**Table of Contents**

Introduction 2

Architecture Design 2

Design Class Diagram 3

Sequence Diagram 4

Database Design 7

Generic User Interface 9

Screenshot 10

Test Document Iteration 1 15

Test Objectives 15

Test Plan 16

Unit Testing 17

Integration Testing 32

System Testing 37

Non-functional requirements test 40

Test Analysis Report 41

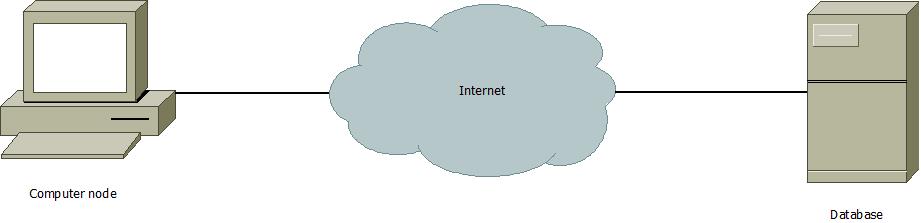
Updated Gantt Chart 42

**Design Document**

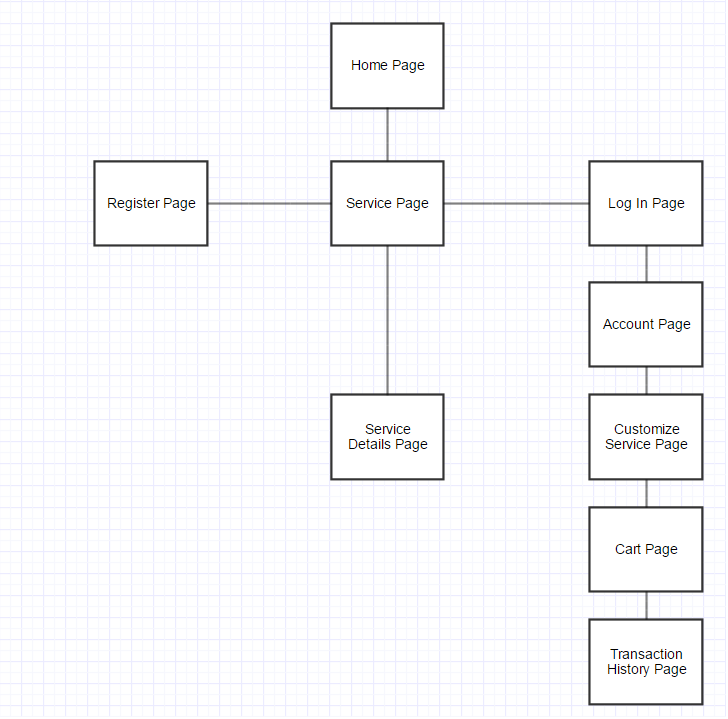
**Introduction**

Nowadays, people tend to rely more and more on the internet. This is due to the fact that internet has provided people with the convenience of discovering new things and reaching out to new people around the world. Besides opening a shop to start their business or to sell their services, people are starting to see the internet as a more economical and convenient alternative to start their business. However, creating such an online platform is not an easy task and requires professional skills.

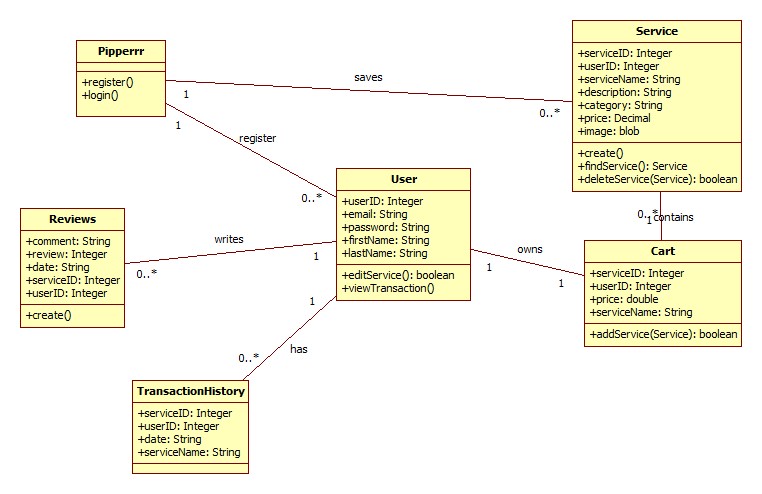
**Architectural Design**



There are three major components required for the system to work. Firstly, the website interface will be displayed by the computer node which allows interaction between the user and the system such as receiving data or input data. The system requires internet access to use so the computer node needs to be connected to the internet. Through the internet, it will be then connected to the database where the system data is stored. The database will manage the traffic of the data going in or out requested by the user.

