**Industrial Internship Report on**

**”Banking Information System”**

**Prepared by**

**Navanshu Maheshwari**

|  |
| --- |
| *Executive Summary* |
| This report provides details of the Industrial Internship provided by upskill Campus and The IoT Academy in collaboration with Industrial Partner UniConverge Technologies Pvt Ltd (UCT).  This internship was focused on a project/problem statement provided by UCT. We had to finish the project including the report in 6 weeks’ time.  My project was Banking Information System which include the implementation of key functionalities including user registration, account management, deposit and withdrawal transactions, fund transfers, account statements, password protection, and a user-friendly interface using SQL database connectivity and Swing components. The project aimed to create a comprehensive banking information system that offers users a seamless experience for managing their accounts and conducting financial transactions. Leveraging Java's capabilities and integrating SQL database connectivity and Swing components, the system ensures secure data storage, efficient user interaction, and robust transaction processing.  This internship gave me a very good opportunity to get exposure to Industrial problems and design/implement solution for that. It was an overall great experience to have this internship. |

**TABLE OF CONTENTS**

[1 Preface 3](#_Toc139702806)

[2 Introduction 4](#_Toc139702807)

[2.1 About UniConverge Technologies Pvt Ltd 4](#_Toc139702808)

[2.2 About upskill Campus 8](#_Toc139702809)

[2.3 Objective 9](#_Toc139702810)

[2.4 Reference 9](#_Toc139702811)

[2.5 Glossary 10](#_Toc139702812)

[3 Problem Statement 11](#_Toc139702813)

[4 Existing and Proposed solution 12](#_Toc139702814)

[5 Proposed Design/ Model 13](#_Toc139702815)

[5.1 High Level Diagram (if applicable) 13](#_Toc139702816)

[5.2 Low Level Diagram (if applicable) 13](#_Toc139702817)

[5.3 Interfaces (if applicable) 13](#_Toc139702818)

[6 Performance Test 14](#_Toc139702819)

[6.1 Test Plan/ Test Cases 14](#_Toc139702820)

[6.2 Test Procedure 14](#_Toc139702821)

[6.3 Performance Outcome 14](#_Toc139702822)

[7 My learnings 15](#_Toc139702823)

[8 Future work scope 16](#_Toc139702824)

# Preface

Summary of the whole 6 weeks’ work.

About need of relevant Internship in career development.

Brief about Your project/problem statement.

Opportunity given by USC/UCT.

How Program was planned



Your Learnings and overall experience.

Thank to all (with names), who have helped you directly or indirectly.

Your message to your juniors and peers.

# Introduction

## About UniConverge Technologies Pvt Ltd

A company established in 2013 and working in Digital Transformation domain and providing Industrial solutions with prime focus on sustainability and RoI.

For developing its products and solutions it is leveraging various**Cutting Edge Technologies e.g. Internet of Things (IoT), Cyber Security, Cloud computing (AWS, Azure), Machine Learning, Communication Technologies (4G/5G/LoRaWAN), Java Full Stack, Python, Front end**etc.



1. UCT IoT Platform **(****)**

**UCT Insight** is an IOT platform designed for quick deployment of IOT applications on the same time providing valuable “insight” for your process/business. It has been built in Java for backend and ReactJS for Front end. It has support for MySQL and various NoSql Databases.

* It enables device connectivity via industry standard IoT protocols - MQTT, CoAP, HTTP, Modbus TCP, OPC UA
* It supports both cloud and on-premises deployments.

It has features to  
• Build Your own dashboard  
• Analytics and Reporting  
• Alert and Notification  
• Integration with third party application(Power BI, SAP, ERP)  
• Rule Engine

 

1. **Smart Factory Platform (****)**

Factory watch is a platform for smart factory needs.

It provides Users/ Factory

* with a scalable solution for their Production and asset monitoring
* OEE and predictive maintenance solution scaling up to digital twin for your assets.
* to unleased the true potential of the data that their machines are generating and helps to identify the KPIs and also improve them.
* A modular architecture that allows users to choose the service that they what to start and then can scale to more complex solutions as per their demands.

Its unique SaaS model helps users to save time, cost and money.





1.  based Solution

UCT is one of the early adopters of LoRAWAN teschnology and providing solution in Agritech, Smart cities, Industrial Monitoring, Smart Street Light, Smart Water/ Gas/ Electricity metering solutions etc.

