**Test plan for Food Aggregator Application**

Table of Contents

[1 Introduction](#_Toc515524393)

[1.1 Purpose](#_Toc515524394)

[1.2 Project Overview](#_Toc515524395)

[2 Scope](#_Toc515524396)

[2.1 In-Scope](#_Toc515524397)

[2.2 Out-of-Scope](#_Toc515524398)

[3 Testing Strategy](#_Toc515524399)

[3.1 Test Objectives](#_Toc515524400)

[3.2 Test Assumptions](#_Toc515524401)

[3.3 Level of Testing](#_Toc515524403)

[3.4 Functional Testing](#_Toc515524405)

[3.5 User Acceptance Testing](#_Toc515524406)

[3.6 Regression Testing](#_Toc515524407)

[4 Execution Strategy](#_Toc515524408)

[4.1 Entry Criteria](#_Toc515524409)

[4.2 Exit criteria](#_Toc515524410)

[4.3 Validation and Defect Management](#_Toc515524411)

[5 Environment Requirements](#_Toc515524412)

[5.1 Test Environments](#_Toc515524413)

[6 Test Deliverables](#_Toc515524412)

[7 Test Approach](#_Toc515524412)

## 

1. **Introduction**
   1. **Purpose**

The purpose of a food aggregator is to simplify the process of ordering food online, offering users convenience, choice, and transparency while optimizing the delivery process for restaurants and delivery partners.

* 1. **Project Overview**

Develop a comprehensive food aggregator app that consolidates restaurant information, menu options, prices, and delivery services into a single platform, providing users with a seamless and convenient way to discover, order, and enjoy a variety of cuisines. This comprehensive test plan ensures systematic testing of the food aggregation system, addressing functional, performance, and security aspects to deliver a robust and reliable product.

**2. Scope**

**2.1 In-scope**

Features to be tested.

* User Authentication
* User Registration
* User Login
* Filter Search
* Filter by Cuisine Type
* Sort by Rating and Popularity
* Order Processing System
* Payment Management
* Adding Payment Methods
* Making Payments
* Transaction History
* User Profile Management
* Restaurant Management
* Create/Update/Delete Restaurant Profile
* Manage Menu Items

* Handle Reviews and Ratings
* Ordering and Delivery
* Promotions and Events

**2.2 Out-of-Scope**

Features not to be tested.

**3.** **Testing Strategy**

**3.1 Test Objectives**

**User Registration and Authentication Testing:**

* Verify that users can successfully create accounts and log in using valid credentials.
* Ensure proper error handling for invalid login attempts and password resets.
* Test account security measures such as password encryption and session management.

**Restaurant Search and Discovery Testing:**

* Confirm that users can search for restaurants based on location, cuisine type, and other filters.
* Validate that search results are accurate and relevant to user queries.
* Test the responsiveness of the search feature under different network conditions.

**Menu Exploration and Customization Testing:**

* Verify that users can view detailed menus with descriptions, images, and prices.
* Test the ability to customize orders, add special instructions, and select delivery or pickup options.
* Ensure that menu items are displayed accurately and consistently across different devices.

**Order Placement and Payment Integration Testing:**

* Confirm that users can place orders seamlessly through the app.
* Test the integration with payment gateways to ensure secure and reliable transactions.
* Validate the handling of order confirmations, receipts, and email notifications.

**Admin Dashboard Testing:**

* Ensure that restaurant owners can access and manage their menus, orders, and promotions.
* Test the functionality of administrative tasks such as menu updates, order processing, and analytics.
* Validate access control measures to prevent unauthorized access to sensitive data.

**3.2 Test Assumptions:**

* All test team members will be present until the end of the project.
* Test and release managers will carry the project until the end.
* A stable internet connection will be available to users.

**3.3 Levels of testing:**

|  |  |  |
| --- | --- | --- |
| **Test Type** | **Description** | **Responsible Parties** |
| **Smoke Testing** | It is used to Determine if a new software build is ready for the next testing phase Smoke tests verify whether the most important features work as expected and that there are no showstopper issues in the build that can potentially lead to blocking the entire testing team. It helps in deciding if the build is flawed or not and hence, prevents the entire team from wasting time or resources. | **QA team** |
| **Functional Testing** | Functional Testing is a type of software testing that validates the software system against the functional requirements/specifications. The purpose of Functional tests is to test each function of the software application, by providing appropriate input, verifying the output against the Functional requirements.  [**Unit testing**](https://www.geeksforgeeks.org/unit-testing-software-testing/) is the type of functional testing technique where the individual units or modules of the application are tested. It ensures that each module is working correctly.  [**Integration testing**](https://www.geeksforgeeks.org/software-engineering-integration-testing/)**,** combined individual units are tested as a group and expose the faults in the interaction between the integrated units.  [**System testing**](https://www.geeksforgeeks.org/system-testing/) is a type of software testing that is performed on the complete integrated system to evaluate the compliance of the system with the corresponding requirements.  [**Usability testing**](https://www.geeksforgeeks.org/usability-testing/) is done to measure how easy and user-friendly a software application is.  [**User acceptance testing**](https://www.geeksforgeeks.org/acceptance-testing-software-testing/) is done by the client to certify that the system meets the requirements and works as intended. It is the final phase of testing before the product release.  [**Regression testing**](https://www.geeksforgeeks.org/software-engineering-regression-testing/)is done to make sure that the code changes should not affect the existing functionality and the features of the application. It concentrates on whether all parts are working or not. |  |
| **Non-Functional Testing** | Non-Functional testing is essential for confirming the software’s reliability and functionality. The [Software Requirements Specification (SRS)](https://www.browserstack.com/guide/software-requirement-specifications-in-agile) serves as the basis for this software testing method, which enables quality assurance teams to check if the system complies with user requirements. Increasing the product’s [usability](https://www.browserstack.com/guide/what-is-usability-testing), effectiveness, [maintainability](https://www.browserstack.com/guide/maintainability-testing), and portability is the goal of non-functional testing. It aids in lowering the manufacturing risk associated with the product’s non-functional components.  [Performance testing](https://www.browserstack.com/guide/performance-testing) eliminates the causes of the software’s sluggish and constrained performance. The software’s reading speed should be as quick as possible. One must create a well-organized and precise specification about the desired speed for Performance Testing. Otherwise, it won’t be evident if the test is a success or a failure. Example: When 1000 users use an application simultaneously, the load time shouldn’t exceed 5 seconds.  **Tools Used:** JMeter |  |

