Software Requirements Specification

**For**

**Bank Account Opening Application**

**Prepared by:**

Alok Kumar

Shailesh Madne

Aman Gour

Nitin Nitnav **15th Dec 2024**

**Table of Contents**

1. **Introduction**  
   1.1 Purpose  
   1.2 Document Conventions  
   1.3 Intended Audience and Reading Suggestions  
   1.4 Product Scope  
   1.5 References
2. **Overall Description**  
   2.1 Product Perspective  
   2.2 Product Functions  
   2.3 User Classes and Characteristics  
   2.4 Operating Environment  
   2.5 Design and Implementation Constraints  
   2.6 User Documentation  
   2.7 Assumptions and Dependencies
3. **External Interface Requirements**  
   3.1 User Interfaces  
   3.2 Hardware Interfaces  
   3.3 Software Interfaces  
   3.4 Communication Interfaces
4. **System Requirements**  
   4.1 Account Opening Process  
   4.1.1 Description and Priority  
   4.1.2 Stimulus/Response Sequences  
   4.1.3 Functional Requirements  
   4.2 Customer Verification  
   4.2.1 Description and Priority  
   4.2.2 Stimulus/Response Sequences  
   4.2.3 Functional Requirements  
   4.3 Document Management  
   4.3.1 Description and Priority  
   4.3.2 Stimulus/Response Sequences  
   4.3.3 Functional Requirements  
   4.4 Account Type Selection (Savings/Current)  
   4.4.1 Description and Priority  
   4.4.2 Stimulus/Response Sequences  
   4.4.3 Functional Requirements  
   4.5 Link to Branch Details  
   4.5.1 Description and Priority  
   4.5.2 Stimulus/Response Sequences  
   4.5.3 Functional Requirements  
   4.6 Notifications and Acknowledgments  
   4.6.1 Description and Priority  
   4.6.2 Stimulus/Response Sequences  
   4.6.3 Functional Requirements
5. **Nonfunctional Requirements**  
   5.1 Performance Requirements  
   5.2 Security Requirements  
   5.3 Data Privacy Requirements  
   5.4 Software Quality Attributes  
   5.5 Compliance Requirements
6. **Database Design**  
   6.1 Table Structures  
   6.1.1 Branch Table  
   6.1.2 Customer Table  
   6.1.3 Account Table  
   6.1.4 Saving Account Details  
   6.1.5 Current Account Details  
   6.2 Entity-Relationship Diagram
7. **API Design and Documentation**  
   7.1 Account Opening API  
   7.1.1 Endpoints and HTTP Methods  
   7.1.2 Input Validation and Error Handling  
   7.1.3 Sample Request/Response  
   7.2 Verification API  
   7.2.1 Endpoints and HTTP Methods  
   7.2.2 Input Validation and Error Handling  
   7.2.3 Sample Request/Response  
   7.3 Branch Linking API  
   7.3.1 Endpoints and HTTP Methods  
   7.3.2 Input Validation and Error Handling  
   7.3.3 Sample Request/Response
8. **Appendices**  
   8.1 Glossary  
   8.2 Analysis Models (ER Diagram, Class Diagram)

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason for Changes** | **Version** |
| Bank Account Opening 1.0 | 15/12/2024 | First version’s specifications are defined | 1.0 |

**1. Introduction**

**1.1 Purpose**

The Bank Account Opening Application is designed to simplify and automate the account opening process for customers and staff. It allows customers to submit applications, upload verification documents, and select account types (Savings or Current). The system ensures regulatory compliance by verifying submitted documents and linking accounts to specific branches. This reduces manual workload, enhances efficiency, and improves the customer experience by minimizing turnaround times.

**1.2 Document Conventions**

* Priority Conventions: Requirements are categorized as "High" or "Low." High-priority requirements are critical for the system’s operation, while low-priority requirements enhance functionality without being essential.
* Font Conventions: Entity names (e.g., Customer, Branch) are capitalized. Glossary-defined terms appear in bold italics.
* Diagram Conventions: Diagrams use standard UML notations for clarity.

