BANK MANAGEMENT (MERN)

Introduction:

Project Name: **Bank Buddy**

Team Members:

1. K Leelamohan (Backend Developer)
2. N Suresh Reddy (Supporter)
3. T Srija Devi (Front End Developer)
4. R Naga Sridevi (Testing)

Description:

Welcome to MS Bank, your all-in-one digital platform designed to make managing your money easier than ever. With our super-friendly app, you can take care of all your banking needs without any hassle.

Our web app is easy to use and lets you access a wide range of banking services tailored to your preferences. Whether you're a banking pro or just starting out, you'll find it a breeze to manage your finances.

You can get detailed information about your accounts and transactions right at your fingertips. Check your balances, view your transaction history, and access your account statements whenever you need. No more guessing or uncertainty – MS Bank makes everything clear and simple.

Signing up and logging in is a piece of cake. Just provide your personal information, contact details, and create your login credentials. Once you're registered, you can securely access your account dashboard.

From the comfort of your home, you can transfer money to others using their account ID, make deposits, and apply for loans with ease. Our real-time updates ensure that your transactions show up in your account immediately.

Applying for a loan is now super simple. Just submit your loan application online, and our admin team will process it quickly. You can track the status of your loan application through your account dashboard.

For bank administrators, our intuitive admin dashboard makes it easy to manage and oversee customer accounts and transactions. They can view user account details, transaction history, and loan applications. We have separate login and registration pages for bank staff to ensure privacy and security.

MS Bank is here to make your banking experience smooth and convenient. With our user-friendly interface, efficient account management, and robust admin features, we provide a hassle-free and enjoyable banking experience for both customers and bank administrators.

Features:

Comprehensive Banking Services: MS Bank offers a wide range of banking options to meet various financial needs. You can manage your accounts, transfer money, make deposits, apply for loans, and more – all easily from one place.

Easy Account Management: With MS Bank, managing your accounts is a breeze. The user-friendly dashboard lets you check your balances, view transaction histories, and access statements effortlessly.

Quick and Secure Transactions: Our app guarantees fast and secure money transfers. Whether you're sending money to friends, family, or making payments, MS Bank ensures your transactions are safe and speedy.

Convenient Deposits: Making deposits is simple with MS Bank. You can easily add funds to your accounts, and the app keeps track of the maturity period for easy withdrawals.

Loan Application Made Simple: Applying for loans is straightforward with MS Bank. You can submit your loan applications online, and our admin team processes them efficiently. You can track your loan status through your account dashboard.

Dedicated Admin Dashboard: Bank administrators have a powerful dashboard to manage the system effectively. They can view user details, monitor transactions, approve loan applications, and oversee the entire banking process.

Real-time Updates: MS Bank provides real-time updates for all transactions, deposits, and loan applications. Both users and administrators receive instant notifications, ensuring a transparent and up-to-date banking experience.

Purpose:

Customer Satisfaction: Providing excellent customer service and ensuring customers have a positive banking experience.

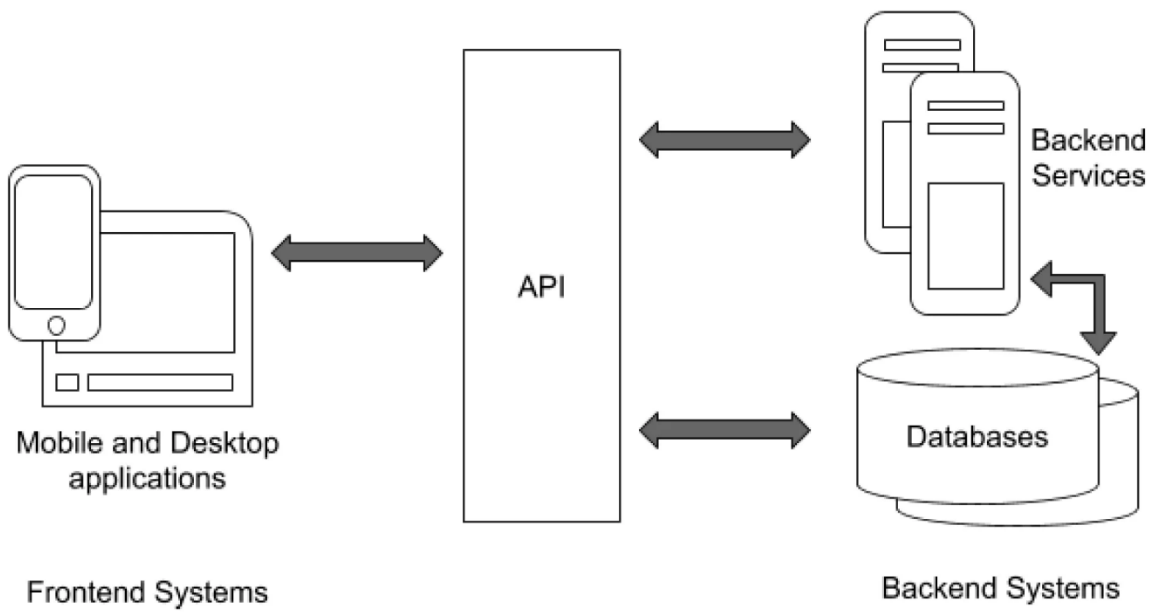
Financial Stability: Maintaining the financial health of the bank by managing assets, liabilities, and capital effectively.

Risk Management: Identifying, assessing, and mitigating risks to protect the bank's assets and ensure regulatory compliance.

Profitability: Generating profits through various banking activities while managing costs and expenses.

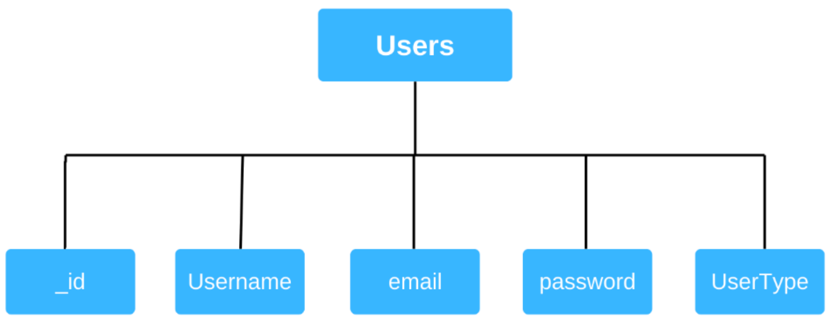
Compliance: Ensuring the bank adheres to laws, regulations, and industry standards to avoid legal issues and penalties.

Architecture:

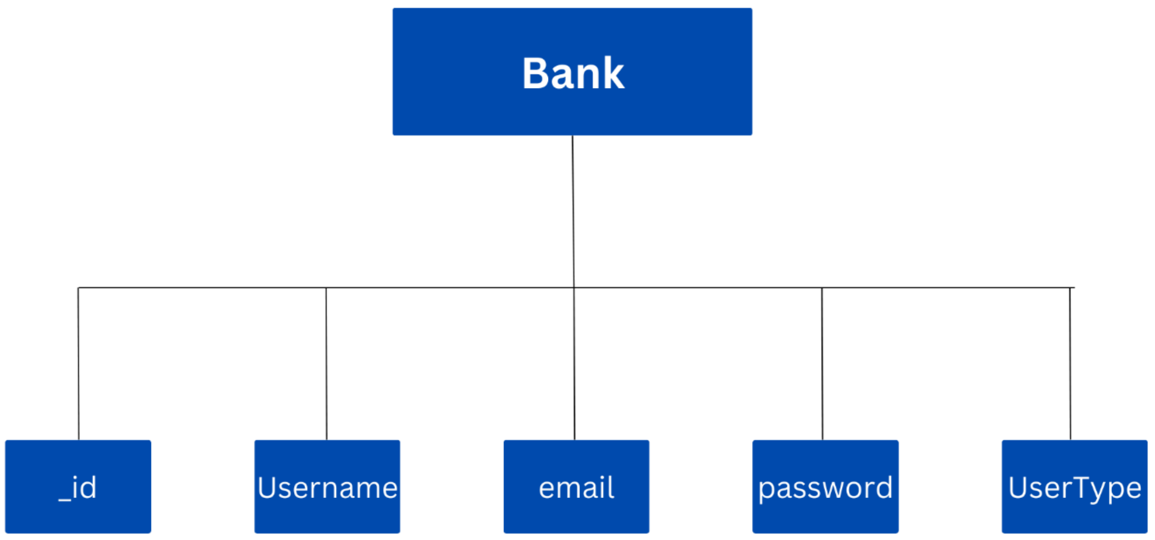


ER Diagrams

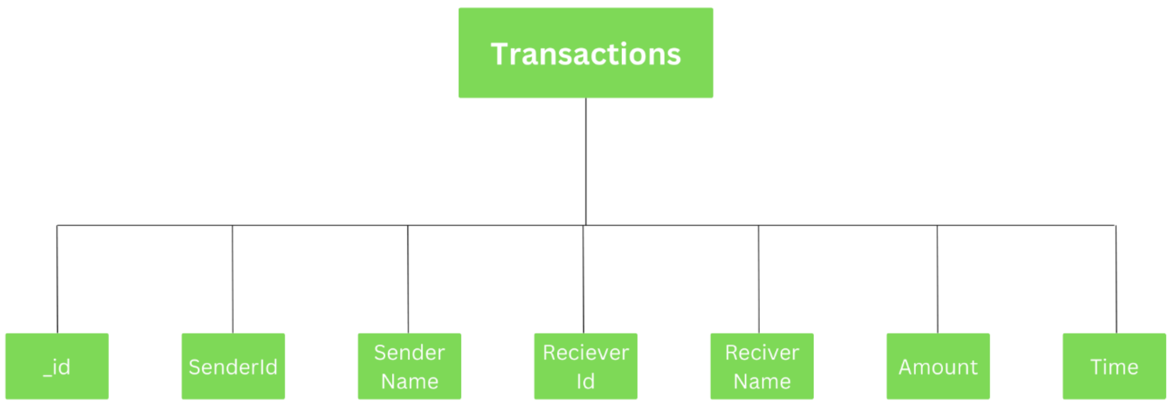
Users



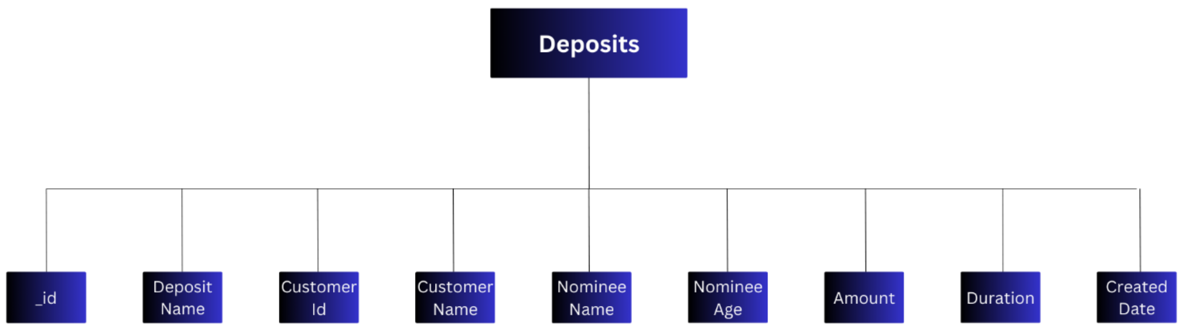
Bank



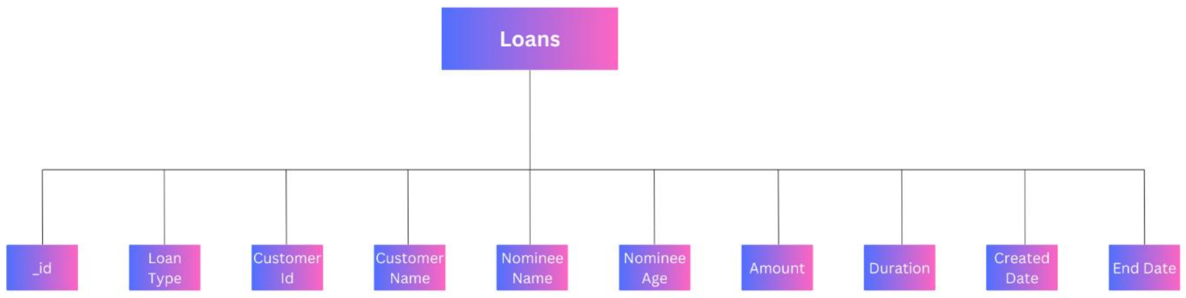
Transactions



Deposits



Loans



Installation:

Install Node.js and npm

First, you need to install Node.js, which includes npm (Node Package Manager).

- Download and install Node.js from here.

Set Project Directory

Create a new directory for your project:

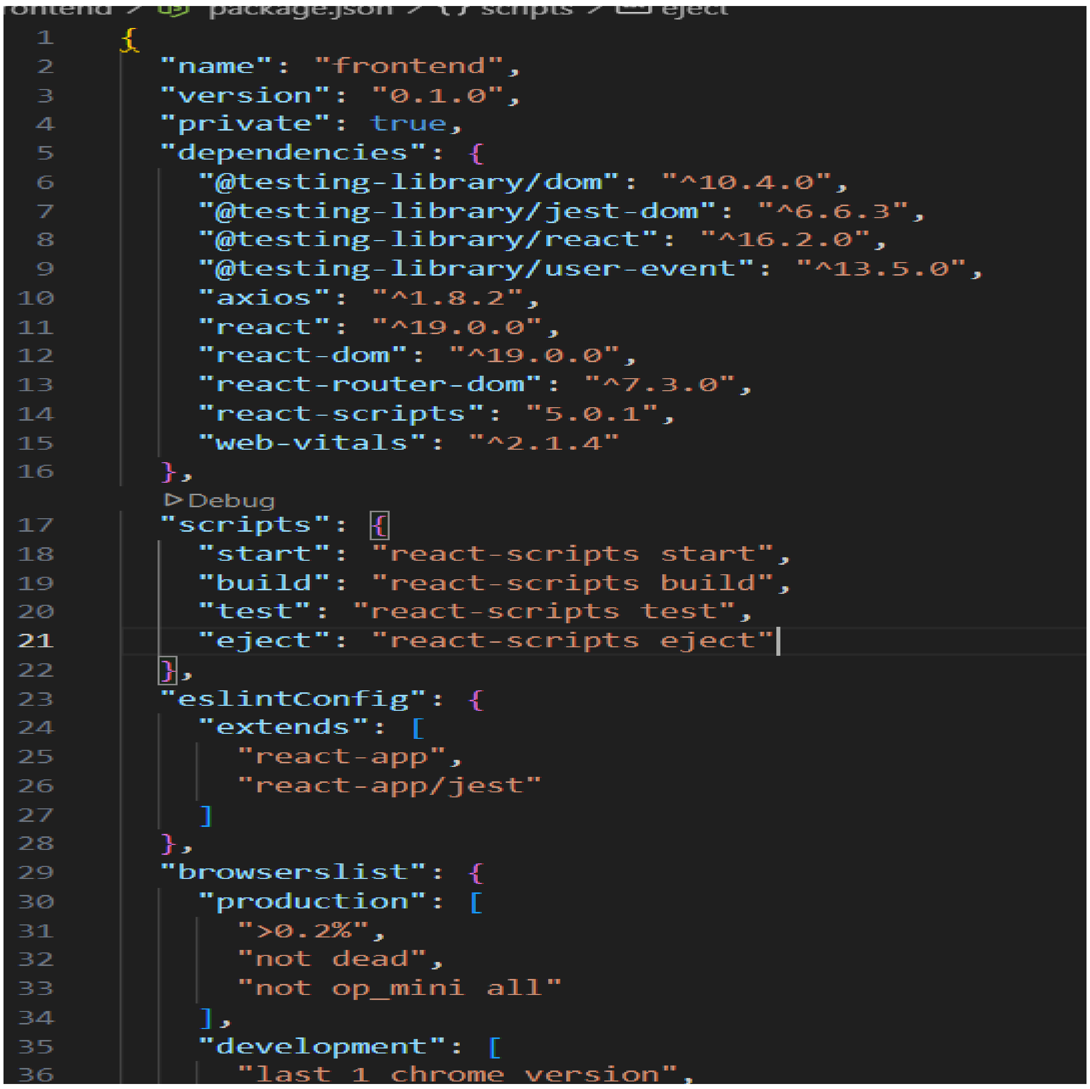
mkdir my-mern-app

cd my-mern-app

Initialize your Project

Initialize a new Node.js project with npm:

npm init -y



```json
{
  "name": "frontend",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "@testing-library/dom": "^10.4.0",
    "@testing-library/jest-dom": "^6.6.3",
    "@testing-library/react": "^16.2.0",
    "@testing-library/user-event": "^13.5.0",
    "axios": "^1.8.2",
    "react": "^19.0.0",
    "react-dom": "^19.0.0",
    "react-router-dom": "^7.3.0",
    "react-scripts": "5.0.1",
    "web-vitals": "^2.1.4"
  },
  "scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build",
    "test": "react-scripts test",
    "eject": "react-scripts eject"
  },
  "eslintConfig": {
    "extends": [
      "react-app",
      "react-app/jest"
    ]
  },
  "browserslist": {
    "production": [
      ">0.2%",
      "not dead",
      "not op_mini all"
    ],
    "development": [
      "last 1 chrome version",
```

Install Backend Dependencies

Install Express.js and other necessary backend packages:

npm install express mongoose

Create Backend Files

Create a basic structure for your backend:

- server.js: Main server file
- models/: Directory for Mongoose models
- routes/: Directory for Express routes

Install Frontend Dependencies

Install Create React App to set up the frontend:

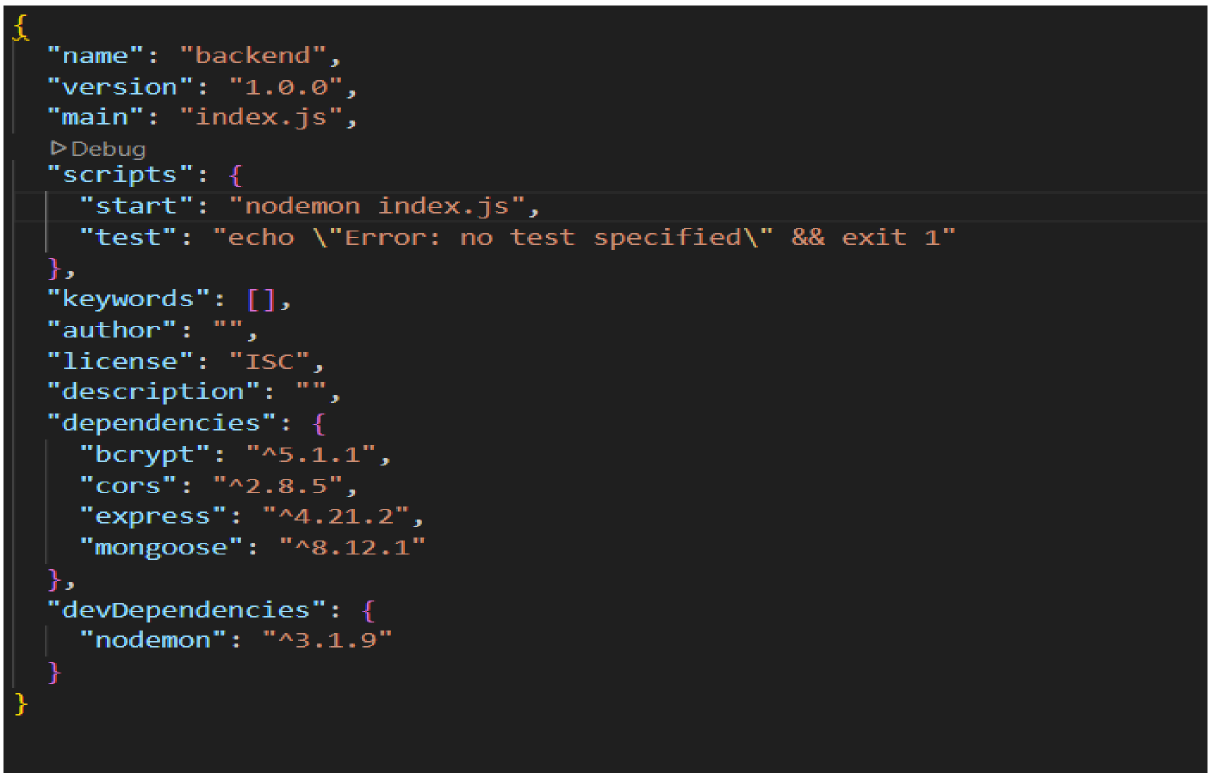
npx create-react-app client

cd client

Run the React Application

Start the React development server:

npm start



```
{
  "name": "backend",
  "version": "1.0.0",
  "main": "index.js",
  ▷Debug
  "scripts": {
    "start": "nodemon index.js",
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "description": "",
  "dependencies": {
    "bcrypt": "^5.1.1",
    "cors": "^2.8.5",
    "express": "^4.21.2",
    "mongoose": "^8.12.1"
  },
  "devDependencies": {
    "nodemon": "^3.1.9"
  }
}
```

Connect Frontend to Backend

In the React app, use Axios or Fetch to make API calls to your Express server. For example, create a new file api.js in your React app:

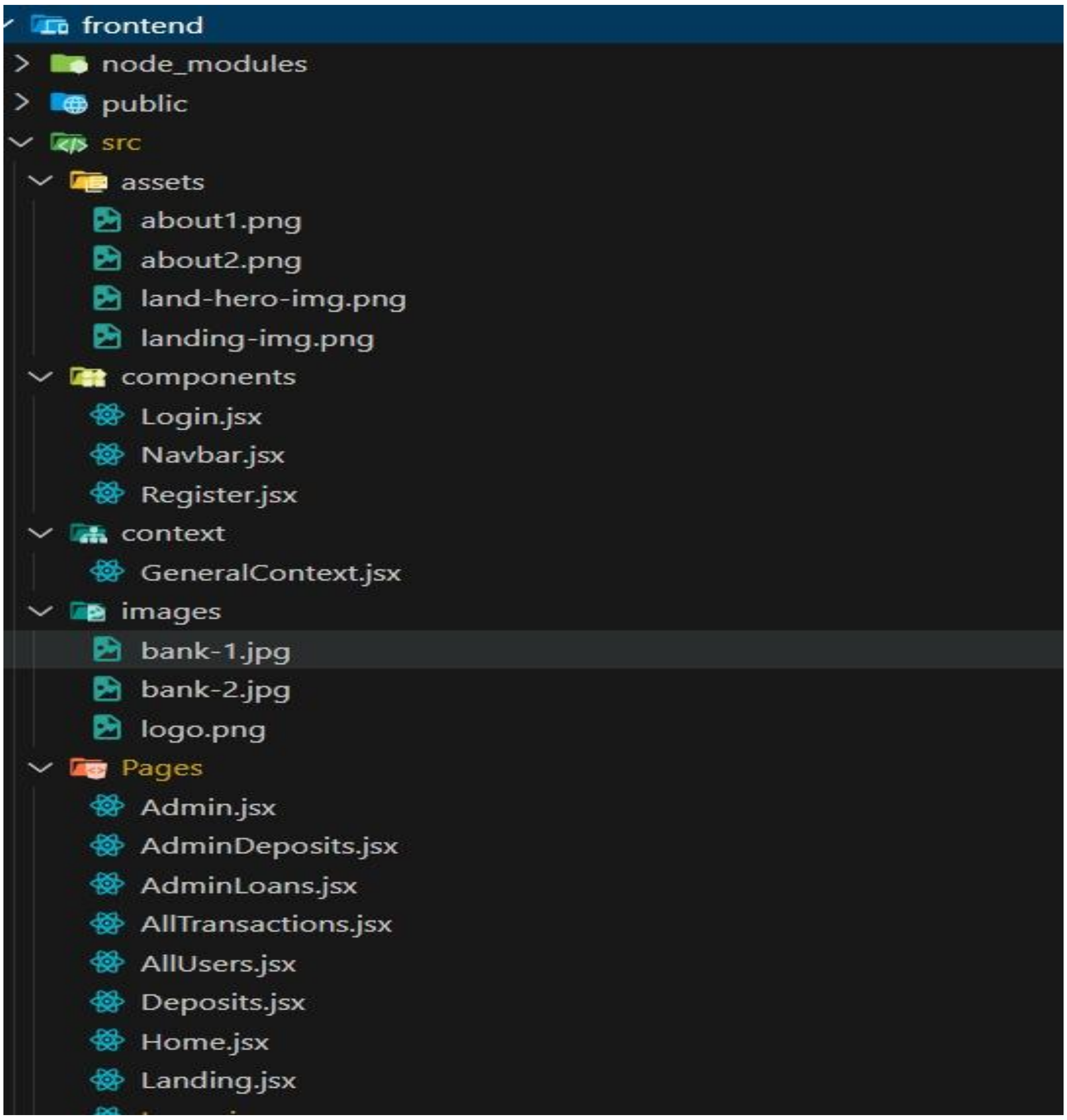
Start Both Servers

Run both servers (backend and frontend) simultaneously. You can use concurrently:

npm start

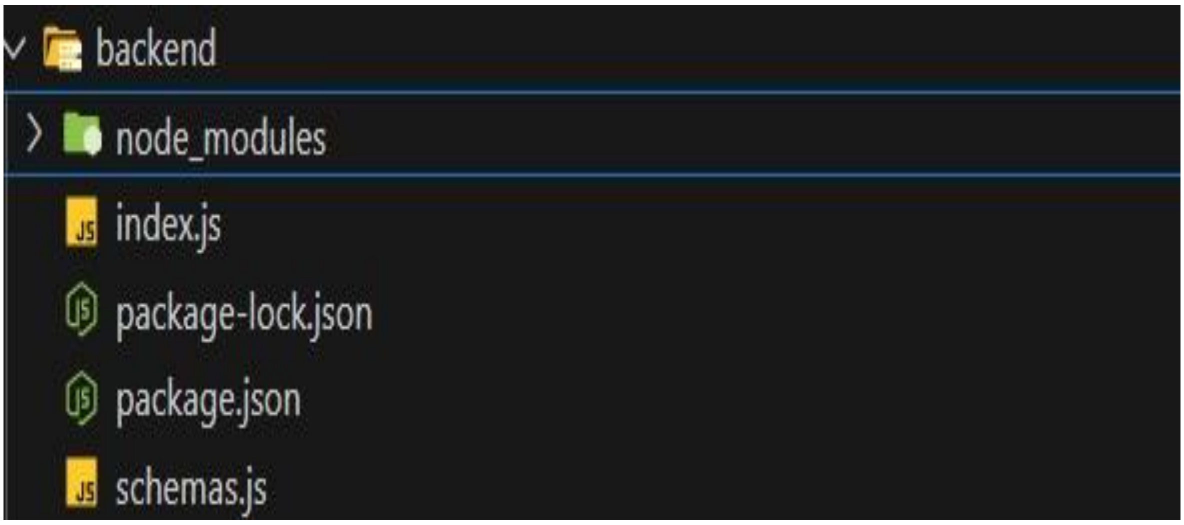
Folder Structure:

Front end:



```
frontend
  > node_modules
  > public
  v src
    v assets
        about1.png
        about2.png
        land-hero-img.png
        landing-img.png
    v components
        Login.jsx
        Navbar.jsx
        Register.jsx
    v context
        GeneralContext.jsx
    v images
        bank-1.jpg
        bank-2.jpg
        logo.png
    v Pages
        Admin.jsx
        AdminDeposits.jsx
        AdminLoans.jsx
        AllTransactions.jsx
        AllUsers.jsx
        Deposits.jsx
        Home.jsx
        Landing.jsx
```

Backend:



```
backend
  node_modules
  index.js
  package-lock.json
  package.json
  schemas.js
```

Running the Application:

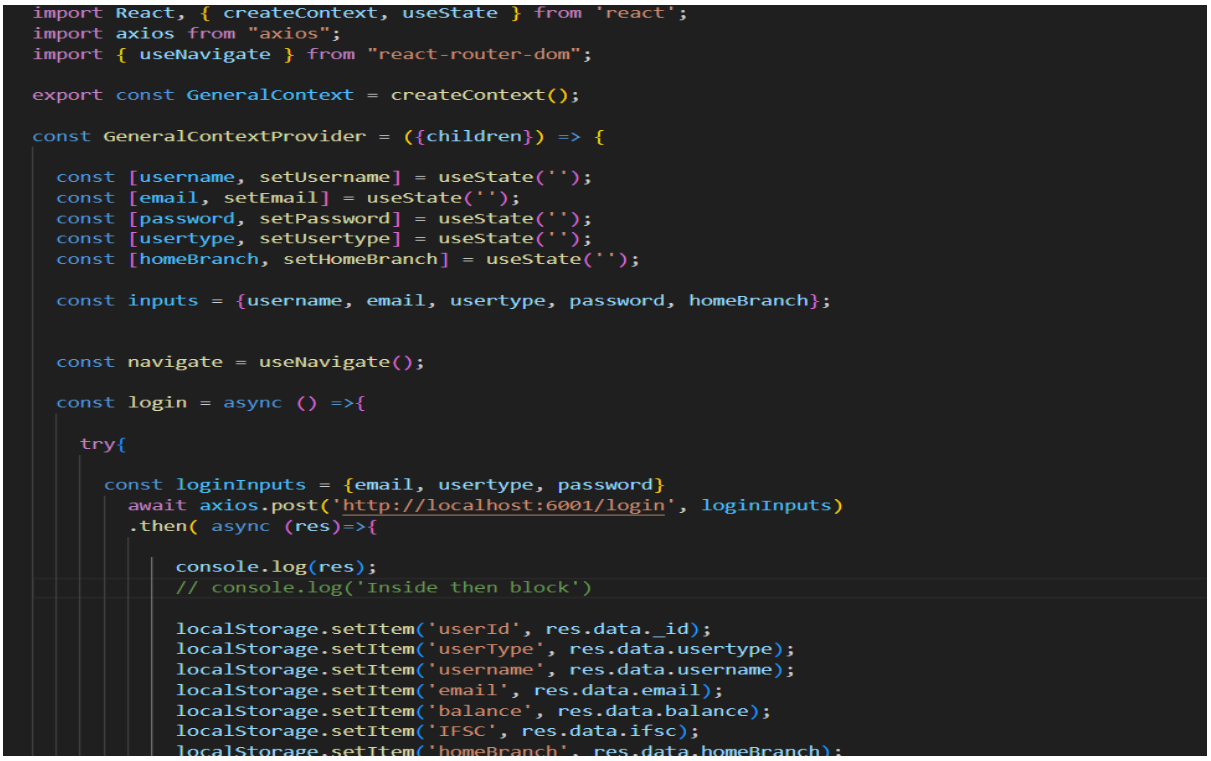
Frontend (Client) Server:

Navigate to the client directory in your terminal.

Run the following command to start the frontend server:

npm start

This will start the frontend application, usually at http://localhost:3000.



```
import React, { createContext, useState } from 'react';
import axios from "axios";
import { useNavigate } from "react-router-dom";

export const GeneralContext = createContext();

const GeneralContextProvider = ({children}) => {

  const [username, setUsername] = useState('');
  const [email, setEmail] = useState('');
  const [password, setPassword] = useState('');
  const [usertype, setUsertype] = useState('');
  const [homeBranch, setHomeBranch] = useState('');

  const inputs = {username, email, usertype, password, homeBranch};


  const navigate = useNavigate();

  const login = async () =>{

    try{

      const loginInputs = {email, usertype, password}
        await axios.post('http://localhost:6001/login', loginInputs)
        .then( async (res)=>{

            console.log(res);
            // console.log('Inside then block')

            localStorage.setItem('userId', res.data._id);
            localStorage.setItem('userType', res.data.usertype);
            localStorage.setItem('username', res.data.username);
            localStorage.setItem('email', res.data.email);
            localStorage.setItem('balance', res.data.balance);
            localStorage.setItem('IFSC', res.data.ifsc);
            localStorage.setItem('homeBranch', res.data.homeBranch);
```

Backend (Server) Server:

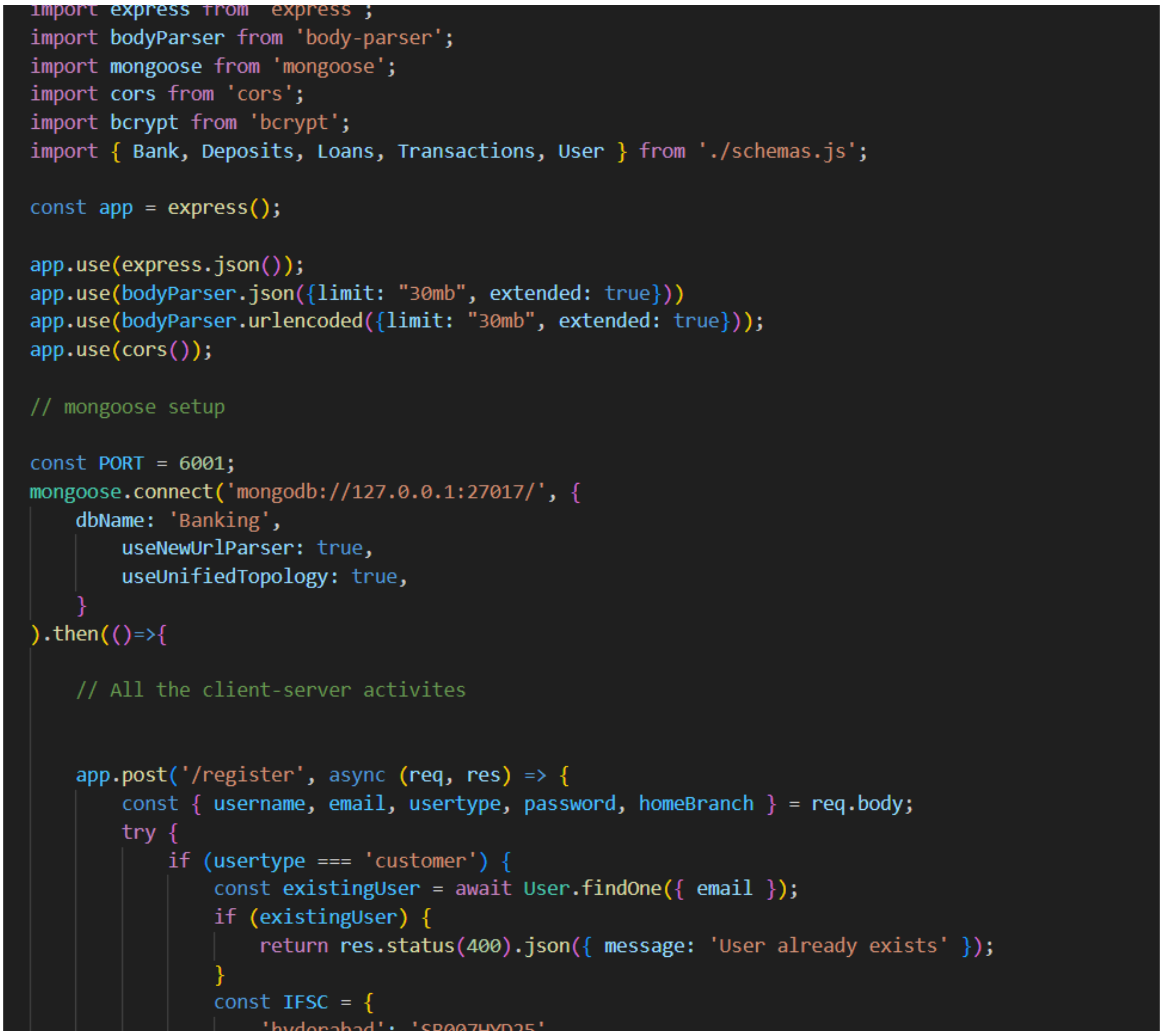
Navigate to the server directory in your terminal.

Run the following command to start the backend server:

npm start

This will start the backend API server, usually at http://localhost:5000 (or another port if specified).

