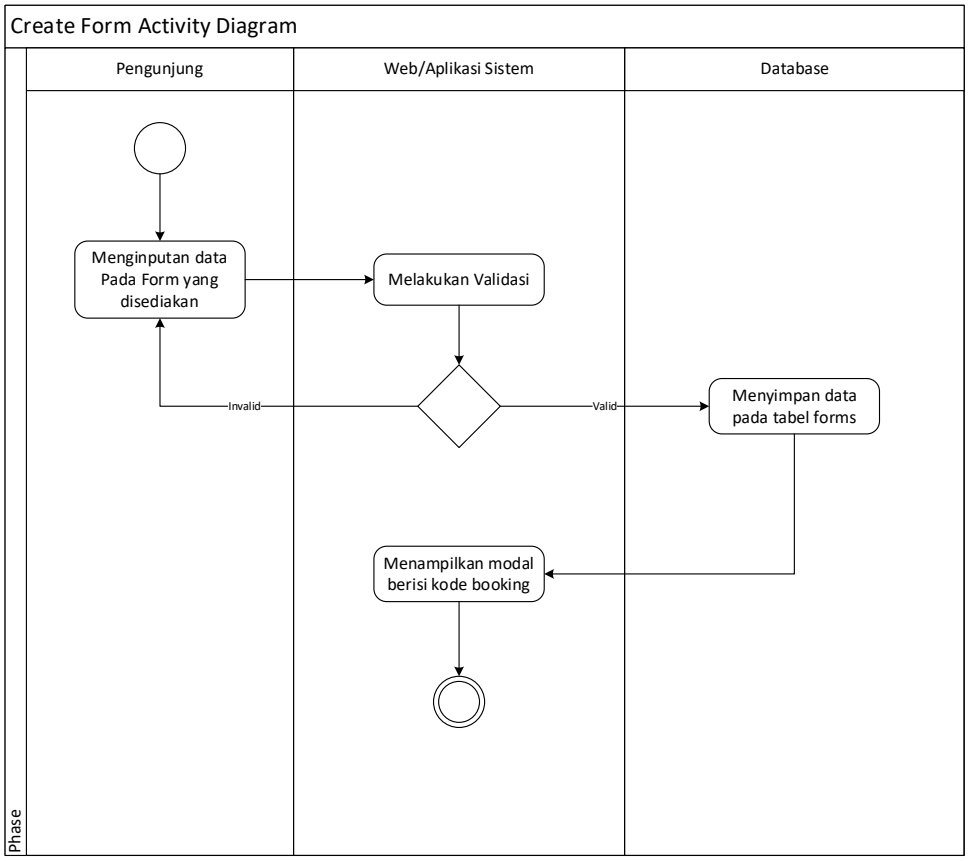
Gambar 3.2 *Use Case Diagram*

Melalui diagram di atas, terdapat dua aktor yaitu Pengunjung dan *Admin*. Pengunjung memiliki akses untuk meng*input* form pendaftaran dan mencetak kode *booking*. Sedangkan aktor *Admin* memiliki akses untuk membuat, melihat, mengedit, dan menghapus data pendaftaran.



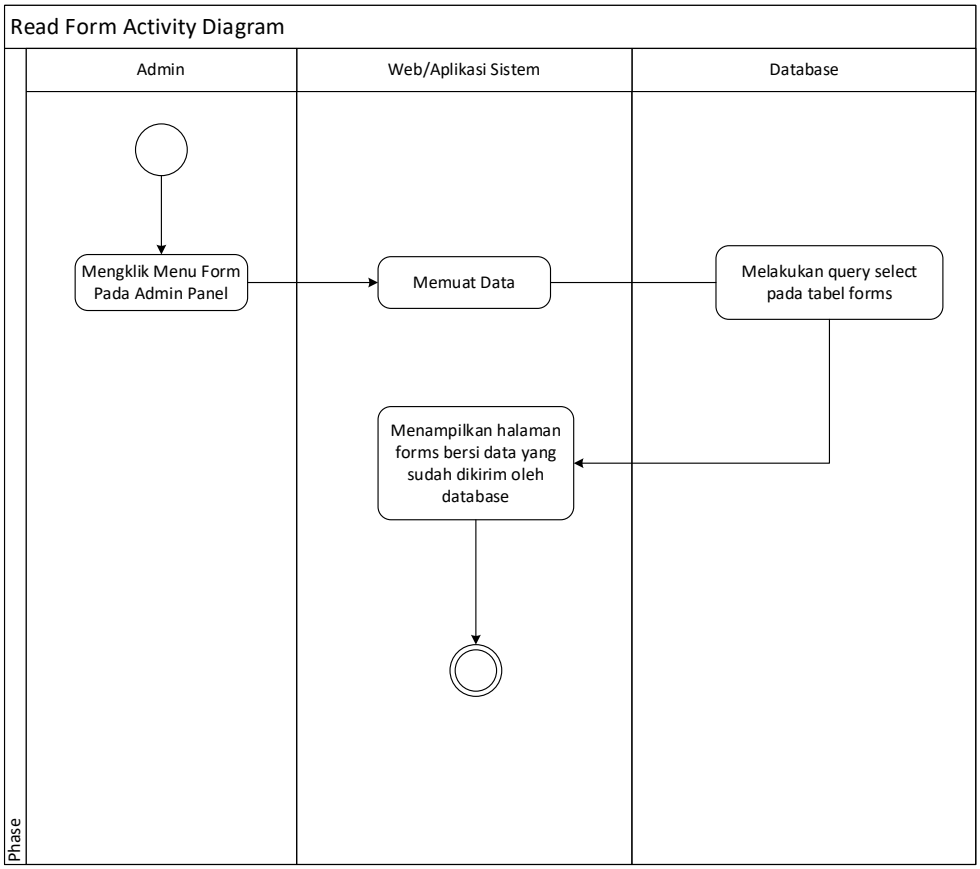
Gambar 3.3 ***Activity Diagram Login Admin***

Alur aktivitas diagram *Login* pada *actor admin* tergambar pada Gambar 3.3. merinci secara visual alur aktivitas yang harus dilalui saat seorang admin melakukan proses login. Ketika admin mengakses atau membuka halaman login, serangkaian langkah yang terstruktur secara sistematis akan dijalankan, mengarah pada otentikasi identitas.