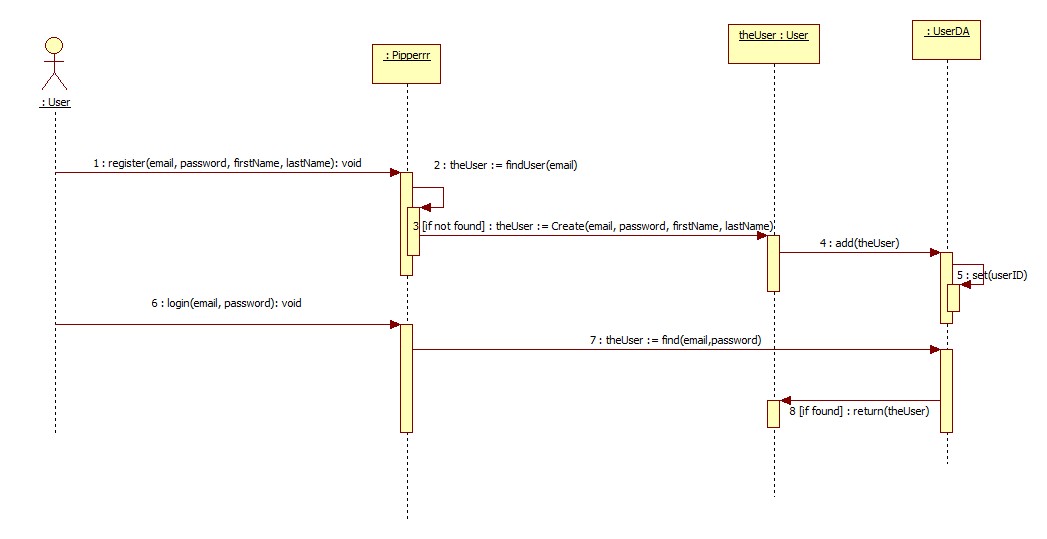


**Design Class Diagram**

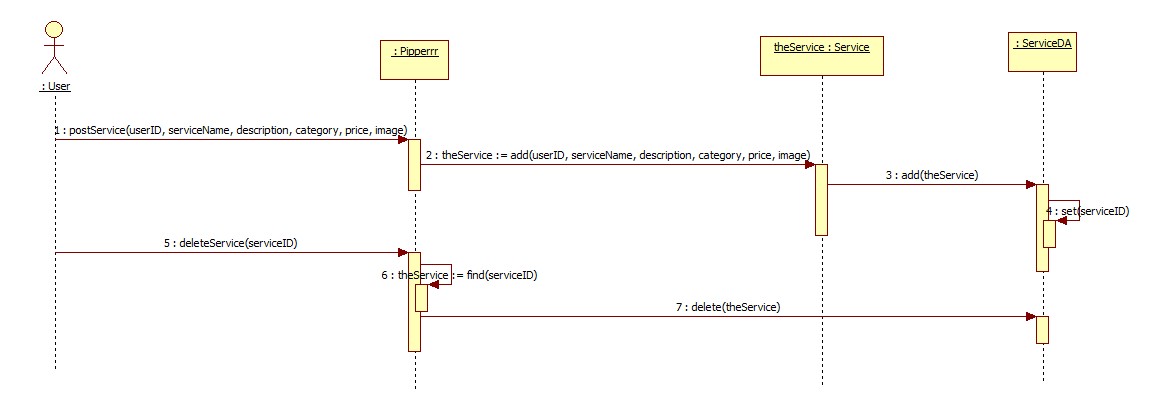


**Sequence Diagram**

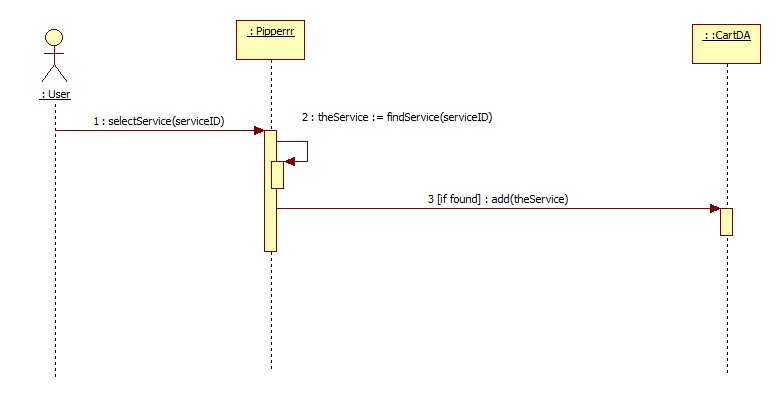
**Use Case 1: Register and Log in**



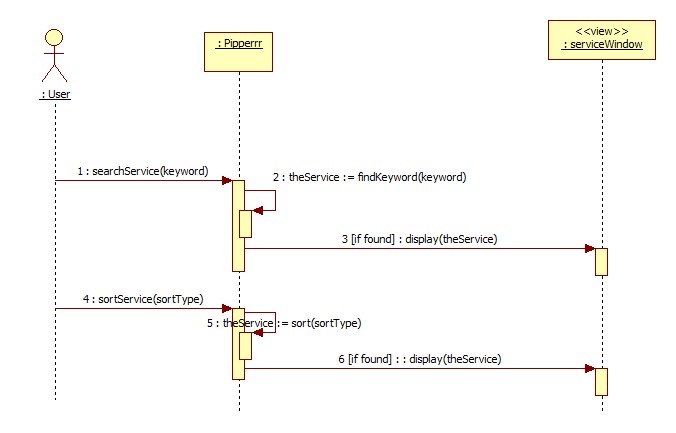
**Use Case 2: Sell services**



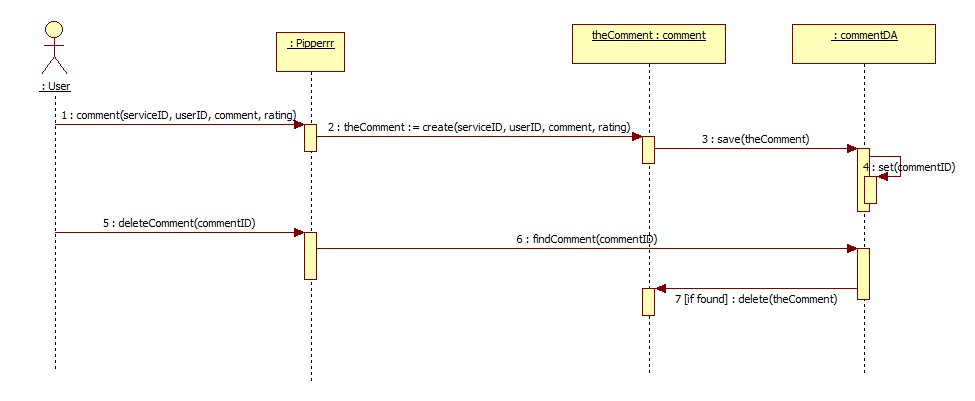
**Use Case 3: Buy services**



**Use Case 4: Search for services**



**Use Case 5: Give comments or ratings**



**Use Case 6: View transaction history**



**Database Design**



Pipperrr contains 5 database tables. The first one is the user, which stores basic information regarding each user. userID is unique and auto increments each time a new row is added into the database. The “email” is the text used to identify the user when logging in, the “password” is the security code for the user to log in, “firstName” holds the first name of the user, and “lastName” holds the last name of the user.



The service tables store the services created by users. Each service created will have a unique serviceID which auto increments. The userID of the user who created will also be recorded for easier reference. The “serviceName” is a text that holds the service name, “description” is the text that describes the service, “category” is used as a tag for easy search, “price” is the decimal that indicates the cost of the service, “image” is a blob that holds the images that are uploaded by the users.



The cart table stores the services that user adds and may want to purchase later. It stores the unique serviceID and the userID who has added it to their cart. The “price” is a double that that holds the data for the price of the service. The “serviceName” is a text that holds the name of the service.

