Shanika Munasinghe

Shanika.uok2@gmail.com

E-Pass book

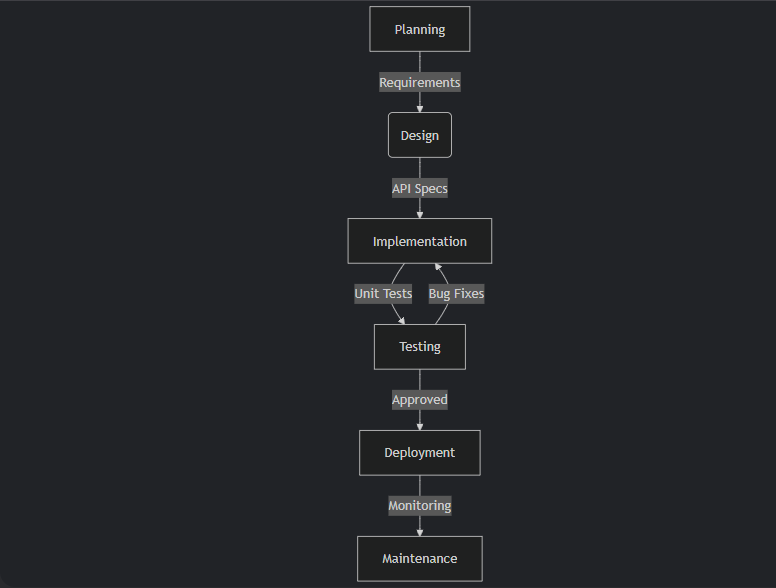
Project Report

**Table of Contents**

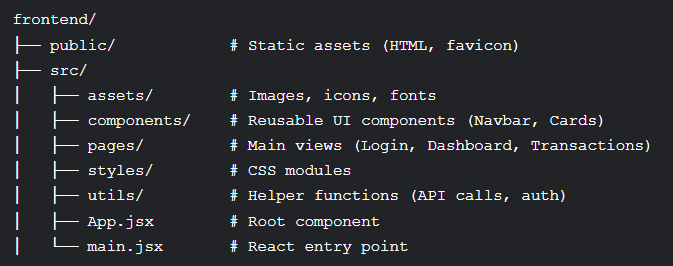
1. **Introduction**
2. **Frontend Documentation**
   * Project Structure
   * Key Components
   * Pages & Routing
   * Styling System
   * State Management
   * API Integration
3. **Backend Documentation**
   * Project Structure
   * API Endpoints
   * Authentication Flow
   * Database Models
   * Error Handling
4. **Setup & Deployment**
5. **Future Enhancements**
6. **Contributing & License**
7. **Introduction**

This document provides comprehensive documentation for the E- Banking platform, a react based web application with a Node.js backend . it includes user authentication, account management , transaction history and financial services.

**SDLC Workflow troubleshooting**



1. **Frontend Documentation**
   * **Project Structure**



* + **Key Components**

|  |  |
| --- | --- |
| Components | Description |
| Navbar.jsx | Navigation bar with user info |
| AccountSummary.jsx | Display account summary |
| Home.jsx | Display all the options and bank pass book interface |
| Login.jsx | Login window |
| Register.jsx | Registration Form and Interface |
| Transactions.jsx | Transaction Page |
| Welcome.jsx | Welcome Page |
| Auth.js | Handles login / Register forms |

* + **Pages & Routing**

|  |  |  |
| --- | --- | --- |
| Page | Route | Description |
| Welcome | / | Landing page with login button |
| Login.jsx | /login | User authentication |
| Register.jsx | /register | New user signup |
| Home.jsx | /home | Dashboard with account overview |
| AccountSummary.jsx | /accounts | List of all user accounts |
| Transactions.jsx | /transactions/:id | Transaction history per account |

**Routing setup (App.jsx)**

<Routes>

<Route path="/" element={<Welcome />} />

<Route path="/login" element={<Login />} />

<Route path="/register" element={<Register />} />

<Route element={<RequireAuth />}>

<Route path="/home" element={<Home />} />

<Route path="/accounts" element={<AccountSummary />} />

<Route path="/transactions/:id" element={<Transactions />} />

</Route>

</Routes>

* + **Styling System**
    - CSS Modules for scoped styling
    - Responsive design using Flexbox/Grid
    - Theming (Dark mode)
  + **State Management**
    - React context for user authentication state
    - Local state for forms and UI interactions
    - React router for navigation
  + **API Integration**
    - Axios for HTTP requests
    - JWT Authentication (Token stored in localStorage)
    - Sample API Call (Login):

