Comsats University Islamabad, Lahore Campus

Object Oriented Programming

Assignment Phase 1

Designing and Planning

Project: Online Banking System

Submitted By:

- Mateen Haider (FA22-BSE-209)
- Aamir Mubeen (FA23-BSE-032)
- Shoaib Shahid (FA23-BSE-137)

Submitted To :

- Mam Muntaha Iqbal

Submission Date

March 22 , 2024

A) Define the requirements and scope of the project

Following are the requirements and scope included in our project of Online Banking System.

Requirements

Bank:

The system should support the functionalities required by a banking institution, including managing customer accounts, facilitating transactions, and providing various banking services through an online platform.

Customer:

Users should be able to register for online banking services, access their accounts securely, and perform various transactions online.

Administrator:

Administrators should have access to manage system configurations, user accounts, security settings, and generate reports.

Account:

Each customer should have one or more accounts associated with their profile, including savings and current accounts.

Savings Account:

Customers should be able to open savings accounts online with features such as interest calculation, deposits, withdrawals, and account statements accessible through the online platform.

Current Account:

Customers should be able to open current accounts online with features such as overdraft limits, checks, deposits, withdrawals, and account statements accessible through the online platform.

Transaction:

The system should support various types of online transactions including deposits, withdrawals, transfers, bill payments, and cheque services.

Transaction History:

A detailed transaction history should be maintained for each account, accessible to customers and administrators through the online platform.

ATM:

Integration with ATMs should allow customers to perform online transactions such as cash withdrawals, balance inquiries, and PIN changes.

Validation:

Robust validation mechanisms should be in place for user authentication, transaction authorization, and data integrity to ensure secure online banking.

Interest Calculator:

The system should calculate interest on savings accounts based on predefined interest rates and account balances, with results visible to customers through the online platform.

Bill Payment Service:

Customers should be able to pay bills for utilities, services, and other expenses online through the banking system.

Funds Transfer Service:

Customers should be able to transfer funds between their own accounts or to other accounts within the same bank or different banks through the online platform.

Cheque Services:

The system should support online cheque issuance, deposit, clearance, and cancellation services.

Utility Services:

Integration with utility providers should facilitate online payments for services like electricity, water, and internet through the banking system.

Scope

1. User Registration and Authentication:

- Users should be able to register for online banking services by providing personal information such as name, address, contact details, and identification documents.
- Robust authentication mechanisms should be implemented, including username/password combinations, biometric authentication (such as fingerprint or facial recognition), and two-factor authentication (2FA) for enhanced security.

2. Dashboard and Account Overview:

- Upon logging in, users should be greeted with a personalized dashboard displaying an overview of their accounts, recent transactions, pending payments, and account balances.
- Account summaries should provide detailed information about each account type, including interest rates, available balances, and any associated fees or restrictions.

3. Account Management:

- Users should have the ability to manage their accounts online, including opening new accounts, closing accounts, and updating account details such as contact information and beneficiaries.
- Features for setting up recurring payments, creating savings goals, and configuring account preferences (e.g., notifications, language preferences) should be available.

4. Transaction Features:

- The system should support a wide range of transaction types, including:
  - ❖ Fund transfers between accounts (within the same bank or to external accounts).
  - ❖ Bill payments for utilities, credit cards, loans, and other services.
  - ❖ Check deposits through mobile check deposit or electronic check scanning.
  - ❖ Wire transfers for domestic and international transactions.
- Real-time transaction processing and immediate confirmation should be provided to users, along with transaction receipts and confirmation emails.

5. Security Measures:

- Implement industry-standard security protocols such as Transport Layer Security (TLS) encryption to protect data transmission between the user's device and the banking server.
- Regular security audits, vulnerability assessments, and penetration testing should be conducted to identify and mitigate potential security risks.
- Multi-layered security measures should be in place to prevent unauthorized access, fraudulent activities, and data breaches.

6. Analytics and Reporting:

- Provide comprehensive analytics and reporting tools for both customers and administrators, enabling them to track spending patterns, monitor account activity, and generate custom reports.

- Advanced analytics features may include budgeting tools, financial forecasting, and personalized recommendations based on spending habits and financial goals.

7. Scalability and Performance:

- Design the online banking system to be scalable and capable of handling increasing user volumes, transaction volumes, and data storage requirements over time.
- Optimize system performance through techniques such as caching, load balancing, and database indexing to ensure responsiveness and reliability during peak usage periods.

