3. ANALISIS DAN DESAIN SISTEM

3.1. Analisis Permasalahan

Aplikasi *Password Manager* telah digunakan oleh banyak orang dimana aplikasi dapat menyimpan data termasuk data pribadi dll. Dalam penggunaan aplikasi *Password Manager* yang ada seperti *LastPass* dan *ProtonPass* menawarkan fitur penyimpanan data secara sentralisasi dengan kepercayaan dari pihak *provider* tersebut. Dengan penyimpanan secara sentralisasi memungkinkan terjadi resiko kehilangan dan manipulasi data. Salah satu resiko kehilangan data yang dapat terjadi pada aplikasi *Password Manager* sentralisasi adalah saat pengguna aplikasi tersebut tidak sengaja kehilangan aplikasi dan membuat data dalam aplikasi tersebut hilang. Saat aplikasi yang hilang ter*install* kembali di perangkat pengguna maupun perangkat lain, data yang hilang tidak akan bisa diambil kembali.

Selain permasalahan kehilangan data dari pihak dalam, resiko kehilangan dan pencurian data dapat terjadi akibat dari serangan pihak luar. Salah satu contohnya bila suatu perusahaan dari penyedia aplikasi tersebut jatuh atau terkena serangan *cyber* dari oknum yang tidak bertanggung jawab, maka data yang disimpan secara terpusat melalui penyedia tersebut dapat beresiko dicuri, dimanipulasi, dihapus, dll.

3.2. Analisis Kebutuhan

Berdasarkan permasalahan yang telah dibahas pada sub bab sebelumnya, melalui teknologi *Blockchain* dan *Smart Contract* yang menawarkan penyimpanan secara terdesentralisasi dapat digunakan untuk mengatasi resiko kehilangan maupun manipulasi data dalam aplikasi *Password Manager.* Data yang telah ditambahkan dalam aplikasi yang dibuat dapat disetor dalam *Blockchain* dan disimpan dalam bentuk yang telah terenkripsi. Data bersifat transparan, akan tetapi hanya pemilik *wallet* saja yang dapat mengakses dan mendekripsi data sesuai dengan masing-masing akun pengguna. Pengguna lain tidak dapat mendekripsi data dari pengguna lain dengan akun *wallet* yang berbeda.

Saat data dalam aplikasi hilang atau telah termanipulasi, data yang telah disetor di dalam *Blockchain* dapat diambil kembali oleh pengguna sehingga data yang hilang dan termanipulasi tersebut dapat dikembalikan dan ditampilkan di dalam aplikasi sesuai dengan data terakhir yang diupload ke dalam *chain*.

3.3. Analisis Data

Struktur data yang akan digunakan akan dibedakan menjadi bentuk data sebelum terenkripsi (*decrypted data*) dan data setelah terenkripsi (*encrypted data*).

3.3.1. *Decrypted* Data

Sebelum data dienkripsi, tipe data yang digunakan berupa objek *JavaScript* dengan struktur sebagai berikut:

```
const credential = {
  id: "123",
  title: "data 1",
  username: "user@gmail.com",
  password: "admin123",
  createdAt: 1717500000000,
  updatedAt: 1717500000000
};
```

Segmen Program 3.1 Tipe Data

Objek diatas merupakan *plaintext* yang dapat dibaca oleh user. Data tersebut akan diubah ke dalam format *JSON* menjadi satu *string* yang dapat mempermudah dalam pemrosesan data.

```
'{"id":"123","title":"data
1","username":"user@gmail.com","password":"admin123","createdAt":1717
500000000,"updatedAt":1717500000000}'
```

Segmen Program 3.2 Tipe Data *JSON*

Data *JSON* diatas kemudian akan diubah menjadi bentuk *array byte* yang merepresentasikan setiap karakter dalam *JSON string*. Proses enkripsi akan dilakukan dengan menggunakan algoritma enkripsi *XChaCha20-Poly1305*. Data akan diubah menjadi *hex string* dalam proses enkripsi.