**1.3 Intended Audience and Reading Suggestions**

* Bank Management: Refer to Sections 4 and 5 for functional and non-functional requirements, focusing on features and compliance.
* Development Team: Focus on Sections 2, 3, and 4 for technical implementation and system integration.
* Testers: Use Sections 4 and 5 for creating test cases and verifying system behavior against defined requirements.
* Regulatory Auditors: Review Sections 4.2 and 5.5 for compliance with financial regulations.

**1.4 Product Scope**

The Bank Account Opening Application streamlines account creation and document validation for Savings and Current accounts. The primary objectives include:

* Automating manual account opening processes.
* Ensuring compliance with Know Your Customer (KYC) regulations.
* Providing real-time status updates to customers.
* Enhancing operational efficiency by reducing errors and turnaround times.

Future enhancements may include advanced analytics, customer segmentation, and integration with CRM systems.

**1.5 References**

1. Banking Regulatory Guidelines for KYC Compliance

**2. Overall Description**

**2.1 Product Perspective**

This system is a standalone application integrating Python Flask (backend), MySQL (database), and ReactJS (frontend). It interfaces with third-party APIs for document verification and notification services. The application eliminates the need for paper-based account opening processes and introduces a streamlined, efficient, and secure workflow. Key integrations include:

* **Document Verification API** for validating identity and address proofs.
* **Branch Management System** for linking accounts to appropriate branches.

**2.2 Product Functions**

The system offers the following core functionalities:

* **Customer Portal**:
  + Input personal details.
  + Select account type (Savings or Current).
  + Upload required documents (e.g., ID proof, address proof).
* **Document Verification**:
  + Validate uploaded documents via external APIs.
  + Flag errors or incomplete submissions for review.
* **Account Creation**:
  + Generate unique account numbers.
  + Link accounts to the selected branch.
* **Notifications**:
  + Notify customers at key stages (application submitted, under review, approved, rejected).
  + Send reminders for missing or incorrect documents.
* **Branch Management**: Admins can view branch-level details and assign applications accordingly.

**2.3 User Classes and Characteristics**

* **Customers**: Primarily interact with the portal for submitting and tracking account applications. Expected to have minimal technical knowledge.
* **Branch Staff**: Responsible for reviewing and approving applications. Must have access to dashboards and training on the system’s operation.
* **Administrators**: Oversee branch operations, manage user roles, and resolve escalations. Require advanced permissions and access to detailed reports.

**2.4 Operating Environment**

* **Backend**: Python Flask application hosted on a cloud server with SSL certification for secure communications.
* **Database**: MySQL database deployed on a scalable environment supporting multiple concurrent connections.
* **Frontend**: ReactJS-based web application tested for compatibility with modern browsers (Chrome, Edge, Firefox).
* **API Integration**: External APIs for document verification and notifications.

**2.5 Design and Implementation Constraints**

* **Regulatory Compliance**: Must adhere to financial industry standards and local laws.
* **File Size Limitations**: Document uploads limited to 10 MB per file to ensure efficient processing.
* **High Availability**: System must ensure 99.9% uptime with minimal disruptions.
* **Security**: Implement end-to-end encryption for all data transfers.

**2.6 User Documentation**

Comprehensive documentation will include:

* **User Guide**: Step-by-step instructions for using the customer portal and staff dashboard.
* **API Documentation**: Technical details for developers integrating with the system’s APIs.
* **Troubleshooting Guide**: Common issues and their resolutions.

**2.7 Assumptions and Dependencies**

* Customers will have access to stable internet connections and modern devices (e.g., smartphones, PCs).
* Document verification relies on third-party services. Downtime or changes in these APIs may impact functionality.
* Notification services require integration with SMS/email providers, which must ensure message delivery.

**3. External Interface Requirements**

**3.1 User Interfaces**

- Customer Portal: For account applications and document uploads.

- Staff Dashboard: For reviewing and approving applications.

**3.2 Hardware Interfaces**

- Compatible with desktops, laptops, and tablets.

**3.3 Software Interfaces**

- Document verification API for identity and address proofs.

- Notification service API for email/SMS updates.

**3.4 Communications Interfaces**

- Communication between back-end and front-end over HTTPS.

