Full Stack Development with MERN

1. Introduction

- **Project Title:** Banking magement
- **Team Members:** C.Venkata Krishna
  S .Hithesh
  S .Chandra Kishore
  N .Venkata Subrahmanya Deepak

2. Project Overview

- **Purpose:**

SVit Bank aims to revolutionize the digital banking experience by providing a comprehensive, user-friendly platform for managing personal finances. The project's goal is to offer a seamless, secure, and efficient banking solution accessible via web and mobile applications. By integrating modern technologies and ensuring robust security measures, Vit Bank seeks to enhance user convenience and trust, streamline banking operations, and provide real-time financial services.
Features:

1. **User Registration and Authentication:**
   a. **Email and Password:** Secure user registration and login using email and password.
   b. **Third-Party Login:** Integration with Google and Facebook for easy and secure login.
2. **Account Management:**
   a. **Dashboard:** Comprehensive account dashboard displaying balances, transaction history, and account statements.
   b. **Profile Management:** Update personal information and manage account settings.
3. **Money Transfers:**
   a. **Peer-to-Peer Transfers:** Transfer funds to other users using account IDs.
   b. **Scheduled Transfers:** Schedule future transfers and recurring payments.
4. **Loan Applications:**
   a. **Online Application:** Submit and track loan applications online.
   b. **Real-Time Updates:** Receive real-time notifications on loan application status.
5. **Transaction Management:**
   a. **Transaction History:** Detailed view of all transactions, searchable by date and

type.
    b. **Categorization:** Automatically categorize transactions for better financial insights.
6. **Real-Time Notifications:**
    a. **Alerts:** Instant notifications for transactions, loan updates, and account activities.
    b. **Customizable Settings:** Users can customize notification preferences.
7. **Security Features:**
    a. **Data Encryption:** All sensitive data is encrypted both in transit and at rest.
    b. **Multi-Factor Authentication (MFA):** Enhanced security through MFA.
8. **Admin Dashboard:**
    a. **User Management:** Admins can view and manage user accounts and permissions.
    b. **Transaction Monitoring:** Monitor and review all transactions for fraud prevention.
    c. **Loan Processing:** Efficiently process and approve loan applications.
9. **Scalability and Performance:**
    a. **Microservices Architecture:** Ensures the system can scale to handle growing user base and transaction volume.
    b. **Caching and CDN:** Improved performance through caching strategies and Content Delivery Network (CDN) integration.
10. **Future Enhancements:**
    a. **Machine Learning:** Integration of ML models for credit scoring and fraud detection.
    b. **Voice Commands:** Support for voice-activated commands using speech-to-text services.

3. Architecture

Frontend:

- **Framework:** React.js
- **Structure:** Component-based for reusability and maintainability.
- **State Management:** React Context API or Redux.
- **Routing:** React Router for SPA experience.
- **UI Libraries:** Material-UI or Bootstrap.
- **Form Handling:** Formik and Yup.
- **API Integration:** Axios or Fetch API.

Backend:

- **Framework:** Node.js and Express.js

- **Structure:** Modular for specific functionalities (user management, transactions, loans).
- **API Endpoints:** RESTful APIs.
- **Middleware:** Authentication (JWT), logging, error handling.
- **Real-Time Communication:** WebSocket or Socket.io.
- **Service Layer:** Abstracts business logic.

Database:

- **Database:** MongoDB with Mongoose
- **Schema Design:**
  - **Users:** Personal details, credentials, settings.
  - **Transactions:** Deposits, withdrawals, transfers.
  - **Loans:** Applications, statuses.
  - **Notifications:** Real-time alerts.
- **Interactions:** CRUD operations, indexing for performance, references for data consistency, ACID transactions.

4. Setup Instructions

Prerequisites:

- Node.js (version 14.x or later)
- MongoDB (version 4.x or later)
- Git
- npm or yarn (package manager)

Installation:

1. Clone the Repository:

```
git clone https://github.com/yourusername/vit-bank.git
cd vit-bank
```

2. Install Dependencies: For the frontend:

```
cd frontend
npm install
# or if using yarn
yarn install
```

For the backend:

```
cd backend
npm install
# or if using yarn
yarn install
```

3. **Set Up Environment Variables:**

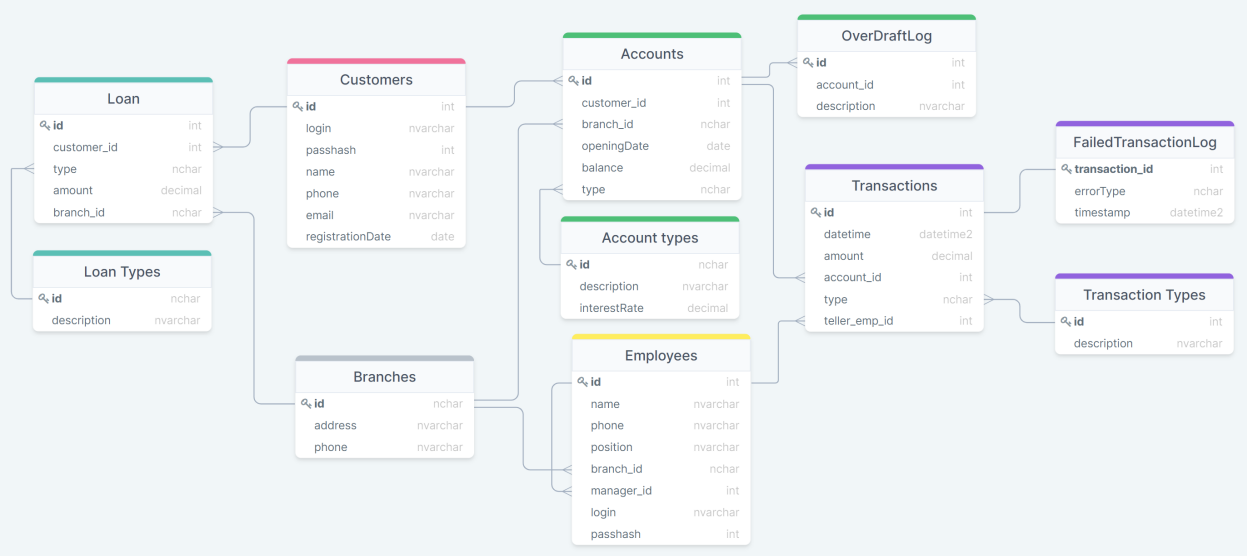
Create a `.env` file in the `backend` directory with the following contents:

```
PORT=5000
MONGODB_URI=mongodb://localhost:27017/vitbank
JWT_SECRET=your_jwt_secret
```

4. **Run MongoDB:** Make sure MongoDB is running on your system. You can start it with:
5. Start the Backend Server:
6. Start the Frontend Development Server:
7. **Access the Application:** Open your browser and navigate to `http://localhost:3000` to access the frontend. The backend API will be running on `http://localhost:5000`.

5. Folder Structure

- **Clientside and Serverside:**



6. Running the Application:

commands to start the frontend and backend servers locally.

- **Frontend:** `npm start` in the client directory.