3.3.2. *Encrypted* Data

Setelah data terenkripsi, data akan diubah menjadi objek baru dengan tipe data objek *JavaScript* dengan struktur sebagai berikut:

```
const encryptedCredential = {
  id: "123",
  encryptedData: "a1b2c3d4e5f6...", // hex string hasil enkripsi
  iv: "1a2b3c4d5e6f708192a3b4c5d6e7f8090a1b2c3d4e5f6071", // hex
string nonce
  createdAt: 1717500000000,
  updatedAt: 1717500000000
};
```

Segmen Program 3.3 Tipe Data objek *Javascript*

Setelah data enkripsi berhasil didapatkan, *hex string* tersebut diubah menjadi *byte array* yang nantinya dapat diubah menjadi *JSON string* kembali dan di*parse* menjadi objek data dalam proses dekripsi seperti pada segmen program 3.4.

```
[
  {
    "id": "123",
    "encryptedData": "a1b2c3d4e5f6...",
    "iv": "1a2b3c4d5e6f708192a3b4c5d6e7f8090a1b2c3d4e5f6071",
    "createdAt": 1717500000000,
    "updatedAt": 1717500000000
  },
  ...
]
```

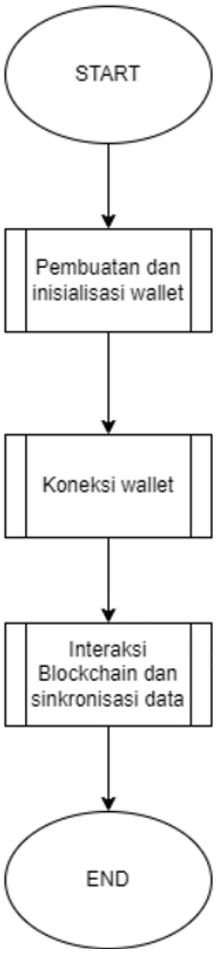
Segmen Program 3.4 objek data setelah di *parse*

3.4. Desain Sistem

Sistem dalam aplikasi *Password Manager* akan dibuat dalam bentuk *website* yang awalnya bersifat sentralisasi. Dalam aplikasi tersebut, terdapat fitur yang digunakan untuk menghubungkan aplikasi sentralisasi dengan *chain* yang bersifat desentralisasi, sehingga memungkinkan penyimpanan data tidak secara terpusat yang hanya dalam bentuk *local*. Saat pengguna ingin melakukan transaksi penyimpanan data dalam *chain*, pengguna harus memiliki aplikasi *Wallet Metamask*, yang bersifat *Ethereum compatible*. Pengguna akan menggunakan *Wallet* untuk autentikasi ke dalam aplikasi sehingga pengguna dapat mengakses data mereka yang telah ada dalam aplikasi. Selain itu, *wallet* juga digunakan untuk menyimpan *currency* dari *Ethereum* sebagai *gas fee* yang akan digunakan nantinya jika pengguna ingin *upload* data ke dalam *chain*. Saat ingin *upload* atau *download* data, akan ada konfirmasi dari *Wallet* dalam bentuk *signature*.

