**PROPOSAL**

**PERANCANGAN DAN PEMBUATAN PERANGKAT LUNAK**

**UNTUK BOOKING RESTAURANT SYSTEM**

**BERBASIS DESKTOP GUI**



**DISUSUN OLEH:**

**CHELSEA OKTAVIANY (2017103242)**

**MUTHIA FIDIASARI (2017103034)**

**RIZAL ABIDIN (2017103292)**

**PROGRAM STUDI INFORMATIKA**

**FAKULTAS INDUSTRI KREATIF**

**INSITUT TEKNOLOGI DAN BISNIS KALBIS**

**JAKARTA**

**2018**

**BAB 1**

**PENDAHULUAN**

**1.1 Latar Belakang**

Perkembangan Teknologi terus menunjukan sebuah perkembangan yang sangat pesat. Hal Tersebut ditunjukan dari munculnnya system untuk memudahkan setiap orang dalam melakukan sesuatu. Kegiatan tersebut meliputi kegiatan harian maupun kegiatan proses bisnis. Dampak dari sistem dapat menyebabkan setiap kegian lebih efektif tetapi setiap sistem memiliki fungsi tersendiri. Fungsi utama dari setiap sistem adalah memberikan sebuah solusi atas kendala kendala yang dihadapi oleh setiap orang baik individual maupun perusahaan untuk menjadi kegiatan yang lebih efektif dan efisien.

Di Indonesia, terdapat kegiatan yang belum dapat dikatan efektif dan efisien dengan perkembangan teknologi ini. Pada industri kuliner khususnya restoran dimana kegiatan proses bisnis dalam pemesanan masih menggunakan cara yang tradisional dan hanya dapat dilakukan di restoran. Jika, konsumen hanya ingin masuk ke restoran dan langsung menerima pesanan makanan yang sebelumnya telah di pesan tanpa harus menunggu lama atau sang konsumen ingin memesan sebuah ruangan dan tempat duduk sehingga di perlukannya sebuah aplikasi yang dapat memenuhi permintaan tersebut.

Maka dari itu, dibutuhkan suatu aplikasi yang membuat pelanggan dapat melakukan pemesanan sebelum datang ke restoran. Hal ini terutama akan sangat membantu untuk jenis restoran yang setiap waktunya memiliki antrian Panjang, memberikan pelanggan untuk menunggu adanya meja yang yang kosong. Aplikasi reservasi restoran menjadi cara yang paling popular untuk memenuhi permintaan pelanggan.

Dampak yang dihasilkan antara lai bagi pemilik restoran, yaitu dapat memaksimalkan proses bisnis yang terjadi di restoran karena telah meminimalisasi waktu tunggu pelanggan.

**1.2 Identifikasi Masalah**

Berdasarkan latar belakang yang dijelaskan di atas maka permasalahan yang timbul adalah:

1. Restoran yang selalu penuh atau memiliki antrian panjang sehingga sering kali terdapat pelanggan yang tidak mendapatkan tempat duduk.
2. Pelanggan tidak mendapatkan kepastian terkait informasi ketersediaan meja dan menu makanan yang kosong sebelum tiba direstoran.
3. Sering terdapat pelanggan yang menunggu lama untuk mendapatkan tempat duduk maupun memesan menu makanan.

**1.3 Rumusan Masalah**

Berdasarkan identifikasi masalah yang telah dijelaskan diatas maka perumusan masalah yang ada, antara lain:

1. Bagaimana membangun sebuah aplikasi *mobile* yang dapat membantu mengurangi antrian dari restoran?
2. Bagaimana agar pelanggan mendapatkan kepastian terkait informasi ketersediaan meja yang kosong sebelum tiba di restoran?
3. Bagaimana membangun aplikasi mobile yang dapat mengurangi waktu tunggu pelanggan yang digunakan untuk mendapat tempat duduk dan memesan menu makanan?

