Office of Systems Integration
"SERVING CALIFORNIA"

Restaurant Management System

Test and Evaluation Master Plan

<Insert Project Logo here>

November, 2021

Health and Human Services Agency, Office of Systems Integration

Revision History

Revision History			
Revision/WorkSite #	Date of Release	Owner	Summary of Changes
OSI Admin #7343	03/26/2009	OSI - PMO	Initial Release

Remove template revision history and insert the Test and Evaluation Master Plan revision history.

Approvals

Name	Role	Date

Insert Project Approvals here.

Template Instructions:

This template offers instructions, sample language, boilerplate language, and hyperlinks written in 12-point Arial font and distinguished by color, brackets, and italics as shown below:

- Instructions for using this template are written in purple-bracketed text and describe how to complete this document. Delete instructions from the final version of this plan.
- *Sample language is written in red italic font and may be used, or modified, for completing sections of the plan. All red text should be replaced with project-specific information and the font changed to non-italicized black.*
- Standard boilerplate language has been developed for this plan. This standard language is written in black font and may be modified with permission from the OSI Project Management Office (PMO). Additional information may be added to the boilerplate language sections at the discretion of the project without PMO review.
- Hyperlinks are written in blue underlined text. To return to the original document after accessing a hyperlink, click on the back arrow in your browser's toolbar. The "File Download" dialog box will open. Click on "Open" to return to this document.

Table of Contents

1. INTRODUCTION

The Restaurant Management System Test and Evaluation Master Plan (TEMP) identifies the tasks and activities to be performed so that all aspects of the system are adequately tested and that the system can be successfully implemented. The TEMP documents the scope, content, methodology, sequence, management of, and responsibilities for test activities.

1.1 Purpose

This document describes the TEMP for the Restaurant Management System Project. The purpose of the TEMP is to describe the methods that will be used to test and evaluate the Restaurant Management system.

Project purpose is to develop an online food ordering system for the restaurant and my teammates will be contributing to this project. The end goal is for the user to order food using our site comfortably and bills should be generated within our system.

Given below are the major functions that Restaurant Management system uses:

- Allow customers to scroll through the menu and select dishes they want
- Allows the Customers to edit the order
- Allows Customer to create an account
- Manage their account
- Login to the system
- Review their order
- Provide payment details
- Receive confirmation in the form of an order number
- View placed order

Tests to be performed are:

- Unit testing
- Functional testing
- System testing
- Interface testing
- Performance/Stress Testing
- Regression Testing
- User Acceptance Testing
- Pilot Testing

1.2 Scope

In the testing phase we are going to test the functionality of the system by designing and testing test cases, which is going to be performed in a scheduled manner. So first of all, we are supposed to check the authentication response of the system when the customer tries to sign-up and login. Whether the customer is able to sign-up properly? Whether the flash messages are displayed when the customer is not filling all columns or filling details in unspecified format i.e., not a proper mail-id format? The same response and idea for this document apply for all the functionality of the system so that it can be rectified as per the plan of this test plan document and WBS.

1.3 References

Sources referenced below should be used as references.

- IEEE STD 1012-2004, Standard for Software Verification and Validation, Table 1, Section 5.4.5 within table (the tables appear prior to the annexes)
- IEEE Standard 1008-1987, Standard for Software Unit Testing;
- IEEE Standard 829-1998, Standard for Software Test Documentation;
- IEEE 1062-1998, Checklist A.7 -- Supplier Performance Standards / Acceptance Criteria; and
- IEEE 1062-1998, Checklist A.10 -- Software Evaluation.

1.3.1 Project WorkSite Repository

List the document name and WorkSite reference number for any documents that can be references for this document.

1.4 Glossary and Acronyms

BPWeb	OSI Best Practices Website http://www.bestpractices.osi.ca.gov
Consultant	A company or consultant who is providing services or products to support the project.
Deliverable	Any tangible work (report, briefing, manual) produced by a project contractor, and required by the contractor's contract/Statement of Work to be provided to the state.
CM	Contract Manager
FM	Functional Manager
IT	Information Technology
IV&V	Independent Verification and Validation
OSI	Office of Systems Integration
PD	Project Director
PM	Project Manager

PMO	Project Management Office
QM	Quality Manager
RFP	Request for Proposals
SOW	Statement of Work
SUT	System under Test

1.5 Document Maintenance

This document will be reviewed and updated as needed, as the project proceeds through each phase of the system development life cycle.

This document contains a revision history log. When changes occur, the document's revision history log will reflect an updated version number as well as the date, the owner making the change, and change description will be recorded in the revision history log of the document.

