**Software Test Document**

FinWise Manager Application

Group: 27

Name (s): Albert Mitchell and Alex Nikirk

Date: 04/21/2024

[1. Introduction 5](#_Toc2001345933)

[1.1 Purpose 5](#_Toc15571033)

[1.2 Scope 5](#_Toc919591296)

[2. Test Plan 5](#_Toc945162705)

[2.1 Included Features 6](#_Toc434613759)

[2.1.1 Account Registration 6](#_Toc1521501866)

[2.1.2 Account Login & Logout 6](#_Toc1077279581)

[2.1.3 Account Password Recovery 6](#_Toc297107749)

[2.1.4 Account User Settings 6](#_Toc1140704012)

[2.1.5 Transaction Management 6](#_Toc1142492516)

[2.1.6 Categorizing Transactions 6](#_Toc554083025)

[2.1.8 Transaction Search and Filter 6](#_Toc769206740)

[2.1.9 Budget Tracking 6](#_Toc413970091)

[2.1.10 Create and Edit Budgets 6](#_Toc161330478)

[2.1.11 Recurring Transactions 7](#_Toc407713599)

[2.1.12 Saving Goals 7](#_Toc2134037562)

[2.1.13 Reports & Insights 7](#_Toc1716099738)

[2.1.14 Security Features 7](#_Toc1742283054)

[2.1.15 Notification System 7](#_Toc1070500327)

[2.1.16 Cross-Platform Compatibility 7](#_Toc571989319)

[2.1.17 Data Synchronization 7](#_Toc448969540)

[2.2 Features not to be Tested 8](#_Toc296158365)

[2.2.1 Advanced Investment Tracking 8](#_Toc44317039)

[2.2.2 Third-Party Integration 8](#_Toc1839631940)

[2.2.3 Multiregional Support 8](#_Toc418125866)

[2.3 Testing Tools and Environment 8](#_Toc1865073071)

[3. Approach 9](#_Toc1807317007)

[3.1 Test Strategy 9](#_Toc491676770)

[3.1.1 Unit Testing 9](#_Toc77531566)

[3.1.2 Integration Testing 9](#_Toc1811305450)

[3.1.3 System Testing 9](#_Toc323667672)

[3.1.4 Cross-Browser and Cross-Device Testing 9](#_Toc332535554)

[3.1.5 User Acceptance Testing (UAT) 10](#_Toc1357157180)

[3.1.6 Accessibility Testing 10](#_Toc1701932119)

[3.1.7 Performance Testing 10](#_Toc1810284562)

[3.1.8 Security Testing 10](#_Toc451824843)

[4. Item Pass/Fail Criteria 10](#_Toc451012712)

[4.1 Criteria 10](#_Toc1660876951)

[4.1.1 Functionality 10](#_Toc775550682)

[4.1.2 Critical Bugs 10](#_Toc1629540125)

[4.1.3 Performance 11](#_Toc213934622)

[4.1.4 Responsiveness 11](#_Toc188249402)

[4.1.5 UI Elements 11](#_Toc1529263573)

[4.1.6 Error Handling 11](#_Toc35530651)

[5. Suspension Criteria and Resumption Requirements 11](#_Toc568231128)

[5.1 Suspension Criteria 11](#_Toc716670535)

[5.1.1 Critical Bugs 11](#_Toc19922416)

[5.1.2 Security Breaches 11](#_Toc1869128288)

[5.1.3 System Downtime 12](#_Toc743516989)

[5.2 Resumption Requirements 12](#_Toc380445059)

[5.2.1 Bug Resolution 12](#_Toc1660808655)

[5.2.2 Security Fixes 12](#_Toc1469317404)

[5.2.3 Infrastructure Availability 12](#_Toc522844321)

[6. Test Deliverables 12](#_Toc382704726)

[6.1 Documents 12](#_Toc786473026)

[6.1.1 Test Cases 12](#_Toc801768203)

[6.1.2 Test Scripts 13](#_Toc814711687)

[6.1.3 Bug Reports 13](#_Toc131592741)

[6.1.4 Test Summary Reports 13](#_Toc921480981)

[7. Testing Tasks 13](#_Toc1245468073)

[7.1 Task Identification 13](#_Toc267359548)

[7.1.1 Test Case Preparation 13](#_Toc703723140)

[7.1.2 Test Environment Setup 13](#_Toc74565583)

