

Muhammad fathi Rafa

2311104022

Jurnal Modul 15

Index.html

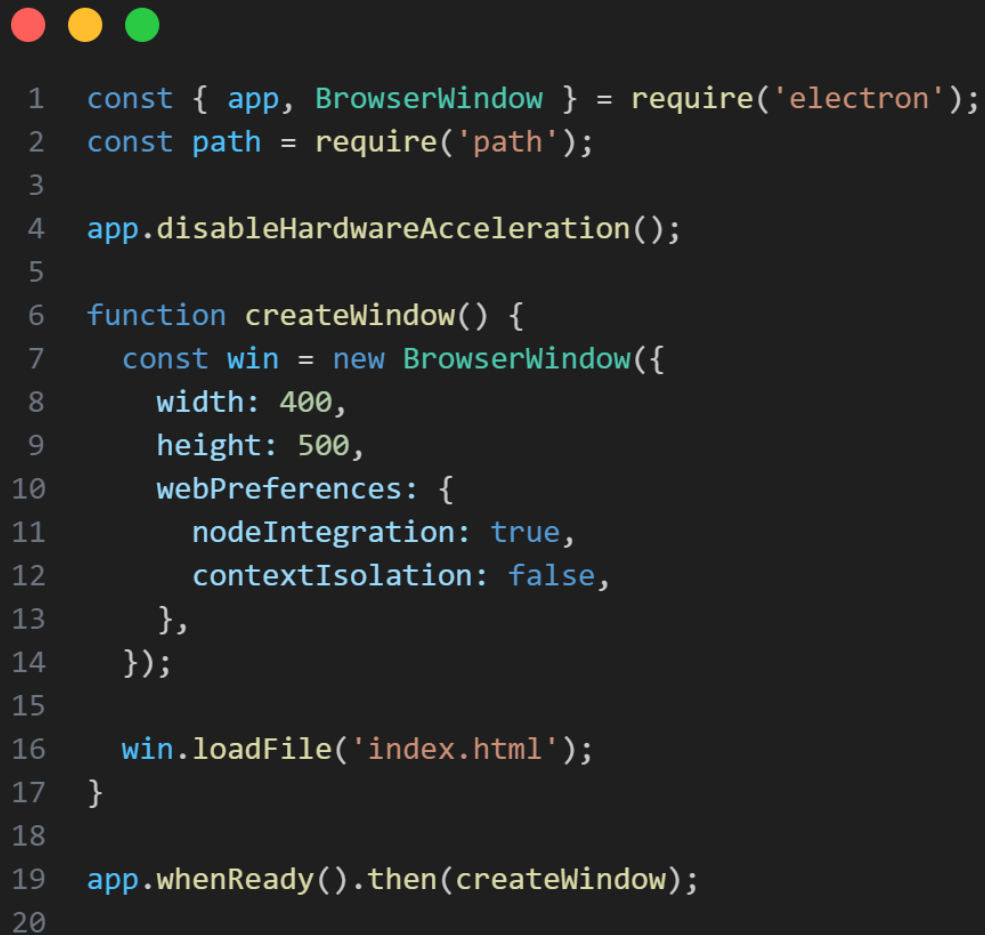
```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4    <meta charset="UTF-8">
5    <title>Secure App</title>
6  </head>
7  <body>
8    <h2>Register</h2>
9    <input id="regUsername" placeholder="Username" /><br>
10   <input id="regPassword" type="password" placeholder="Password" /><br>
11   <button id="registerBtn">Daftar</button>
12
13   <h2>Login</h2>
14   <input id="loginUsername" placeholder="Username" /><br>
15   <input id="loginPassword" type="password" placeholder="Password" /><br>
16   <button id="loginBtn">Login</button>
17
18   <pre id="message"></pre>
19
20   <script src="renderer.js"></script>
21 </body>
22 </html>
23
```

Penjelasan:

HTML di atas adalah antarmuka sederhana untuk aplikasi desktop yang dibuat dengan Electron, yang menyediakan dua fitur utama: Registrasi dan Login. Terdapat dua bagian formulir, masing-masing memiliki input untuk username dan password. Tombol "Daftar" akan memicu proses registrasi, sementara tombol "Login" akan memicu proses masuk. Elemen `<pre id="message">` digunakan untuk menampilkan pesan hasil validasi atau status kepada pengguna. Di bagian akhir, file `renderer.js` dihubungkan melalui `<script>`,

yang bertanggung jawab menangani logika interaksi pengguna dan proses penyimpanan atau verifikasi data.

Main.js



```
1  const { app, BrowserWindow } = require('electron');
2  const path = require('path');
3
4  app.disableHardwareAcceleration();
5
6  function createWindow() {
7    const win = new BrowserWindow({
8      width: 400,
9      height: 500,
10     webPreferences: {
11       nodeIntegration: true,
12       contextIsolation: false,
13     },
14   });
15
16   win.loadFile('index.html');
17 }
18
19 app.whenReady().then(createWindow);
20
```

Penjelasan:

`const { app, BrowserWindow } = require('electron');`

Mengimpor modul utama dari Electron, yaitu `app` untuk mengatur siklus hidup aplikasi, dan `BrowserWindow` untuk membuat jendela antarmuka pengguna.

