

|  |
| --- |
| Project Document |
| Shipping Helper System for Customers and Deliverer |

|  |  |
| --- | --- |
| **Group 2** | |
| **Group Members** | Truong Vu Binh – Student1267009  Nguyen Van Dong - Student1220013 |
| **Supervisor** | Mr. Kien Luong Trung |
| **Project code** |  |

- Hanoi, 12/2022 -

**Table of Contents**

[Acknowledgement 4](#_Toc123330759)

[Definition and Acronyms 4](#_Toc123330760)

[1 Project Introduction 5](#_Toc123330761)

[1.1 Project Information 5](#_Toc123330762)

[1.2 Project Team 5](#_Toc123330763)

[1.3 Product Background 5](#_Toc123330764)

[1.4 Existing Systems 5](#_Toc123330765)

[1.5 The proposed system 5](#_Toc123330766)

[1.6 Project Scope & Limitations 6](#_Toc123330767)

[2 Project Management Plan 7](#_Toc123330768)

[2.1 The purpose and target of a project 7](#_Toc123330769)

[2.2 Management Approach 8](#_Toc123330770)

[2.3 Responsibility Assignments 8](#_Toc123330771)

[2.4 Project Communications 9](#_Toc123330772)

[2.5 Project Work Plan 9](#_Toc123330773)

[2.6 Configuration Management 9](#_Toc123330774)

[3 Software Requirement Specification 11](#_Toc123330775)

[3.1 User Requirements 11](#_Toc123330776)

[3.2 Functional Requirements 15](#_Toc123330777)

[3.3 Non-Functional Requirements 15](#_Toc123330778)

[3.4 Other Requirements… 16](#_Toc123330779)

[4 Software Design Description 17](#_Toc123330780)

[4.1 System Design 17](#_Toc123330781)

[4.2 Database Design 18](#_Toc123330782)

[4.3 Detailed Design 19](#_Toc123330783)

[5 Software Testing Documentation 24](#_Toc123330784)

[5.1 1. Scope of Testing 24](#_Toc123330785)

[5.2 Test Strategy 24](#_Toc123330786)

[5.3 Test Plan 24](#_Toc123330787)

[5.4 Test Cases 24](#_Toc123330788)

[5.5 Test Reports 25](#_Toc123330789)

[6 Release Package & User Guides 25](#_Toc123330790)

[6.1 Deliverable Package 25](#_Toc123330791)

[6.2 Installation Guides 25](#_Toc123330792)

[6.3 Additional Information 25](#_Toc123330793)

# Acknowledgement

During the time of studying, we have well provided and imparted for all the necessary and highly valuable of professional knowledge by our teacher in Aptech, we have our faithful, careful style and creative mind in studying in the future.

Project is a good opportunity to apply appropriate for our knowledge and subjects. While performing project, we have actual and valuable experience. After one term, we were getting all one’s thoughts together and labor at a task of project. During executing this project, we have received a lot of helping from our guider, teachers.

Especially, we would thank to our teacher- Mr: Kien Luong Trung - for his enthusiastic, thoughtful, and careful guidance.

# Definition and Acronyms

*[Fill all the definitions, acronyms,… used within the document] in the table format as below]*

|  |  |
| --- | --- |
| **Acronym** | **Definition** |
| BA | Business Analysis |
| BR | Business Rule |
| ERD | Entity Relationship Diagram |
| GUI | Graphical User Interface |
| PM | Project Manager |
| SDD | Software Design Description |
| SPMP | Software Project Management Plan |
| SRS | Software Requirement Specification |
| UC | Use Case |
| API | Application Program Interface |

# Project Introduction

## Project Information

This is the project report documentation, as well as the overall documentation for the 4th semester project, written and provided by the 2nd group of class C2005L.

This project is to create a customer-to-customer friendly platform for shipments across Vietnam.

## Project Team

Our team consist of 2 members: Me, Nguyen Van Dong, who oversees this documentation, as well as Truong Vu Binh, who is the main developer of the project.

Mr. Luong Trung Kien is our supervisor for the project, who provide us support and guidance throughout the course of the project.

## Product Background

In Vietnam, the rise of e-commerce has begun in late 2010s with companies like Lazada and Shopee are the first to join the fields. As the pandemic strike, the demand for such services significantly increases. However, the companies’ services are catered around business-to-customer model instead of customer-to-customer. The same can also be said for international business such as Amazon, although their business are a combination of business-to-customer and customer-to-customer service. Furthermore, most if not all business tends to focus on relationships between customers, while the middle-man roles tend to be ignored.

With that in mind, we came together to create an application with an idea in mind, a customer-to-customer friendly solution that emphasis not only the customers, but also the middleman in the business relationship.

## Existing Systems

As previously stated, most companies’ businesses models focus on relationships between customers, while leaving the middleman behind too little to no importance.