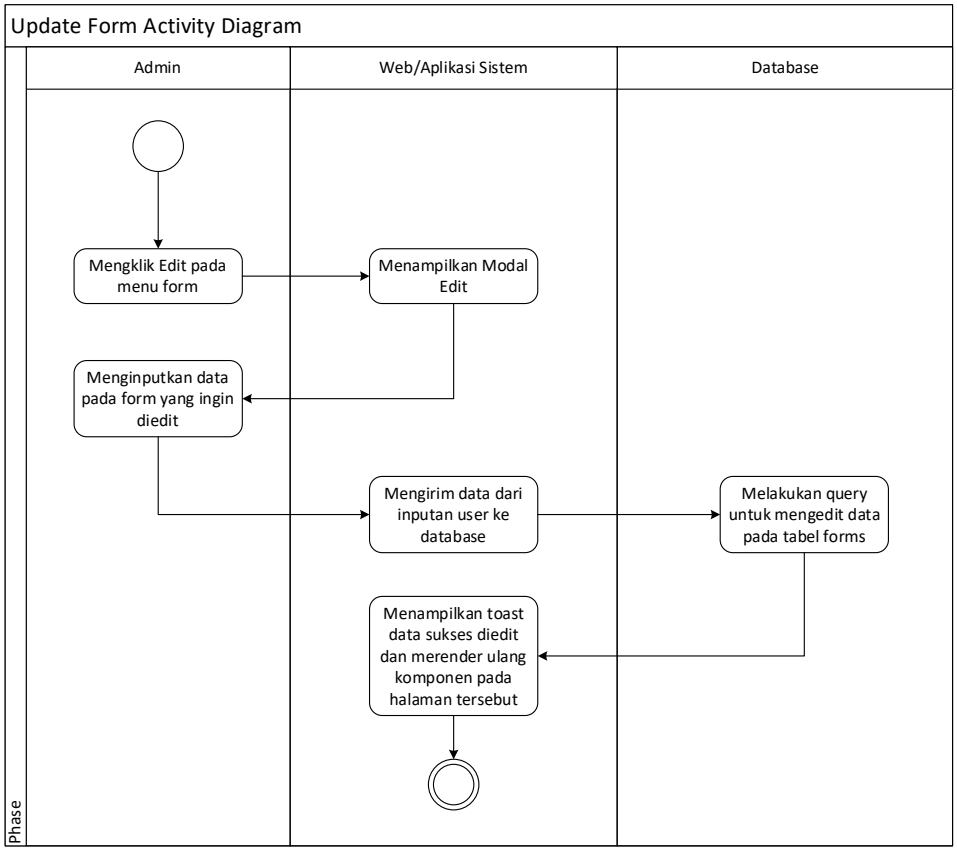
Gambar 3.4 ***Activity Diagram Create Form***

Alur aktivitas diagram *Create Form* pada *actor* pengunjung tergambar pada Gambar 3.4. merinci secara visual alur aktivitas yang harus dilalui saat seorang pengunjung melakukan proses pembuatan form ketika pengunjung melakukan input data dan melakukan submit terjadi validasi oleh sistem kemudian ketika data lolos validasi akan disimpan dalam database dan website menampilkan kode booking yang didapat.



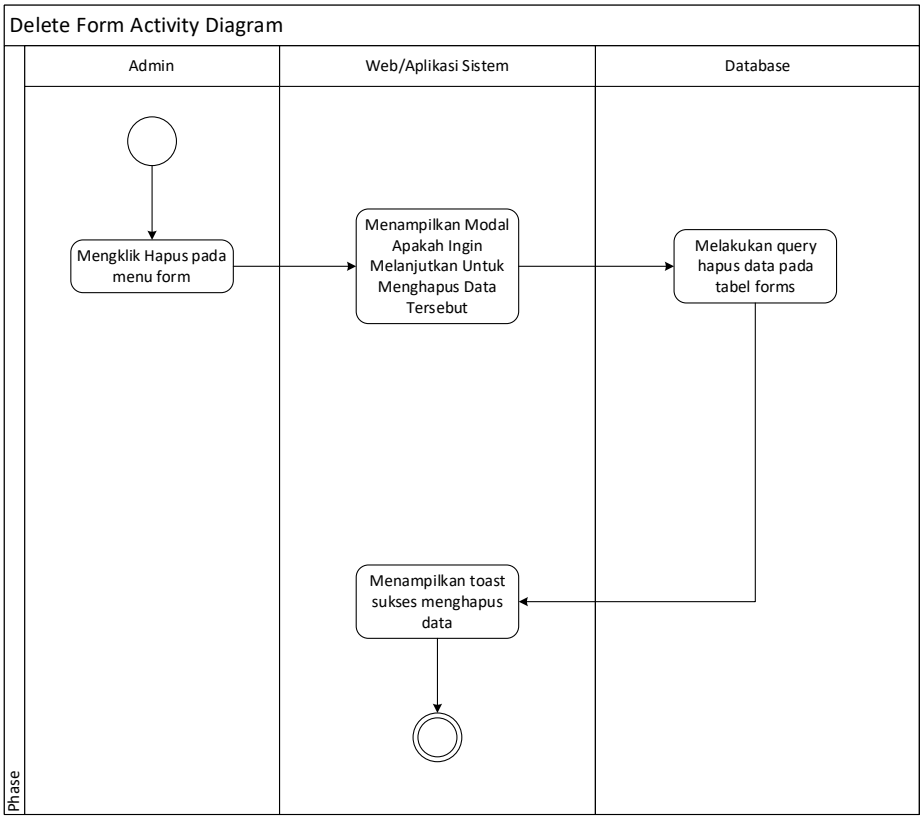
Gambar 3.5 ***Activity Diagram Read Form***

Alur aktivitas diagram *Read Form* pada *actor* admin tergambar pada Gambar 3.5. merinci secara visual alur aktivitas yang harus dilalui saat seorang admin saat melakukan klik pada menu form lalu website akan melakukan permintaan request ke database untuk melakukan query selecet pada tabel forms kemudian website meresponse dengan menampilkan halaman yang menampilkan data yang didapat dari query database.



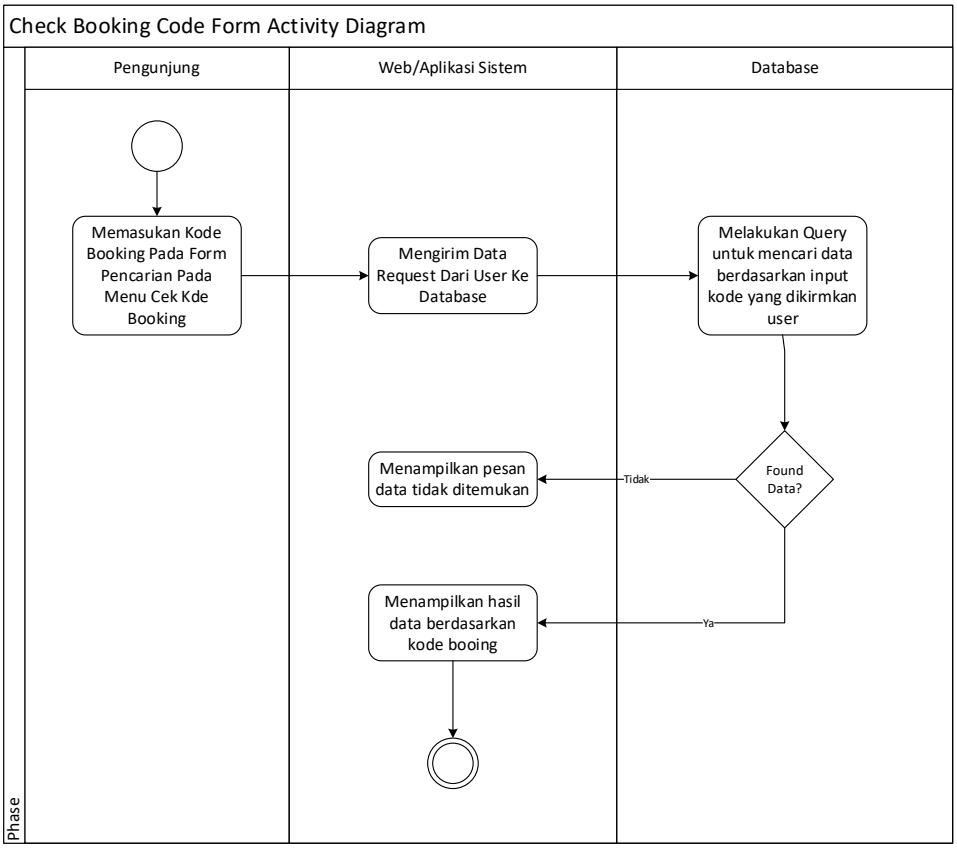
Gambar 3.6 ***Activity Diagram Update Form***