[7.1.3 Test Execution 14](#_Toc323815619)

[7.1.4 Regression Testing 14](#_Toc1806439680)

[7.1.5 Reporting 14](#_Toc1916863655)

[8. Environmental Needs 14](#_Toc103205656)

[8.1 Test Environment 14](#_Toc598402677)

[8.1.1 Mobile Devices 14](#_Toc856911955)

[8.1.2 Browser Compatibility 14](#_Toc1647885279)

[8.1.3 Network Connection 14](#_Toc774335470)

[8.1.4 Testing Server 15](#_Toc380284782)

[8.1.5 Testing Database Environment 15](#_Toc2108535094)

[8.1.6 Monitoring and Logging Tools 15](#_Toc1177838369)

[9. Responsibilities 15](#_Toc543516214)

[9.1 Roles and Responsibilities 15](#_Toc681833421)

[9.1.1 QA Leads 15](#_Toc620216823)

[9.1.2 Testers 15](#_Toc1803521928)

[9.1.3 Developers 15](#_Toc270061141)

[9.1.4 Product Managers 16](#_Toc588250872)

[10. Staffing and Training Needs 16](#_Toc968954871)

[10.1 Staff Requirements 16](#_Toc1707262789)

[10.1.1 Training on New Features 16](#_Toc1241760925)

[10.1.2 Training on Testing Tools 16](#_Toc2055501271)

[11. Schedule 16](#_Toc88474757)

[11.1 Timeline 17](#_Toc691511741)

[11.1.1 Week 1 17](#_Toc1798874330)

[11.1.2 Weeks 2-4 17](#_Toc1810197676)

[11.1.3 Week 5 17](#_Toc1132263973)

[12. Risks and Contingencies 17](#_Toc239383419)

[12.1 Risks 17](#_Toc1382767554)

[12.1.1 Unforeseen Bugs 17](#_Toc202213094)

[12.1.2 Environment Setup Delays 17](#_Toc458262595)

[12.2 Contingency Plans 17](#_Toc306776499)

[12.2.1 Buffer Time 17](#_Toc212773129)

[12.2.2 Pre-Setup of Environments 18](#_Toc1686749752)

[13. Approvals 18](#_Toc996974699)

[13.1 Sign-off 18](#_Toc1170481154)

[13.1.1 QA Manager 18](#_Toc150895705)

[13.1.2 Project Manager 18](#_Toc213814929)

[13.1.3 Product Owner 18](#_Toc802440100)

# 1. Introduction

The Introduction section of the Test Plan sets the stage for the entire testing process, outlining the objectives and how the document will guide the testing efforts.

## 1.1 Purpose

The purpose of this Test Plan is to provide a detailed framework for the testing phase of the FinWise Manager application. It defines the strategies and activities required to ensure that the application meets its design specifications and user requirements. The Test Plan will serve as a guide for the QA team to validate the functionality, reliability, performance, and security of the application, with an emphasis on user satisfaction and adherence to regulatory standards.

## 1.2 Scope

This document outlines the comprehensive testing strategy for FinWise Manager, delineating the scope of testing, identifying the features and components to be tested, establishing pass/fail criteria, and specifying the resources and schedules involved. It also identifies testing deliverables, roles and responsibilities, environmental needs, and the risks and contingencies associated with the test phase. This Test Plan aligns closely with the Software Requirements Specification (SRS) and Software Design Document (SDD) to ensure a coherent and efficient testing process.

# 2. Test Plan

This section outlines the specific components of the FinWise Manager application that are subject to testing. Each feature is integral to the application's functionality and the user's experience. The functionality, stability, and responsiveness of each feature will be subjected to a series of tests to ensure they operate correctly under various conditions.

## 2.1 Included Features

### 2.1.1 Account Registration

A user can successfully create an account with Finwise Manager.

### 2.1.2 Account Login & Logout

A user can login to their account with correct credentials and they can successfully log out of their Finwise Manager account.

### 2.1.3 Account Password Recovery

A user can reset or recover their Finwise Manager account password.

### 2.1.4 Account User Settings

A user can personalize their account experience by adjusting settings such as email preferences, privacy controls, language and region settings, accessibility options, and profile information.

### 2.1.5 Transaction Management

This involves the core functionality of managing transactions. The user can add, edit, and delete transactions.

