CMPE 492

Senior Design Project II

Test Plan Report

Project Name: Cakery App

Project Members:

Selin Akyurt - 17878041578

Melisa Uzulu - 3906408574

Recep Berkay Engin - 22271682594

Supervisor: Yücel ÇİMTAY

Jury Members: Aslı Gençtav, Venera Adanova

Table of Contents

1. INTRODUCTION

Cakery, mainly has two interdependent applications: Cakery User App and Cakery Seller App. Both applicable roles; user and seller want a system that passes the full cycle of manual testing. Given the specificity of each application, it's very important to have the same quality. The Test Plan Report has been created to facilitate communication within the development team. This document describes methodologies and several approaches for testing facilities. The objectives, scope, test responsibilities, test levels, scheduled significant milestones, bug triage, requirements and test approach are all included.
What will be the test deliverables and what is considered to be in and out of scope have all be en clearly defined in this document.

1.1 Scope

The document's primary focus is on validating data in report output using the GUI that co mplies with the client's specified requirements specifications.

1.1.1 In Scope

Testing is required for all website functionalities outlined in the software requirement requ irements.

Module Name	Role	Description
Register and Login	User/Seller	A user/seller can have a account. User/Seller can create a account. The user/seller logs in to his previously registered account with his id and password.
User Profile Display	User	User's profile page should be displayed with all the relevant details such as name, phone, address etc.
Dashboard Display	User	User's dashboard page should be displayed with all the relevant details such as restaurant cards, related restaurant information, restaurant picture etc.
Ordering	User	User should be able to place the order successfully and recieve the confirmation message with the orderID.

Module Name	Role	Description
Add Location	User	The new locaiton details should be updated succewssfully and displayed in the user’s profile with new details.
Menu and Item Creation	Seller	A seller can create and manage the menu and items available for purchase.
History of Orders	Seller	A seller can view the history of orders placed and track their status.
New Orders	Seller	A seller can view and manage new orders as they come in.
Custom Cake Creation	Seller	A seller can create and manage custom cake orders.

1.1.2 Out of Scope

In the testing plan of Cakery, some functionalities are identified as outside of scope. Features listed below are not planned to be tested. Listed functionalities have been considered out of scope due to time constraints and complexity of implementation of testing process.

-Payment process

-Delivery tracking

-Application security

-Network performance

-Hardware Compatibility

1.2 Quality Objectives

The primary quality objectives of the testing plan for Cakery are to ensure that the application is fully functional, easy to use and meets the expectations and needs of its users. Any alterations, additions, or deletions to the functional specifications, design specifications, or requirements documents shall be documented and tested as thoroughly as possible given th e remaining project time and the test team's capabilities. The secondary goals of testing will be to find and expose all problems and dangers associated with them, inform the project team of all known problems, and make sure that all problems are properly fixed before release. The

program must be carefully and methodically tested to ensure that all system components are examined and that any bugs discovered are afterwards resolved.

1.3 Roles and Responsibilities

Role	Responsibilities
QA Lead	Review and approve issues after going over test cases.
QA	Run the test case and report any errors.
Developer	Fix issues reported by QA, implement changes based on QA Manager's feedback.

2. TEST CASES

The following section provides a comprehensive testing plan for Cakery. Test cases cover various modules including user registration and login, profile and dashboard displays, ordering, and custom cake creation. This testing strategy's goal is to guarantee that the app works as intended and offers each user a positive experience. The test cases were created to properly test the app's functioning while covering a variety of scenarios. The outcomes of these tests should offer insightful information into any potential problems or defects with the app and assist the development team in effectively resolving them.

2.1 Test Case Tables

*User Display

Test Case ID	Test Scenario	Test Steps	Expected Results	Pass/Fail
UPD-001	User can view his/her profile	1. Login to the app 2. Click on the "Profile" button	User's profile page should be displayed with all the relevant details such as name, email, phone number, address etc.	Pass
UPD-002	User can update his/her profile	1. Login to the app 2. Click on the "Profile" button 3. Click on the "Edit Profile" button 4. Update the relevant details 5. Click on the "Save" button	User's profile details should be updated successfully	Pass
UPD-003	User can change his/her password	1. Login to the app 2. Click on the "Profile" button 3. Click on the "Change Password" button 4. Enter the old password, new password and confirm password 5. Click on the "Save" button	User's password should be changed successfully	Fail