Alur aktivitas diagram *Update Form* pada *actor* admin tergambar pada Gambar 3.6. merinci secara visual alur aktivitas yang harus dilalui saat seorang admin saat melakukan klik edit pada menu form lalu website akan melakukan permintaan request ke database untuk melakukan query edit data pada tabel forms kemudian website meresponse dengan menampilkan halaman yang menampilkan data yang didapat dari query database.



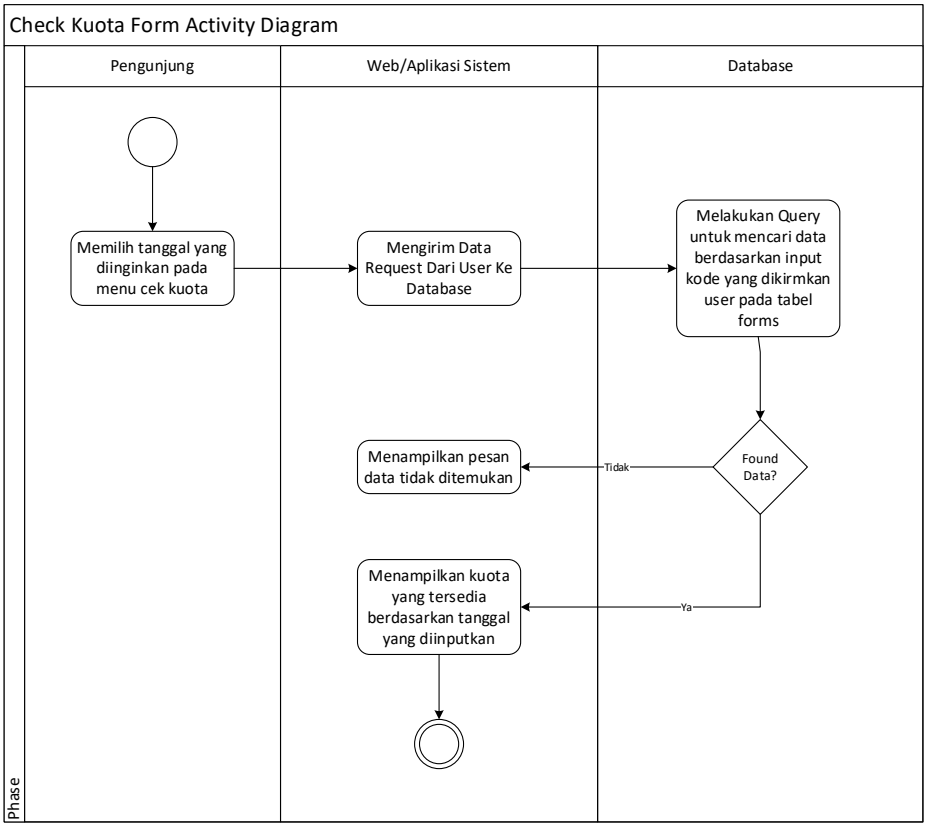
Gambar 3.7 *Activity Diagram Delete Form*

Alur aktivitas diagram *Delete Form* pada *actor* admin tergambar pada Gambar 3.7. merinci secara visual alur aktivitas yang harus dilalui saat seorang admin saat melakukan klik delete pada menu form lalu website akan melakukan permintaan request ke database untuk melakukan query delete data pada tabel forms kemudian website meresponse dengan menampilkan halaman perubahan setelah terjadi penghapusan data



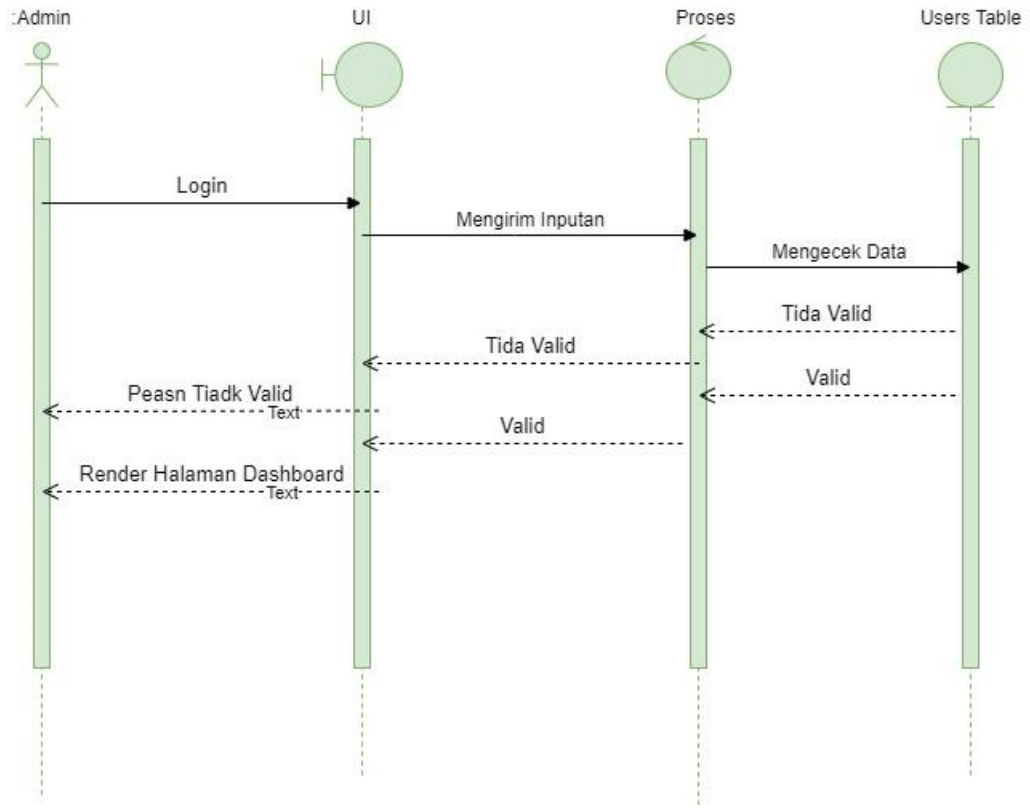
Gambar 3.8 ***Activity Diagram Check Booking Code***

Alur aktivitas diagram *Check Booking Code* pada *actor* pengunjung tergambar pada Gambar 3.8. merinci secara visual alur aktivitas yang harus dilalui saat seorang pengunjung menginputkan kode booking lalu system akan mengirimkan inputan user ke database untuk dicari data yang sesuai kemudian ditampilkan oleh system data tersebut.