1. Navigate to the `frontend` directory.
   Command: cd frontend
   npm start.

Backend: `npm start` in the server directory.

2. Navigate to the `Backend` directory.
   Command: cd Backend
   npm start.

7. API Documentation

Base URL:`http://localhost:5000/api`

1. User Registration and Authentication

1.1 Register User

- **Endpoint:**`/api/auth/register`
- **Method:**`POST`
- **Parameters:**
  - `name` (string) - Required
  - `email` (string) - Required
  - `password` (string) - Required

Example Request:

```
{
  "name": "John Doe",
  "email": "johndoe@example.com",
  "password": "password123"
}
```

Example Response:

2. Login User

- **Endpoint:**`/api/auth/login`
- **Method:**`POST`
- **Parameters:**
  - `email` (string) - Required
  - `password` (string) - Required

Example Request:

Example Response:

```
{
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
  "user": {
    "id": "60d21b4667d0d8992e610c85",
    "name": "John Doe",
    "email": "johndoe@example.com"
  }
}
```


3. Account Management

Get User Profile

- **Endpoint:** `/api/users/me`
- **Method:** `GET`
- **Headers:**
  - `Authorization: Bearer <token>`

Example Response:


```
{
  "id": "60d21b4667d0d8992e610c85",
  "name": "John Doe",
  "email": "johndoe@example.com",
  "createdAt": "2023-07-12T00:00:00.000Z"
}
```


4. Update User Profile

- **Endpoint:** `/api/users/me`
- **Method:** `PUT`
- **Headers:**
  - `Authorization: Bearer <token>`
- **Parameters:**
  - `name` (string) - Optional
  - `email` (string) - Optional

- `password` (string) - Optional

5.. Loans

Apply for a Loan

- **Endpoint:** `/api/loans`
- **Method:** `POST`
- **Headers:**
  - `Authorization: Bearer <token>`
- **Parameters:**
  - `amount` (number) - Required
  - `term` (number) - Required (in months)
  - `reason` (string) - Required

Example Request:

```
{
  "amount": 10000,
  "term": 24,
  "reason": "Home renovation"
}
```

Example Response:

```
{
  "message": "Loan application submitted successfully",
  "loan": {
    "id": "60d21b4667d0d8992e610c88",
    "amount": 10000,
    "term": 24,
    "reason": "Home renovation",
    "status": "pending",
    "date": "2023-07-14T00:00:00.000Z"
  }
}
```

Get Loan Status

- **Endpoint:** `/api/loans/:id`
- **Method:** `GET`
- **Headers:**
  - `Authorization: Bearer <token>`
- **Parameters:**
  - `id` (string) - Loan ID

Example Response:

8. Authentication :

Authentication and Authorization

In Vit Bank, authentication and authorization are crucial to ensure the security of user data and access control. Here's an overview of how these are handled:

1. JWT (JSON Web Tokens)

- **Purpose:** JWT is used to authenticate and authorize users securely.
- **Process:**
  a. **User Registration:** When a user registers, their credentials (name, email, password) are saved in the database after hashing the password for security.
  b. **User Login:** Upon successful login, the server generates a JWT, which is signed with a secret key. This token is then sent to the client.
  c. **Token Storage:** The client stores the JWT, typically in local storage or cookies.

d. **Authenticated Requests:** For subsequent requests to protected endpoints, the client includes the JWT in the `Authorization` header as `Bearer <token>`.

2. Middleware

- **Purpose:** Middleware functions in the backend verify the JWT for protected routes.
- **Process:**
    a. **Token Verification:** The middleware extracts the token from the `Authorization` header and verifies it using the secret key.
    b. **User Identification:** If the token is valid, the middleware extracts user information from the token payload and attaches it to the request object for use in the route handler.
    c. **Access Control:** If the token is invalid or missing, the middleware responds with an appropriate error message and status code, preventing unauthorized access.

3. Token Expiration and Refresh

- **Purpose:** To enhance security, tokens have an expiration time.
- **Process:**
    a. **Token Expiration:** JWTs are configured with an expiration time (e.g., 1 hour). After expiration, the token is no longer valid.
    b. **Refresh Tokens:** To avoid frequent logins, a refresh token mechanism can be implemented. The refresh token is long-lived and can be used to obtain a new access token without re-authentication.

4. Secure Password Storage

- **Purpose:** Ensure user passwords are stored securely.
- **Process:**
    a. **Hashing:** Passwords are hashed using a strong algorithm (e.g., bcrypt) before being stored in the database.
    b. **Salting:** Salting is applied to the passwords to add an extra layer of security against rainbow table attacks.

5. Role-Based Access Control (RBAC)

- **Purpose:** Differentiate access levels for different types of users (e.g., customers, administrators).
- **Process:**
    a. **User Roles:** Users are assigned roles during registration or through an admin interface.
    b. **Access Control:** Middleware checks the user's role and ensures they have the necessary permissions to access specific routes or perform certain actions.

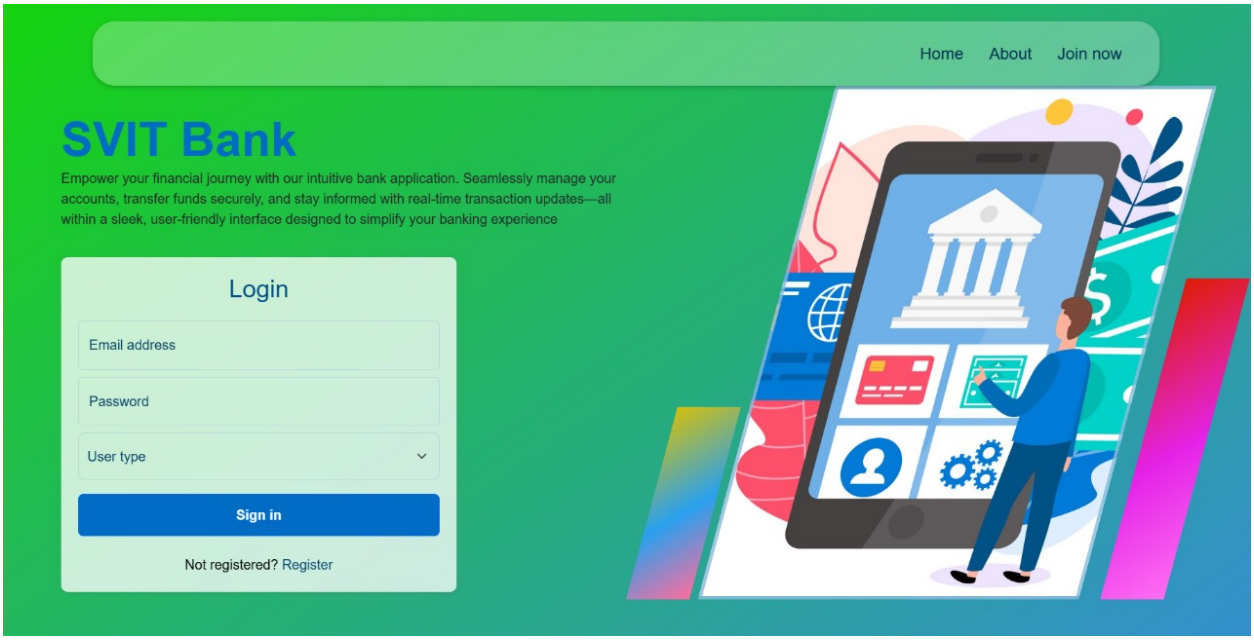
```
const jwt = require('jsonwebtoken');
const config = require('config');

function auth(req, res, next) {
  const token = req.header('Authorization').split(' ')[1];
  if (!token) return res.status(401).send('Access denied. No token provided.');

  try {
    const decoded = jwt.verify(token, config.get('jwtSecret'));
    req.user = decoded;
    next();
  } catch (ex) {
    res.status(400).send('Invalid token.');
  }
}

module.exports = auth;
```