*Dashboard Display

Test Case ID	Test Scenario	Test Steps	Expected Results	Pass/Fail
DSD-001	User can view the dashboard	1. Login to the app 2. Click on the "Dashboard" button	User's dashboard page should be displayed with all the relevant information such as bakery cards, menu and item pictures etc.	Pass
DSD-002	User can view a bakery menu	1. Login to the app 2. Click on the "Dashboard" button 3. Click on one of the seller cards	User's selected bakeries menu information should be displayed with all the relevant details such as menu image, title, description etc.	Pass
DSD-003	User can view bakery items	1. Login to the app 2. Click on the "Dashboard" button 3. Click on one of the seller cards 4. Click on a menu	User's selected bakeries menu item information should be displayed with all the relevant details such as item title, description, image, price etc.	Pass

*Ordering

Test Case ID	Test Scenario	Test Steps	Expected Results	Pass/Fail
ORD-001	User can place an order	1. Login to the app 2. Select the desired cake 3. Add the cake to the cart 4. Proceed to checkout 5. Enter the delivery details 6. Confirm the order	User should be able to place the order successfully and receive the confirmation message with the order ID	Pass
ORD-002	User can clear cart	1. Login to the app 2. Select the desired cake 3. Add the cake to the cart 4. Click on the "Clear Cart" button next to the order	User's order should be cancelled successfully and the status	Pass

*Add Location

Test Case ID	Test Scenario	Test Steps	Expected Results	Pass/Fail
ALC-001	User can add a new location	1. Login to the app 2. Click on the "Add Location" button 3. Enter the new location details such as name, address, phone number etc. 4. Click on the "Save" button	The new location should be added successfully and displayed in the user's profile with the entered details	Pass
ALC-002	User can add location using Google Maps	1. Login to the app 2. Click on the "Find My Location" button next to the location to be edited 3. Update the location details 4. Click on the "Save" button	The location details should be updated successfully and displayed in the user's profile with the new details	Pass
ALC-003	User can delete an existing location	1. Login to the app 2. Click on the "Delete" button next to the location to be deleted 3. Confirm the deletion when prompted	The location should be deleted successfully and removed from the user's profile	Fail

*Menu and Item Creation

Test Case ID	Test Scenario	Test Seps	Expected Results	Pass/Fail
MC-001	Seller can add a new menu	1. Log in to the app as a seller 2. Navigate to the menu management page 3. Click on the "Add Menu" button 4. Enter a unique name for the menu 5. Click on the "Save" button	The new category should be added to the menu with the entered name	Pass
MC-002	Seller can add a new menu item	1. Log in to the app as a seller 2. Navigate to the menu management page 3. Click on the "Add Item" button 4. Enter a unique name for the item 5. Enter a description for the item 6. Enter a price for the item	The new item should be added to the menu with the entered details and assigned to the selected menu	Pass

Test Case ID	Test Scenario	Test Seps	Expected Results	Pass/Fail
MC-003	Seller can delete a menu item	1. Log in to the app as a seller 2. Navigate to the menu management page 3. Click on the "Delete" button next to an existing item	The changes made to the item details should be saved and reflected in the menu	Pass

*History Of Orders

Test Case ID	Test Scenario	Test Steps	Expected Results	Pass/Fail
HO-001	Seller can view the history of orders	1. Log in to the app as a seller 2. Navigate to the order history page	The seller should be able to view a list of all past orders with relevant details, such as order number, customer details, order items, and order status	Pass

*New Orders

Test Case ID	Test Scenario	Test Steps	Expected Results	Pass/Fail
NO-001	Seller can view new orders	1. Log in to the app as a seller 2. Navigate to the new orders page	The seller should be able to view a list of all new orders that have not yet been processed, with relevant details such as order number, customer details, order items, and order status	Pass
NO-002	Seller can update the status of a new order	1. Log in to the app as a seller 2. Navigate to the new orders page 3. Select an order to process 4. Update the order status to "In Progress" or "Completed"	The seller should be able to update the status of a new order so that order is proceeded and visible in "Order History"	Pass