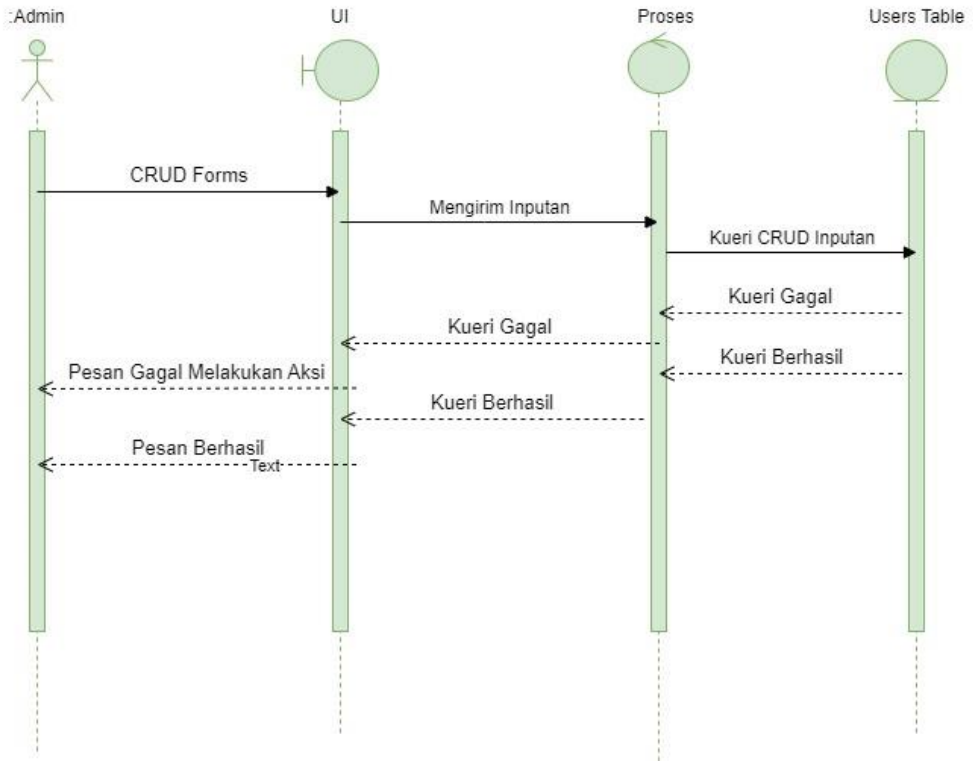
Gambar 3.9 ***Activity Diagram Kuota Check***

Alur aktivitas diagram *Chek Kuota* pada *actor* pengunjung tergambar pada Gambar 3.9. merinci secara visual alur aktivitas yang harus dilalui saat seorang pengunjung memilih tanggal pendakian kemudia mengkik submit lalu web akan mengirim input dari user ke database ntuk dilakukan query kemudian database akan mengembalikan hasil dari query ke website untuk ditampilkan ke pengunjung.



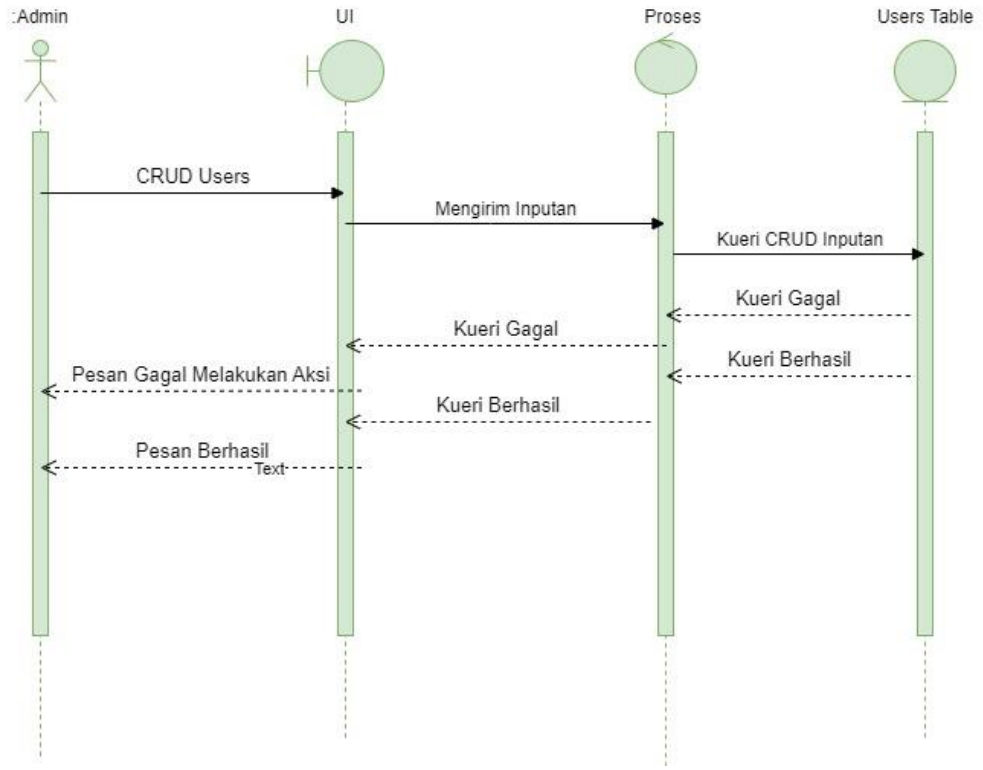
Gambar 3.10 ***Sequence Diagram Admin Login***

Gambar 3.10. merinci secara visual alur aktivitas yang admin lakukan pertama admin melakukan login kemudian request inputan oleh admin di proses oleh api kemudian ketika login berhasil admin dapat menggunakan fitur CRUD users/forms.



