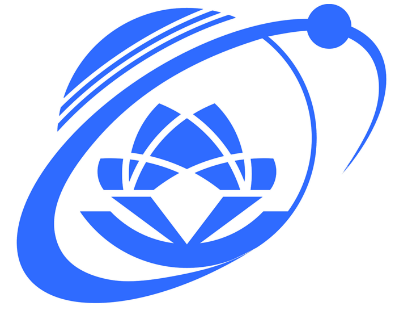
**VIETNAM NATIONAL UNIVERSITY HO CHI MINH CITY**

**UNIVERSITY OF INFORMATION TECHNOLOGY**

**FACULTY OF SOFTWARE ENGINEERING**

**------------------------------**

****

SMALL RESTAURANT MANAGEMENT SOFTWARE

Test Plan Document

Project Code:

Document Code:

Instructors :

**Nguyen Trinh Dong**

Team members :

* **Nguyen Anh Tuan Ngoc 21522379**
* **Nguyen Thi Thuy Trinh 21522719**

**TP. HỒ CHÍ MINH - 2023**

TABLE OF CONTENTS

1 Introduction 3

2 Business Background 3

3 Test Objectives 3

4 Scope 3

5 Test types Identified 3

6 Problems Perceived 3

7 Architecture 3

8 Environment 3

9 Assumptions 3

10 Functionality 3

11 Security 4

12 Performance 4

13 Usability 5

14 Test Team Organization 6

15 Schedule 6

16 Defects Classification Mechanism 6

17 Configuration Management 6

18 Release Criteria 6

Record of change

\*A - Added M - Modified D - Deleted

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Effective Date | Changed Items | A\* M, D | Change Description | New Version |
| 26-November-2023 | Basic Test Plan | A | New document | 1.0 |
| 30-November  -2023 | Final Test Plan | M | Modify document | 2.0 |

SIGNATURE PAGE

**ORIGINATOR:** Trinh Dong Nguyen 18-Jun-2023

Tester

**REVIEWERS:** Trinh Dong Nguyen 19-Jun-2023

Test Leader

**APPROVAL:** Trinh Dong Nguyen 19-Jun-2023

Project Leader

# Introduction

1.1 Purpose

This document encompasses a comprehensive description of all test scenarios executed within the context of the project. It provides in-depth scenarios for all elements slated for testing, as outlined in the test plan (please consult the document labeled ... for additional details). All user interfaces will undergo testing in accordance with the User Interface Test Case Specification, while other components will be evaluated using the respective test case specifications provided below.

* NOTE: Due to the incompletion of the detailed design, some parts of pages are not clarified such as static textual contents, links, and images. Thus most of the test cases specified in this document are not completed. Only functional test cases are completed

1.2 General information

This test plan is to define the scope, objectives, approach, and resources required for the testing of the small restaurant management software. This plan will be used to guide the testing activities throughout the software development and deployment process.

* This test plan consists of the following sections:

1. Introduction

Provides an overview of the test plan.

1. Business Background

Describes the context of the restaurant management software.

1. Test Objectives

Outlines the specific objectives and goals of the testing process.

1. Scope

The components of the system to be tested (hardware, software, middleware, etc.) are defined as “in scope“

The components of the system that will not be tested also need to be clearly defined as being “out of scope.”

1. Test types Identified

Lists the different types of testing to be performed, such as functionality, security, performance, and usability testing.

1. Problems Perceived

Identifies potential issues or challenges that may arise during testing.

1. Architecture

Describes the software architecture and its components.

1. Environment

Specifies the hardware, software, and network configurations required for testing.

9 Assumptions

10 Functionality

11 Security

12 Performance

13 Usability

14 Test Team Organization

15 Schedule

16 Defects Classification Mechanism

17 Configuration Management

18 Release Criteria

# Business Background

The restaurant management software is being developed for a small restaurant business. The restaurant operates as a sole proprietorship and was established in January 2015. It is located in downtown XYZ City, State ABC, occupying a leased space of approximately 1,000 square feet.

