Test Plan for FOODIFY

Duong Ngoc Yen - Quality Engineer

Verma Nandini - Lead Developer, Front - end Developer

Kolady Anamika Martin - Quality Manager, Release Engineer

Jim Sean - Back - end Developer

Anandarajan Sindini - Project Manager

Team Runtime Terror

School of Computer Science and Engineering

Nanyang Technological University, Singapore

Version History

Version #	Implemented By	Revision Date	Approved By	Approval Date	Reason
1.0	Kolady Anamika Martin Verma Nandini	29/10/2022	Duong Ngoc Yen	30/10/2022	Initial version

Table of Contents

1. Test Plan Identifier - TP 1.0

The test plan for Foodify contains the scope, approach, and schedule of the testing activities to be undertaken for Foodify throughout its lifecycle as well as during the maintenance phase. The fundamental testing criteria and cases have been documented and the basic functionality has been implemented and tested in the software application. This plan will gradually transition to the master plan as the testing becomes more robust and comprehensive. The plan contains information about the resources and procedures for testing of Foodify, and how the testing process is controlled, and configuration is managed throughout the product life cycle.

2. Introduction

The Test Plan provides a comprehensive overview of the scope, approach, resources, and schedule of all testing processes and activities to be performed for Foodify. The plan identifies the items and features to be tested, the types of testing to be performed, the personnel responsible for testing, the resources and schedule required to complete testing, and the risks associated with the plan. Four types of tests will be carried out, namely, Unit Testing, Integration Testing, System Testing and User Acceptance Testing. More specifically, we will be using a black-box approach.

1. **Unit Testing:** Unit testing will be performed to test the individual units or components of Foodify in isolation. This will validate each component of the application. Unit testing will be performed during the development phase.
2. **Integration Testing:** The individual tests will now be tested together to test the combined functionality of the Foodify application.
3. **System Test:** Quality Assurance will perform independent testing to test the entire system as a whole and provide feedback to the development team.
4. **User Acceptance Test:** Selected end users will be asked to use the application and provide their feedback which forms the user acceptance test results.

3. Test Items (Functions)

The following functional test items are identified for the above mentioned tests:

3.1 Unit testing

3.1.1. Side Bar - There are 6 pages accessible from the Side Bars: Home, Add Food Items, My Food Items, Sell Food Items, Explore Recipes, Marketplace. When the User clicks on a tab, the system should navigate the User to the respective pages.

3.1.2. Page Navigation - The drop down present at the top right of the screen of Foodify: Home, Add Food Item, My Food Items, Sell Food Item, Explore Recipes, Marketplace. The user should be redirected to the respective pages when the buttons are clicked.

3.1.3. Logout Button - The system must navigate the User to the start page, and log out from the system when the user clicks on the logout button.

3.1.4. Choose Food Item - When a User clicks on the Food items input in the selling post page, there is a drop down for the user to choose. The drop downs must contain all the Food Items the User currently has.

3.1.5. Viewing Food Items - When the user clicks on the View Food Items page, he should be able to view all the Food Items added by him/her previously.

3.1.6. Delete Food Item - When the user clicks on the “Delete Food Items” button on the View Food Items page, the user should be able to delete whichever food item they wish to delete.

3.1.7. Choose Food Item - When a User clicks on the Food items input in the selling post page, there is a drop down for the user to choose. The drop downs must contain all the Food Items the User currently has.

3.1.8. View Marketplace - When the user clicks on the Marketplace button from the dropdown menu, he should be able to view all the posts on the Marketplace.

3.2 Integration Testing

3.2.1 Sign Up - The user must be able to register a new account using Firebase Authentication

3.2.2 Login - The user should be able to login into the system using their credentials upon verification by Firebase Authentication.

3.2.3 Logout- The user should be able to log out of the system.

3.2.4 Change Password- The user should be able to change password from old to new password and is able to log in using the new password.

3.2.5 Edit Account- The user should be able to change their profile, including: Name, Phone number

3.2.6. Addition of Food Item - The user must be able to login and add food items to their account.

3.2.7. View Food Items - The user must be able to login, add food items to their account and see all their food items added under the view food items page.

3.2.8. Post Food Items - The user must be able to login, add food items to their account, see all their food items added under the view food items page and post a food item for sale on the marketplace.

3.2.9. View Marketplace - The user must be able to login, add food items to their account, see all their food items added under the view food items page, delete any food item they wish to delete, post a food item for sale on the marketplace and view food items posted by them and other users on the marketplace.