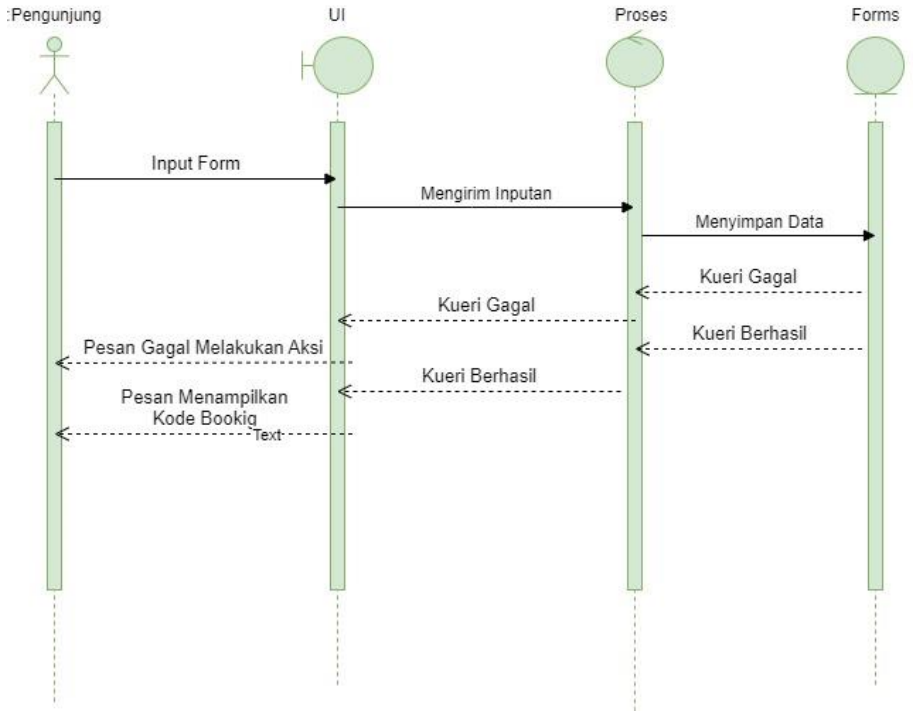
Gambar 3.11 ***Sequence Diagram Admin Crud Forms***

Gambar 3.11. merinci secara visual alur aktivitas yang admin lakukan saat melakukan Create Read Update dan Delete pada fitur forms kemudian melakukan kueri pada tabel forms lalu sistem aka mengembalikn response berhasil atau gagal.



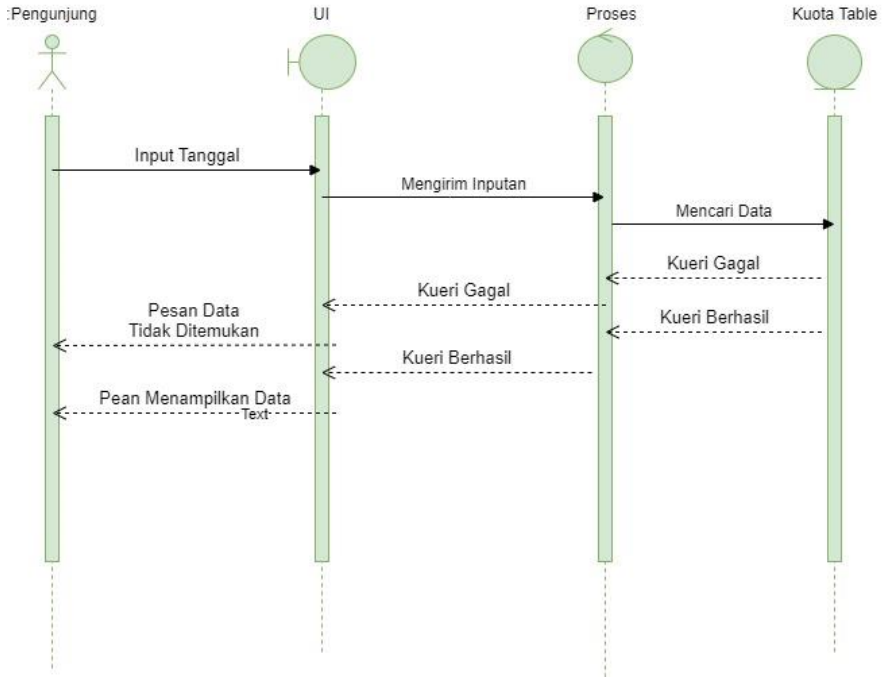
Gambar 3.12 ***Sequence Diagram Admin Crud Users***

Gambar 3.12. merinci secara visual alur aktivitas yang admin lakukan saat melakukan Create Read Update dan Delete pada fitur users kemudian melakukan kueri pada tabel users lalu sistem aka mengembalikn response berhasil atau gagal.



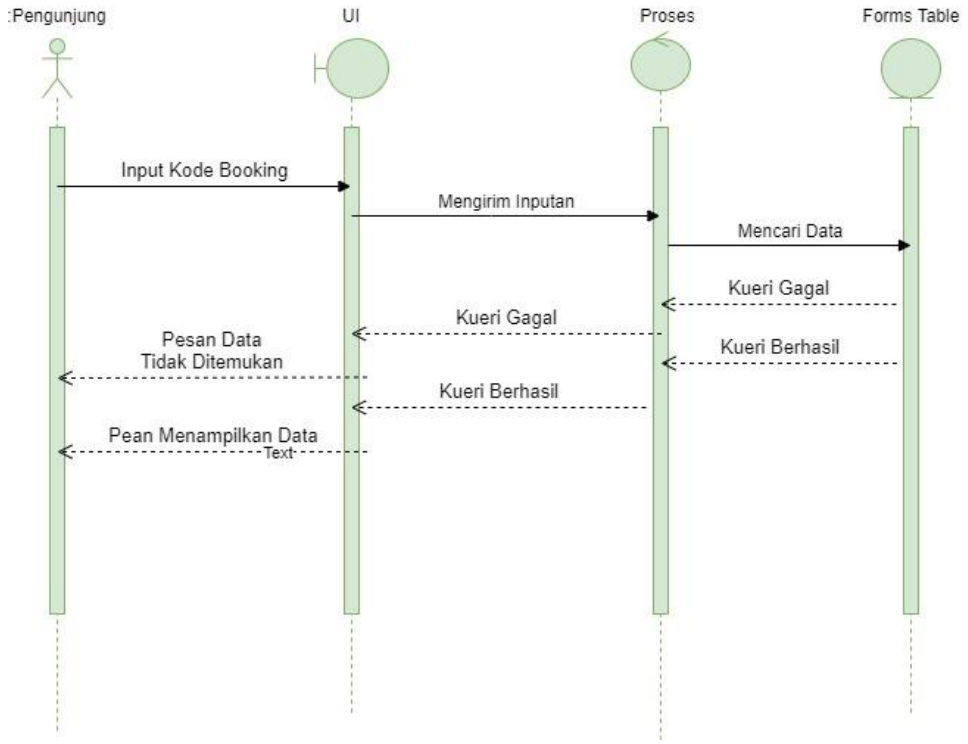
Gambar 3.13 *Sequence Diagram* Pengunjung *Create Form*

Gambar 3.13. merinci secara visual alur aktivitas yang pengunjug lakukan saat menginputkan data pada formular pendaftaran kemudian sistem mengirim inutkan ke database untuk dilakukan kueri penyimpanan data kemudian hasilnya akan diproses oleh sistem untuk mengembalikan response ke pengunjung.



