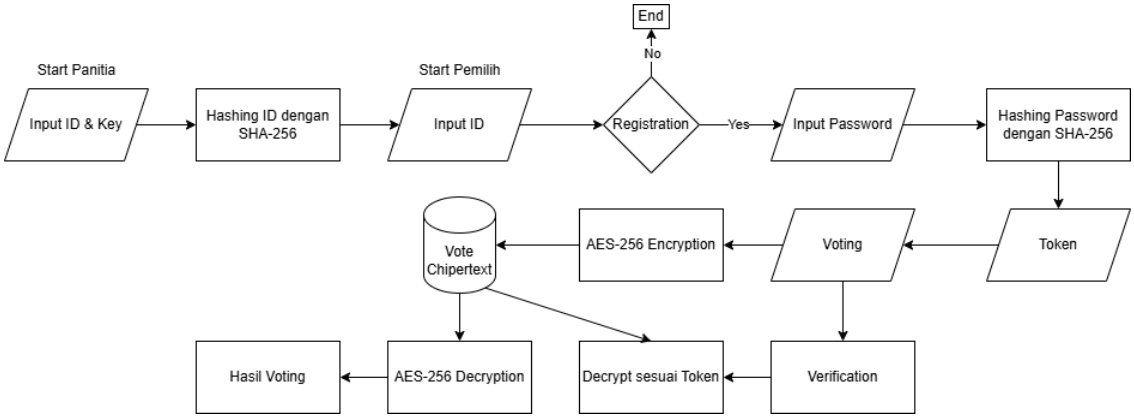
BAB IV
PEMBAHASAN

4.1 Skema Aplikasi *E-Voting* Menggunakan AES-256 dan SHA-256

Pengembangan aplikasi *e-voting* menggunakan algoritma AES-256 dan SHA-256 didasarkan pada Gambar 4.1. Skema dalam Gambar 4.1 menggambarkan keseluruhan proses aplikasi *e-voting* dengan memanfaatkan algoritma AES-256 dan SHA-256 sebagai berikut.



Gambar 4.1 Skema *E-Voting* menggunakan AES-256 dan SHA-256

1. Panitia meng-*input* data pemilih berupa ID setiap pemilih yang akan melakukan voting.
2. Aplikasi akan melakukan *hashing* pada ID dan nilai *hash* yang dihasilkan akan disimpan dalam *database* ID.
3. Admin meng-*input* kunci untuk enkripsi dan dekripsi.
4. Pemilih meng-*input* ID dan ID di-*hashing* untuk disamakan dengan nilai *hash* ID dalam *database* ID.
5. Jika tidak terdapat nilai *hash* ID yang sama, maka pemilih tidak dapat lanjut ke halaman berikutnya. Jika terdapat nilai *hash* yang sama, maka pemilih diminta untuk meng-*input* kata sandi atau *password* yang akan di-*hashing* dan digunakan sebagai token.
6. Aplikasi akan melakukan *hashing* pada kata sandi yang di-*input* dan menyimpannya dalam *database* hasil voting.

7. Pemilih melakukan voting dan hasil voting dienkripsi lalu disimpan dalam *database* hasil voting dan dipasangkan dengan nilai *hash* dari kata sandi.
8. Jika pemilih ingin melakukan verifikasi hasil pilihan, pemilih akan diminta untuk meng –*input* token yang didapat kemudian aplikasi akan mendekripsi hasil voting yang bersesuaian dengan token.
9. Aplikasi akan mendekripsi seluruh hasil voting, kemudian semuanya akan dihitung dan ditampilkan pada Aplikasi.

4.2 Konstruksi Aplikasi *E-Voting* menggunakan AES-256 dan SHA-256

Aplikasi *e-voting* memiliki dua komponen utama dalam proses pengamanan data. Komponen pertama adalah pengamanan data menggunakan AES-256 dan SHA-256, yang digunakan untuk proses enkripsi, dekripsi, serta *hashing*. Komponen kedua adalah komponen antarmuka untuk aplikasi *e-voting*.