9. User Interface



Home About Join now

Our Commitment to Security

Ensuring the security of your financial transactions is our top priority. We employ state-of-the-art encryption protocols and multi-factor authentication to safeguard your sensitive information. Our continuous monitoring and proactive security measures provide you with peace of mind, knowing that your online banking activities are protected against unauthorized access and fraud.

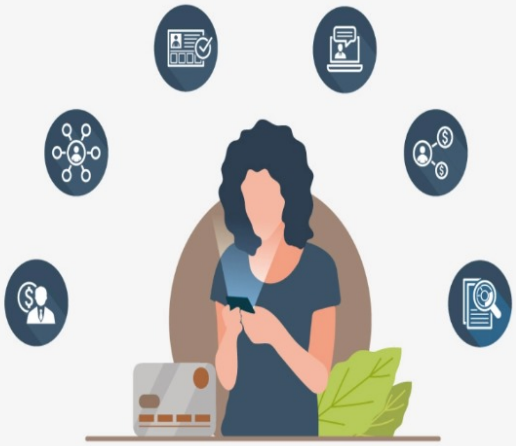
Join now!!

Home About Join now

Customer-Centric Innovation

At SVIT, we are dedicated to enhancing your banking experience through innovative digital solutions. Our online platform is designed with your convenience in mind, offering intuitive features such as mobile banking, real-time account monitoring, and easy fund transfers. We strive to continually innovate and adapt to meet your evolving financial needs, ensuring seamless access to banking services anytime, anywhere.

Join now!!