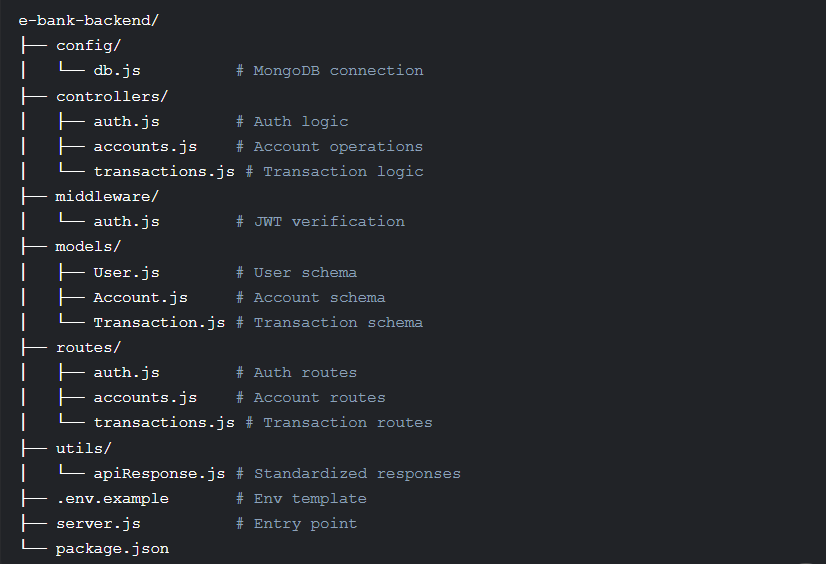
export const loginUser = async (credentials) => {

const response = await axios.post('/api/auth/login', credentials);

return response.data;

};

1. **Backend Documentation**
   * **Project Structure**



* + **Environment setup**

**Requirements**

* Node.js v18+
* MongoDB Atlas account
* Postman (for API testing)

**Installation**

git clone https://github.com/your-repo/e-bank-backend.git

cd e-bank-backend

npm install

**configuration**

1. Rename .env.example to .env
2. Configure variables:

MONGODB\_URI=mongodb+srv://<username>:<password>@cluster0.mongodb.net/e-bank

JWT\_SECRET=your\_jwt\_secret\_here

JWT\_EXPIRES\_IN=1d

PORT=5000

**Run modes**

**Command description**

npm start production mode

npm run dev development (with nodemon)

npm test run test

* + **Database Models**
    1. **User model**

{

username: { type: String, unique: true, required: true },

email: { type: String, unique: true, required: true },

password: { type: String, select: false },

accounts: [{ type: mongoose.Schema.Types.ObjectId, ref: 'Account' }],

createdAt: { type: Date, default: Date.now }

}

* + 1. **Account model**

{

user: { type: mongoose.Schema.Types.ObjectId, ref: 'User' },

accountNumber: { type: String, unique: true },

type: { type: String, enum: ['Savings', 'Current'] },

balance: { type: Number, default: 0 }

}

* + 1. **Transaction model**

{

fromAccount: { type: mongoose.Schema.Types.ObjectId, ref: 'Account' },

toAccount: { type: mongoose.Schema.Types.ObjectId, ref: 'Account' },

amount: { type: Number, required: true },

type: { type: String, enum: ['Deposit', 'Withdrawal', 'Transfer'] },

timestamp: { type: Date, default: Date.now }

}

* + **API Endpoints**

Authentication

|  |  |  |
| --- | --- | --- |
| Endpoint | Method | Description |
| /api/auth/register | POST | Register new user |
| /api/auth/login | POST | User login |

Sample request (Register):

{

"username": "john\_doe",

"email": "john@example.com",

"password": "securePassword123"

}

**Accounts**

|  |  |  |
| --- | --- | --- |
| End point | Method | Description |
| /api/accounts | POST | Create new account |
| /api/accounts | GET | List user accounts |

Sample response:

{

"status": "success",

"data": {

"account": {

"accountNumber": "1234567890",

"type": "Savings",

"balance": 5000

}

}

}

* + **Authentication flow**

A screenshot of a computer

AI-generated content may be incorrect.

* + **Error Handling**

**Standard error response:**

{

"status": "error",

"message": "Account not found",

"code": 404

}

**Common error codes :**

|  |  |
| --- | --- |
| **Code** | **Description** |
| 400 | Bad request |
| **401** | **Unauthorized** |
| **404** | **Not found** |
| **500** | **Server error** |

* + **Testing**

**Run unit tests**

npm test

**test coverage**

* Authentication (100%)
* Account operations (85%)
* Transactions (75%)
  + **Troubleshooting**

|  |  |
| --- | --- |
| **Issue** | **Solution** |
| MongoDB Connection fails | Verify atlas IP whitelist |
| JWT Errors | Check token expiration and secret |
| CORS Issues | Ensure app.use(cors()) is enabled |
| Validation errors | Review mongoose schemas |

1. **Future Enhancements**

**Planned Features:**

* Fund transfers between accounts
* Bill payment integration
* PDF statement generation
* Admin dashboard