**1.4 Maksud dan Tujuan**

Penelitian ini dimaksudkan untuk membangun aplikasi yang dapat di akses kapanpun dan dimanapun, dengan tujuan:

1. Pelanggan dapat melakukan booking tempat sebelum datang ke restoran yang akan mempermudah proses pemesanan.
2. Mengurangi antrian dari restoran dan waktu tunggu pelanggan untuk memesan menu.

**1.5 Batasan Masalah**

Aplikasi Booking restoran ini mempunyai beberapa Batasan antara lain:

1. Aplikasi ini tidak akan dilengkapi dengan system pelaporan manajemen.
2. Aplikasi tidak disertai dengan pembayaran pemesanan menu makanan dan ruangan yang dilakukan oleh pelanggan.

**BAB 2**

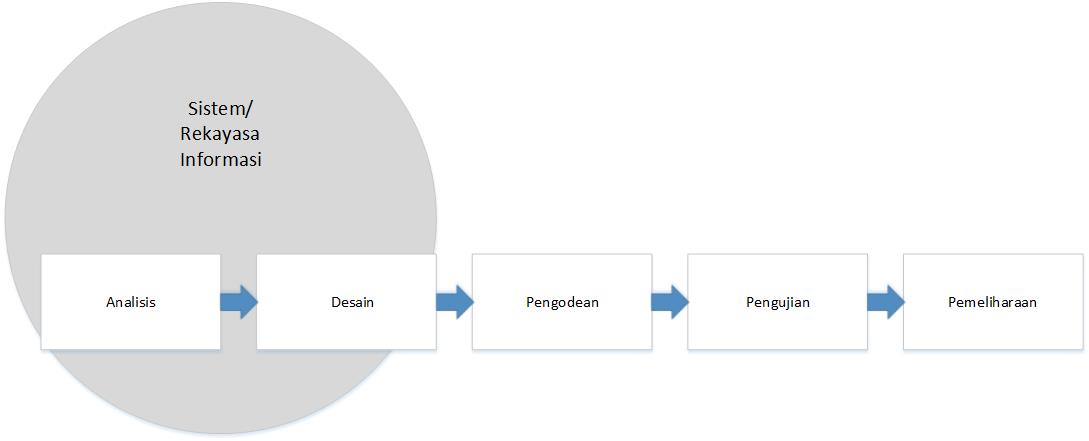
**Metodologi Pengembangan**

**2.1 Metodologi Pengembangan Perangkat Lunak**

Metode yang digunakan dalam pengembangan sistem *booking online* ini adalah metode *waterfall*. Alasan menggunakan metode ini adalah karena metode *waterfall* melakukan pendekatan secara sistematis dan berurutan dalam membangun suatu sistem. Proses metode waterfall yaitu pada pengerjaan dari suatu sistem dilakukan secara berurutan. Sistem yang dihasilkan akan berkualitas baik, dikarenakan pelaksanaannya secara bertahap sehingga tidak terfokus pada tahapan tertentu.

Model SDLC air terjun (*waterfall*) sering juga disebut model sekuensial liner atau alur hidup klasik (*classic life cycle*). Model air terjun menyediakan pendekatan alur hidup perangkat lunak secara sekuensial atau terurut dimulai dari analisis, desain, pengodean, pengujian, dan tahap pendukung (*support*).

Berikut adalah gambar model air terjun (*waterfall*):



Gambar 1.1 Illustrasi model *waterfall*

Pada tahun 1960-an, dan 1970-an, proyek pengembangan perangkat lunak merupakan pekerjaan yang sangat memakan biaya dan waktu karena pengembangan perangkat lunak ini difokuskan pada perencanaan dan pengendalian. *Model Waterfall*, dalam [rekayasa sistem](https://id.wikipedia.org/w/index.php?title=Rekayasa_sistem&action=edit&redlink=1) dan [rekayasa perangkat lunak](https://id.wikipedia.org/wiki/Rekayasa_perangkat_lunak), adalah untuk membantu mengatasi kerumitan yang terjadi akibat proyek-proyek pengembangan perangkat lunak.