Take Shopee for instance, the app has been in Vietnam for a long time. As such, they provide a huge catalogue of services and products for their customers, while giving an option for distributors/participated owners to distribute/promote/sell and take orders from customers. Once the order is completed, the customer would then retrieve their order within 2 to 3 days. But that begs a question, is there a business entity, or other middleman that does all the delivery? For Shoppe in particular, they have a system of deliverers, where they perform and finish the tasks in days. This, in fact, leaves the middleman/shipping merchants role redundant, with little to no importance emphasis on them.

## The proposed system

And that leads us to this system. Putting all the businesses and logistical aside, this system is a webapp catered for shipping merchants and customers, which aims to provide a much easier and more approachable access to deliverance rather than traditional postal delivery systems.

The system, instead of having the shipping merchants on the passive, where they are forced to deliver on an assigned path, shipping merchants can now take the jobs of their choice, at their own pace, and allow them to tackle the delivery as they deemed the most optimized for them.

## Project Scope & Limitations

Due to time constraint and limitations, the final product is far from what originally planned for the completed product. There would be a big need of revamping the current UI, as well as certain functionality to ensure they work as intended. Furthermore, some functions, while currently in functional state, need polishing and fine-tuned to improve user experience, as well as giving a better presentation

In this version, we have completed basic functionality, which have passed internal testing for basic usage. We hope the application is usable with a good UI and user experience, as well as provide good experience for not only the customer, but also the delivery merchants

# Project Management Plan

## The purpose and target of a project

### Project Objectives

This project aims to provide a functional web application, with fundamental features and a functional system. Along with that, we hoped to learn new knowledge about .NET web developing, and how a web application could be developed

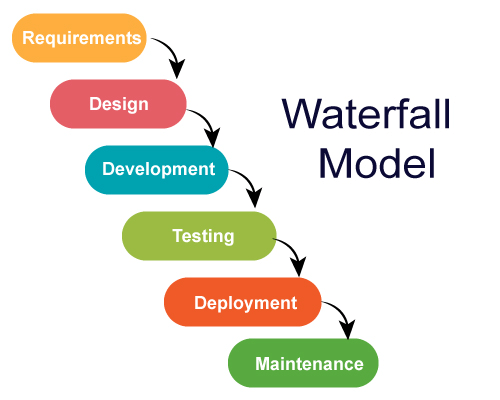
But most importantly, we wish to broaden our understanding of C# programming language, Web development, as well as database management and additional knowledge we picked up while creating the app.

### Project Risks

|  |  |  |
| --- | --- | --- |
| **Risks** | **Details** | **Solution** |
| Time constraints | Due to limited time allocated to the project, some of functionality in which originally in our backlog could not be implemented in time | Implement basic functionality, as well as required backends for such functions. |
| Technical Knowledges | One of our members has little to no proficiency in using the programming language that the group is using for the projects, as well as the knowledge needed to use source control solution like Git and GitHub | Our first week of the project are spent on helping each other reviewing .NET and C# programming language fundamentals, as well as giving basic tutorials of using GitHub as a collaboration tool for development |
| Environment setting up | One of our team members has issues importing the projects into the IDE to begin their contributions | To help resolve the issue, we spent a day discussing the issue, as well has conducting remote assisting session to fasten the issue resolving |

## Management Approach

### Project Process



The above image is what describe the “Waterfall Model”, a common model used in various projects. Our working process shares the same model, except for the “Design” phase, as none of the team member are well-versed in such field, and training to catch up with the design basic would require time. Due to limited and conflicting time and schedule, we decided to proceed without the Design phase.

### Quality Management

To ensure the project overall flow, we agreed upon using proper C# coding convention, as well as separating our codebase into various layers for easier maintenance. We also use GitHub to manage the project codebase, where all features’ implementations are separate into different branches and being reviewed by each other before merging to the main branch, where the branch will also be deleted should the merge is completed. We also used Discord to communicate during the project.

### Training Plan

On the first week of the project, we spend time communicating about the tools, and database design we will be using throughout the project, as well as learning how to use source control solution such as Git and GitHub to effectively collaborate with each other.

On the second week onward, we start setting up the base for the project, as well as implementing and testing the functionality and features as we proceed with what was planned in the first week.

## Responsibility Assignments

## Project Communications

The team uses Discord, an instant messaging and social platform as our primary means of communication throughout the project. We also use GitHub for managing codebase and collaborating on the development process.

## Project Work Plan



## Configuration Management

### Document Management

This document was written in Microsoft Word

Some diagrams used in this document are created using diagram.io, a free web application for creating simple yet elegant and informative diagrams.

### Source Code Management

All Source code are managed/archived by Truong Vu Binh. He created and provided the project repository, as well as managing changes to the source codes by reviewing and approving pull requests (PR). Due to the time constraint and scheduling, we decided to use this method, as this would help us avoid code conflicting and ensure everyone shared the same codebase, as well as avoid any unnecessary data loss due to unforeseen issues that may arise.

### Tools & Infrastructures

Our team uses the following tools and software to help aid the development process:

* C# Programming Language
* .NET 6 SDK
* **Visual Studio:** Our go to option when programming .NET applications. It has a plethora of tools to help us quickly get started on the project server-related functions, as well as databases.
* **Visual Studio Code:** Visual Studio is a great IDE, but it lacks supports for HTML and front-end developments in general, as such, we switch to Visual Studio Code as our code editor when we are developing the View aspect of the project.
* **Azure Data Studio:** Azure Data Studio is a free database management software that allow us to quickly glance at our project database, as well as helping us write queries faster and align well with the SQL coding convention thanks to its autocompletion feature.

# Software Requirement Specification

Our web application is created with 3 user groups in mind, each with a scoped set of permissions and features available to them. These are:

**For Shipping Merchants/Driver/Deliverer:**

* View the list of available offers posted by shop merchants
* View detail of a selected offer
* Accept an offer in listing
* Update the offer status after accepting an offer
* Edit their profile
* Update their password

**For Shop Merchants:**

* View the list of offers posted by the user
* Create a new Offer
* Update and Delete offers if the offer is not being accepted
* Edit their profile
* Update their password
* Report a shipping merchant offense

**For Admin:**

* Manage the list of all offers
* Manage user accounts
* Reply to customer messages
* View list of reported shipping merchants

## User Requirements

|  |  |
| --- | --- |
| **Use Case ID** | UC-01 |
| **Use Case Name** | Sign into the application |
| **Actors** | Guest |
| **Description** | User signs in to the application |
| **Pre-condition** | User is connected to the internet |
| **Post Condition** | User is signed in and redirect user to the dashboard |
| **Trigger** | User wishes to take an offer or post an offer |
| **Main flow** | |  |  |  | | --- | --- | --- | | No | Actor | Action | | 1 | Guest | Access the ShipLink home page | | 2 | System | Display the home page | | 3 | Guest | Tap/click on Sign In/Join Us link at the top of the page | | 4 | System | Display the sign in page | | 5 | Guest | Fill in their username and password then tap/click on Sign in | | 6 | System | Authenticate the user using the given credentials | | 7 | System | Display error if invalid, else redirect user to the dashboard | |
| **Alternative Flow** | N/A |
| **Exception Flow** | N/A |
| **Business Flow** | N/A |

|  |  |
| --- | --- |
| **Use Case ID** | UC-02 |
| **Use Case Name** | Sign out of the application |
| **Actors** | Registered User |
| **Description** | User signs out of the application |
| **Pre-condition** | User is connected to the internet |
| **Post Condition** | User is signed out and redirect user to the home page |
| **Trigger** | User wishes to sign out oof the application |
| **Main flow** | |  |  |  | | --- | --- | --- | | No | Actor | Action | | 1 | Guest | Access the ShipLink page | | 2 | System | Display the dashboard | | 3 | Guest | Tap/click on Sign Out at the left side of the dashboard | | 4 | System | Sign the user out of the system | | 5 | System | Rediect user to the home page | |
| **Alternative Flow** | N/A |
| **Exception Flow** | N/A |
| **Business Flow** | N/A |

|  |  |
| --- | --- |
| **Use Case ID** | UC-04 |
| **Use Case Name** | Accept a posted offer |
| **Actors** | Registered User |
| **Description** | User wants to accept an offer on the dashboard |
| **Pre-condition** | User is connected to the internet and has signed in to the application as a shipping merchant |
| **Post Condition** | Offer is saved to their Accepted list |
| **Trigger** | Shipping merchant wished to accept an offer |
| **Main flow** | |  |  |  | | --- | --- | --- | | No | Actor | Action | | 1 | User | Access the ShipLink home page | | 2 | System | Display the dashboard page if user is signed in | | 3 | User | Tap/click on Detail button on the offer listing | | 4 | System | Display the Offer Detail page | | 5 | User | Check Offer detail, then tap/click Accept Offer | | 6 | System | Acknowledge the acceptance and save the offer to the shipping merchant Accepted List | | 7 | System | Display a success message when finished | |
| **Alternative Flow** | N/A |
| **Exception Flow** | N/A |
| **Business Flow** | N/A |
|  |  |

|  |  |
| --- | --- |
| **Use Case ID** | UC-05 |
| **Use Case Name** | Post an Offer |
| **Actors** | Registered User |
| **Description** | User post an offer to the application |
| **Pre-condition** | User is connected to the internet, and it’s signed in as a shop merchant |
| **Post Condition** | Offer is saved to the database |
| **Trigger** | User post an offer to the application |
| **Main flow** | |  |  |  | | --- | --- | --- | | No | Actor | Action | | 1 | Guest | Access the ShipLink home page | | 2 | System | Display the dashboard page if user is signed in | | 3 | Guest | Tap/click on New Offer link at the left side of the dashboard | | 4 | System | Display the New Offer form page | | 5 | Guest | Fill in the destination address, price, note to shipping merchants, as well as uploading an image of the package, package description and tap Create Offer | | 6 | System | Save the information to the database | | 7 | System | Redirect back to the Posted Offer list | |
| **Alternative Flow** | N/A |
| **Exception Flow** | N/A |
| **Business Flow** | N/A |

