Software Requirements Specification

AI-Powered Digital Banking Platform

Version: 1.0
Date: January 2026

Prepared for:
Internal Development & Academic Evaluation

Contents

1 Introduction

1.1 Purpose

This Software Requirements Specification (SRS) document describes the functional and non-functional requirements of the **AI-Powered Digital Banking Platform**. The purpose of this document is to provide a clear and structured description of system functionality, user roles, constraints, and technical architecture for development, academic evaluation, and future maintenance.

1.2 Scope

The project focuses on building a secure, modern digital banking system that enables users to perform essential banking operations through a web-based platform. The system integrates Artificial Intelligence (AI) for fraud detection and customer support intelligence and follows industry-standard DevOps practices such as CI/CD. The system simulates a simplified core banking platform rather than a basic CRUD application.

1.3 Definitions, Acronyms, and Abbreviations

AI	Artificial Intelligence
CI/CD	Continuous Integration / Continuous Deployment
ML	Machine Learning
SRS	Software Requirements Specification
UI	User Interface

2 Overall Description

2.1 Product Perspective

The AI-Powered Digital Banking Platform is a web-based system consisting of:

1. A single web frontend application
2. A core banking backend API
3. An AI/ML service
4. A relational database

The system follows a service-oriented architecture, where AI services operate independently but integrate with the core banking backend through secure APIs.

2.2 Product Functions

The major functions of the system include:

- User authentication and role-based access control
- Bank account management
- Secure fund transfers

- Transaction history and statement generation
- AI-based fraud detection
- Chat-based customer support

2.3 User Classes and Characteristics

Customer: End users of the banking platform. Customers perform daily banking operations such as viewing balances, transferring funds, and accessing statements.

Support: Bank support staff responsible for assisting customers via chat and reviewing flagged transactions.

Admin: System administrators with full access privileges responsible for system monitoring, user management, and operational control.

2.4 Operating Environment

- **Client:** Web browser (Chrome, Edge, Firefox)
- **Server:** Hosted on a server or containerized environment
- **Deployment:** Docker-based deployment

2.5 Design and Implementation Constraints

- Secure authentication mechanisms must be used.
- Data consistency and transactional integrity (ACID compliance) must be ensured.
- CI/CD practices must be followed.
- Core banking logic must be separated from AI/ML services.

3 System Features and Requirements

3.1 User Authentication and Authorization

Description: The system shall authenticate users and authorize access based on roles (Customer, Support, Admin).

- Users shall be able to log in securely.
- The system shall enforce role-based access control.
- Unauthorized access shall be prevented.

3.2 Account Management

Description: Allows users to view and manage bank account information.

- Customers shall be able to view account details and balances.
- Customers shall be able to update profile information.
- Admins shall be able to manage and control user accounts.

3.3 Fund Transfer

Description: Enables customers to transfer funds securely between accounts.

- The system shall validate sufficient balance before processing transfers.
- All transactions shall be recorded.
- Suspicious or duplicate transfers shall be prevented or flagged.

3.4 Statement Generation

Description: Provides transaction history and statements.

- The system shall generate transaction statements.
- Statements shall include deposits, withdrawals, and transfers.
- Statements shall be available for audit and review purposes.

3.5 Fraud Detection (AI-Powered)

Description: Uses machine learning techniques to detect suspicious transaction behavior.

- The system shall analyze transaction patterns.
- Anomalies such as unusually large or rapid transfers shall be detected.
- Suspicious transactions shall be flagged for administrative review.

3.6 Chat Support

Description: Enables communication between customers and support staff.

- Customers shall be able to initiate chat support.
- Support staff shall be able to respond to customer queries.
- The system may support AI-assisted responses.

4 Non-Functional Requirements

4.1 Security Requirements

- Secure authentication mechanisms shall be used.
- Sensitive data shall be protected.
- Role-based access control shall be enforced.

4.2 Performance Requirements

- The system shall respond to user actions within acceptable time limits.
- The system shall support concurrent users.

4.3 Reliability and Availability

- The system shall ensure data integrity.
- The system shall handle failures gracefully.

4.4 Maintainability

- The system shall follow a modular architecture.
- Code shall be version-controlled using Git.

5 Technology Stack

5.1 Frontend

Next.js (React Framework)
A single web-based frontend application is used for all user roles (Customer, Support, and Admin). Role-based routing and access control are implemented at the application level.