3.2.10. Contact Seller - The user must be able to contact the food seller to purchase/find out more about a food item.

3.2.11. Recipes Recommendation - The user must be able to view recipe recommendations of their added Food Items.

3.3 System Testing

3.3.1. Smoke testing - The user must be able to login to the application and be able to navigate across and use all the features designed.

3.3.2. Stress Testing - Multiple users must be able to use the application simultaneously.

3.3.3 Scalability Testing- The Database should be able to process queries from multiple users simultaneously without crashing

3.3.4. Recoverability Testing - The data stored for any user must be stored in the database if the website crashes or the device loses internet connectivity.

3.4. User Acceptance Testing

For the User Acceptance Testing phase, a beta version of the web application will be released to a selected beta tester and their feedback will be used for launch and the users will be asked to give their feedback.

4. Software Risk Issues

Some of the foreseeable and potential software risks are:

4.1. Integration of Spoonacular API and potential changes might impact certain functionality of Foodify

4.2. Ability to use and understand a new package/tool and so on

4.3. Maintenance of certain complex functions

4.4. Modifications to components with a past history of failure

4.5. Poorly documented modules or change requests

Unit testing will help identify potential areas within the software that are risky. If the unit testing discovers a large number of defects or a tendency towards defects in a particular area of the software, this will be considered as an indication of potential future problem(s). A suggested approach to tackle such a scenario is to define where the risks are by having brainstorming with the team.

5. Features to be Tested

The features to be tested will be given a risk rating, which is one of three levels: High, Medium, Low.

High level risks have the highest importance and must be tested and debugged as soon as possible. Medium level features are to be tested only after high level bugs. Low level features are not as important and will only be tested once all other features are working properly.

Item	Risk
Authentication	High
Add Food Item	High
View Food Items	Medium
Delete Food Item	Low
Post Food Item	Medium
View Marketplace	High

6. Features not to be Tested

All the features implemented in the application will be tested, since all the features are high priority necessary to run the application.

7. Approach

A test approach specifies how testing would be performed. It is the test strategy implementation of a project. A Test approach has two techniques:

Reactive - An approach in which design and coding are completed first before testing.

Proactive - An approach in which the test design process is initiated as early as possible with an aim to find and fix the defects before the build is created.

7.1 Testing Approaches

Apart from the above stated two techniques, a test approach based on different context and perspective may be categorized into following types:

7.1.1 Methodological Approach

This approach consists of all the methods which are pre-determined and pre-defined, to carry out the testing activity. The methods covered under this approach are used to test a software product from each different perspective and requirements, ranging from static analysis of the programming code to dynamic testing of the application.

7.1.2 Analytical Approach

Analytical approaches involve, selecting and defining the approaches based on the analysis of some factors or conditions associated with the software product, which may produce some significant changes to the testing environment, such as on requirement basis, risk based, defect severity basis, defect priority basis, etc. For example, when defining and preparing the test approach based on risk, it may include an approach/method to consider the area of higher risks, for the execution under the testing phase, whereas that of lower risk may be taken up at a later stage.

7.1.3 Regression Averse Approach

It includes preparation of an approach based on the usage of existing test material and automation of regression test suites.

7.1.4 Model-based Approach

Approach based on some specific model, build up on some sort of statistical or mathematical value associated with the software product entity or its functionality, such as rate of failures, etc. Generally, it may be seen as a preventive approach, which should be initiated as soon as the model used in the SDLC is identified and selected, in the early stage of development life cycle.

7.1.5 Dynamic or Heuristic Approach

The approach involves heuristic testing types such as exploratory testing, where tests are designed, created, and executed as per the current scenario and is not pre-planned unlike other approaches. Basically, it is a reactive test approach which carries out the simultaneous execution and evaluation of the test cases.

7.1.6 Standard Compliant Approach

As the name suggests, the approaches are based on some specific regulation, guidelines, or industry standards such as IEEE 829 standard. The testing activities covered under this approach like designing and creation of test cases follows and adheres to the given specified standards.

7.1.7 Consultative Approach

This approach is based on the recommendation and suggestion given by the experts or other professionals from the business or technical domain outside the boundary of the organization, which may include end users also.

7.2. Steps in Testing

A test plan approach can be summarized with the six steps illustrated in Figure 1.

