BAB 3
PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Organisasi

Ketika melakukan pelaksanaan kerja magang, penulis memiliki peranan sebagai *software developer fullstack* secara *freelance*. Kerja magang dilakukan secara *work from home* atau bekerja dari rumah dikarenakan masa pandemi. Kerja magang dibimbing oleh Ibu Mariani Uli Sitompul selaku manajer keuangan di UD Lumba-Lumba. Pengerjaan proyek dilakukan dengan proses wawancara terhadap manajer keuangan dan diskusi untuk proses digitalisasi. Proses komunikasi dilakukan melalui *Whatsapp* yang bertujuan untuk memberikan *update* proyek secara berkala yaitu selama seminggu sekali. Selain memberikan *update*, diskusi seminggu sekali ini juga bertujuan untuk memberikan pengajaran cara pakai aplikasi hasil digitalisasi tersebut.

3.2 Tugas yang Dilakukan

Tugas yang dilakukan selama pelaksanaan kerja magang di UD Lumba-Lumba adalah melakukan proses digitalisasi keuangan dengan cara membangun sistem *invoice* yang berbasis *web*. Pengerjaan proyek dilakukan dengan *ReactJS* sebagai *framework* utama. Komponen seperti *bootstrap* juga digunakan untuk mengembangkan aplikasi dan membangun *front-end*. Selain itu, proyek ini juga memanfaatkan *firebase* untuk menyimpan data secara daring.

Setelah itu, beberapa tanggung jawab yang diberikan selama kegiatan kerja magang adalah sebagai berikut.

1. Mempelajari *framework ReactJS* dan *bootstrap*

2. Wawancara manajer keuangan untuk mendapatkan *User Requirement* yang

dibutuhkan untuk membangun *website*

3. Sosialisasi proyek sebagai tahap awal melakukan digitalisasi bagian keuangan

3.3 Uraian Pelaksanaan Magang

Dalam menguraikan pelaksanaan kerja magang di UD Lumba-Lumba, akan dijelaskan aktivitas yang dilakukan disaat magang, *framework* yang digunakan dalam proses magang, perancangan aplikasi, hingga implementasi sistem *invoice generator*.

3.3.1 Aktivitas Kerja Magang

Pelaksanaan kerja magang dilaksanakan selama sembilan minggu seperti yang diuraikan pada Tabel 3.1.

Tabel 3.1. Pekerjaan yang dilakukan tiap minggu selama pelaksanaan kerja magang

Minggu Ke -	Pekerjaan yang dilakukan
1	Mempelajari ReactJS, Instalasi node js pada komputer
2	Pembuatan Komponen Layout/UI yang diperlukan pada proyek, Wawancara dengan manajer keuangan untuk mendapatkan *user requirement*
3	Penambahan *navigation bar*, Penambahan proses *add customer*, penambahan *add itema*, dan *upload firebase*
4	Wawancara terkait keperluan *invoice*, Penambahan fitur *edit* dan *delete* pada halaman *customer* dan *item*
5	Pembuatan *invoice* dengan desain tabel
6	Mengganti desain tabel *invoice* dari tabel menjadi desain *e-commerce* ketika memilih *item* untuk aksesibilitas, Melakukan *debugging* dan *bug fixing*
7	Menambahkan fitur menampilkan *list* dari *invoice* berdasarkan *customer* sesuai dengan permintaan
8	Integrasi fitur tampilan *invoice* agar bisa diunduh sebagai *file pdf*
9	Sosialisasi cara pemakaian *invoice* dan *bug fixing*

3.3.2 Framework yang Digunakan

Pembuatan aplikasi *invoice* menggunakan *ReactJS* sebagai *framework* utama dari pembangunan aplikasi. Komponen *front-end* menggunakan *bootstrap* dan kombinasi ikon dari *react icon* untuk menampilkan beberapa bagian menu. Selain itu, pembuatan aplikasi juga dibantu oleh *package* seperti *yarn*, penggunaan *library* untuk mengubah halaman *html* menjadi berbentuk *pdf file*, dan penggunaan *Ant Design React* untuk membangun *layout* dari *invoice*.

3.3.3 Perancangan dan Implementasi Sistem

Perancangan dan implementasi sistem akan ditampilkan melalui *user requirements* yang didapatkan dari hasil wawancara, *entity relationship diagram* dari *invoice generator*, *data flow diagram* untuk menjelaskan aliran data yang digunakan dalam sistem, lalu yang terakhir adalah *screenshot* langsung dari aplikasi yang juga disertai dengan *flowchart* pendukung.

A User Requirements

User requirements adalah ekspektasi dari *klien* ketika aplikasi dijalankan. *User requirements* biasanya ditulis ketika mendiskusikan *use case* dari suatu proyek. Definisi dari kebutuhan ini dikerjakan oleh *customer* atau *product manager* yang mengerti atau akan menggunakan sistem tersebut (Kraeling dan Tania, 2019). Dalam kasus ini, penulis mendapatkan *user requirements* dari manajer keuangan yang akan menggunakan aplikasi ini. Berikut merupakan rangkuman dari *user requirements*.

A.1 Halaman Utama

Berikut adalah *requirements* untuk halaman utama atau *dashboard*:

1. Menampilkan Hari dan Tanggal komputer di Halaman Depan
2. Menampilkan *navigation bar* di sebelah kiri untuk berpindah-pindah halaman
3. Menampilkan tabel berisi *customer* yang ketika diklik akan memunculkan *list invoice* dari *customer* tersebut
4. *Invoice* bisa diunduh sebagai *pdf file*

A.2 Manage Item

Berikut adalah *requirements* untuk halaman *manage item*:

1. Menampilkan tombol *add item* untuk menambahkan data
2. *Form* untuk mengisi proses *add item* ditampilkan dalam *modal*
3. Menampilkan tabel yang berisi semua *item* beserta harganya
4. Di dalam tabel yang bersisi *item*, ditampilkan pula tombol *edit* dan *delete* yang keduanya menggunakan *modal*

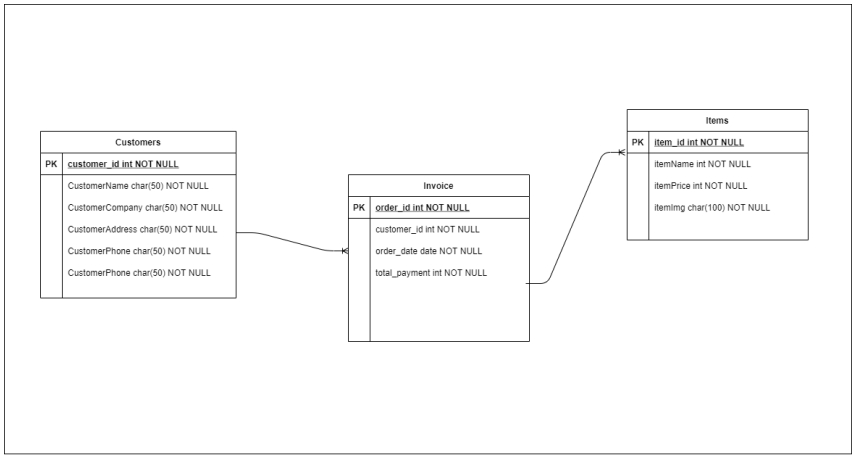
A.3 Manage Customer

Berikut adalah *requirements* untuk halaman *manage customer*:

1. Menampilkan tombol *add customer* untuk menambahkan data
2. *Form* untuk mengisi proses *add customer* ditampilkan dalam *modal*
3. Menampilkan tabel yang berisi semua *customer*
4. Didalam tabel yang bersisi *customer*, ditampilkan pula tombol *edit* dan *delete* yang keduanya menggunakan *modal*

B Entity Relationship Diagram

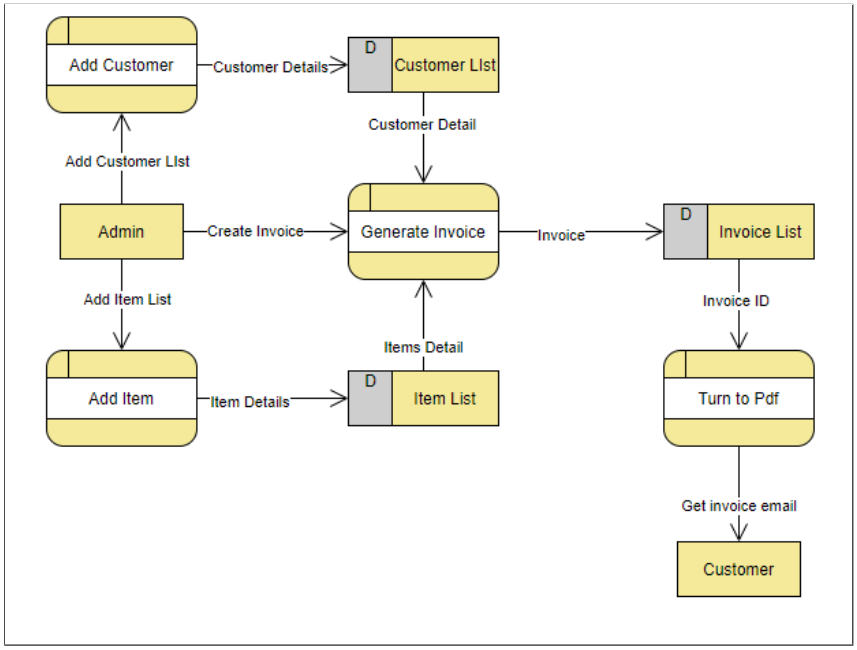
Entitas yang ada pada aplikasi adalah *customer*, *item*, dan *invoice*. Dapat dilihat pada Gambar 3.1 relasi-relasi antara ketiga entitas tersebut. Entitas *invoice* mempunyai relasi *one to many* terhadap *item* karena memungkinkan bagi sebuah *invoice* untuk memiliki beberapa *item*. *Customer* juga memiliki relasi *one to many* dengan *invoice* karena memungkinkan bagi satu *customer* untuk mendapatkan beberapa *invoice* sekaligus.



Gambar 3.1. *Entity Relationship Diagram* dari Aplikasi *Invoice Generator*

C Data Flow Diagram

Proses aliran data dari sistem *Invoice Generator* dapat ditinjau pada Gambar 3.2. Seorang *admin* bisa melakukan tiga buah proses. Proses pertama adalah *add item* untuk menambahkan barang yang akan disimpan pada *data store item list*. Setelah itu, ada proses *add customer* untuk menambahkan *customer* yang nantinya akan disimpan pada *data store customer list*. Lalu, proses yang terakhir adalah *generate invoice* untuk membuat *invoice* sesuai dengan *customer* dan *item* yang tersimpan. Setelah melakukan pembuatan *invoice*, *invoice* tersebut disimpan dalam *data store* dan jika dibutuhkan bisa dikeluarkan dalam bentuk *pdf files* lalu dikirimkan kepada *customer*.



Gambar 3.2. *Data Flow Diagram* dari Aplikasi *Invoice Generator*

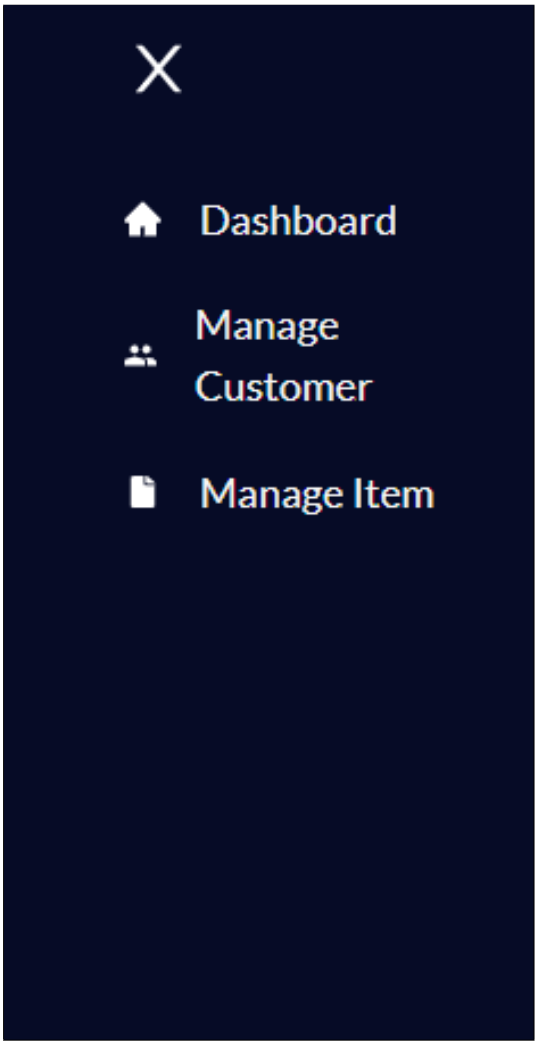
D Implementasi Sistem

Implementasi sistem dilakukan dengan menampilkan *flowchart* dan *screen-shot* dari aplikasi yang telah dibuat. *Flowchart* akan menampilkan proses logika dan alur kerja dari aplikasi yang telah dibuat mulai dari *navigation bar* hingga menyimpan data ke *firebase*. *Flowchart* ini juga akan dilengkapi dengan *screenshot* aplikasi yang mendukung.

D.1 Halaman Utama

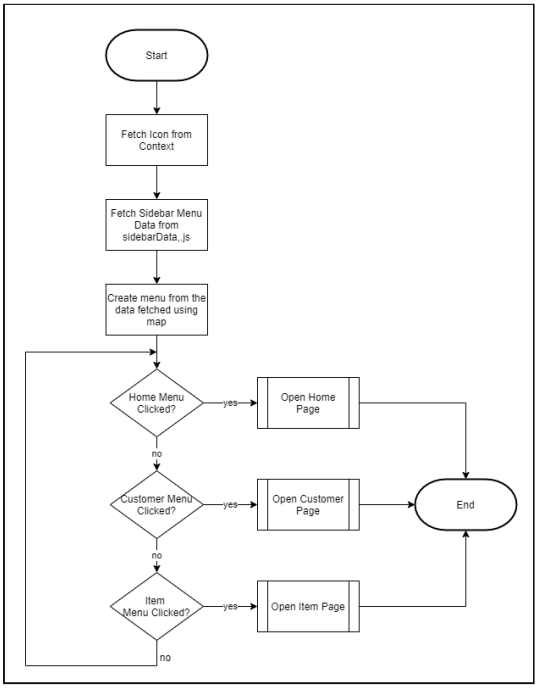
Halaman utama adalah halaman yang akan pertama kali tampak ketika *user* membuka aplikasi. Di dalam halaman ini, terdapat beberapa hal. Pertama, ada *navigation bar* yang bisa di munculkan dengan menekan *icon* pada pojok kiri atas dan dapat digunakan untuk berpindah-pindah halaman. Nantinya, *navigation bar* ini akan muncul di semua halaman. Gambar 3.3 menampilkan *navigation bar* yang belum diklik, dan Gambar 3.4 menampilkan *navigation bar* yang sudah diklik dan menampilkan menu untuk menuju ke halaman lain.