2. Participants Roles and Responsibilities in Test and Evaluation

This section describes the roles and responsibilities of the Restaurant Management System staff with regard to Test and Evaluation.

In the test designing and test validation the following members are working:

- UDITI GUPTA 20BCE1445
- KHUSHI MATTU 20BCE1189

All appropriate project staff will be trained on their responsibilities by their manager/lead when they join the project. Project meetings are used to brief staff on any changes to the process.

2.1 Project Director

Uditi Gupta and Khushi Mattu are responsible for the final decision on all Test and Evaluation issues and in case of any dispute, their decision is final and binding.

2.2 Project Manager

Khushi Mattu is the Project Manager (PM)and is responsible for confirming Test and Evaluation results.

2.3 Test Manager

Uditi Gupta is the Test Manager (TM) is responsible for overseeing the Test and Evaluation process, including creating test plans, execution, review, and coordinating acceptance.

2.4 Test Team

Khushi and Uditi are responsible for testing or assisting with testing of the Prime Contractor's system.

2.5 Quality Manager

Khushi and Uditi are the Quality Manager (QM) and their role is to monitors the prime contractor's testing efforts and participates in any reviews.

2.6 Independent Verification and Validation

N/A

3. Test Strategy and Method

By applying correct order of test strategies and method we are going to make sure that the application / website behaves and performs exactly as it was fashioned and expected. These testing strategies and methodologies included end to end testing, system and unit testing – all aimed at ensuring top quality. It is a wise business decision to adopt the paths, which will be helpful in using all the testing methods to ensure our software is the without any errors. It also includes White box and black box testing techniques. Functional and non-functional testing using the right strategies and methodologies helps to build a robust and sustainable testing framework.

3.1 Unit Testing

Unit Testing is the first level of testing which is done by developers and also called component testing.Here, a section of code would be isolated and the correctness of the code would be verified. Every function and procedure would be tested. In this way, we will be able to identify bugs at an early development stage.

There are two types- Blackbox testing and whitebox testing.

Whitebox Testing, also known as Glassbox Testing will test the internal structure, design and coding.It will verify the broken paths, flow of inputs, expected outputs and the conditional loops.It will involve testing a series of predefined inputs against expected or desired outputs so that when a specific input does not result in the expected output, it means we have encountered a bug. First we need to understand the source code, then we need to create test cases which have to be followed by execution. We can optimize our code by finding the hidden errors.

Blackbox Testing, also known as functional testing is efficient for large segment of codes. a black box test evaluates all relevant subsystems, including UI/UX, web server or application server, database, dependencies, and integrated systems. First, the requirements and specifications of the system are identified.Tester chooses valid inputs (positive test scenario) to check whether SUT processes them correctly. Also, some invalid inputs (negative test scenario) are chosen to verify that the SUT is able to detect them. Tester determines expected outputs for all those inputs. Software tester constructs test cases with the selected inputs.The test cases are executed.Software tester compares the actual outputs with the expected outputs. Then we can identify if there is a bug.

Statistical Testing makes use of statistical methods to determine the reliability of the program. Statistical testing focuses on how faulty programs can affect its operating conditions.Software is tested with the test data that statistically models the working environment.Failures are collated and analyzed.

From the computed data, an estimate of program's failure rate is calculated.A Statistical method for testing the possible paths is computed by building an algebraic function.

Statistical testing is a bootless activity as the intent is NOT to find defects.

The level of security for test facilities, system software, and proprietary components such as software, data, and hardware should be high.

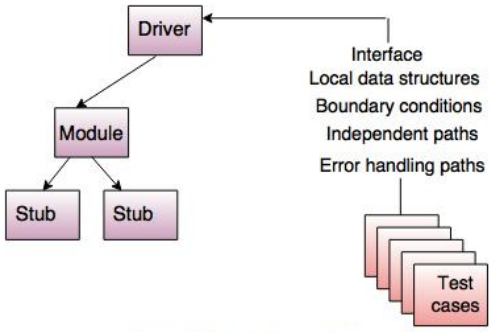


Fig. - Unit test environment

Source: Google

3.2 Functional Testing

Functional testing validates the software system against the functional requirements/specifications.Here, we test each function of the software application, by providing appropriate input, verifying the output against the Functional requirements.Functional testing mainly involves black box testing and it is not concerned about the source code of the application. This testing checks User Interface, APIs, Database, Security, Client/Server communication and other functionality of the Application Under Test. The testing can be done either manually or using automation.Understand the Functional RequirementsFirst, we have to Identify the test inputs or test data based on requirements. then, we have to Compute the expected outcomes with selected test input values. then, Execute test cases. finally, Compare actual and computed expected results

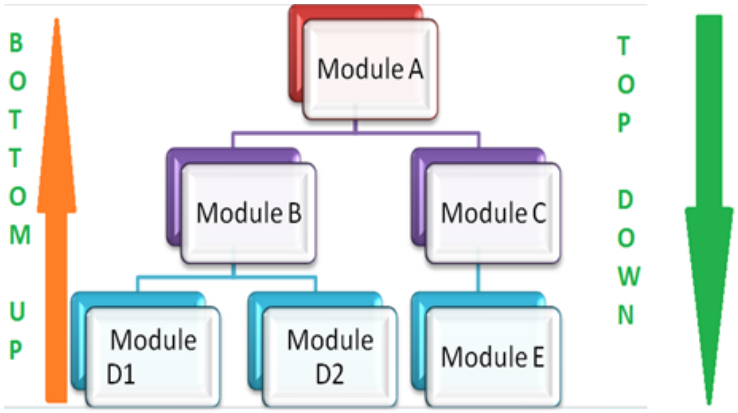
3.3 Integration Testing

Integration testing is the process of testing the interface between two software units or module. It's focus on determining the correctness of the interface. The purpose of the integration testing is to expose faults in the interaction between integrated units. Once all the modules have been unit tested, integration testing

is performed. it is usually done by testers. Integration testing is of two types: top down and bottoms up approach.

In bottom-up testing, each module at lower levels is tested with higher modules until all modules are tested. The primary purpose of this integration testing is, each subsystem is to test the interfaces among various modules making up the subsystem. This integration testing uses test drivers to drive and pass appropriate data to the lower level modules.

Top-down integration testing technique used in order to simulate the behaviour of the lower-level modules that are not yet integrated.In this integration testing, testing takes place from top to bottom. First high-level modules are tested and then low-level modules and finally integrating the low-level modules to a high level to ensure the system is working as intended.



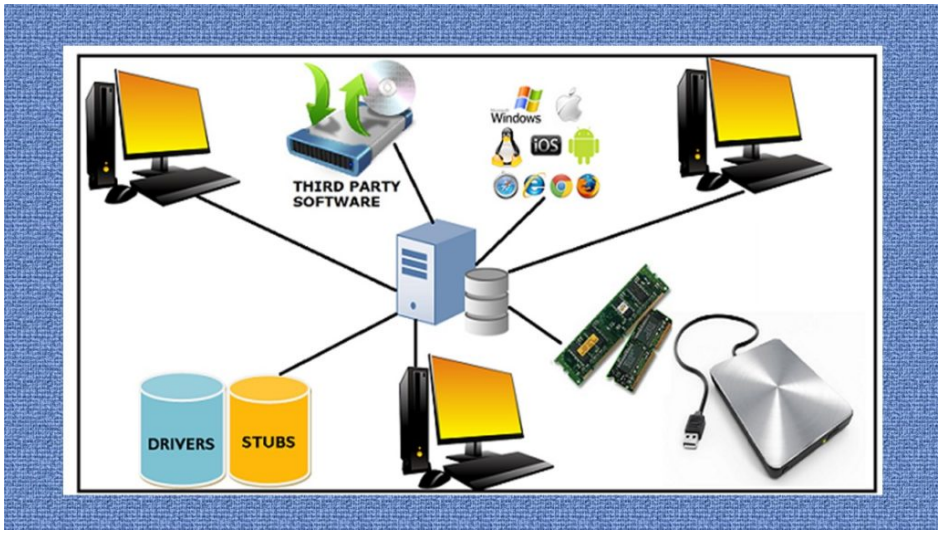
3.4 System Testing

System Testing will be performed on a complete integrated system to evaluate the compliance of the system with the corresponding requirements.System testing will detect defects within both the integrated units and the whole system. The result of system testing will be the observed behavior of a component or a system when it is tested. To test the system as a whole, requirements and expectations should be clear and the tester needs to understand the real-time usage of the application too.

Also, most used third-party tools, versions of OSes, flavors and architecture of OSes can affect the system's functionality, performance, security, recoverability or installability.

Therefore, while testing the system a clear picture of how the application is going to be used and what kind of issues it can face in real-time can be helpful

First, we need to create a testing environment for better quality testing. Then, we need to generate test cases for the testing process. After the generation of the test case and the test data, test cases are executed. then we will be able to identify the defects in the system. It will help us test the side effects of the testing process.If the test is not successful then again the test is performed.



