



Laporan Progres Mingguan - **SmartBizAdmin**



Kelompok: 5

- **Muhammad Fikri Haikal Ariadma / 10231063**
- **Irfan Zaki Riyanto / 10231045**
- **Micka Mayulia Utama / 10231053**
- **Ika Agustin Wulandari / 10231041**



Mitra: Kost Al-Fitri D'Carjoe



Pekan ke-: 11



Tanggal: 25/04/2025



Progress Summary

Pada pekan ke-11, tim **SmartBizAdmin** telah menyelesaikan sistem autentikasi (login dan register) dengan validasi data dan keamanan dasar. Fitur inti pertama yang sesuai dengan kebutuhan mitra, seperti fitur sistem keuangan dan telah dibangun serta berjalan sesuai fungsinya. Kemudian integrasi antar frontend dan backend juga sudah dilakukan, memastikan data dapat diproses dan ditampilkan dengan baik. Selain itu, tim telah melakukan demo kepada mitra untuk menunjukkan perkembangan, menerima masukan dan mencatat saran perbaikan untuk iterasi berikutnya.



Accomplished Tasks

- Implementasi sistem autentikasi (login/register)
 - Implementasi fitur inti (user)
 - Integrasi frontend-backend untuk fitur yang sudah ada
 - Demo progress ke mitra
-



Challenges & Solutions

- **Challenge 1:** Menjaga keamanan sistem login agar tidak rentan terhadap serangan seperti SQL injection
☒ **Solution:** Menggunakan hash password dengan bcrypt serta prepared statements/ORM untuk pengamanan data pengguna.
 - **Challenge 2:** Data tidak langsung muncul atau terlambat tampil di frontend setelah dikirim dari backend
☒ **Solution:** Menambahkan mekanisme loading dan error handling di frontend, serta memperbaiki struktur respon API agar sinkronisasi berjalan lancar
-



Next Week Plan

- 🗓️ Implementasi fitur inti #2 dan #3
 - 🛠️ Penyempurnaan UI/UX
 - 🔧 Pengujian integrasi
 - 📅 Demo progress ke mitra
-



Contributions

- 🧑 **Muhammad Fikri Haikal Ariadma / 10231063**
→ Demo progress mitra, Desain halaman, integrasi backend dan frontend
 - 🧑 **Irfan Zaki Riyanto / 10230145**
→ Menyusun struktur backend, Menyusun route, Menghubungkan kode dengan database
 - 🧑 **Micka Mayulia Utama / 10231053**
→ Membuat laporan MD
 - 🧑 **Ika Agustin Wulandari / 10231041**
→ membuat laporan md
-



Screenshots / Demo



Config:

```
1 // config/db.js
2
3 const mysql = require('mysql2/promise'); // Import promise wrapper
4
5 // Membuat koneksi ke database MySQL
6 const pool = mysql.createPool({
7   host: 'localhost', // Ganti dengan host database Anda
8   user: 'root',       // Ganti dengan username MySQL Anda
9   password: '',       // Ganti dengan password MySQL Anda
10  database: 'smartbizadmin', // Ganti dengan nama database Anda
11  waitForConnections: true,
12  connectionLimit: 10,
13  queueLimit: 0
14 });
15
16 module.exports = pool;
17
```

Gambar diatas menunjukkan konfigurasi khusus untuk model keuangan, yang berisikan pengaturan database, model schema transaksi dan definisi variabel penting yang dibutuhkan untuk menjalankan fitur keuangan (pencatatan transaksi, pemasukkan dan pengeluaran).