Gambar 3.3. Tampilan *Navigation Bar* sebelum diklik



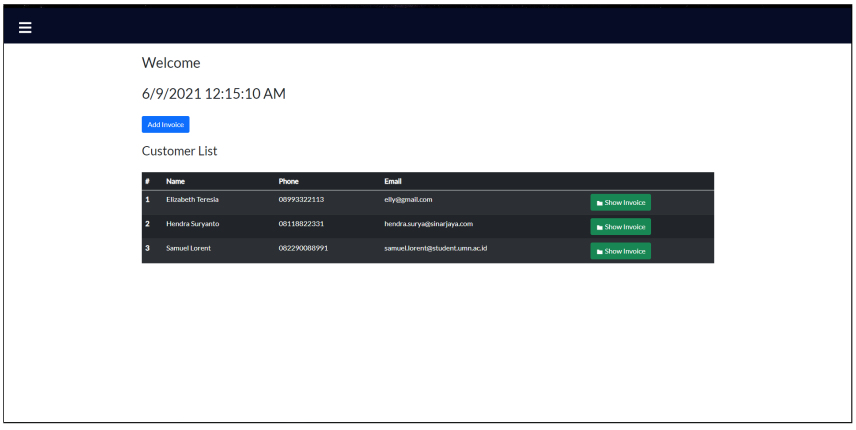
Gambar 3.4. Tampilan *Navigation Bar* sesudah diklik

Flowchart dari proses *navbar* ini dapat dilihat pada Gambar 3.5, di mana dapat terlihat proses perpindahan halaman dari suatu halaman ke halaman yang lain.

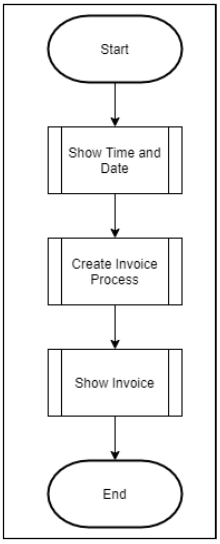


Gambar 3.5. Tampilan *flowchart* dari proses perpindahan *navigation bar*

Gambar 3.6 adalah tampilan dari *dashboard*. Disini dapat terlihat ada suatu elemen yang berisi tanggal, waktu dan elemen yang berupa *button* untuk menambahkan *invoice*. Selain itu, terdapat daftar *customer* yang bisa dicek *invoice* apa saja yang ada pada *customer* tersebut.

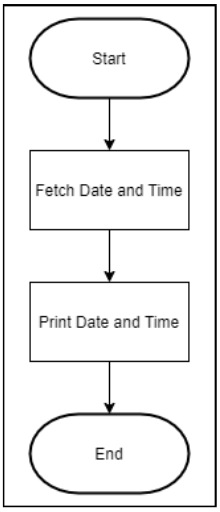


Gambar 3.6. Tampilan dari *homepage* atau *dashboard*

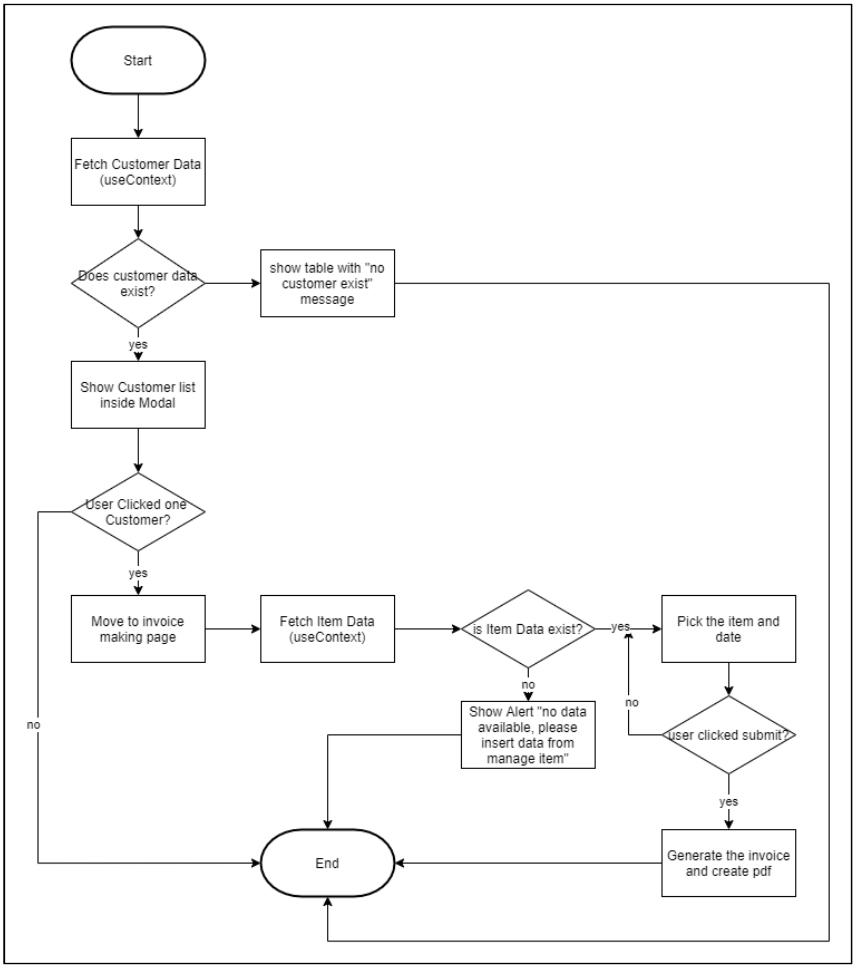


Gambar 3.7. Tampilan dari *flowchart homepage*

Gambar 3.7 merupakan *flowchart* utama dari halaman *dashboard*. Proses *show time and date* digunakan untuk menampilkan tanggal dan waktu pada *homepage*. Proses pengammbilan tanggal ini cukup sederhana karena variabel tanggal dan waktu langsung diambil dari waktu lokal komputer. Gambar 3.8 berikut menampilkan *flowchart* dari proses tersebut.