8. Compliance and Regulatory Requirements:

- Ensure compliance with regulatory standards such as the Payment Card Industry Data Security Standard (PCI DSS), General Data Protection Regulation (GDPR), and financial regulations enforced by regulatory authorities in relevant jurisdictions.
- Maintain audit trails, data retention policies, and documentation to demonstrate compliance with legal and regulatory requirements.

B) Identify the classes and methods needed for your application.

Following are the classes and their respective methods that would be included in our Online Banking System:

1)Bank

Attributes:

- ❖ bankName (private): String
- ❖ bankAddress (private): String
- ❖ customers (private): List<Customer>
- ❖ accounts (private): List<Account>

Methods:

- Bank(String bankName, String bankAddress): Constructor to initialize bank details.
- getBankName(): String (public): Returns the name of the bank.
- setBankName(String bankName): void (public): Sets the name of the bank.
- getBankAddress(): String (public): Returns the address of the bank.
- setBankAddress(String bankAddress): void (public): Sets the address of the bank.
- addCustomer(Customer customer): void (public): Adds a customer to the bank's customer list.
- removeCustomer(Customer customer): void (public): Removes a customer from the bank's customer list.
- getCustomers(): List<Customer> (public): Returns the list of customers.
- addAccount(Account account): void (public): Adds an account to the bank's account list.
- removeAccount(Account account): void (public): Removes an account from the bank's account list.
- getAccounts(): List<Account> (public): Returns the list of accounts.

2) Customer

Attributes:

- ❖ customerId (private): int
- ❖ firstName (private): String
- ❖ lastName (private): String
- ❖ email (private): String
- ❖ address (private): String
- ❖ phoneNumber (private): String

Methods:

- Customer(int customerId, String firstName, String lastName, String email, String address, String phoneNumber): Constructor to initialize customer details.
- getCustomerId(): int (public): Returns the customer ID.
- getFirstName(): String (public): Returns the first name of the customer.
- setFirstName(String firstName): void (public): Sets the first name of the customer.
- getLastName(): String (public): Returns the last name of the customer.
- setLastName(String lastName): void (public): Sets the last name of the customer.
- getEmail(): String (public): Returns the email address of the customer.
- setEmail(String email): void (public): Sets the email address of the customer.
- getAddress(): String (public): Returns the address of the customer.
- setAddress(String address): void (public): Sets the address of the customer.
- getPhoneNumber(): String (public): Returns the phone number of the customer.
- setPhoneNumber(String phoneNumber): void (public): Sets the phone number of the customer.

3) Administrator

Attributes:

- ❖ adminId (private): String

Methods:

- ❖ Administrator(String adminId): Constructor to initialize administrator details.
- ❖ verifyUserCredentials(String username, String password): boolean (public): Verifies the login credentials of users.
- ❖ approveAccount(Account account): void (public): Approves account creation requests.
- ❖ blockAccount(Account account): void (public): Blocks or suspends a user's account.
- ❖ generateReports(): void (public): Generates various reports like transaction history, account summaries, etc.

4) Account

Attributes:

- ❖ accountId (private): String
- ❖ balance (private): double
- ❖ accountType (private): String
- ❖ owner (private): Customer
- ❖ transactions (private): List<Transaction>
- ❖ atmPin(private)

Methods:

- Account(String accountId, double balance, String accountType, Customer owner): Constructor to initialize account details.
- getAccountId(): String (public): Returns the account ID.
- setAccountId(String accountId): void (public): Sets the account ID.
- getBalance(): double (public): Returns the balance of the account.
- setBalance(double balance): void (public): Sets the balance of the account.
- getAccountType(): String (public): Returns the type of the account.
- setAccountType(String accountType): void (public): Sets the type of the account.
- getOwner(): Customer (public): Returns the owner of the account.
- setOwner(Customer owner): void (public): Sets the owner of the account.
- getTransactions(): List<Transaction> (public): Returns the list of transactions associated with the account.
- addTransaction(Transaction transaction): void (public): Adds a transaction to the account's transaction history.
- setAtmPin(String accountId) : sets ATM pin for specific account
- getAtmPin(String accountId): returns the accountId for specific account

5) Saving Account

Attributes:

- ❖ interestRate (private): double

Methods:

- SavingsAccount(String accountId, double balance, String accountType, Customer owner, double interestRate): Constructor to initialize savings account details.
- getInterestRate(): double (public): Returns the interest rate of the savings account.
- setInterestRate(double interestRate): void (public): Sets the interest rate of the savings account.

- calculateInterest(): double (public): Calculates and returns the interest earned based on the account balance and interest rate.

6) Current Account

Attributes:

- ❖ overdraftLimit (private): double

Methods:

- CurrentAccount(String accountId, double balance, String accountType, Customer owner, double overdraftLimit): Constructor to initialize current account details.
- getOverdraftLimit(): double (public): Returns the overdraft limit of the current account.
- setOverdraftLimit(double overdraftLimit): void (public): Sets the overdraft limit of the current account.
- withdraw(double amount): boolean (public): Withdraws the specified amount from the account, allowing overdraft within the limit.

7) Transaction

Attributes:

- ❖ transactionId (private): int
- ❖ amount (private): double
- ❖ date (private): Date
- ❖ description (private): String

Methods:

- Transaction(int transactionId, double amount, Date date, String description): Constructor to initialize transaction details.
- getTransactionId(): int (public): Returns the transaction ID.
- setTransactionId(int transactionId): void (public): Sets the transaction ID.
- getAmount(): double (public): Returns the amount involved in the transaction.
- setAmount(double amount): void (public): Sets the amount involved in the transaction.
- getDate(): Date (public): Returns the date of the transaction.
- setDate(Date date): void (public): Sets the date of the transaction.
- getDescription(): String (public): Returns the description of the transaction.
- setDescription(String description): void (public): Sets the description of the transaction.

8) TransactionHistory

Attributes:

- transactions (private): List<Transaction>

Methods:

- TransactionHistory(): Constructor to initialize the transaction history.
- addTransaction(Transaction transaction): void (public): Adds a transaction to the transaction history.
- getTransactions(): List<Transaction> (public): Returns the list of transactions.
- clearTransactions(): void (public): Clears the transaction history.
- generateReport(): void (public): Generates a report of the transaction history.

9)ATM

Attributes:

- atmId (private): String
- location (private): String
- cashAvailable (private): double

Methods:

- ATM(String atmId, String location, double initialCash): Constructor to initialize ATM details.
- getAtmId(): String (public): Returns the ID of the ATM.
- setAtmId(String atmId): void (public): Sets the ID of the ATM.
- getLocation(): String (public): Returns the location of the ATM.
- setLocation(String location): void (public): Sets the location of the ATM.
- getCashAvailable(): double (public): Returns the amount of cash available in the ATM.
- setCashAvailable(double cashAvailable): void (public): Sets the amount of cash available in the ATM.
- withdrawCash(double amount): boolean (public): Withdraws cash from the ATM if sufficient funds are available.
- depositCash(double amount): void (public): Deposits cash into the ATM.

10)Validations

Attributes:

No attribute

Methods:

- isEmailValid(String email): boolean (public): Validates the format of an email address.
- isPhoneNumberValid(String phoneNumber): boolean (public): Validates the format of a phone number.
- isAmountValid(double amount): boolean (public): Validates if the amount is non-negative.
- isAccountNumberValid(String accountNumber): boolean (public): Validates the format of an account number.
- isPasswordValid(String password): boolean (public): Validates the strength of a password (e.g., minimum length, special characters).

11) InterestCaclculator

Attributes:

No attribute

Methods:

- calculateInterest(SavingsAccount savingsAccount): double (public): Calculates the interest earned on a savings account based on its balance and interest rate.

12)BillPaymentServices

Attributes:

No attribute

Methods:

- payElectricityBill(Account account, String provider, double amount): boolean (public): Processes a payment for electricity bill from the specified account to the given provider for the specified amount.

- payEducationBill(Account account, String institution, double amount): boolean (public): Processes a payment for education bill from the specified account to the given institution for the specified amount.
- payWaterBill(Account account, String utilityCompany, double amount): boolean (public): Processes a payment for water bill from the specified account to the given utility company for the specified amount.
- payGasBill(Account account, String provider, double amount): boolean (public): Processes a payment for gas bill from the specified account to the given provider for the specified amount.
- payOtherBill(Account account, String payee, double amount): boolean (public): Processes a payment for a bill to a payee other than electricity, education, water, or gas, from the specified account for the specified amount.

13) FundsTransferServices

Attributes:

No attribute

Methods:

- transferWithinBank(Account senderAccount, Account recipientAccount, double amount): boolean (public): Transfers funds from the sender's account to the recipient's account within the same bank.
- transferToExternalAccount(Account senderAccount, String recipientAccountNumber, String recipientBank, double amount): boolean (public): Transfers funds from the sender's account to an external account in another bank.