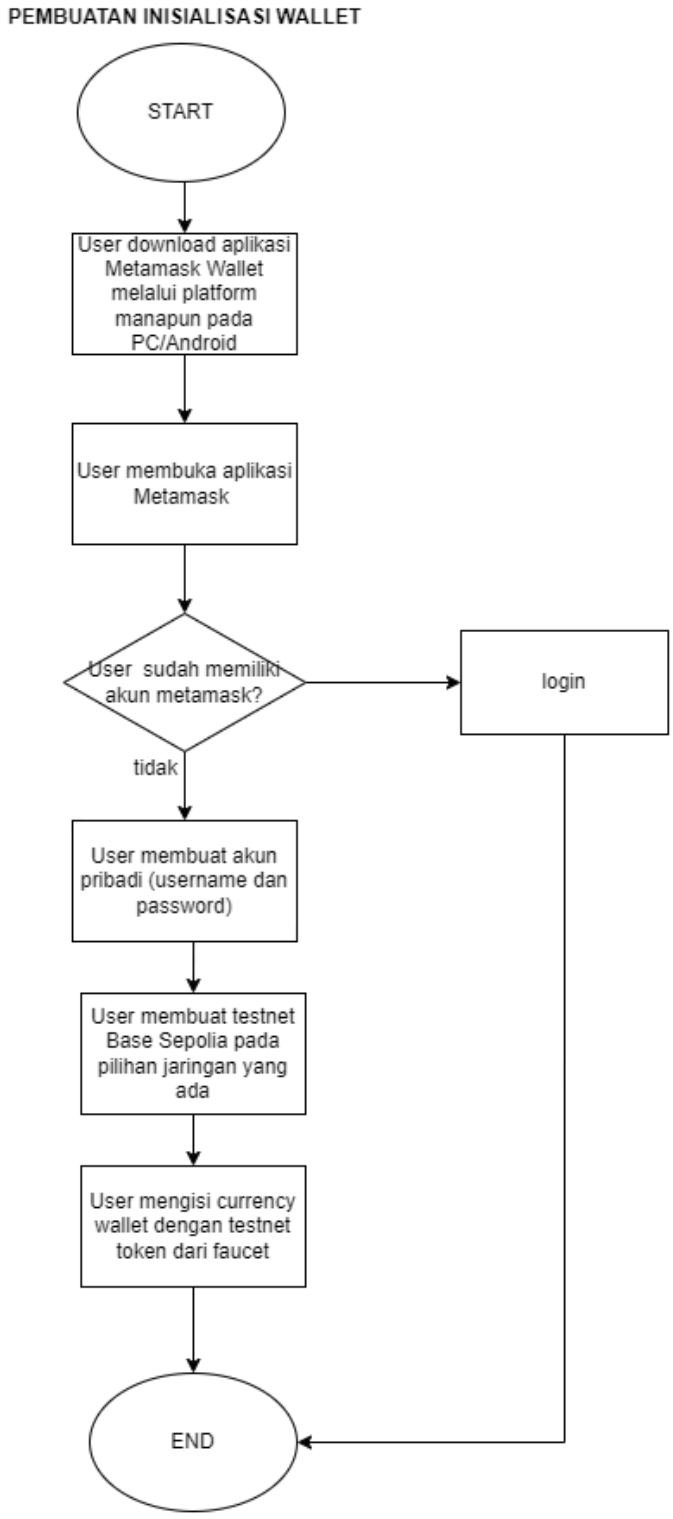
Langkah pertama yang akan dilakukan *user* adalah *download* aplikasi *Metamask* yang dapat di*install* melalui *platform* yang ada seperti *PlayStore* maupun *browser* yang ada dalam

perangkat. Setelah aplikasi berhasil ter*install*, maka pengguna dapat melakukan inisialisasi pertama dan koneksi pada aplikasi *wallet*. Jika berhasil, maka interaksi pada *Blockchain* dan sinkronisasi data dapat dilakukan.