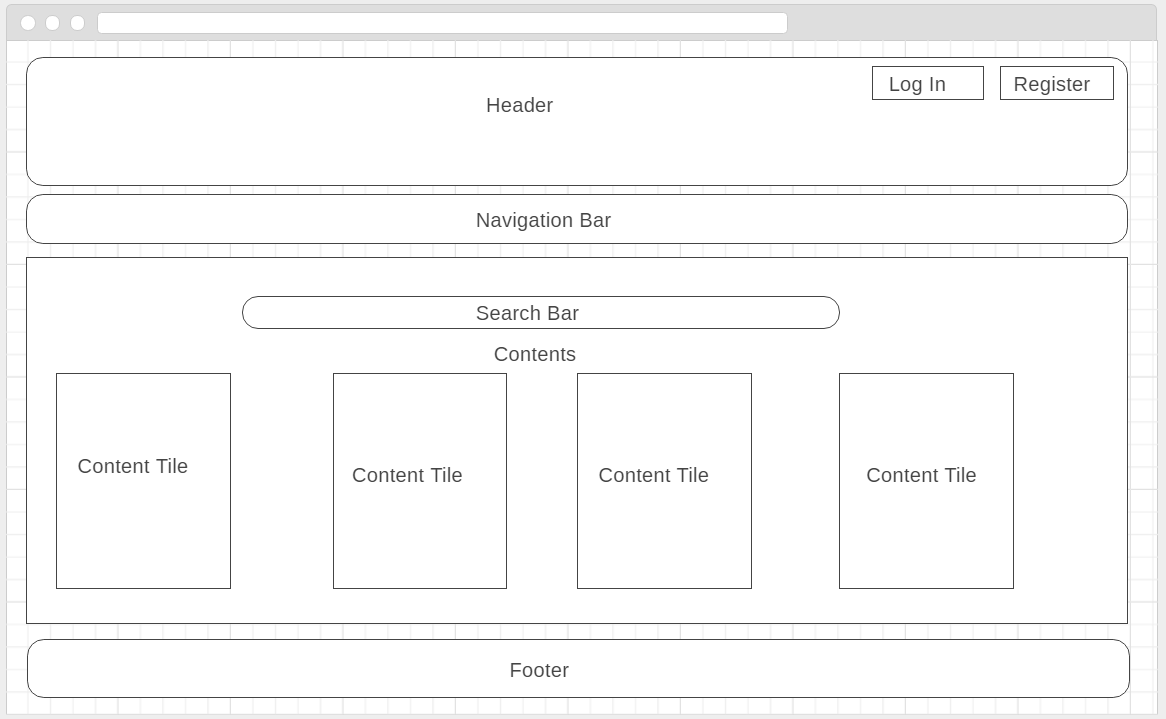


The review table stores the comments and ratings for each service. The unique serviceID is stored to refer which comments belong to which service and the userID to know who gave the comment. The “comment” is a text that holds the comment text input by the user and the “review” holds the integer for the user to rate the service post from in a range of 1 to 5. The date holds the text of when the service was posted.



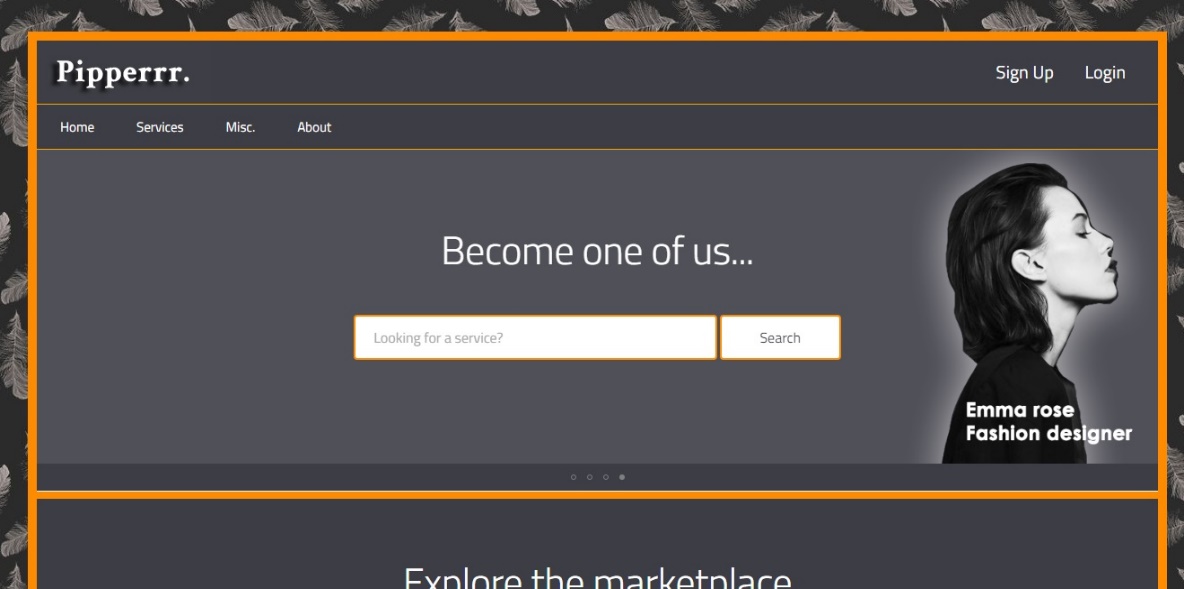
The transaction table stores the information of which services the user has bought or purchased. The serviceID and userID are stored for easier referencing and date to record when did the transaction occur. The “date” holds the text of when a purchase was made. The “serviceName” is a text that indicates the service name that was purchased.