`const path = require('path');`

Mengimpor modul `path` dari Node.js yang digunakan untuk menangani path file secara

cross-platform.

```
app.disableHardwareAcceleration();
```

Menonaktifkan akselerasi GPU untuk menghindari masalah seperti error GPU process (yang sering muncul di beberapa sistem).

```
function createWindow() { ... }
```

Fungsi ini membuat jendela utama aplikasi dengan ukuran 400x500 piksel.

Properti webPreferences disetel agar nodeIntegration aktif (memungkinkan penggunaan Node.js di dalam halaman HTML), dan contextIsolation dinonaktifkan (agar integrasi bekerja dengan cara lama — cocok untuk aplikasi sederhana, tapi perlu hati-hati terhadap risiko keamanan di aplikasi besar).

Memuat file HTML utama (index.html) ke dalam jendela yang dibuat.

```
app.whenReady().then(createWindow);
```

Menjalankan createWindow() saat Electron sudah siap — memastikan jendela tidak dibuat sebelum modul Electron selesai diinisialisasi.

Renderer.js

```

1  const fs = require('fs');
2  const path = require('path');
3  const crypto = require('crypto');
4
5  const DATA_FILE = path.join(__dirname, 'users.json');
6
7  if (!fs.existsSync(DATA_FILE)) {
8    fs.writeFileSync(DATA_FILE, JSON.stringify([]));
9  }
10
11 document.getElementById('registerBtn').addEventListener('click', register);
12 document.getElementById('loginBtn').addEventListener('click', login);
13
14 function hashPassword(password) {
15   return crypto.createHash('sha256').update(password).digest('hex');
16 }
17
18 function loadUsers() {
19   return JSON.parse(fs.readFileSync(DATA_FILE));
20 }
21
22 function saveUsers(users) {
23   try {
24     fs.writeFileSync(DATA_FILE, JSON.stringify(users, null, 2));
25     console.log("✅ users.json berhasil disimpan");
26   } catch (err) {
27     console.error("❌ Gagal menyimpan:", err);
28   }
29 }
30
31 function isValidUsername(username) {
32   return /^[A-Za-z]{3,20}$/.test(username); // hanya huruf A-Z, 3-20 karakter
33 }
34
35 function isValidPassword(password, username) {
36   const panjang = password.length >= 8 && password.length <= 20;
37   const simbol = /[!@#%&*]/.test(password);
38   const tidakTermasukUsername = !password.toLowerCase().includes(username.toLowerCase());
39   return panjang && simbol && tidakTermasukUsername;
40 }
41
42 function register() {
43   const username = document.getElementById('regUsername').value;
44   const password = document.getElementById('regPassword').value;
45
46   if (!isValidUsername(username)) {
47     return tampilkanPesan("Username harus 3-20 huruf alfabet A-Z/a-z");
48   }
49   if (!isValidPassword(password, username)) {
50     return tampilkanPesan("Password harus 8-20 karakter, ada simbol (!@#%&*), dan tidak mengandung username");
51   }
52
53   const users = loadUsers();
54   if (users.find(user => user.username === username)) {
55     return tampilkanPesan("Username sudah terdaftar!");
56   }
57
58   const hashed = hashPassword(password);
59   users.push({ username, password: hashed });
60   saveUsers(users);
61
62   tampilkanPesan("Registrasi berhasil!");
63 }
64
65 function login() {
66   const username = document.getElementById('loginUsername').value;
67   const password = document.getElementById('loginPassword').value;
68   const users = loadUsers();
69   const hashed = hashPassword(password);
70
71   const found = users.find(user => user.username === username && user.password === hashed);
72   if (found) {
73     tampilkanPesan("Login berhasil!");
74   } else {
75     tampilkanPesan("Login gagal. Username atau password salah.");
76   }
77 }
78
79 function tampilkanPesan(msg) {
80   document.getElementById('message').textContent = msg;
81 }
82

```

Penjelasan:

Kode `render.js` ini merupakan logika utama untuk menangani proses registrasi dan login dalam aplikasi Electron, dengan mengutamakan secure coding practices. Saat registrasi, input username dan password divalidasi menggunakan regex: username hanya boleh berisi huruf A-Z/a-z sepanjang 3–20 karakter, dan password harus 8–20 karakter, mengandung simbol khusus (!@#\$%^&*), serta tidak boleh mengandung username. Password yang valid akan di-hash menggunakan algoritma SHA-256 sebelum disimpan ke file `users.json`. Data pengguna dibaca dan ditulis menggunakan modul `fs`, dan file `users.json` akan dibuat otomatis jika belum ada. Pada proses login, password user akan di-hash dan dibandingkan dengan data yang sudah tersimpan untuk verifikasi. Semua pesan hasil aksi ditampilkan ke pengguna menggunakan fungsi `tampilkanPesan`. Pendekatan ini memastikan input tervalidasi dengan baik, password tersimpan dengan aman, dan kesalahan ditangani dengan jelas tanpa menyebabkan aplikasi crash.

Users.json

```
[
  {
    "username": "rafa",
    "password": "e681a793d302a32469cb61ea93f39de51ad77d2b3df7879a46ba40d154dfe655"
  }
]
```

Berisi penyimpanan untuk semua user

Output:

