**Pasale.com**



**Ayush Pandey**

College ID: [170279]

NCC ID: [00174308]

Computing Project

Level 5 Diploma in Computing

Softwarica College of IT & E-Commerce

Kathmandu, Nepal

July 28, 2019

# Acknowledgement

I am eternally indebted to all those who supported me while designing and developing this project. I am thankful to all who taught me and whose feedbacks went to make the project successful. This especially includes Softwarica College of IT and E-Commerce for providing such a good ecosystem for learning.

I am thankful to the module teacher Sudeep Lal Bajimaya for guiding me throughout the project and providing tips for analyzing and designing the project and also helping in the documentation of the project.

I am also grateful towards other teacher such as Manoj Shrestha sir for his training on searching and referencing internet materials to make the documentation of the system more applicable.

Lastly, I want to show my gratitude towards my family and friend who taught me and were supportive throughout the entirety of the project

Ayush Pandey

07/28/2019

Kathmandu, Nepal

# Abstract

This report documents all the activities executed in each of the phases of software development lifecycle of the project titled – ‘Pasale.com’. All the requirements, use cases with scenario description, system architecture, the static and dynamic models and implementation details along with testing report and user manual is included in this report.

This project is an ecommerce application which is designed to be used for buying products. It allows user to buy product at the comfort of their place. In simple terms, it helps users to buy product that they need by providing all information about the product.

The technical aspect of the system consists Laravel, a popular PHP framework as back-end language with some packages. Also, HTML, CSS and JavaScript was used for front end programming. For database, MySQL was used.

# Table of Contents

[Acknowledgement 2](#_Toc15186111)

[Abstract 3](#_Toc15186112)

[Table of Contents 4](#_Toc15186113)

[Table of Figures 6](#_Toc15186114)

[Introduction 8](#_Toc15186115)

[1.1: Background 8](#_Toc15186116)

[1.2: Justification 8](#_Toc15186117)

[1.3: Overview 8](#_Toc15186118)

[1.4: Aims 9](#_Toc15186119)

[1.5: Objective 9](#_Toc15186120)

[2: Analysis 10](#_Toc15186121)

[2.1: Introduction 10](#_Toc15186122)

[2.2: Analysis Methodology 10](#_Toc15186123)

[2.3: Feasibility Study 12](#_Toc15186124)

[2.4: Software Requirement Specification (SRS) 12](#_Toc15186125)

[2.4.1: Functional Requirement 13](#_Toc15186126)

[2.4.2: Non-Functional Requirements 15](#_Toc15186127)

[2.4.3: MoSCoW Prioritization 16](#_Toc15186128)

[2.4.4: Hardware/Software Specification 18](#_Toc15186129)

[2.4.5: Use-case Diagram 19](#_Toc15186130)

[2.4.6: Initial Class diagram 26](#_Toc15186131)

[3: Design 28](#_Toc15186132)

[3.1: Introduction 28](#_Toc15186133)

[3.2: Project Design Plan 28](#_Toc15186134)

[3.2.1: Structural Modelling 28](#_Toc15186135)

[3.2.2: Behavioral Modelling 30](#_Toc15186136)

[3.2.2.1: Activity Diagram 30](#_Toc15186137)

[3.2.2.2: Sequence Diagram 33](#_Toc15186138)

[3.2.3: Database Modelling 39](#_Toc15186139)

[3.2.3.1: Entity Relationship Diagram 39](#_Toc15186140)

[3.2.3.2: Data Dictionary 40](#_Toc15186141)

[3.2.4: Architectural Modelling 42](#_Toc15186142)

[3.2.5: UI Modelling 42](#_Toc15186143)

[3.2.5.1: Prototypes 42](#_Toc15186144)

[4: Implementation 45](#_Toc15186145)

[5: Testing 46](#_Toc15186146)

[6: Other Project Issues 54](#_Toc15186147)

[6.1: Limitation of the Project 54](#_Toc15186148)

[6.2: Future Work 54](#_Toc15186149)

[6.3: Risk Management 54](#_Toc15186150)

[6.4: Configuration Management 57](#_Toc15186151)

[6.5: User Manual 57](#_Toc15186152)

[Conclusion 59](#_Toc15186153)

[Appendix 60](#_Toc15186154)

# Table of Figures

[Figure 1: Admin DFD 5](#_Toc15184529)

[Figure 2: User DFD 5](#_Toc15184530)

[Figure 3: Sign in and Registration Use Case 13](#_Toc15184531)

[Figure 4: Admins Use Case 15](#_Toc15184532)

[Figure 5: Users interaction Use Case 17](#_Toc15184533)

[Figure 6: User profile management Use Case 19](#_Toc15184534)

[Figure 7: Initial Class Diagram 21](#_Toc15184535)

[Figure 8: Class diagram notations 23](#_Toc15184536)

[Figure 9: Final Class diagram 23](#_Toc15184537)

[Figure 10: Activity diagram notations 24](#_Toc15184538)

[Figure 11: User Activity Diagram 25](#_Toc15184539)

[Figure 12: Admin Activity diagram 26](#_Toc15184540)

[Figure 13: Sequence diagram notations 27](#_Toc15184541)

[Figure 14: Register sequence 28](#_Toc15184542)

[Figure 15: Login Sequence 29](#_Toc15184543)

[Figure 16: Shopping cart sequence 30](#_Toc15184544)

[Figure 17: User Sequence diagram 31](#_Toc15184545)

[Figure 18: Admin Sequence diagram 32](#_Toc15184546)

[Figure 19: ER Diagram 33](#_Toc15184547)

[Figure 20: Landing page 36](#_Toc15184548)

[Figure 21: Product Page 37](#_Toc15184549)

[Figure 22: Shopping Cart 37](#_Toc15184550)

[Figure 23: Checkout form 38](#_Toc15184551)

[Figure 24: Test T01 40](#_Toc15184552)

[Figure 25: Test T02 41](#_Toc15184553)

[Figure 26: Test T02 fixed 41](#_Toc15184554)

[Figure 27: Test T03 42](#_Toc15184555)

[Figure 28: Test T03 passed 42](#_Toc15184556)

[Figure 29: Test T04 43](#_Toc15184557)

[Figure 30: Test T04 passed 43](#_Toc15184558)

[Figure 31: Test T05 44](#_Toc15184559)

[Figure 32: Test T05 passed 44](#_Toc15184560)

[Figure 33: Testing 45](#_Toc15184561)

[Figure 34: Test U01 45](#_Toc15184562)

[Figure 35: Test U02 46](#_Toc15184563)

[Figure 36: Test U03 47](#_Toc15184564)

[Figure 37: Test U04 47](#_Toc15184565)

[Figure 38: Project folder tree 51](#_Toc15184566)

[Figure 39: User Manual/Help 52](#_Toc15184567)

# Introduction

In the growing trend of online services, one of the major services is an ecommerce platform. Every people are interested in buying the products they need with a click of a button in the comfort of their place.

With the growing ecommerce market all over the world, this project is commenced so that users can buy the product they need seeing all the product details with ease. It is also easier to maintain commerce record in computer rather than a manual.