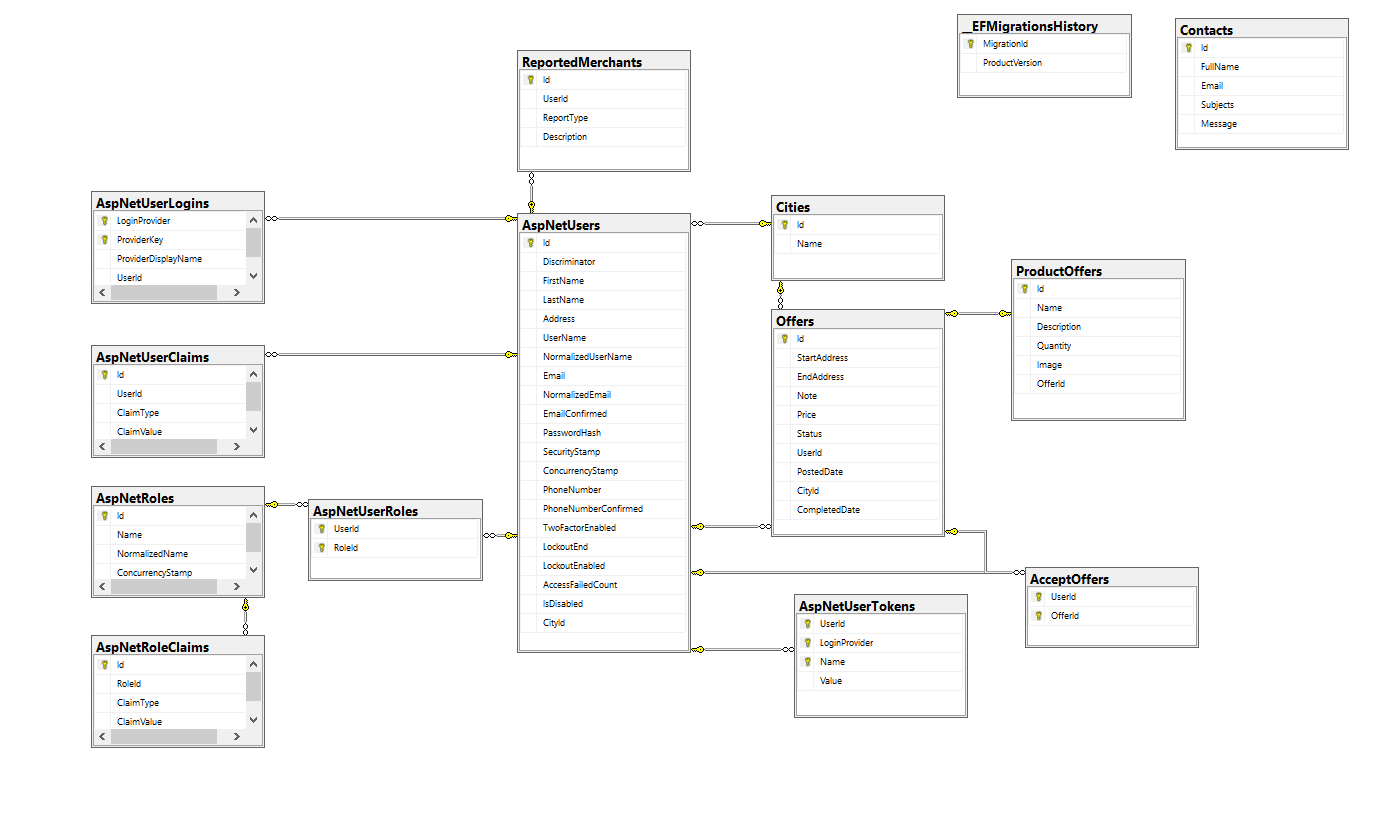
|  |  |
| --- | --- |
| **Use Case ID** | UC-06 |
| **Use Case Name** | Manage offers |
| **Actors** | Admin |
| **Description** | Manage all offers |
| **Pre-condition** | User is connected to the internet and it’s signed in as admin |
| **Post Condition** | System display all offers in the system |
| **Trigger** | Admin wants to see the list of all posted offers |
| **Main flow** | |  |  |  | | --- | --- | --- | | No | Actor | Action | | 1 | Guest | Access the ShipLink home page | | 2 | System | Display the home page | | 3 | Guest | Tap/click on Sign In/Join Us link at the top of the page | | 4 | System | Display the sign in page | | 5 | Guest | Fill in their username and password then tap/click on Sign in | | 6 | System | Authenticate the user using the given credentials | | 7 | System | Display error if invalid, else redirect user to the admin dashboard, show the offer list | |
| **Alternative Flow** | N/A |
| **Exception Flow** | N/A |
| **Business Flow** | N/A |