### 2.1.6 Categorizing Transactions

The user can organize transactions into different categories to better understand their spending habits.

### 2.1.8 Transaction Search and Filter

Allow users to search for specific transactions and filter transactions based on criteria such as date, amount, category, or description, enhancing accessibility and ease of use.

### 2.1.9 Budget Tracking

The user can track their spending against budgets they set.

### 2.1.10 Create and Edit Budgets

The user can create and modify their budget.

### 2.1.11 Recurring Transactions

Enable users to set up recurring transactions for regular expenses or income, automating repetitive tasks and ensuring accurate tracking of financial activities.

### 2.1.12 Saving Goals

The user can create and monitor their progress towards specific financial goals.

### 2.1.13 Reports & Insights

This will cover the generation of financial reports, insights, and summaries that help users understand their spending behavior and financial health.

### 2.1.14 Security Features

Ensures that user data encryption, secure login processes, and any additional security measures like two-factor authentication function correctly.

### 2.1.15 Notification System

Allows users to receive in-app notifications, texts, and email reminders for bills, budget limits, and saving goals.

### 2.1.16 Cross-Platform Compatibility

The app can function correctly across different devices and operating systems, maintaining design consistency and user experience.

### 2.1.17 Data Synchronization

To ensure seamless user experience, data synchronization automatically backs up and connects user information to the servers associated with each account. This process guarantees consistency of information across all devices, enabling users to access their data effortlessly regardless of the device they're using.

## 2.2 Features not to be Tested

This section delineates the features that are outside the scope of the current testing phase, usually due to their development status or strategic decisions to postpone testing to a later stage.

### 2.2.1 Advanced Investment Tracking

As this feature is still in the early stages of development, it will not be tested in the current cycle. However, in the future the app will include real-time market data integration, personalized investment recommendations based on user goals and risk tolerance, portfolio diversification analysis, performance benchmarking against market indices, and interactive visualization tools for better understanding of investment trends and performance.

### 2.2.2 Third-Party Integration

Any integrations with third-party services or APIs, such as bank account syncing, that are not yet fully implemented or are awaiting partnerships, will be excluded from this round of testing.

### 2.2.3 Multiregional Support

For this round of testing, language translations and region-specific features will be excluded. This ensures that the current testing phase focuses on fully functional aspects, while future multiregional enhancements are evaluated in subsequent testing cycles.

## 2.3 Testing Tools and Environment

To ensure the stability and quality of FinWise Manager throughout its lifecycle, industry standard testing products will be combined with a comprehensive approach utilizing a variety of specialized testing tools and environments tailored to different aspects of the application. For backend logic written in Node.js, Jest or Mocha coupled with Supertest will efficiently handle unit and integration testing, while React will be employed for frontend testing to ensure compatibility across various devices and browsers. Additionally, Postman or Insomnia will aid in API testing, and tools like Apache JMeter or Gatling will assess performance under load. Establishing testing environments that mimic production, including staging and sandbox environments, is essential, alongside implementing continuous integration and deployment (CI/CD) pipelines through platforms like Jenkins or GitLab CI to automate testing and deployment processes. By leveraging this diverse set of testing tools and environments, FinWise Manager can achieve comprehensive testing coverage, enhancing its stability and quality.

3. Approach

The approach section outlines the testing methodologies that will be utilized to evaluate the features of the FinWise Manager application. It defines how the testing will be conducted to ensure comprehensive coverage.

## 3.1 Test Strategy

The test strategy for FinWise Manager is a balanced blend of manual and automated testing procedures designed to cover all aspects of the application. Testing will occur in iterative cycles, allowing for feedback and improvements at each stage of development.

### 3.1.1 Unit Testing

Developers will write automated unit tests for individual components to validate that each piece of code performs as expected in isolation.

### 3.1.2 Integration Testing

Integration testing will assess how well different components work together.

### 3.1.3 System Testing

The entire application will undergo system testing to validate complete and integrated software functionality.

### 3.1.4 Cross-Browser and Cross-Device Testing

Cross-browser and cross-device testing to ensure consistent functionality and appearance across different browsers and devices, including mobile phones and tablets.

### 3.1.5 User Acceptance Testing (UAT)

UAT will be conducted with actual users to ensure the app meets their expectations and real-world usage conditions.

### 3.1.6 Accessibility Testing

Accessibility testing will be done to ensure that the application is usable by individuals with disabilities and complies with accessibility standards such as WCAG (Web Content Accessibility Guidelines).