## 1.1: Background

This system tries to replace the concept of buying and selling products by computerizing it which consequently effectively manages time and cost. It replaces traditional and manual system of purchasing products. In general, this system fulfills the user needs without compromising quality, cost and time.

This project is a web application that helps the user to purchase products by ease without manual physically intensive shopping. All the information related to users, products, orders is kept in a secure environment

## 1.2: Justification

This project is a web application which helps user to buy a product from the comfort of their place rather than going to a shop. This project helps in computerizing this process. All the information is kept and managed so user can benefit from the service.

## 1.3: Overview

This project helps the user to buy products online by seeing every details of the product. Moreover, there are also some features that helps to ease this process like shopping cart, payment gateway integration, reliable search options, etc. This application consists of following feature:

* Validation and verification
* User registration and login
* Searching Products
* Integration of Shopping cart
* Payment gateway integration using stripe
* Ordering the product
* Guest checkout (buy without logging in the application)
* Discount coupon codes

## 1.4: Aims

This project aims to create a hub where all market product will be available at the best price possible.

## 1.5: Objective

* to implement the new system replacing traditional commerce system
* to maintain user information in a secured encrypted state
* to provide different features to users so they can easily purchase products without any hassle
* to maintain all products, category, orders data in a secured and encrypted environment

# 2: Analysis

## 2.1: Introduction

Analysis is a detail study of a complex problem to establish the understanding of the problem which further helps as the basis for problem solving and decision making. Analysis is performed to divide a big complex problem into subproblems to have the better understanding. Analysis helps in finding all the requirements. In software development, analysis helps in gathering user requirements, software functions, etc. which helps in better and systematic development of the software. (ReQtest, 2018)

## 2.2: Analysis Methodology

The goal of analysis is to fully understand the systems requirement and then formalize it. For this any appropriate analysis methodology should be used. In any analysis methodology a certain defined step is carried to ensure the maximum understanding of the system. For choosing an analysis methodology we must first know about the system about to be created.

For this project I am using **Hard system approach methodology**. This methodology is a highly structured and is used to address both qualitative and quantitative problems of the system. It is the approach that follows a set number of logical steps, which is often times iterative, analyze the system and its functions. This methodology uses Structured Systems Analysis and Design Method (SSADM). In this approach a Data Flow Diagram (DFD) should be created which shows the flow of data of the system. (UKEssays, 2017)

For the software development methodology, I have chosen Iterative process. This will be a good choice as the system now is small but is scalable for future upgrades.



Figure 1: Admin DFD

The above DFD shows the flow of admin. The admin should be able to go to a dashboard after logging in. There they can view, edit and delete category, products, users and can also view the order of users.



Figure 2: User DFD

The above DFD shows the flow of user. The user should be able to log in. Then, they can view products and add it to a cart to finally buy it by providing user details.

## 2.3: Feasibility Study

Feasibility study is a part of analysis which evaluates the systems potential for success, therefore evaluates five types of study that includes technical feasibility, social feasibility, financial feasibility, legal feasibility. This is usually done to find any problems related to systems development. This study also provides the organization information that could manage or prevent any occurring or potential risks.

* **Economic Feasibility**: This study typically involves cost/benefit analysis of the system. Furthermore, it helps the organization to determine the cost, benefits and viability of the system before any resources is allocated. Here, cost is not analyzed as it is an academic project. For my project, this feasibility study helps me to determine the positive benefits that the system will provide, and whether the system is scalable as per the increment in users.
* **Social Feasibility**: This study involves the study of social factors such as political condition, environment around the system, cultural factors, etc. For my project, this feasibility study helps to determine if my there will be any effect in my product due to availability of similar products. It also helps in studying whether the product will be sustainable in the market.
* **Technical Feasibility**: This study assesses the technical resources available for the completion of the project. It also helps the organization for the evaluation of hardware, software, and other technology requirement of the proposed system. For my project, this feasibility study helps to evaluate if the technical resource is available, is the system going to be compatible with modern emerging technologies, etc.
* **Legal Feasibility**: This study investigates if any aspects of the system is in conflict with legal requirements like data protection act, zoning laws or social media law. For my project, this feasibility study helps to evaluate if any change in government policy affects the system as there's a big role of trade laws and taxation.

## 2.4: Software Requirement Specification (SRS)

Software Requirement Specification is a detail documentation of the system that is being developed including its functional and non-functional requirements and other relevant cases that helps in making the quality of the system better. It contains all the requirements required for system development. (Inflectra, 2018)

### 2.4.1: Functional Requirement

A functional requirement describes what the system should do. It is the whole function of the system and how it works. It describes the behavior between inputs and outputs. The functional requirements of this project are listed below:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **Title** | **Description** | **Rational** | **Dependencies** |
| **FR1** | Admin Registration/ Login | Administrator should be able to login | To get access to the admin dashboard | N/A |
| **FR2** | Authentication | The data credentials should be authenticated | To give access to the authorized user only | FR1 |
| **FR3** | Add a Category | Admin should be able to add a new category | To add a category for dividing different types of product | FR2 |
| **FR4** | Add Brand | Admin should be able to add a new brand | To add a brand which provides the product | FR4 |
| **FR5** | Add Product | Admin should be able to add a new product | To add products in order to sell | FR5 |
| **FR6** | Update Category | Admin should be able to update an existing category | To edit existing detail of the category | FR3 |
| **FR7** | Update Brand | Admin should be able to update an existing brand | To edit existing detail of the brand | FR4 |
| **FR8** | Update Product | Admin should be able to update an existing product | To edit existing detail of the product | FR5 |
| **FR9** | Delete Category | Admin should be able to delete an existing brand | To delete an unused category | FR3 |
| **FR10** | Delete Brand | Admin should be able to delete an existing brand | To delete obsolete brands | FR4 |
| **FR11** | Delete Product | Admin should be able to delete an existing product | To delete any product | FR5 |
| **FR12** | User Signup | User should be able to register their account | To add their information for buying products |  |
| **FR13** | User Login | User should be able to be logged in. | To buy the product | FR12 |
| **FR14** | View Category, Brand, Product | User should be able to view all the categories, brands and products. | To view all the categories, brand and product | FR3, FR4, FR5 |
| **FR15** | Search Product | User should be able to search for products | To search for a required product | FR3, FR4, FR5 |
| **FR16** | Buy Product | User should be able to buy the product | To buy any product | FR3, FR4, FR5, FR13 |
| **FR17** | Add a Cart | User should have to function to add items to a cart and buy in bulk | To add products to a cart and buy all at once | FR3, FR4, FR5, FR13 |
| **FR18** | View/Update User Information | User should be able to view and update their information | To edit their own information | FR13 |
| **FR19** | Delete User | User should be able to delete their account | To delete their information if not used | FR13 |
| **FR20** | Cancel the bought product | User should be able to cancel any bought products. | To cancel any bought products if not needed anymore | FR13, FR16, FR17 |