Gambar 3.1 *Flowchart* Sistem Aplikasi dari segi *user*

3.4.1. Pembuatan dan Inisialisasi *Wallet*



Gambar 3.2 *Flowchart* Inisialisasi *Wallet*

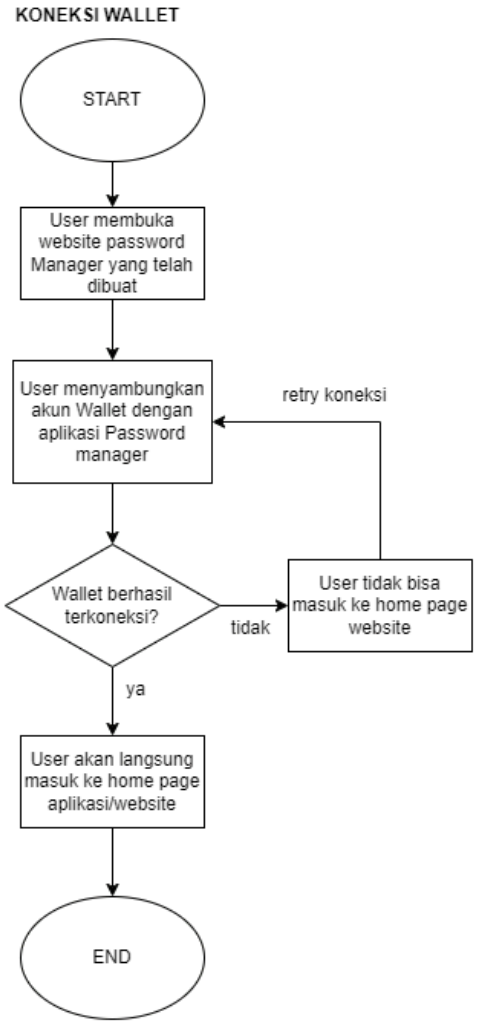
Pembuatan dan Inisialisasi *Wallet* akan dilakukan oleh pengguna sebelum mengakses ke dalam aplikasi *Password Manager.* Setelah aplikasi berhasil ter*install*, pengguna dapat langsung masuk ke dalam aplikasi untuk pembuatan akun pribadi. Transaksi *Blockchain* dapat

dilakukan tiap pemilik akun pada perangkat yang berbeda. Setelah pengguna berhasil membuat akun, maka pengguna telah memiliki *username*, *public key* serta *private key* di dalam aplikasi *wallet*. Di dalam *wallet*, juga terdapat *currency ETH* yang akan digunakan dalam transaksi nanti.

3.4.2. Pembuatan *Testnet*

Testnet merupakan jaringan dalam *Blockchain* yang digunakan sebagai jaringan uji coba. Jaringan ini bersifat tidak nyata dan dapat berguna untuk melakukan pengujian sistem tanpa perlu menggunakan aset asli. Pengguna dapat langsung membuat jaringan *testnet* baru yaitu *Base Sepolia Testnet* yang berada di dalam wallet dengan mengisi data yang ada. *Base Sepolia Testnet* menggunakan *currency ETH* sama seperti *Ethereum*, dimana *currency* tersebut dapat diperoleh melalui *faucet*.

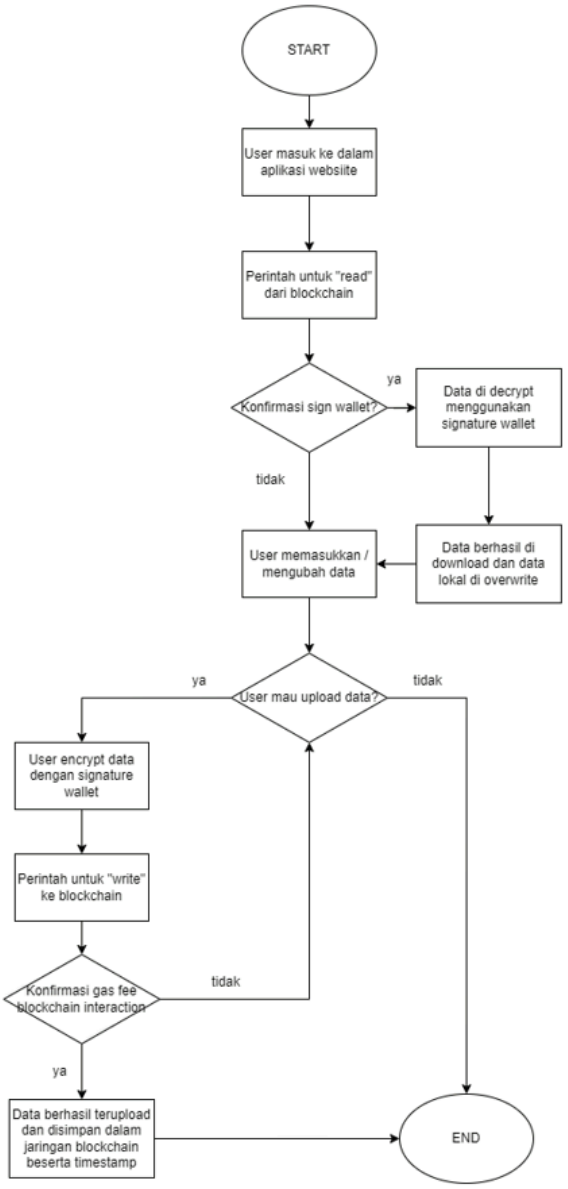
3.4.3. Koneksi *Wallet* dengan Aplikasi *Password Manager*