Source: Google

3.5 Interface Testing

Interface Testing will verify whether the communication between two different software systems is done correctly. Interface Testing is performed to evaluate whether systems or components pass data and control correctly to one another. It is to verify if all the interactions between these modules are working properly and errors are handled properly.

We start with the development configuration of the interface. After the configuration of the interface and the development initialization, the configuration is needed to be verified as per the requirement. In simple words, verification takes place.

After the configuration and development stage, validation of the interface is necessary.After the completion of the project, when the project reaches it's working stage, the interface is set to be monitored for its performance. One thing that has to be kept in mind is that the code should be defect-free, to interrogate this, tests have to be conducted to verify that any added code doesn't bring defects with it.After the

accomplishment of the work, validation of data and workflow takes place. So, it's important to maintain the authenticity of the program.

3.6 Performance/Stress Testing

Stress Testing verifies stability & reliability of software application. The goal of Stress testing is measuring software on its robustness and error handling capabilities under extremely heavy load conditions and ensuring that software doesn't crash under crunch situations. It even tests beyond normal operating points and evaluates how software works under extreme conditions.

First, we have to gather the system data, analyze the system, define the stress test goals.Then, we have to create the Stress testing automation scripts, generate the test data for the stress scenarios.Then,we run the Stress testing automation scripts and store the stress results followed by analyzing the Stress Test results and identifying bottlenecks.Lastly we have to fine-tune the system, change configurations, optimize the code with goal meet the desired benchmark.
Commonly used metrics are –

Measuring Scalability & Performance

- Pages per Second: Measures how many pages have been requested / Second
- Throughput: Basic Metric – Response data size/Second
- Rounds: Number of times test scenarios have been planned Versus Number of times a client has executed

Application Response

- Hit time: Average time to retrieve an image or a page
- Time to the first byte: Time is taken to return the first byte of data or information
- Page Time: Time is taken to retrieve all the information in a page

Failures

- Failed Connections: Number of failed connections refused by the client (Weak Signal)
- Failed Rounds: Number of rounds it gets failed
- Failed Hits: Number of failed attempts done by the system (Broken links or unseen images)

3.7 Regression Testing

Regression testing helps to confirm that a recent program or code change has not adversely affected existing features.Regression Testing is full or partial selection of already executed test cases which are re-executed to ensure existing functionalities work fine.
In order to do Regression Testing process, we need to first debug the code to identify the bugs. Once the bugs are identified, required changes are made to fix it, then the regression testing is done by selecting relevant test cases from the test suite that covers both modified and affected parts of the code.

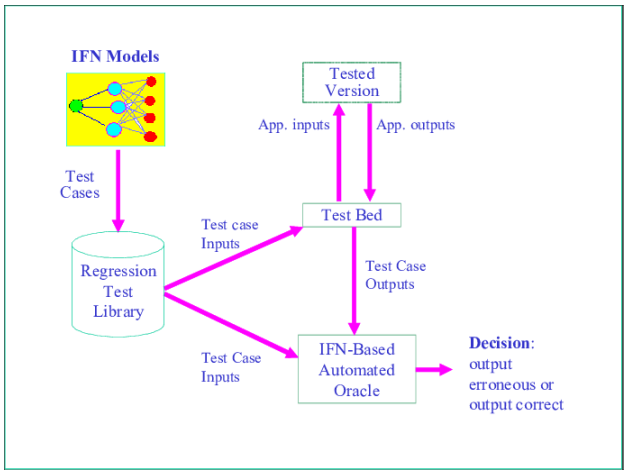
Software maintenance is an activity which includes enhancements, error corrections, optimization and deletion of existing features. These modifications may cause the system to work incorrectly. Therefore, Regression Testing becomes necessary.

Retest All

- This is one of the methods for Regression Testing in which all the tests in the existing test bucket or suite should be re-executed. This is very expensive as it requires huge time and resources.

Effective Regression Tests can be done by selecting the following test cases-

- Test cases which have frequent defects
- Functionalities which are more visible to the users
- Test cases which verify core features of the product
- Test cases of Functionalities which has undergone more and recent changes
- All Integration Test Cases
- All Complex Test Cases
- Boundary value test cases
- A sample of Successful test cases
- A sample of Failure test cases



Source: Google

3.8 User Acceptance Testing

User Acceptance Testing (UAT) is a type of testing performed by the end user or the client to verify/accept the software system before moving the software application to the production environment. UAT is done in the final phase of testing after functional, integration and system testing is done.The main Purpose of UAT is to validate end to end business flow. It does not focus on cosmetic errors, spelling mistakes or system testing. User Acceptance Testing is carried out in a separate testing environment with production-like data setup. It is kind of black box testing where two or more end-users will be involved. Client and users perform user acceptance testing.