**Generic User Experience**

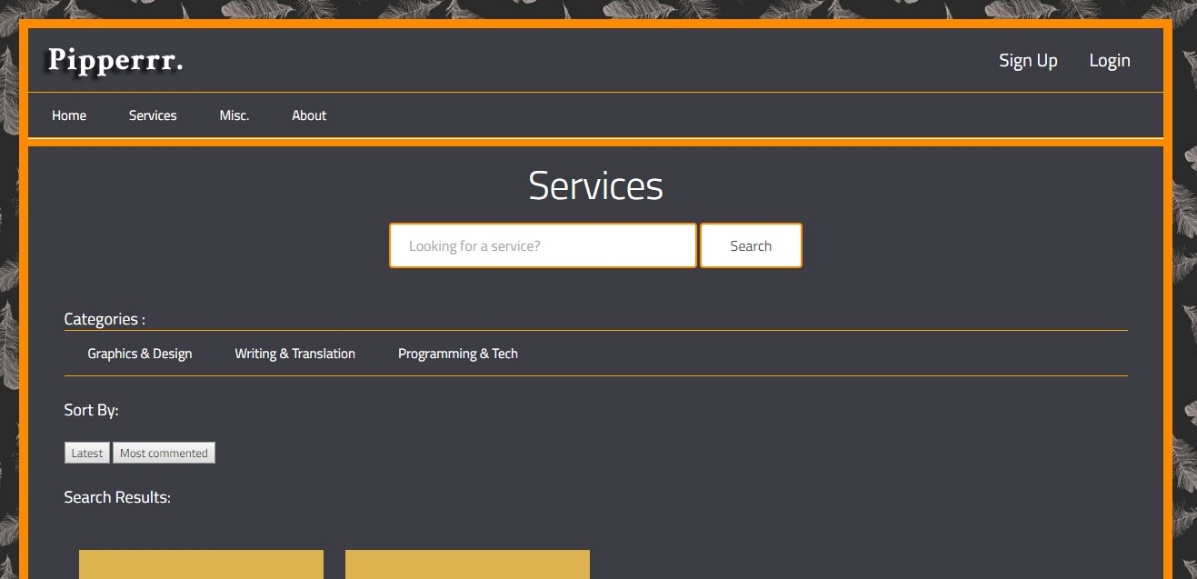


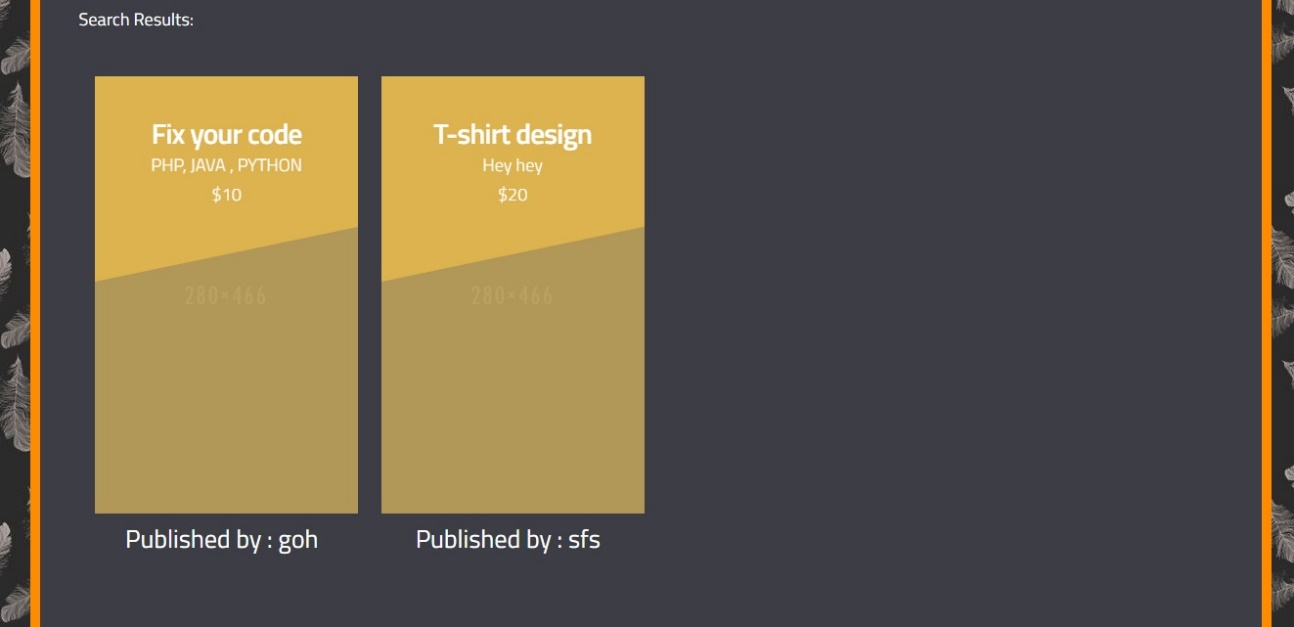
The wireframe above describes the general layout of the website pages as they are all almost identical to each other. The wireframe above has the header on the top, log in and register button on the top right of the header. The navigation bar is below the header and the content section is in the middle of the page. Finally, the footer is at the bottom of the page. As for the other pages, the search and content tiles are optional.