Ensure that you have all the dependencies installed by running npm install in both the client and server directories before starting the servers.



```
import express from 'express';
import bodyParser from 'body-parser';
import mongoose from 'mongoose';
import cors from 'cors';
import bcrypt from 'bcrypt';
import { Bank, Deposits, Loans, Transactions, User } from './schemas.js';

const app = express();

app.use(express.json());
app.use(bodyParser.json({limit: "30mb", extended: true}))
app.use(bodyParser.urlencoded({limit: "30mb", extended: true}));
app.use(cors());

// mongoose setup

const PORT = 6001;
mongoose.connect('mongodb://127.0.0.1:27017/', {
    dbName: 'Banking',
        useNewUrlParser: true,
        useUnifiedTopology: true,
    }
).then(()=>{

    // All the client-server activites


    app.post('/register', async (req, res) => {
        const { username, email, usertype, password, homeBranch } = req.body;
        try {
            if (usertype === 'customer') {
                const existingUser = await User.findOne({ email });
                if (existingUser) {
                    return res.status(400).json({ message: 'User already exists' });
                }
                const IFSC = {
                    'hyderabad': 'SB007HYD25'
```

API Documentation:

When building a web application or an API, it's essential to have a well-organized codebase. This not only makes the application easier to maintain but also enhances its scalability and readability. One of the best practices to achieve this is by separating your route files based on different functionalities. Here's the theory behind it:

1. Modularity and Separation of Concerns

By creating separate route files for different API functionalities, you adhere to the principle of modularity and separation of concerns. This means each module or file is responsible for a specific part of the application. For example, authentication, loans, deposits, and transactions are different functionalities, so they should be managed independently.

2. Easier Maintenance

When your route files are separated, it's easier to maintain and update your code. If you need to make changes to the authentication logic, you can do so without affecting the loan or deposit functionalities. This isolation helps in quickly identifying and fixing bugs or adding new features.

3. Improved Readability and Organization

A well-structured codebase is easier to read and understand. When you separate your routes into different files, it becomes clear where to find and manage specific functionality. This organization helps new developers onboard quickly and understand the project structure.

4. Scalability

As your application grows, the number of routes and functionalities will increase. Having separate route files allows you to scale your application more efficiently. You can add new routes or functionalities without cluttering a single file with too much code.

Authentication:

User Authentication: User provides credentials (username and password) to the server.

Session Creation: If the credentials are correct, the server creates a session and stores user information in it.

Session ID Issuance: The server sends a session ID to the user.

Session ID Storage: The user stores the session ID (typically in cookies).

Access Protected Routes: The user includes the session ID in subsequent requests to access protected routes.

User Interface:

Simplicity and Clarity

- **Minimalist Design:** Use a clean and simple design to avoid overwhelming users with too much information. Focus on essential elements and remove any unnecessary clutter.
- **Clear Navigation:** Ensure that navigation is intuitive and straightforward. Use clear labels and icons to guide users through the app.

User-Centric Design

- **Personalization:** Tailor the user experience based on individual preferences and behaviors. Provide personalized recommendations and insights.
- **Accessibility:** Design the UI to be accessible to all users, including those with disabilities. Use appropriate color contrasts, font sizes, and screen reader compatibility.

Security and Trust

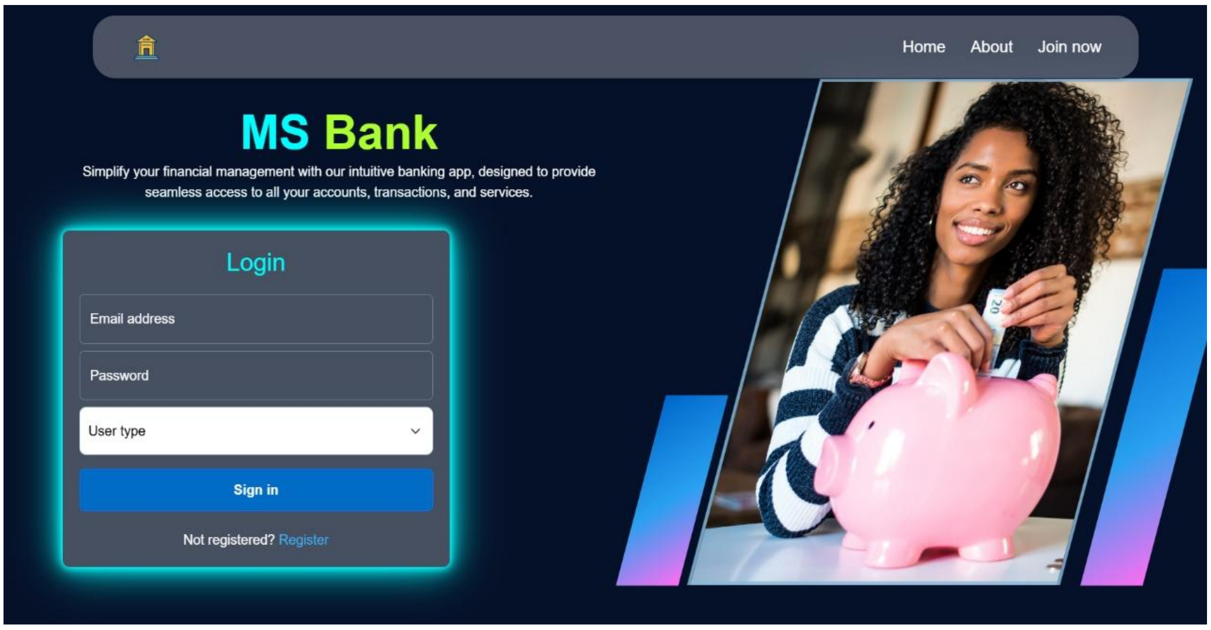
- **Secure Authentication:** Implement strong authentication methods such as two-factor authentication (2FA) or biometric authentication (fingerprint, facial recognition) to ensure user security.
- **Transparency:** Clearly communicate security measures and privacy policies to build trust with users.

Consistency

- **Uniform Design Language:** Maintain consistency in design elements such as colors, fonts, and button styles throughout the app. This helps users feel familiar and comfortable with the interface.
- **Consistent User Experience:** Ensure that the user experience is consistent across different devices and platforms (e.g., mobile, web).

Demo:

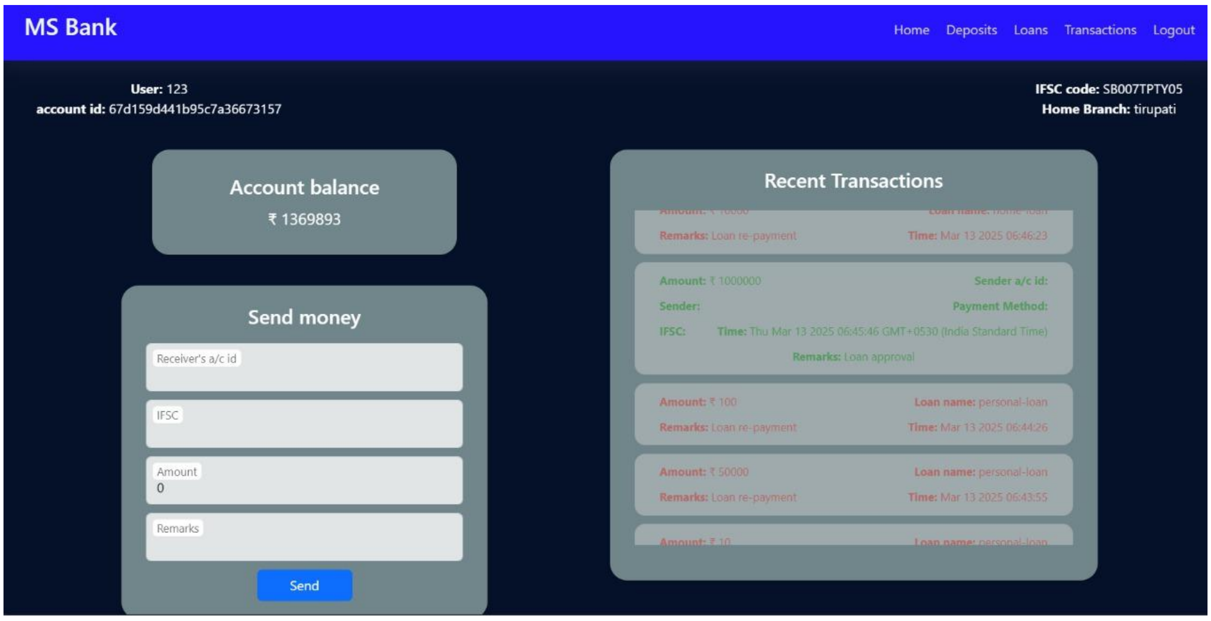
Home



MS Bank

Simplify your financial management with our intuitive banking app, designed to provide seamless access to all your accounts, transactions, and services.