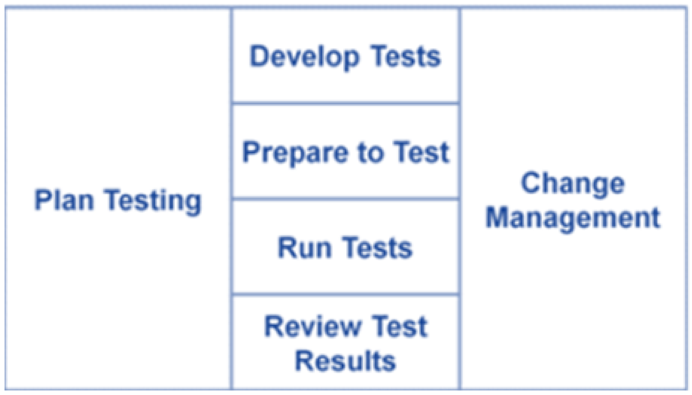

Figure 1: Steps of the test plan

The activities performed in each of these steps are described below:

7.2.1 Plan Testing - This is where the timescale, scope, quality, risk, and resources are decided in advance, and kept up to date throughout the User Acceptance Testing, including Entry Criteria and Exit Criteria.

7.2.2. Develop Tests - This involves numerous activities shown Fig-2 in order to develop the formal tests required to run any form of testing:

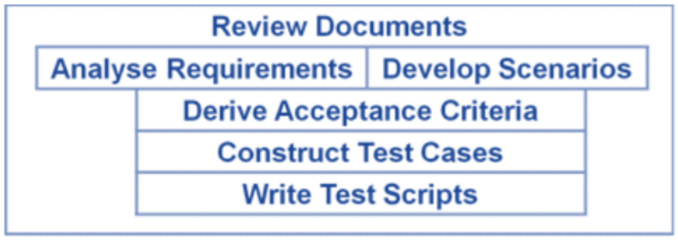

Figure 2: Activities in the Develop Tests Step

- **Review Documents**

  This is a key quality process of checking all documentation produced during the development of the system.

- **Develop Scenarios**

  This involves the preparation of scenarios that use the system, using techniques such as Use Cases.

- **Analyze Requirements**

  This involves understanding the formal tests and looking for what is inconsistent and missing from what is actually required.

- **Derive Acceptance Criteria**

  Once the previous two activities are underway or completed then the set of questions to be asked about the system to see if it matches the capability needed are prepared.

- **Construct Test Cases**

  Test Cases are the set of specific inputs and expected results which enable one or more Acceptance Criteria to be proved.

- **Write Test Scripts**

  Test scripts are the operational instructions for running Test Cases. It includes what has to be done to the system, what has to be measured, and how to do the measurement.

7.2.3. Prepare to Test

These are the activities other than developing the tests that are required to allow testing to take place:

7.2.3.1.Preparing Test Data - Building the data files that are required to run the test cases.

7.2.3.2.Preparing the environment to run the tests - Making sure that the people, processes, hardware, software etc. are all in place to enable the testing to take place.

7.2.3.3.Creating three key documents:

- **Test Procedure** - The instructions about how to run the tests on the test system including the Test Scripts.
- **Entry Criteria** - What must be done before testing starts.
- **Test Item Transmittal Report** - The instructions from the developers about the system version released for testing.

7.2.4. Run Tests

Run Tests comprises of running the tests and recording the results:

- Running the tests involves using the input and expected results from the Test Cases and applying the Test Scripts and other elements of the Test Procedure to run the tests.
- Recording the results involves recording inside the Test Log the order in which the activities were performed, and the events that occurred when the test was run. If any test has actual results that differ from the expected

results, have the information recorded in an Incident Report. The Incident Severity is also decided at this point.

7.2.5. Review the Results

The acceptability of the system is assessed once the tests have been performed. A simple method involves determining how many outstanding incidents were there and their corresponding severity. However, this is not sufficient as a mere count of incidents does not give much detail into their impact on what the organization aims to achieve with the system. Therefore, the test results need to be checked and traced to see what effect they have on:

- Business or System Impact,
- Requirements and Scenarios

This analysis ensures that a balanced decision can be made as to whether the system passes these particular tests and also allows it to make effective recommendations about its use. The results of all this activity are then recorded in a Test Summary Report.

7.2.6 Change Management

This process involves making an impact analysis for any changes for their effect on the system. At the starting stage of testing, or even when it gets closer, changes can have a major influence on how effective the testing will be.

7.3 Approach for Foodify

For our project we have followed a methodological approach.

Tests will be carried out as per the documented test cases stored in the Test Summary Report. The Quality Assurance engineers will be responsible for creating the test runs. The tester will execute the tests in the Test Summary Report and mark each case as Pass/Fail. The tester should record down the actual results in the Test Summary Report. When tests are marked as Fail, developers that are responsible for the corresponding feature should look into the errors and correct the system.