1. Predictive Maintenance

UCT is providing Industrial Machine health monitoring and Predictive maintenance solution leveraging Embedded system, Industrial IoT and Machine Learning Technologies by finding Remaining useful life time of various Machines used in production process.



## About upskill Campus (USC)

upskill Campus along with The IoT Academy and in association with Uniconverge technologies has facilitated the smooth execution of the complete internship process.

USC is a career development platform that delivers **personalized executive coaching** in a more affordable, scalable and measurable way.



Seeing need of upskilling in self paced manner along-with additional support services e.g. Internship, projects, interaction with Industry experts, Career growth Services

<https://www.upskillcampus.com/>

upSkill Campus aiming to upskill 1 million learners in next 5 year



## The IoT Academy

The IoT academy is EdTech Division of UCT that is running long executive certification programs in collaboration with EICT Academy, IITK, IITR and IITG in multiple domains.

## Objectives of this Internship program

The objective for this internship program was to

 ☛ get practical experience of working in the industry.

 ☛ to solve real world problems.

 ☛ to have improved job prospects.

 ☛ to have Improved understanding of our field and its applications.

 ☛ to have Personal growth like better communication and problem solving.

## Reference

[1] https://www.uniconvergetech.in/about-us

[2] https://docs.oracle.com/en/java/

[3] https://dev.mysql.com/doc/

# Problem Statement

Develop a prototype of a Banking Information System in Core Java that provides a working preview of the key functionalities of a real banking system. The prototype should demonstrate the core features and flow of the system, showcasing its functionality and usability.

Key Functionality to Include in the Prototype:

1. User Registration: Implement a simplified user registration process where users can provide basic details to create an account.
2. Account Management: Develop the ability to create and manage user accounts, including assigning unique account numbers and tracking account balances.
3. Deposit and Withdrawal: Enable users to make deposits and withdrawals from their accounts, updating the account balance accordingly.
4. Fund Transfer: Implement a simplified version of fund transfer functionality, allowing users to transfer funds between their accounts or to other registered users.
5. Account Statements: Provide users with a preview of their account statements, displaying transaction history, dates, amounts, and remaining balances.
6. Password Protection: Develop a basic login system with password authentication to ensure secure access to user accounts.
7. Error Handling: Implement basic error handling mechanisms to handle common exceptions, such as insufficient funds and invalid transactions, and display relevant error messages to users.
8. User Interface: Design a user-friendly interface for the prototype that allows users to navigate through the system, perform banking operations, and view relevant information.
9. Persistence: Implement basic data persistence by storing user account information and transaction history temporarily during the prototype session.

By developing this prototype, stakeholders will have a tangible working preview of the key features and functionality of the Banking Information System. This will allow them to evaluate the system's usability, identify any necessary improvements or enhancements, and make informed decisions for further development and deployment of the complete system.

# Existing and Proposed solution

The proposed solution for the Banking Information System Java project aims to create a comprehensive and user-friendly platform that encompasses key functionalities including user registration, account management, deposit and withdrawal transactions, fund transfers, account statements, password protection, and a responsive user interface. The solution leverages SQL database connectivity for efficient data management and utilizes Swing components to ensure an intuitive user experience. Here's an outline of the proposed solution:

**1. User Registration:**

* A user-friendly registration form will capture user details such as username, password, email, and personal information.
* Input validation will ensure data integrity and prevent erroneous entries.
* Upon successful registration, user data will be securely stored in the MySQL database.

**2. Account Management:**

* Users will have access to a dashboard displaying their account details, including balance and transaction history.
* Account information will be retrieved from the database and displayed using Swing components.
* Users can update their contact details and preferences through a user-friendly interface.

**3. Deposit and Withdrawal:**

* Users will have the option to deposit funds into their accounts by entering the desired amount.
* Withdrawal transactions will subtract the specified amount from the account balance.
* Appropriate validation checks will be implemented to prevent overdrawn accounts and ensure successful transactions.

**4. Fund Transfer:**

* A secure fund transfer module will enable users to transfer money between their accounts or to other user accounts.
* Validation will be performed to confirm the availability of funds and verify recipient account details.
* Successful transfers will be reflected in updated account balances and transaction records.

**5. Account Statements:**

* Users will be able to generate and view account statements that display their transaction history.
* The system will retrieve transaction data from the database and present it in a clear and organized format.
* Filters and sorting options will enhance the user's ability to navigate their transaction history.