4.2.1 *Pseudocode* Aplikasi *E-Voting* menggunakan AES-256 dan SHA-256

Bagian ini menjelaskan *pseudocode* yang dipakai pada aplikasi *e-voting* menggunakan AES-256 dan SHA-256 yang dibagi dalam lima bagian yaitu *pseudocode* untuk *generate* kunci, enkripsi, dekripsi, *hashing*, dan program utama**.**

1. *Pseudocode generate* kunci

   Kunci yang akan digunakan dalam aplikasi *e-voting* di-*input* terlebih dahulu ke dalam program dalam bentuk *string*. Fungsi *KeytoStateKey*() akan mengubah kunci yang awalnya *string* menjadi sebuah matriks. Fungsi *KeyExpansion()* akan diterapkan kepada matriks kunci untuk membuat *RoundKey* 1 sampai *RoundKey* 14. *Pseudocode* dari algoritma tersebut adalah sebagai berikut.

```
Procedure GenerateKunci
/*generate kunci untuk AES-256*/
DEKLARASI
  Key: string
  StateKey[0:2], RoundKey[0:15]: list
ALGORITMA
  read(key)
  StateKey[0:2] ← KeytoStateKey(Key)
```

```
RoundKey[0:15] ← KeyExpansion(StateKey[0:2])
```

2. *Pseudocode* enkripsi

   Enkripsi pada aplikasi *e-voting* menggunakan kunci yang di-*input* oleh panitia untuk digunakan dalam AES-256. Data akan diubah mejadi sebuah matriks 4x4 yang disebut *State* oleh fungsi *PlaintoState()*. Kunci yang akan digunakan adalah RoundKey[0:15] yang akan mengenkripsi data dalam 15 *round* dalam fungsi *EncryptAES()*. *Input* dalam *pseudocode* ini adalah data dan RoundKey[0:15]. *Output* dari *pseudocode* ini adalah cipherteks dari data. *Pseudocode* dari algoritma tersebut adalah sebagai berikut.

```
Procedure EnkripsiData
/*mengenkripsi data menggunakan AES-256*/
DEKLARASI
  RoundKey[0:15], State: list
  Plainteks, Cipherteks: string
ALGORITMA
  read(RoundKey[0:15])
  read(Plainteks)
  State ← PlaintoState(Plainteks)
  Cipherteks ← EncryptAES(State, RoundKey[0:15])
  write(Cipherteks)
```

3. *Pseudocode* dekripsi

   Dekripsi cipherteks dalam aplikasi *e-voting* juga menggunakan kunci yang di-*input* oleh panitia untuk digunakan dalam AES-256. Kunci yang akan digunakan adalah RoundKey[0:15] yang akan mendekripsi data dalam 15 *round* dalam fungsi *decryptAES().* Hasil dekripsi akan berupa matriks *state* yang akan diubah menjadi *string* plainteks denga fungsi *StatetoPlain()*. Input dalam pseudocode ini adalah cipherteks dan *RoundKey[0:15]*. *Output* dari *pseudocode* ini adalah plainteks data. *Pseudocode* dari algoritma tersebut adalah sebagai berikut.

```
Procedure DekripsiData
/*mendekripsi data menggunakan AES-256*/
```

```
DEKLARASI
   RoundKey[0:15], State: list
   Cipherteks, Data: string
ALGORITMA
   read(RoundKey[0:15])
   read(Cipherteks)
   State ← DecryptAES(Cipherteks, Roundkey[0:15])
   Plainteks ← StatetoPlain(State)
   write(Plainteks)
```

4. *Pseudocode hashing*

   SHA-256 digunakan sebagai fungsi *hash* yang akan digunakan untuk menghasilkan *message digest* pada aplikasi *e-voting*. *Message digest* akan digunakan sebagai token verifikasi untuk memverifikasi ID, token, dan juga *password* panitia. *Pseudocode* dari algoritma tersebut adalah sebagai berikut.

```
Procedure Hashing
/*melakukan hashing menggunakan SHA-256*/
DEKLARASI
   Data, MD: string
ALGORITMA
   read(Data)
   MD ← SHA(Data)
  write(MD)
```

5. *Pseudocode* program utama

   Pseudocode program utama *e-voting* menggabungkan *procedure generate* kunci, enkripsi, dekripsi, dan *hashing* yang akan melakukan pengamanan data voting. Terdapat fungsi tambahan yaitu *Verification()* yang melakukan verifikasi *message digest* (MD) dari ID dan Token. *Pseudocode* dari program utama adalah sebagai berikut.

```
Procedure ProgramUtama
```

```
/*program utama e-voting menggunakan AES-256 & SHA 256*/
DEKLARASI
  Key, Data, ID, Token, MD_ID, MD_Token, Cipherteks: string
ALGORITMA
  read(Key)
  read(ID)
  MD_ID ← Hashing(ID)
  Verification(MD_ID)
  MD_Token ← Hashing(Token)
  Cipherteks ← EnkripsiData(Data, Key)
  Verification(MD_Token)
  Data ← DekripsiData(Cipherteks, Key)
  write(Data)
```