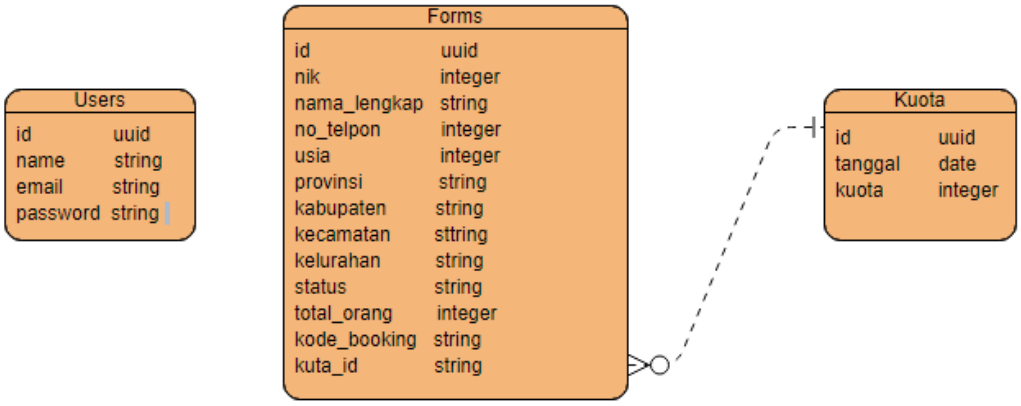
Gambar 3.14 *Sequence Diagram* Pengunjung *Check Kuota*

Gambar 3.14. merinci secara visual alur aktivitas yang pengunjug saat melakukan pengecekan sisa kuota yang ada berdasarkan tanggal yang diinputkan.



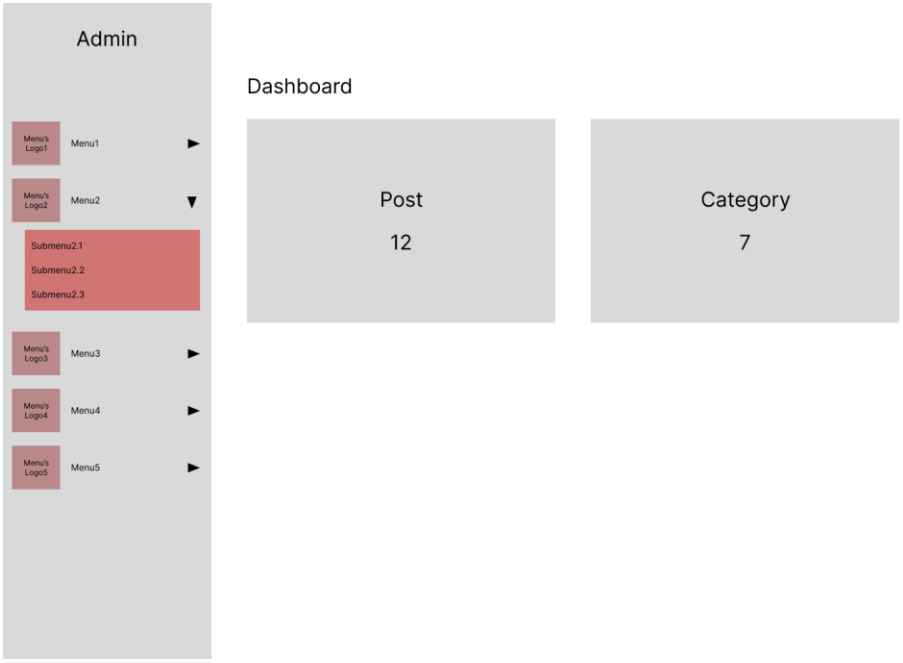
Gambar 3.15 ***Sequence Diagram*** **Pengunjung** ***Check Booking Code***

Gambar 3.15. merinci secara visual alur aktivitas pengunjung saat melakukan pengecekan kode booking berdasarkan kode booking yang pengunjung inputkan.



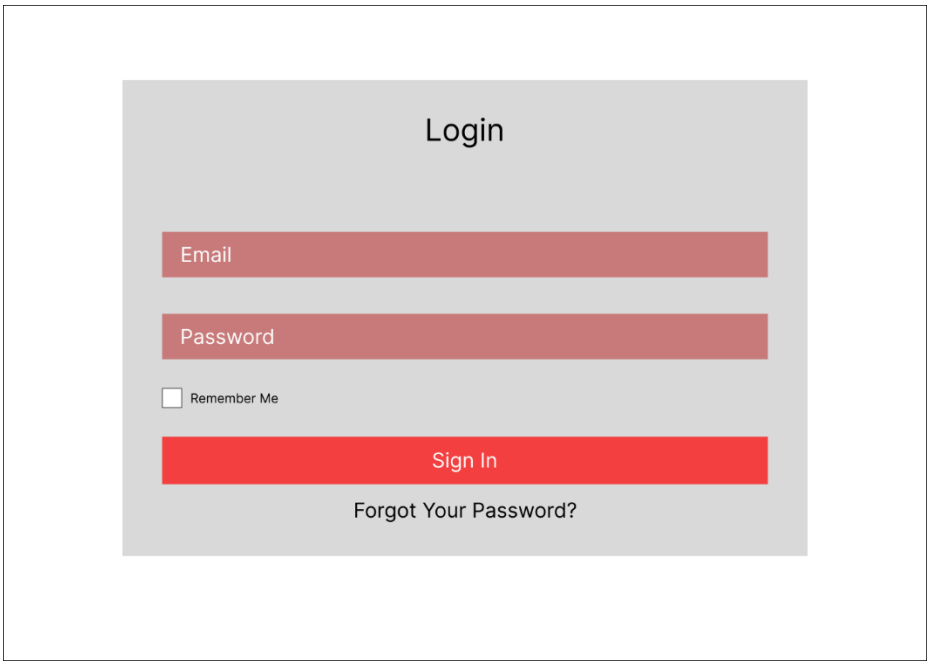
Gambar 3.12 *Entity Relationship Diagram*

Pada Gambar 3.12, terdapat Entity-Relationship Diagram (ERD) yang terdiri dari tiga entitas, yaitu "users", "forms", dan "kuota". Dalam ERD ini, terdapat relasi antara tabel "kuota" dan "forms", yang dikenal sebagai relasi "one-to-many". Dalam konteks ini, artinya setiap entitas dalam tabel "kuota" dapat memiliki banyak entitas yang terhubung dalam tabel "forms". Di sisi lain, tabel "users" tidak memiliki relasi yang terhubung dengan tabel lainnya.



