*Manage Guru*

TEST DOCUMENTATION

VERSION HISTORY

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Version #** | **Implemented**  **By** | **Revision**  **Date** | **Approved**  **By** | **Approval**  **Date** | **Reason** |
| 1.0 | *N L RAMYA* | *16/11/17* | *<name>* | *<mm/dd/yy>* | Test Documentation draft |
| 2.0 | N L RAMYA | 19/11/17 |  |  | Based on review by team c5.added security testing |
|  |  |  |  |  |  |

TABLE OF CONTENTS

[**1**](#_1fob9te) **INTRODUCTION 3**

[**2**](#_3o7alnk) **TESTING MODEL 3**

[**3**](#_tyjcwt) **TESTING TYPES BEING USED 3**

[3.1](#_3dy6vkm) LIST ALL THE DIFFERENT TYPES of tests WHICH you plan to run 3

[3.2](#_3dy6vkm) LIST THE PHASES OF THE LIFECYCLE and the V&V done for   
 each phase in the project 3

[3.3](#_1t3h5sf) test adequacy criteria for your project 3

[**4**](#_4d34og8) **TEST CASES 3**

[4.1](#_2s8eyo1) Test case 1 3

[4.2](#_17dp8vu) Test case 2 3

[4.3](#_3rdcrjn) Test case 3 3

[4.4](#_26in1rg) Test case 4 4

[4.5](#_lnxbz9) Test case 5 4

[4.6](#_35nkun2) Test case 6 4

[4.7](#_1ksv4uv) Test case 7 4

[4.8](#_44sinio) Test case 8 4

[4.9](#_2jxsxqh) Test case 9 5

[4.10](#_z337ya) Test case 10 5

[**5**](#_3j2qqm3) **TEST RESULTS 5**

[TEST CASE # 5](#_1y810tw)

[OBJECTIVE/functionality 5](#_4i7ojhp)

[Test Kind 5](#_2xcytpi)

[TEST STATUS 5](#_1ci93xb)

[Who 5](#_3whwml4)

[When 5](#_2bn6wsx)

# INTRODUCTION

The Test Plan document documents specifies the strategy being followed for Testing of the project and sample of the test cases being written for the project

Manage guru is an application used for monitoring data that is generated from applications. We are dividing our design into 3 components and have test cases for each of them. The components are

1. Management System
2. Cloud system
3. Analytics system

We propose to use the demonstrative model to go about the testing phase because we want to make sure that the software runs and solves the problem. This model works well for us as we decide our test cases as we generate our system and there is no predefined test set. We would test the delay in real time update using this model.

We could use a destructive model as well at points to see when our application fails and figure out solutions to solve it.

# TESTING MODEL

The testing model that we would propose to use is the demonstrative model. We define

the test cases as we go about the implementation and fix them when they are

encountered. The model also makes sure that the system meets a certain reliability

requirement at the end of it.

# TESTING TYPES BEING USED

## LIST ALL THE DIFFERENT TYPES OF TESTS WHICH YOU PLAN TO RUN

Usability Testing - This type of testing is used to make sure that our application is easy to use and understand. It is to make sure that the user does not face difficulty in using our application.

Operational Testing - This type of testing is used to make sure that our application has functional components. We use this type of testing to check our components and to make sure that there is no data loss in case of loss of connectivity.

Performance Testing - This is done to make sure that we get our results from our cloud with minimum delay. We perform this type of testing to make sure that our data is real time.

Compatibility Testing - This type of testing is done to make sure that our application is compatible with all different Operating systems.

Security Testing - To make sure that our system is not vulnerable to unauthorized access.

**LIST THE PHASES OF THE LIFECYCLE AND THE V&V DONE FOR EACH PHASE IN THE PROJECT**

Requirements: The project was first discussed amongst all team members. Once this was discussed, we presented our idea to our guide where he introduced additional requirements. Once the idea was Validated by the professor, we had another meeting to discuss the feasibility of the project in the short period given. We also made a Software requirements specification document to specify all of our requirements in a formal fashion.

Design: The design was created in multiple steps. We first decided the UI components that we would require and designed our web application. Based on these components, we decided the data we would require to go about building our database. We then designed our analytics page. We wet about the design based on the domain and its requirements which we talked about in our design document.

Implementation: We created a web application to manage the restaurant's resources. This starts with the owner setting up an account for his business and then setting up accounts for all of his employees. Once this is done, the customers are required to log in based on the credentials decided by the owner.

We also have updation of orders, billing and inventory in this management app. Orders are taken by waiters and stored on the database. Once the customer leaves, a bill is generated. Inventory has 2 parts. The first part shows all the items in the existing inventory and the second is placing orders for new items.

All of this data is flushed to the azure cloud. We use franchise ID here to store the details of multiple outlets of the restaurant. We also have API’s to flush data to the cloud and retrieve data. We also show logs and reports of orders in real time to the owner who is running an application on his system. Lastly, we provide analytics for an easier understanding of the restaurants functioning.

Testing: We will be performing usability, operational, performance and compatibility testing. We will be using demonstrative and destructive testing in some places as well.

Evolution: We propose to add features to make the real time system more generic than to just add a franchise ID on cloud in the future.