**6. Password Protection:**

* User passwords will be stored using secure encryption techniques to ensure confidentiality.
* Password change functionality will require users to provide their existing password and a new secure password.
* Robust password policies will be enforced to enhance account security.

**7. User Interface :**

* The user interface will be developed using Java's Swing library, ensuring a consistent and visually appealing design.
* Components such as buttons, text fields, labels, and tables will be strategically placed to provide an intuitive user experience.
* Real-time feedback and error messages will guide users through their interactions.

**8. SQL Database Connectivity:**

* The solution will utilize JDBC for seamless connectivity with the MySQL database.
* SQL queries will be employed to manage data transactions, including user registration, account updates, and transaction recording.

The proposed solution combines the power of SQL database connectivity and Swing components to create a robust and user-centric banking information system. By prioritizing security, usability, and functionality, this solution aims to provide users with a reliable platform for managing their banking operations efficiently and securely.

## Code submission (Github link): https://github.com/navanshu1234/upskill\_campus/tree/master/bankingsystem

## Report submission (Github link) : https://github.com/navanshu1234/upskill\_campus/tree/master/bankingsystem

# Proposed Design/ Model

The Banking Information System project aims to develop a Java-based application that provides essential functionalities for a banking system. The project includes user registration, account management, deposit and withdrawal transactions, fund transfers, account statements, password protection, and a user-friendly interface. The system utilizes SQL database connectivity and Swing components to achieve interaction with the database and create a friendly user interface.

The Banking Information System is built using a layered architecture consisting of the following components:

* Presentation Layer: The user interface is developed using Swing components.
* Business Logic Layer: This layer contains the Java code responsible for implementing the banking functionalities, including user registration, account management, transactions, and password protection.
* Data Access Layer: SQL queries are used to interact with the database to store and retrieve user account and transaction data.

The sysyem consists of following functions:

**i. Banking Information class:**

* A public banking information class is created which extends the frame class to access java frame components. The frame components are used to design the front end of the banking information system through which the user interact with the system.
* Two functions are declared in the constructor of this class for creating user interface and database connectivity.

**ii. User interface function:**

* The user interface function is created to initialize all the components used in creating the front end of the banking information system.
* In this function, the frame components are used in creation of user interface. Components such as ‘JLabel’ is used to display the username field, password field, amount field.
* Different kind of buttons are initialized in this function through which user interact with the system. The ‘JButton’ component is used to initialize the buttons for different functionalities such as register, login, deposit, withdrawal, transfer and logout.
* The buttons provide ‘onclick’ event trigger for various functionalities using ActionListerner interface.

**iii. Database connectivity function:**

* A function for database connection is created to connect the banking information system to database which store the data of the user account information.
* This function uses a connection variable to establish a connection with database through driver manager by passing the link to database, username and password parameter in it.
* This function will connect the system to database of user which match the username and password.

**iv. Database create table function:**

* This create table function is created to create a new table of account information of a user if not present in the database.
* In this function a statement to create new table is implemented which is executed to create a table of username, password and balance of the user in the database.

**v. User Register function:**

* This function is created to register a new user in the banking information system
* Through this function, a new user register after input of detail of user.

**vi. User login Function:**

* This function is created to login the existing user in the banking information system so the user can interact with the system.
* The existing user access the system by entering username and password detail.
* The username and password is verified by matching it with username and corresponding password present in the database.
* If the password is correct then it will display login successful message otherwise login failed message.

**vii. Home Screen function:**

* After successful login of user, the user can access different functionalities such as withdraw, deposit, transfer and show account information through home screen.

The home screen consist of button which link the on click functionalities of other function such as withdraw, deposit, transfer, show account information and logout

**viii. Deposit Amount function:**

* This function provides the functionality to deposit a certain amount of value in user account.
* This function updates the value of amount in user account in the database.
* This function is implemented as:

Statement = connection.createStatement();

String query = "UPDATE accounts SET balance=balance+" + amount + " WHERE username='" + currentUser + "'";

statement.executeUpdate(query);

**ix. Withdraw amount function:**

* This function provides the functionality to withdraw a certain amount of value from user account.
* This function updates the value of amount in user account in the database.
* This function is implemented as:

if (balance >= amount) {

query = "UPDATE accounts SET balance=balance-" + amount + " WHERE username='" + currentUser + "'";

statement.executeUpdate(query);

x.**Transfer amount function:**

* This function provides the functionality to transfer the amount from one account to another user account.
* The function select the users account in which transfer is performed and execute the database query to make changes in the amount of both user account by depositing the amount in one account and deducting from another account.

