TED UNIVERSITY

2023-2024 Spring Term

CMPE 492 Senior Project

Test Plan Report

FocuZone

Team Members:

Şevval Kaplan - 37463197404

Hüseyin Sina Ceylan - 10003284536

Şevval Yardımcı - 55780306880

Eren Sakarya - 12521841116

Table of Contents

1. Introduction 3
   1.1. Scope 3
      1.1.1. In Scope 3
      1.1.2. Out of Scope 4
   1.2. Quality Objective 4
   1.3. Roles and Responsibilities 5
2. Test Methodology 6
   2.1. Overview 6
   2.2. Testing Levels 6
      2.2.1. Unit Testing 6
      2.2.2. Integration Testing 7
      2.2.3. System Testing 7
      2.2.4. Acceptance Testing 7
   2.3. Bug Triage 7
   2.4. Suspension Criteria and Resumption Requirements 8
   2.5. Completion of the Test 8
3. Test Deliverables 8
   3.1. Test Plan 8
   3.2. Test Scenarios 9
      3.2.1. Scenario 1: Operability and Error-Freeness of Mobile App Side 9
      3.2.2. Scenario 2: Functionality of the Image Detection System Used in the Environmental Analysis Feature 10
   3.3. Test Reports 10
   3.4. Test Records 11
4. Recourse and Environment Needs 11
   4.1. Project Environment 11
      4.1.1. Database 11
      4.1.2. Mobile Devices Project 12
   4.2. Testing Tools 12
      4.2.1. Unit Testing 12
      4.2.2. Test Case Tool 12
   4.3. Test Environment 13
      4.3.1. Development Workstation 13
      4.3.2. Mobile Devices 13
5. References 14

1. Introduction

1.1. Scope

This section elaborates on the scope and limitations of our testing efforts for the project, providing detailed explanations. We will cover many different aspects to make sure every part of the project is working smoothly.

1.1.1. In Scope

Testing the Mobile Application Side: Testing of basic functionalities and features on the mobile application side.

- **Screen Transitions:** It will be ensured that the transitions of all screens of the application occur correctly and fluently. This feature is one of the important factors affecting user experience. Therefore, it is one of the focal points of our tests.
- **Firebase Integration:** Since some parts of the application are integrated with the Firebase system, when data is saved and retrieved on Firebase, it will be checked whether these transactions are correct and error-free.
- **Data Accuracy:** It will be checked that the data retrieved from and saved to the Firebase Real-Time Database is displayed correctly within the application. This is critical to the core functionality of the application.
- **Visual and Animation Checks:** It will be tested whether the visual elements and animations within the application work properly. This will ensure that the user interface is displayed as desired.

Testing the Environmental Analysis Feature: Testing whether the feature in the application, called the environmental analysis feature, works correctly.

- **Image Recognition:** It should be tested whether the feature can correctly recognize certain images. Since it is one of the main features of the application, it must work correctly and smoothly.

Testing and Proper Functioning of Notification Features: Testing whether the notification features of the application are working correctly. This means notifications are sent on time, have the right content, and are presented to the user for viewing or interaction.

Testing User Authentication and Membership Features: Testing whether basic membership functions such as users registering to the application, logging in and resetting their passwords work correctly. This allows users to manage their accounts securely.

1.1.2. Out of Scope

- Deep performance tests beyond basic speed and responsiveness checks are out of scope.
- Testing on specific devices or operating systems not specified in the scope is excluded.
- Security testing beyond basic authentication checks are out of scope.
- Localization and internationalization tests are out of scope.
- Accessibility tests for disabled users are out of scope.
- Compatibility testing with all possible device configurations is out of scope.
- Legislation compliance tests are out of scope.

1.2. Quality Objective

- The response time of our applications must be consistent across different devices.
- The response time of our applications must be under 1 second at a 95% confidence level.
- Our application provides optimum performance even during times of heavy use and should work without interruption in heavy user traffic.
- In case of any problems, users are presented with clear error messages. At the same time, in case of errors, a maximum response time of 4 hours is provided to identify and resolve the problem.
- The interface of our application should be user-friendly. In this way, it is aimed for users to navigate the application easily.
- User feedback should be monitored regularly. This feedback should be used to improve the application.
- All user data, including login credentials, must be securely protected and any possibility of unauthorized access or data leakage must be prevented in advance.
- Compliant with Turkish Data Protection Laws (KVKK)
- The application should be accessible from both tablets and smartphones.
- All system components are easily testable and testing processes can be carried out efficiently.
- Continuous tests are carried out for code quality and integration and the system is constantly evaluated.