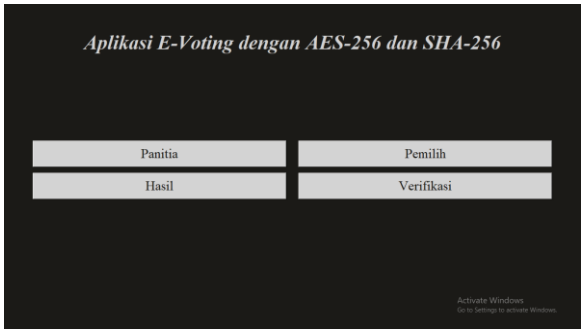
4.2.2 Tampilan Utama

Pada bagian sebelumnya sudah dipaparkan algoritma deskriptif, skema, dan *pseudocode* yang menjadi dasar pembuatan program aplikasi sederhana untuk aplikasi *e-voting* yang menggunakan kriptografi AES-256 dan SHA-256. Program ini dibuat menggunakan bahasa pemrograman *Python* dengan bantuan berbagai *library* didalamnya seperti tkinter, matplotlib, dan pandas.

Setelah menyelesaikan tahapan perancangan model dan pengkodean program, maka diperoleh aplikasi *e-voting* yang mengimplementasikan algoritma AES-256 dan SHA-256 untuk melakukan perahasiaan identitas serta pilihan voting. Berikut adalah tampilan aplikasi web yang sudah dibuat.

a. Tampilan halaman menu utama

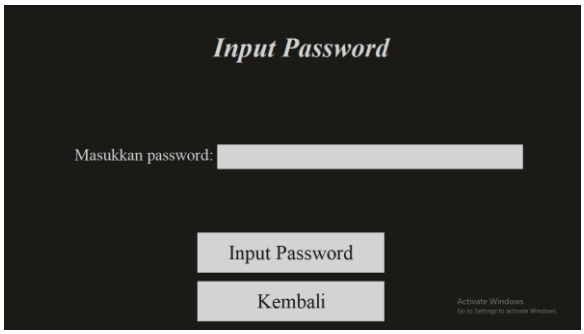
Halaman menu utama yang ditunjukkan pada Gambar 4.2 dalam program aplikasi *e-voting* memiliki 4 tombol utama, yaitu tombol panitia, pemilih, hasil dan verifikasi. Tombol panitia digunakan untuk akses halaman *input* kunci untuk AES-256 dan juga penginputan ID pemilih. Tombol pemilih digunakan untuk akses halaman voting. Tombol hasil digunakan untuk mengakses halaman hasil voting. Tombol verifikasi digunakan untuk melakukan verifikasi pilihan pengguna dengan token.



Gambar 4.2 Tampilan Halaman Menu Utama

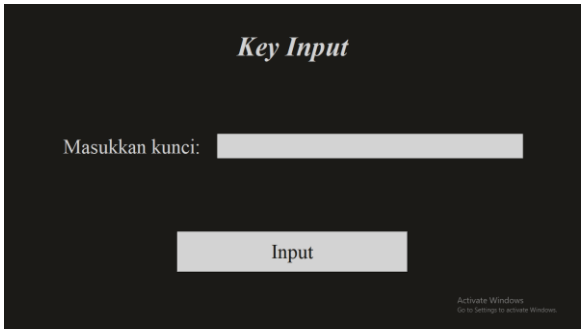
b. Tampilan halaman panitia

Halaman panitia akan terbagi menjadi empat halaman utama, yaitu halaman *input password, input* kunci, pilihan voting selesai atau *input* ID dan *input* ID pemilih. Pada halaman *input password*, panitia akan diminta untuk memasukkan *password* yang sesuai. Panitia dapat lanjut ke halaman berikutnya untuk mengatur akses hasil voting, memasukkan kunci dan juga ID pemilih jika *password* sesuai. Panitia tidak dapat lanjut ke halaman berikutnya jika *password* tidak sesuai. Halaman *input password* ditunjukkan pada Gambar 4.3.



