Software Engineering project 2023/2024
Ca' Foscari University of Venice

Testing Plan
1.0

AgileMasters

01/11/2023

Document Informations

NomeProgetto	Acronimo
Deliverable	Testing plan
Delivery date	14/11/2023
Team Leader	Alberto Tomasin 892614@stud.unive.it
Team members	André Ramolivaz - 891923@stud.unive.it, Simone Dinato - 892539@stud.unive.it, Mirco De Zorzi - 891275@stud.unive.it

Document History

Version	Issue Date	Stage	Changes	Contributors (reviewer)
0.1	02 /11/2023	Draft	Finish chapter 1	892539, (891923)
0.2	03/11/2023	Draft	Finish chapter 2	891275, (891923)
0.3	07/11/2023	Draft	Finish chapter 3	891275, (892539)
0.4	09/11/2023	Draft	Finish chapter 4	891923, (892539)
0.5	10/11/2023	Draft	Finish chapter 5	892539, (891923)
0.6	10/11/2023	Draft	Finish chapter 6	891275, (892539)
0.7	10/11/2023	Draft	Finish chapter 7	891275, (891923)
1.0	13/11/2023	Final	Finish document	(891275, 891923, 892539)

Indice

1. Introduction

1.1. Document overview

The **Testing plan** in this university context defines the testing procedures that will accompany the development of system components and their integration. It is important to note that this document will be dynamic and continually evolving to adapt to the project's needs.

Testing is not intended to guarantee the absence of defects but rather to discover and identify potential malfunctions. Detecting and correcting errors in the early stages of the project is crucial as it prevents these errors from accumulating and becoming increasingly challenging to identify over time.

Furthermore, the **Testing plan** will be strictly based on the requirements analysis conducted previously, ensuring that all tests are aligned with the project's objectives and specifications. Our testing methodology will be guided by a thorough understanding of the requirements, enabling a precise evaluation of the system components.

1.2. Glossary

2. Testing techniques

In this section of the document, we will outline the **testing techniques** selected to ensure the app functions optimally. The rationale behind the choice of these methodologies will also be provided.

A multifaceted approach that incorporates various testing techniques will be adopted.

The predominant testing strategy will be **bottom-up**, which involves verifying each individual module of the app before proceeding to assemble them for broader system tests. This method requires sustained effort and may seem burdensome, but it is crucial for the early detection and resolution of issues that, if unaddressed, could jeopardize the overall development.

Furthermore, we will implement **comparison tests** with each release of a new app version, to ensure that it conforms to and does not regress from existing functionalities. Given that our team adheres to the principles of **Agile methodology**, the comparison testing technique efficiently integrates, allowing us to continuously assess product iterations and ensure performance consistency after each development cycle.

In the realm of system evolution, our approach moves away from the prototype model to fully embrace **Agile methodology**. This chapter describes how **FinanceFlow** will evolve through incremental releases, with the team adapting flexibly to changes rather than following a rigid sequence of prototypes. Our strategy is to develop, test, and release in short, manageable cycles, enabling continuous improvement based on user feedback and system performance analyses. This ensures that the app remains cutting-edge, both technologically and as an educational resource for our users.

3. Requirements traceability

The following section explains how the individual requirement will be verified. For the sake of pure illustrative simplicity they will be summarized in a table.

3.1. Functional requirements check

As far as functional requirements are concerned, they will be assessed by taking as reference the specification indicated in the requirements analysis. The example table is as follows:

Functionality	Identifies the functionality performed by the requirement
Reference	Reference to the RS-XX specification in the requirements analysis document
Test objective	Specifies what needs to be verified
Evaluation criteria	Decide whether the test was successful or not.
Priority	The priority

Below you can see the table containing all the necessary information:

Functionality	Reference	Test objective	Evaluation criteria	Priority
Account creation	RS-01	Ensuring the user intention to sign-up goes as planned	Write in the PostgreSQL database a record for the new user	High
Login	RS-02, RS-03	Ensure the user is registered inside the system	Search in the PostgreSQL database a record that matches the user	High
Browse, Search and Select stocks	RS-04	User can see, search and select from a list of available stocks	The system provides the correct stock based on the query	High
Stock graph	RS-05	User can see a graphical representation of the performance of a stock over time.	Generated the correct graph of the selected stock	High
Stock purchase	RS-06	User can purchase one or more stocks that they have selected.	The user is presented with a confirmation screen that shows the details of the purchase, such as the stock symbol, number of shares, and purchase price. The user has the opportunity to cancel the purchase before it is executed.	High
Stock sell	RS-07	User can purchase one or	The user is presented with	High

		more stocks that they own	a confirmation screen that shows the details of the sale, such as the stock symbol, number of shares, and sale price. The user has the opportunity to cancel the sale before it is executed.	
Balance	RS-08	Enable user to view the current balance of the account	The account balance is displayed correctly and completely.	High
Out of money	RS-24	Make sure the user doesn't make a purchase that makes its account balance negative	Before executing transaction check if the balance will be negative	High
Market information	RS-09	User can access information about market dynamics, such as stock market indices, sector performance, and economic indicators.	The information about market dynamics is accurate, up-to-date, and relevant to the user's investment needs.	High
Chatbot	RS-10	User can ask the chatbot a question and receive a response.	The chatbot's response is accurate, informative, and relevant to the user's question.	High
Profile update	RS-17, RS-18, RS-19	User can see and edit its profile	The correct profiled is pulled from PostgreSQL and the update query is successful	High

3.2. Non-Functional requirements check

As far as non-functional requirements are concerned, as there is a clear and precise metric, different validation techniques may be used with respect to functional requirements.

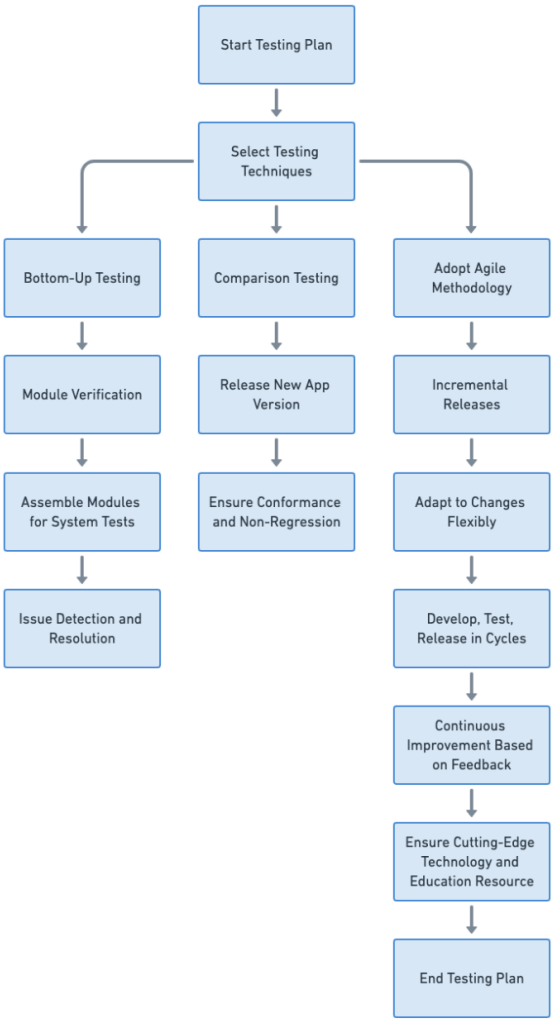
In particular, they will take as reference the measures defined in the requirements analysis. We remember that they are defined as follows:

Speed	Response and processing times, including the amount of data and types of resources that will be processed

Ease of use	The level of user-friendliness of the system through interaction with the user interface.
Robustness	La capacità del prodotto software di funzionare come previsto nonostante la presenza di errori.
Portability	Level of adaptability of the software in different environments.

If the test has a different result from the one reported, an update of the analysis document with the new measurement detected.

4. Testing schedule



The testing plan for our application is designed to be thorough and iterative, ensuring that each component functions optimally before being integrated into the larger system. The following is a descriptive overview of our testing schedule and techniques, as visualized in the accompanying flowchart:

- **Start Testing Plan**: We initiate our testing process with a clear plan that outlines the objectives and scope of our testing activities.
- **Select Testing Techniques**: A variety of testing techniques are chosen to address different aspects of the application's functionality and performance.
- **Bottom-Up Testing**: This is the primary strategy where we start by testing individual modules in isolation.
    - **Module Verification**: Each module is tested independently to ensure it meets the required specifications.
    - **Assemble Modules for System Tests**: After individual testing, modules are assembled to verify their interactions and overall system behavior.
    - **Issue Detection and Resolution**: Early detection of issues at the module level allows for timely resolutions, preventing potential system-wide problems.
- **Comparison Testing**: With each new release, we perform comparison tests to:
    - **Release New App Version**: Ensure that the new version is ready for deployment.
    - **Ensure Conformance and Non-Regression**: Confirm that the new version maintains all existing functionalities and does not introduce regressions.
- **Adopt Agile Methodology**: Our testing approach is aligned with Agile principles, which emphasize flexibility and continuous improvement.
    - **Incremental Releases**: The application evolves through frequent and incremental updates, allowing for rapid adaptation to feedback and changing requirements.
    - **Adapt to Changes Flexibly**: Our team remains responsive to change, ensuring that the application can pivot as needed without being constrained by a rigid development structure.
    - **Develop, Test, Release in Cycle**s: Short, manageable cycles of development, testing, and release facilitate ongoing refinement and enhancement of the application.
    - **Continuous Improvement Based on Feedback**: User feedback and system performance data are integral to our process, guiding the continuous improvement of the application.
- **Ensure Cutting-Edge Technology and Education Resource**: We are committed to maintaining the application at the forefront of technology and as a valuable educational resource for our users.
- **End Testing Plan**: The testing cycle concludes with a review of outcomes and preparation for the next iteration, ensuring that the application remains robust and reliable.

5. Test cases specifications

The Test Specification section outlines a detailed description of each test case for the FinanceFlow application. Each test case is presented in the form of a specification table, including the following fields:

- **Test Case ID**: A unique identifier to distinguish different tests.
- **Functionality**: Assertions that must be verified to execute the test.
- **Input Values**: Actions and test data necessary for testing.
- **Expected Outcomes**: The anticipated results for each input.

In addition to the initial fields, each test record is dynamically updated during the execution of the test to provide a comprehensive understanding:

- **Observed Outputs**: The actual outcomes observed during the test execution. While they are ideally aligned with the expected outcomes, differences are not necessarily indicative of errors.
- **Testing Platform**: The specific device or platform on which the test was conducted.

- **Outcome**: Indicates whether the test result was positive or negative.
- **Test Date**: The date on which the test was performed.

Important: the test cases presented are part of an ongoing and evolving testing process, and additional test cases will be defined and executed as the project progresses. The final outcomes and the list of all test cases will be available on the final version of the document.

ID	TC1
Functionality	User registration
Input	A user valid registration data
Expected Outcomes	Successful registration with a user account
Observed Outputs	
Testing Platform	
Outcome	
Test Date	

ID	TC2
Functionality	Registration confirmation
Input	Click on the Register button
Expected Outcomes	Verification email
Observed Outputs	
Testing Platform	
Outcome	
Test Date	

ID	TC3
Functionality	Login Process
Input	Users credentials
Expected Outcomes	Successful login into the application

Observed Outputs	
Testing Platform	
Outcome	
Test Date	

6. Requirements

The following chapter is designed to describe the minimum and recommended hardware and software requirements for running the **FinaceFlow** application.

6.1. Minimum

- **Operating system:** Android 5.0 (Lollipop)
- **RAM:** at least 1.5GB or RAM
- **Storage space:** at least 200MB of free space for app installation
- **CPU**: ARMv7 CPU

These requirements will let you run the application, but it might not behave as expected. We suggest referring to the suggested requirements for the best experience.

6.2. Suggested

- **Operating system**: Android 5.0 (Oreo) or higher
- **RAM**: 2GB or more
- **Storage space**: at least 250MB of free storage space
- **CPU**: ARMv8 CPU or newer for optimal performance

6.3. Development

Furthermore, in order to carry out the tests, it will be essential to have a thorough knowledge of everything related to:

- Flutter framework and Dart programming language
- SQL and NoSQL query languages
- REST APIs

7. Constraints

Below are the rules and limits to be observed in order to consider valid testing.

Testing, as indicated in the project plan, must be completed by 10/12/2023 if the project proceeds on schedule. The deadline for testing will be 10/01/2024, so have a few days to correct any bugs before the final delivery on 15/01/2024.

By this date, the system must be able to perform all the actions foreseen in the requirements document.

8. References

The following documents were taken into consideration for the preparation of the following paper:

- Analysis and specification documents related to previous years' projects.
- Course slides provided by the teacher.