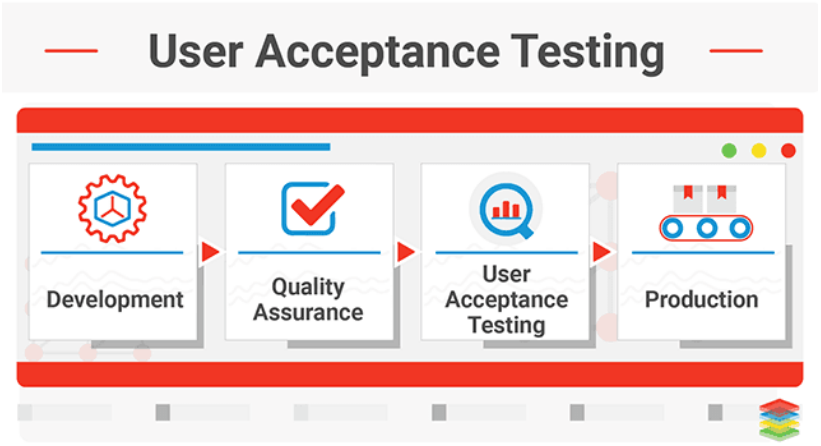
First, we analyze Business Requirements followed by creation of UAT test plan. We then identify the Test Scenarios and create UAT Test Cases which is followed by preparation of Test Data(Production like Data).Then the Test cases are run and results are recorded. Thus, we can confirm business objectives.

User Acceptance Testing is carried out in a separate testing environment with production-like data setup. So, in this testing we make sure that the software/system is ready for final production and deployment.

We would use 2 techniques:

1) ALPHA -feedback from employees

2) BETA -feedback from limited range of customers/end users



Source: Google

3.9 Pilot Testing

Pilot Testing is defined as a type of Software Testing that verifies a component of the system or the entire system under a real-time operating condition. The purpose of the Pilot Test is to evaluate the feasibility, time, cost, risk, and performance of a research project.
This testing is done exactly between the UAT and Production.
In Pilot testing, a selected group of end users try the system under test and provide the feedback before the full deployment of the system.
In other words, it means to conduct a dress rehearsal for the usability test that follows.
Pilot Testing helps in early detection of bugs in the System.

This testing verifies a component of the system or the entire system under a real time operating condition. The purpose of the pilot Testing is to evaluate the feasibility, time, cost, risk and performance of a research project.

Main idea behind pilot testing:

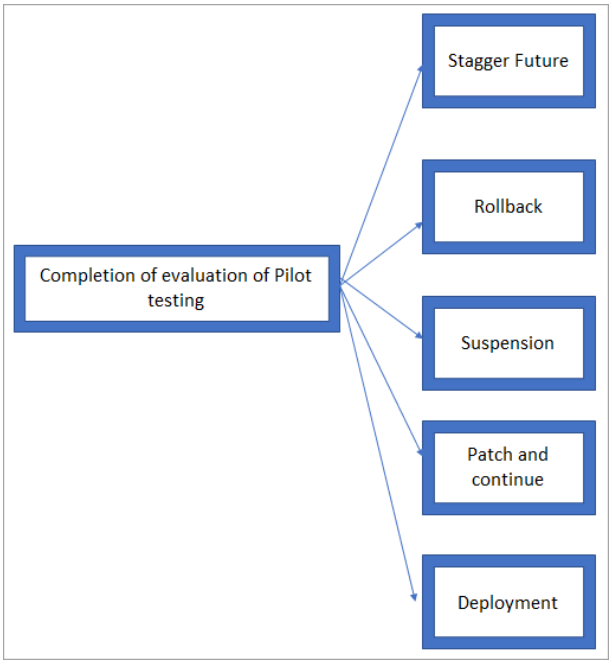
1) To evaluate the feasibility, cost and other attributes.

2) To better utilize time and resources.

3) To find end users reaction towards the software.

4) To find whether software is successful or not.

5) To provide another chance for development team.



Source: Google

4. Evaluation Criteria

The criterion of relevance is used to assess whether the project fulfils an important function from a development perspective ("priority"), and whether its design was fundamentally suited to achieving the goals associated with the project ("validity of the results chain"). This means that an assessment is made of whether the project appropriately addressed an important development goal, took into account the strategic requirements of the partner country and the German Federal Government, and was coordinated with other donors.
Relevance will be the best approach for evaluating the test. As it the goal oriented , it will help our project to achieve its goal of ordering food with the help of our site.