|  |  |
| --- | --- |
| **Use Case ID** | UC-07 |
| **Use Case Name** | Manage users |
| **Actors** | Admin |
| **Description** | Manage registered users in the system |
| **Pre-condition** | User is connected to the internet and it’s signed in as an admin |
| **Post Condition** | Admin can manage users |
| **Trigger** | Admin wishes to see a list of registered user information |
| **Main flow** | |  |  |  | | --- | --- | --- | | No | Actor | Action | | 1 | Guest | Access the ShipLink home page | | 2 | System | Display the home page | | 3 | Guest | Tap/click on Sign In/Join Us link at the top of the page | | 4 | System | Display the sign in page | | 5 | Guest | Fill in their username and password then tap/click on Sign in | | 6 | System | Authenticate the user using the given credentials | | 7 | System | Display error if invalid, else redirect user to the admin dashboard, | | 8 | Admin | Tap Manage User at the left side of the dashboard | | 9 | System | Display the list of users | |
| **Alternative Flow** | N/A |
| **Exception Flow** | N/A |
| **Business Flow** | N/A |

|  |  |
| --- | --- |
| **Use Case ID** | UC-07 |
| **Use Case Name** | Register as a shipping merchant |
| **Actors** | Guest, Admin |
| **Description** | User register as a shipping merchant |
| **Pre-condition** | User is connected to the internet |
| **Post Condition** | User is registered and redirected to the dashboard |
| **Trigger** | User wishes to partake to the system as a driver |
| **Main flow** | |  |  |  | | --- | --- | --- | | No | Actor | Action | | 1 | Guest | Access the ShipLink home page | | 2 | System | Display the home page | | 3 | Guest | Tap/click on Sign In/Join Us link at the top of the page | | 4 | System | Display the sign in page | | 5 | Guest | Tap the Create a new account link | | 6 | System | Display the Register form | | 7 | System | On account type, user select account type to be “Shipping Merchant”, then provide their First Name, Last Name, Email Address, Phone Number, and tap Register | | 8 | System | Save the information to the database, redirect user to the Dashboard | |
| **Alternative Flow** | N/A |
| **Exception Flow** | N/A |
| **Business Flow** | N/A |

## Functional Requirements

### System Functional Overview

**

## Non-Functional Requirements

### Reliability

We make sure that:

* All forms have proper validation put in
* All basic functionality works properly and reliable

### Availability

As this project is a web application, it is available to almost any modern device with a web browser.

### Security

All functionalities have proper authentication method implemented and secured

### Maintainability

The codebase is separated into different layers, allowing easier maintainability, as well as improve readability for future should there be a need to implement additional functions, or making modification to existing codebase

### Portability

The project is a web application, as such, the app can be used by either mobile devices, tablets, and/or personal computers.

## Other Requirements…

Aside from the mentioned functional requirements, we also take extra efforts to make sure that:

* Our user interface is in functional state, no overlap elements, or elements being misplaced
* The codebase is easy to read, and maintain, so our future code maintenance would be easier.
* The codebase is optimized for reliability, all methods are made to throw friendly error messages for user in case of any operation failure
* All required items for deploying, such as database scripts, the project documentation, the presentation slides, and the web application codebase itself are included within the team repository

# Software Design Description

## System Design

### System Architecture

Text

Description automatically generated with medium confidence

Our project following a hybrid of MVC and N-tier architecture. Whereas our main applications are created with strictly MVC design architecture in mind. And our controller logic lies in different layers, such as the Services, Repository, the Core (database models, database migrations history, ) Common layer

### Package Diagram

As per the above screenshot, below is the detailed explanation

* **ShippingHelper**: Our primary web application. This project strictly follows the MVC design architecture and the presentation of the application. All controllers will execute necessary actions by calling the respective services.
* **ShippingHelper.Common**: This layer contains extension methods, and other utilities method for various purposes (for instance: File Uploading, generate friendly date/time)
* **ShippingHelper.Core:** This layer contains the Database configurations, the application database models, and all database modification history
* **ShippingHelper.Repository:** This layer containsmethodsused to interact with the database, this layer includes a base repository that all other repositories are implementing, as well as the Unit of Work class for calling the repository method of choice.
* **ShippingHelper.Services:** This layer contains calls for each respective repository methods. Controller Action Methods would consume these services, which in turns, call the repository method.
* **ShippingHelper.ViewModels:** This layer contains models to be used in the web application. Since we do not want to directly tamper with the database model, we used these View Models as a means of data collections from forms, where then the View Models data will then be converted to their respective database models.

