**Project Title: Banking Management System**

**Team Members:**

**Akhil Vutla -2310030110**  
Harsha Vardhan -2310030109

Shivaanjan -2310030325

Ahsan -2310030213

**1. Introduction**

The **Banking Management System** is designed to streamline, automate, and enhance banking operations including customer account management, transactions, loan processing, and security. This project aims to create a secure, scalable, and user-friendly digital platform that helps banks manage their services efficiently. This document outlines the methodology and application development process involved in the creation of the system.

**2. Methodology**

**2.1 Software Development Life Cycle (SDLC)**

For the development of this system, we adopted the **Agile Software Development** methodology. Agile is chosen due to its iterative approach, which allows continuous evaluation and improvement of the system through multiple development cycles known as "sprints."

**Key Phases of SDLC Followed:**

1. **Requirement Gathering**
2. **System Design**
3. **Implementation (Coding)**
4. **Testing**
5. **Deployment**
6. **Maintenance**

**2.2 Requirement Gathering**

The first stage involved identifying and analyzing the core requirements of the banking system. The stakeholders included:

* Bank Managers
* IT Administrators
* Bank Employees
* Customers

**Functional Requirements:**

* Account creation (savings/current)
* Login and authentication system
* Deposit/withdrawal operations
* Funds transfer
* Loan application and processing
* Account statement generation
* Admin module for system management

**Non-Functional Requirements:**

* High security
* Scalability for growing number of users
* Fast and responsive user interface
* Backup and data recovery support

**2.3 Feasibility Study**

Before moving into design, a feasibility analysis was performed:

* **Technical Feasibility:** The project can be implemented using current technologies such as Python/Django for backend and ReactJS for frontend.
* **Operational Feasibility:** The system aligns with current banking procedures and can integrate with existing frameworks.
* **Economic Feasibility:** Development and deployment costs are within acceptable limits considering cloud deployment and open-source tools.

**3. Application Development**

**3.1 System Architecture**

The Banking Management System follows a **three-tier architecture**:

1. **Presentation Layer (Frontend):** User Interface developed using ReactJS and Bootstrap.
2. **Application Layer (Backend):** Handles business logic and is built using Django (Python).
3. **Data Layer (Database):** MySQL or PostgreSQL used for managing customer and transaction data.

**3.2 Technology Stack**

| **Component** | **Technology Used** |
| --- | --- |
| Frontend | ReactJS, Bootstrap |
| Backend | Python (Django) |
| Database | MySQL/PostgreSQL |
| Authentication | JWT (JSON Web Tokens) |
| Hosting | AWS/Heroku |
| Version Control | Git & GitHub |

**3.3 Modules of the System**

**3.3.1 User Module**

* Customer Registration & Login
* View account balance
* Deposit & Withdraw funds
* Transfer money
* View transaction history

**3.3.2 Admin Module**

* Approve/Reject account creation
* Manage user accounts
* Generate reports
* Monitor suspicious activities

**3.3.3 Employee Module**

* Assist with customer onboarding
* Process loan applications
* Respond to customer queries

**3.4 Database Design**

Key Tables:

* Users: Stores user credentials and personal details.
* Accounts: Stores account type, balance, and status.
* Transactions: Logs all fund movements with timestamp.
* Loans: Tracks loan applications, status, and repayment.
* Admins: Stores admin login and access data.

Entity-Relationship (ER) modeling was used for designing the database schema.

**4. Implementation Process**

**4.1 Frontend Implementation**

Using ReactJS, a dynamic and responsive interface was built. Pages include:

* **Login/Signup Page**
* **Dashboard**
* **Transfer Funds**
* **Transaction History**
* **Loan Application Form**
* **Admin Console**

**4.2 Backend Implementation**

Django was used to handle the backend services:

* Routing and APIs (RESTful)
* Data validation
* Business logic processing
* Session and Token management
* Middleware for error handling

**4.3 Security Measures**

Banking applications require robust security features:

* **Password Encryption**: Using bcrypt and hashlib
* **JWT Authentication**: Secure login sessions
* **Input Validation**: Prevent SQL injection and XSS
* **HTTPS Enforcement**: SSL certificate used
* **Role-based Access Control**: Admin, Employee, Customer levels

**4.4 Integration**

APIs were integrated using **Axios** in the frontend, communicating with Django REST APIs on the backend.

* GET/POST methods for account info and transactions
* Authentication headers with tokens
* JSON responses handled and rendered on frontend

**5. Testing**

The system underwent rigorous testing in each development cycle.

**5.1 Unit Testing**

* Each module and function tested individually using **PyTest**.
* Backend APIs tested for correct response codes.

**5.2 Integration Testing**

* Ensured communication between frontend and backend using Postman and Selenium.

**5.3 System Testing**

* Simulated full operations such as deposit, withdrawal, and transfers to validate end-to-end flow.

**5.4 User Acceptance Testing (UAT)**

* A sample group of users evaluated the UI/UX.
* Feedback was incorporated into final sprint changes.

**6. Deployment**

The final product was deployed using:

* **Cloud Platform:** Heroku (for quick deployment) / AWS EC2 (for scalability)
* **Database Hosting:** Amazon RDS or ClearDB
* **Continuous Integration/Deployment (CI/CD):** GitHub Actions used for automatic builds and tests.

**7. Maintenance & Future Enhancements**

The project includes provisions for post-deployment maintenance, including:

* Weekly backups
* Error logging
* Security patch updates
* User feedback integration

**Planned Features for Future:**

* Mobile App Interface (React Native)
* Biometric Authentication
* AI-driven customer support chatbot
* Real-time fraud detection using machine learning

**8. Conclusion**

This Banking Management System demonstrates a secure, scalable, and user-friendly digital banking solution. The application of Agile methodology allowed rapid iteration and responsiveness to feedback, resulting in a robust product ready for real-world use. The system meets all primary objectives: security, efficiency, and ease of use, and is adaptable for future enhancements.