Guru99 Banking Project

Test Plan
V1.0

Prepared for
Guru99.com

1. Introduction:

- The manual testing project aims to ensure the quality, functionality and reliability of the banking web application, which provides essential financial services.
- The project includes functional, integration, system, and regression testing to cover all aspects of the application's behavior.
- Jira will be utilized for comprehensive bug reporting and tracking, while Excel will serve as the central repository for detailed test cases and version control.

2. Test Items

The 10 critical modules of the banking web application:
- New Customer
- New Account
- Mini-Statement
- Edit Customer
- Edit Account
- Delete Customer
- Delete Account
- Customized Statement
- Balance Enquiry
- Login / Logout

3. Features to be tested

- Each module will undergo extensive testing to verify its functionality, end-to-end capabilities and integration with other modules.

4. Features not to be tested

- External services such as API and Non-functional aspects will not be covered in this testing phase.

5. Test Environment

- *Operating System: macOS 14.0*
- *Web Browser: Google Chrome 120.0.6099.129*
- *Network: Wi-fi*

6. Test Deliverables:

- Detailed Test Cases:
    - Each test case in Excel includes:
        - *Test case ID*
        - *Test Scenario*
        - *Test Cases*
        - *Test Steps*
        - *Test Data*
        - *Expected Result*
        - *Actual Result*
        - *Pass / Fail*
- Jira Reports:
    - Regularly updated reports on the status of identified bugs, including priority, severity, and resolution progress.

7. Testing Schedule:

- Functional Testing:
    - [5.07.2023]: In-depth testing of individual modules for expected behavior.
    - [8.07.2023]: Ensures each module meets specified requirements.
- Integration Testing:
    - [9.07.2023]: Validates interactions between modules.

    - [12.07.2023]: Verifies seamless collaboration between different features.
- System Testing:
    - [14.07.2023]: Tests end-to-end functionality.
    - [17.07.2023]: Verifies the application as a whole.
- Regression Testing:
    - Ongoing throughout the testing phases to ensure new developments do not adversely affect existing functionality.

8. Testing Resources:

- Test Manager:
  - Oversees the entire testing process.
  - Ensures adherence to the test plan

- Lead Tester:
  - Coordinates testing efforts.
  - Reviews and approves test cases.

- Functional Testers:
  - Responsible for testing individual modules.

- Regression Testers:
  - Focuses on identifying any regression issues.

9. Test Case Design:

- *Functional Test Cases:* Covering various scenarios such as valid and invalid inputs, edge cases, and boundary conditions.

- *System Test Cases:* Ensure that the entire software system, including all its components and modules, functions correctly as a unified whole.

- *Integration Test Cases:* Verifying interactions between modules.

- *Regression Test Cases:* Ensuring existing functionalities are not compromised by new developments.

10. Test Execution:

- *Functional Testing:*
  - Test cases executed as per the test schedule.

- *Integration Testing:*
  - Focus on validating the seamless integration of modules.

- *System Testing:*
  - Comprehensive testing to verify end-to-end functionality.

- *Regression Testing:*
  - Continuous testing throughout the project to catch any unintended side effects.

11. Defect Tracking:

- Process for Reporting:
  - All defects reported promptly in Jira.

- Priority and Severity:
  - Assigned based on the impact on functionality and business processes.

12. Testing Risks:

- **Scope Creep:**
  Risk: Uncontrolled changes or additions to the project scope, affecting testing timelines and resources.

Mitigation: Regularly review and freeze project requirements. Clearly communicate the impact of scope changes on testing.

- **Limited Test Coverage:**
  Risk: Inadequate coverage of critical functionalities or scenarios in testing.
  Mitigation: Conduct comprehensive test case reviews to identify gaps in coverage. Prioritize testing based on critical business functionalities.

- **Environmental Issues:**
  Risk: Unavailability or instability of the test environment, hindering testing progress.
  Mitigation: Regularly check and maintain the test environment. Establish contingency plans for unexpected environmental issues.

13. Test Sign-Off:

- **Bug identification & reporting:**
  - No critical bugs outstanding.
  - All high-priority bugs resolved.

- **Approval Process:**
  - Involves the test manager and relevant stakeholders.

- **Documentation Verification:**
  - All project documentation, including user manuals and technical documentation, is reviewed and verified for accuracy and completeness.

- **End-to-End Business Processes:**
  - Verification that end-to-end business processes, from initiation to completion, are tested and meet business

requirements.

- **Execution Schedule:**
    - A detailed schedule outlining when each test case will be executed, considering dependencies and critical paths.

- **Real-Time Reporting:**
    - *Monitoring:* Continuous real-time reporting of test execution progress, providing visibility to the test manager and stakeholders.

14. Conclusion:

- **Summary:**
    - The testing process guarantees the application's functionality, integration and end-to-end capability in order to meet production standards.
    - Comprehensive testing validates that all critical features and functionalities align with the project's objectives.

- **Lessons Learned:**
    - Improved test case design.
    - Recognized the importance of a stable and controlled test environment for accurate testing results.
    - Developed more efficient test case review processes to identify and address issues promptly.

- **Recommendations:**
    - Suggests incorporating more extensive automated testing for repetitive tasks in future projects, especially for regression tests.
    - Advocate for automated test data generation tools to enhance efficiency and accuracy in data setup for testing.