1.3. Roles and Responsibilities

Roles	Responsibilities
Database Administrator	It ensures the performance, security, and integrity of the database. Manages database tests during the QA process. It also ensures that the database system is working correctly.
Developers	Tests software codes. Responds to bug reports and makes fixes to the application based on feedback from the QA team. Works in accordance with coding standards to ensure the software is testable.
Software Architect	It is responsible for the overall design of the software. It creates the structure of the software in terms of testability. Plans and designs the architecture of the software to meet QA requirements.
Tester	It tests different parts of the software. They create bug reports and evaluate the quality of the software. They also implement test scenarios and evaluate the usability, performance, and reliability of the software. Thus, they play an important role in ensuring that the software reaches the desired quality standards.
QA	It deals with the management and control of the QA process. Creates and runs test strategies. Manages resources. It also guides the members of the team. It monitors whether the QA process is compatible with the project goals.
QA Manager	Responsible for managing the QA process. Creates strategies, manages resources, prepares test plans and directs the team. It plays a critical role to improve the quality of the project.

2. Test Methodology

2.1. Overview

When starting the FocuZone project, we decided that the functionality, performance quality and user experience of this project should always be as high quality as possible. In line with these goals, we adhere to a structured testing methodology to provide users with a safe and fluid experience, so we chose to use agile testing methodology and continuous testing methodology in the FocuZone project. The main reason why we wanted to use these methodologies was to ensure that both methods are safe and they are minimizing the risk of a problem in the project with constant checks.

Agile methodology's approach to responding to change rather than comprehensive planning, and its approach of dividing large test elements into small parts, helps us successfully apply the test phases of the project. At the same time, with sprints, scrums, sprint review meetings and retro meetings, which are a benefit of Agile, the dynamic communication within the team is established very firmly about the developments of the test phases and we got rapid solution of the problems encountered because we always stay in touch because of these meetings. Also, we used continuous testing, due to its very solid work with Agile Methodology. It is used in the FocuZone project because continuous testing is the most suitable method that increases product quality and can be used to perform tests frequently and in a short time.

2.2. Testing Levels

2.2.1. Unit Testing

The Unit Testing phase is based on checking the functioning of each component in the FocuZone project separately from other components and proving that it is correct in terms of functionality. For example, our 4 main features in the FocuZone project, the to-do list feature, the study assistant feature, the sound level meter feature and finally the study environment photo analyzer, are unit tested specifically for each of these features.

2.2.2. Integration Testing

Integration testing phase allows us to make sure that the components are working in harmony with each other in order to check whether the components tested separately during the unit testing phase cause errors or bugs when integrated with each other.

2.2.3. System Testing

System testing involves testing the entire system to check the suitability and functionality of the system according to specified requirements. It tests the overall interaction of components, end-to-end user flow. With the completion of this test, the risk of error in the project is minimized.

2.2.4. Acceptance Testing

Acceptance testing involves testing Functional and Non-Functional aspects of the system such as performance, security, usability, accessibility, compatibility, and reliability. The main purpose of this test is to prove that the system is ready to go live. In order to receive approval at the end of this test, the system must be ready for user experience, be free of bugs, and comply with all requirements.

2.3. Bug Triage

Bug triage, also known as defect triage, which is of great importance in the software development lifecycle, is used to find and solve errors. In the FocuZone project, great importance is given to ensuring that a user does not encounter bugs. Therefore, bug triage is performed regularly both at the beginning and throughout the project. Our team reports both the bugs encountered during development phase and the bugs encountered during bug scanning sessions held by our team. These bugs are then rated according to their importance and the time period in which their solution will take aka “story point” which it called in agile methodology. It is then assigned to a developer, and with this system, bugs are reported as soon as they are encountered. Critical bugs that significantly hinder the user experience are given the highest priority and are resolved immediately without any negative feedback from user interaction. This systematic error detection process helps us effectively resolve problems and maintain the integrity of our software products.

2.4. Suspension Criteria and Resumption Requirements

In the FocuZone project, suspension criteria and resumption requirements have been created to minimize damage to the testing process in case of unforeseen situations or problems encountered during testing, and to ensure that the testing process remains effective and efficient.