## TEST ADEQUACY CRITERIA FOR YOUR PROJECT

1. Ease of usability to the customer.
2. Compatibility with multiple OS and browser versions
3. Security by login
4. No loss of data in situations such as loss of internet connectivity.

# TEST CASES

## TEST CASE 1

* Objective : Checking for orders to be updated realtime
* Prerequisites : Application for data entry and a database
* Procedure : Enter all order details from the waiter page and submit.
* Test Environment/Data to be used : Different types of data and invalid data such as entering a string for the table number.
* Expected Result : Will give an error saying that the datatype is wrong.

## TEST CASE 2

* Objective : Checking for inventory orders and inventory to be updated realtime.
* Prerequisites : Application for updating inventory and inventory orders and a a database
* Procedure to be followed: Enter all the details of new inventory orders / update existing inventory.
* Test Environment/Data to be used : Providing wrong data types to the database. Providing new category of food.
* Expected Result : Wrong data type will cause an error whereas a new category in inventory orders should not.

## TEST CASE 3

* Objective : Compatibility Testing: Application runs on all Operating systems.
* Prerequisites (if any): Require multiple OS systems and devices
* Procedure to be followed: Run application on different systems
* Test Environment/Data to be used: Testing to see if the application runs on different systems.
* Expected Result: It should be independent of the OS.

## TEST CASE 4

* Objective : UI Testing : Check if the application is simple to use
* Prerequisites (if any) : None
* Procedure to be followed : Use the web application to see if the flow is easy to understand and clear. Make sure that there are hyperlinks to access multiple pages.
* Test Environment/Data to be used : Accessing the web page in different ways
* Expected Result : Easy access to resources and pages.

## TEST CASE 5

* Objective: Making sure there is no loss of data when there is no internet connection.
* Prerequisites (if any) : Management front end to update order requests into a local database.
* Procedure to be followed: Maintain a queue of requests to keep track of all the requests that haven’t been sent to cloud yet.
* Test Environment/Data to be used: Check how the system performs when internet connection is back.
* Expected Result: Data will be updated in bulk through the queue when internet connection is back.

## TEST CASE 6

* Objective: Security testing : We run tests to make sure that users can’t login without a valid username and password.
* Prerequisites (if any): Have a login page where the user enters their credentials.
* Procedure to be followed: We make sure that the security requirements are met in the following ways:

1. User can’t login without a valid username and password.

2. Only an admin of the webpage can create accounts for the manager. The manager is in turn responsible for creating accounts for waiters and other staff. Hene no external user can create accounts.

3. Password is encrypted when it is sent via POST.

* Test Environment/Data to be used: Login can be done with a valid username and password.
* Expected Result: No unauthorised person can access our system

## TEST CASE 7

• Objective: Checking if analytics is reading data displaying results in real time from the cloud.  
 • Prerequisites (if any): Data that is uploaded in real time to the cloud  
 • Procedure to be followed: Update orders, bills or inventory on the management end and check if its updated on the analytics application  
 • Test Environment/Data to be used: Data is sent from the management application. Check if this is updated properly on analytics app.  
 • Expected Result: Updating happens in real time with the correct data

## TEST CASE 8

• Objective: Make sure that the waiter can’t enter incorrect information into the database.  
 • Prerequisites (if any): Application to enter data  
 • Procedure to be followed: Check if the table number is a positive number  
 • Test Environment/Data to be used: This is checked on the waiters page during entry  
 • Expected Result: Alerts an error if invalid.

## TEST CASE 9

* Objective/functionality being tested: Each worker should have only one page
* Prerequisites (if any): Web pages for each user.
* Procedure to be followed: When a person logs in, based on who he is, he is redirected to a page that is designed for his role.
* Test Environment/Data to be used: Make a waiter log in and a chef login. Users data is accessed to check the user type.
* Expected Result: Each of the types of users will be directed to different pages.

## TEST CASE 10

* Objective/functionality being tested: When a user logs out, he should not be present in the list of currently active users.
* Prerequisites (if any): Web page for viewing the details of online users
* Procedure to be followed:Check based on login time and logout time.
* Test Environment/Data to be used: Check if user is online when he logs out with another system.
* Expected Result: Logged out user will not be in the online list

# TEST RESULTS

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Case num** | **Objective** | **kind** | **status** | **who** | **when** |
| **1** | **Check if orders / inventory are updated real time to cloud** | **operational testing** | **completed** | **ramya priyanka** |  |
| **2** | **Check if app runs on all systems** | **compatibility testing** | **completed** | **nachiketa** |  |
| **3** | **check if app is simple to use** | **usability testing** | **completed** | **ramya** |  |
| **4** | **security by login** | **functionality testing** | **completed** | **hariharan** |  |
| **5** | **check if analytics is real time** | **PERFORMANCE TESTING** | **completed** | **nachiketa**  **nirmit** |  |
| **6** | **CHECK FOR INCORRECT ENTRIES** | **FUNCTIONALITY TESTING** | **COMPLETED** | **hariharan** |  |
| **7** | **make sure there is no data loss** | **operational testing** | **in process** | **hariharan** |  |