## 3.4 Functional Testing

Features to be tested.

* User Authentication
* User Registration
* User Login
* Filter Search
* Filter by Cuisine Type
* Sort by Rating and Popularity
* Order Processing System
* Payment Management
* Adding Payment Methods
* Making Payments
* Transaction History
* User Profile Management
* Restaurant Management
* Create/Update/Delete Restaurant Profile
* Manage Menu Items
* Handle Reviews and Ratings
* Ordering and Delivery
* Promotions and Events

Participants:

|  |  |  |
| --- | --- | --- |
| **Tester’s Name** | **Department/ Area** | **Role** |
| Shweta |  | Tes QAt TesterManager |
| Sushma |  | Tes QAt Aad |

## 

## 3.5 User Acceptance Testing

Features to be tested.

* User Authentication
* User Registration
* User Login
* Filter Search
* Filter by Cuisine Type
* Sort by Rating and Popularity
* Order Processing System
* Payment Management
* Adding Payment Methods
* Making Payments
* Transaction History
* User Profile Management
* Restaurant Management
* Create/Update/Delete Restaurant Profile
* Manage Menu Items

* Handle Reviews and Ratings
* Ordering and Delivery
* Promotions and Events

Participants:

|  |  |  |
| --- | --- | --- |
| **Tester’s Name** | **Department/ Area** | **Role** |
| Sushma |  | Test ManQAger |
| Shweta |  | Test LeadQA |

## 

## 3.6 Regression Testing

Participants:

|  |  |  |
| --- | --- | --- |
| **Tester’s Name** | **Department/ Area** | **Role** |
| Shweta |  | Test Man LQA ager |
| Sushma |  | Test Lead LQA |

**4 Execution strategy**

**4.1 Entry criteria**

|  |  |  |  |
| --- | --- | --- | --- |
| **Entry Criteria** | **Test Team** | **Technical Team** | **Notes** |
| *Test environment(s) is available* |  |  |  |
| *Test data is available* |  |  |  |
| *Code has been merged successfully* |  |  |  |
| *Development has completed unit testing* |  |  |  |
| *Test scripts are completed, reviewed, and approved by the Project Team* |  |  |  |

**4.2 Exit criteria**

|  |  |  |  |
| --- | --- | --- | --- |
| **Exit Criteria** | **Test Team** | **Technical Team** | **Notes** |
| *100% Test Scripts executed* |  |  |  |
| *90% pass rate of Test Scripts* |  |  |  |
| *No open Critical and High severity defects* |  |  |  |
| *All remaining defects are either cancelled or documented as Change Requests for a future release* |  |  |  |
| *All expected and actual results are captured and documented with the test script* |  |  |  |
| *All test metrics collected based on reports from daily and Weekly Status reports* |  |  |  |
| *All defects logged inDefect Tracker/Spreadsheet* |  |  |  |
| *Test environment cleanup completed and a new back up of the environment* |  |  |  |

**4.3 Validation and Defect Management**

* Defects will be logged using a defect tracking tool (e.g., Jira).
* Defect severity levels: Critical, Major, Minor.
* Defect priorities: High, Medium, Low.

|  |  |
| --- | --- |
| **Severity** | **Impact** |
| *1 (Critical)* | * *Functionality is blocked and no testing can proceed* * *Application/program/feature is unusable in the current state* |
| *2 (High)* | * *Functionality is not usable and there is no workaround, but testing can proceed* |
| *3 (Medium)* | * *Functionality issues but there is workaround for achieving the desired functionality* |
| *4 (Low)* | * *Unclear error message or cosmetic error which has minimum impact on product use.* |

# 

# **5. Environment Requirements:**

**5.1 Test Environments:**

* Operating system: Windows 10
* Browser: Google Chrome, Edge
* Hardware: Intel i5 processor, 8GB RAM.
* Software

**6. Test Deliverables:**

* Test Plan
* Test Cases
* Test scripts
* Defect Reports
* Test Execution reports
* Performance test Report

**7. Test Approach:**

* Manual testing will be used to test all the functionalities of the web application
* Automated testing will be used to test the performance and load of the web application.