**Software Engineering Lab Experiment No. 6**

**Aim:** To prepare a software test plan for the application.

**Objective:** The objective of this experiment is to create a test plan for the application to ensure its functionality, security, and performance meet the specified requirements.

**Requirements:**

1. Computer with internet access.
2. Sample software project or problem statement for requirements analysis.
3. Word processing software for creating the lab report.

**Concept:**

**Software test plan:**

A software test plan is a detailed document outlining the objectives, scope, resources, schedules, and approach to testing a software application. It describes the testing strategy, including test objectives, methodologies, test environments, entry and exit criteria, responsibilities, and risk assessment. This plan serves as a roadmap for the testing process, ensuring systematic and comprehensive testing of the software to identify and resolve any issues before its release.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Software Test Plan- Bank Application (Account Transactions)**  **Introduction:**  A Spring Boot bank application is a software solution built using the Spring Boot framework for developing Java-based enterprise applications. The application typically provides a set of features and functionalities related to banking operations, account management, transactions, and loans.  **Objective:**  The primary goal of the testing process is to identify, document, and address defects within the account transaction functionality. This encompasses a thorough examination of account creation, fund transfers, transaction history, and any other related processes. By validating and rectifying potential issues, the aim is to enhance user satisfaction and confidence in the account transaction process.  **Risk and Risk mitigation**   |  |  | | --- | --- | | **Risk** | **Mitigation** | | Inadequate Error Handling and Recovery | Implement robust error-handling mechanisms and conduct systematic testing for error scenarios. Use JUnit testing. | | Incomplete Loan Processing: | Conduct thorough functional testing with tools like Selenium | | Compatibility Issues with Various Browsers and Devices: | Implement cross-browser and cross-device testing using tools like Selenium or Cypress to ensure the Bank application’s compatibility and functionality across various environments. | | Inadequate Handling of Concurrent User Loads | Perform load testing using tools like JMeter or LoadRunner to simulate heavy user traffic and identify performance bottlenecks. |   **Types of Testing:**  **Unit Testing:**  This phase involves breaking down the account transaction functionality into its individual units, thoroughly assessing their functionality in isolation. It includes tests for account creation, fund transfers, transaction history, and balance verification to ensure each unit works as intended.  **Black Box - Functional Testing:**  The primary focus here is on the external behavior of the account transaction section without considering the internal code structure. It encompasses the testing of user interactions, including creating accounts, transferring funds, and checking transaction history.  **Load Testing:**  This testing type evaluates the account transaction's performance under varying user loads. Simulating different levels of concurrent users, it ensures stability, scalability, and the ability to manage peak traffic without performance degradation or system crashes.  **Scope:**  Testing will focus on the core account transaction functionality, including its critical elements like creating accounts, fund transfers, and transaction history.  Out of Scope:  Testing will not extend to other features or areas of the application not directly related to account transactions, such as loan processing, customer information, admin login, ATM, or unrelated backend systems.  **Tools:**  Unit testing  **JUnit** is a widely used testing framework for Java applications, including Spring Boot projects. It provides annotations to identify test methods, asserts for testing expected results, and other features to facilitate the testing process.  Black-box testing  Selenium: Widely used for automated testing of web applications, Selenium is versatile and supports multiple programming languages. It enables the testing of web applications across various browsers and platforms.  Load Balancing  JMeter: While primarily used for load testing, JMeter can also perform functional and black-box testing by simulating heavy loads on servers, networks, or objects to assess their performance under varied conditions.  **Schedule:**  Unit Testing: Week 1 – Rigorous testing of individual account transaction components.  Functional Testing: Weeks 2-3 – Detailed examination of the account transaction functionality and user experience.  Load Testing: Week 4 – Simulating various load conditions to validate the account transaction application’s performance and stability. |

**Conclusion:** We have created a software plan for the intended application