In all our testing methods, we are going to keep the evaluation criteria on the basis of Boolean values i.e., TRUE/PASS or FALSE/FAIL.

5. Incident Management

Incidents are identified through user reports, solution analyses, or manual identification. Once identified, the incident is logged and investigation and categorization can begin. Categorization is important to determining how incidents should be handled and for prioritizing response resources.Once incident tasks are assigned, staff can begin investigating the type, cause, and possible solutions for an incident. After an incident is diagnosed, you can determine the appropriate remediation steps. This includes notifying any relevant staff, customers, or authorities about the incident and any expected disruption of services.

In our project, incidents are identified using manual identification and after proper incident diagnosis, we will determine the appropriate remediation steps. If any errors occur during the testing phase, then it will be given a unique incident tracking log and our team will take care of it on the basis of the priority tag given to it. So, if the test case fails then it will be redirected to the developer of that component to rectify the code and the component will follow the same procedures of testing again.

6. Requirements Traceability Matrix

Business requirement	Technical requirement	Test case id
Login	If the user password and id are valid. LOGIN	1-10
Product order	If all valid as per requirements. Confirm	11-20
Bill	If the cost of all the items is correct. CONFIRM	21-30
Admin profile	if addition of all menu items and prices are correct. CONFIRM	31-40

7. Review and Approval Process

1.The team makes initial proposals.

2. The manager assigns tasks.

3. The team receives the tasks.

4. The team submits initial drafts.

5. The team requests changes.

6. The team submits the final version.

7. The team brings the elements together.

8. The submission is approved or rejected.

9. The project is completed.

At the end of each test stage, a Test Summary Report is created, reviewed and approved. Team will be involved in the reviews and the manager has the authority to approve, halt, or resume testing for each stage of testing to be performed on the project. Once the Test Summary Report is approved, authority to move to the next testing stage is allowed.

8. Test Resources

Xampp should be in running state before running any of the tests and MySQL configuration with Xampp should be Cleary defined with username and password.

9. Test Schedule

All activities will be done on 12-11-21.

10. Test Plan Template

Refer to excel

11. Test Case Report

Refer to excel

12. Test Log

Refer to excel

13. Incident Tracking Log

The Incident Tracking is used in conjunction with Test Case Report to create a chronological record about the actual execution of tests. The results of the tests provide detailed evidence for incident reporting and enable reconstruction of testing as needed.

The Incident Tracking Log is a component of a larger group of project system test documents. The Incident Tracking Log should be utilized for all testing stages to capture the details of unsuccessful test cases. This log documents the case and incident information.

If any errors occur during testing phase, then it will be given a unique incident tracking log and our team will take care of it on the basis of the priority tag given to it. So, if the test case fails then it will be redirected to the developer of that component to rectify the code and the component will follow the same procedures of testing again.

14. Corrective Action Plan (CAP):

This plan will address the approach of how the incident will be corrected and in which test stage each requirement/deficiency will be completed. if the test case fails then it will be redirected to the developer of that component to rectify the code and the component will follow the same procedures of testing again

The Corrective Action Plan (CAP) is used to capture any requirements and/or deficiencies that were not completed successfully during a testing stage. This plan will address the approach of how the incident will be corrected and in which test stage each requirement/deficiency will be completed. Each incident included in the tracking log will be included in the CAP for documentation purposes.

15. Test Summary Report

The Test Summary Report derives its information from the results of the testing effort and should summarize all testing activities. The Test Summary Report will provide a formal reporting mechanism for approval related to the specific testing stage. The Test Summary Report is a component of a larger group of project system test documents. The Test Summary Report should be utilized for all testing stages and requires approval before the next testing stage or implementation processes can begin.

16. AppendicesTest Plan

GENERAL INFORMATION					
Test Stage:	☐Unit ☐Performance	☐Functionality ☐Regression	☐Integration ☐Acceptance	☐System ☐Pilot	☐Interface
	Specify the testing stage for this plan.				

Test Case Number	Test Description	Requirement Fulfilled	Expect
TCC001	To login successfully valid email and password should be there.		Should be redire
TCC002	Invalid email and password should not be entered		Should be not re Homepage
TCH001	About Us Section should redirect us to the footer of the page		Should be redire Homepage
TCH002	Order Button should redirect us to the Menu		Should be redire
TCA001	Dish Name should be one of the dishes in the menu		Should be able t
TCA002	View order history should redirect us to the bill		Should be redire

Note: Use the information in this plan to define responsibilities, schedules, and resource requirements and to create test procedures.

