Software Engineering Final

ABC Bank Application

Abstract:

The banking app project aims to provide a user-friendly platform for managing banking activities such as account balance inquiries, fund transfers, and transaction history review. The motivation behind this project is to offer customers convenient access to their banking services from the internet. Through this app, users can use financial transactions.

Software Requirements Specification (SRS)

1. Introduction

The banking app is a software solution designed to provide users with convenient access to various banking services, including account management, fund transfers, and transaction history review. This document outlines the functional and non-functional requirements of the banking app, detailing its features, constraints, and interfaces.

2. Scope

The banking app aims to cater to individual customers' banking needs, offering them a user-friendly platform accessible via web and mobile devices. The scope of the app includes:

- User registration and authentication
- Account management (checking, savings, investment)
- Fund transfers between different account types
- Transaction history review

- Security measures to safeguard user data and transactions

3. Functional Requirements

3.1 User Registration and Authentication

- The app shall allow users to register with their personal information, including name, address, email, and phone number.
- Users shall be required to create a unique username and password for authentication.
- Passwords shall be securely stored using encryption techniques.
- Upon registration, users shall receive a verification email or SMS to activate their account.

3.2 Account Management

- Users shall be able to view their account details, including balance, account number, and transaction history.
- The app shall support different types of accounts, including checking, savings, and investment.
- Users shall have the option to deposit funds into their accounts and withdraw funds as needed.
- Account balances shall be updated in real-time to reflect transactions.

3.3 Fund Transfers

- Users shall be able to transfer funds between their own accounts (e.g., from checking to savings).

- Transfer transactions shall be recorded and reflected in both the sender's and recipient's transaction history.

3.4 Transaction History

- Users shall have access to a comprehensive transaction history, showing details of all transactions conducted.
- Transaction history shall include information such as transaction type, amount, date, and involved accounts.

4. Non-Functional Requirements

4.1 Performance

- The app shall provide fast response times for user interactions, ensuring smooth navigation and transaction processing.
- System downtime shall be minimized through regular maintenance and monitoring to ensure high availability.

4.2 Security

- User data shall be encrypted during transmission and storage to prevent unauthorized access.
- Authentication mechanisms shall be implemented to verify the identity of users and protect against unauthorized account access.
- The app shall comply with industry-standard security protocols and regulations to ensure data privacy and integrity.

5. Interfaces

5.1 User Interface

- The app shall feature an intuitive user interface with clear navigation and user-friendly controls.
- Responsive design shall be implemented to ensure compatibility with various devices and screen sizes.

5.2 External Interfaces

- The app shall integrate with external banking systems for fund transfers and account verification purposes using MySQL and php

6. Constraints

- The app shall be developed using technologies that are supported across different platforms and devices.
- Compliance with banking regulations and standards shall be ensured to maintain legality and security.

Software Design Description (SDD)

1. Introduction

The Software Design Description (SDD) provides a detailed overview of the architectural design and system components of the banking app. This document outlines the high-level design decisions, architectural patterns, and data flow within the system.

2. Architectural Design

The banking app follows a multi-tier architecture, consisting of the following layers:

2.1 Presentation Layer

- Responsible for rendering the user interface and handling user interactions.
- Utilizes HTML and PHP for web-based interfaces, and native UI frameworks for mobile applications.

2.2 Application Layer

- Implements the business logic and application functionality.
- Handles user authentication, account management, and transaction processing.
- Developed using server-side scripting languages such as PHP for backend processing.

2.3 Data Access Layer

- Facilitates interaction with the database and persistence of data.
- Utilizes SQL queries to retrieve, update, and manipulate data stored in the database.
- Implements database access using MySQL or a similar relational database management system.

3. Component Design

The banking app comprises several key components, each serving specific functions:

3.1 User Authentication Component

- Responsible for authenticating users during login and registration processes.
- Utilizes encryption techniques to secure user credentials and prevent unauthorized access.

3.2 Account Management Component

- Handles account creation, retrieval, and modification operations.
- Implements CRUD (Create, Read, Update, Delete) operations for user accounts and associated data.

3.3 Transaction Processing Component

- Manages fund transfers, deposits, and withdrawals between user accounts.
- Validates transaction requests, checks account balances, and updates transaction history records.

3.4 User Interface Component

- Generates dynamic user interfaces based on user interactions and system responses.
- Ensures responsiveness and accessibility across different devices and screen sizes.