## Database Design

*A picture containing text, screenshot, indoor

Description automatically generated*

Our application uses the ASP.NET Core Identity system, as such, there are additional table in which are automatically generated when we implement the identity system. There tables are marked with the AspNet prefix in its name.

## Detailed Design

### Sign in to the system

Diagram

Description automatically generated

### Add a new Offer

Diagram

Description automatically generated

### Shipping merchants accept a new offer

Diagram

Description automatically generated

### Shop Merchants edit an offer they created

Diagram

Description automatically generated

# Software Testing Documentation

## Scope of Testing

When conduct our testing, we only test the repository layer of our application, using NUnit testing framework, we wrote test cases for methods that returns data to us. As such, for other operation, such as Creating, Updating, or Deleting, they are all manually tested by the person responsible for testing on our team.

## Test Strategy

### Testing Types

Our tests are written and run-in services layer. With this, we can be ensured that all data are properly updated, added, removed, and retrieved in an isolated, mocking environment, using mocked data from Mockaroo, a test data generating tool which is available for free

### Test Levels

Our tests are written and run-in services layer. With this, we can be ensured that all data are properly updated, added, removed, and retrieved in an isolated, mocking environment, using mocked data from Mockaroo, a test data generating tool which is available for free

### Supporting Tools

Our test cases are written using the following tools:

* **NUnit testing framework**: a popular testing framework that has built-in support in our IDE
* **Moq**: a mocking library allow us to create mock data to test our repository functionality without ever having to directly interact to the project database

All test cases are white box tests, and conducted in an isolated environment, meaning, the test cases are performed by using mock data, with a mock-up database context required for all the repository to work.

## Test Plan

### Human Resources

All tests are conducted by the whole team. Binh, who is our group developer, perform Unit testing on mocking environment. And me, Dong, who also perform testing manually to make sure all tests that passed the white box testing phase would also pass manual testing

### Test Environment

Our test cases are written using the following tools:

* **NUnit testing framework**: a popular testing framework that has built-in support in our IDE
* **Moq**: a mocking library allow us to create mock data to test our repository functionality without ever having to directly interact to the project database

All test cases are white box tests, and conducted in an isolated environment, meaning, the test cases are performed by using mock data, with a mock-up database context required for all the repository to work.

### Test Milestones

## Test Cases

### Sign in

|  |  |  |
| --- | --- | --- |
| **Test case Description** | **Test Procedure** | **Expected Output** |
| Display Sign-in screen | Go to the app homepage and tap “Sign-in/Join Us” | The Sign in screen shows up |
| Sign in using valid credentials for the admin role | Go to the app homepage and tap “Sign-in/Join Us”  Enter valid account credentials  Tap Sign-in | The admin page shows up |
| Sign in using valid credentials for the shop merchant role | Go to the app homepage and tap “Sign-in/Join Us”  Enter valid account credentials  Tap Sign-in | The user page shows up, redirect to the created offer screen |
| Sign in using valid credentials for the shipping merchant role | Go to the app homepage and tap “Sign-in/Join Us”  Enter valid account credentials  Tap Sign-in | The user page shows up, redirect to the available offer screen |
| Failed sign in due to empty username | Go to the app homepage and tap “Sign-in/Join Us”  Enter account password but leave username empty  Tap Sign-in | Do not submit the sign in form until the username field is filled |
| Failed sign in due to empty password | Go to the app homepage and tap “Sign-in/Join Us”  Enter account username but leave password empty  Tap Sign-in | Do not submit the sign in form until the password field is filled |
| Failed sign in due to invalid credentials | Go to the app homepage and tap “Sign-in/Join Us”  Enter account credentials  Tap Sign-in | The “Incorrect login attempts” message appears |

### Sign out

|  |  |  |
| --- | --- | --- |
| **Test case Description** | **Test Procedure** | **Expected Output** |
| Sign out of the system | On any screen, tap “Sign Out” button | The homepage appears |

### Add an offer

|  |  |  |
| --- | --- | --- |
| **Test case Description** | **Test Procedure** | **Expected Output** |
| Users add a new offer | Go to the app homepage and tap “Sign-in/Join Us”  Enter valid account credentials  Tap “Sign in”  Tap “New Offer”  Fill in the Destination address  Fill in the Offer Price  Fill in the offer description  Select an image to represent their shipping package  Fill in a message to the shipping merchant  Tap Create | The offer is added, and user is redirected to their created offer list |
| Users leave destination address empty | Go to the app homepage and tap “Sign-in/Join Us”  Enter valid account credentials  Tap “Sign in”  Tap “New Offer”  Leave Destination address field empty  Fill in the Offer Price  Fill in the offer description  Select an image to represent their shipping package  Fill in a message to the shipping merchant  Tap Create | When user tap Create, the message “Destination address is required” is displayed, prevent the form from submitting |
| Users leave offer price empty | Go to the app homepage and tap “Sign-in/Join Us”  Enter valid account credentials  Tap “Sign in”  Tap “New Offer”  Fill in Destination address  Leave Offer Price empty  Fill in the offer description  Select an image to represent their shipping package  Fill in a message to the shipping merchant  Tap Create | When user tap Create, the message “Offer price is required”, prevent the form from submitting |