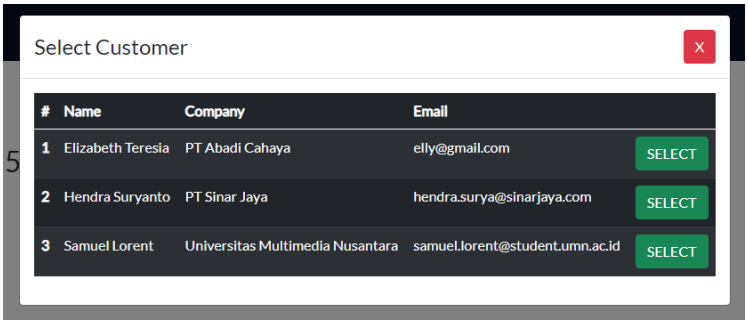


Gambar 3.8. Tampilan dari *flowchart* pada proses *show time and date*



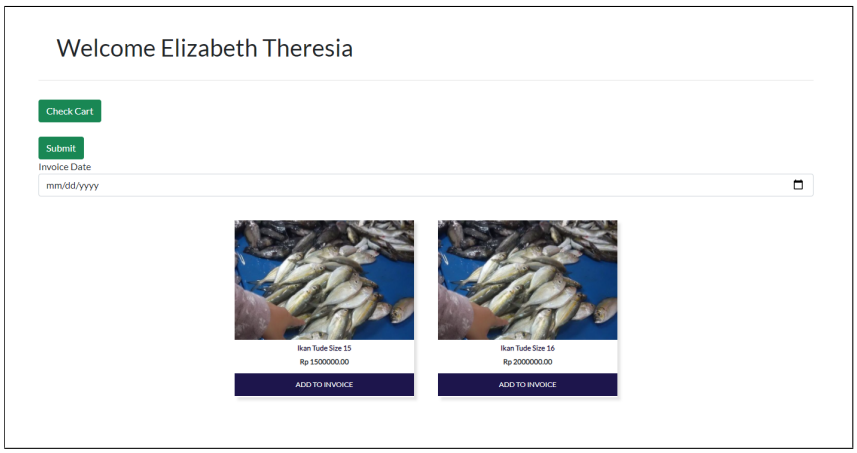
Gambar 3.9. Tampilan dari *flowchart* pada proses pembuatan *invoice*

Selanjutnya, ada proses pembuatan *invoice* yang dapat dilihat pada Gambar 3.9. Gambar ini menjelaskan alur dari komputer yang pertama-tama mengambil data *customer* dari sebuah *context*, lalu daftar *customer* ditampilkan dalam suatu *modal* seperti yang terlihat pada Gambar 3.10.



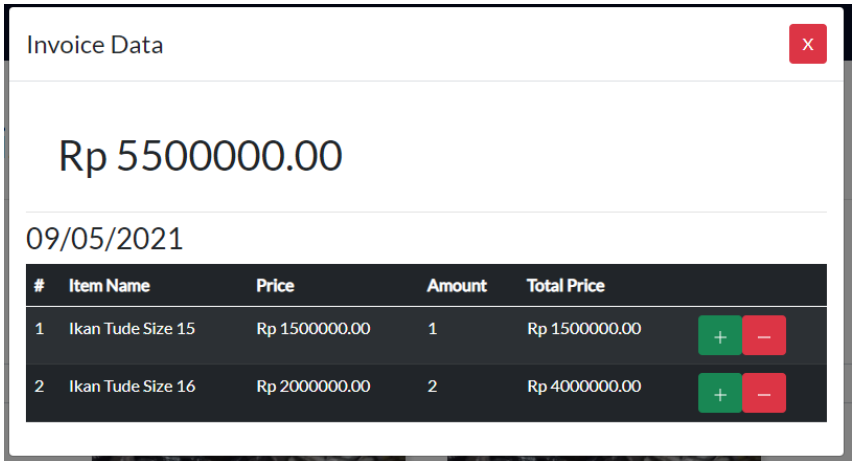
Gambar 3.10. Tampilan dari *modal*

Setelah itu, barulah ada proses lanjutan di mana terjadi perpindahan halaman ke halaman baru berisikan gambar produk dan proses *input* tanggal *invoice*. Pada halaman ini, *user* bisa memilih *item* yang ditampilkan dengan gambar *item* tersebut untuk di *input* ke dalam *invoice* dengan cara menekan tombol *add to invoice* yang berada di bawah masing-masing *item*. Terdapat juga dua *button*, yang atas untuk mengecek barang apa saja yang masuk ke dalam *invoice*, lalu tombol kedua untuk memasukkan data ke dalam *invoice* yang akan segera dibuat menjadi *pdf file*. Gambar dari halaman tersebut dapat dilihat pada Gambar 3.11.



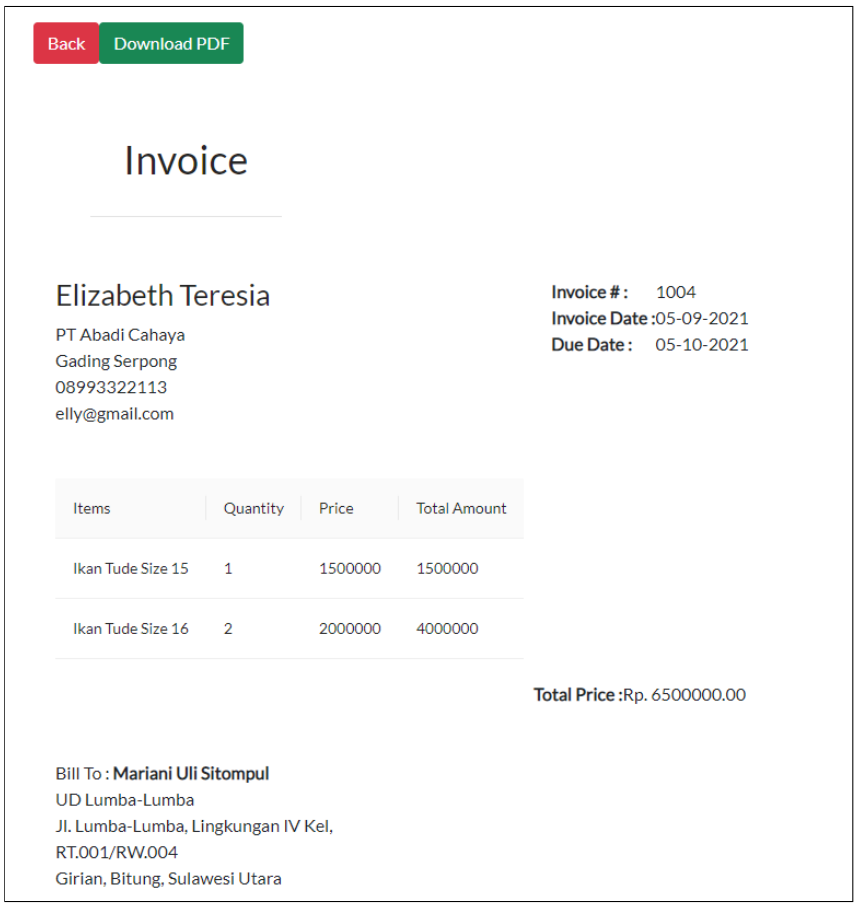
Gambar 3.11. Tampilan dari Proses Pemilihan *item*

Ketika tombol pertama diklik, maka akan muncul semacam data *invoice* di mana data tersebut berisi produk yang dimasukkan beserta jumlah total dari *invoice* tersebut. Gambar yang menunjukkan hasil dari proses tersebut dapat terlihat pada Gambar 3.12.