### 2.4.2: Non-Functional Requirements

Non-Functional requirement should specify how the system should behave. It is as any requirement that is the quality attribute of any system. This is necessary to make the system more reliable, secure and effective. The non-functional requirements of this project are:

|  |  |  |
| --- | --- | --- |
| ID | Title | Description |
| NFR1 | Efficiency | The system should be efficient for handling high number of requests |
| NFR2 | Security | The system should be secured and admin right should be authorized to admin only |
| NFR3 | Scalability | The system should be able to handle the growing data capacity |
| NFR4 | Data Integrity | The system should assure the stored data is authentic and accurate |
| NFR5 | Availability | The system should be available all the time |
| NFR6 | Reliability | The system should be reliable to perform functions without any failure |
| NFR7 | Maintainability | The system should be easily maintainable with easy error detection and fixes |
| NFR8 | Manageability | The system and its components should be easy to manage |
| NFR9 | Interoperability | This feature helps to facilitate the operation of system with other compatible systems |

### 2.4.3: MoSCoW Prioritization

MoSCoW Prioritization is a prioritization technique for managing the requirements of the system. It prioritizes the requirements in three level of importance: must have, should have, could have and would/won’t have. (Madsen, 2017)

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Requirements** | **MoSCoW** | **Rational** |
| FR1 | Admin Login | M | For admin to get access to the system |
| FR2 | Authentication | M | For authorized user to get access |
| FR3 | Add a Category | M | To add a category |
| FR4 | Add Brand | S | To add a brand |
| FR5 | Add Product | M | To add a product |
| FR6 | Update Category | S | To update a category information |
| FR7 | Update Brand | S | To update a brand information |
| FR8 | Update Product | S | To update product information |
| FR9 | Delete Category | M | To delete obsolete category |
| FR10 | Delete Brand | S | To delete obsolete brand |
| FR11 | Delete Product | M | To delete obsolete products |
| FR12 | User Signup | M | For users to register their details in the system |
| FR13 | User Login | M | To log in the system by user |
| FR14 | View Category, Brand, Product | M | To view all categories and products |
| FR15 | Search Product | S | To be able to search for required product |
| FR16 | Buy Product | M | To buy a product |
| FR17 | Add a Cart | S | To add products to cart and buy in bulk |
| FR18 | View/Update User Information | S | To update the user’s information |
| FR19 | Delete User | M | To delete their account if not needed anymore |
| FR20 | Cancel the bought product | W | For cancelling the product which has been ordered. |
| NFR2 | Security | M | To keep the data securely |
| NFR4 | Data Integrity | M | To make sure the data is correct |
| NFR5 | Availability | M | To make sure the application is available anytime |
| NFR3 | Scalability | S | To make sure the system performance won’t degrade after application growth |
| NFR7 | Maintainability | M | To maintain the system more efficiently |
| NFR8 | Manageability | M | To manage the system |
| NFR9 | Interoperability | S | To integrate the use of this system with other system |

### 2.4.4: Hardware/Software Specification

For developing and accessing the system**,** certain hardware and software is required. The hardware specification for this project is given below:

**Hardware Specification**

* Processor: any Intel or AMD based CPU with 1.5GHz clock speed
* RAM: 4GB or higher
* Display: SVGA with 1024\*768 pixels +
* Peripherals: Mouse and Keyboard

**Software Specification**

* OS: Windows 7 or higher or any Linux distro or Mac OS
* Browser: Google Chrome v74 or Mozilla Firefox or Microsoft Edge
* Server: XAMPP 7.3.2
* Database: MySQL
* IDE/Editor: Visual Studio Code

### 2.4.5: Use-case Diagram

A Use-case diagram is a behavioral diagram that represents the systems interaction with the user (known as actors). (Rouse, 2007) The main components of use case diagram are:

* Actors: It represent the user of the system.
* Use cases: It defines the process of the function in the system.
* Associations: This shows the relation of use cases and actors.
* System boundary box: It shows the system and its functions inside of it.

The diagrams below display the use-case for Pasale.com where two different actors access the system with different rights.

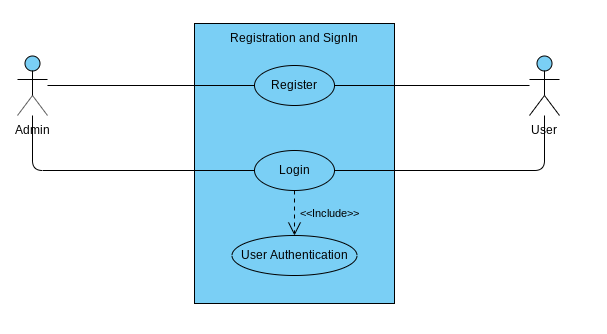


Figure 3: Sign in and Registration Use Case

|  |  |
| --- | --- |
| **S. N** | **Use Case: Sign In** |
| **Explanation** | The basic requirement of application through which user can access  the application with valid credential in-order to do further process such  as buy products, view profile, etc. |
| **Primary Actor** | User |
| **Secondary Actor** | Admin |
| **Main Flow** | 1. User login with the valid credential.  2. User navigated to the home page.  3. User proceeds the further process. |
| **Second Flow** | 1. User input invalid credential  2. System detects and validate the user input.  3. The possible validation error message is shown to the user.  4, User is expected to enter correct credential.  5. Repeat from step 2 if wrong credential input otherwise |

|  |  |
| --- | --- |
| **S. N** | **Use Case: Registration** |
| **Explanation** | To login the application and manipulate the require information user  need to register first. |
| **Primary Actor** | User |
| **Secondary Actor** | None |
| **Main Flow** | 1. User clicks on signup button to registration.  2. A signup form will appear.  3. User needs to fill up the required data into the field and submit.  4. System validates and verify the input data.  5. System store the input data into the database.  6. User will redirect to the login page to login into the application. |
| **Second Flow** | 1. User input invalid data to the field.  2. System validates and verify the data.  3. System show appropriate validation message.  4. User is required to input correct data into the field.  5. The system will repeat the process from step 4 until correct  data input. |

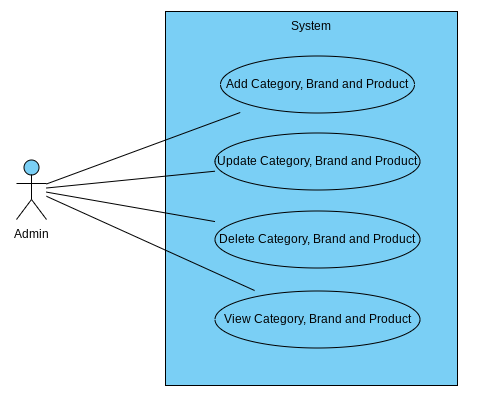


Figure 4: Admins Use Case

|  |  |
| --- | --- |
| **S. N** | **Use Case: Add Category, Brand, Product** |
| **Explanation** | Admin needs to add a category, brand and product with necessary  details in order to sell goods user. |
| **Primary Actor** | Admin |
| **Secondary Actor** | None |
| **Main Flow** | 1. Admin login to the system with the valid credential.  2. The admin navigates to the admin panel in order to add.  3. Admin click add button to add and save into the database. |
| **Second Flow** | N/A |

|  |  |
| --- | --- |
| **S. N** | **Use Case: Update Category, Brand, Product** |
| **Explanation** | This use case needs to maintain data integrity whenever there in  incorrect data input. |
| **Primary Actor** | Admin |
| **Secondary Actor** | None |
| **Main Flow** | 1. Admin login with the valid credential.  2. After login admin navigates to the admin panel where the  admin clicks on the update button to edit details.  3. The admin submits the edit form.  4. The system validates and verifies the input data.  5. System store the changes into the database.  6. Admin will redirect to the existing page with data. |
| **Second Flow** | N/A |

|  |  |
| --- | --- |
| **S. N** | **Use Case: Delete Category, Brand, Product** |
| **Explanation** | This use case explains how admin can delete any data if it is not needed anymore |
| **Primary Actor** | Admin |
| **Secondary Actor** | None |
| **Main Flow** | 1. Admin login into the application with the valid credential.  2. Admin check if the category, brand or product is not needed.  3. If it is not needed admin can successfully delete it from the |
| **Second Flow** | N/A |