# Test Objectives

This testing aims to achieve the following objectives:

1. Prevent defects: Identify and address software bugs and issues early in the development process to minimize rework and delays.
2. Evaluate work products: Verify the completeness and accuracy of functional requirements, design documents, and user stories before implementation.
3. Verify requirement fulfillment: Ensure all software functionalities adhere to the specified requirements and user expectations.
4. Validate user experience: Confirm the software operates intuitively and meets the needs of both restaurant staff and customers.
5. Reduce risk: Minimize the likelihood of software defects impacting the smooth operation of the restaurant.
6. Gather feedback: Collect valuable insights from users to inform future development iterations.

# Scope

**4.1 Inclusions:** Features to be tested: This section defines a list of features/modules, functional or non-functional requirements of the Application Under Test, that need to be tested.

**Example**: The following features are to be tested:

Login page

Products

Product details

Shopping cart

This test plan covers all functionalities of the restaurant management software, including:

* Module 1: Login: Test the login functionality to access the restaurant management system.
* Module 2: Table Management:
* Add and delete and edit tables
* Assigning tables to reservations and walk-in customers
* Table transfer
* Tracking table status (occupied, vacant, being cleaned,reserved).
* Module 3: Food Management
* Add, delete and edit foods in the restaurant's menu.
* Module 4: Food Category Management
* Add, delete and edit food categories.
* Module 5: Order Taking and Processing
* Taking orders from customers , including selecting items, specifying quantities, and adding special instructions.
* Sending orders to the kitchen
* Tracking order status (received, in preparation, ready, etc.).
* Modifying and cancelling orders
* Applying discounts and promotions
* Module 6: Billing and Payments:
* Generating bills
* Accepting various payment methods
* Issuing refunds
* Module 7: Employee Management
  + Add, delete and edit employee details
  + Assigning roles.
* Moule 8: Reporting
  + Generate report on revenue,
  + Generate report popular menu items.

**4.2 Exclusions:** Features NOT to be tested: This section defines a list of features/modules, which are not to be tested. It also mentions the reason for exclusion.

Example: Payment using UPI need not be tested, as the feature is going to be disabled.

Not applicable

# Test types Identified

**Test Levels:**

1. Unit testing: In Unit/Component testing, the objective is to find as many defects as possible. Fixing defects at the unit level saves cost, effort, and time.

**Test Types:**

Function Test

|  |  |
| --- | --- |
| Purpose | * Formula Calculation and Correct Processing Conditions * Processing Data and Correct Output |
| Object | * Input Data: Field data * Output Data: Success message |
| Implementation Approach | Execute a use case, use case scenario, or function using both valid and invalid data to verify the following:   * Expected results occur when using valid data. * Appropriate errors or notifications are displayed when using invalid data. |
| Completion Criteria | * The task changes are successfully saved in the database. * All testcases designed were been executed * All errors were indentified with clear reason for deleloper to fix |
| Exceptions | Identify or describe items or issues that may impact the functional testing |

# Problems Perceived

During the testing of the small restaurant management software, the following potential problems or challenges have been identified:

1. Software defects: These are errors, anomalies, or unexpected behaviors in the software being tested.
2. Inadequate requirements: Sometimes, customer or user requirements are not clearly defined or complete, resulting in incomplete test coverage or an inability to determine the success of the software.
3. Insufficient testing resources: When there are not enough resources (such as time, personnel, servers, data) to perform thorough and effective testing.
4. Schedule conflicts: When stakeholders are not aligned on timing and priorities, leading to testing disruptions or an inability to meet deadlines.
5. System complexity: When a software system has complex architecture or design, testing and identifying defects can become more challenging.
6. Lack of customer interaction and feedback: When there is insufficient feedback or interaction from customers or users, testing may not reflect the real-world issues they may encounter.