User Home



User: 123
account id: 67d159d441b95c7a36673157

IFSC code: SB007TPTY05
Home Branch: tirupati

Account balance
₹ 1369893

Recent Transactions

Remarks: Loan re-payment **Time:** Mar 13 2025 06:46:23

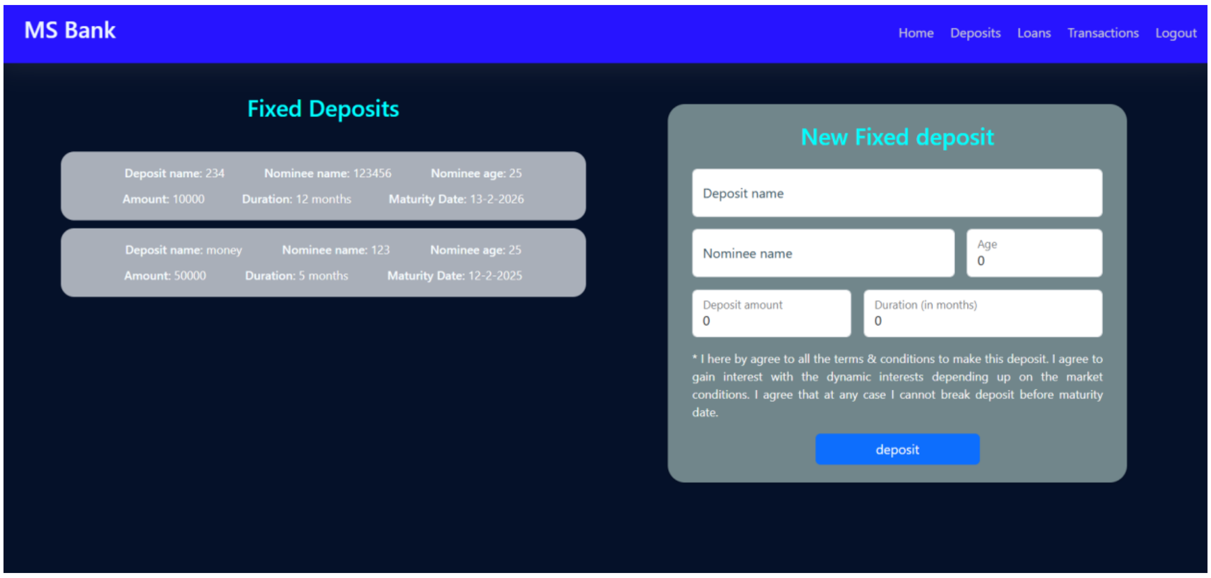
Amount: ₹ 1000000 **Sender a/c id:**
Sender: **Payment Method:**
IFSC: **Time:** Thu Mar 13 2025 06:45:46 GMT+0530 (India Standard Time)
Remarks: Loan approval

Amount: ₹ 100 **Loan name:** personal-loan
Remarks: Loan re-payment **Time:** Mar 13 2025 06:44:26

Amount: ₹ 50000 **Loan name:** personal-loan
Remarks: Loan re-payment **Time:** Mar 13 2025 06:43:55

Amount: ₹ 10 **Loan name:** personal-loan

User Deposits



MS Bank

Home Deposits Loans Transactions Logout

Fixed Deposits

Deposit name: 234 **Nominee name:** 123456 **Nominee age:** 25
Amount: 10000 **Duration:** 12 months **Maturity Date:** 13-2-2026

Deposit name: money **Nominee name:** 123 **Nominee age:** 25
Amount: 50000 **Duration:** 5 months **Maturity Date:** 12-2-2025

New Fixed deposit

Deposit name

Nominee name

Age
0

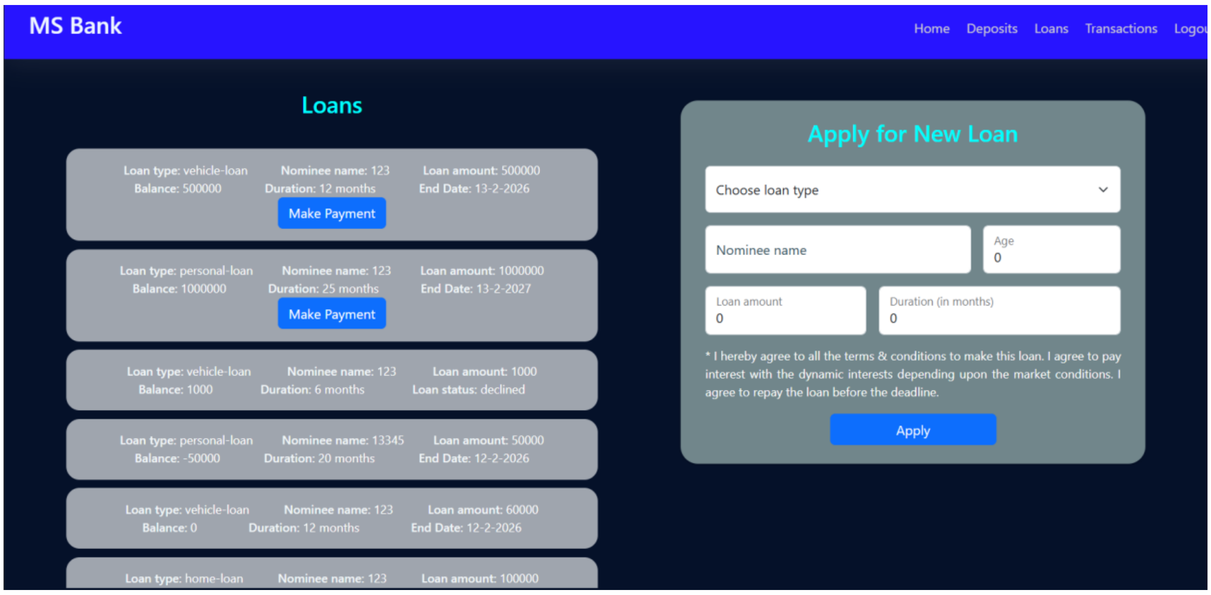
Deposit amount
0

Duration (in months)
0

* I here by agree to all the terms & conditions to make this deposit. I agree to gain interest with the dynamic interests depending up on the market conditions. I agree that at any case I cannot break deposit before maturity date.

deposit

User Loans



MS Bank

Home Deposits Loans Transactions Logo

Loans

Loan type: vehicle-loan **Nominee name:** 123 **Loan amount:** 500000
Balance: 500000 **Duration:** 12 months **End Date:** 13-2-2026
Make Payment

Loan type: personal-loan **Nominee name:** 123 **Loan amount:** 1000000
Balance: 1000000 **Duration:** 25 months **End Date:** 13-2-2027
Make Payment

Loan type: vehicle-loan **Nominee name:** 123 **Loan amount:** 1000
Balance: 1000 **Duration:** 6 months **Loan status:** declined

Loan type: personal-loan **Nominee name:** 13345 **Loan amount:** 50000
Balance: -50000 **Duration:** 20 months **End Date:** 12-2-2026

Loan type: vehicle-loan **Nominee name:** 123 **Loan amount:** 60000
Balance: 0 **Duration:** 12 months **End Date:** 12-2-2026

Loan type: home-loan **Nominee name:** 123 **Loan amount:** 100000

Apply for New Loan

Choose loan type

Nominee name

Age
0

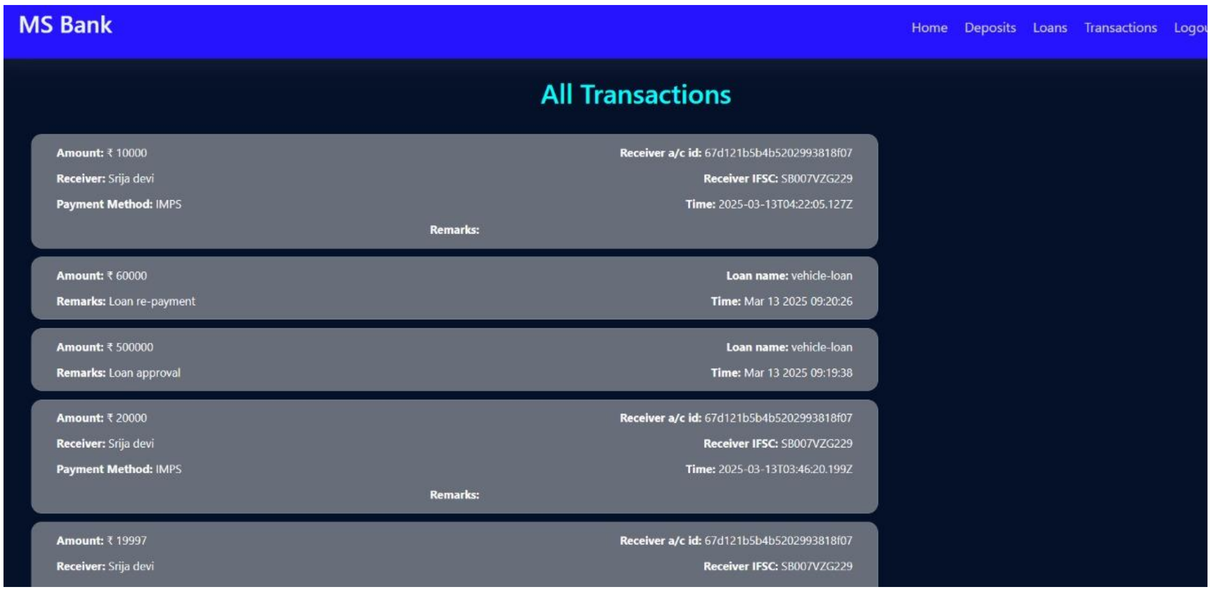
Loan amount
0

Duration (in months)
0

* I hereby agree to all the terms & conditions to make this loan. I agree to pay interest with the dynamic interests depending upon the market conditions. I agree to repay the loan before the deadline.