### Edit an offer

|  |  |  |
| --- | --- | --- |
| **Test case Description** | **Test Procedure** | **Expected Output** |
| Users edit a posted offer | Go to the app homepage and tap “Sign-in/Join Us”  Enter valid account credentials  Tap “Sign in”  Tap the pencil icon on a listing  Fill in the Destination address  Fill in the Offer Price  Fill in the offer description  Select an image to represent their shipping package  Fill in a message to the shipping merchant  Tap Create | The offer is updated, and user is redirected to their created offer list |
| Users leave destination address empty when editing their offer | Go to the app homepage and tap “Sign-in/Join Us”  Enter valid account credentials  Tap “Sign in”  Tap “New Offer”  Leave Destination address field empty  Fill in the Offer Price  Fill in the offer description  Select an image to represent their shipping package  Fill in a message to the shipping merchant  Tap Create | When user tap Save, the message “Destination address is required” is displayed, prevent the form from submitting |
| Users leave offer price empty when editing their offer | Go to the app homepage and tap “Sign-in/Join Us”  Enter valid account credentials  Tap “Sign in”  Tap “New Offer”  Fill in Destination address  Leave Offer Price empty  Fill in the offer description  Select an image to represent their shipping package  Fill in a message to the shipping merchant  Tap Create | When user tap Save, the message “Offer price is required”, prevent the form from submitting |

### Delete an offer

|  |  |  |
| --- | --- | --- |
| **Test case Description** | **Test Procedure** | **Expected Output** |
| Users delete a posted offer | Go to the app homepage and tap “Sign-in/Join Us”  Enter valid account credentials  Tap “Sign in”  Tap the rubbish bin icon on a listing  Tap Delete on confirm deletion | The offer is deleted, and user is redirected to their created offer list |
| Users attempts to delete an offer that is accepted | Go to the app homepage and tap “Sign-in/Join Us”  Enter valid account credentials  Tap “Sign in” | No delete icon is displayed for the offer listing |

### View all accounts

|  |  |  |
| --- | --- | --- |
| **Test case Description** | **Test Procedure** | **Expected Output** |
| Admin wants to view a list of users | Go to the app homepage and tap “Sign-in/Join Us”  Enter valid admin account credentials  Tap “Sign in”  Tap “User managements” | The list of registered users appears |
| Search a user | Go to the app homepage and tap “Sign-in/Join Us”  Enter valid account credentials  Tap “Sign in”  Tap “User Management”  Enter a keyword in the searchbox | The table is refreshed with data that matched the keyword |
| Next page | Go to the app homepage and tap “Sign-in/Join Us”  Enter valid account credentials  Tap “Sign in”  Tap “User Management”  Tap Next page | The table is refreshed with the next records |
| Previous page | Go to the app homepage and tap “Sign-in/Join Us”  Enter valid account credentials  Tap “Sign in”  Tap “User Management”  Tap Previous page | The table is refreshed with the previous records |
| Show a number of entries per page | Go to the app homepage and tap “Sign-in/Join Us”  Enter valid account credentials  Tap “Sign in”  Tap “User Management”  Tap a number in the Shows {number} of entries er page | The table is refreshed with a requested number of records per page |

### View created/available offers

|  |  |  |
| --- | --- | --- |
| **Test case Description** | **Test Procedure** | **Expected Output** |
| Users wants to view a list of offers | Go to the app homepage and tap “Sign-in/Join Us”  Enter valid user account credentials  Tap “Sign in” | The list of created/available offers is shown |
| Search an offer | Go to the app homepage and tap “Sign-in/Join Us”  Enter valid account credentials  Tap “Sign in”  Enter a keyword in the searchbox | The table is refreshed with data that matched the keyword |
| Next page | Go to the app homepage and tap “Sign-in/Join Us”  Enter valid account credentials  Tap “Sign in”  Tap Next page | The table is refreshed with the next records |
| Previous page | Go to the app homepage and tap “Sign-in/Join Us”  Enter valid account credentials  Tap “Sign in”  Tap Previous page | The table is refreshed with the previous records |
| Show a number of entries per page | Go to the app homepage and tap “Sign-in/Join Us”  Enter valid account credentials  Tap “Sign in”  Tap a number in the Shows {number} of entries er page | The table is refreshed with a requested number of records per page |

## Test Reports

All tests passed, both in white box test phase, and manual testing phase, meaning that all implemented functionality is working as intended