# Architecture

* The small restaurant management software is developed using the WinForms framework and follows a three-tier architecture. This architecture separates the software into three distinct layers, each with its own responsibilities and functionalities. The three layers are:
* Presentation Layer (PL)

This layer represents the user interface of the application.

It is implemented using WinForms, which provides a visual interface for users to interact with the software.

The presentation layer handles user input, displays information to the user, and captures user actions.

* Business Logic Layer (BLL)

The business logic layer serves as an intermediary between the presentation layer and the data access layer.

It contains the core business rules and processes of the restaurant management software.

This layer validates and processes user input, performs calculations, and enforces business rules.

* Data Access Layer (DAL)

The data access layer is responsible for interacting with the underlying database.

It handles tasks related to data retrieval, storage, and manipulation.

This layer uses appropriate data access technologies, such as ADO.NET or Entity Framework, to communicate with the database.

* Database: The software utilizes a relational database management system: Sql Server to store and manage data related to menu items, order history, customer profiles, and inventory.

# Environment

## Hardware

4.1. Hardware Requirements

4.1.1. CPU >= 1.0 GHz

4.1.2. RAM >= 256 MB

4.1.3. Hard Disk Space available >= 50 MB

4.1.4. Graphic Card with Memory size >= 64MB

4.1.5. Internet Connected

- CPU>=

- RAM>=

- Hard Disk Space Available >=

## Software

- Window >=7

- Microsoft SQL Server 2019

- Microsft Visual Studio 2022

4.1. Hardware Requirements

4.1.1. CPU >= 1.0 GHz

4.1.2. RAM >= 256 MB

4.1.3. Hard Disk Space available >= 50 MB

4.1.4. Graphic Card with Memory size >= 64MB

4.1.5. Internet Connected

# Assumptions

# Functionality

***Constraints and Resolutions***

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Customer Constraints** | **Infosys Limitations** |
| Constraint 1 |  |  |
| Constraint 2 |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

***Risk Identified & Mitigation Planned***

***Test Strategy***

***Automation Plans***

***Deliverables***

# Security

***Constraints and Resolutions***

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Customer Constraints** | **Infosys Limitations** |
| Constraint 1 |  |  |
| Constraint 2 |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

***Risk Identified & Mitigation Planned***

***Test Strategy***

***Automation Plans***

***Deliverables***

# Performance

***Constraints and Resolutions***

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Customer Constraints** | **Infosys Limitations** |
| Constraint 1 |  |  |
| Constraint 2 |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

***Risk Identified & Mitigation Planned***

***Test Strategy***

***Automation Plans***

***Deliverables***

# Usability

***Constraints and Resolutions***

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Customer Constraints** | **Infosys Limitations** |
| Constraint 1 |  |  |
| Constraint 2 |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

***Risk Identified & Mitigation Planned***

***Test Strategy***

***Automation Plans***

***Deliverables***

***Compatibility Constraints and Resolutions***

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Customer Constraints** | **Infosys Limitations** |
| Constraint 1 |  |  |
| Constraint 2 |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

***Risk Identified & Mitigation Planned***

***Test Strategy***

***Automation Plans***

***Deliverables***

# Test Team Organization

|  |  |  |
| --- | --- | --- |
| Worker | Role | Specific Responsibilities/ Comments |
| Nguyễn Anh Tuấn Ngọc | Test Leader/ Tester | Providers management oversight  Responsibilities:   * Review the SRS * Make the test plan * Make checklist * Monitor Test Progress * Design the additional test cases * Executing additional test cases. * Reports * Login * Employees Management * Reading reference documents to provide necessary information to the team |
| Nguyễn Thị Thùy Trinh | Test Designer/ Tester | * Write the SRS * Design the test cases * Review the test plan * Executing test cases for the main functionalities * Food Management * Category Food Management * Table Management * Order * Payment * Recover from errors |