**Screenshot**

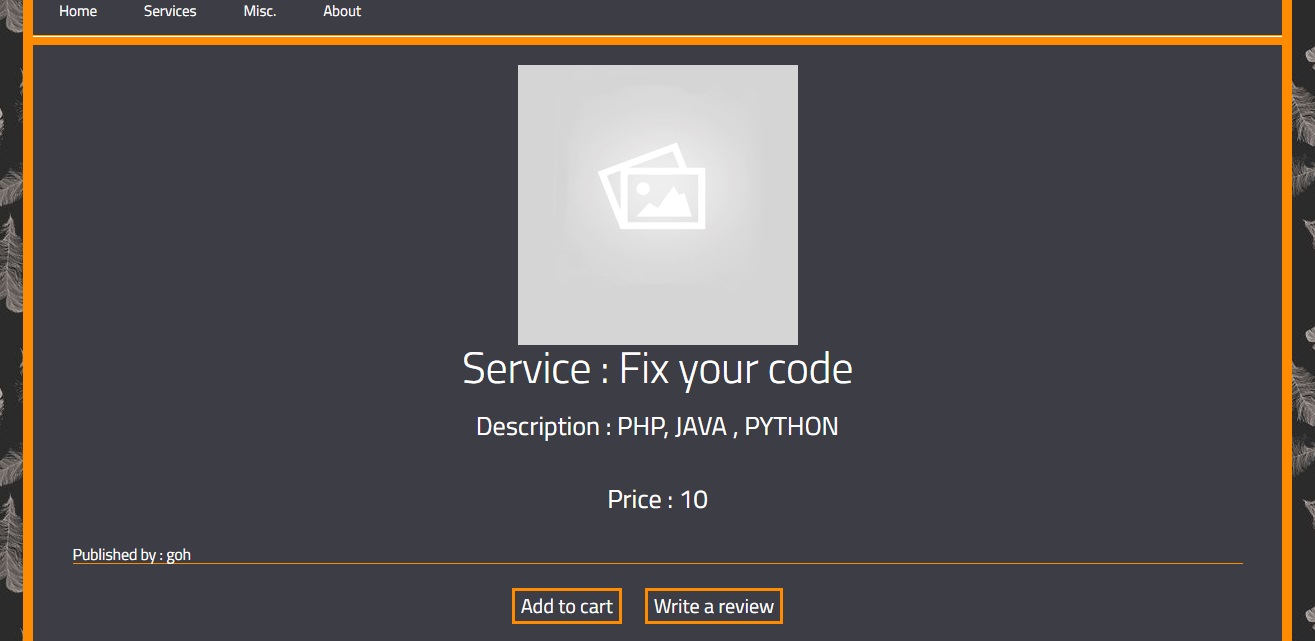
****

**Home Page**

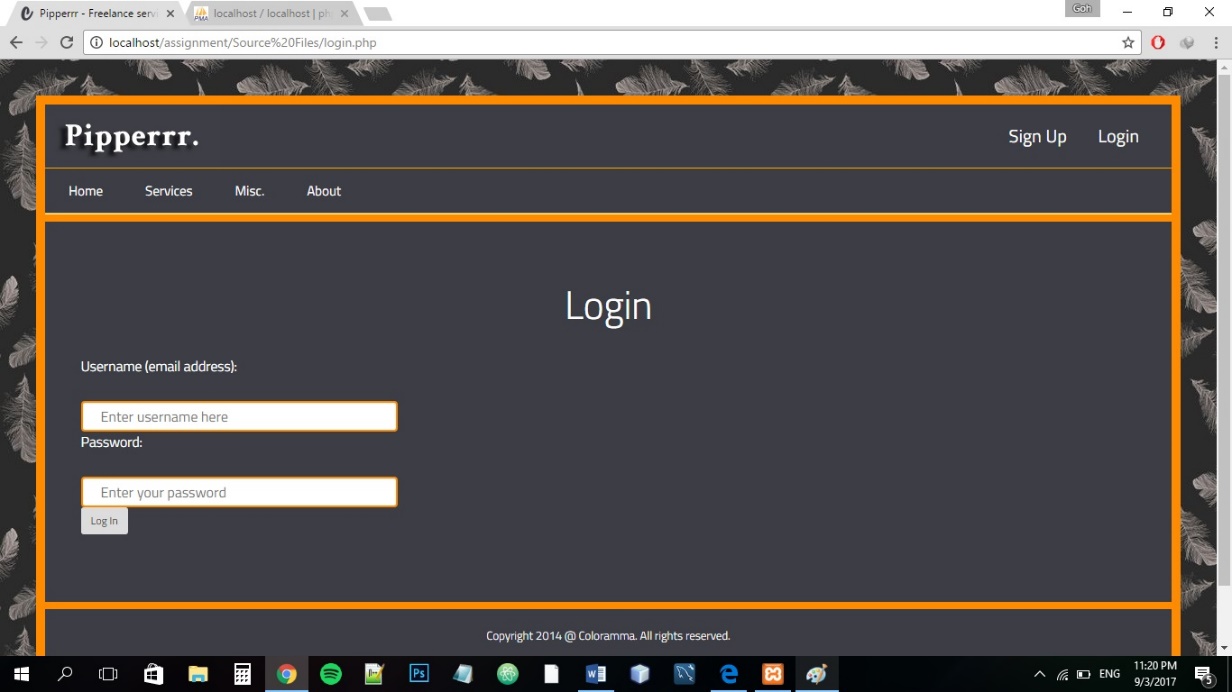
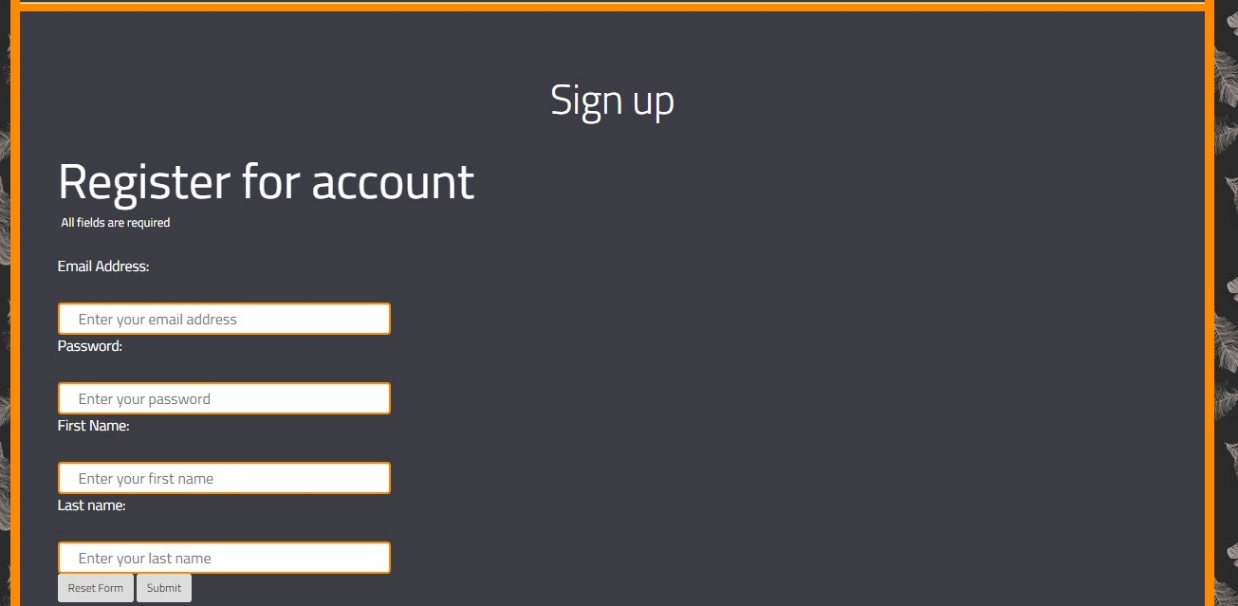
****