*Custom Cake Creation

Test Case ID	Test Scenario	Test Steps	Expected Results	Pass/Fail
CC-001	Seller can create a custom cake order	1. Log in to the app as a seller 2. Navigate to the custom cake creation page 3. Choose the cake type, size, flavor, and frosting 4. Choose any additional decorations or messages and add image 5. Click on the "Add to Cart" button	The custom cake order should be added to the seller's cart and displayed in the "New Orders" page for processing	Pass

3. TEST METHODOLOGY

3.1 Overview

General software development approaches unfortunately face many limitations. It focuses on assuming that the software requirements are remain still throughout the project. However, this is not true since the complexity of the software projects increase, and client needs and wants change over time. Keeping up with the changes makes the launch of software applications delay and increases the overall cost. This is why testing is a huge important part of any software development since it helps to ensure that the software application meets the user wants, needs and quality.

Our project “Cakery” is an agile project. With agile project approach we have more flexibility, and it brings an iterative approach where we collaborate between us developers. With the incremental agile development approach, we break down the whole project into smaller increments. We test each increment and see if they meet the required needs before continuing with new increments. Our main goal to choose this approach was working on the project as fast as we can while communicate between each other and continuously refine and improve each increment in our project based on the testing result we get. This method also allowed us to quickly work on needed changed and adjust the project according to its requirements based on feedbacks, bugs and testing results. We usually spend on 2-3 weeks on each increment.

With this flexible and collaborative approach, we are aiming to meet the desired requirements of our project with higher quality while it saves us time and reducing risks.

3.2 Testing Types

- Black Box Testing: Black Box Testing or Behavioral/Partition testing is a testing type where testing focuses on the functional requirement of the software. It gets a set of input conditions that will test the specific requirement of the project. An example from our project would be placing a cake order from a bakery scenario. Test case would be ordering placement to do so we would open our Cakery App, login, choose a bakery and choose a cake from its menu, add it to the cart, choose an address and submit the order. To make sure this order successfully placed, we use the orderId and ensure that seller gets that order as a return. We repeat the steps with different users, different bakeries, and different items to make sure the feature is working -.

- GUI Testing: GUI Testing refers to the testing of the graphical user interface elements of the application. It is used to make sure that application functioning properly and provides a good user experience. This testing type includes visual design, layout, navigation etc. Before starting our project, we decided what kind of a visual we want from this application. We continued the whole project with same font and color choices. We tested the visual aspect of our project using the emulators and widget testing. Widget testing is a testing form in Flutter which allowed us to test the UI components we used in our project. We used widget testing to make sure each widget we applied into our project works correctly with specific test cases that interacts with individual widgets. Widgets that we used and tested mostly includes, buttons, text inputs, tapping, scrolling through and so on.

- Integration Testing: Integration Testing is used to construct the whole project with each increment and different components. It ensures that the whole project work together as wanted without bugs and errors associated with interacting the program. This also includes the testing report. To give an example from our application we can say that when we order an item from user app, we needed to integrate the same order for the seller app and show it from the chosen bakery's orders. We did bunch of tests with different input combinations to verify that order details, selected item, delivery address are correctly displayed in the specific bakery's order history. For this example, we focused on the integration between the order placement from the user app and order history from the seller app. We ensured the two components work together successfully.

- Functional Testing: Functional testing is used to find out unexpected behaviors of the program within the features and functionalities. It tests whether the characteristics of the functional testing are correct and in line with the wanted needs and provides a positive user experience. We are testing our applications a whole since it's in a correlation with two different apps and we test each component also suits with the corresponding application.

- System Testing: System testing is used for testing the entire system as a whole, with each component and interfaces. It aims to ensure all functions are correct and meets the requirements. At this current stage of our project, we can not test the whole application since we continue the development. However, we will be testing the whole project with different input combinations when our development finishes. As for now, we are testing as far we come with the project.

- Performance Testing: Performance testing is used for testing the application's performance under different conditions. It aims to make sure system performance can handle expected number of users and transactions. An example of performance testing from our project would

be different user and seller logins. We tested both user and seller application with different user and seller profiles. Our application performance handles different user and seller profiles fine as wanted. We are also checking the optimal time of our application pages to load, besides the first launch/start of the emulator, loading times of our application is also quick and as wanted.