# Schedule

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| STT | Task | Members | Duration (days) | Start Date | EndDate |
| 1 | Make the SRS | Nguyen Thi Thuy Trinh | 4 | 20/11/2023 | 24/11/2023 |
| 2 | Make the test plan | Nguyen Anh Tuan Ngoc | 4 | 26/11/2023 | 30/11/2023 |
| 3 | Review the SRS and test plan | Nguyen Anh Tuan Ngoc,  Nguyen Thi Thuy Trinh | 2 | 1/12/2023 | 2/12/2023 |
| 4 | Design the test cases | Nguyen Anh Tuan Ngoc,  Nguyen Thi Thuy Trinh | 3 | 3/12/2023 | 6/12/2023 |
| 5 | Execute the test case | Nguyen Anh Tuan Ngoc,  Nguyen Thi Thuy Trinh | 3 | 7/12/2023 | 10/12/2023 |
| 6 | Recover from the errors | Nguyen Thi Thuy Trinh | 3 | 12/12/2023 | 15/12/2023 |
| 7 | Write the test report and done | Nguyen Anh Tuan Ngoc | 2 | 17/12/2023 | 19/12/2023 |

# Defects Classification Mechanism

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Type of Defects | Functionality | Performance | Security | Usability | Compatibility |
| Critical | * Add, Delete, Edit Food, Category, Table * Add and delete the employees * Order * Payment |  |  |  |  |
| Major | * Report |  |  |  |  |
| Minor | * The function runs not stable * Edit the account information |  |  |  |  |
| Cosmetics |  |  |  |  |  |

***Defects Logging and Status Changing Mechanism***

***Turn Around Time for defect fixes***

# Configuration Management

# Release Criteria

# APPENDIX 01: TEST CASE SPECIFICATION TEMPLATE

| ID | Test Case Field | Description |
| --- | --- | --- |
| 1 | Test case ID | Test case ID Each test case should have a unique ID |
| 2 | Test Priority | Test Priority is very useful while executing tests. There are types of priorities:  - Low  - Medium  - High |
| 3 | Test Designed by | Name of test case writer (tester) |
| 4 | Date of test designed | The date on which the tests were created |
| 5 | Test Executed by | Tester’s Name |
| 6 | Date of the Test Execution | Date of test execution |
| 7 | Name or Test Title | The title should provide a brief description of the test case, such as "Reset password". The title is quite important because it is often the first or only thing you see when glancing at a list of test cases. Clear titles are the key to helping testers quickly find the right test cases. |
| 8 | Description/Summary of Test | Detailed description for the test case (test case). In this section, you can also set up categories to organize test cases into logical groups. |
| 9 | Pre-condition | Any requirements that need to be completed before test case execution |
| 10 | Test Steps | Test Steps Test steps, give the tester a numbered list of steps to be performed in the system, making the test case easier to understand.  There should be 3-8 testing steps per test case. Too many steps will make it difficult for programmers and testers to reproduce the steps when a bug report is issued based on the test case. |
| 11 | Test Data | Test Data You can enter test data directly into the test data fields, or specify a separate file containing test data for 1 or more test cases. By using such a test data file, you avoid hard-coding test data in test cases, so a single test case can be used to test a set of test cases. |
| 12 | Expected Results | Expected Results Refers to the expected results including errors or messages appearing on the screen. The tester needs to know the expected result to evaluate whether this test case is successful or not. Details about the optimal level of this field vary depending on the situation. |
| 13 | Post-Condition | Post-Condition What is the state of the system after running the test case? |
| 14 | Status (Fail/Pass) | Status (Fail/Pass) Mark this field as failed if the actual result is not the same as the expected result.  Mark this field as Pass if the actual result is the same as the expected result. |
| 15 | Notes/Comments/Questions | Notes/Comments/Questions If some special conditions need to be noted related to the fields above. |
| 16 | Requirements | Requirements List of requirements for a specific testing cycle. |
| 17 | Attachments/References | Attachments/References Files and documents attached to the test case, such as screenshots and other supporting documents |
| 18 | Automation? (Yes/No) | Automation? (Yes/No) Fill in "YES" when the test cases using test automation. |