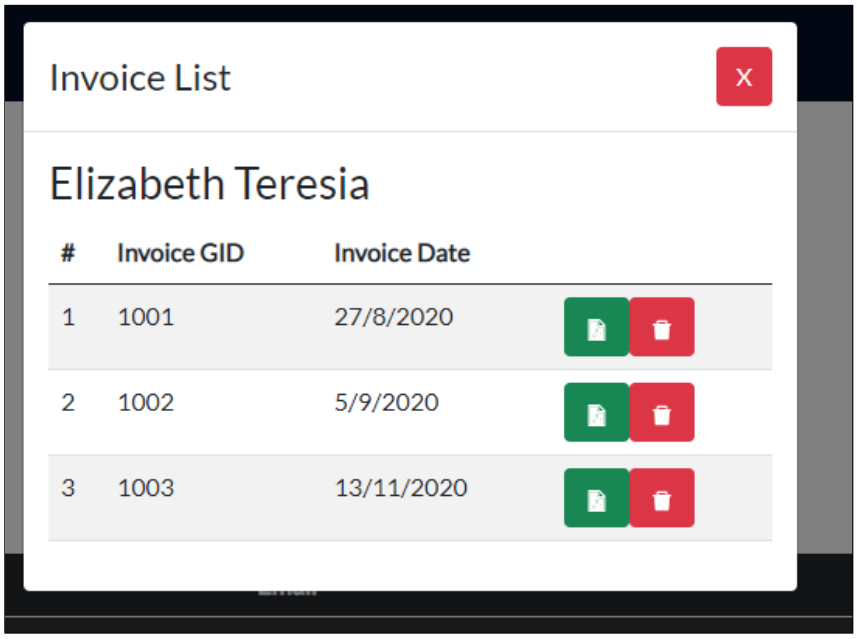
Gambar 3.12. Tampilan dari *invoice data*

Jika tombol *submit* ditekan, program akan langsung berpindah halaman ke halaman *invoice* dan program akan langsung memprosesnya menjadi *file pdf* lalu menyimpannya agar dapat diunduh kembali. Halaman *pdf* dapat dilihat pada Gambar 3.13.



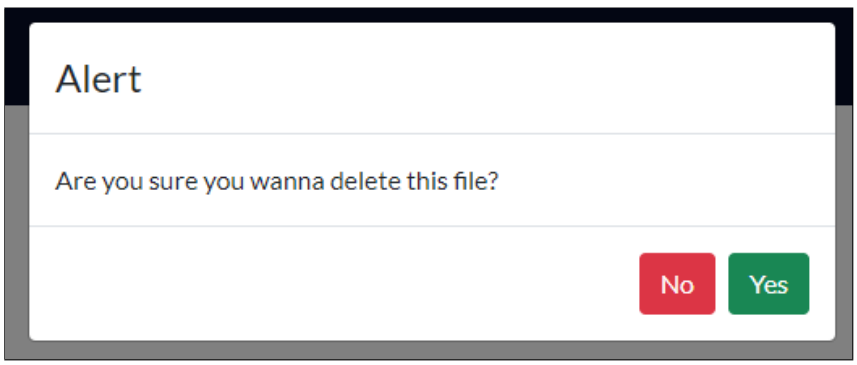
Gambar 3.13. Tampilan dari *Invoice*

Proses terakhir yang dapat dilakukan pada halaman utama ini adalah *delete invoice* atau menghapus *invoice* yang ada. Selain menghapus, *user* juga dapat mengunduh ulang *invoice* tersebut dengan bentuk *pdf file*. Gambar 3.14 memperlihatkan *invoice list* tersebut.



Gambar 3.14. Tampilan dari *invoice list*

Jika tombol *delete* ditekan, maka akan muncul suatu modal yang mengkonfirmasikan penghapusan. Jika *user* mengkonfirmasinya, maka akan muncul juga *alert* yang menyatakan bahwa data tersebut sudah dihapus. Konfirmasi penghapusan dapat dilihat pada Gambar 3.15, lalu *alert* untuk penghapusan dapat dilihat pada Gambar 3.16.

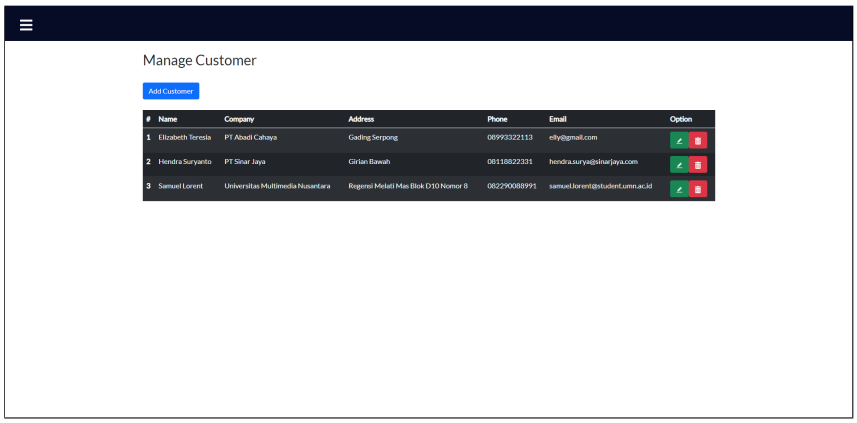


Gambar 3.15. Tampilan dari konfirmasi penghapusan

Gambar 3.16. Tampilan dari *alert* penghapusan

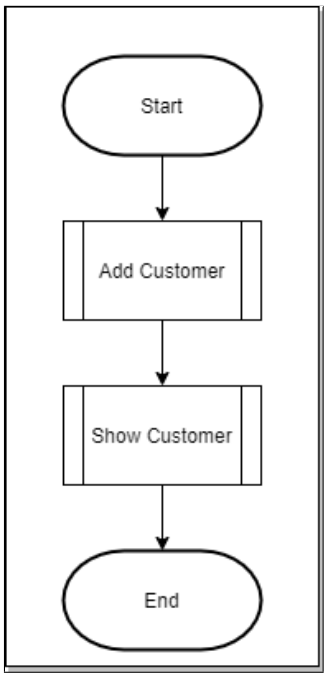
D.2 Halaman Customer

Halaman *customer* terlihat seperti Gambar 3.17 di mana terdapat sebuah *button* untuk menambahkan *customer* dan tabel yang menampilkan daftar *customer*. Didalam tabel tersebut juga ada *button* yang berguna untuk menghapus atau mengubah data pada tabel.



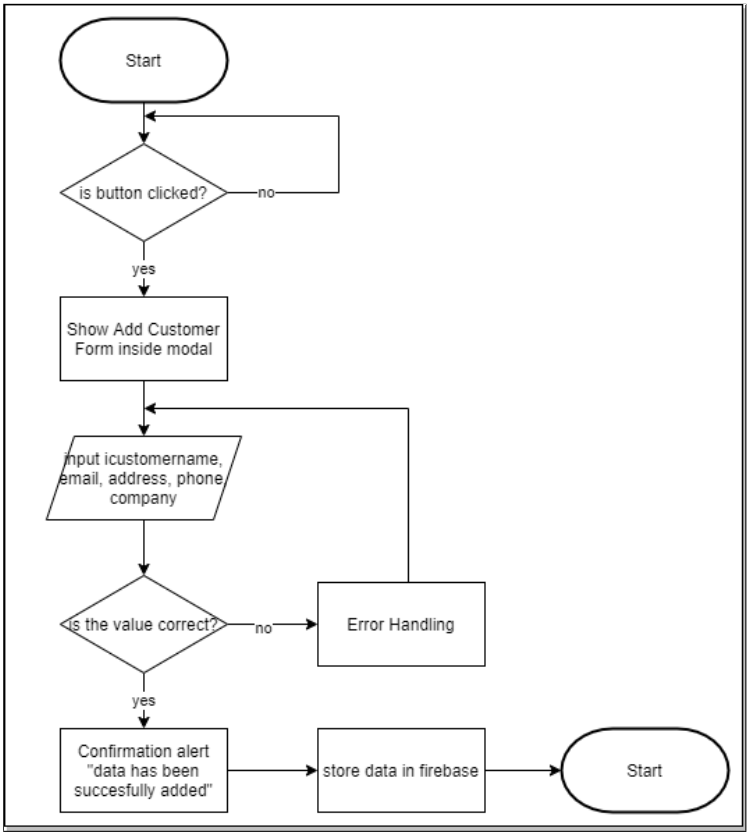
Gambar 3.17. Tampilan dari Halaman *Customer*