14) ChequeServices

Attributes:

No attribute

Methods:

- issueCheque(Account payerAccount, String payeeName, double amount): boolean (public): Issues a cheque from the payer's account to the specified payee for the specified amount.
- depositCheque(Account payeeAccount, Cheque cheque): boolean (public): Deposits a cheque into the payee's account.

15)UtilityServices

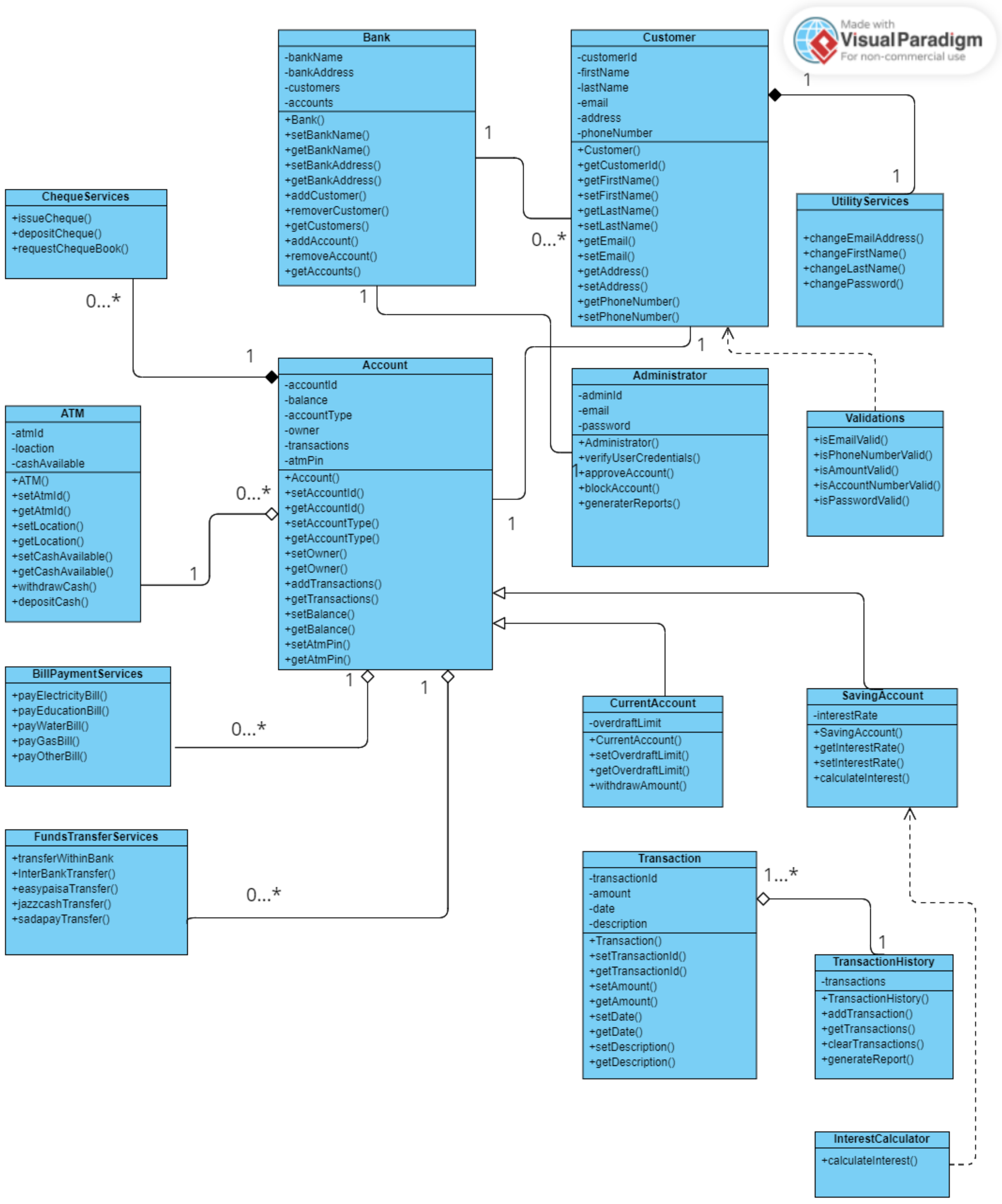
Attributes:

No attribute

Methods:

- changeFirstName(Customer customer, String newFirstName): boolean (public): Changes the first name for the specified customer.
- changeLastName(Customer customer, String newLastName): boolean (public): Changes the last name for the specified customer.
- changePassword(Customer customer, String currentPassword, String newPassword): boolean (public): Changes the password for the specified customer.