This function can be implemented as:

if (balance >= amount) {

query = "UPDATE accounts SET balance=balance-" + amount + " WHERE username='" + currentUser + "'";

statement.executeUpdate(query);

query = "UPDATE accounts SET balance=balance+" + amount + " WHERE username='" + recipient + "'";

statement.executeUpdate(query);

**x. Show Account information Function:**

* This function provides the functionality to show the current amount present in user account.
* This function is implemented by executing the query as:

Statement statement = connection.createStatement();

String query = "SELECT balance FROM accounts WHERE username='" + currentUser + "'";

ResultSet resultSet = statement.executeQuery(query);

if (resultSet.next()) {

double balance = resultSet.getDouble("balance");

statementArea.setText("Account statement for " + currentUser + "\n");

statementArea.append("Balance: " + balance + "\n");

## Interfaces

# Performance Test

Performance testing for the Banking Information System Java project involves assessing how the system behaves under different load conditions for key functionalities. Here are various performance test scenarios that can be conducted for each functionality:

**1. User Registration:**

* **Test Scenario:** Simulate a gradual increase in the number of concurrent users registering.
* **Objective:** Evaluate how the system responds as the number of registration requests increases.

**2. Account Management:**

* **Test Scenario:** Simulate multiple users accessing and updating their account details simultaneously.
* **Objective:** Assess the system's ability to handle concurrent account management activities.

**3. Deposit and Withdrawal Transactions:**

* **Test Scenario:** Execute a mix of deposit and withdrawal transactions with varying transaction volumes.
* **Objective:** Measure the response time and throughput for processing financial transactions.

**4. Fund Transfer:**

* **Test Scenario:** Simulate fund transfers between accounts, considering varying transaction complexities.
* **Objective:** Evaluate the system's performance in processing fund transfers accurately and promptly.

**5. Account Statements:**

* **Test Scenario:** Generate account statements for a group of users with different transaction histories.
* **Objective:** Assess the time taken to retrieve and present account statements with varying data sizes.

**6. Password Protection:**

* **Test Scenario:** Simulate concurrent login attempts with incorrect passwords.
* **Objective:** Determine how the system handles authentication requests under load and assess any potential vulnerabilities.

**7. User Interface Responsiveness:**

* **Test Scenario:** Execute user actions like button clicks, data input, and navigation across the application.
* **Objective:** Measure the responsiveness of the user interface under different interaction scenarios.

**8. Database Performance Test:**

* **Test Scenario:** Execute database-intensive operations like generating account statements for a large number of users.
* **Objective:** Measure the database's responsiveness and efficiency under heavy load.

## Test Plan/ Test Cases

**Test Case 1: Successful User Registration**

* **Objective:** To verify that a user can successfully register with valid information.
* **Steps:**
  1. Navigate to the registration page.
  2. Enter valid user details (username, password, email, personal information).
  3. Click the "Register" button.
* **Expected Result:** User is registered successfully and a confirmation message is displayed.

**Test Case 2: Invalid User Registration**

* **Objective:** To ensure that the system rejects invalid or incomplete user registration data.
* **Steps:**
  1. Navigate to the registration page.
  2. Enter incomplete or invalid user details.
  3. Click the "Register" button.
* **Expected Result:** An error message is displayed, indicating that the registration is unsuccessful.

**2. Account Management:**

**Test Case 3: View Account Details**

* **Objective:** To verify that users can view their account details accurately.
* **Steps:**
  1. Log in to the system.
  2. Navigate to the account management section.
  3. Click on the "View Account Details" option.
* **Expected Result:** User's account details are displayed correctly.

**Test Case 4: Update Account Information**

* **Objective:** To ensure that users can update their account information successfully.
* **Steps:**
  1. Log in to the system.
  2. Navigate to the account management section.
  3. Click on the "Update Account" option.
  4. Make changes to account information.
  5. Click the "Save" button.
* **Expected Result:** Account information is updated, and a confirmation message is displayed.

**3. Deposit and Withdrawal:**

**Test Case 5: Deposit Funds**

* **Objective:** To confirm that users can deposit funds into their account.
* **Steps:**
  1. Log in to the system.
  2. Navigate to the deposit section.
  3. Enter the deposit amount.
  4. Click the "Deposit" button.
* **Expected Result:** Account balance is updated, and the transaction is recorded.

**Test Case 6: Withdraw Funds**

* **Objective:** To verify that users can withdraw funds from their account.
* **Steps:**
  1. Log in to the system.
  2. Navigate to the withdrawal section.
  3. Enter the withdrawal amount.
  4. Click the "Withdraw" button.
* **Expected Result:** Account balance is updated, and the transaction is recorded.

**4. Fund Transfer:**

**Test Case 7: Successful Fund Transfer**

* **Objective:** To ensure that users can successfully transfer funds between accounts.
* **Steps:**
  1. Log in to the system.
  2. Navigate to the fund transfer section.
  3. Enter recipient account details and transfer amount.
  4. Click the "Transfer" button.
* **Expected Result:** Funds are transferred between accounts, and both account balances are updated.

**Test Case 8: Insufficient Funds for Fund Transfer**

* **Objective:** To verify that the system prevents fund transfers if the sender has insufficient funds.
* **Steps:**
  1. Log in to the system.
  2. Navigate to the fund transfer section.
  3. Enter recipient account details and transfer amount exceeding account balance.
  4. Click the "Transfer" button.
* **Expected Result:** An error message is displayed, and the transfer is declined.

## Test Procedure

**1. User Registration:**

**Test Procedure 1: Successful User Registration**

1. Navigate to the registration page.
2. Enter valid user details (username, password, email, personal information).
3. Click the "Register" button.
4. Verify that the user is redirected to a confirmation page.
5. Log in using the registered credentials and confirm successful login.

**Test Procedure 2: Invalid User Registration**

1. Navigate to the registration page.
2. Enter incomplete or invalid user details.
3. Click the "Register" button.
4. Verify that an error message is displayed, indicating unsuccessful registration.
5. Attempt to log in using the incomplete/invalid credentials and confirm login failure.

**2. Account Management:**

**Test Procedure 3: View Account Details**

1. Log in to the system.
2. Navigate to the account management section.
3. Click on the "View Account Details" option.
4. Verify that the displayed account details match the user's information.

**Test Procedure 4: Update Account Information**

1. Log in to the system.
2. Navigate to the account management section.
3. Click on the "Update Account" option.
4. Make changes to account information (e.g., contact details).
5. Click the "Save" button.
6. Verify that the changes are saved and reflected in the account details.

**3. Deposit and Withdrawal:**

**Test Procedure 5: Deposit Funds**

1. Log in to the system.
2. Navigate to the deposit section.
3. Enter the deposit amount.
4. Click the "Deposit" button.
5. Verify that the account balance is updated by the deposited amount.

**Test Procedure 6: Withdraw Funds**

1. Log in to the system.
2. Navigate to the withdrawal section.
3. Enter the withdrawal amount.
4. Click the "Withdraw" button.
5. Verify that the account balance is updated after the withdrawal.

**4. Fund Transfer:**

**Test Procedure 7: Successful Fund Transfer**

1. Log in to the system as sender.
2. Navigate to the fund transfer section.
3. Enter recipient account details and transfer amount.
4. Click the "Transfer" button.
5. Verify that both sender's and recipient's account balances are updated correctly.

**Test Procedure 8: Insufficient Funds for Fund Transfer**

1. Log in to the system as sender.
2. Navigate to the fund transfer section.
3. Enter recipient account details and transfer amount exceeding account balance.
4. Click the "Transfer" button.
5. Verify that an error message is displayed, indicating insufficient funds.

## Performance Outcome

The performance outcomes for the Banking Information System Java project, encompassing key functionalities like user registration, account management, deposit and withdrawal, fund transfer, account statements, password protection, and user interface using SQL database connectivity and Swing components, are crucial indicators of its effectiveness and efficiency. Here are the expected performance outcomes:

**1. Responsive User Interface:** The user interface should respond promptly to user interactions, ensuring a smooth and engaging user experience. Actions like button clicks, data input, and navigation should have minimal delay.

**2. Quick Registration Process:** The user registration process should be swift, allowing new users to complete their registration without significant delays. The system should handle concurrent registration requests efficiently.

**3. Account Management Efficiency:** Viewing and updating account details should occur promptly, ensuring users can access and modify their information without frustration. Account updates should reflect accurately in real time.

**4. Fast Transaction Processing:** Deposit and withdrawal transactions, as well as fund transfers, should be processed swiftly. Users should experience minimal waiting times during these critical financial activities.

**5. Timely Account Statement Generation:** Account statements should be generated promptly, providing users with a clear overview of their transaction history within a reasonable timeframe.

**6. Secure Password Protection:** Password protection mechanisms should provide rapid authentication while maintaining strong security. Users should experience quick login procedures without compromising data safety.

**7. Scalability under Load:** The system should demonstrate scalability, maintaining acceptable performance levels even under heavy user loads. The application's responsiveness should not degrade as user numbers increase.

**8. Minimal Resource Consumption:** The application should be resource-efficient, utilizing system resources like memory and CPU power judiciously. High resource consumption could lead to system slowdowns or crashes.

**9. Consistent Database Connectivity:** The system's connection to the SQL database should remain stable and reliable. This ensures that data can be retrieved and stored efficiently without interruptions.

**10. Reliable Data Integrity:** All financial transactions and data storage should maintain data integrity. Deposits, withdrawals, and fund transfers should be accurately recorded, preventing discrepancies.

# My learnings

Throughout the development of the Banking Information System Java project with key functionalities including user registration, account management, deposit and withdrawal, fund transfer, account statements, password protection, and user interface using SQL database connectivity and Swing components, I've gained valuable insights and learning experiences:

**1. Comprehensive Understanding of Banking Operations:** I learned how banks function at a technological level, from user registration to transaction processing and account management. This understanding deepened my knowledge of the financial sector.

**2. Java Programming Proficiency:** Developing this project enhanced my Java programming skills. I gained proficiency in working with Swing components for creating graphical user interfaces and utilizing SQL database connectivity for efficient data management.

**3. User-Centric Design Approach:** I realized the importance of putting users at the center of design decisions. The user interface's usability and intuitiveness directly impact the overall user experience.

**4. Data Security Best Practices:** Implementing password protection using encryption techniques highlighted the significance of data security. I learned about hashing passwords and applying secure authentication methods.

**5. Iterative Development and Refinement:** Developing the project in stages allowed for iterative improvement. Regularly refining functionalities based on feedback and testing results improved the overall quality of the system.

**6. Rigorous Testing Methods:** Extensive testing, including functional, performance, and security testing, proved pivotal in identifying and addressing issues early. This experience solidified the importance of rigorous testing in software development.

**7. Handling Financial Transactions:** Implementing deposit, withdrawal, and fund transfer functionalities taught me the intricacies of processing financial transactions accurately and securely.

**8. Real-world Database Connectivity:** Working with SQL database connectivity provided hands-on experience in managing data, executing queries, and ensuring data integrity in real-world scenarios.

**9. Collaboration and Communication:** The project required coordination among team members, ensuring consistent communication and collaborative problem-solving. These skills are crucial for successful software development projects.

**10. Documentation's Value:** Creating comprehensive documentation, including test plans, reports, and user manuals, demonstrated its importance in maintaining, improving, and transferring the project.

# Future work scope

While the current version of the Banking Information System Java project covers key functionalities like user registration, account management, deposit and withdrawal, fund transfer, account statements, password protection, and user interface using SQL database connectivity and Swing components, there are several avenues for future work and enhancements:

**1. Enhanced User Interface:**

* Consider modernizing the user interface using more advanced UI libraries or frameworks to provide a more visually appealing and interactive experience.
* Implement responsive design principles to ensure seamless user interactions across different devices and screen sizes.

**2. Multi-Language Support:**

* Extend the system to support multiple languages, catering to a diverse user base and ensuring better accessibility.

**3. Advanced Security Measures:**

* Implement two-factor authentication (2FA) for an additional layer of security during login.
* Explore biometric authentication methods such as fingerprint or facial recognition.

**4. Multi-Currency Support:**

* Enable users to operate accounts in various currencies, facilitating international transactions.

**5. Mobile App Development:**

* Develop a mobile application for the banking system to provide users with the flexibility to manage their accounts on the go.

**6. Transaction Alerts:**

* Implement real-time transaction alerts through email or SMS to keep users informed about account activities.

**7. Enhanced Reporting and Analytics:**

* Develop more advanced account statement generation with filtering options, graphical representations, and export functionalities.

**8. Integration with Financial APIs:**

* Integrate the system with financial APIs to provide real-time market data, exchange rates, and investment tracking.