Metode air terjun atau yang sering disebut metode waterfall sering dinamakan siklus hidup klasik (classic life cycle), dimana hal ini menggambarkan pendekatan yang sistematis dan juga berurutan pada pengembangan perangkat lunak, dimulai dengan spesifikasi kebutuhan pengguna lalu berlanjut melalui tahapan-tahapan perencanaan (planning), permodelan (modeling), konstruksi (construction), serta penyerahan sistem ke para pelanggan/pengguna (deployment), yang diakhiri dengan dukungan pada perangkat lunak lengkap yang dihasilkan (Pressman, 2012). Tahapan metode waterfall dapat dilihat pada gambar di bawah ini.

Kebutuhan

Sistem

Spesifikasi Kebutuhan

Perangkat Lunak

Desain Perangkat Lunak

Implementasi Perangkat Lunak

Uji Mesin Formal Integrasi Perangkat Lunak

Operasi dan Perawatan

**Gambar 1.2** Penyajian sederhana dari model pengembangan waterfall

Pada gambar 1.2, sebuah model *waterfall* memacu tim pengembang untuk memerinci apa yang seharusnya perangkat lunak lakukan sebelum sistem tersebut dikembangkan.

Model ini memungkinkan pemecahan misi pengembangan yang rumit menjadi beberapa langkah logis (desain, kode, pengujian, dan seterusnya) dengan beberapa langkah yang pada akhirnya akan menjadi produk akhir yang siap pakai. Pendekatan ini membuat perangkat lunak yang lebih besar, mudah diatur dan selesai tepat pada waktunya tanpa biaya yang berlebihan. Tahapan-tahapan kebutuhan sistem disebut analisis sistem, analisis, dan pengumpulan kebutuhan data konsumen, atau analisis kebutuhan pengguna. Tahap implementasi yang biasa disebut *code* dan *debug*; dan tahapan pengujian yang mencakup uji tingkat komponen, uji tingkat produk, dan uji tingkat sistem.

Metode *waterfall* memiliki beberapa tahapan yaitu:

* ***Requirement Analisis***

Tahap ini pengembang sistem diperlukan komunikasi yang bertujuan untuk memahami perangkat lunak yang diharapkan oleh pengguna dan batasan perangkat lunak tersebut. Informasi ini biasanya dapat diperoleh melalui wawancara, diskusi atau survei langsung. Informasi dianalisis untuk mendapatkan data yang dibutuhkan oleh pengguna.

* ***System Design***

Spesifikasi kebutuhan dari tahap sebelumnya akan dipelajari dalam fase ini dan desain sistem disiapkan. Desain Sistem membantu dalam menentukan perangkat keras *(hardware)* dan sistem persyaratan dan juga membantu dalam mendefinisikan arsitektur sistem secara keseluruhan.

* ***Implementation***

Pada tahap ini, sistem pertama kali dikembangkan di program kecil yang disebut *unit,* yang terintegrasi dalam tahap selanjutnya. Setiap *unit*dikembangkan dan diuji untuk fungsionalitas yang disebut sebagai *unit testing*.

* ***Integration & Testing***

Seluruh *unit* yang dikembangkan dalam tahap implementasi diintegrasikan ke dalam sistem setelah pengujian yang dilakukan masing-masing *unit*. Setelah integrasi seluruh sistem diuji untuk mengecek setiap kegagalan maupun kesalahan.

* ***Operation & Maintenance***

Tahap akhir dalam model *waterfall*. Perangkat lunak yang sudah jadi, dijalankan serta dilakukan pemeliharaan. Pemeliharaan termasuk dalam memperbaiki kesalahan yang tidak ditemukan pada langkah sebelumnya. Perbaikan implementasi *unit* sistem dan peningkatan jasa sistem sebagai kebutuhan baru.

**2.2 Analisis Kebutuhan Perangkat Lunak**

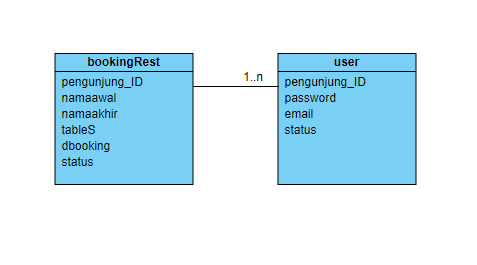
Proses pengumpulan kebutuhan dilakukan secara intensif untuk mespesifikasikan kebutuhan perangkat lunak agar dapat dipahami perangkat lunak seperti apa yang dibutuhkan oleh *user*. Spesifikasi kebutuhan perangkat lunak pada tahap ini perlu untuk didokumentasikan.

Pemodelan UML atau *Unified Modeling Language* adalah bahasa pemodelan untuk membangun perangkat lunak yang dibangun dengan menggunakan teknik pemrograman berorientasi objek.

Dalam pemodelan UML ini kami menggunakan 3 model diagram, yaitu: *Class Diagram, Use case Diagram,* dan *Activity Diagram.*

**2.2.1 Class Diagram**

*Class Diagram,* diagram ini terdiri dari *class*, *interface*, *association*, dan *collaboration*. Diagram ini menggambarkan objek-objek yang ada di sistem.



Gambar 2.1 Class Diagram *Booking Restaurant System*

**2.2.2 Use Case Scenario**

Use case Skenario: Login

Aktor: Admin

Skenario:

|  |  |
| --- | --- |
| Aksi Aktor | Reaksi Sistem |
| Skenario Normal | |
| 1. Memasukan pengunjung ID dan password |  |
|  | 1. Memeriksa valid tidaknya data yang dimasukan dengan memeriksa basis data, pengunjung |
|  | 1. Masuk ke halaman utama aplikasi *booking restaurant system* sesuai dengan jenis pengguna (admin atau pengunjung). |
| Skenario Alternatif | |
| 1. Memasukan pengunjung ID dan password |  |
|  | 1. Memeriksa valid tidaknya data yang dimasukan dengan memeriksa basis data, admin atau pengunjung. |
|  | 1. Menampilkan pesan login tidak valid. |
| 1. Memasukan pengunjung ID dan password yang valid. |  |
|  | 1. Memeriksa valid tidaknya data yang dimasukan dengan memeriksa basis data, admin atau pengunjung. |
|  | 1. Masuk ke halaman utama aplikasi *booking restaurant sytem* sesuai dengan jenis pengguna (admin atau pengunjung). |

Use case Skenario: Memeriksa Login

Aktor: Admin

Skenario:

|  |  |
| --- | --- |
| Aksi Aktor | Reaksi Sistem |
| Skenario Normal | |
|  | 1. Memeriksa ke variable session sebagai penanda login apakah pengguna sudah login. |
|  | 1. Mengembalikan status login, sudah login atau belum. |

Use case Skenario: Menambah Data Pengunjung

Aktor: Admin

Skenario:

|  |  |
| --- | --- |
| Aksi Aktor | Reaksi Sistem |
| Skenario Normal | |
|  | 1. Memeriksa status login. |
| 1. Memasukan data pengunjung pada form yang sudah disediakan. |  |
|  | 1. Memeriksa valid atau tidaknya data masukan. |
|  | 1. Menyimpan data pengunjung ke basis data pengunjung. |
|  | 1. Menampilkan pesan sukses disimpan. |
| Skenario Alternatif | |
|  | 1. Memeriksa status login |
| 1. Memasukan data pengunjung pada form yang sudah disediakan. |  |
|  | 1. Memeriksa valid atau tidaknya data masukan. |
|  | 1. Mengeluarkan pesan bahwa data masukan tidak valid. |
| 1. Memperbaiki data masukan. |  |
|  | 1. Memeriksa valid atau tidaknya data masukan |
|  | 1. Menyimpan data pengunjung ke basis data pengunjung. |
|  | 1. Menampilkan pesan sukses disimpan. |

Use case Skenario: Mengubah Data Pengunjung

Aktor: Admin

Skenario:

|  |  |
| --- | --- |
| Aksi Aktor | Reaksi Sistem |
| Skenario Normal | |
|  | 1. Memeriksa Status Login. |
| 1. Memasukan pengunjung ID dan data pengunjung pada form yang disediakan. |  |
|  | 1. Mencari pengunjung ID yang akan diubah menurut data kunci. |
|  | 1. Memeriksa valid tidaknya data masukan. |
|  | 1. Mengubah data pengunjung sesuai dengan data masukan. |
|  | 1. Menyimpan data yang sudah diubah ke basis data pengunjung. |
|  | 1. Menampilkan pesan bahwa data telah sukses diubah dan disimpan. |
| Skenario Alternatif | |
|  | 1. Memeriksa Status Login. |
| 1. Memasukan pengunjung dan data pengunjung pada form yang disediakan. |  |
|  | 1. Mencari pengunjung ID yang akan diubah menurut data kunci. |
|  | 1. Memeriksa valid tidaknya data masukan. |
|  | 1. Menampilkan pesan bahwa data masukan tidak valid |
| 1. Memperbaiki data masukan. |  |
|  | 1. Mencari pengunjung ID yang akan diubah menurut data kunci. |
|  | 1. Memeriksa valid tidaknya data masukan. |
|  | 1. Mengubah pengunjung ID sesuai dengan data masukan. |
|  | 1. Menyimpan data yang sudah diubah ke basis data pengunjung ID. |
|  | 1. Menampilkan pesan bahwa data telah sukses diubah dan disimpan. |

Use case Skenario: Menghapus Data Pengunjung

Aktor: Admin

Skenario:

|  |  |
| --- | --- |
| Aksi Aktor | Reaksi Sistem |
| Skenario Normal | |
|  | 1. Memeriksa Status Login. |
| 1. Memasukan pengunjung ID dan memilih hapus. |  |
|  | 1. Mencari pengunjung ID yang akan dihapus menurut data kunci. |
|  | 1. Memeriksa valid tidaknya data masukan. |
|  | 1. Menghapus data dosen. |
|  | 1. Menampilkan pesan bahwa data telah sukses dihapus. |
| Skenario Alternatif | |
|  | 1. Memeriksa Status Login. |
| 1. Memasukan pengunjung ID dan memilih hapus. |  |
|  | 1. Mencari pengunjung ID yang akan dihapus menurut data kunci. |
|  | 1. Memeriksa valid tidaknya data masukan. |
|  | 1. Menampilkan pesan bahwa data masukan tidak valid |
| 1. Memperbaiki data masukan. |  |
|  | 1. Mencari pengunjung ID yang akan dihapus menurut data kunci. |
|  | 1. Memeriksa valid tidaknya data masukan. |
|  | 1. Menghapus data pengunjung ID. |
|  | 1. Menampilkan pesan bahwa data telah sukses dihapus. |

Use case Skenario: Melihat Data Pengunjung

Aktor: Admin

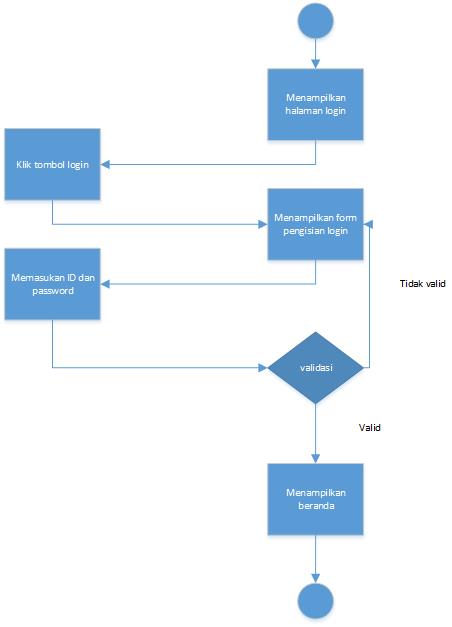
Skenario:

|  |  |
| --- | --- |
| Aksi Aktor | Reaksi Sistem |
| Skenario Normal | |
|  | 1. Memeriksa Status login. |
|  | 1. Menampilkan data pengunjung seluruhnya dalam bentuk tabel. |

**2.2.3 Activity Diagram**

Activity diagram, sesuai dengan namanya diagram ini menggambarkan tentang aktivitas yang terjadi pada sistem. Dari pertama sampai akhir, diagram ini menunjukkan langkah-langkah dalam proses kerja sistem yang kita buat. Sebagai contoh, langkah-langkah memasak air. Tetapi kita akan menjelaskan dengan bentuk grafik. Struktur diagram ini juga mirip dengan flowchart.

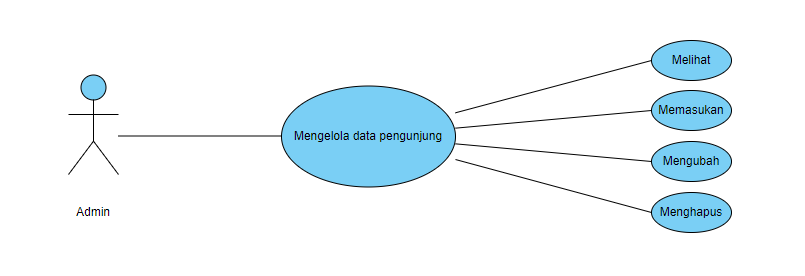
Flowchart Proses Login:



Gambar 3.1 *Flowchart login*

**2.2.4 Use Case Diagram**

Use Case Diagram adalah gambaran graphical dari beberapa atau semua actor, use case, dan interaksi diantaranya yang memperkenalkan suatu sistem. Use case diagram tidak menjelaskan secara detail tentang penggunaan use case, tetapi hanya memberi gambaran singkat hubungan antara use case, actor, dan sistem. Di dalam use case ini akan diketahui fungsi-fungsi apa saja yang berada pada sistem yang dibuat.



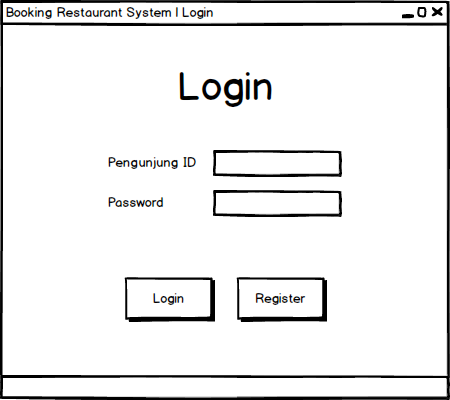
Gambar 4.1 *Use case diagram*

Pada gambar 4.1 mendefinisikan aktor apa saja yang terlibat dalam penggunaan aplikasi. Dari sisi pengguna ada Admin, dan pengunjung dimana admin memiliki hak akses untuk melihat, memasukan, mengubah, dan menghapus data pada tabel data pengunjung. Admin dan pengunjung hanya memiliki hak untuk melihat tabel booking setelah booking telah dipenuhi.

**2.8 Desain Perangkat Lunak**

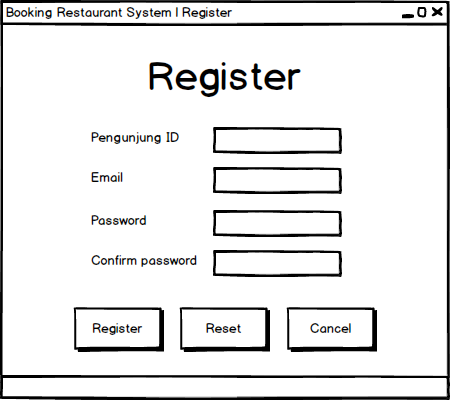
Desain perangkat lunak *Booking Restaurant System* ini adalah:

1. **Desain Antarmuka admin**



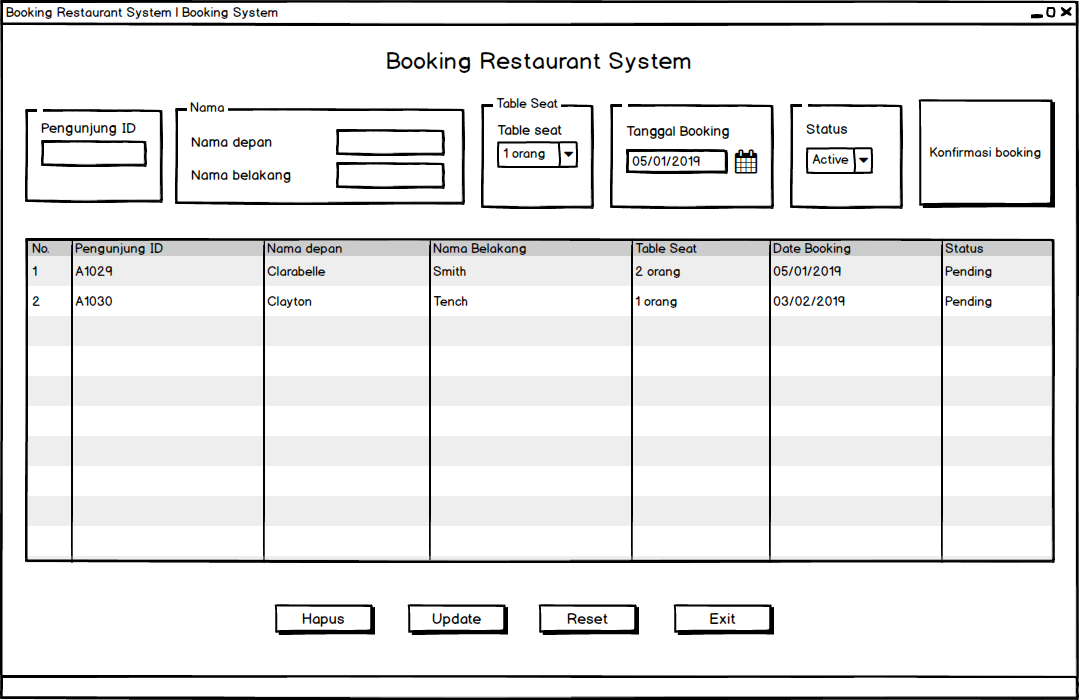
Gambar 1 Form *login*

Pada gambar 1 merupakan halaman antarmuka *login* untuk admin, dimana *admin* perlu mengisi pengunjung ID dan *password* yang telah ditentukan sebelumnya agar bias mengakses aplikasi.



Gambar 2 Form *register*

Pada gambar 2 merupakan antarmuka register untuk melakukan pendaftaran sebelum mengakses aplikasi *Booking Restaurant System*.



Gambar 3 Desain aplikasi *Booking Restaurant System*

Pada gambar 3 merupakan halaman admin untuk mengelola data pengunjung, dimana admin bias melakukan operasi menambah, mengubah, dan menghapus data pengunjung dengan tombol-tombol yang ditampilkan melalui mengisi *form data* yang telah disediakan. Halaman ini juga menampilkan daftar yang telah disediakan. Halaman ini juga menampilkan daftar *data* pengunjung dalam bentuk tabel yang sudah dimasukan oleh *admin*.

**2.9 Pembuatan Kode Program**

Desain harus ditranslasikan ke dalam program perangkat lunak. Hasil dari tahap ini adalah program komputer sesuai dengan desain yang telah dibuat pada tahap desain.

**2.10 Pengujian**

Pengujian berfokus pada perangkat lunak secara dari segi *logic* dan fungsional dan memastikan bahwa semua bagian sudah diuji. Hal ini dilakukan untuk mengurangi kesalahan (*error*) dan memastikan keluaran yang dihasilkan sesuai dengan diinginkan.

**2.11 Pemeliharaan**

Tidak menutup kemungkinan sebuah perangkat lunak mengalami perubahan ketika sudah dikirimkan ke *user*. Perubahan bisa terjadi karena adanya kesalahan yang muncul dan tidak terdeteksi saat pengujian atau perangkat lunak harus beradaptasi.

**DAFTAR PUSTAKA**

|  |  |
| --- | --- |
| [1] | codepolitan. [Online]. Available: https://www.codepolitan.com/mengenal-uml-diagram-use-case#. [Accessed 22 Desember 2018]. |
| [2] | J. Simamarta, "Rekayasa Perangkat Lunak," Penerbit Andi, Yogyakarta, 2010. |