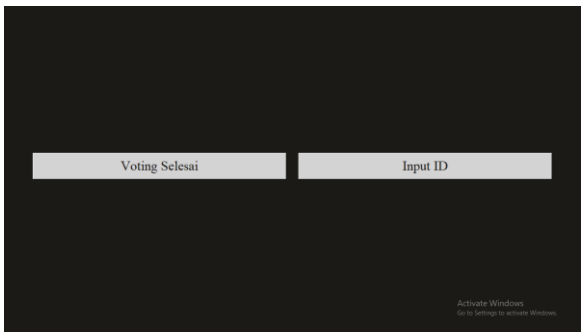
Gambar 4.3 Tampilan Halaman Input *Password* Panitia

Pada halaman *input* kunci, panitia akan diminta untuk memasukkan kunci yang akan digunakan untuk enkripsi d an dekripsi pilihan pemilih menggunakan AES-256. Halaman *input* kunci ditunjukkan pada Gambar 4.4.



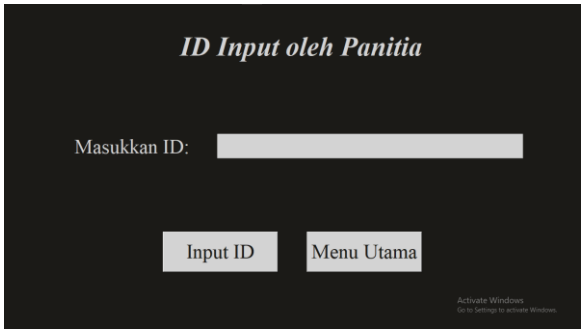
Gambar 4.4 Tampilan Halaman *Input* Kunci Panitia

Pada halaman pilihan voting selesai atau *input* ID, panitia dapat memilih apakah voting sudah selesai terlaksana atau ingin memasukkan ID pemilih. Apabila panitia memutuskan bahwa voting selesai, maka pemilih tidak dapat melakukan voting lagi dan hasil keseluruhan dapat ditampilkan. Halaman pilihan voting ditunjukkan pada Gambar 4.5.



Gambar 4.5 Halaman Voting Selesai atau *Input* ID

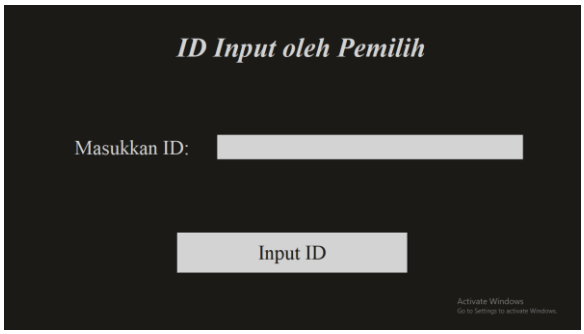
Pada halaman *input* ID pemilih, panitia akan diminta untuk memasukkan ID untuk pemilih yang memiliki hak untuk memilih. Panitia dapat beralih ke menu utama setelah selesai memasukkan semua ID pemilih. Halaman *input* Id pemilih ditunjukkan pada Gambar 4.6.



Gambar 4.6 Tampilan Halaman *Input* ID Panitia

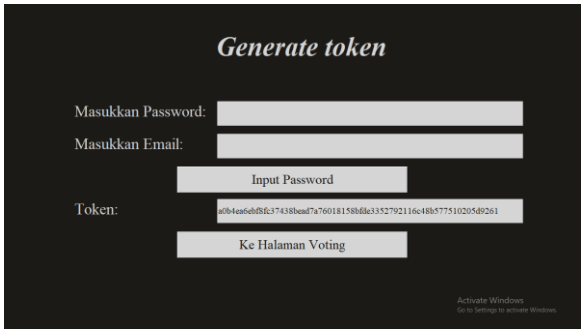
c. Tampilan halaman pemilih

Halaman pemilih akan terbagi menjadi tiga halaman utama, yaitu halaman *input* ID, *generate token*, dan voting. Pada halaman *input* ID, pemilih akan diminta untuk memasukkan ID yang sudah didaftarkan oleh panitia. Hanya pemilih yang sudah didaftarkan panitia saja yang dapat melakukan voting. Pemilih dapat lanjut ke halaman berikutnya untuk membuat token dan juga melakukan voting jika ID sesuai. Pemilih tidak dapat lanjut ke halaman berikutnya jika ID tidak sesuai. Halaman *input* ID ditunjukkan pada Gambar 4.7.