Gambar 3.18 merupakan *flowchart* utama dari halaman *customer* di mana terdapat dua proses utama, yaitu proses *add customer* yang akan menampilkan *modal* untuk menambahkan data *customer*, lalu *show customer* untuk menampilkan data *customer* dalam bentuk tabel dan juga bisa mengubah data dan menghapus data dari *customer*.

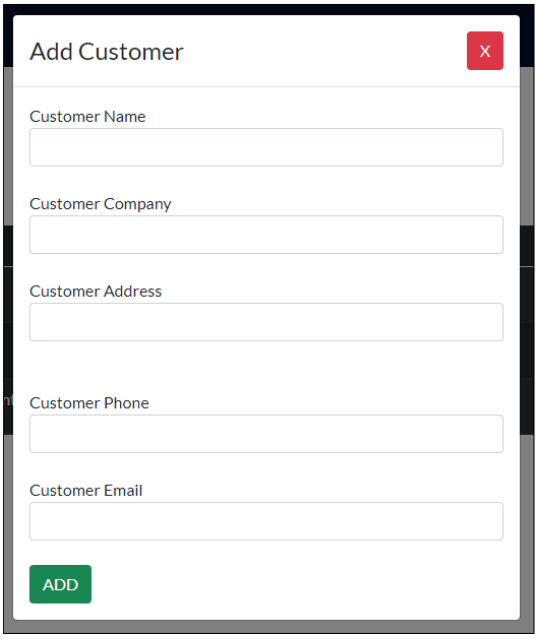


Gambar 3.18. Tampilan dari *flowchart* halaman *Customer*

Gambar 3.19 menunjukkan *flowchart* dari proses penambahan *customer*. Pertama, ketika tombol ditekan halaman akan memunculkan suatu *modal* yang berisikan *form* untuk menambahkan customer. Setelah data berhasil ditambahkan, akan muncul alert yang mengkonfirmasi penambahan data, dan data akan langsung disimpan pada *database firestore*. Tampilan halaman *add customer* dapat terlihat pada Gambar 3.20. *Alert* yang dimunculkan ketika proses penambahan *customer* sukses dapat terlihat pada Gambar 3.21.

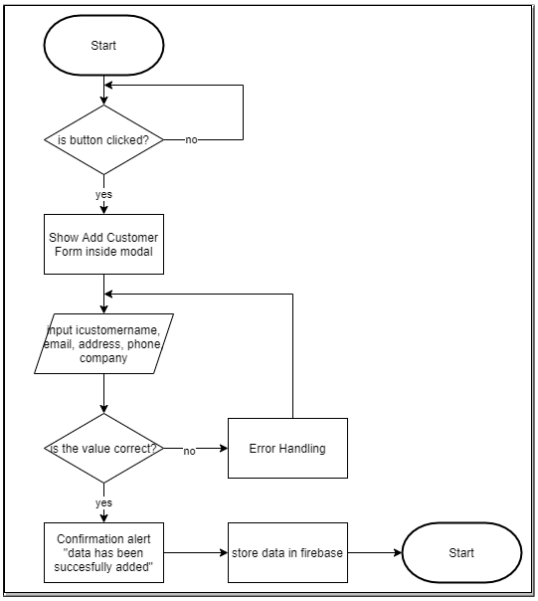


Gambar 3.19. Tampilan dari *flowchart* penambahan *Customer*



Gambar 3.20. Tampilan dari *modal* Halaman *Add Customer*

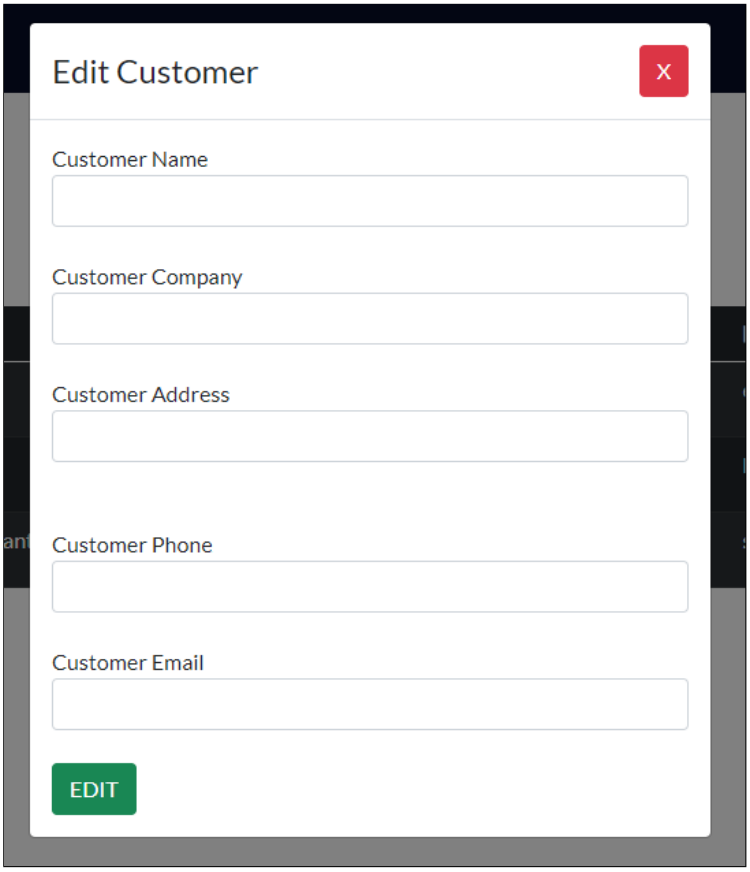
Gambar 3.21. Tampilan dari *Alert* ketika penambahan data berhasil



Gambar 3.22. Tampilan dari *flowchart Show Customer*

Selain menunjukkan *customer*, tabel juga bisa melakukan *data editing* dan *delete data* dengan menekan tombol yang bersangkutan pada tabel. Hal ini terlihat pada *flowchart* Gambar 3.22 di mana terdapat proses yang memungkinkan hal tersebut dapat terjadi.