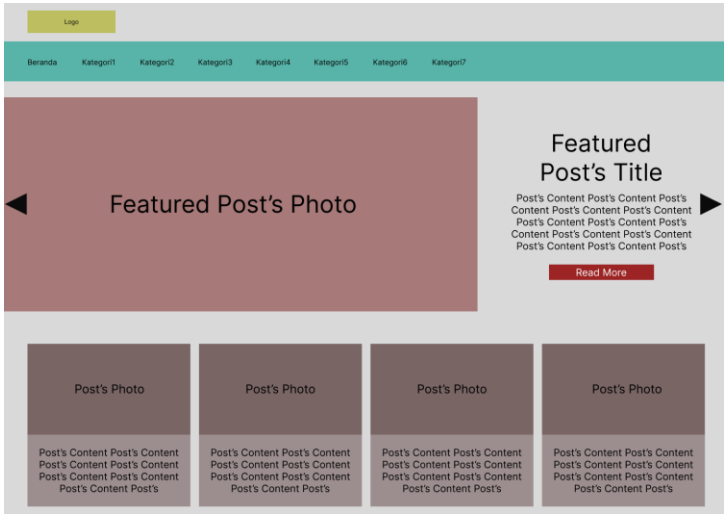
Gambar 3.13 *Mockup Admin DashbardPage*

Pada Gambar 3.13, terdapat mockup admin dashboard page yang dapat diakses ketika adin sudah melakukan proses autentikasi pada halaman login sebelum bisa megakses dashboard page.



Gambar 3.14 ***Mockup Admin LoginPage***

Pada Gambar 3.14, terdapat mockup login admin page. Saat admin sudah mengisi credentials pada form login tersebut admin akan langsung diarakan ke halaman admin dashboard ketika credentials yang dimasukan benar

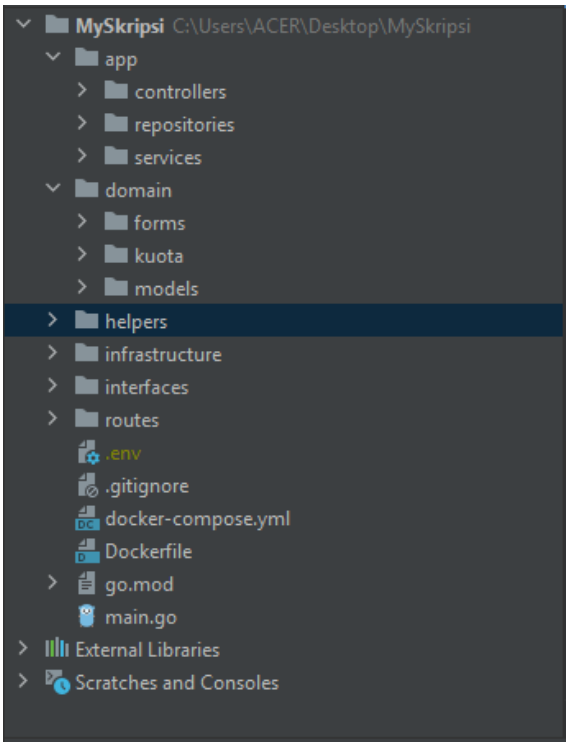


Gambar 3.15 ***Mockup HomePage***

Pada Gambar 3.15, terdapat mockup homepage yang dapat diakses leh semua pengunjung tanpa harus melakukan autentikasi.

3.3.2.3. *Construction*

Proses pembangunan ini didasarkan pada rancangan atas prototipe terakhir yang dirancang sebelumnya dan disetujui. Sistem ini dibangun menggunakan bahasa pemrograman *Golang* untuk menghandle request yang dilakukan oleh user kemudian server melakukan query pada database lalu mengembalikan response berdasarkan request yang dilakukan dalam bentuk json untuk dikonsumsi oleh front end untuk ditampilkan dalam tampilan yang menarik menggunakan react js.



Gambar 3. 7 Stuktur Program Golang

Dalam Gambar 3.7, diilustrasikan struktur program yang mengadopsi Clean Architecture dalam pengembangan

menggunakan bahasa pemrograman Go. Struktur program ini dirancang untuk mencapai organisasi yang terstruktur, serta kemudahan dalam pemeliharaan dan pengembangan.

Di dalam direktori "app," terdapat subdirektori "controller," yang berperan sebagai lapisan pertama dalam menangani permintaan yang dikirimkan oleh pengguna. Fungsi utama dari "controller" adalah melakukan validasi terhadap data yang diterima dari pengguna, memverifikasi keabsahan kredensial yang diberikan, dan melakukan langkah-langkah persiapan awal sebelum memproses permintaan lebih lanjut.

Selanjutnya, terdapat subdirektori "service," yang bertugas sebagai lapisan penghubung antara "controller" dan "repository." Lapisan ini mengambil alih tanggung jawab untuk meneruskan permintaan yang telah divalidasi dari "controller" ke "repository" untuk melakukan kueri ke dalam basis data. Selain itu, "service" juga dapat mengimplementasikan logika bisnis tambahan yang diperlukan sebelum atau setelah interaksi dengan basis data.

Pada direktori "repository," interaksi dengan basis data terjadi. Lapisan ini bertanggung jawab atas pengambilan data dari basis data berdasarkan permintaan yang telah diteruskan dari "service." Hasil kueri kemudian dikirim kembali ke "service" untuk diproses lebih lanjut.

Setelah hasil kueri diproses dalam "service," respons akhir disusun dan dikirimkan kembali ke "controller." Dari "controller," respons ini diberikan kepada pengguna sebagai hasil akhir dari permintaan yang telah dilakukan.

Dalam struktur ini, penerapan Clean Architecture memungkinkan pemisahan tugas yang jelas antara lapisan-lapisan

yang berbeda. "Controller" menangani interaksi awal dengan pengguna dan validasi data, "service" mengendalikan logika bisnis dan interaksi dengan basis data, dan "repository" bertanggung jawab atas akses ke basis data. Pemisahan ini menyediakan fleksibilitas dalam mengelola perubahan dan mempermudah pengujian serta pemeliharaan komponen-komponen yang terlibat.

Folder "helper": Direktori ini berfungsi untuk menampung fungsi atau utilitas kecil yang dapat digunakan di berbagai bagian program. Fungsi-fungsi ini membantu dalam tugas-tugas umum, seperti pengelolaan waktu, manipulasi string, atau validasi khusus.

Folder "infrastructure": Di dalam direktori ini, biasanya terdapat kode yang berkaitan dengan konfigurasi dan pengaturan infrastruktur, seperti koneksi basis data, konfigurasi layanan eksternal, atau setup lingkungan.