Register

Username

Email address

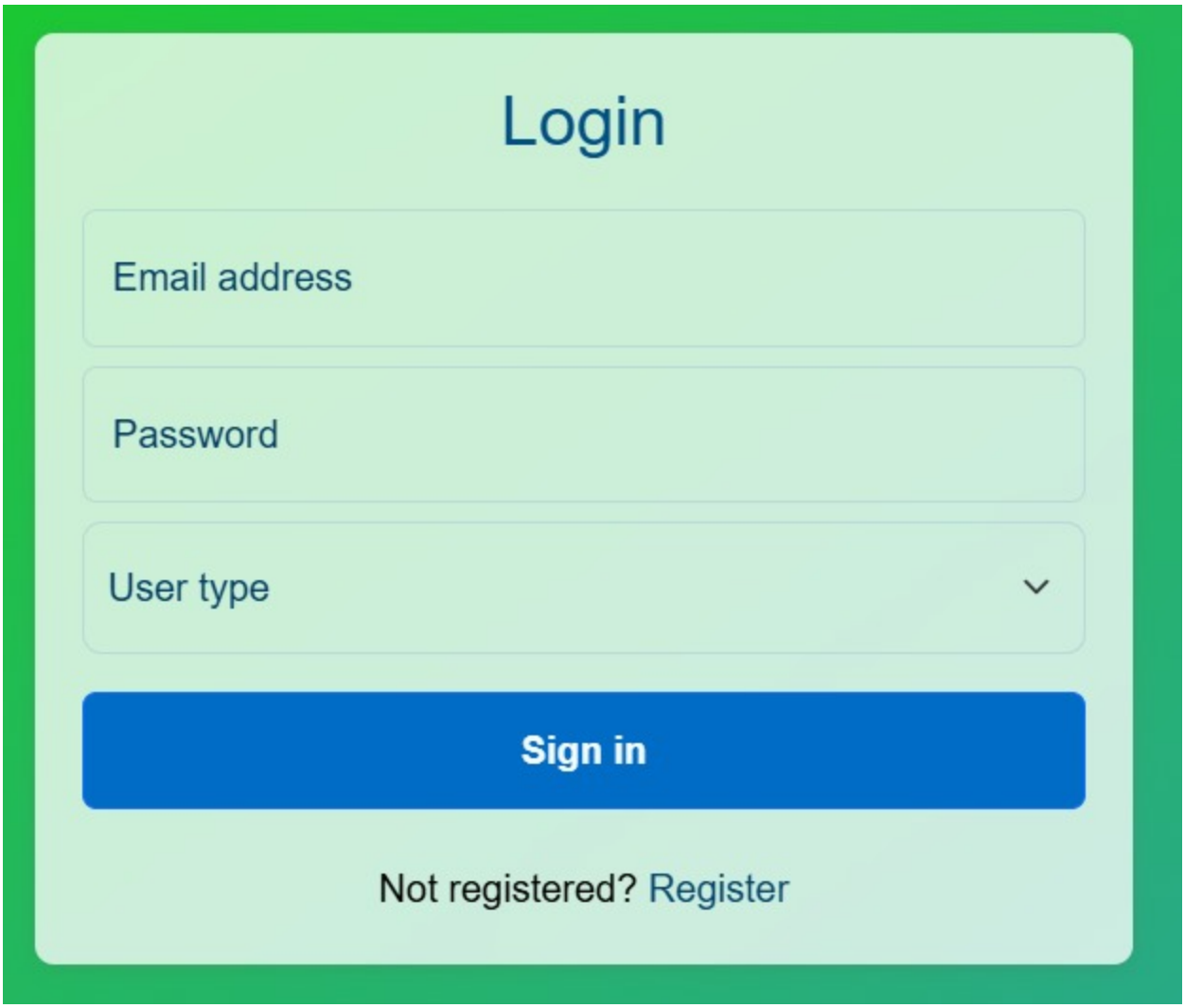
Password

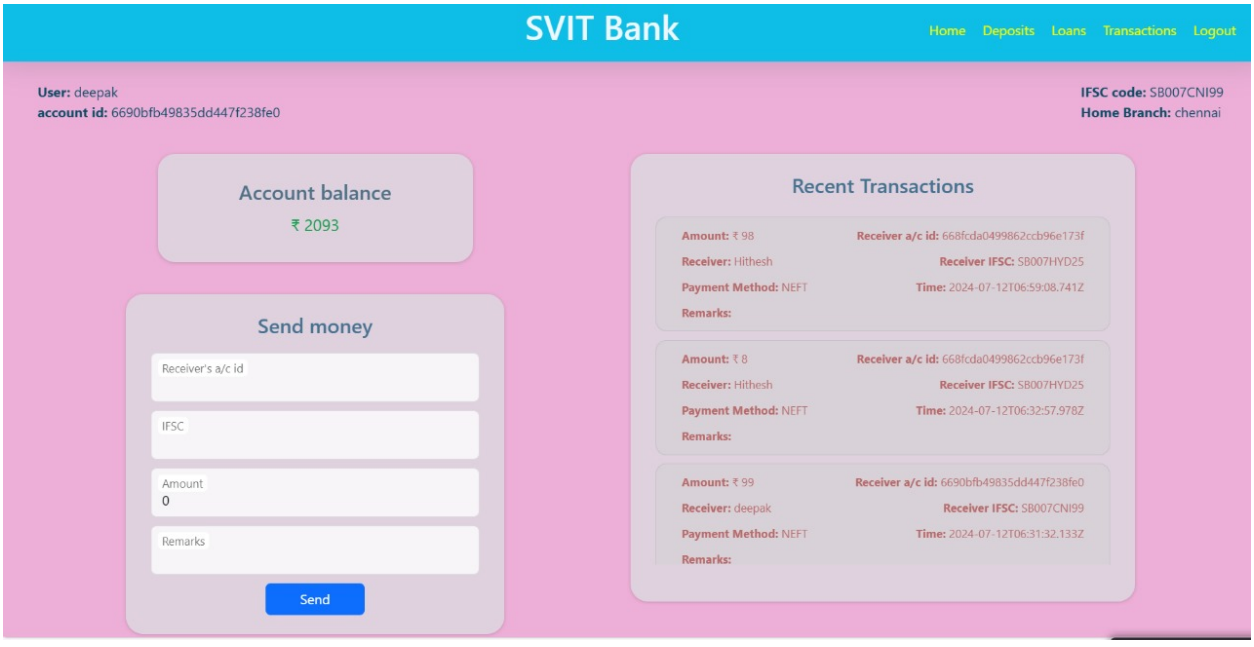
User type

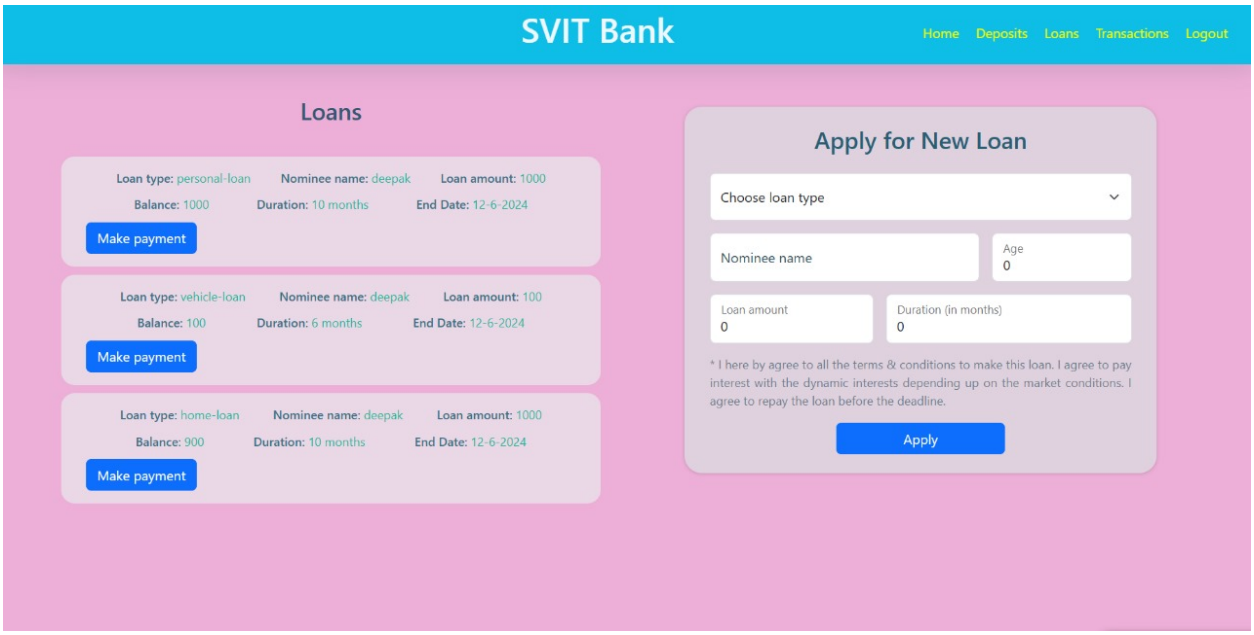
Home branch

Sign up

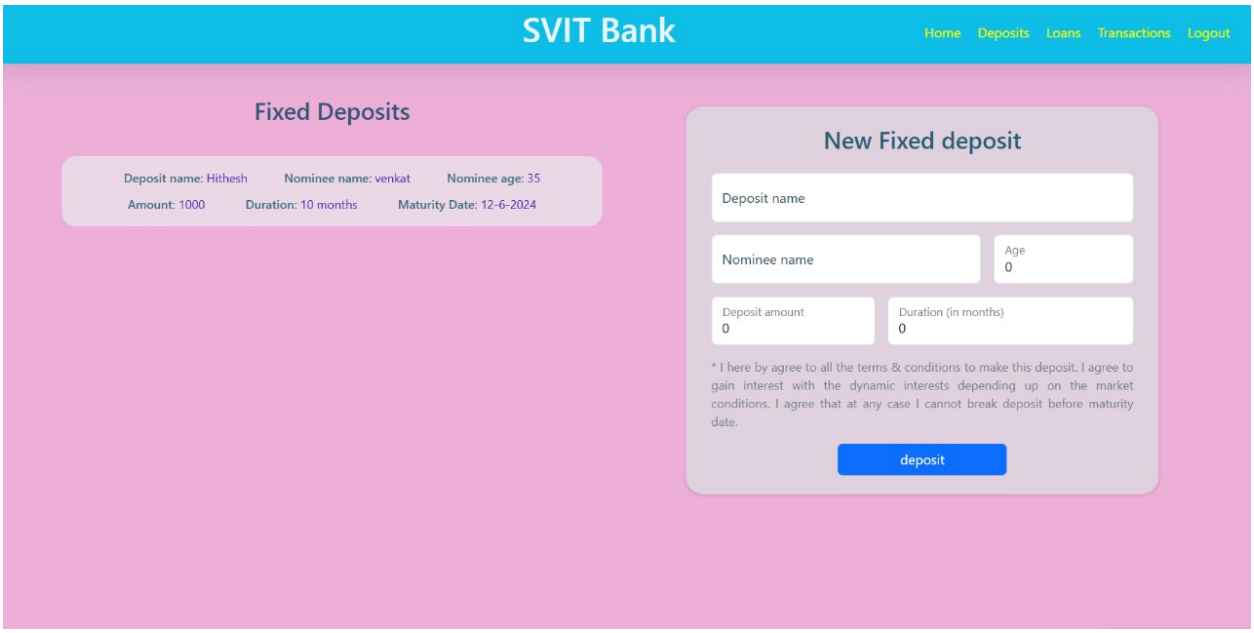
Already registered? Login

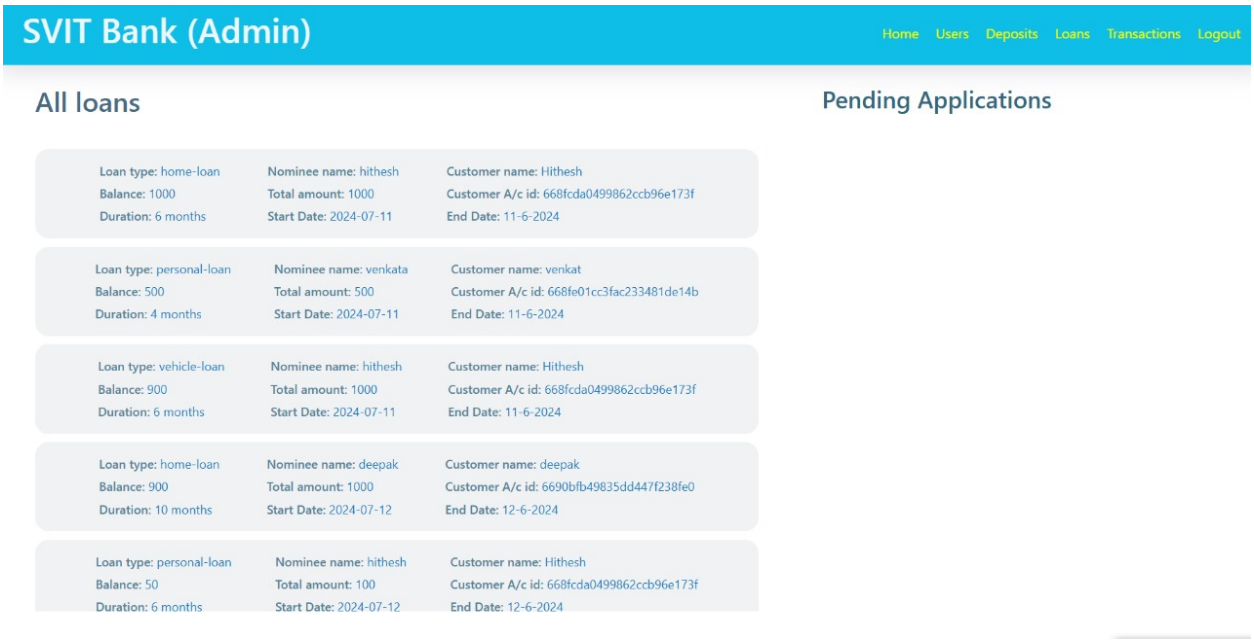

Login
Email address
Password
User type
Sign in
Not registered? Register

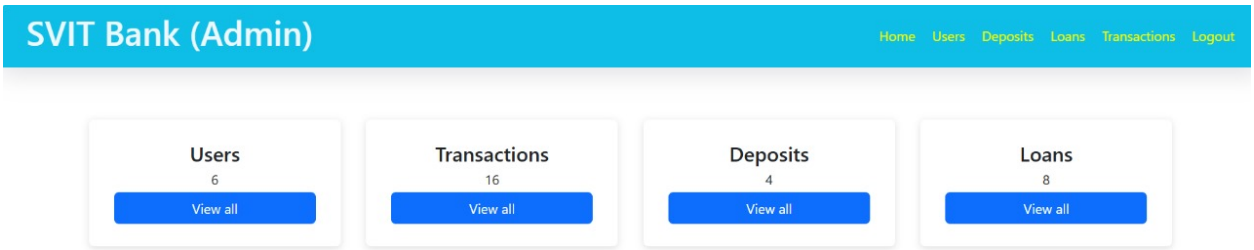

SVIT Bank
Home
Deposits
Loans
Transactions
Logout
User: deepak
account id: 6690bfb49835dd447f238fe0
IFSC code: SB007CNI99
Home Branch: chennai
Account balance
₹ 2093
Send money
Receiver's a/c id
IFSC
Amount
0
Remarks
Send
Recent Transactions
Amount: ₹ 98
Receiver a/c id: 668fcda0499862ccb96e173f
Receiver: Hithesh
Receiver IFSC: SB007HYD25
Payment Method: NEFT
Time: 2024-07-12T06:59:08.741Z
Remarks:
Amount: ₹ 8
Receiver a/c id: 668fcda0499862ccb96e173f
Receiver: Hithesh
Receiver IFSC: SB007HYD25
Payment Method: NEFT
Time: 2024-07-12T06:32:57.978Z
Remarks:
Amount: ₹ 99
Receiver a/c id: 6690bfb49835dd447f238fe0
Receiver: deepak
Receiver IFSC: SB007CNI99
Payment Method: NEFT
Time: 2024-07-12T06:31:32.133Z
Remarks:


SVIT Bank
Home Deposits Loans Transactions Logout
Loans
Loan type: personal-loan Nominee name: deepak Loan amount: 1000
Balance: 1000 Duration: 10 months End Date: 12-6-2024
Make payment
Loan type: vehicle-loan Nominee name: deepak Loan amount: 100
Balance: 100 Duration: 6 months End Date: 12-6-2024
Make payment
Loan type: home-loan Nominee name: deepak Loan amount: 1000
Balance: 900 Duration: 10 months End Date: 12-6-2024
Make payment
Apply for New Loan
Choose loan type
Nominee name
Age
0
Loan amount
0
Duration (in months)
0
* I here by agree to all the terms & conditions to make this loan. I agree to pay interest with the dynamic interests depending up on the market conditions. I agree to repay the loan before the deadline.
Apply


SVIT Bank
Home Deposits Loans Transactions Logout
All Transactions
Amount: ₹ 98
Receiver: Hithesh
Payment Method: NEFT
Remarks:
Receiver a/c id: 668fcda0499862ccb96e173f
Receiver IFSC: SB007HYD25
Time: 2024-07-12T06:59:08.741Z
Amount: ₹ 8
Receiver: Hithesh
Payment Method: NEFT
Remarks:
Receiver a/c id: 668fcda0499862ccb96e173f
Receiver IFSC: SB007HYD25
Time: 2024-07-12T06:32:57.978Z
Amount: ₹ 99
Receiver: deepak
Payment Method: NEFT
Remarks:
Receiver a/c id: 6690bfb49835dd447f238fe0
Receiver IFSC: SB007CNI99
Time: 2024-07-12T06:31:32.133Z
Amount: ₹ 100
Remarks: Loan re-payment
Loan name: home-loan
Time: Jul 12 2024 11:05:35
Amount: ₹ 100
Sender a/c id: 668fcda0499862ccb96e173f


SVIT Bank
Home Deposits Loans Transactions Logout
Fixed Deposits
Deposit name: Hithesh Nominee name: venkat Nominee age: 35
Amount: 1000 Duration: 10 months Maturity Date: 12-6-2024
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


SVIT Bank (Admin)
Home Users Deposits Loans Transactions Logout
All loans
Pending Applications
Loan type: home-loan Nominee name: hithesh Customer name: Hithesh
Balance: 1000 Total amount: 1000 Customer A/c id: 668fcda0499862ccb96e173f
Duration: 6 months Start Date: 2024-07-11 End Date: 11-6-2024
Loan type: personal-loan Nominee name: venkata Customer name: venkat
Balance: 500 Total amount: 500 Customer A/c id: 668fe01cc3fac233481de14b
Duration: 4 months Start Date: 2024-07-11 End Date: 11-6-2024
Loan type: vehicle-loan Nominee name: hithesh Customer name: Hithesh
Balance: 900 Total amount: 1000 Customer A/c id: 668fcda0499862ccb96e173f
Duration: 6 months Start Date: 2024-07-11 End Date: 11-6-2024
Loan type: home-loan Nominee name: deepak Customer name: deepak
Balance: 900 Total amount: 1000 Customer A/c id: 6690bfb49835dd447f238fe0
Duration: 10 months Start Date: 2024-07-12 End Date: 12-6-2024
Loan type: personal-loan Nominee name: hithesh Customer name: Hithesh
Balance: 50 Total amount: 100 Customer A/c id: 668fcda0499862ccb96e173f
Duration: 6 months Start Date: 2024-07-12 End Date: 12-6-2024



10. Testing

Testing Strategy

In SVit Bank, ensuring the reliability and functionality of the application is paramount. The testing strategy encompasses both manual and automated testing approaches to validate different aspects of the system.

1. Unit Testing

- **Purpose:** Verify individual units or components of the backend and frontend.
- **Tools:**
  - **Backend:** Jest, Mocha, Chai for Node.js
  - **Frontend:** Jest, React Testing Library for React components
- **Process:** Developers write unit tests for functions, modules, and components to ensure they work as expected. Mocking and stubbing techniques are used to isolate units for testing.

2. Integration Testing

- **Purpose:** Test interactions between various modules and components.
- **Tools:**
  - **Backend:** Supertest, Postman for API testing
  - **Frontend:** Cypress, Selenium for UI testing
- **Process:** Integration tests validate how different parts of the application work together. API endpoints are tested for correct responses, data flow, and error handling.

3. End-to-End (E2E) Testing

- **Purpose:** Test the entire application flow from start to finish.
- **Tools:**
  - **Backend & Frontend:** Cypress, Selenium
- **Process:** E2E tests simulate real user scenarios, ensuring all functionalities work seamlessly together. They cover user journeys like registration, login, transactions, and error scenarios.

4. Security Testing

- **Purpose:** Identify and fix security vulnerabilities.
- **Tools:** OWASP ZAP, Burp Suite, Security Headers checkers
- **Process:** Security testing involves penetration testing, vulnerability scanning, and code reviews to mitigate risks like SQL injection, XSS attacks, and data breaches.

5. Performance Testing

- **Purpose:** Evaluate application performance under various conditions.
- **Tools:** Apache JMeter, LoadRunner, New Relic
- **Process:** Performance tests measure response times, throughput, and resource usage to ensure the application handles expected loads without degradation.

6. User Acceptance Testing (UAT)

- **Purpose:** Validate the application meets business requirements and user expectations.
- **Process:** Conducted by stakeholders or end-users in a production-like environment to confirm usability, functionality, and overall satisfaction.

7. Continuous Integration/Continuous Deployment (CI/CD)

- **Purpose:** Automate testing and deployment processes to maintain code quality.
- **Tools:** Jenkins, CircleCI, GitLab CI/CD
- **Process:** CI/CD pipelines run automated tests (unit, integration, E2E) and deploy changes to staging or production environments based on predefined workflows.

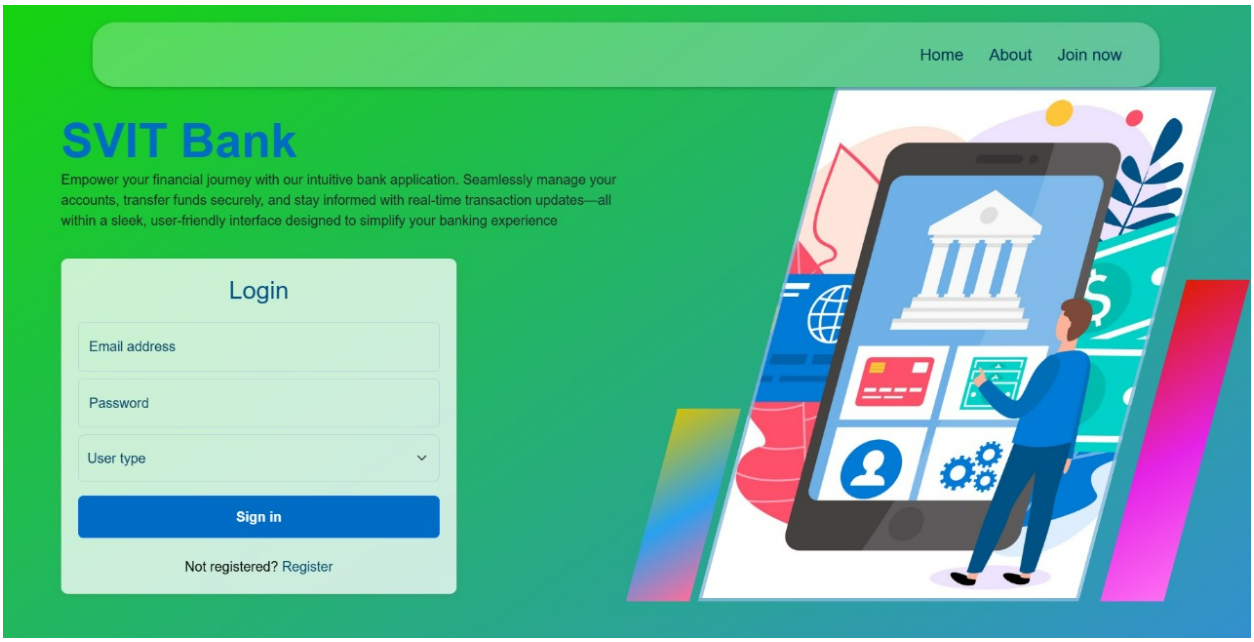
8. Monitoring and Logging

- **Purpose:** Monitor application performance and detect issues in real-time.
- **Tools:** ELK Stack (Elasticsearch, Logstash, Kibana), Prometheus, Grafana
- **Process:** Logs and metrics are collected to track errors, performance metrics, and user activities, aiding in proactive troubleshooting and optimization.

11. Screenshots or Demo

Video Link:
https://drive.google.com/file/d/1ya1RUessGisqCgBjotRSqHEUZB3HDSa/view?usp=drive_link

Screenshots:



Home About Join now

Our Commitment to Security

Ensuring the security of your financial transactions is our top priority. We employ state-of-the-art encryption protocols and multi-factor authentication to safeguard your sensitive information. Our continuous monitoring and proactive security measures provide you with peace of mind, knowing that your online banking activities are protected against unauthorized access and fraud.

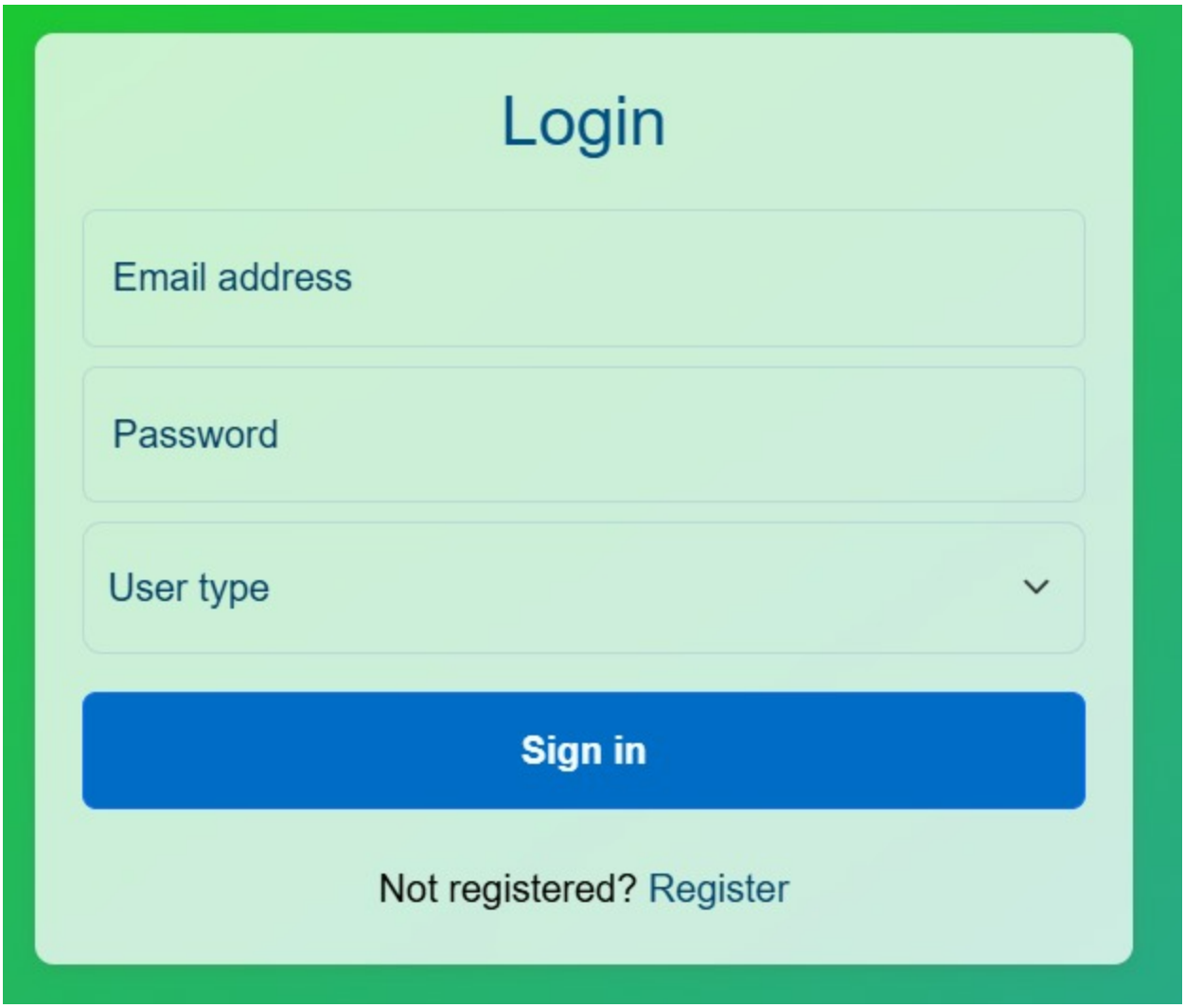
Join now!!