Gambar 3.3 *Flowchart* Koneksi *Wallet*

Setelah pembuatan Aplikasi *Password Manager* berbasis *website* telah dibuat, maka pengguna dapat langsung menyambungkan *wallet* dengan aplikasi tersebut untuk autentikasi. Jika pengguna belum membuat akun *wallet*, maka pengguna tidak bisa mengakses ke dalam *website*, begitu juga dengan jika pengguna tidak bisa mengakses *wallet* (lupa *password* atau akun hilang), maka pengguna tidak dapat masuk ke dalam *website Password Manager* tersebut. Apabila berhasil masuk, maka pengguna akan langsung diarahkan ke dalam *home page* dari *website*.

3.4.4. Interaksi *Blockchain* dan Sinkronisasi Data



Gambar 3.4 *Flowchart* Interaksi *Blockchain* dan Sinkronisasi Data

Saat pengguna ingin melakukan transaksi untuk penyimpanan data ke dalam *blockchain*, pengguna dapat mengakses terlebih dahulu *website Password Manager* yang telah terkoneksi dengan *wallet* masing-masing pengguna. Jika berhasil, maka akan muncul perintah untuk *read* dari *blockchain*. Jika pengguna ingin *download* data, maka akan muncul konfirmasi *sign* dari *wallet*, dan pengguna dapat memberikan konfirmasi sehingga data dapat di *download* dan *overwrite* dalam aplikasi. Pengguna dapat meng*edit* data yang kemudian dapat disetor kembali ke dalam *chain*.

Jika pengguna tidak ingin *download* data, maka data lokal yang telah ada di dalam aplikasi dapat langsung diubah/di*edit* yang kemudian dapat disetor ke dalam *chain*. Pengguna dapat memilih ingin menyetor data atau hanya menyimpannya secara lokal (jika tidak ingin terlalu sering *upload* data agar menghemat biaya). Jika pengguna ingin menyetor data, maka akan muncul konfirmasi *sign* dari *wallet* dan pengguna dapat berhasil mengenkripsi data. Setelah itu akan muncul perintah *write* dari *blockchain* dan pengguna dapat konfirmasi biaya (*gas fee*) agar data berhasil ter-*upload* ke dalam *chain* beserta waktu *upload* akan ditampilkan (*timestamp*).

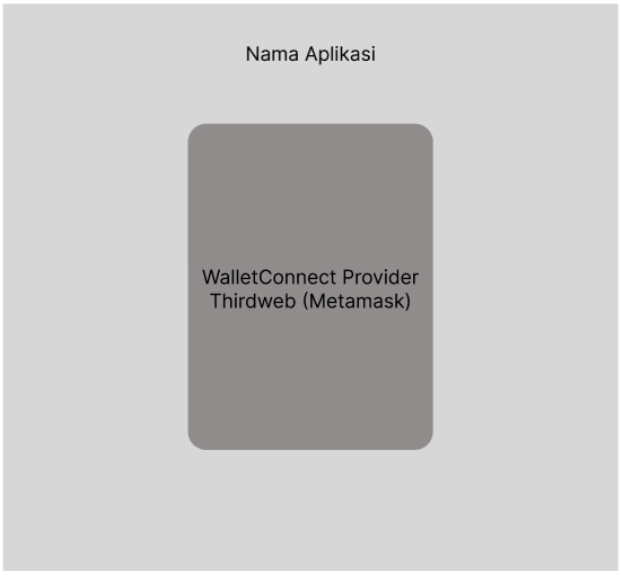
3.5. Desain Aplikasi

Aplikasi yang akan dibuat merupakan aplikasi *Password Manager,* yaitu sebuah *platform* untuk menyimpan dan mengelola suatu data seperti *password*, nama, *email*, dan catatan lainnya. Aplikasi ini akan dibuat seperti aplikasi *Password Manager* pada umumnya, dimana dari segi fungsional berfungsi untuk penyimpanan data. Pada aplikasi yang akan dibuat kali ini tidak terdapat fitur kompleks dimana hanya dari segi fungsional terdapat fitur - fitur utama untuk menyimpan data, meng*edit* data, menghapus data, fitur akun, serta fitur untuk transaksi dengan sistem *chain*.