****

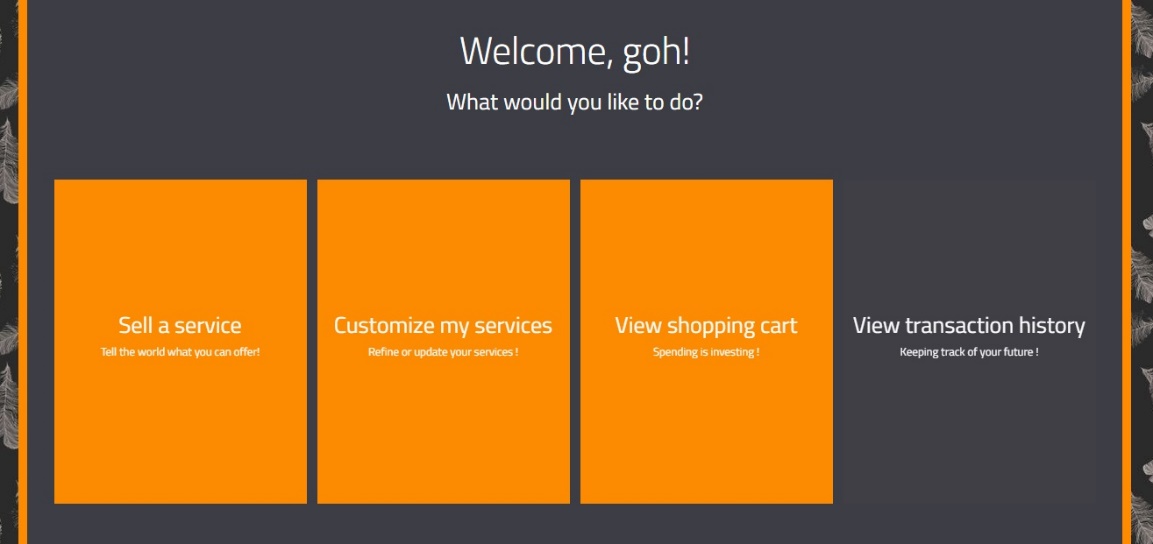
**Services Page**

****

**Details of a service page**

****

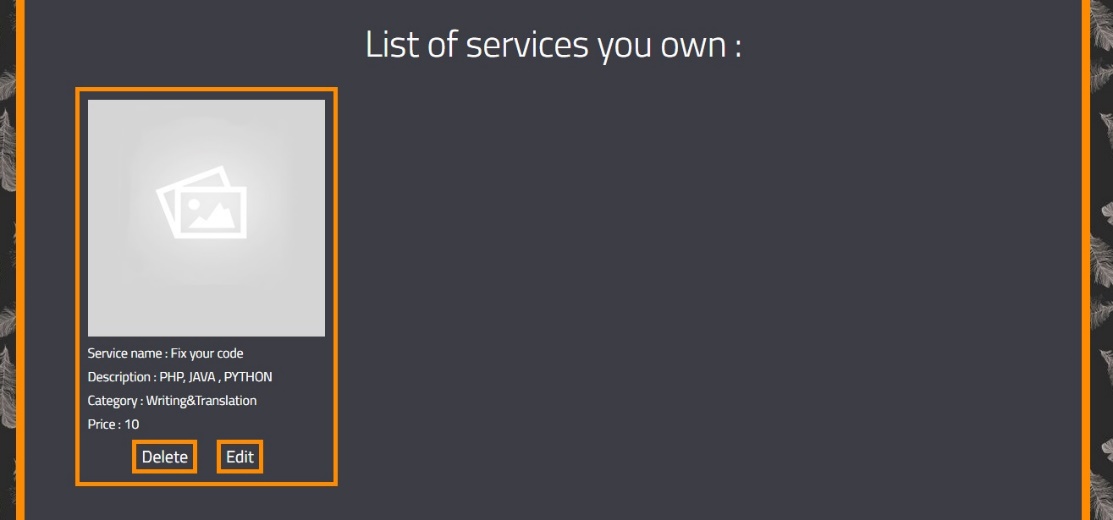
**Login and signup**

****

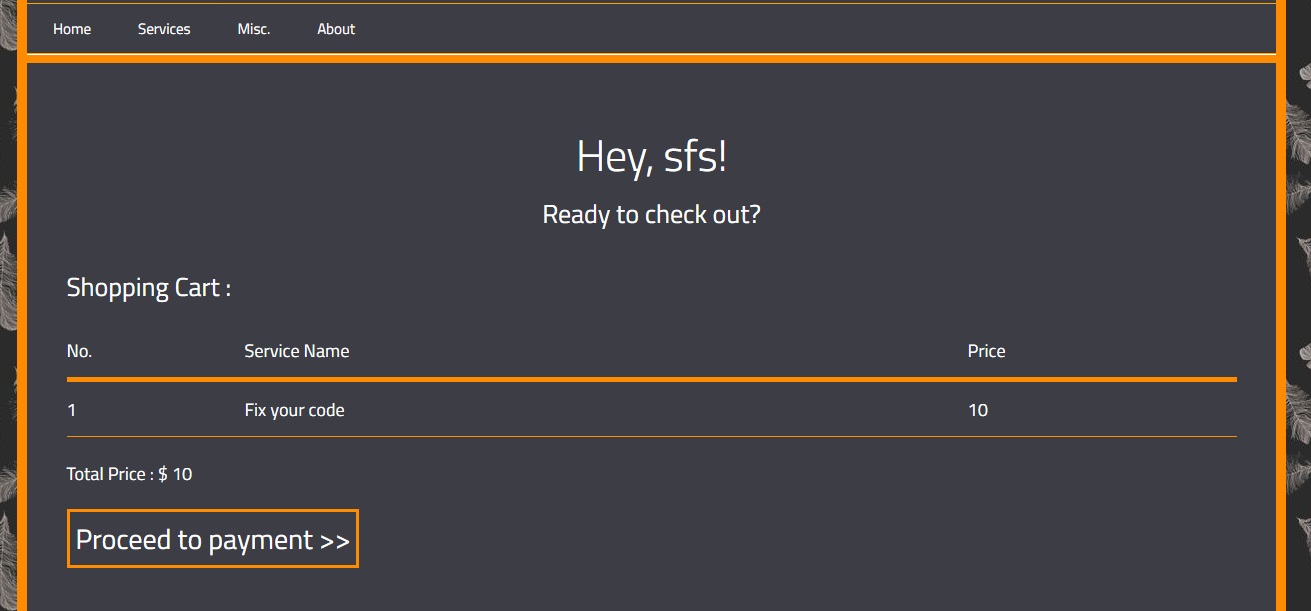
**Account page**

****

**Sell a service page**

****

**Customize service page**

****

**Cart Page**

**Test Document**

**Iteration 1**

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case Number | Use Case Name | Description | Student Name |
| - | Interface Design | Design of Pipperrr interface. | Goh Zen Hao |
| 1 | Register and log in | Enter account details to Pipperrr. | Goh Zen Hao / Wong Quon Shen |
| 2 | Add service | Enter service post details to Pipperrr to sell service. | Goh Zen Hao |
| 3 | Buy service | Purchase the service. | Wong Quon Shen |