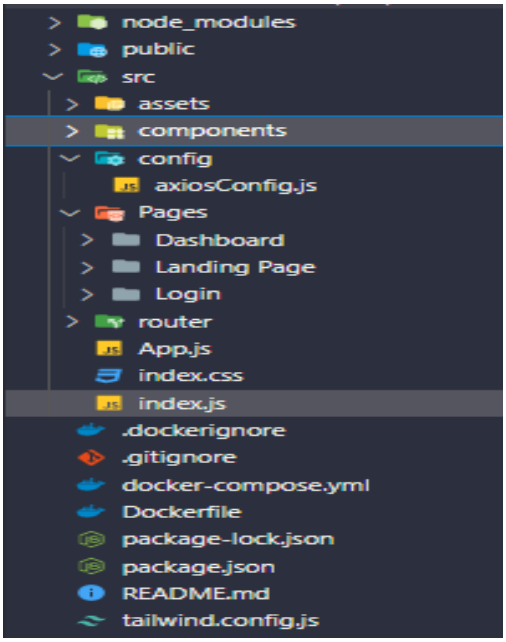
Folder "interfaces": Direktori ini mengandung definisi antarmuka (interfaces) yang menggambarkan bagaimana lapisan-lapisan dalam arsitektur berinteraksi satu sama lain. Interfaces ini berfungsi sebagai kontrak yang mendefinisikan metode atau fungsi yang harus diimplementasikan oleh komponen-komponen tertentu.

Folder "routes": Pada direktori ini, terletak definisi rute atau endpoint-endpoint yang akan ditangani oleh server. Ini berfungsi sebagai penghubung antara permintaan dari klien (pengguna) dan bagian-bagian dari sistem yang akan menangani permintaan tersebut.

File "main.go": Berkas ini adalah titik awal dari aplikasi. Di sini, Anda akan menemukan inisialisasi komponen-komponen penting seperti router, server HTTP, dan pengaturan awal lainnya. Berkas "main.go" juga akan menghubungkan komponen-

komponen yang diperlukan sesuai dengan prinsip Clean Architecture.

File "docker-compose.yml": Berkas ini adalah konfigurasi untuk Docker Compose, yang digunakan untuk mengatur lingkungan pengembangan yang terisolasi. Dalam berkas ini, akan mendefinisikan layanan-layanan yang diperlukan, seperti basis data, cache, atau layanan eksternal lainnya, serta pengaturan lingkungan yang relevan.



Gambar 3. 8 Struktur Program React Js

Dalam Dalam konteks Gambar 3.8, kita dapat mengamati struktur yang terorganisir dengan baik untuk proyek pengembangan aplikasi. Struktur ini memberikan kerangka yang jelas bagi komponen-komponen yang akan digunakan dalam pembangunan aplikasi web.

Di dalam proyek ini, "node_modules" adalah direktori yang berfungsi untuk menyimpan semua dependensi atau modul pihak

ketiga yang dibutuhkan oleh proyek. Modul-modul ini memiliki peran penting dalam mengintegrasikan fitur-fitur yang telah dihasilkan oleh komunitas pengembang, memperluas kemampuan aplikasi yang akan dibangun.

"Public" merupakan direktori yang menampung berkas-berkas yang dapat diakses langsung oleh pengguna. Ini mencakup berkas-berkas seperti halaman HTML, gambar, dan aset-aset statis lainnya yang diperlukan untuk merancang tampilan visual aplikasi.

Di dalam subfolder "assets," berbagai jenis aset seperti gambar, ikon, dan file multimedia lainnya dapat diakses dan dikelola. Dengan pemisahan ini, manajemen aset menjadi lebih teratur, mempermudah perawatan dan penyajian visual yang kohesif.

"Components" adalah subfolder yang berfungsi sebagai tempat penyimpanan komponen-komponen antarmuka pengguna yang dapat digunakan kembali. Dalam subfolder ini, komponen-komponen tersebut terpisah dari logika aplikasi, memudahkan perawatan dan pembaruan. Pemisahan ini juga mempromosikan penggunaan ulang yang efisien.

"Config" adalah subfolder yang berisi konfigurasi khusus yang relevan dengan aplikasi. Sebagai contoh, berkas "axiosConfig.js" digunakan untuk mengatur konfigurasi dalam penggunaan pustaka Axios, yang mengelola komunikasi HTTP. Pengaturan ini memastikan efektivitas dan konsistensi dalam komunikasi jaringan.

Selanjutnya, subfolder "dashboard" berfungsi sebagai wadah untuk komponen-komponen yang berkaitan dengan tampilan halaman dashboard. Dalam subfolder ini, informasi dan elemen-

elemen penting untuk tampilan dashboard ditempatkan, menciptakan tampilan yang informatif dan mudah digunakan.

Subfolder "landingpage" memuat komponen-komponen yang berfokus pada tampilan halaman landingpage. Di sini, desain dan elemen-elemen disusun untuk menyambut pengguna dengan tampilan awal yang menarik dan informatif.

Lalu, subfolder "login" berisi komponen-komponen yang terkait dengan tampilan halaman login. Dalam subfolder ini, pengguna dapat memasukkan kredensial mereka untuk mengakses aplikasi.

"Direktori router" memuat berkas-berkas yang bertanggung jawab atas pengaturan rute atau navigasi di dalam aplikasi. Pengaturan rute ini memungkinkan pengguna untuk berpindah antara halaman-halaman aplikasi dengan efisien.

berkas "index.js" memiliki peran sentral sebagai titik awal aplikasi. Di dalamnya, inisialisasi framework atau pustaka dilakukan, komponen-komponen terhubung, dan elemen tempat aplikasi akan di-render ditetapkan.

Dengan struktur yang rinci dan terorganisir seperti ini, proyek akan mendapatkan manfaat dari kode yang lebih mudah diatur, dipahami, dan dikelola selama proses pengembangan berlangsung.

3.3.2.4. *Cutover*

Akhir dari proses ini adalah pengujian dari sistem yang sudah selesai dibuat. Pengujian yang peneliti akan lakukan menggunakan *metode Blackbox Testing* dengan tujuan mengurangi potensi cacat sistem. Setelah lolos uji, masuk ke tahap deployment atau production.

3.3.3. Deployment

Tahap deployment merupakan fase krusial dalam siklus pengembangan perangkat lunak, di mana aplikasi atau sistem yang telah dikembangkan diuji, dikonfigurasi, dan diimplementasikan dalam lingkungan produksi. Pada tahap ini, proses pengiriman dan peluncuran aplikasi dilakukan secara resmi setelah melalui serangkaian uji coba dan persiapan yang matang. Tujuan utama dari tahap deployment adalah memastikan bahwa aplikasi siap untuk digunakan dengan performa optimal dan sesuai dengan kebutuhan pengguna serta tujuan bisnis. Proses ini dilaksanakan setelah website berhasil melewati uji coba black box testing. Selanjutnya, dilakukan deployment menggunakan virtual private server (VPS) dengan memanfaatkan teknologi Docker sebagai wadah (container).

3.3.4. Kesimpulan

Tahap terakhir dalam penelitian ini adalah penarikan kesimpulan dan juga saran atas tahap-tahap penelitian yang telah dilaksanakan sebelumnya.