**4. System Requirements**

**4.1 Account Opening Process**

**4.1.1 Description and Priority**

The account opening process is a high-priority feature that allows customers to submit personal details, select account types, and track application status in real-time.

**4.1.2 Stimulus/Response Sequences**

* **Stimulus**: Customer initiates an application.
* **Response**: System validates entries, stores data, and updates the status to "Pending Verification."
* **Stimulus**: Customer selects account type and submits the application.
* **Response**: System confirms submission and notifies the customer.

**4.1.3 Functional Requirements**

* **Validation**: Ensure all required fields (e.g., name, address) are completed.
* **Account Types**: Offer options for Savings and Current accounts.
* **Status Updates**: Display application progress in real-time.

**4.2 Document Verification**

**4.2.1 Description and Priority**

This high-priority feature ensures compliance by validating uploaded documents using external APIs.

**4.2.2 Stimulus/Response Sequences**

* **Stimulus**: Customer uploads required documents.
* **Response**: System verifies the documents.

**4.2.3 Functional Requirements**

* **File Types**: Support for PDF, JPG, and PNG formats.
* **Feedback**: Notify customers of missing or invalid documents.
* **API Integration**: Use a reliable document verification service.

**4.3 Branch and Customer Linking**

**4.3.1 Description and Priority**

Essential feature for associating customer accounts with branches.

**4.3.2 Stimulus/Response Sequences**

* **Stimulus**: Admin assigns branch to customer application.
* **Response**: System updates the database and notifies the customer.

**4.3.3 Functional Requirements**

* **Branch Selection**: Allow admins to assign branches manually.
* **Audit Logs**: Maintain a record of branch assignments for accountability.

**4.4 Notifications**

**4.4.1 Description and Priority**

Critical feature to keep customers informed about their application status.

**4.4.2 Stimulus/Response Sequences**

* **Stimulus**: Application status changes (e.g., verified, approved).
* **Response**: Notification sent via email/SMS.

**4.4.3 Functional Requirements**

* **Integration**: Use a scalable notification service for timely updates.
* **Customizable Messages**: Support templates for various status updates.

**5. Nonfunctional Requirements**

**5.1 Performance Requirements**

The system must handle up to 100 concurrent users without significant delays.

Response Time: Document uploads and API calls must complete within 3-5 seconds.

Scalability: Ensure the system can scale to accommodate future growth.

**5.2 Safety Requirements**

Ensure proper error handling to prevent data loss or corruption.

Provide secure backups of all customer and account data.

**5.3 Security Requirements**

All sensitive data must be encrypted in transit and at rest.

Implement two-factor authentication for staff and admin logins.

**5.4 Software Quality Attributes**

Reliability: Ensure 99.9% uptime with proactive monitoring.

Maintainability: Use modular code for easy updates and debugging.

User-Friendliness: Interfaces must be intuitive and easy to navigate.

**5.5 Compliance Requirements**

Adhere to global data protection regulations.

Ensure compliance with local banking and KYC standards.

**Appendices**

**Appendix A: Glossary**

* **KYC**: Know Your Customer
* **API**: Application Programming Interface
* **HTTPS**: HyperText Transfer Protocol Secure