- User Acceptance Testing (UAT): UAT testing is performed by the client or end-user and making sure the application works as they wanted and meets the needs. In this case since this is a senior year project, we developers are act as end users for the purpose of the testing and evaluating the project. We always investigate it without having bias, and we are making sure doing the best we can and improve the project as much as we can. We are also planning to improve and launch the "Cakery" app after the graduation so much more UAT testing will be applied to this project in the future where end users and clients differ.

- Alpha Testing: Alpha Testing is a type of testing where the software is tested in a controlled environment before its release. Due to our project development, we are not at this stage of testing yet. However, we plan on doing alpha testing in the future.

3.3 Bug Triage

Defect prioritization is defined as a critical step in software testing that requires reviewing software bugs, prioritizing those bugs, and assigning them to the appropriate team to resolve them. The bug prioritization phase is an important part of the software development lifecycle. In this section, we will talk about what to do after detecting a bug in our test scenarios.

Our application and project management are based on Google Cloud Firestore Services. We use Flutter for the project development part. First of all, we divide the errors in the system into three general categories. These;

1- Errors that need immediate intervention by the developer (for example, rolling back the new version, eliminating the error that appeared with the new version),

2- Errors that do not require immediate resolution (if the error becomes more frequent, if the error affects more than one user in the future),

3- Errors that never require developer action and can be ignored (bugs related to features that no user has used),

We divide it into three. When we discover a new error in the system, we will add this error to the working environment so that everyone is aware of the error. We will then find the reasons that caused the error to appear, together with the responsible developer, and assign the error. Then the developer will implement the error and check if it is working properly. The developer will consider the three categories we identified when fixing the bug. After correcting the error, we will provide control again. We'll continue to take a look at other bugs and improvements once the bug is gone.

3.4 Suspension Criteria and Resumption Requirements

Suspension Criteria and Continuation Requirements is one of the important elements we consider when developing our application. These elements are defined as reference to the

conditions under which an application may be suspended or temporarily suspended and the requirements to continue normal operation. Our project focuses on online ordering. If there are any issues with ordering (which may include factors such as network connectivity issues, user leaving the App or locking their device, or a user-initiated request to pause the app), the app can be suspended to prevent it from consuming system resources unnecessarily. The continuation criterion is defined as when the dependent components are reusable, and the defect has been successfully resolved. Resume Requirements, on the other hand, can be defined as outlining the steps required to revert the application to its previous state after suspension. This may include reloading data, reconnecting network connections. For example, if the user is disconnected from the network during suspend, the user may be prompted to wait for the application to reconnect and reload the necessary data. In addition, if the user's session has expired within the suspension period, the user may be prompted to log in again before being allowed to continue using the application.

3.5 Test Completeness

Testing completeness is defined as the extent to which an application is thoroughly tested to ensure it meets functional and non-functional requirements.

Our testing part will be complete when the product ordering part, server-side and front-end part works well. In the product ordering test section, it performs correct order number matching, correct product identification and correct user matching. On the server side, if we can retrieve and store the correct data from the server, our server testing will be complete. In addition, we have successfully completed testing the interactions between the different components of the application to ensure that the user and vendor applications work together seamlessly, and we have also completed our integration testing. In the non-functional test part, our application includes performance and usability tests. Functional testing includes testing the features of the vendor and user apps, ensuring that they are working as intended. Our online cake ordering application, which we developed with Flutter, has gone through many different tests to ensure it is complete. We used manual testing techniques to cover all possible scenarios, edge cases and user input.

4. TEST DELIVERABLES

Test deliverables for Cakery involve test cases, test plan, methodology, test reports and any other relevant document. The test cases created for Cakery modules have been designed to cover through different scenarios to test overall application functionality and quality. Test plan covers objectives of testing process, approaches, roles and responsibilities of the development team and schedule of testing activities.

5. RESOURCE & ENVIRONMENT NEEDS

5.1 Testing Environment

A test environment refers to the setup used for testing software applications to ensure that they function correctly and meet user requirements. The test environment includes hardware, software, and other necessary resources required for executing tests on the software application. In case of our project Cakery, which is a cake ordering app developed with Flutter, our test environment includes the necessary components that simulate the production environment. For our project, main testing environments are emulated environments and cloud-based environments. For emulated environments we used emulators in the Android Studio. For cloud-based environments we used Firebase Cloud Firestore.

Emulated Environments:

Emulators can be used to test the software application that hardware testing is not available. We also choose to test our project using emulators for many reasons. Main reason is because of its efficiency. Using emulators allowed us to quickly see our program input visually and catch the errors and bugs. It reduced the time and cost needed for testing. It is accessible for all three members of our group project, and it is flexible. We also choose emulators because our research showed us, they provide consistent and reliable testing. Another benefit of using emulators in our project was to see how our application works with different mobile devices without physically buying each device. This was a huge gain for our project in terms of seeing the portability of our applications. Due to the limitations of our testing environments and limitations of our software/hardware we were only able to test our project in Android mobile devices thus our project will be Android based in the first stage. However, we are aiming to improve our testing environments and test our application with an IOS mobile device as well.

Emulators used in this project are:

- Nexus 5 API 30 | Android 11.0 | Google Play | x86
- Pixel 2 API Android 11.0 Google Play | x86
- Pixel 3 API 30 | Android 11.0 | Google Play | x86

Cloud-Based Environments:

Cloud-Based environments allows developers to test their applications in a scalable and flexible way with dynamic resources that allocate as needed. It is a process of testing the software application in a cloud-based environment. We used this testing environment approach in our project because it allowed us to set up and configure test environments quickly and easily without needing external hardware. Cloud storage also allowed us to store our data on a remote server and we were able to access it whenever we needed it. We use Firebase-Cloud Firestore developed by Google and it is a cloud-based NoSQL document type database. During the testing of our application, it allowed us to create test data and perform different combination of test inputs. It is also cost friendly and was perfect choice for our project due its flexibility and lack of complexity of setting up.

5.2 Testing Tools

Testing tools are software applications that help ensure your application works correctly before you publish it. Various testing tools are available for Flutter, including unit tests, widget tests, and integration tests. There are several test tools that we can use to ensure the functionality and performance of the online cake ordering application we have developed thanks to Flutter.

In addition to these tools, other testing tools are also used, such as Flutter Test and Widget Test, which provide different levels of testing capabilities for Flutter applications. Flutter Test is used to test the functionality and performance of the app, while Widget Test is used to test individual widgets and their interactions with other widgets. We do these tests manually for our application. We used unit tests to verify that the individual functions and methods, widget work properly. we used tests to verify the UI of the individual widgets and integration tests to verify that all parts of the app work together as expected. For this, we prefer to use the emulator part of Android Studio frequently. Every time we add a new feature or make changes to any feature, we test the feature instantly so that it does not affect users. Apart from that, we used Flutter Driver which is used to test the user and vendor interface. Thanks to the Flutter Driver, we used automated tests that simulate user interactions with the app, such as tapping buttons, switching between screens, and entering text into forms. In addition to these tools, we used the Dart Observatory tool to analyze memory usage and performance. In this way, before activating our application, we checked whether there is a section that is not working correctly and for its intended purpose.

6. REFERENCES

[1] Kirti Bhandge, Tejas Shinde, Dheeraj Ingale, Neeraj Solanki, Reshma Totare,"A Proposed System for Touchpad Based Food Ordering System Using Android Application", International Journal of Advanced Research in Computer Science Technology (IJARCST 2015).

[10] Abhishek Singh, Adithya R, Vaishnav Kanade, Prof. Salma Pathan" ONLINE FOOD ORDERING SYSTEM" International Research Journal of Engineering and Technology (IRJET) 2018.

[3] Sarika Rane1 , Varad Potdar , Harsh Trivedi, Yash Ughade, "Designing White Box Test Cases for Online Food Delivery System", Computer Engineering, Shah and Anchor Kutchii Engineering College, Mumbai, India, 2021.

[4] D. Kotelenets, Test Plan Template.

[5] Flutter, Testing Flutter Apps, https://docs.flutter.dev/testing

[6] Flutter, Integration Testing, https://docs.flutter.dev/testing/integration-tests

[7] Modus Create, Introduction to Flutter Widget Testing, https://moduscreate.com/blog/introduction-to-flutter-widget-testing/

[8] GeeksForGeeks, Types of Software Testing, https://www.geeksforgeeks.org/types-software-testing/

[9] Atlassian, Different Types of Software Testing, https://www.atlassian.com/continuous-delivery/software-testing/types-of-software-testing