**Test Objectives**

1. To detect as many bugs, errors, system failures, and system faults as the tester can possibly find within the source code.
2. To ensure the main functionalities of Pipperrr are able to work as intended and satisfy the user requirements.
3. To compile a clear documentation on the test analysis, results and test methodology.
4. To uncover any errors when integrating the system caused from unit testing.
5. To ensure the functional, non-functional, and performance requirements are met through the testing phase.

**Test Plan**

In the testing phase, there are three levels of tests which will be carried out and recorded.

|  |  |  |
| --- | --- | --- |
| Type of Test | When to initiate | Aim |
| Unit Testing | When one of the smallest part of the whole source code is completed. | Unit testing uses white box testing. Source code are to be scrutinized and tested for faults or errors. Code inspection is carried out to make sure the module is meeting the requirements which have been set. |
| Integration Testing | When each component is ready to be combined, integrated from Unit Testing. | Can implement either white box or black box testing. This allows us to test the overall structure of the system and ensures that all the parts from Unit testing are working well together. |
| System Testing | When all components are integrated together. | The system is tested in terms of functionality and performance. Ensuring the requirements of each use cases are met. The non-functional requirements are also tested. |

**Unit Testing**

|  |  |
| --- | --- |
| Test Case | Sign up validation |
| Test Data | |  |  | | --- | --- | | Email | [gohzenhao@gmail.com](mailto:gohzenhao@gmail.com) | | Password | Password | | FirstName | Goh | | LastName | Zen-Hao |   All details entered:   |  |  | | --- | --- | | Email |  | | Password | Password | | FirstName | Goh | | LastName | Zen-Hao |   Email absent:   |  |  | | --- | --- | | Email | [gohzenhao@gmail.com](mailto:gohzenhao@gmail.com) | | Password |  | | FirstName | Goh | | LastName | Zen-Hao |   Password Absent:  FirstName absent:   |  |  | | --- | --- | | Email | [gohzenhao@gmail.com](mailto:gohzenhao@gmail.com) | | Password | Password | | FirstName |  | | LastName | Zen-Hao |  |  |  | | --- | --- | | Email | [gohzenhao@gmail.com](mailto:gohzenhao@gmail.com) | | Password | Password | | FirstName | Goh | | LastName |  |   LastName absent :  Duplicate email entered:   |  |  | | --- | --- | | Email | [gohzenhao@gmail.com](mailto:gohzenhao@gmail.com) | | Password | Password | | FirstName | Jeff | | LastName | Bob | |
| Source Code |  |
| Expected output: | |  |  | | --- | --- | | Input | Output | | All details entered | Validation pass | | Email absent | Validation Fail. Popup display “ required field” | | Password absent | Validation fail. Popup display “ required field” | | firstName absent | Validation fail. Popup display “ required field” | | lastName absent | Validation fail. Popup display “ required field” | | Duplicate email | Validation fail. Message “ Email already taken” | |
|  | All details entered:    Email absent:    Password absent:    firstName absent    lastName absent    duplicate email : |

|  |  |
| --- | --- |
| Test case | Clear fields |
|  |  |
| Test Data | All details entered:   |  |  | | --- | --- | | Email | [gohzenhao@gmail.com](mailto:gohzenhao@gmail.com) | | Password | Password | | FirstName | Goh | | LastName | Zen-Hao | |
| Source Code |  |
| Expected output | All fields are empty |
| Actual output |  |

|  |  |
| --- | --- |
| Test case | Login Validation |
| Test Data | |  |  | | --- | --- | | Email | gohzenhao@gmail.com | | Password | password |   All details entered correctly   |  |  | | --- | --- | | Email |  | | Password | password |   Email absent   |  |  | | --- | --- | | Email | gohzenhao@gmail.com | | Password |  |   Password absent   |  |  | | --- | --- | | Email | gohzenhao@gmail.com | | Password | pas |   Details entered wrongly |
| Source Code |  |
| Expected output | |  |  | | --- | --- | | Input | Output | | All details correct | Success .Display : Home page | | Email absent | Fail. Pop up message “required” | | Password absent | Fail. Pop up message “required” | | Details incorrect | Fail. Error message “Incorrect email or password” | |
| Actual output | Details correct:    Email absent:    Password absent:    Details incorrect: |

|  |  |
| --- | --- |
| Test case | Create new service validation |
| Test Data | |  |  | | --- | --- | | serviceName | Fix your code | | description | PHP,JAVA,PYTHON | | Category | One of the radio buttons | | Price | 10 | | Image | - |   All details entered correctly  serviceName absent   |  |  | | --- | --- | | serviceName |  | | description | PHP,JAVA,PYTHON | | Category | One of the radio buttons | | Price | 10 | | Image | - |   Description absent:   |  |  | | --- | --- | | serviceName | Fix your code | | description |  | | Category | One of the radio buttons | | Price | 10 | | Image | - |   Category absent:   |  |  | | --- | --- | | serviceName | Fix your code | | description | PHP,JAVA,PYTHON | | Category |  | | Price | 10 | | Image | - |  |  |  | | --- | --- | | serviceName | Fix your code | | description | PHP,JAVA,PYTHON | | Category | One of the radio buttons | | Price |  | | Image | - |   Price absent: |
| Source Code |  |
| Expected output | |  |  | | --- | --- | | Input | Output | | Details correct | Success. Popup message “service created” | | serviceName absent | Fail. Pop up message “required” | | Description absent | Fail. Pop up message “required” | | Category absent | Fail. Pop up message “required” | | Price absent | Fail. Pop up message “required” | |
| Actual output | Details correct:    serviceName absent:    description absent:    category absent    Price absent |