Aplikasi *Password Manager* akan dibuat berbasis *website* menggunakan platform *Next.js* yang merupakan *framework* dari *react*. *Next.js* bersifat *full framework (tech stack)* yang dapat digunakan secara praktis untuk *website full stack*. Data yang disimpan tidak menggunakan *database* seperti *php My Admin* atau *My SQL*, melainkan di dalam *browser*. Data yang disimpan masih bersifat terpusat yang kemudian akan didesentralisasikan ke dalam *chain*.

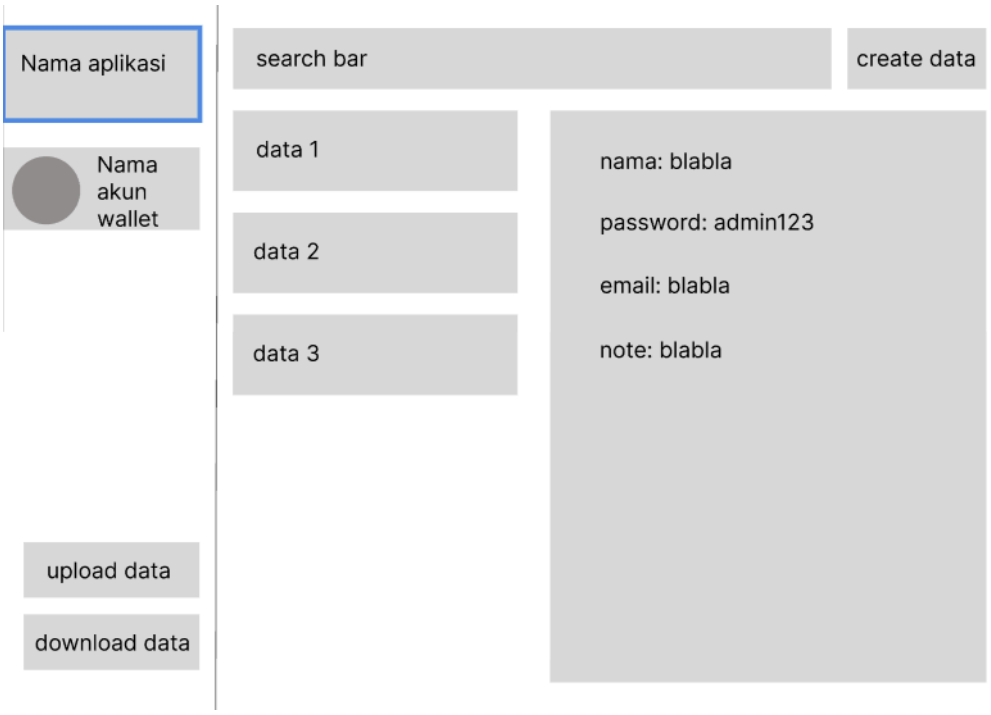
3.5.1. Autentikasi

Sebelum pengguna masuk ke dalam halaman utama dari *website*, pengguna diminta untuk *login* terlebih dahulu pada halaman *login*. Pengguna dapat *login* menggunakan *wallet Metamask* sesuai akun *Metamask* masing-masing pengguna.