Controller Keuangan:

```

1  const db = require('../config/db');
2
3  // Ambil semua data transaksi (income & expense)
4  const getAllTransactionsDetail = (req, res) => {
5      const sql = `
6          SELECT t.id, t.type, t.amount, t.description, t.category, t.payment_method,
7              DATE_FORMAT(t.created_at, '%Y-%m-%d') as tanggal,
8              u.username as created_by
9          FROM TRANSACTIONS t
10         JOIN users u ON t.created_by = u.id
11         ORDER BY t.created_at DESC
12     `;
13
14     db.query(sql, (err, results) => {
15         if (err) {
16             console.error('✖ Error mengambil data transaksi:', err);
17             return res.status(500).json({ error: 'Gagal mengambil data transaksi' });
18         }
19
20         res.status(200).json(results);
21     });
22 };
23
24 // Tambah transaksi baru
25 const createTransactionDetail = (req, res) => {
26     const { type, amount, description, category, payment_method, created_by } = req.body;
27
28     if (!type || !amount || !created_by) {
29         return res.status(400).json({ error: 'Kolom wajib (type, amount, created_by) harus diisi' });
30     }
31
32     const sql = `
33         INSERT INTO TRANSACTIONS
34             (type, amount, description, category, payment_method, created_at, updated_at, created_by)
35         VALUES (?, ?, ?, ?, ?, NOW(), NOW(), ?)
36     `;
37
38     db.query(sql, [type, amount, description, category, payment_method, created_by], (err, result) => {
39         if (err) {
40             console.error('✖ Error saat menambahkan transaksi:', err);
41             return res.status(500).json({ error: 'Gagal menambahkan data transaksi' });
42         }
43
44         res.status(201).json({
45             message: 'Transaksi berhasil ditambahkan',
46             id: result.insertId,
47         });
48     });
49 };
50
51 module.exports = {
52     getAllTransactionsDetail,
53     createTransactionDetail,
54 };
55

```

Gambar diatas menunjukkan penanganan logika bisnis dari fitur keuangan.

Controller User:

```

1 // src/controllers/userController.js
2
3 const db = require('../config/db');
4
5 // Ambil semua data pengguna
6 const getAllUsers = (req, res) => {
7   const sql = `SELECT id, username, email FROM users`; // Sesuaikan query dengan struktur tabel Anda
8
9   db.query(sql, (err, results) => {
10     if (err) {
11       console.error('✖ Error mengambil data pengguna:', err);
12       return res.status(500).json({ error: 'Gagal mengambil data pengguna' });
13     }
14     res.status(200).json(results);
15   });
16 };
17
18 module.exports = {
19   getAllUsers,
20 };
21

```

Gambar diatas menunjukkan penanganan logika bisnis dari fitur user. Untuk manajemen user

🔑 **auth, coffeeshop, inventaris, keuangan, kos:**

```

1 // routes/auth.js
2
3 const express = require('express');
4 const bcrypt = require('bcrypt');
5 const router = express.Router();
6 const pool = require('../config/db');
7 const jwt = require('jsonwebtoken'); // Import JWT untuk menghasilkan token
8
9 // Endpoint untuk registrasi
10 router.post('/register', async (req, res) => {
11   const { username, password } = req.body;
12
13   if (!username || !password) {
14     return res.status(400).send('Username dan password harus diisi');
15   }
16
17   try {
18     // Hash password sebelum disimpan ke database
19     const hashedPassword = await bcrypt.hash(password, 10);
20
21     // Simpan username dan hashed password ke database
22     await pool.query('INSERT INTO users (username, password) VALUES (?, ?)', [username, hashedPassword]);
23
24     res.status(201).send(`User ${username} berhasil didaftarkan`); // Use 201 Created status
25   } catch (err) {
26     console.error('Error saat registrasi:', err); // Log the actual error to the console
27
28     // Check for specific MySQL duplicate entry error
29     if (err.code === 'ER_DUP_ENTRY') {
30       res.status(409).send('Username sudah digunakan'); // 409 Conflict for duplicate
31     } else {
32       // Send a generic server error for other issues
33       res.status(500).send('Terjadi kesalahan pada server saat mendaftarkan user');
34     }
35   }
36 });