|  |  |
| --- | --- |
| Test case | Add to cart |
| Test Data | |  |  | | --- | --- | | Email | gohzenhao@gmail.com | | Password | password |   Logged in as same user who created the service   |  |  | | --- | --- | | Email | Goh1996@gmail.com | | Password | password |   Logged in as different user who created the service |
| Source Code |  |
| Expected output | |  |  | | --- | --- | | Input | Output | | Add a service to cart that is created using the same account. | Failed. Display error message “ You cannot add a service that you own!” | | Add a service to cart that is created using a different account. | Success. Display message “ Added to cart !” | |
| Actual output | |  |  | | --- | --- | | Input | Output | | Add a service to cart that is created using the same account. | Failed. Display error message “ You cannot add a service that you own!” | | Add a service to cart that is created using a different account. | Success. Display message “ Added to cart !” | |

|  |  |
| --- | --- |
| Test case | Delete service |
| Test Data | |  |  | | --- | --- | | Email | gohzenhao@gmail.com | | Password | password |   Logged in as user :  Wants to delete the service he created :   |  |  | | --- | --- | | serviceName | Fixing code | | Description | PHP, JAVA, PYTHON | | Category | Programming&Tech | | Price | 20 | | Image | none | |
| Source Code |  |
| Expected output | |  |  | | --- | --- | | Input | Output | | User clicks on delete button of the service. | Success. The data of the service is deleted from database and is not shown on the interface. | |
| Actual output | |  |  | | --- | --- | | Input | Output | | User clicks on delete button of the service. | Success. The data of the service is deleted from database and is not shown on the interface. | |

**Integrated test type**

|  |  |
| --- | --- |
| Test case | Create new service and display it in interface |
| Test Data | |  |  | | --- | --- | | Email | gohzenhao@gmail.com | | Password | password |   Logged in as user : |
| Expected output | |  |  | | --- | --- | | Input | Output | | Sign in as gohzenhao@gmail.com | First name of user correctly displayed at top right corner. | |
| Actual output | Test successful.  Before singing in :    After signing in : |

|  |  |
| --- | --- |
| Test case | Create new service and display it in interface. |
| Test Data | |  |  | | --- | --- | | Email | gohzenhao@gmail.com | | Password | password |   Logged in as user :  Create new service :   |  |  | | --- | --- | | serviceName | Fixing code | | Description | PHP, JAVA, PYTHON | | Category | Programming&Tech | | Price | 20 | | Image | none | |
| Expected output | |  |  | | --- | --- | | Input | Output | | Create new service | Service added to database and displayed correctly in interface. | |
| Actual output | Test successful. Service created and displayed correctly.  Before creating service :    After creating service : |

|  |  |
| --- | --- |
| Test case | Add a service to cart and display it in interface |
| Test Data | |  |  | | --- | --- | | Email | Goh1996@gmail.com | | Password | password |   Logged in as user :  Add this service :   |  |  | | --- | --- | | serviceName | Fixing code | | Description | PHP, JAVA, PYTHON | | Category | Programming&Tech | | Price | 20 | | Image | none | |
| Expected output | |  |  | | --- | --- | | Input | Output | | Add service | Service added to cart and displayed correctly. | |
| Actual output | Test successful. Service added and displayed correctly.  Before adding service :    After adding service : |
| Test case | Total cost for cart |
| Test Data | |  |  | | --- | --- | | Email | Goh1996@gmail.com | | Password | password |   Logged in as user :  Add this service :   |  |  | | --- | --- | | serviceName | Fixing code | | Description | PHP, JAVA, PYTHON | | Category | Programming&Tech | | Price | 20 | | Image | none | |
| Expected output | |  |  | | --- | --- | | Input | Output | | Add another service to cart | Total cost correctly returned. | |
| Actual output | Test successful. Service added and total cost calculated correctly.  Before adding another service :    After adding another service : |

**System testing**

|  |  |
| --- | --- |
| Use case #1 | Login |
| Description | To allow user to login after signing up |
| Screenshot |  |
| Test Action | User enters email and password and press log in. |
| Expected output | After having validate the email and password. The system directs the user to the homepage. The top right corner displays the user’s firstname and a button to manage his/her account and sign out. |
| Actual output | The system directs the user to the homepage. The top right corner displays the user’s firstname and a button to manage his/her account and sign out. |

|  |  |
| --- | --- |
| Use case #2 | Sell service |
| Description | To allow user to create a new service and publish it. |
| Screenshot |  |
| Test Action | User enters service details. |
| Expected output | The service is created and added to the list of services. The service is also added to the user’s account to allow him to customize. |
| Actual output | The service is created and added to the list of services. The service is also added to the user’s account to allow him to customize. |

|  |  |
| --- | --- |
| Use case #3 | Buy service |
| Description | To allow user to add the service to the cart. |
| Screenshot |  |
| Test Action | User clicks on the “ADD TO CART” Button. |
| Expected output | A message displaying “ Added to cart” is shown and the service is added to the shopping cart. |
| Actual output | A message displaying “ Added to cart” is shown and the service is added to the shopping cart. |

**Non-functional requirements test**

In each of the use case tests (Reference to assignment 1):

|  |  |
| --- | --- |
| Usability: The application interface is simplified. Not too crowded. | Successful. |
| Performance: Respond to user signup, login, create service within 3 second. | Successful. |
| Platform compatibility: Run on different web browsers without any errors. | Partially Successful. Encounter some error when running on IE. |
| Efficiency: Functions can be done with minimum steps. | Successful. |
| Security: Website should be encrypted. | Requirement not met. Have not encrypted the codes. |
| Disaster recovery: Must have backup of database. | Successful. |

**Test Analysis Report**

**Report and summary**

Coming up with a platform to sell/buy services was thought to be easy initially. However, implementing the functions that the website provides proved to be quite a challenging task. During implementation, we encountered many errors. We did not manage to fully complete Iteration 1, thus may result in delay of the progress of the Gantt chart. Upload image and displaying them is still under testing and implementation. Resolving this issue may be performed in Iteration 2.

Overall, the tests were successful in identifying important flaws that we may have overlooked. Besides the issue with displaying images, other requirements set for Iteration 1 have been met.

**Updated Gantt Chat**