5.2 Backend (Core Banking API)

Django REST Framework (Python)
Implements core banking logic including authentication, authorization, account management, fund transfers, statement generation, and integration with AI services through RESTful APIs.

5.3 AI / ML Layer

Python (scikit-learn, pandas)
Responsible for AI-powered fraud detection and transaction behavior analysis. The AI layer operates as a logically independent service and communicates with the core backend through secure APIs.

5.4 Database

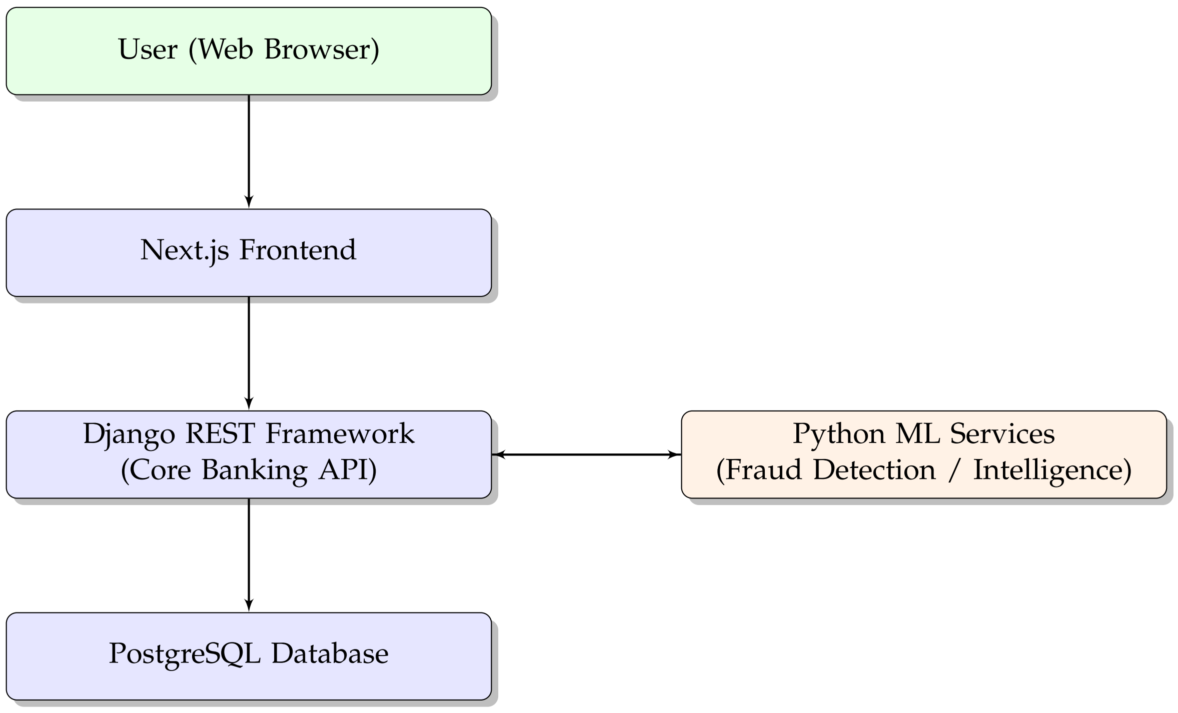
PostgreSQL
Stores users, accounts, transactions, statements, audit logs, and fraud indicators. PostgreSQL is chosen for its strong ACID compliance and suitability for financial transaction systems.

5.5 DevOps and Tools

- Git & GitHub – Version control
- Docker – Containerization
- Jenkins – CI/CD pipeline

6 System Architecture

6.1 High-Level Architecture Diagram


User (Web Browser)
Next.js Frontend
Django REST Framework (Core Banking API)
Python ML Services (Fraud Detection / Intelligence)
PostgreSQL Database

6.2 Architecture Description

1. Users interact with the system via the Next.js web application.
2. The frontend communicates with the Django REST Core Banking API.
3. The backend processes business logic and persists data in PostgreSQL.
4. For transaction analysis and intelligence, the backend interacts with Python-based AI/ML services.
5. Responses are securely returned to the frontend based on authenticated user roles.

7 Conclusion

This SRS defines a secure, scalable, and industry-aligned AI-powered digital banking platform. The selected technology stack improves development efficiency while preserving enterprise-grade architecture, security, and scalability. The system fully satisfies all functional and non-functional requirements and is suitable for academic evaluation and practical implementation.