```

```

37
38 // Endpoint untuk Login
39 router.post('/login', async (req, res) => {
40   const { username, password } = req.body;
41
42   if (!username || !password) {
43     return res.status(400).send('Username dan password harus diisi');
44   }
45
46   try {
47     // Cari user berdasarkan username
48     const [rows] = await pool.query('SELECT * FROM users WHERE username = ?', [username]);
49
50     if (rows.length > 0) {
51       const user = rows[0];
52
53       // Verifikasi password dengan bcrypt
54       const isMatch = await bcrypt.compare(password, user.password);
55
56       if (isMatch) {
57         // Jika password cocok, kirimkan token
58         const token = jwt.sign({ username: user.username }, 'your-secret-key', { expiresIn: '1h' });
59         res.json({ message: 'Login berhasil', token });
60       } else {
61         res.status(401).send('Password salah');
62       }
63     } else {
64       res.status(401).send('Username atau password salah'); // More generic message for security
65     }
66   } catch (err) {
67     console.error('Error saat login:', err); // Log the actual error
68     res.status(500).send('Terjadi kesalahan pada server saat login');
69   }
70 });
71
72 module.exports = router;
73

```

Gambar diatas adalah keseluruhan fitur utama dari sistem smartbiz admin

user routes:

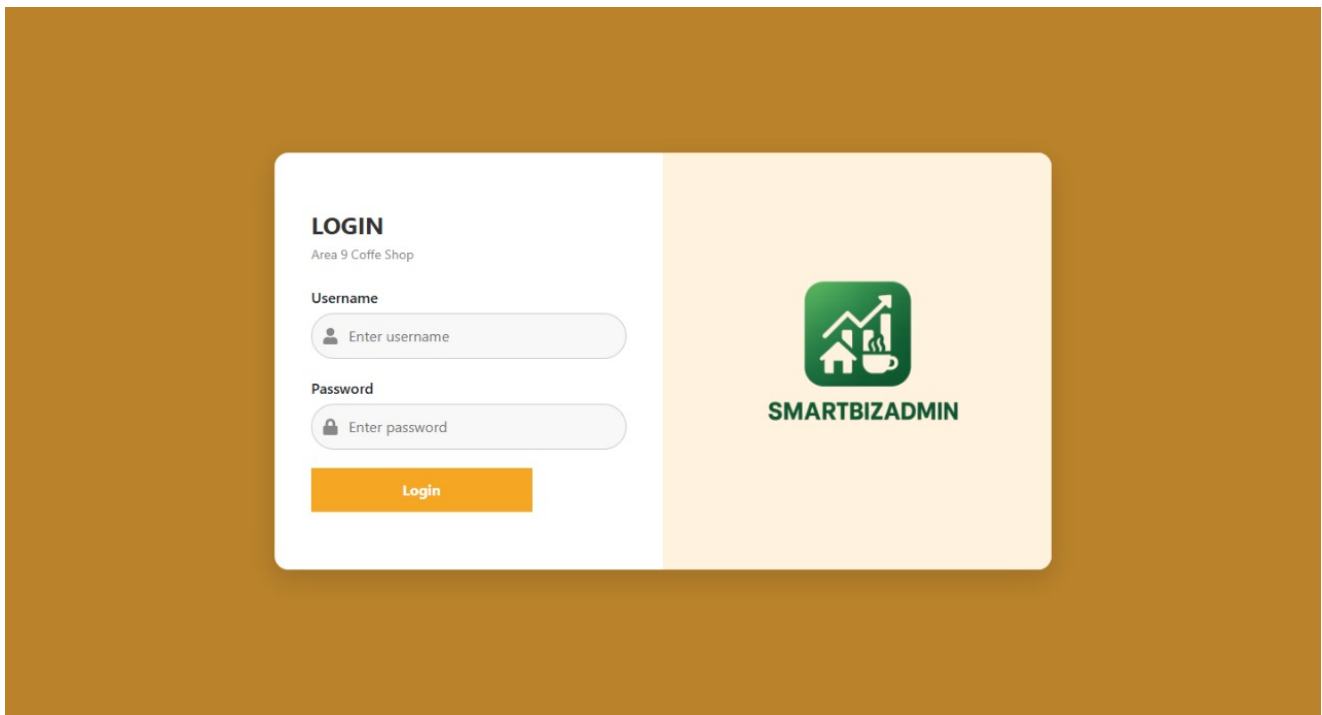
```

1 // src/routes/userRoutes.js
2
3 const express = require('express');
4 const router = express.Router();
5 const userController = require('../controllers/UserController');
6
7 // Endpoint untuk mendapatkan data pengguna
8 router.get('/users', userController.getAllUsers);
9
10 module.exports = router;
11