C) Create a class diagram for the project.



D) Plan the development timeline and assign tasks to team members.

Development Timeline

Phase 1: Design and Planning

- **Duration**: Approximately 1-2 weeks
- **Tasks:**
    1. Requirements Gathering: Gathering detailed requirements from stakeholders including bank representatives, potential users, and administrators.
    2. System Design: Designing the overall system architecture, database schema, and user interface mockups based on the gathered requirements.
    3. Planning: Breaking down the project into smaller tasks, estimate resource requirements, and create a project plan outlining milestones and deadlines.
    4. Technology Selection: Evaluating and selecting appropriate technologies and frameworks for development based on project requirements and team expertise.
    5. Risk Assessment: Identifying potential risks and developing mitigation strategies to address them throughout the project lifecycle.
- **Deliverables:**
    - System architecture diagrams
    - Database schema
    - User interface mockups
    - Project plan with milestones and deadlines

Phase 2: Implementation

- **Duration:** Approximately 2-4 weeks
- **Tasks:**
    1. Backend Development: Developing the backend functionality including user authentication, account management, transaction processing, and integration with external services.

2. Frontend Development: Implement the user interface components based on the designed mockups, ensuring responsiveness and usability across devices.
3. Database Implementation: Set up and configuring the database system, implement data models, and establish database connections for storing and retrieving user and transaction data.
4. Security Implementation: Implementing security measures such as encryption, authentication mechanisms, and authorization rules to ensure the confidentiality and integrity of user data.

- **Deliverables:**
  - Functional backend system
  - User interface with frontend components
  - Implemented database system
  - Security features integrated into the system

Phase 3: User Interface and Integration

- **Duration**: Approximately 2-3 weeks
- **Tasks:**

  User Interface Refinement: Refining the user interface based on feedback from stakeholders and usability testing, making adjustments to improve user experience and accessibility.
- **Deliverables:**
  - Refined user interface with improved usability
  - Accessibility features implemented

Phase 4: Testing and Final Submission

- **Duration**: Approximately 1-2 weeks
- **Tasks:**
  1. Functional Testing: Conducting thorough testing of the entire system to identify and resolve any bugs, errors, or inconsistencies in functionality.
  2. Security Testing: Performing security assessments and penetration testing to identify vulnerabilities and ensure that the system is resilient against potential threats.

3. Performance Testing: Evaluating the performance of the system under various load conditions to ensure scalability, responsiveness, and reliability.
4. User Acceptance Testing (UAT): Inviting stakeholders and end-users to participate in UAT sessions to validate that the system meets their requirements and expectations.
5. Documentation and Finalization: Preparing comprehensive documentation including user manuals, technical specifications, and system documentation for final submission.

- **Deliverables:**
  - Fully tested and validated system
  - Documentation package
  - Finalized system ready for deployment

Overall Timeline:

- **Total Duration**: Approximately 10(min)-12(max) weeks | 1.5-2 months
- **Milestones:**
  - Design and Planning: Weeks 1-2
  - Implementation: Weeks 2-4
  - User Interface and Integration: Weeks 2-3
  - Testing and Final Submission: Weeks 2-3

Tasks Assignment to Team Members

1. Mateen Haider(FA22-BSE-209):

- ❖ **Implementing Business Logic:** Aamir can work on implementing the business logic and functionality of the online banking system. This includes coding the classes and methods for account management, transaction processing, and other core functionalities.
- ❖ **Handling User Authentication:** Aamir can also be responsible for implementing user authentication features, including login/logout functionality and user credential validation.

2. Aamir Mubeen (FA23-BSE-032):

- ❖ **Designing UI Screens**: Mateen can focus on designing the user interface screens for the online banking system. This includes creating visual designs for login screens, account dashboard, transaction history, and other user interaction points.
- ❖ **Implementing UI Components**: Mateen can then implement the UI components based on the designed screens using JavaFX or another suitable UI framework.

3. Shoaib Shahid (FA23-BSE-137):

- ❖ **Integrating UI with Business Logic:** Shoaib can focus on integrating the UI components with the backend business logic. This involves connecting the user interface elements to the corresponding methods and functionalities implemented by Aamir.
- ❖ **Testing and Debugging:** Shoaib can also take responsibility for testing the application, identifying and fixing any bugs or issues, and ensuring that the system works smoothly and as expected.