APPENDIX A : TEST CASE REPORT

(Use one template for each test case)

GENERAL INFORMATION			
Test Stage:	☐Unit ☐Functionality ☐Integration ☐System ☐Interface ☐Performance ☐Regression ☐Acceptance ☐Pilot Specify the testing stage for this test case.		
Test Date:	mm/dd/yy	**System Date, if applicable:**	mm/dd/yy
Tester:	Specify the name(s) of who is testing this case scenario.	**Test Case Number:**	Specify a unique test number assigned to the test case.
Test Case Description:	Provide a brief description of what functionality the case will test.		
Results:	☐Pass ☐Fail	**Incident Number, if applicable:**	Specify the unique identifier assigned to the incident.
INTRODUCTION			
Requirement(s) to be tested:	Identify the requirements to be tested and include the requirement number and description from the Requirements Traceability Matrix.		
Roles and Responsibilities:	Describe each project team member and stakeholder involved in the test, and identify their associated responsibility for ensuring the test is executed appropriately.		
Set Up Procedures:	Describe the sequence of actions necessary to prepare for execution of the test.		
Stop Procedures:	Describe the sequence of actions necessary to terminate the test.		
ENVIRONMENTAL NEEDS			

Hardware:	Identify the qualities and configurations of the hardware required to execute the test case.
Software:	Identify system and application software required to execute the test case. Specify any software that the test case will interact with.
Procedural Requirements:	Describe any constraints on the test procedures necessary to execute the test case.
TEST	
Test Items and Features:	Identify and describe the items and features that will be exercised by the test case. Group the test cases into logically related scenarios that test related items and features. For each item or feature, a reference to its associated requirement source should be included.
Input Specifications:	Define each input required to execute the test case, and reference any required relationships between inputs.
Procedural Steps:	Describe the sequences of actions necessary to prepare and execute the test case. Provide detailed test procedures for each test case; explain precisely how each test case will be executed.
Expected Results of Case:	Describe the outcome anticipated from the test case. Specify the criteria to be used to determine whether the item has passed or failed.
ACTUAL RESULTS	
Output Specifications:	Define all of the outputs and features required of the test case and provide expected values. While executing the test, record and describe the visually observable outputs as they occur. Produce tangible evidence of the output such as a screen print. At the conclusion, describe the actual outcome. Indicate whether the test passed or failed, and identify any discrepancies between the expected results and the actual results.

APPENDIX B : TEST LOG

(Use this template to log all test cases and results)

Requirement Number Tested	Test Case Number	Test Case Description	Date Tested	Test Stage Tested	Pass (P)	Fail (F)
Use the requirement number included in the Requirements Traceability Matrix	TCC001	To login successfully valid email and password should be there.	12/11/21	Unit, Functional, Integration, System, Interface, Performance, Regression, Acceptance, Pilot	P	F
	TCC002	Invalid email and password should not be entered	12/11/21			
	TCH001	About Us Section should redirect us to the footer of the page	12/11/21			
	TCH002	Order Button should redirect us to the Menu	12/11/21			
	TCA001	Dish Name should be one of the dishes in the menu	12/11/21			
	TCA002	View order history should redirect us to the bill	12/11/21			
				TOTAL (Pass/Fail)	0	0

Appendix C : Incident Tracking Log

(Use one template for each failed test case)

GENERAL INFORMATION			
Test Stage:	☐Unit ☐Functionality ☐Integration ☐System ☐Interface ☐Performance ☐Regression ☐Acceptance ☐Pilot Specify the testing stage for this test case.		
Tester:	Specify the name(s) of who tested this case scenario.	**Test Case Number:**	Specify a unique test number assigned to the test case.
Test Description:	Provide information that applies to the log entries and identify items being tested, such as versions or revision levels. Describe environmental attributes in which the testing is being conducted.		
Incident Number:	Specify the unique identifier assigned to the incident.		
ACTIVITIES AND EVENTS			
Environmental Information:	Record any environmental conditions that were specific for the test execution.		
Unusual Events:	Record details of what happened before and after the occurrence of unexpected events and document conditions surrounding the test.		
INCIDENT TRACKING			
Summary of Incident:	Provide a summary of the incident, including any relevant information to the incident.		
Inputs:	Describe the exact sequences of actions taken when executing the test case that resulted in the incident.		
Expected Results:	Describe the expected results of the test case.		