```

Gambar diatas adalah endpoint API yang terhubung ke user controller

Halaman Login:



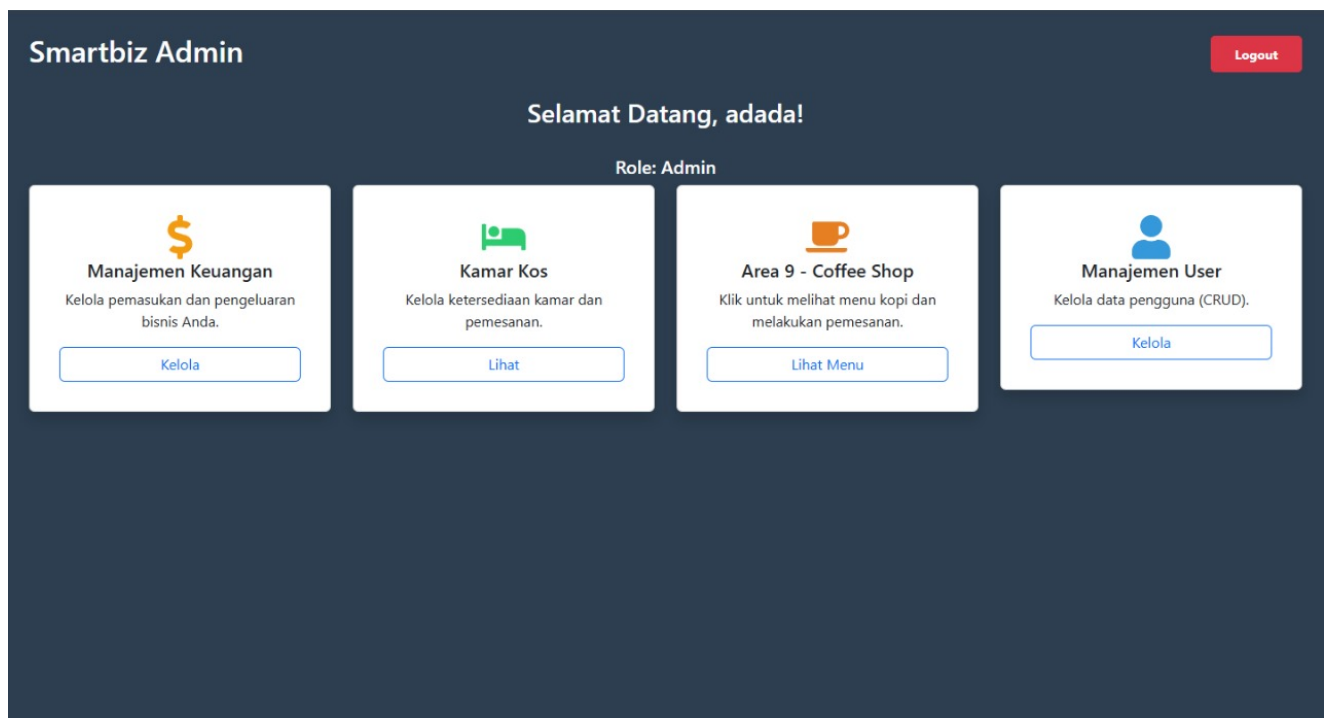
Gambar diatas adalah halaman untuk autentikasi pengguna dari login

Halaman Register:

Login'. On the right, a light orange box contains the 'SMARTBIZADMIN' logo, which features a green square with a white house, coffee cup, and bar chart icon." data-bbox="83 478 916 796"/>