The Quality Assurance Engineers will review the test runs in the Test Summary Report and review alongside the Quality Assurance Manager accordingly. The test cases will be presented to the Quality Assurance Manager before they are finally marked as passed. When testing is deemed to be complete by the QA Manager, a test report will be submitted to the project manager who will mark the testing as complete if the application works as it should.

Following are some of the factors that were considered when selecting the test approach:

- The nature of the Foodify web application.
- Risks of product or risk of failure for the environment and the team.

- Experience and expertise of the people in the proposed tools and techniques.

8. Item Pass/Fail Criteria

This section deals with defining when an item has passed or failed a test in this project.

In our project, for each test case, the tester is equipped with the proper input and the right output. The test case will only be deemed to have passed if the result matches exactly with the ideal output defined in the test case. If the output is not exactly the same, the test case will be deemed to have failed. In case it is not clear whether the output matches the ideal result, the Quality Assurance Manager will be contacted who might consult the Project Manager in case necessary.

The completion criteria for this plan are:

- 100% of unit testing cases are complete
- 90% of integration testing cases are complete and 10% cases or lesser have minor defects
- A minimum of 5 users have given their feedback for User Acceptance Testing.
- System Testing is done with at least 5 external testers.

9. Suspension Criteria and Resumption Requirements

9.1 Suspension Criteria

In Suspension, the testing team will suspend the testing activities based on some criteria. The test team decides whether to suspend the complete or a selected part of the software testing process. Suspension can occur when the external components are not readily available or when a serious defect is detected. Suspension is also known as Test-Stop criteria for the testing process. Furthermore, testing processes are suspended mainly when the actual result is not the same as that of the expected result. Technically speaking, suspension occurs when the output of the parallel program is not identical to the output of the sequential program. The figure below depicts when the testing process might be suspended. The testing team needs to stop the testing if a defect is detected at a point after which the testing is not valid for the test-case.

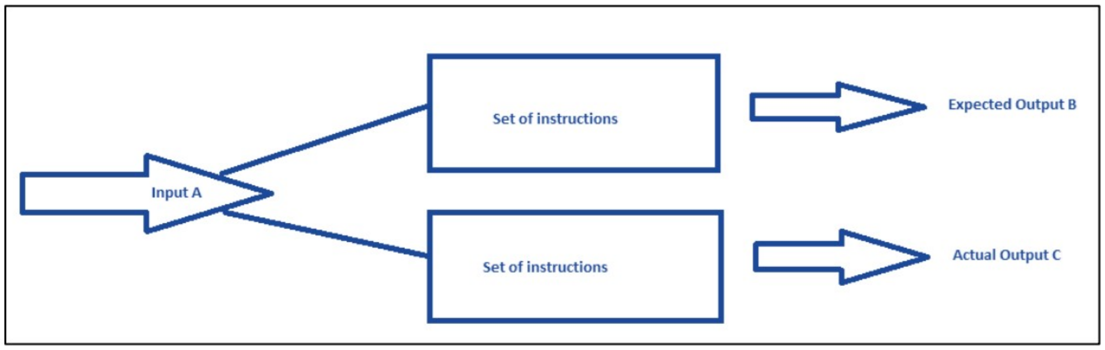

Figure 3: Suspension of the Testing process

In our project, if the testing team is suspending any testing process, a reasonable justification and relevant corresponding documentation which defines the acceptable level of the defect is to be provided. There may be various scenarios to suspend the testing process such as network, hardware, or the defect in the code. Below are a few common criteria that lead to the suspension in the software testing process:

- Hardware or software or API is unavailable.
- Any serious defect which highly impacts the testing progress.
- The testing process can be limited by the defects in the build.
- Any problem related to connectivity.
- Test resources are not available when needed.
- Schedule of the project (Giving priority to deliverables).
- The output is not the same as expected.

- Functionality does not work as specified in SRS (System Requirement Specification).

In our project, if the team members report that 40% of test cases have failed, the Lead Developer will be informed about the test cases which have failed and testing will be suspended until the development team fixes all the failed cases.

9.2 Resumption

Resumption is restarting or resuming the process which is invoked after the suspension criteria are met. It is the contrary process of suspension. It involves verification of the defect due to which suspension was invoked during the testing process.

Below are a few common criteria to resume the testing process:

- Previously unavailable hardware or software resources are available now.
- Issues due to which suspension occurs are resolved.
- No further defect has been found in the resumption technique.
- Output meets what is being expected.

To summarize, the suspension criteria specifies the criteria to be used to suspend all or a portion of the testing activities while the resumption criteria specifies when testing can resume after it has been suspended.