Actual Results:	Describe the actual results of the test case.
Abnormalities:	Describe any abnormalities of the test results and how the actual results differed from the expect results.
Date and Time of Incident:	Record the data and time of the incident.
Procedural Step:	Describe the procedural step of the test case where the incident occurred.
Environment:	Describe the environmental factors that existed when the test case was executed.
Attempts to Repeat:	Describe the number of attempts to repeat execution of the test and note if the incident occurred consistently or intermittently across attempts.
Impact:	Describe the impact this test incident will have on other testing activities.

APPENDIX D : CORRECTIVE ACTION PLAN (CAP)

(Use one template for each incident)

GENERAL INFORMATION					
Test Stage:	☐Unit ☐Functionality ☐Integration ☐System ☐Interface ☐Performance ☐Regression ☐Acceptance ☐Pilot Specify the testing stage where the requirement was not met or the deficiency occurred.				
Incident Number:	Specify the unique identifier assigned to the incident.	**Test Case Number:**	Specify the unique test case number.		
Test Description:	Describe the items being tested, such as versions or revision levels. Describe environmental attributes in which the testing is being conducted.				
Requirement not met:	Describe the requirement and/or deficiency that were not successfully completed during a testing stage. Identify the requirement number and details from the Requirements Traceability Matrix.				
Results of Incident Correction Re-Test:	☐Pass ☐Fail	**Re-Test Case Number, if applicable:**	Specify the unique test case number.	**Date of Incident Correction Re-Test:**	mm/dd/yy
INCIDENT					
How was the Incident Identified:	Describe how the incident was indentified during the test stage.				
Incident Description:	Describe the incident, in detail.				
APPROACH TO COMPLIANCE					
Corrective Action:	Describe the necessary steps to correct the incident.				

Roles and Responsibilities:	Describe each project team member and stakeholder involved in the correction and re-test, and identify their associated responsibilities for ensuring the test is executed appropriately.
Interim Activities (until compliance is reached):	Describe any processes or procedures that need to be followed until the incident is corrected.
Approval of Approach:	Describe the approval process required for the correction of the incident.
SCHEDULE FOR COMPLIANCE	
Major Milestones:	Provide the milestones and dates for the correction of the incident.
Dependencies:	List any dependencies to other tests, tasks, projects, etc.
COST OF COMPLIANCE	
Time:	List any additional cost in time due to the correction of the incident.
Resources:	List any additional cost in resources due to the correction of the incident.
Fiscal Impact:	List any additional fiscal impact due to the correction of the incident.
CONCLUSION AND NEXT STEPS	
Checkpoints:	Designate checkpoints to ensure the correction of the incident.
Re-assessment of Incident:	Describe the steps required to re-assess the incident once it has been corrected and re-tested. If the re-test is unsuccessful, include how this open incident will be handled. If the incident cannot be corrected, describe how it will be dealt with. Include any timeframes, if appropriate.
Approvals and Certification of Compliance:	Describe the approval required for the re-assessment/certification of compliance once the incident has been corrected and re-tested successfully.

Formal Review of Status:	Describe the steps necessary for a formal status review of the correction of the incident.

APPENDIX E : TEST SUMMARY REPORT

(Complete one template for each test stage)

GENERAL INFORMATION	
Test Stage:	☐Unit ☐Functionality ☐Integration ☐System ☐Interface ☐Performance ☐Regression ☐Acceptance ☐Pilot Specify the testing stage for this Test Summary Report.
Summary Review Date:	mm/dd/yy
TEST SUMMARY	
Test Summary:	Identify the items tested, any relevant version/revision level and summarize the evaluation of the test items. Reference any pertinent system test documents such as the test case, test log, and incidents.
Variances:	Report observed variances of the test items from the test cases. Specify known reasons for each variance found.
Assessment:	Report observations on the breadth of the testing process based upon the system test documents, test plan, and test cases.
Summary of Results:	Summarize the overall results of the testing process. Identify all anomalies or incidents found during testing, discuss incident resolutions, and any unresolved incidents.
Evaluation:	Provide an overall evaluation based upon the test results and the number of incidents resolved or unresolved. Summarize the testing activities and the next testing steps.
Corrective Action Plan (CAP):	Identify any unresolved requirements or incidents not completed with this phase of testing and include them in the CAP.
APPROVALS	
<Signature>:	Identify the person and title that must authorize and sign off for this testing stage. Once all signatures are obtained, the next testing stage/implementation may begin.

<Signature>:	Identify the person and title that must authorize and sign off for this testing stage. Once all signatures are obtained, the next testing stage/implementation may begin.
<Signature>:	Identify the person and title that must authorize and sign off for this testing stage. Once all signatures are obtained, the next testing stage/implementation may begin.