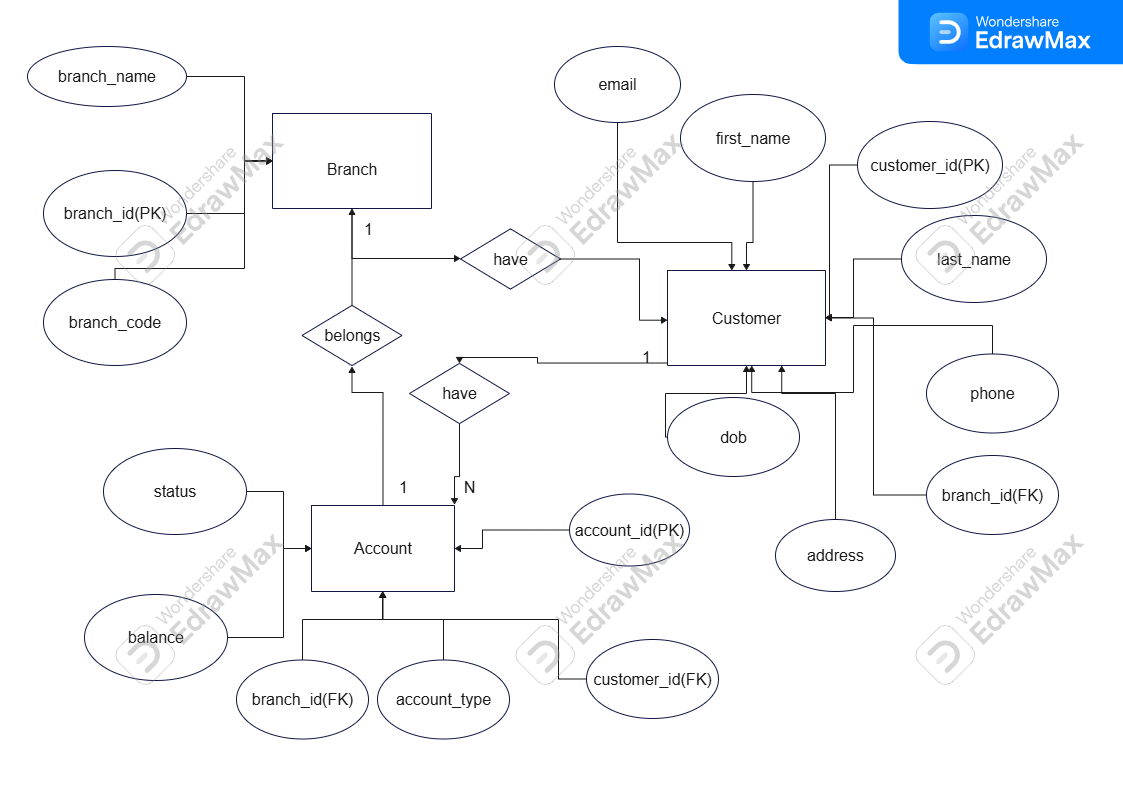
**Appendix B: Analysis Models**

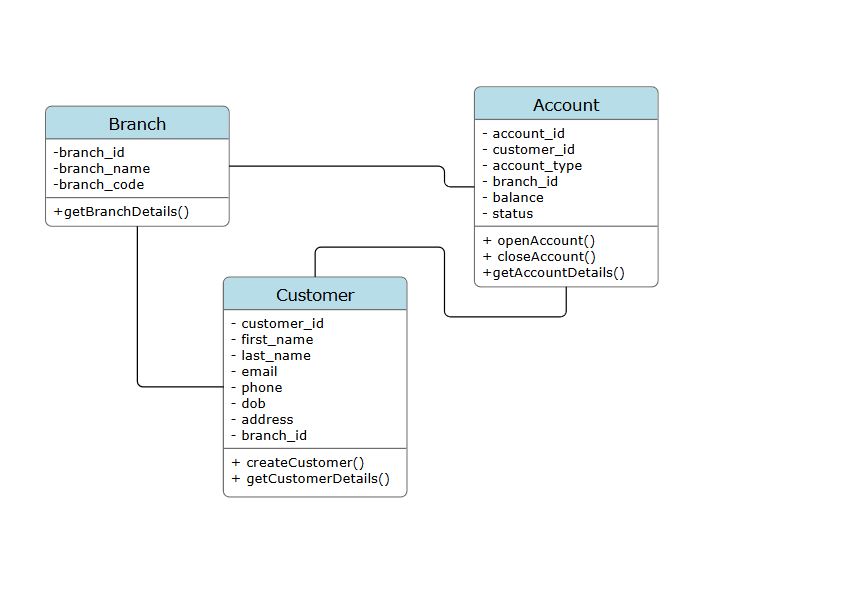
* **ER Diagram**: Visual representation of database structure.
* **Class Diagram**: Highlights interactions between users and the system.

**Appendix C: To Be Determined List**

* Evaluate integration with additional analytics tools for customer insights.
* Explore options for multilingual support in future versions.

ER – Diagram

Class Diagram



Screenshot:

