RAM-BANK Project: Detailed Requirement Documentation

Project Overview

RAM-BANK (Random Access Money Bank) is an online banking system designed to offer a seamless, secure, and efficient banking experience to users. It will support essential banking services such as account creation, balance inquiry, fund transfer, and transaction history, along with advanced features like 2-Factor Authentication (2FA), WhatsApp and email OTP verification, document submission for account verification, and real-time chat for customer support. The project will be built using the MERN stack, and hosted on AWS with a custom domain, ensuring scalability and security.

1. User Roles

1. Bank Customer:
   - Can create an account, view balance and transaction history, transfer funds, etc.
2. Bank Staff:
   - Can verify documents, communicate with users, manage transactions, etc.
3. Admin:
   - Manages system functionality, verifies accounts, monitors transactions, ensures compliance, etc.
4. Blog Contributor (Optional):
   - Can create and manage blog posts.

2. Functional Requirements

- Account Creation: Users can register, submit documents, and verify via OTP.
- Login: Supports email/password, Google Sign-In, and 2FA.
- Fund Transfers: Users can transfer funds securely with OTP confirmation.

- Real-Time Chat: Chat feature for user-staff communication.

- Notifications: Email, WhatsApp, and SMS notifications.

- Blog Section: Allows blog posts related to banking.

3. Non-Functional Requirements

- Performance: Handles up to 10,000 concurrent users.

- Security: 2FA, encrypted data, HTTPS, regular audits.

- Scalability: Scalable to support additional services.

- Reliability: 99.9% uptime with AWS hosting.

4. Technical Requirements

- Frontend: Vite React.js with Tailwind CSS.

- Backend: Node.js with Express.js, MongoDB, MySQL.

- Authentication: Google Authenticator for 2FA, Google OAuth 2.0.

- Hosting: AWS with custom domain and subdomains.

5. User Interface Design

- User Pages: Home, Dashboard, Transaction, Document Submission, Chat, Blog.

- Staff/Admin Pages: Admin Dashboard, Verification, Chat Management.

6. Testing and Quality Assurance

- Unit Testing: Jest for backend API, React Testing Library for frontend.

- Integration Testing: End-to-end tests for key functionalities.

- Load Testing: Handles up to 10,000 concurrent users.

- Security Testing: Regular vulnerability assessments.

7. Documentation

- User Documentation: Guide for users on how to use the system.

- Developer Documentation: API documentation, database schema, setup instructions.

8. Regulatory Compliance

- Data Protection: Compliance with GDPR.

- Financial Regulations: Adherence to KYC standards.

- Audit and Logging: Secure storage of transaction history for audits.