Suspension criteria outline the conditions under which testing activities can be temporarily stopped, such as critical system failures, while resumption requirements outline the steps necessary for testing to continue after the problems that caused the test to be suspended are eliminated. By adhering to these criteria and requirements, FocuZone maintains the quality of the testing process and takes care not to exceed the time allocated for testing.

2.5. Completion of the Test

In test completion, which is the last step in the testing life cycle, we report all the testing activities and results that we have finalized in the FocuZone project. The main purpose of this test is to decide whether our software product is ready for production release. During the test completion process, it is verified whether all test cases are executed or not, the results are analyzed, the bugs found are identified, and finally a test completion report is created, and these results are evaluated by the team. In short, this document contains a detailed report of the test results, identified problems and their solutions. FocuZone completes testing with great attention to detail and adherence to established procedures, ensuring the delivery of high-quality software products to its users.

3. Test Deliverables

3.1. Test Plan

Before starting the tests of our project, we wanted to start by creating a test plan on how we will do these tests and which parts the test should consist of. Thanks to this plan, we could more easily observe the progress of our tests and implement the changes we deemed necessary.

There are two important parts we need to test in our project. The first of these parts is our mobile application side. Since our project is about a mobile application, we focused on doing more tests on this side. Since we have a lot of screens on this side and some of these screens are connected to the Firebase system, we had to do a lot of tests here. We needed

to make sure that transitions between screens, some technical coding parts, and visual and animation parts were not corrupted, and that the data saved in Firebase and pulled from the Firebase Realtime Database system were displayed correctly. The second important part was that we had to test whether the environmental analysis feature, which is a feature in our application, was working correctly and how accurately it could detect images. Before integrating the system, we use in this feature directly into the application, this part should have been tested with manually selected photographs that could detect it in a way that suits its purpose. In line with our test plans, we wanted to test these places in general use by creating some test scenarios and we reported them in our section below.

3.2. Test Scenarios

Below you can see the test scenarios we made for the parts we emphasized that need to be tested in the test plan.

3.2.1. Scenario 1: Operability and Error-Freeness of Mobile App Side

My purpose in the scenario here is to test whether the user receives any errors while navigating through the application.

Test Scenario:

- The intro pages of the application are being passed.
- From the login screen, you go to the membership screen and try the membership part.
- After successfully signing up, the password is changed from the forgot password screen to try the password change feature.
- After the password change is completed, we log in to our account.
- First, notification permission is requested and after giving the permission, we create a test task on the to-do feature screen.
- We examine whether we receive notifications according to the task's due time, whether the task has been edited successfully, and its display on the screen.
- Since we do not see any errors in our to-do feature, we move on to our other screen, the Pomodoro feature.
- Here, we set up a short working period for testing purposes and check whether the time tracking animation we created gives errors or not, and also whether a notification will be sent when the study time is over and the tree icon will change.

- Since we do not observe any problems with this feature, we go to the screen of the sound meter feature, which is another feature of ours.
- We use our Sound Meter feature to test the animations on the screen and the result values by manually creating certain sound levels.
- Since we do not see a problem with the Sound meter screen, we move on to the screen of our last feature, which is the environment analysis feature.
- After making sure that our last feature interprets the photo sent to the system correctly and returns it to us, we open our profile dialog.
- After observing the total number of completed tasks, total minutes worked, and total number of completed work periods in the Profile dialog, we successfully exited the application with our exit button and ended our mobile test.

3.2.2. Scenario 2: Functionality of the Image Detection System Used in the Environmental Analysis Feature

The purpose of this test is to check whether the image detection system installed for the environment analysis feature in the application is working correctly.

Test Scenario:

- It was checked that the photo was sent correctly from the application to the system where image detection will be performed.
- It was checked whether the system was performing the operations correctly to detect clutter and evaluate the color scale in the environment.
- It was checked whether the photo and color scale results detected by the system were successfully sent to the mobile application.

3.3. Test Reports

As a result of the tests, we concluded that our application is suitable for general use. In the tests we conducted in our application, navigating and making requests within the application is very fast and provides comfort to the user. In our tests, we observed that we did not experience any UI errors on the application screens. All our UIs and animations work as they should. Additionally, no errors were observed in our application's systems integrated with Firebase. Our application can save data to the Firebase Realtime Database without any errors and can also retrieve the necessary data from there. We are also considering making

situations more user-friendly in the future, such as refreshing the screen during screen transitions in certain situations we saw while testing our application. This report, which we prepared with all these test results, helped us a lot to evaluate our application and improve its performance in some parts.

3.4. Test Records

We generally carried out the tests manually and proceeded by directly fixing the errors we saw. Especially in the application part, we were able to log important parts of the code and observe these places thanks to the messages we put in the logs. These logs also helped us track progress. When we saw any problem or situation that needed to be fixed, we were able to see the progress of the application by using Android Studio's debug system. In addition to seeing the test progress and records with logs in the application, we were also able to create UITests and conduct short experiments with the Espresso library we added to Android Studio. In this way, we were able to monitor the manual tracking we did with the same logs with an automatic system. These test records contributed a lot to our progress as they could clearly show us whether the data in the application was received or whether there were errors in the UI.

4. Recourse and Environment Needs

4.1. Project Environment

4.1.1. Database

The database will be having an essential role in storing data for register/sign-in, to-do list and storing the images for the Environmental Photo Analysis feature. The chosen database system is selected to meet these requirements of the project.

Database System: Google’s Firebase, a real time database will be used to handle user data which will be consisting of some attributes and images. Firebase’s easy to use interface and its capability feature with the language we are using.

Type: NoSQL Database

Version: Latest Version of Firebase

Features: Firebase is well documented cloud-hosted NoSQL that lets us store and sync data between user in real-time, and its storage system for images made it suitable for our usage in the application. Its ability to pass user data safely and fast aligned with the application's requirements.

Firebase is generally a good choice as it is covering all of our needs for photo analysis and other features combined.

4.1.2. Mobile Devices Project

While we developed the application with a real Android device, we also used some Android phone models as emulators in the development environment. The mobile device we used physically was the Samsung Galaxy A32 model, and the model of the device we used on the emulator was the Google Pixel 3 XL model. We also observed the operation of the application in different versions by using both API 33 and API 34 versions of the device model in the development environment we used on the emulator. Since we developed our application by looking at the Android device versions in general use, the minimum SDK of the Android version of the device to be used as the working environment must be 26. We selected these devices because our application is currently only designed for Android devices.

4.2. Testing Tools

4.2.1. Unit Testing

Making sure that all of the features work when they are intended to be crucial. To achieve this, all of the features will be tested both on an android device and Android Studio. The environmental photo analysis feature will be tested on VS Code. VS Code offers a straightforward syntax containing the necessary error messages and couple features that creates a better unit testing environment.

4.2.2. Test Case Tool

To get better understanding of test cases on the back-end part all of the back-end testing was done under SwaggerUI. SwaggerUI is a web-based UI tool that allows you to visualize the results of the test case executions. Integrating SwaggerUI to the backend testing process will enhance the streamlining on testing workflow.

4.3. Test Environment

4.3.1. Development Workstation

The workstation of developers serves as a primary coding and testing tool providing the necessary resources for efficient software development. The workstations are equipped with the necessary hardware support to create the needs of environmental photo analysis feature. They are equipped with strong enough GPUs and 32 gigabytes of RAM so that we could get the results for the coding faster. These configurations also provide the resources for the developers to write and test the codes efficiently, creating a consisting and productive development process across different platforms.

4.3.2. Mobile Devices

As a testing environment, in addition to a physical Android device in the mobile application section, we also used the emulators offered by Android Studio, as we mentioned before. Thanks to the Device Manager section in Android Studio, we were able to use any version and model of the phone as an emulator. We were able to expand our testing area, especially by using emulators with different screen sizes and different versions of the test environment. While it was not possible to perform such a variety of tests on a physical device, with emulators we were able to perform tests that we could not do with physical devices and create a much more stable application.

5. References

- Levels of testing: a complete approach to quality assurance. (2023, August 18). Testsigma Blog. https://testsigma.com/blog/levels-of-testing/
- *What is Test Completion and Test Completion Activity?* (2024, March 28). Testsigma Blog. https://testsigma.com/blog/test-completion/
- *Bug Triage/Defect Triage | What, why & how to improve?* (2024, January 2). Testsigma Blog. https://testsigma.com/blog/bug-triage/