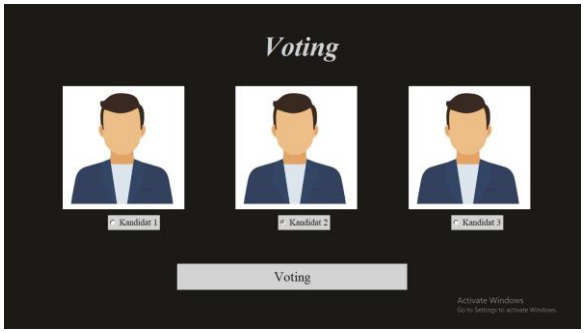
Gambar 4.7 Tampilan *Input* ID Pemilih

Pada halaman *generate* token, pemilih akan diminta untuk melakukan *input password* dan email perorangan yang akan diubah menjadi token dengan SHA-256. Token tersebut disimpan oleh pemilih untuk melakukan verifikasi pilihan nantinya. Setelah melakukan pembuatan token, pemilih dapat lanjut ke halaman voting. Halaman *generate* token ditunjukkan pada Gambar 4.8.



Gambar 4.8 Tampilan Halaman *Generate* Token

Pada halaman voting, pemilih akan melakukan voting pilihannya. Hasil voting nantinya akan dienkripsi dengan AES-256. Hasil voting yang sudah dienkripsi akan disimpan dalam sebuah *database* hasil voting. Setelah voting, pemilih dapat melakukan verifikasi pilihan dengan token atau melihat keseluruhan hasil voting. Halaman voting ditunjukkan pada Gambar 4.9.



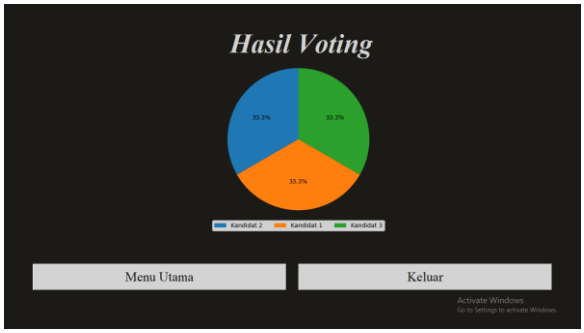
Gambar 4.9 Tampilan Halaman Voting

d. Tampilan halaman hasil

Pada halaman verifikasi pilihan dengan token, pemilih akan diminta untuk memasukkan token yang sudah dihasilkan pada halaman *generate* token. Jika token sesuai dengan apa yang ada di dalam *database*, maka program akan menampilkan pilihan voting pemilih. Jika token tidak sesuai, maka program akan memunculkan *error* bahwa token tidak sesuai. Halaman verifikasi pilihan ditunjukkan pada Gambar 4.10.

Gambar 4.10 Tampilan Halaman Verifikasi dengan Token

Pada halaman hasil voting, program akan melakukan dekripsi pada semua data hasil voting di *database*. Setelah didekripsi, pemilih dapat melihat hasil voting untuk tiap-tiap kandidat voting dalam bentuk diagram lingkaran jika seluruh pemilih sudah melakukan voting. Halaman hasil voting hanya bisa diakses apabila panitia menyatakan voting sudah selesai. Gambar 4.11 menunjukkan halaman hasil voting.



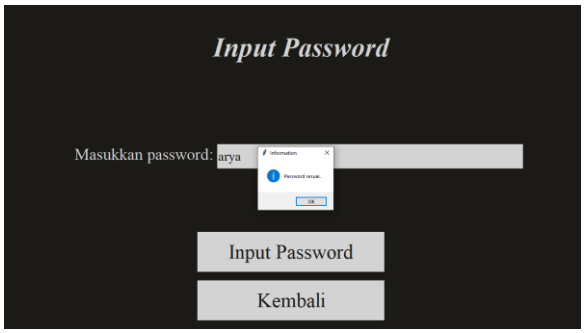
Gambar 4.11 Halaman Hasil Voting

4.3 Validasi Aplikasi *E-Voting* menggunakan AES-256 dan SHA-256

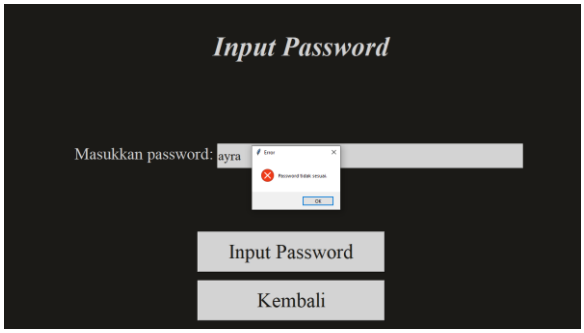
Bagian ini akan menjelaskan proses validasi yang ada pada aplikasi *e-voting* menggunakan AES-256 dan SHA-256. Proses validasi yang akan dilakukan adalah validasi *password* panitia, ID pemilih, serta pilihan voting pemilih.