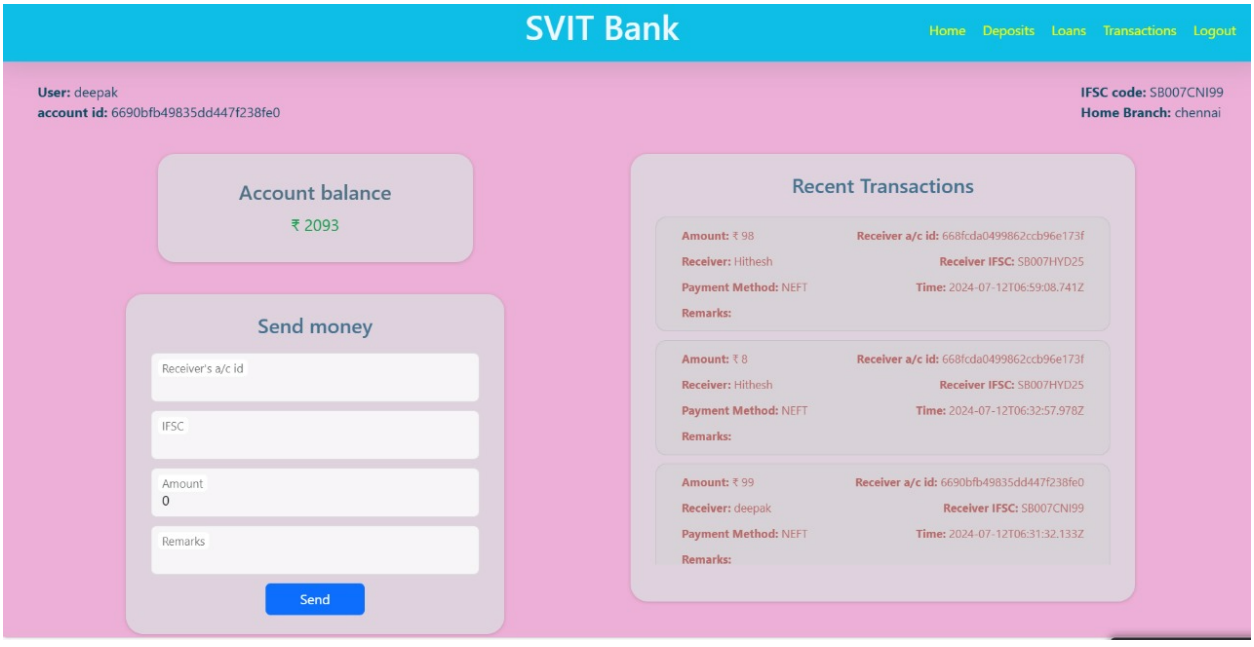
Home About Join now

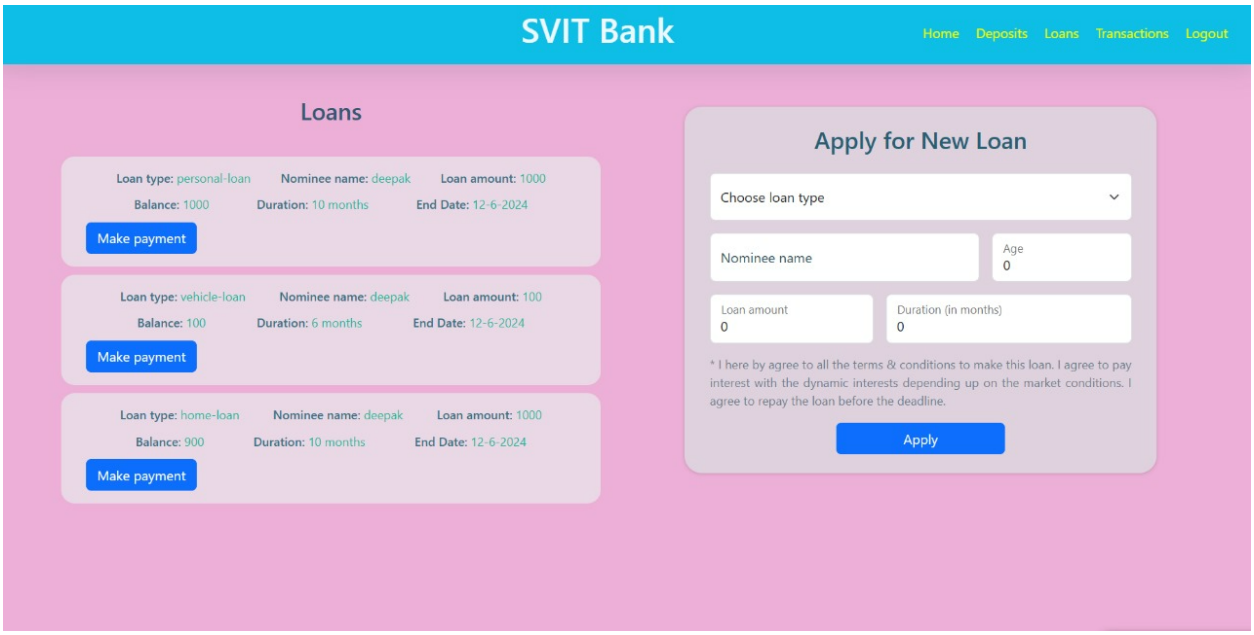
Customer-Centric Innovation

At SVIT, we are dedicated to enhancing your banking experience through innovative digital solutions. Our online platform is designed with your convenience in mind, offering intuitive features such as mobile banking, real-time account monitoring, and easy fund transfers. We strive to continually innovate and adapt to meet your evolving financial needs, ensuring seamless access to banking services anytime, anywhere.

Join now!!


Login
Email address
Password
User type
Sign in
Not registered? Register


SVIT Bank
Home
Deposits
Loans
Transactions
Logout
User: deepak
account id: 6690bfb49835dd447f238fe0
IFSC code: SB007CNI99
Home Branch: chennai
Account balance
₹ 2093
Send money
Receiver's a/c id
IFSC
Amount
0
Remarks
Send
Recent Transactions
Amount: ₹ 98
Receiver a/c id: 668fcda0499862ccb96e173f
Receiver: Hithesh
Receiver IFSC: SB007HYD25
Payment Method: NEFT
Time: 2024-07-12T06:59:08.741Z
Remarks:
Amount: ₹ 8
Receiver a/c id: 668fcda0499862ccb96e173f
Receiver: Hithesh
Receiver IFSC: SB007HYD25
Payment Method: NEFT
Time: 2024-07-12T06:32:57.978Z
Remarks:
Amount: ₹ 99
Receiver a/c id: 6690bfb49835dd447f238fe0
Receiver: deepak
Receiver IFSC: SB007CNI99
Payment Method: NEFT
Time: 2024-07-12T06:31:32.133Z
Remarks:



12. Known Issues:

- **Transaction Duplication Bug**
  - **Description:** Rare cases of duplicated transactions due to multiple submit clicks.
  - **Impact:** Incorrect balance calculations; confusion for users.
  - **Status:** Under investigation; implementing validation to prevent duplicates.
- **Login Timeout Issue**
  - **Description:** Premature logout during high server load.
  - **Impact:** Disrupted user experience with frequent logins.
  - **Status:** Investigating session management; optimizing server stability.
- **UI Rendering Delay**
  - **Description:** Slow rendering on slower networks or older devices.
  - **Impact:** User frustration due to delayed interaction.
  - **Status:** Improving frontend performance; implementing progressive loading.
- **Email Confirmation Delay**
  - **Description:** Delays in activation emails; potential spam folder issues.
  - **Impact:** Delayed access for users; activation uncertainty.
  - **Status:** Enhancing email server configurations; providing alternative activation.
- **Cross-Browser Compatibility**
  - **Description:** Features may not function on certain browsers.
  - **Impact:** Layout errors; feature dysfunction on unsupported browsers.
  - **Status:** Enhancing CSS and JavaScript compatibility; cross-browser testing.
- **Data Synchronization Issue**
  - **Description:** Backend and frontend data sync failures.

    - **Impact:** Incorrect account info; transaction discrepancies.
    - **Status:** Improving backend service reliability; data integrity checks.
- **Accessibility Challenges**
    - **Description:** Limited accessibility support; screen reader issues.
    - **Impact:** Barriers for users with disabilities.
    - **Status:** Implementing WCAG guidelines; usability testing with assistive tech.

13. Future Enhancements:

1. Enhanced Notification System

- **Feature:** Implement real-time notifications for transaction updates, account activities, and important alerts.
- **Benefit:** Improve user engagement and provide timely information to users.

2. Advanced Security Features

- **Feature:** Integrate biometric authentication (fingerprint, face recognition) for enhanced security.
- **Benefit:** Strengthen user authentication and protect against unauthorized access.

3. Personalized Financial Insights

- **Feature:** Develop AI-driven analytics to offer personalized financial advice based on user spending patterns and goals.
- **Benefit:** Empower users with actionable insights to make informed financial decisions.

4. Integration with External Financial Services

- **Feature:** Enable seamless integration with third-party financial services (e.g., investment platforms, credit scoring agencies).
- **Benefit:** Expand service offerings and provide holistic financial management solutions.

5. Enhanced Mobile Banking Features

- **Feature:** Implement mobile check deposit, bill payment, and peer-to-peer payment functionalities.
- **Benefit:** Enhance convenience and flexibility for users managing finances on mobile devices.

6. Improved User Interface and Experience (UI/UX)

- **Feature:** Revamp UI/UX design to prioritize simplicity, accessibility, and intuitive navigation.
- **Benefit:** Enhance user satisfaction and usability across all devices and platforms.

7. Automated Savings and Goal Setting

- **Feature:** Introduce tools for automated savings transfers and goal tracking.
- **Benefit:** Encourage savings habits and help users achieve financial goals effectively.

8. Blockchain Integration for Secure Transactions

- **Feature:** Explore blockchain technology for secure and transparent transaction processing.
- **Benefit:** Enhance transaction security, reduce processing costs, and improve auditability.

9. Voice-Activated Banking

- **Feature:** Develop voice-activated commands for basic banking operations via virtual assistants (e.g., Siri, Google Assistant).
- **Benefit:** Provide hands-free banking convenience and accessibility for users.

10. Community and Social Features

- **Feature:** Introduce community forums or social features for financial discussions, tips sharing, and customer support.
- **Benefit:** Foster a sense of community among users and enhance customer engagement.