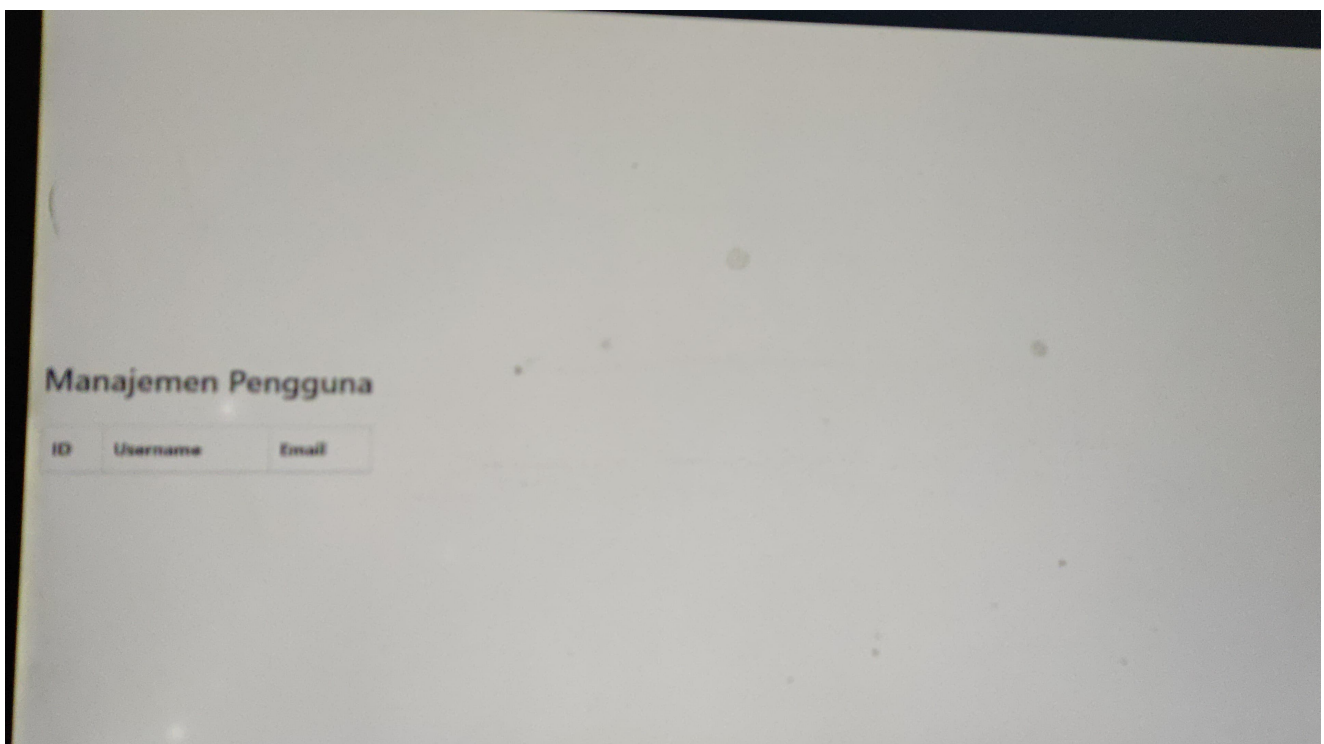
Gambar diatas adalah halaman untuk autentikasi pengguna dari register

Halaman Dashboard Admin:



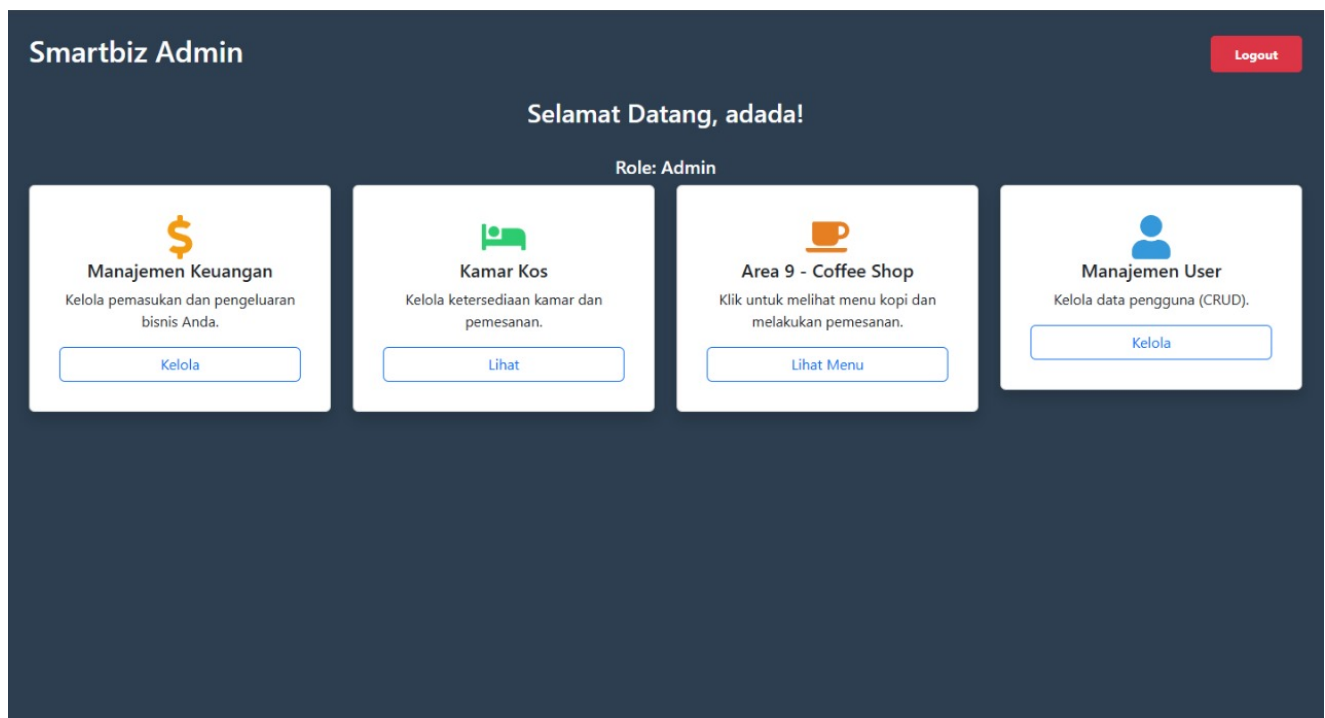
Gambar diatas adalah tampilan utama setelah login admin. Menampilkan ringkasan data penting menu fitur.

Halaman Manajemen User:



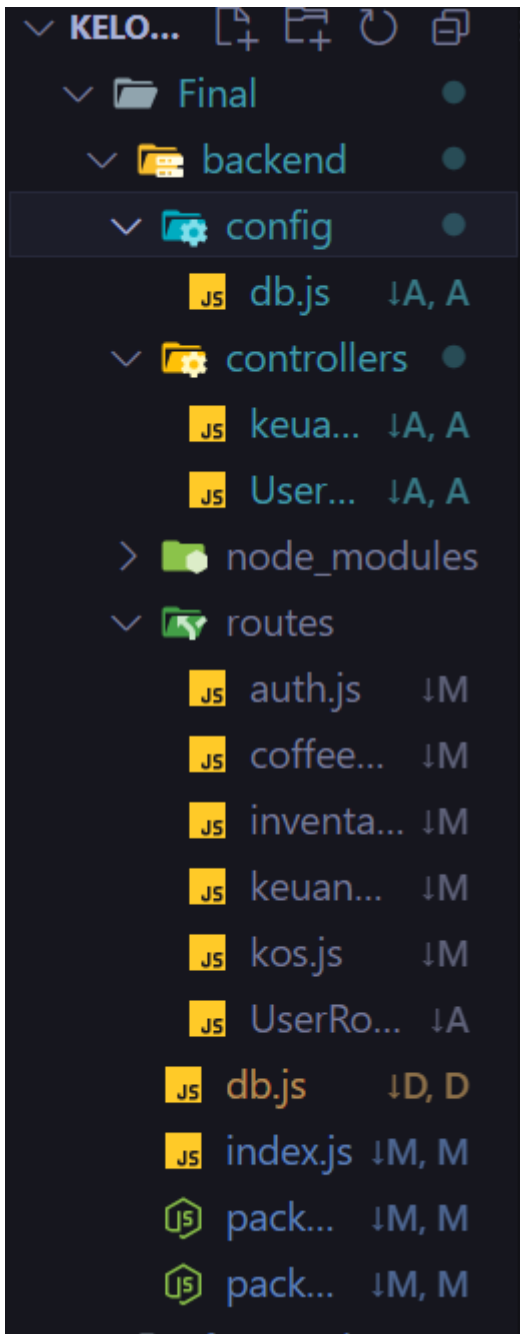
Fitur ini memungkinkan admin untuk melihat daftar user, mengedit profil user, atau menghapus user dari sistem. Biasanya ada tabel dengan kolom nama, email, role, dan tombol aksi.