Gambar 3.5 Tampilan halaman *login website*

3.5.2. Home Page dan Create Data



Gambar 3.6 Tampilan halaman utama *website*

Halaman utama *website* akan menampilkan nama aplikasi, nama akun *wallet*, data yang tersimpan secara lokal berisi nama/*username*, *password*, *email*, dan *note*. Selain itu juga terdapat *search bar* untuk mempermudah pengguna dalam mencari data. Terdapat fitur

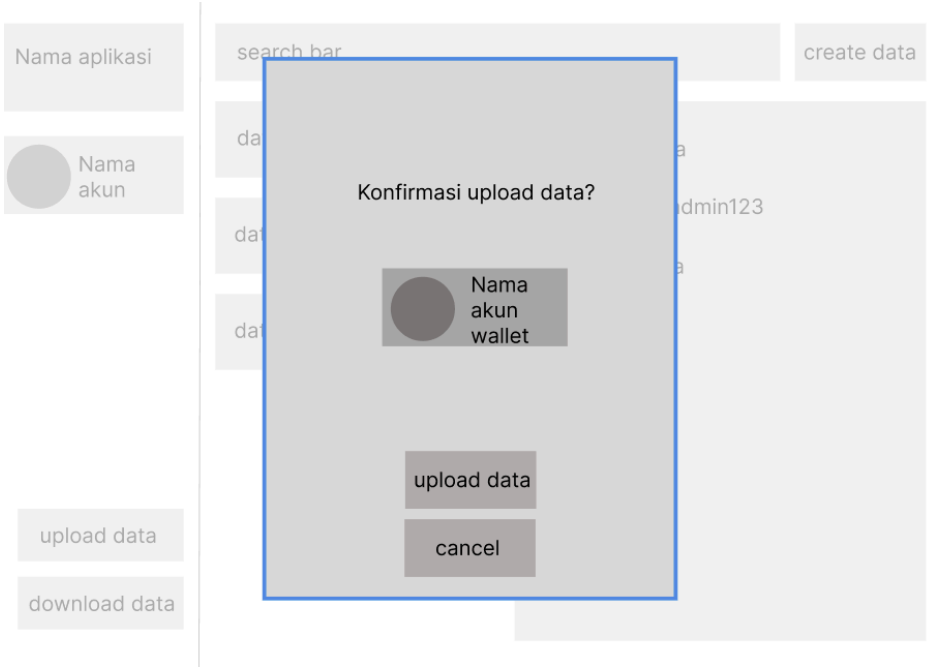
create data untuk menambah data *login* baru. Fungsi dari halaman utama ini ialah untuk menampilkan data yang dapat dilihat pada bagian kanan dalam gambar 3.6. Tulisan data 1, 2 dan 3 merupakan judul dari data yang mau disimpan. Pada pojok kiri bawah terdapat dua *button* sebagai fitur untuk transaksi dengan *blockchain* (*upload/write* dan *download/read* data). Pada fitur di sebelah kanan atas digunakan untuk menambah data baru yang dapat dilihat pada gambar 3.7. Pengguna dapat memasukkan nama, *password*, *email*, dan catatan bebas.

Gambar 3.7 Tampilan halaman *create* data baru

3.5.3. *Edit* Data

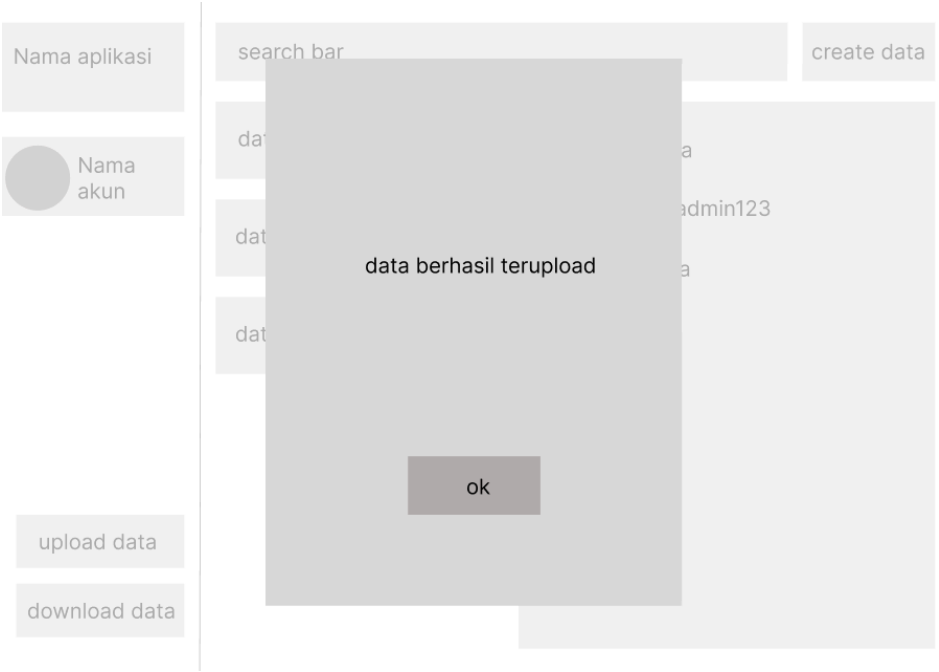
Saat pengguna ingin meng*edit* data, maka pengguna dapat menekan salah satu data yang ingin di*edit* pada halaman *home page,* dan pengguna dapat langsung mengganti data dan simpan data kembali pada penyimpanan aplikasi.