Apply

User Transactions



All Transactions

Amount: ₹ 10000
Receiver a/c id: 67d121b5b4b5202993818f07
Receiver: Srija devi
Receiver IFSC: SB007VZG229
Payment Method: IMPS
Time: 2025-03-13T04:22:05.127Z
Remarks:

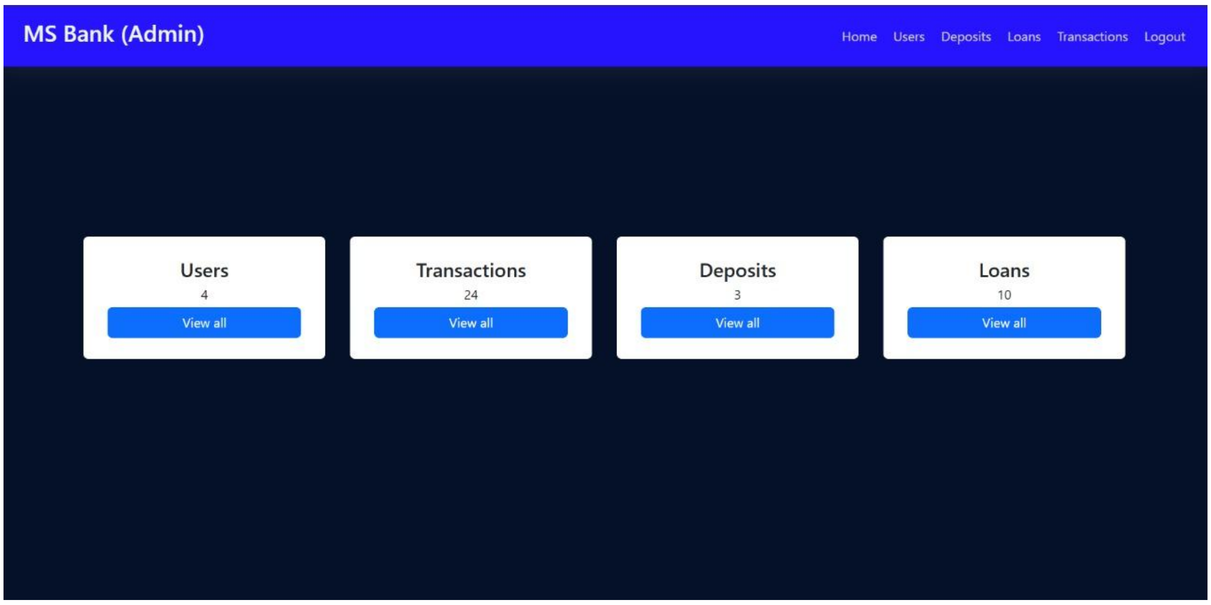
Amount: ₹ 60000
Loan name: vehicle-loan
Remarks: Loan re-payment
Time: Mar 13 2025 09:20:26

Amount: ₹ 500000
Loan name: vehicle-loan
Remarks: Loan approval
Time: Mar 13 2025 09:19:38

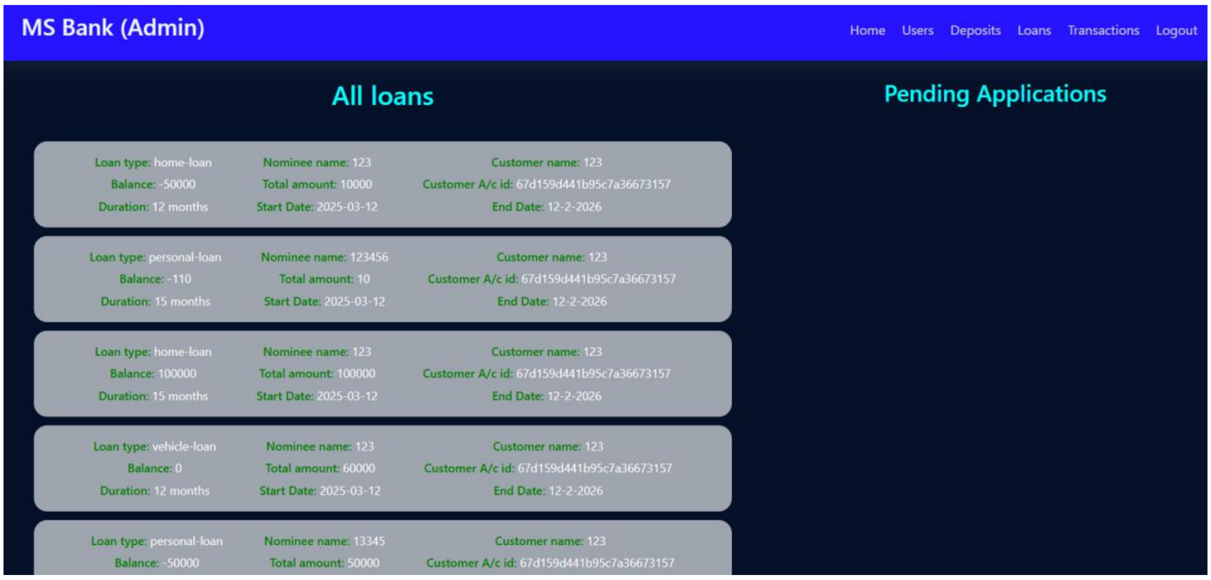
Amount: ₹ 20000
Receiver a/c id: 67d121b5b4b5202993818f07
Receiver: Srija devi
Receiver IFSC: SB007VZG229
Payment Method: IMPS
Time: 2025-03-13T03:46:20.199Z
Remarks:

Amount: ₹ 19997
Receiver a/c id: 67d121b5b4b5202993818f07
Receiver: Srija devi
Receiver IFSC: SB007VZG229

Admin Home



Admin Loans



All loans

Pending Applications

Loan type: home-loan Nominee name: 123 Customer name: 123
Balance: -50000 Total amount: 10000 Customer A/c id: 67d159d441b95c7a36673157
Duration: 12 months Start Date: 2025-03-12 End Date: 12-2-2026

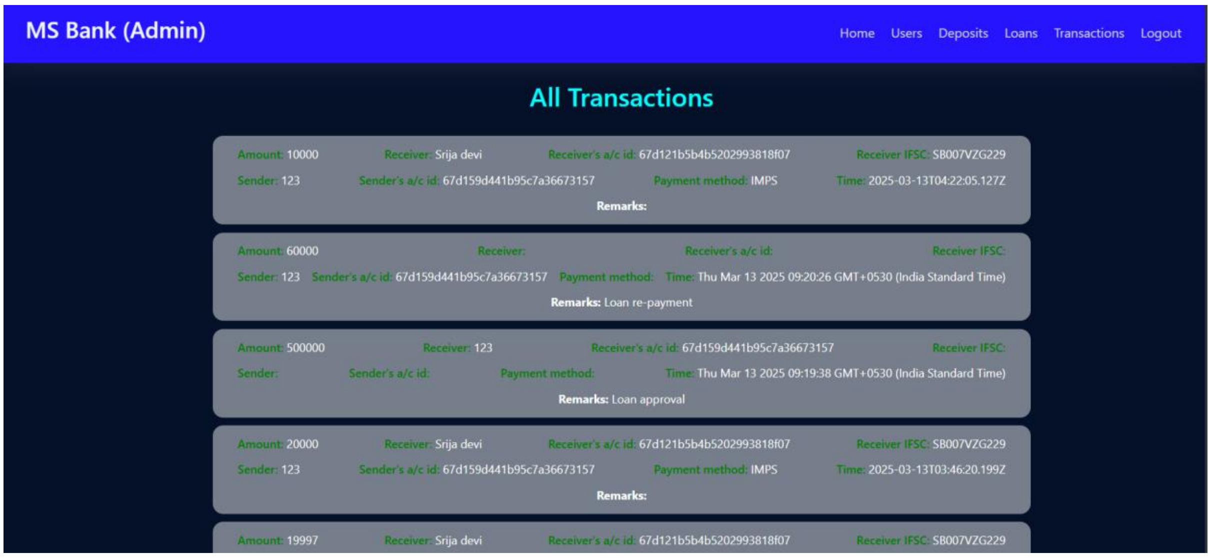
Loan type: personal-loan Nominee name: 123456 Customer name: 123
Balance: -110 Total amount: 10 Customer A/c id: 67d159d441b95c7a36673157
Duration: 15 months Start Date: 2025-03-12 End Date: 12-2-2026

Loan type: home-loan Nominee name: 123 Customer name: 123
Balance: 100000 Total amount: 100000 Customer A/c id: 67d159d441b95c7a36673157
Duration: 15 months Start Date: 2025-03-12 End Date: 12-2-2026

Loan type: vehicle-loan Nominee name: 123 Customer name: 123
Balance: 0 Total amount: 60000 Customer A/c id: 67d159d441b95c7a36673157
Duration: 12 months Start Date: 2025-03-12 End Date: 12-2-2026

Loan type: personal-loan Nominee name: 13345 Customer name: 123
Balance: -50000 Total amount: 50000 Customer A/c id: 67d159d441b95c7a36673157

Admin Transactions



All Transactions

Amount: 10000 Receiver: Srija devi Receiver's a/c id: 67d121b5b4b5202993818f07 Receiver IFSC: SB007VZG229
Sender: 123 Sender's a/c id: 67d159d441b95c7a36673157 Payment method: IMPS Time: 2025-03-13T04:22:05.127Z
Remarks:

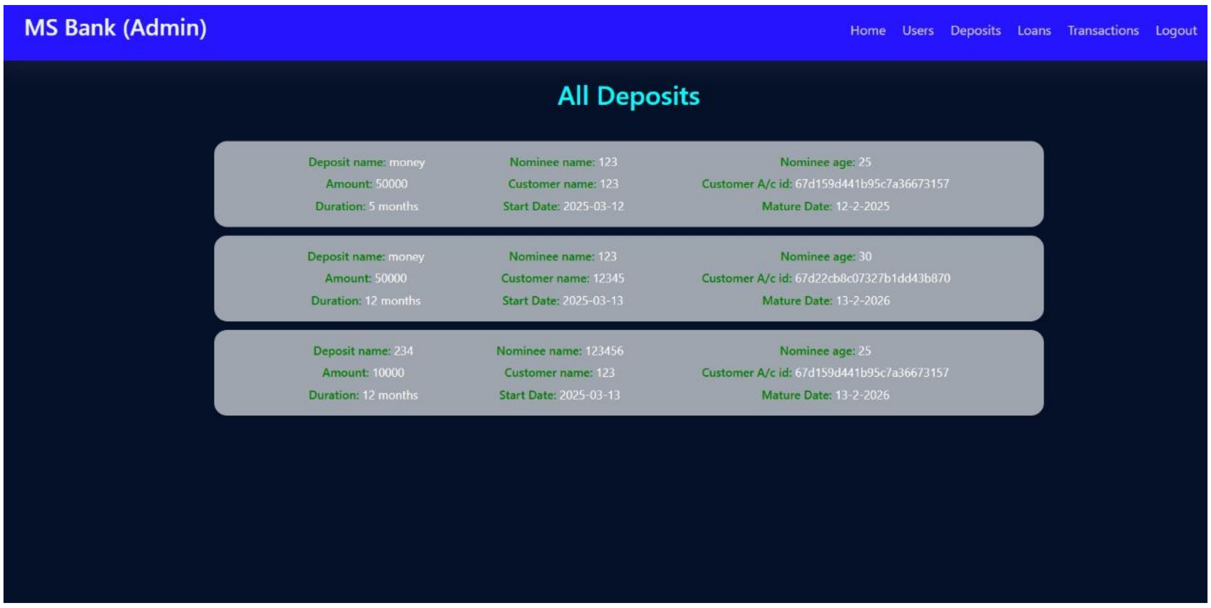
Amount: 60000 Receiver: Receiver's a/c id: Receiver IFSC:
Sender: 123 Sender's a/c id: 67d159d441b95c7a36673157 Payment method: Time: Thu Mar 13 2025 09:20:26 GMT+0530 (India Standard Time)
Remarks: Loan re-payment

Amount: 500000 Receiver: 123 Receiver's a/c id: 67d159d441b95c7a36673157 Receiver IFSC:
Sender: Sender's a/c id: Payment method: Time: Thu Mar 13 2025 09:19:38 GMT+0530 (India Standard Time)
Remarks: Loan approval

Amount: 20000 Receiver: Srija devi Receiver's a/c id: 67d121b5b4b5202993818f07 Receiver IFSC: SB007VZG229
Sender: 123 Sender's a/c id: 67d159d441b95c7a36673157 Payment method: IMPS Time: 2025-03-13T03:46:20.199Z
Remarks:

Amount: 19997 Receiver: Srija devi Receiver's a/c id: 67d121b5b4b5202993818f07 Receiver IFSC: SB007VZG229

Admin Deposits



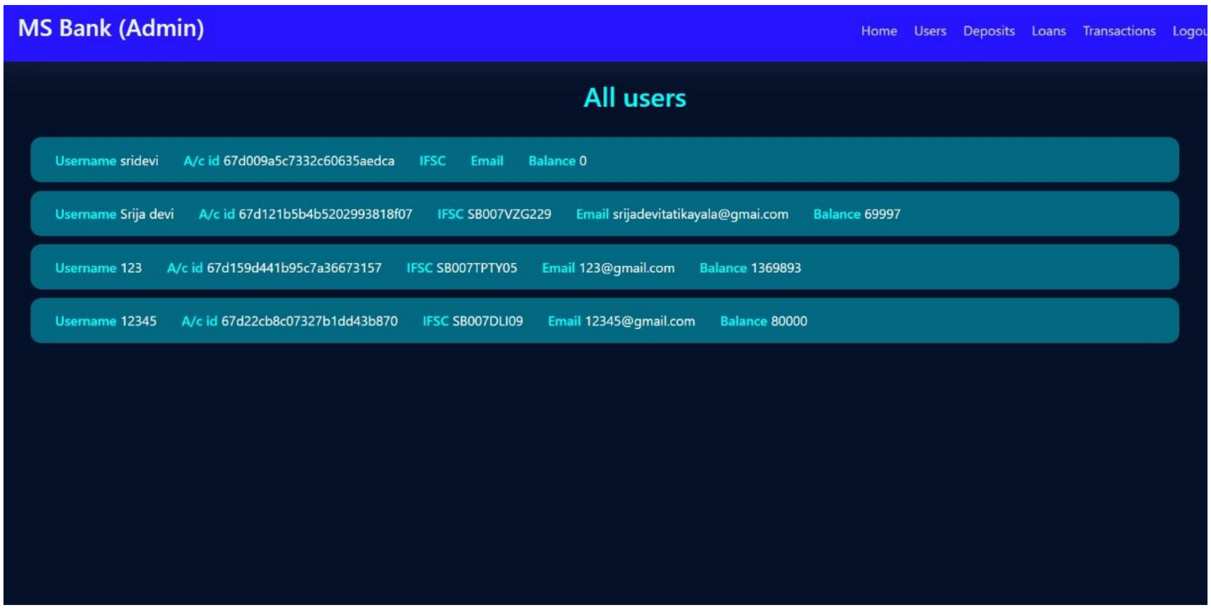
All Deposits

Deposit name: money Nominee name: 123 Nominee age: 25
Amount: 50000 Customer name: 123 Customer A/c id: 67d159d441b95c7a36673157
Duration: 5 months Start Date: 2025-03-12 Mature Date: 12-2-2025

Deposit name: money Nominee name: 123 Nominee age: 30
Amount: 50000 Customer name: 12345 Customer A/c id: 67d22cb8c07327b1dd43b870
Duration: 12 months Start Date: 2025-03-13 Mature Date: 13-2-2026

Deposit name: 234 Nominee name: 123456 Nominee age: 25
Amount: 10000 Customer name: 123 Customer A/c id: 67d159d441b95c7a36673157
Duration: 12 months Start Date: 2025-03-13 Mature Date: 13-2-2026

Admin Users



All users

Username sridevi A/c id 67d009a5c7332c60635aedca IFSC Email Balance 0

Username Srija devi A/c id 67d121b5b4b5202993818f07 IFSC SB007VZG229 Email srijadevitatikayala@gmai.com Balance 69997

Username 123 A/c id 67d159d441b95c7a36673157 IFSC SB007TPTY05 Email 123@gmail.com Balance 1369893

Username 12345 A/c id 67d22cb8c07327b1dd43b870 IFSC SB007DLI09 Email 12345@gmail.com Balance 80000

Testing:

Purpose: To validate that the application meets user requirements and expectations.

- Users test the entire application to ensure it meets their needs and provides a good user experience.
- Gathering feedback from users and stakeholders to make improvements.

Known Issues:

1. **Database Connection Issues**

Connecting to MongoDB can sometimes present challenges, especially in production setups. These issues might include:

- Incorrect or incomplete MongoDB URI.
- IP address restrictions when using services like MongoDB Atlas.
- MongoDB server not running (in the case of local installations).

2. **Project Structure Issues**

A poorly designed project architecture can lead to:

- Difficulty in maintaining and scaling the project.
- Cluttered codebase, making it hard to debug or modify.
- Reduced collaboration efficiency in team environments.

3. **Use Best Practices:**

- Keep your code modular to ensure reusability and easier testing.
- Use environment variables (e.g., `.env` files) for sensitive information like API keys or database credentials.
- Follow consistent naming conventions and file structures.

4. **Documentation:** Maintain proper documentation for your project structure so new developers can quickly understand the layout

Adopt MVC Architecture: The **Model-View-Controller (MVC)** design pattern separates the application into three interconnected components:

- **Model:** Handles data logic, such as database schema definitions and interactions.
- **View:** Manages the user interface, such as rendering web pages or returning JSON responses.
- **Controller:** Acts as an intermediary, processing user inputs, interacting with the model, and determining responses.

Future Enhancement:

Enhanced Financial Management Tools:

- **Budgeting Tools:** Provide users with tools to create and manage budgets, set financial goals, and track expenses.
- **Investment Options:** Offer features for users to invest in stocks, mutual funds, and other financial instruments directly from the app.
- **Loan and Mortgage Calculators:** Include calculators to help users estimate loan payments and mortgage rates.

Improved User Interface and Experience

- **Dark Mode:** Offer a dark mode option to reduce eye strain and improve readability in low-light conditions.
- **Responsive Design:** Ensure the application is fully responsive and provides a seamless experience across all devices, including smartphones, tablets, and desktops.