4.3.1 Validasi *Password* Panitia

Password yang di-*input* oleh panitia akan dimasukkan ke dalam fungsi *hash* yang digunakan, yaitu SHA-256. *Password* yang dimasukkan akan diubah menjadi sebuah *message digest* oleh program dan dibandingkan dengan nilai yang ada di *database*. Apabila *message digest* yang dihasilkan sama dengan yang ada di database, maka panitia tersebut telah berhasil diotentikasi dan dapat memasukkan kunci dan ID pemilih seperti yang ditunjukkan pada Gambar 4.12. Jika *message digest* tidak sama dengan yang ada di database, maka *password* yang dimasukkan tidak valid, sehingga proses otentikasi gagal seperti yang ditunjukkan pada Gambar 4.13.



Gambar 4.12 Validasi *Password* Panitia yang Berhasil



Gambar 4.13 Validasi *Password* Panitia yang Gagal

4.3.2 Validasi ID Pemilih

Validasi ID pemilih diawali dengan memasukkan ID tiap pemilih oleh panitia. Setiap ID pemilih yang di-*input* oleh panitia haruslah memiliki nilai yang berbeda atau unik sehingga nilai *hash* yang dihasilkan akan berbeda. Pemilih melakukan input ID sebelum melakukan generate token dan voting. Program akan melakukan *hashing* kepada ID tersebut dan nilai *hash*-nya akan disamakan dengan nilai *hash* yang ada di *database*. Nilai *hash* ID dikatakan tervalidasi jika kedua nilai *hash* sama seperti pada gambar 4.14. Pemilih tidak tervalidasi apabila tidak terdapat nilai hash yang sama seperti yang ditunjukkan pada Gambar 4.15. Nilai *hash* milik pemilih akan dihapus dari *database* sehingga pemilih tidak dapat memilih lagi seperti yang ditunjukkan pada Gambar 4.16.

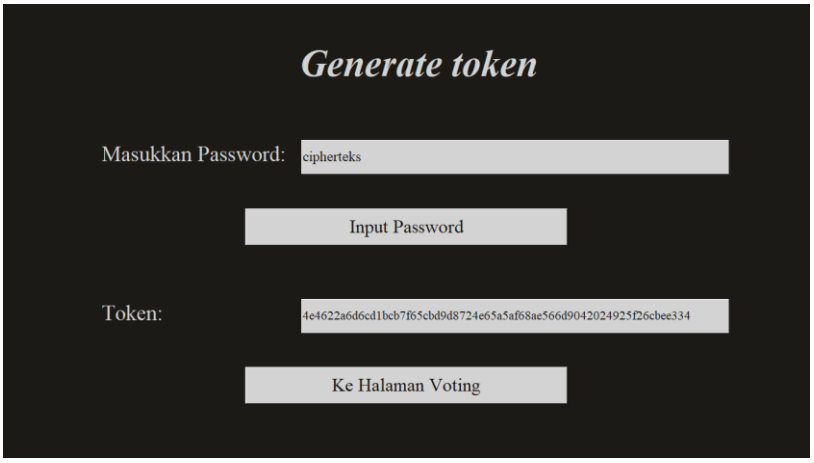
Gambar 4.14 Validasi ID Pemilih Terdaftar