Penambahan Fungsi Logout:



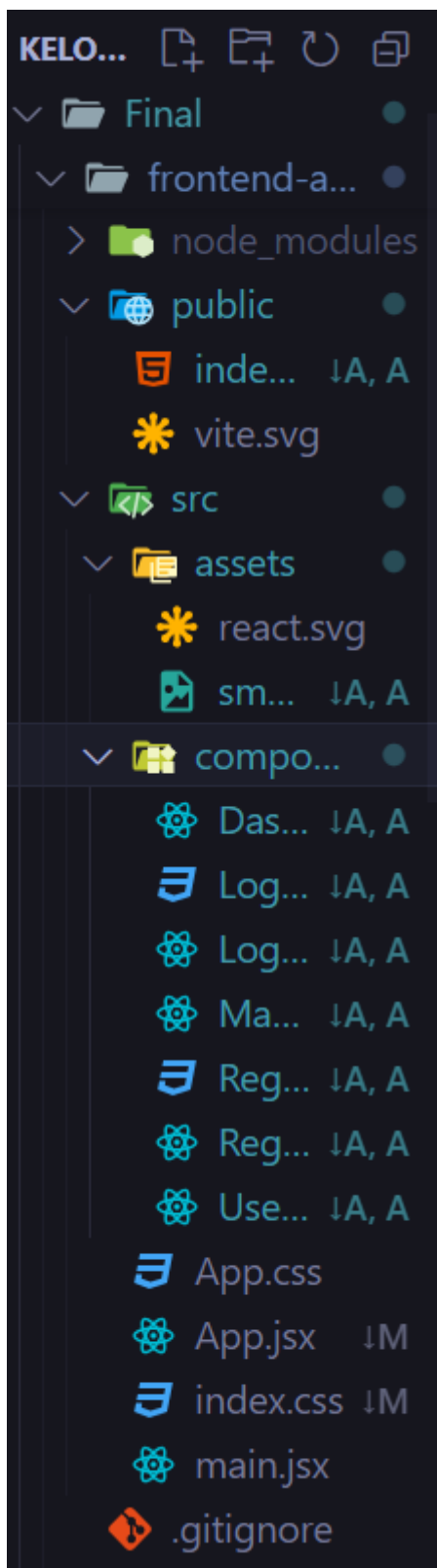
Logout yang berada di dalam dashboard admin terletak di kanan atas, memungkinkan pengguna keluar dari sesi aktif mereka, menghapus token autentikasi, dan mengarahkan kembali ke halaman login. Ini penting untuk keamanan sistem.

 **backend yang baru:**



Menandakan bahwa struktur backend telah diperbarui agar lebih modular dan efisien, memisahkan fitur berdasarkan fungsinya (seperti auth, keuangan, dll), serta menyederhanakan integrasi dengan frontend melalui API yang konsisten.

Frontend yang baru:



Menandakan bahwa struktur frontend telah diperbarui agar lebih modular dan efisien, memisahkan fitur berdasarkan fungsinya (seperti auth, keuangan, dll), serta menyederhanakan integrasi dengan frontend melalui API yang konsisten.

Demo Progress ke Mitra:

Setelah menghubungi mitra kami melakukan perjanjian untuk bertemu pada Jumat, 25 April 2025 di malam hari .