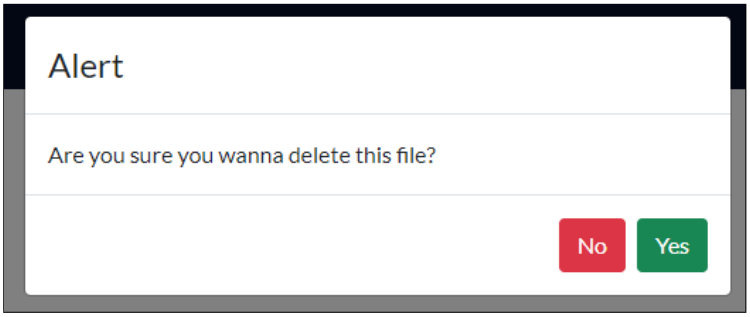
Ketika tombol *edit* ditekan, maka akan muncul *form* didalam suatu *modal*. *Modal* tersebut dapat dilihat pada Gambar 3.23. Jika semua data telah benar dan tombol *edit* ditekan, maka akan muncul *alert* yang menyatakan bahwa data berhasil di *edit*. *Alert* terlihat pada Gambar 3.24.



Gambar 3.23. Tampilan dari *Modal Edit Customer*

Gambar 3.24. Tampilan dari *Alert* ketika *Customer* berhasil di*edit*

Ketika tombol *delete* ditekan, maka akan muncul *confirmation pages* berupa *modal* untuk konfirmasi penghapusan. Bila terkonfirmasi, makan akan muncul *alert* tanda berhasil menghapus. *Modal* dapat terlihat pada Gambar 3.25, dan *alert* yang bersangkutan dapat terlihat pada Gambar 3.26.

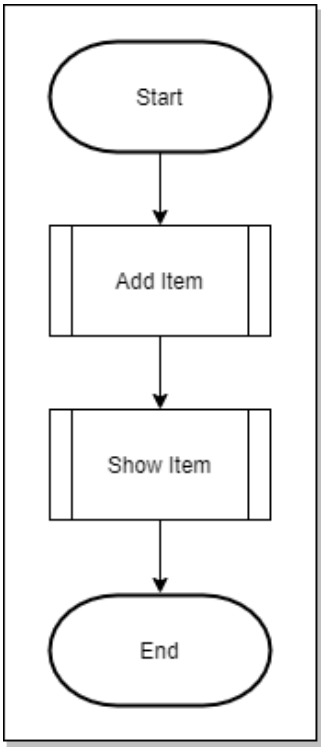


Gambar 3.25. Tampilan dari *Modal Delete Customer*

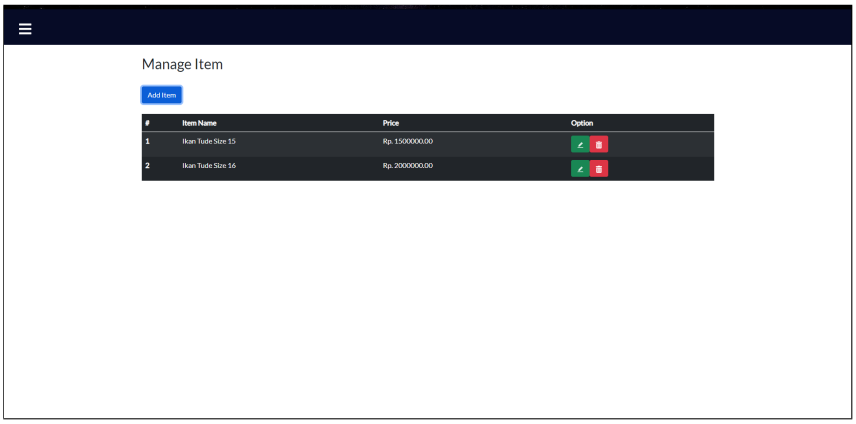
Gambar 3.26. Tampilan dari *Alert* ketika *Customer* berhasil di*delete*

D.3 Halaman Item

Halaman *item* mempunyai konsep yang mirip dengan halaman *customer*. Hanya saja, halaman *item* juga menggunakan foto atau gambar untuk dimasukkan ke data mereka. Walaupun terdapat perbedaan, namun proses yang terdapat pada halaman *item* tetaplah mirip dengan proses yang terjadi pada *item*. Proses tersebut dapat dilihat pada *flowchart* di Gambar 3.27. Tampilan dari halaman *item* dapat terlihat pada Gambar 3.28.



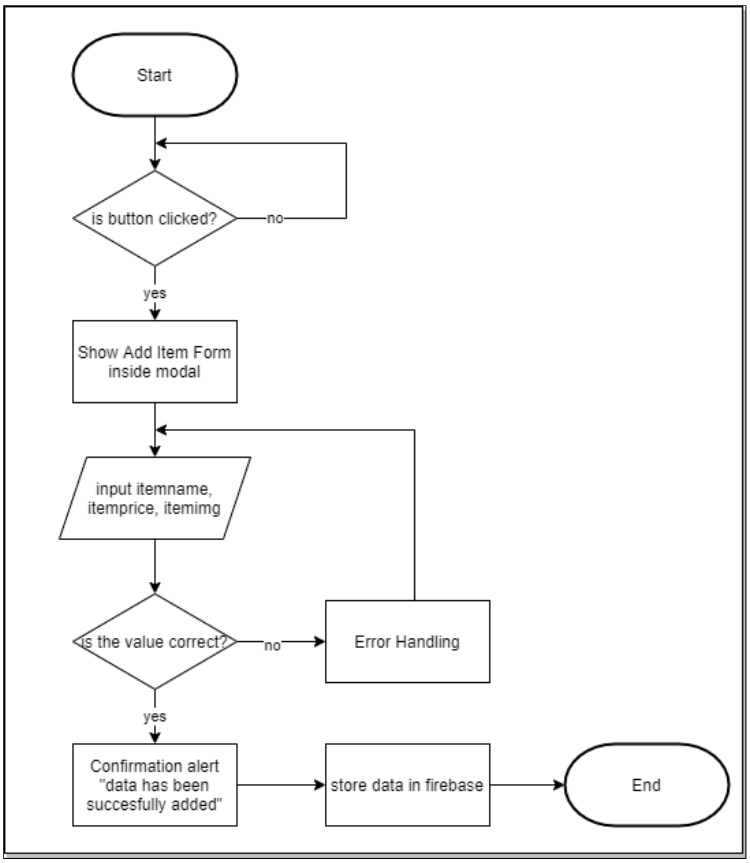
Gambar 3.27. Tampilan dari *Flowchart* halaman *Item*



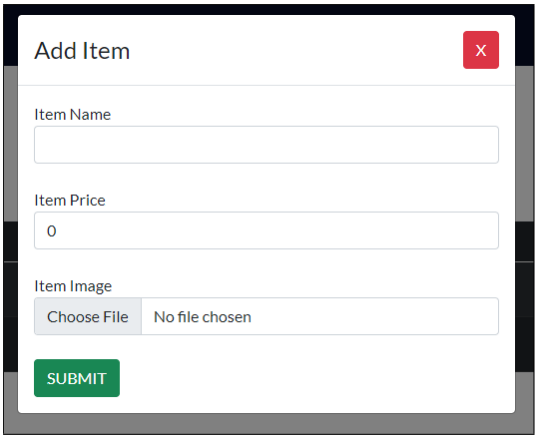
Gambar 3.28. Tampilan dari Halaman *Item*

Gambar 3.29 menunjukkan *flowchart* dari proses penambahan *item*. Pertama, ketika tombol ditekan halaman akan memunculkan suatu *modal* yang berisikan *form* untuk menambahkan item. Setelah data berhasil ditambahkan, akan muncul alert yang mengkonfirmasi penammbahan data, dan data akan langsung disimpan pada *database firestore*. Sebelumnya, akan ada juga pengecekan data yang diunggah apakah benar merupakan *image data*. Tampilan *modal* untuk *add*

item terlihat pada Gambar 3.30. *Alert* yang tercipta setelah proses tersebut berhasil dapat terlihat pada Gambar 3.31.

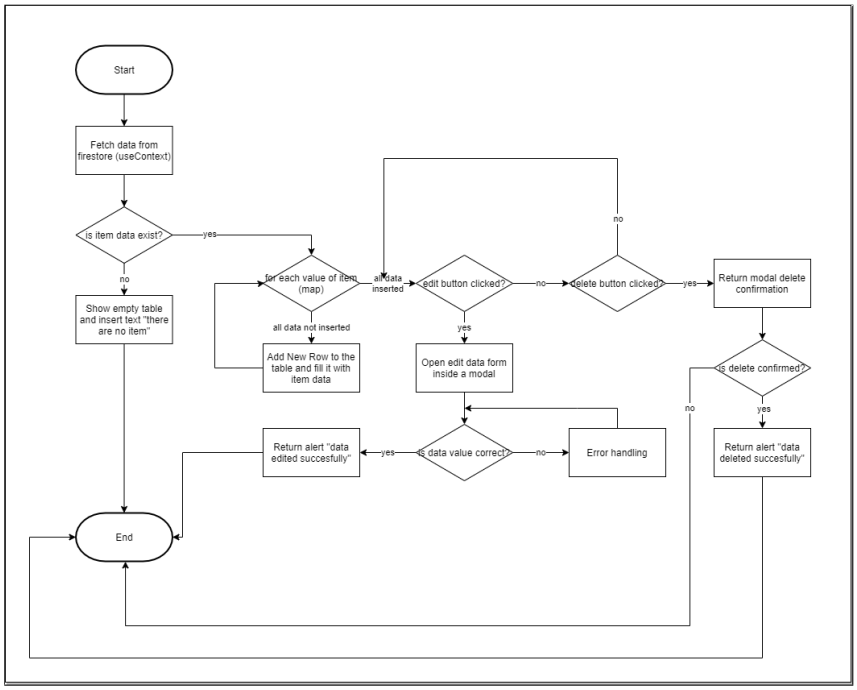


Gambar 3.29. Tampilan dari *flowchart* penambahan *item*



Gambar 3.30. Tampilan dari *modal* Halaman *Add item*

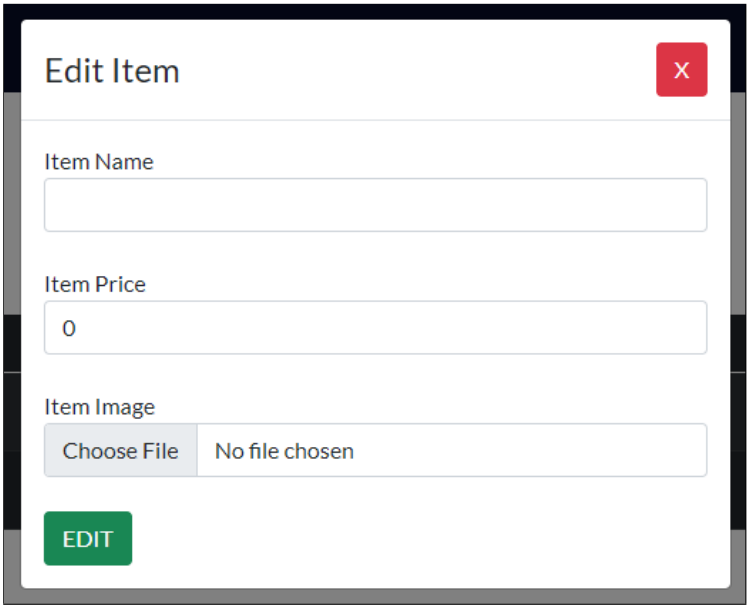
Gambar 3.31. Tampilan dari *Alert* ketika penambahan data berhasil



Gambar 3.32. Tampilan dari *flowchart Show item*

Selain menunjukkan *item*, tabel juga bisa melakukan *data editing* dan *data deleting* dengan menekan tombol yang bersangkutan pada tabel. Hal ini terlihat pada *flowchart* Gambar 3.32 di mana terdapat proses yang memungkinkan hal tersebut.

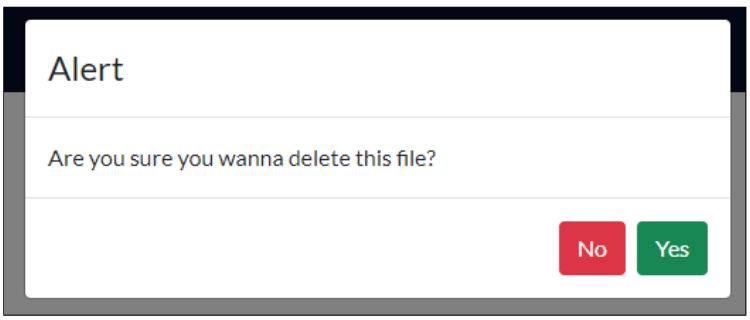
Ketika tombol *edit* ditekan, maka akan muncul *form* di dalam suatu *modal*. *Modal* tersebut dapat dilihat pada Gambar 3.33. *Alert* yang ditampilkan ketika proses berhasil dapat terlihat pada Gambar 3.34



Gambar 3.33. Tampilan dari *Modal Edit item*

Gambar 3.34. Tampilan dari *Alert* ketika *item* berhasil di*edit*

Ketika tombol *delete* ditekan, maka akan muncul *confirmation pages* berupa *modal* untuk konfirmasi penghapusan. Bila terkonfirmasi, maka akan muncul *alert* tanda berhasil menghapus. Proses konfirmasi ditunjukkan pada Gambar 3.35 dan *alert* yang ditampilkan terdapat pada Gambar 3.36.



Gambar 3.35. Tampilan dari *Modal Delete item*

Gambar 3.36. Tampilan dari *Alert* ketika *item* berhasil di*delete*

3.4 Kendala dan Solusi yang Ditemukan

Selama pelaksanaan kerja magang, ada beberapa kesulitan atau kendala yang dihadapi oleh penulis, yaitu:

1. Belum mempunyai pemahaman terhadap ReactJS
2. Adanya kesulitan mengubah *boolean state* untuk memunculkan *form* yang berbentuk *dropdown*
3. Kondisi pandemi yang membuat pekerjaan dilakukan secara *work from home* menjadikan komunikasi sulit

Setiap kendala yang terjadi harus memiliki solusi untuk memaksimalkan kerja magang. Berikut adalah solusi yang digunakan oleh penulis:

1. Melakukan riset dan pembelajaran ReactJS melalui beberapa dokumentasi yang ada seperti dokumentasi *ReactJS* (Anthony et al., 2017), *Ant Design*, dan juga *Bootstrap* (Singh dan Bhatt, 2016)
2. Mengubah strategi dan membuat *form* menjadi *modal*
3. Melakukan wawancara dan komunikasi yang intens melalui *Whatsapp* dan telepon untuk menghindari miskomunikasi