10. Test Deliverables

Test Deliverables are the test artifacts which are given to the stakeholders of a software project during the SDLC (Software Development Life Cycle). Some of the deliverables are provided before the testing phase commences and some are provided during the testing phase and rest after the testing phase is completed. The deliverables included in this Test Plan are:

10.1. Test Cases

Test cases are the set of positive and negative executable steps of a test scenario which has a set of pre-conditions, test data, expected result, post-conditions, and actual results.

10.2. Test report

This contains the test results and the summary of test execution activities.

10.3. Test Plan Document

Test plan document is a document which contains the plan for all the testing activities to be done to deliver a quality product. The test Plan document is derived from the Product Description, SRS, or Use Case documents for all future activities of the project. It is usually prepared by the Test Lead or Test Manager.

10.4. Revision Logs

The revision history of all the documents will be duly documented and stored.

10.5. Defect logs with solutions implemented

All the defects identified will be documented and logged in the appropriate documents as well as the solution implemented for the respective defect.

11. Test Tasks

The following activities must be completed:

i. Test plan prepared
ii. Items to be tested are identified
iii. Identify method of conducting tests
iv. Personnel assigned to each test case
v. Conduct testing
vi. Fix bugs and errors which are discovered
vii. Create defect logs
viii. Create test report

12. Environmental Needs

For the testing of Foodify, following specifications and requirements have to be met:

- Linux, Mac OS X, or Windows.
- git (used for source version control).
- An IDE. Visual Studio Code is our flagship IDE. Any appropriate IDE can be used.

13. Staffing and Training Needs

Training will be required to set up the application on the testers' machines as well as to get acquainted with the ReactJS testing frameworks. The Lead Developer will be responsible for providing training on how to run the application and the structure of the code. The QA Engineers can consult the front-end and back-end developers if they run into any problem(s). The QA Manager will be responsible for finding methods to train the testers on the ReactJS testing frameworks.

14. Responsibilities

Role	Responsibilities
Project Manager	Deciding which features should be tested. Monitoring the errors found. Collecting and analyzing the final testing report.
Lead Developer	Providing required training in code details. Providing solutions for running the code. Conducting white-box testing on the entire system
Release Engineer	Collecting details of test runs to make sure software meets requirements before deployment.
Front-End Developer	Conducting white-box testing on the front-end components.
Back-End Developer	Conducting white-box testing on the back-end components.
QA Engineer	Conducting black-box testing. Reporting test logs to the QA manager Generating additional test cases as needed.
QA Manager	Stating the risk and contingency plan for the different phases of the test. Monitoring testing activities and ensuring all testing resources are available. Setting overall testing strategy.

15. Schedule

The details for the testing phase for Foodify are provided in the Project Plan. Our planned activities will be as follows:

15.1.1 A daily backlog will be maintained to keep track of items which are yet to be completed. The Quality Assurance Manager will continuously monitor this backlog and will help the Quality Assurance Engineers if the backlog gets too large.

15.1.2 Number of days assigned to each test case will depend on their risk level. High risk will get 6 days, medium risk will get 5 days and low risk will get 4 days. This is to ensure that too much time is not wasted on testing low risk items.

15.1.3 For any bugs that are discovered, the result will be immediately updated on the GitHub Projects Board and the Lead Developer will be notified immediately.

16. Risks and Contingencies

The risks and methods to mitigate them are as follows:

Risk	Contingency
Shortage of Testers	Members of the development team as well as management will pitch in to help with the testing if the QA team gets overwhelmed..
Improper Training for Testers	The QA Manager and Lead Developer will be responsible to provide all the required training.
Improper communication amongst testers and developers	The daily testing backlog will ensure testing is on track. Testers will be required to report any bugs they discover immediately to the Lead Developer.

17. Approvals

Role	Test Type	Approval Criteria
Release Manager	Black Box	User Acceptance Testing meets acceptance criteria.
Project Manager	Overall	The application works as required and user acceptance testing meets client needs.
Lead Developer	White Box	System performs in accordance with functional requirements.
QA Manager	Black Box	All test cases are covered.

18. References

Document	Link
Project Plan	https://drive.google.com/file/d/12HsSjnkHaIpRkSfJX_9foEPu21dU4xM2/view?usp=sharing
System Requirements Specification	https://drive.google.com/file/d/1Cg2jFAuaWHIAe_pdZJs2pdI-yOfPpJ-n/view?usp=sharing
IEEE Test Plan Standard	https://drive.google.com/file/d/1hsy8kAdX4YkSG7CiVKiw9eIPog7laMWr/view?usp=sharing
ReactJS Environment setup guide	https://reactjs.org/docs/create-a-new-react-app.html