### 3.1.7 Performance Testing

Automated tests will simulate various user loads to ensure the application maintains performance benchmarks.

### 3.1.8 Security Testing

Specialized tests will examine the app for vulnerabilities and ensure that user data remains secure against potential threats.

# 4. Item Pass/Fail Criteria

## 4.1 Criteria

Tests that do not meet these criteria will be marked as failed and will require remediation followed by retesting to confirm that the issues have been addressed.

### 4.1.1 Functionality

Features must operate according to the specifications outlined in the SRS document. Any deviation will result in failure.

### 4.1.2 Critical Bugs

No critical bugs should be present. A critical bug is defined as one that causes system crashes or leads to significant data loss.

### 4.1.3 Performance

Application load times must be under 2 seconds for core functionalities on standard broadband connections.

### 4.1.4 Responsiveness

The app must display properly and remain responsive across different devices, including a range of smartphones and tablets.

### 4.1.5 UI Elements

All interactive elements should respond to user input without unexpected delays or errors.

### 4.1.6 Error Handling

The application must handle errors gracefully, providing informative feedback to the user without abrupt crashes or data loss.

# 5. Suspension Criteria and Resumption Requirements

This section specifies the conditions under which testing for FinWise Manager will be temporarily halted and the requirements that must be met to resume testing. It ensures that testing efforts are efficient and effective, pausing only when necessary to address significant issues that could impact the quality of the final product.

5.1 Suspension Criteria

### 5.1.1 Critical Bugs

If bugs are found that cause the application to crash, lose data, or severely impact core functionality, testing will be halted until these issues are addressed.

### 5.1.2 Security Breaches

Should any security vulnerabilities that pose significant risks be discovered, testing will pause to prioritize fixes.

### 5.1.3 System Downtime

Unexpected downtime in critical testing infrastructure or support systems that prevent testing from being carried out effectively.

## 5.2 Resumption Requirements

Testing will resume once the following conditions are met:

### 5.2.1 Bug Resolution

All critical bugs identified must be fixed, verified by the development team, and confirmed through a retest.

### 5.2.2 Security Fixes

Any security issues must be resolved and validated through security testing to ensure the threat has been neutralized.

### 5.2.3 Infrastructure Availability

The necessary testing infrastructure must be restored and fully operational.

# 6. Test Deliverables

This section outlines the expected outputs from the testing process. These deliverables are essential for documenting the testing phase and providing accountability and traceability for FinWise Manager.

## 6.1 Documents

These documents will serve as a basis for evaluating the application's quality and understanding areas that require improvement. They provide a historical record of testing efforts and outcomes, which is crucial for future development cycles and audits.

### 6.1.1 Test Cases

Detailed descriptions of test scenarios and conditions, including expected inputs and outputs.

### 6.1.2 Test Scripts

Automated scripts used for conducting tests, particularly for regression and performance testing.

### 6.1.3 Bug Reports

Documentation of all bugs discovered during testing, their severity, steps to reproduce, and the status of their resolution.

### 6.1.4 Test Summary Reports

Comprehensive overviews provided at the end of testing cycles, summarizing testing activities, outcomes, and any outstanding issues.

# 7. Testing Tasks

This section details the tasks required to ensure comprehensive testing of FinWise Manager, structured to cover the entire testing lifecycle.

## 7.1 Task Identification

### 7.1.1 Test Case Preparation

Developing detailed test cases based on the requirements specified in the SRS. This includes defining the conditions under which each test will be conducted, expected outcomes, and how each test relates to specific requirements.

### 7.1.2 Test Environment Setup

Configuring the necessary hardware, software, network settings, and data setups to replicate production environments as closely as possible. This task also involves ensuring all tools and resources are available and functional.

### 7.1.3 Test Execution

Conducting the tests as per the test cases. This includes running both manual and automated tests, documenting the results, and comparing them against expected outcomes.

### 7.1.4 Regression Testing

Following any code changes, regression testing will be performed to ensure that new code has not adversely affected existing functionality.

### 7.1.5 Reporting

Creating detailed reports that summarize the testing outcomes, including documentation of any defects found, their severity, and the impacts on the system. Reports also highlight successes and any areas for improvement.

# 8. Environmental Needs

Ensuring the testing environment accurately reflects the production environment or the anticipated user conditions is crucial for reliable testing.