3.5.4. *Upload/write* dan *Download/read* Data



Gambar 3.8 Tampilan halaman *upload* data

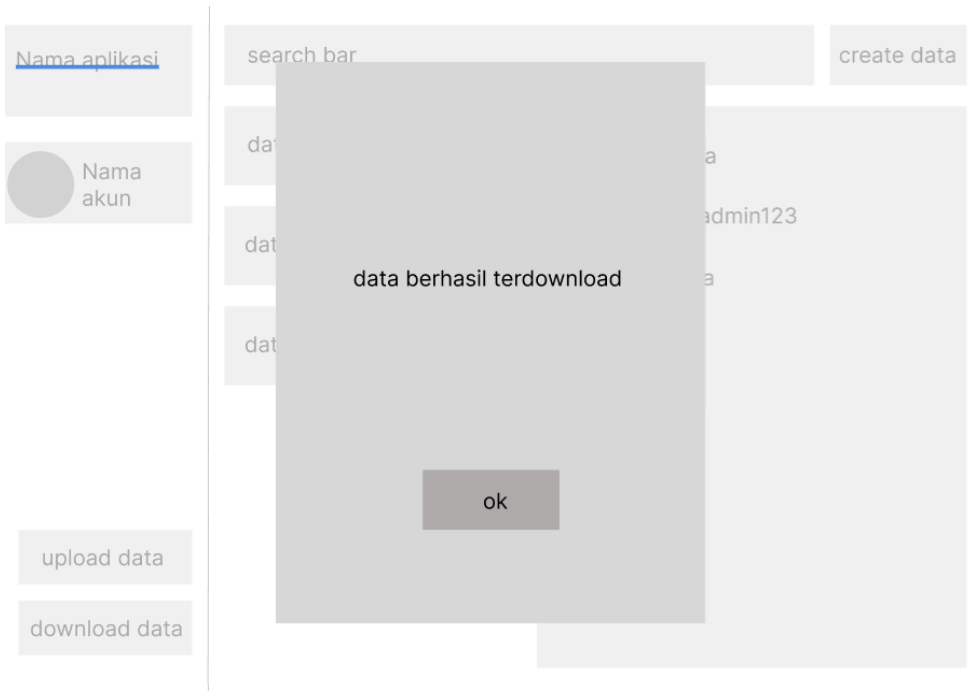
Saat pengguna ingin *upload* data, maka akan muncul konfirmasi *sign wallet* sebagai tanda tangan digital pengguna untuk melakukan transaksi penulisan data ke dalam *chain* (gambar 3.8). Jika berhasil maka akan muncul tulisan berhasil *upload* (gambar 3.9).



Gambar 3.9 Tampilan konfirmasi halaman *upload* data

Pengguna dapat membatalkan transaksi bila tidak jadi setor data. Selain itu, pengguna dapat *download* kembali data yang pernah disetor ke dalam *chain* kembali kepada penyimpanan lokal. Data yang di *read* merupakan data terakhir yang disetor ke dalam *chain*.

Gambar 3.10 Tampilan halaman *download* data



Gambar 3.11 Tampilan konfirmasi halaman *download* data