# Release Package & User Guides

## Deliverable Package

| **No.** | **Deliverable Item** | **Description** |
| --- | --- | --- |
| 1 | Schedule/Task Tracking | Provided in section 6.3.2 in this document |
| 2 | Source Codes | Provided in project repository |
| 3 | Database Script(s) | Provided in project repository |
| 4 | Final Report Document | Provided in project repository |
| 5 | Test Cases Document | Provided in section 5.4 in this document |
| 6 | Slide | Provided in project repository |

## Installation Guides

### System Requirements

|  |  |
| --- | --- |
| **Server** | **Client** |
| 1GHz Dual-core CPU or higher | 1GHz Dual-core CPU or higher |
| SQL Server 2017 or later | Any web browser with latest update |
| .NET 6 Runtime | Windows 7 or later |
| Windows 8.1 or later | At least 1GB of RAM or more |
| IIS Role or feature enabled | 10GB of free hard disk space |
| 20GB of free hard disk space | Internet connection |
| At least 2GB of RAM or more |  |

### Installation Instruction

**For Server:**

* Enable Internet Information Services (IIS) found in Optional Features in Windows
* Install latest version of .NET Runtime, SQL Server
* Add the web app into IIS
* Query the database using the given SQL Scripts to have the database required for the webpage to work

## Additional Information

### Project Diary/Reviewing Reports

Theses files are included within the “doc” folder inside the repository. Please see the separated “Project Diary” and Project Report Forms inside the “doc” folder

### Task list

The table below is the breakdown of all group member responsibility

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Task** | **Start time** | **Finish time** | **Implemented by** | **Evaluation(%)** |
| **Feasibility study phase** |  |  |  |  |
| Writing Documentation | 08/12/2022 | 09/12/2022 | Nguyen Van Dong | 50 |
| Idea Contribution | 07/10/2022 | 07/10/2022 | Nguyen Van Dong | 15 |
| Idea Contribution | 08/10/2022 | 08/10/2022 | Truong Vu Binh | 15 |
| Finalising Ideas | 10/10/2022 | 10/10/2022 | Truong Vu Binh  Nguyen Van Dong | 20 |
| **Requirement Analysis phase** |  |  |  |  |
| System Designing | 15/10/2022 | 17/10/2022 | Truong Vu Binh | 50% |
| System Designing | 15/10/2022 | 17/10/2022 | Nguyen Van Dong | 50% |
| **Design phase** |  |  |  |  |
| Database Designing | 20/10/2022 | 20/10/2022 | Truong Vu Binh | 50% |
| Graphical User Interface Designing | 25/10/2022 | 05/11/2022 | Nguyen Van Dong | 50% |
| **Development phase** |  |  |  |  |
| Signing-in/Signing-out the system | 10/11/2022 | 10/11/2022 | Truong Vu Binh | 15% |
| Offers managing functions for shop merchants | 13/11/2022 | 16/11/2022 | Truong Vu Binh | 10% |
| System User Managing Function | 14/12/2022 | 18/12/2022 | Nguyen Van Dong | 10% |
| Update an offer status shipping merchant | 16/11/2022 | 17/11/2022 | Nguyen Van Dong | 15% |
| Edit User Profile | 19/11/2022 | 21/11/2022 | Nguyen Van Dong | 10% |
| View Offer Details | 23/11/2022 | 24/11/2022 | Truong Vu Binh | 10% |
| Searching data in Offer/User/Reports listing | 15/11/2021 | 15/11/2021 | Truong Vu Binh | 10% |
| Report Users | 15/11/2021 | 15/11/2022 | Truong Vu Binh | 10% |
| Send messages | 13/11/2021 | 14/11/2022 | Truong Vu Binh | 10% |
| **Testing phase** |  |  |  |  |
| Testing sign-in, sign-out ability | 28/11/2021 | 28/11/2022 | Truong Vu Binh | 15% |
| Testing Offer managing systems for shop merchants | 28/11/2022 | 28/11/2022 | Truong Vu Binh | 10% |
| Testing Update offer status for shipping merchants | 28/11/2022 | 28/11/2022 | Nguyen Van Dong | 10% |
| Testing View Offer Detail page | 29/11/2022 | 29/11/2022 | Nguyen Van Dong | 15% |
| Testing the ability to search data in table using fuzzy search | 29/11/2021 | 29/11/2021 | Truong Vu Binh | 10% |
| Testing the user managing function for admin | 30/11/2022 | 30/11/2022 | Nguyen Van Dong | 10% |
| Testing the Edit profile for all users | 01/12/2022 | 01/12/2022 | Nguyen Van Dong | 10% |
| Testing the ability to Report Shipping Merchants | 01/12/2022 | 02/12/2022 | Truong Vu Binh  Nguyen Van Dong | 10% |
| Testing the Send Message function | 03/12/2022 | 03/12/2022 | Truong Vu Binh  Nguyen Van Dong | 10% |