## 8.1 Test Environment

### 8.1.1 Mobile Devices

A range of smartphones and tablets with different screen sizes and operating systems, including Android and iOS, to ensure the application functions well across all major platforms.

### 8.1.2 Browser Compatibility

Testing across various web browsers such as Google Chrome, Mozilla Firefox, Microsoft Edge, and Safari to ensure seamless functionality.

### 8.1.3 Network Connection

A stable and controllable network environment that allows simulation of different online conditions, such as low bandwidth, high latency, or disconnected states.

### 8.1.4 Testing Server

A dedicated server that mimics the live server environment where the application will be deployed, including similar processing power, memory, and configurations.

### 8.1.5 Testing Database Environment

Verify the functionality and configuration of the chosen database environment, using MySQL or MongoDB, to guarantee uniformity in data handling and storage.

### 8.1.6 Monitoring and Logging Tools

Utilize New Relic or ELK Stack for monitoring application performance metrics, tracking system behavior, and resolving potential issues.

# 9. Responsibilities

Clearly defined roles and responsibilities ensure that each team member understands their tasks and how they contribute to the testing process.

## 9.1 Roles and Responsibilities

### 9.1.1 QA Leads

Responsible for the overall planning, coordination, and oversight of the testing process. They define the testing standards and ensure adherence to best practices.

### 9.1.2 Testers

Execute the tests according to the plan. They are responsible for identifying issues, documenting their findings, and retesting once fixes have been applied.

### 9.1.3 Developers

Address and resolve any bugs or issues identified during testing. They work closely with testers to understand the problems and update the application as needed.

### 9.1.4 Product Managers

Provide final approval on the application's release. They ensure that the product meets all business and customer requirements and decide on go/no-go for launch.

# 10. Staffing and Training Needs

Ensuring that the team is adequately staffed and properly trained is crucial for the test plan's success.

## 10.1 Staff Requirements

The current Quality Assurance (QA) team is well-equipped to handle the testing requirements for FinWise Manager. However, as the application incorporates new features and possibly integrates new testing tools training will be necessary.

### 10.1.1 Training on New Features

All QA personnel will receive detailed training on the new features and functionalities of FinWise Manager to understand the expected behaviors and potential complexities in testing.

### 10.1.2 Training on Testing Tools

If new testing tools or updates to existing tools are to be used, comprehensive training sessions will be organized to ensure all team members are proficient in using these tools effectively.

# 11. Schedule

A well-defined schedule helps manage time effectively, ensuring that all testing activities are completed within the allocated timelines.

## 11.1 Timeline

### 

### 11.1.1 Week 1

Test Planning – Finalize and approve the test plan, complete the training on new features and tools.

### 11.1.2 Weeks 2-4

Test Execution – Conduct all planned tests, including unit, integration, system, and user acceptance tests.

### 11.1.3 Week 5

Reporting and Analysis – Compile test results, document findings, perform an initial bug fix cycle, and prepare a detailed test summary report.

# 12. Risks and Contingencies

Identifying potential risks and preparing contingency plans are essential to mitigate any negative impacts on the testing phase.

## 12.1 Risks

### 12.1.1 Unforeseen Bugs

Discovery of severe bugs that could significantly delay the testing schedule.

### 12.1.2 Environment Setup Delays

Potential delays in configuring or accessing required testing environments or tools.

## 12.2 Contingency Plans

### 12.2.1 Buffer Time

Include buffer time in the schedule to accommodate unexpected delays or extended bug fix periods.

### 12.2.2 Pre-Setup of Environments

Ensure that all testing environments are set up and fully operational before the commencement of the testing phase to avoid delays.

# 13. Approvals

The final approvals from key stakeholders are necessary to confirm that the test plan meets all requirements and is ready for implementation.

## 13.1 Sign-off

### 13.1.1 QA Manager

Confirms that the test plan is comprehensive, and the QA team is prepared to execute it.

### 13.1.2 Project Manager

Ensures that the test plan aligns with the overall project schedule and resources.

### 13.1.3 Product Owner

Verifies that the testing scope adequately covers all user and business requirements.

|  |  |  |  |
| --- | --- | --- | --- |
| Role | Name | Signature | Date |
| QA Manager |  |  |  |
| Project Manager |  |  |  |
| Product Owner |  |  |  |