4. Data Flow

The following data flow diagram illustrates the flow of data within the banking app:

- User interactions trigger requests to the application layer, where business logic is executed.

- Application layer components communicate with the data access layer to retrieve or update data stored in the database.
- Data retrieved from the database is processed and presented to the user via the presentation layer.

5. Rationale for Design Decisions

- The multi-tier architecture was chosen for its scalability, maintainability, and separation of concerns.
- Server-side scripting languages like PHP were selected for their wide support and compatibility with web servers and databases.
- SQL databases were chosen for their relational structure, ACID compliance, and support for transaction management.

6. Deployment Considerations

- The banking app can be deployed on a web server with support for PHP and MySQL.
- Web hosting providers offering LAMP (Linux, Apache, MySQL, PHP) or similar stacks are suitable for deployment.
- Secure deployment practices, including HTTPS encryption and server hardening, should be followed to protect user data.

7. Conclusion

The Software Design Description (SDD) provides a comprehensive overview of the architectural design and system components of the banking app. By detailing the design decisions, component functionalities, and data flow, this document serves as a guide for implementation, testing, and deployment phases of the project.

Software Test Documentation (STD)

1. Introduction

The Software Test Documentation (STD) outlines the plan and specifications for verifying and validating the banking app. This document describes the testing approach, test cases, and expected results to ensure the functionality, reliability, and security of the application.

2. Test Plan

The test plan outlines the overall testing approach, including the scope, objectives, and methodologies to be employed during the testing process.

2.1 Scope

- The testing scope encompasses all functional and non-functional aspects of the banking app, including user authentication, account management, transaction processing, and data security.
- Testing will cover both positive and negative scenarios to validate the application's behavior under different conditions.

2.2 Objectives

- Ensure the correctness and completeness of application functionalities.
- Verify the reliability and stability of the application under normal and stress conditions.
- Validate the security measures implemented to protect user data and prevent unauthorized access.
- Identify and address any defects or issues that may arise during testing.

2.3 Methodologies

- Black-box testing: Focuses on testing the application's external behavior without knowledge of its internal structure.
- White-box testing: Examines the internal logic and code structure of the application to ensure thorough coverage of all paths and conditions.
- Integration testing: Validates the interactions between different components and modules of the application.
- User acceptance testing (UAT): Involves real users testing the application to ensure it meets their requirements and expectations.

3. Test Cases

The test cases outline specific scenarios and conditions to be tested, along with the expected results and pass/fail criteria.

3.1 User Authentication

- Test Case 1: Valid Login Credentials
    - Description: Attempt to log in with valid username and password.
    - Expected Result: User should be successfully authenticated and directed to the dashboard.
    - Pass/Fail Criteria: Login is successful without errors.

3.2 Account Management

- Test Case 2: Create New Checking, Savings and Investment Account
    - Description: Attempt to create a new user account with valid details by clicking link on new user

- Expected Result: New account should be created successfully and reflected in the database.
- Pass/Fail Criteria: Account creation process completes without errors.

3.3 Transaction Processing

- Test Case 3: Transfer Funds
  - Description: Initiate a fund transfer between two user accounts.
  - Expected Result: Funds should be debited from the source account and credited to the target account.
  - Pass/Fail Criteria: Transfer completes successfully without discrepancies in balances.

4. Test Results

After executing the test cases, the following results were observed:

- Test Case 1: Passed (Login successful)
- Test Case 2: Passed (New accounts created)
- Test Case 3: Passed (Funds transferred successfully)

5. Conclusion

The Software Test Documentation (STD) provides a structured approach to testing the banking app, ensuring that all functionalities are thoroughly evaluated and validated. By following the outlined test plan and executing the test cases, the application's quality, reliability, and security can be assured, leading to a successful deployment for end users.

Files Created/Modified

1. **login.html** - added functionality to open up specific users bank dashboard when logging in
2. **authenticate_login.php** - redirect to success page if user is in login_tbl and password correct
3. **dashboard.php -** handles html and php interactions for bank home dashboard
4. **checking.php -** handles html and php interactions of the checking account
5. **savings.php -** handles html and php interactions of the savings account including interest rate calculations
6. **investment.php -** handles html and php interaction of the investment account including interest rate calculations
7. **deposit.php -** handles deposit for each account type
8. **withdraw.php -** handles withdrawal for each account type
9. **transfer.php -** handles transfers from one account to another