Gambar 4.15 Validasi ID Pemilih Tidak Terdaftar

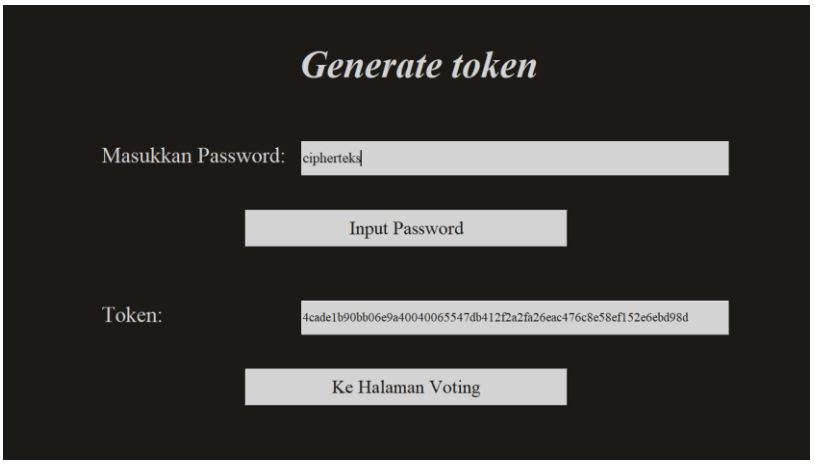
Gambar 4.16 Validasi ID Pemilih Sudah Voting

4.3.3 Validasi Pilihan Voting Pemilih

Token yang dihasilkan pada program aplikasi voting akan berbeda untuk setiap pemilih. Hal ini dikarenakan setiap *password* yang akan diubah menjadi token akan digabungkan terlebih dengan ID unik tiap pemilih. Akibatnya nilai *hash* atau token yang dihasilkan oleh setiap pemilih tidak mungkin sama dengan pemilih lainnya. Misalkan sebuah pemilih dengan ID "kriptografi" dan *password* "cipherteks" akan memiliki token berupa "4e4622a6d6cd1bcb7f65cbd9d8724e65a5af68ae566d9042024925f26cbee334" seperti yang terlihat pada Gambar 4.17. Sedangkan pemilih dengan ID "Kriptografi" dan *password* "cipherteks" akan memiliki token berupa "4cade1b90bb06e9a40040065547db412f2a2fa26eac476c8e58ef152e6ebd98d" seperti yang terlihat pada Gambar 4.18. Hal ini mengakibatkan setiap pemilih untuk memiliki *password* yang sama namun akan memiliki token yang berbeda.



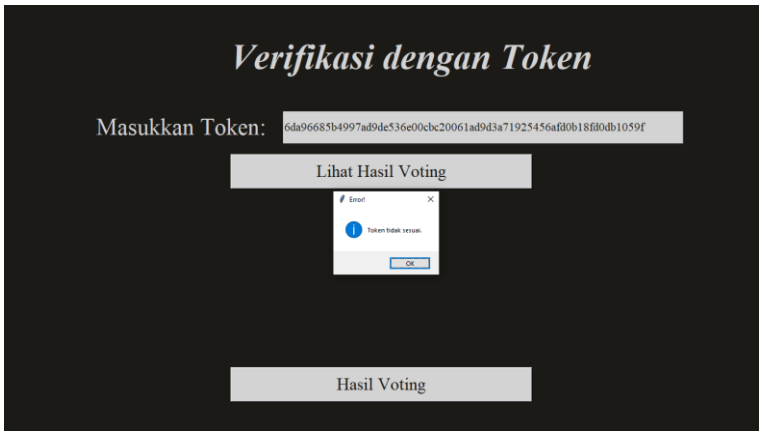
Gambar 4.17 Token untuk "kriptografi”



Gambar 4.18 Token untuk "Kriptografi"

Validasi pada pilihan pemilih dilakukan oleh program dengan membandingkan token yang dimasukkan oleh pemilih pada halaman verifikasi dengan token yang ada di *database*. Apabila terdapat token yang sama, maka program akan mendekripsi cipherteks voting yang berpasangan dengan token tersebut seperti yang ditunjukkan pada Gambar 4.19. Jika pilihan yang dimasukkan pemilih sesuai dengan yang tertera di halaman voting, maka hasil voting tersebut akan tervalidasi. Jika pilihan tidak sesuai, maka hasil voting tidak tervalidasi. Proses ini juga menegaskan prinsip nirpenyangkalan, karena pemilih tidak dapat membantah atau memodifikasi pilihan yang telah mereka buat setelah token mereka diverifikasi. Apabila token yang dimasukkan tidak ada di dalam *database*, maka program akan memunculkan notifikasi bahwa token tidak sesuai seperti yang ditunjukkan pada Gambar 4.20.

Gambar 4.19 Verifikasi Token yang Berhasil



Gambar 4.20 Verifikasi Token yang Tidak Berhasil