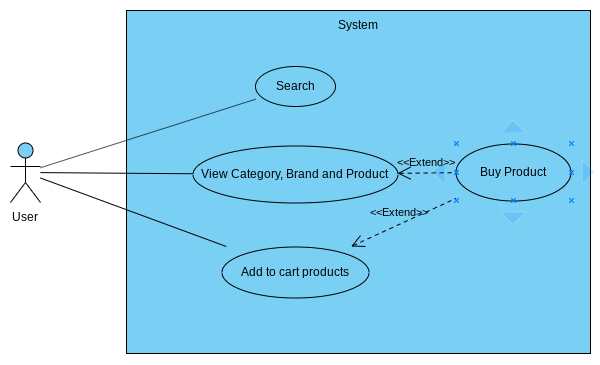


Figure 5: Users interaction Use Case

|  |  |
| --- | --- |
| **S. N** | **Use Case: Search Product** |
| **Explanation** | It is and important feature which helps user to look for the product  they need. So, to get a specific product instant with the best possible  matches for the given input from a large pool of data. |
| **Primary Actor** | User |
| **Secondary Actor** | None |
| **Main Flow** | 1. User must visit the application/website.  2. In the header there is a search bar where input box is present  1. User must type the name to get any result |
| **Second Flow** | N/A |

|  |  |
| --- | --- |
| **S. N** | **Use Case: View Category, Brand, Product** |
| **Explanation** | User should be able to see all available category, brand and products  in the web application |
| **Primary Actor** | User |
| **Secondary Actor** | None |
| **Main Flow** | 1. There is no need to login into the application in order to view  the category, brand and product |
| **Second Flow** | N/A |

|  |  |
| --- | --- |
| **S. N** | **Use Case: Buy Product** |
| **Explanation** | The main feature of the system is to buy a product. |
| **Primary Actor** | User |
| **Secondary Actor** | None |
| **Main Flow** | 1. User login with the valid credential.  2. After login user navigates to the product page to see the  product details.  3. User buys the product after providing important details. |
| **Second Flow** | 1 User input wrong credential.  2 The system validates and verifies the input data.  3 System show appropriate validation message.  4 User is required to input correct data.  5 Repeat from step 1 from the main flow section until valid  credential was input |

|  |  |
| --- | --- |
| **S. N** | **Use Case: Add Cart** |
| **Explanation** | An additional feature of the system is to add a cart and buy product in  bulk. |
| **Primary Actor** | User |
| **Secondary Actor** | None |
| **Main Flow** | 1. User login with the valid credential.  2. After login user navigates to the product page to see the  product details.  3. User adds products to cart.  4. User checks out the cart after providing important details. |
| **Second Flow** | 1 User input wrong credential.  2 The system validates and verifies the input data.  3 System show appropriate validation message.  4 User is required to input correct data.  5 Repeat from step 1 from the main flow section until valid  credential was input. |

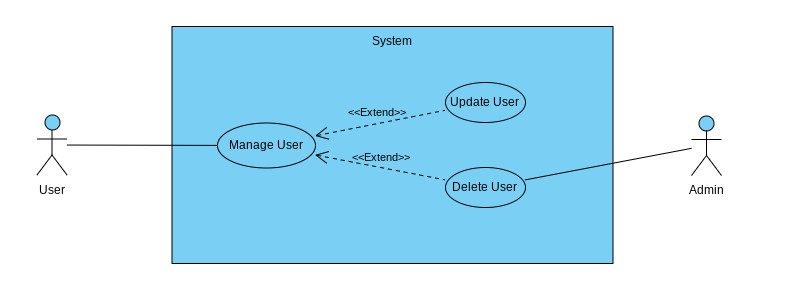


Figure 6: User profile management Use Case

|  |  |
| --- | --- |
| **S. N** | **Use Case: View Profile** |
| **Explanation** | A fundamental requirement of the application, the user has the right to  visit and view their individual profile. |
| **Primary Actor** | User |
| **Secondary Actor** | None |
| **Main Flow** | 1. User login into the system with the valid credential.  2. User navigates to the profile page and view profile. |
| **Second Flow** | N/A |

|  |  |
| --- | --- |
| **S. N** | **Use Case: Update Profile** |
| **Explanation** | User may input invalid or incorrect data so to maintain data integrity  update case is required. |
| **Primary Actor** | User |
| **Secondary Actor** | None |
| **Main Flow** | 1. User login with the valid credential.  2. User navigates to the profile page.  3. User click on the update button and the form will open with  existing data.  4. User edit the require data into the field and click on the submit  button to submit the profile form.  5. System validates and verify the input data.  6. System updates the data into the database.  7. System redirects to the existing page. |
| **Second Flow** | N/A |

|  |  |
| --- | --- |
| **S. N** | **Use Case: Delete Profile** |
| **Explanation** | A fundamental requirement of the system, every user has the right to  delete their individual profile if the user doesn’t want to use the  application. |
| **Primary Actor** | User |
| **Secondary Actor** | None |
| **Main Flow** | 1. User login to the system with the valid credential.  2. User navigates to the profile page.  3. User click delete button to delete their profile. |
| **Second Flow** | N/A |

### 2.4.6: Initial Class diagram

#### 2.4.6.1: Natural Language Analysis (NLA)

Pasale.com is an online multi-vendor ecommerce platform. Different vendors can sell their products through this platform. All the products are separated in different categories and they also represent brands if they are. Administrator should be able to add all the things from the admin panel. Users should be able to search for the products they need. User should be able to create a cart and buy the products in bulk. User should also be able to update their information if any needed.

The nouns and verbs are separated from above scenario to further help in finding required classes and methods.

**Nouns**: Product, Category, Brand, Cart, Vendor, User, Admin, Order

**Verb**: sell, buy, update, delete, create, insert, edit, view

After filtering the nouns for repetition and ambiguity, the candidate class and all the potential methods are generated

**Candidate** **Class**

|  |  |  |
| --- | --- | --- |
| Category | Brand | Product |
| Cart | User | Admin |

**Potential** **Methods**

|  |  |  |
| --- | --- | --- |
| Sell products | Buy products | Insert category, brand, product |
| Create Cart | Update category, brand, product | Delete category, brand, product |
| Search |

#### 2.4.6.2: Initial Class Diagram

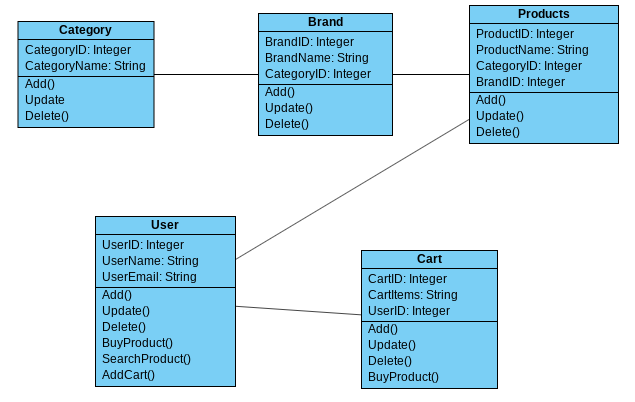


Figure : Initial class Diagram

# 3: Design

## 3.1: Introduction

Design is transforming the software and user's requirement to any suitable form of representation like step processes or diagrams that further helps the developer in implementing the software. It generally shows the workflow of the system

## 3.2: Project Design Plan

Certain modelling methods are used while designing the system. All the methodologies are selected based on the software that is being designed. The modelling that are being designed are listed below:

* Structural Modelling
* Behavioral Modelling
* Database Modelling
* Architectural Modelling
* UI Modelling

### 3.2.1: Structural Modelling

Structural modelling is referred as a diagram that represents how the individual components interact to create the whole system. In this project, structural modelling has been utilized by using Class diagram and Data flow diagram.

#### 3.2.1.1: Final Class Diagram

A class diagram is a static diagram that shows the systems structure representing the classes, their properties, all operations and the relationship between the classes.

##### 3.2.1.1.1: Justification

* Models the static view of the system
* Shows the relationship between all classes
* Shows the datatype used by all attributes

##### 3.2.1.1.2: Notation

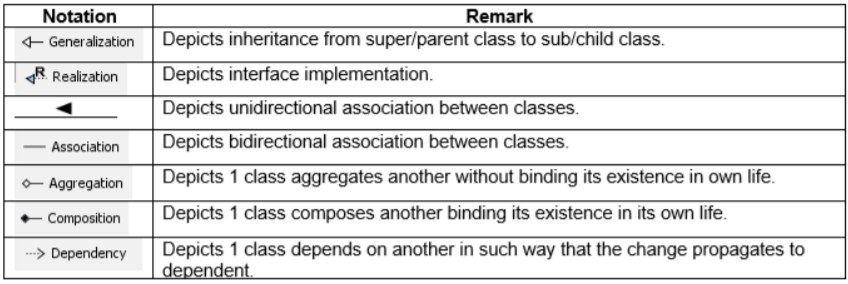


Figure : Class diagram notations

##### 3.2.1.1.3: Diagram



Figure : Final Class diagram

The above class diagram is made using Visual Paradigm and is based on UML. The diagram explains all the class and its relation with other classes.

## 3.2.2: Behavioral Modelling

Behavioral Modelling is the design diagram that represents the dynamic features of the system showing business processes.

### 3.2.2.1: Activity Diagram

Activity diagram is a behavioral model that shows the flow from one activity to another in the system.

#### 3.2.2.1.1: Justification

* Represents dynamic flow of data between different activities
* Models step by step flow with used business logic
* Models complex workflow in operations between/within activities

#### 3.2.2.1.2: Notation

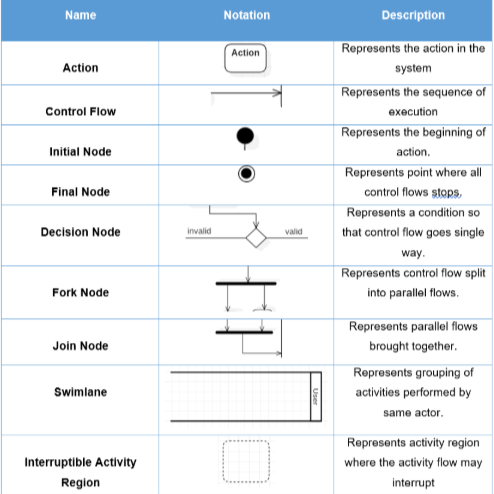


Figure : Activity diagram notations

#### 3.2.2.1.3: Diagram

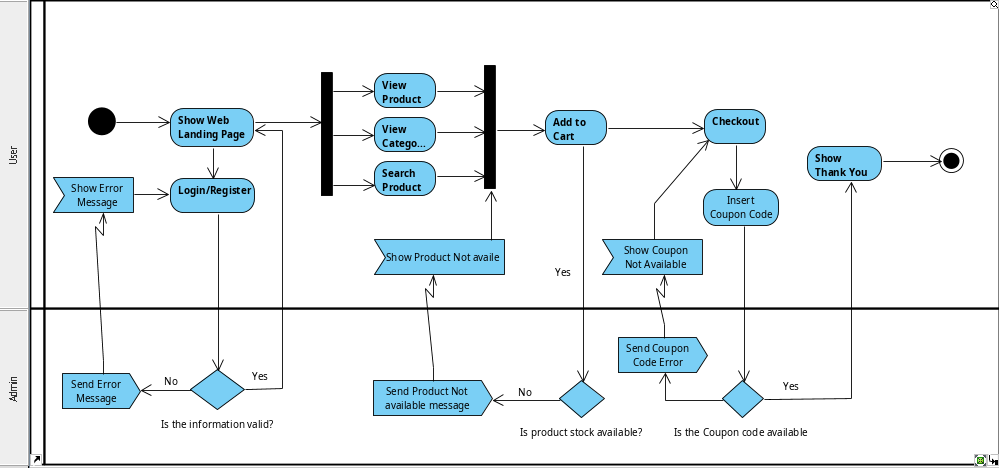


Figure : User Activity Diagram

The above activity diagram represents user activity in the system. After landing in the web app, the user can choose login or register. If the information provided while logging in is valid user is directed to the landing page otherwise an error message in shown. After logging in the user can view product, view all categories, search for any product. Then the product can be added to cart if it is in stock. Then the user can check out by providing correct information. While checking out the user can also provide any coupon codes if they have to receive the discount. After successful order of the products, a thank you page is shown and the program ends.

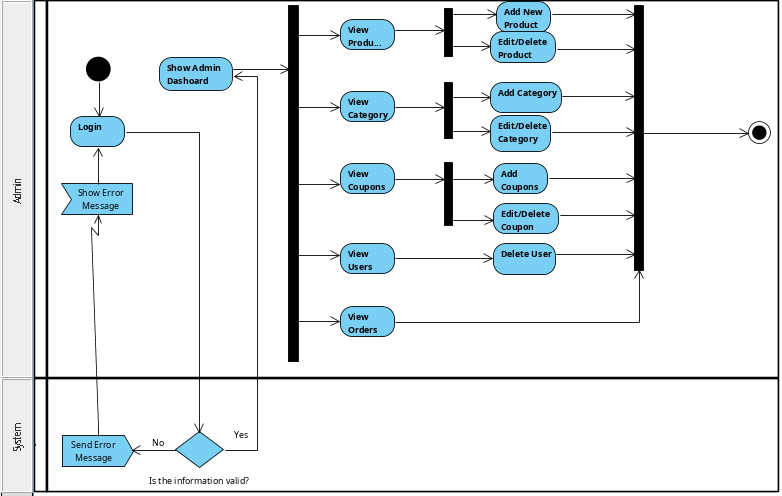


Figure : Admin Activity diagram

The following activity diagram shows the activity flow of the admin. Admin login the system providing the username and password which is validated in the system. If validity passes then admin is sent to dashboard else back to login page. In the dashboard the admin can add, edit and delete categories, product, users. Furthermore, the admin can also view the order details of the user.

### 3.2.2.2: Sequence Diagram

A sequence diagram shows the interaction between objects within the system in a hierarchal order.

#### 3.2.2.2.1: Justification

* Helps in visualizing the interaction between objects
* Helps in finding a logical problem
* Can help in the reverse-engineering of the system

#### 3.2.2.2.2: Notation

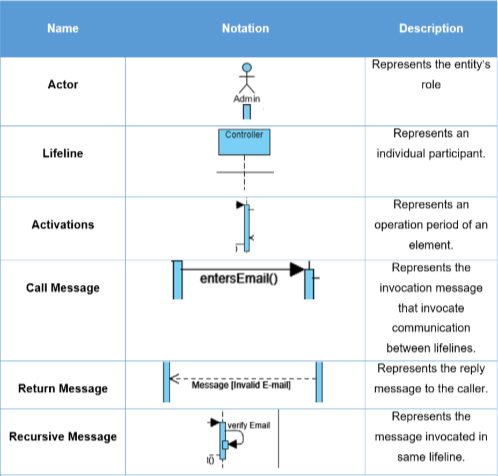


Figure : Sequence diagram notations

#### 3.2.2.2.3: Diagrams

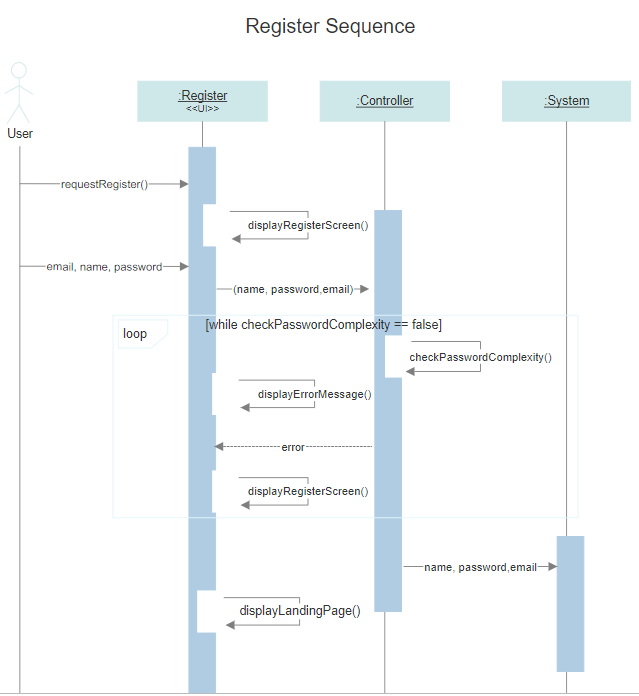


Figure : Register sequence

The following sequence show how the user can register themselves in the application. After the user provides email name and password in the registration form, it is first validated for any validation used. Here, the password should be at least 8 characters long. So, if the validation passes the landing page is shown by logging in the registered user. Otherwise, the registration page is reloaded showing the errors that occurred in validation.

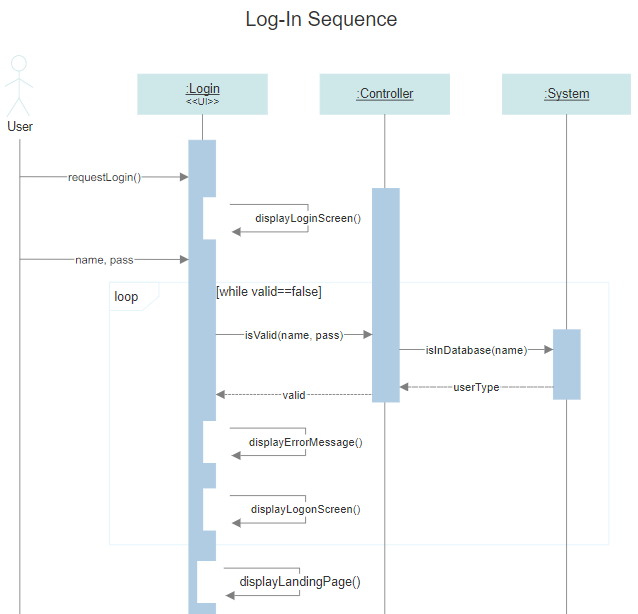


Figure : Login Sequence

The following sequence diagram show how the user logs in. The user provides their log in information in the login page. If the detail is not valid, then an error message is shown. But if the validation and verification passes, the user is redirected to the landing page from where they can buy any products.

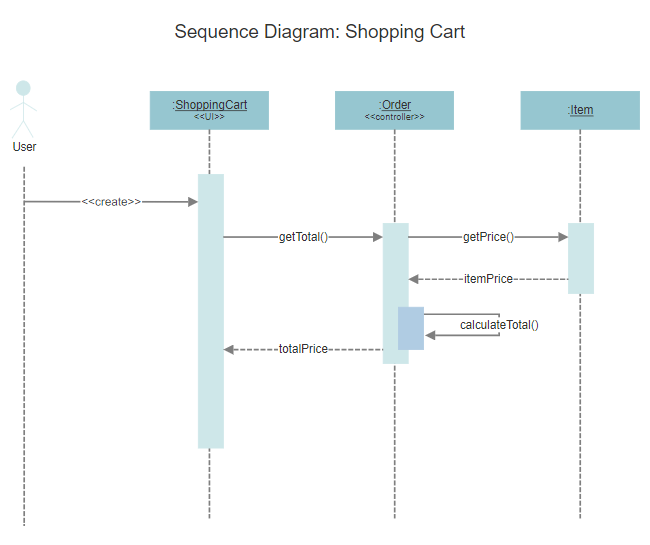


Figure : Shopping cart sequence

The following sequence show how the user can create a cart to buy any products they want. This cart is maintained in the session. So, if the session gets destroyed then the products of cart is also lost. Here, the cart controller requests for price of product from the database. Then the controller calculates the subtotal, taxes, etc. Then total price is shown in the cart form.

The below sequence diagram shows the main interaction of the user, system and the database after user logs in the system. After user gets the view of landing page, they can get all product available in the database. They can add the product to the cart. The system gets the product price from the database and calculates the total price, taxes and discounts.

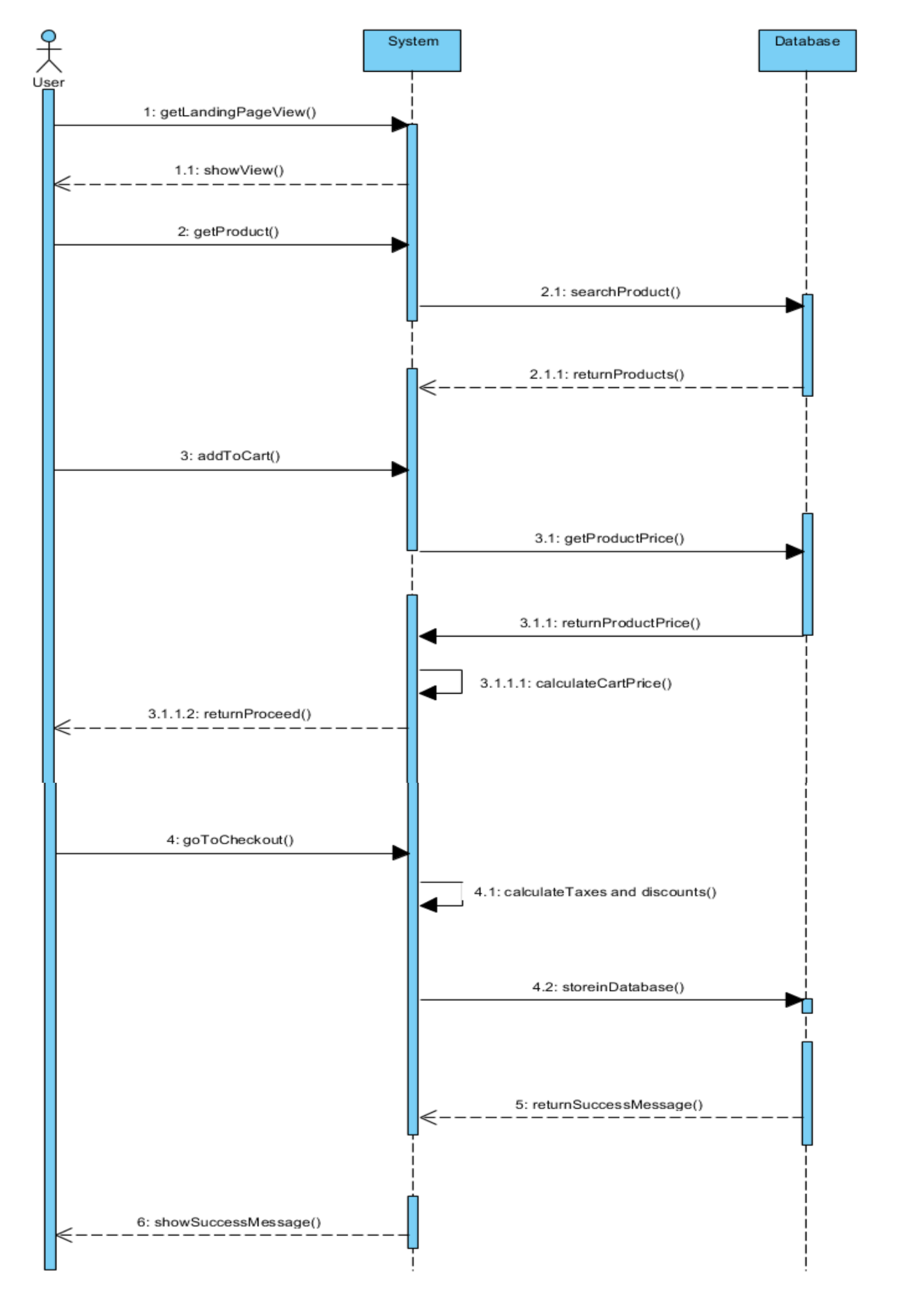


Figure : User Sequence diagram

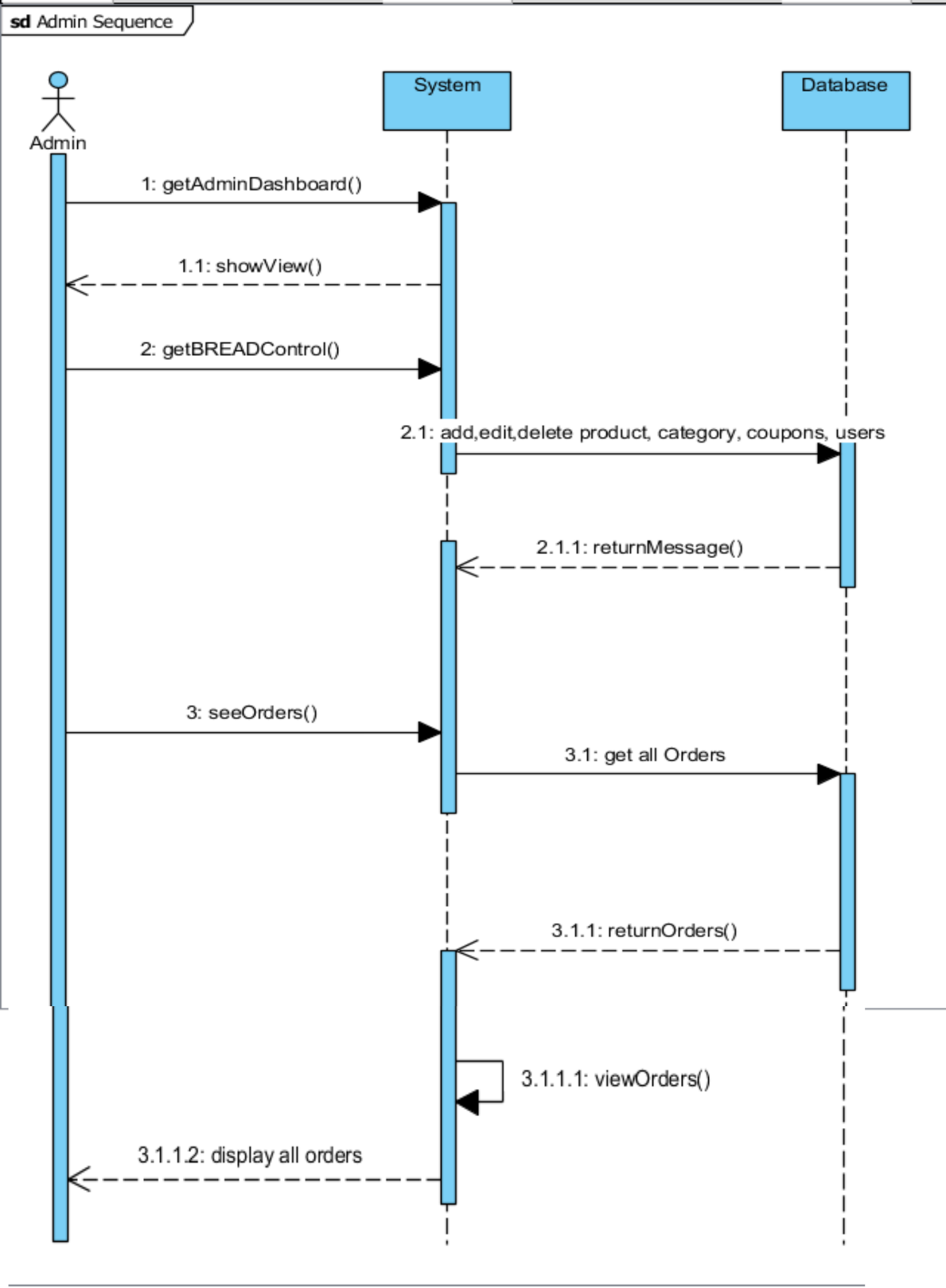


Figure : Admin Sequence diagram

Above sequence diagram shows the sequence performed by the admin. Here, in this system admin can only add, edit, delete, read, browse all categories, products, coupons. Admin can also view all orders ordered by the users.

## 3.2.3: Database Modelling

A database model is the logical design that represents the layout of the database and how the data will be stored. An ER diagram along with a data dictionary is drawn in this modelling.

### 3.2.3.1: Entity Relationship Diagram

ER diagram represents the relationship between all the entities in the database. The ER Diagram of the project is shown below:

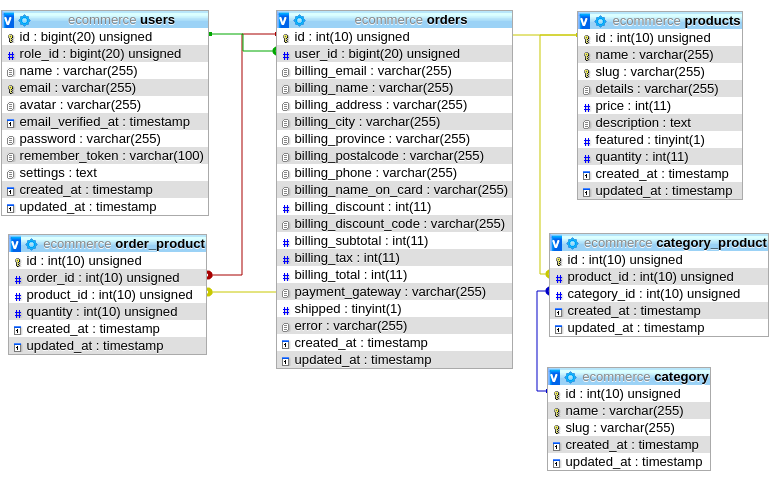


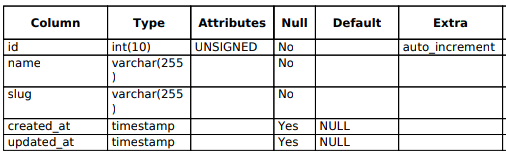
Figure : ER Diagram

### 3.2.3.2: Data Dictionary

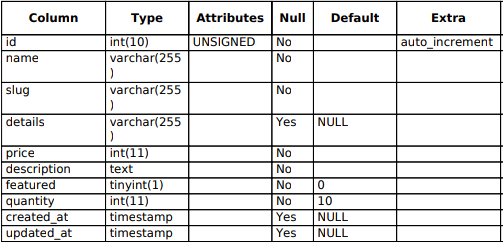
Data dictionary is a set of files consisting the metadata of all entities of the database.

The data dictionary of the project is represented below:

Category Table:



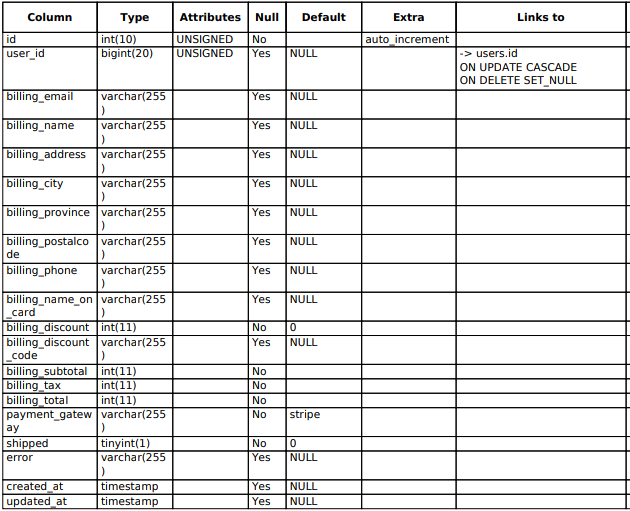
Product Table:



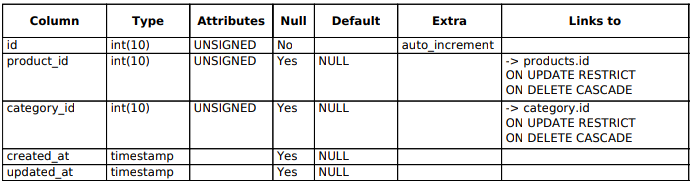
User Table:



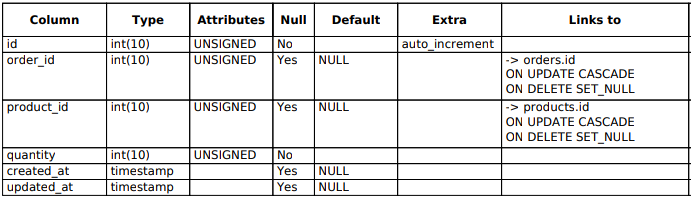
Order Table:



Category\_product Table:



Order\_product Table:



## 3.2.4: Architectural Modelling

Architectural modelling is a model that is designed to study different architectural aspects of the system. This project follows 3-tier architecture which is divided into following parts:

* Client Layer: contains the UI of the application, also known as presentation layer.
* Business Layer: Here all the business logics like calculations, validation, etc. occurs
* Data Layer: In this layer all the data are stored in or are received from the database

## 3.2.5: UI Modelling

### 3.2.5.1: Prototypes

Prototyping is an early sample design of the system UI. Prototype is designed to have a simple sketch of how the application might look.

The prototype of this project is shown below:

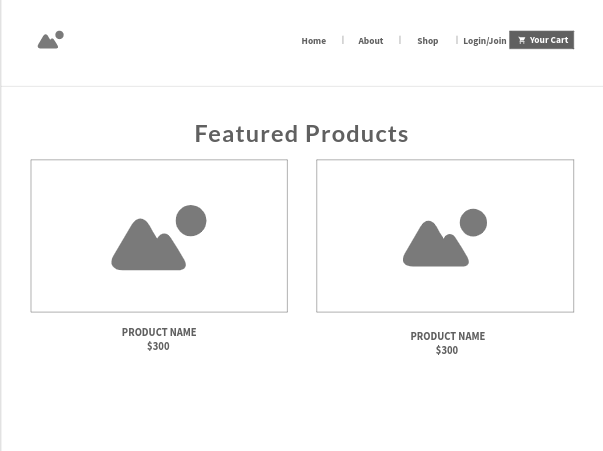


Figure : Landing page

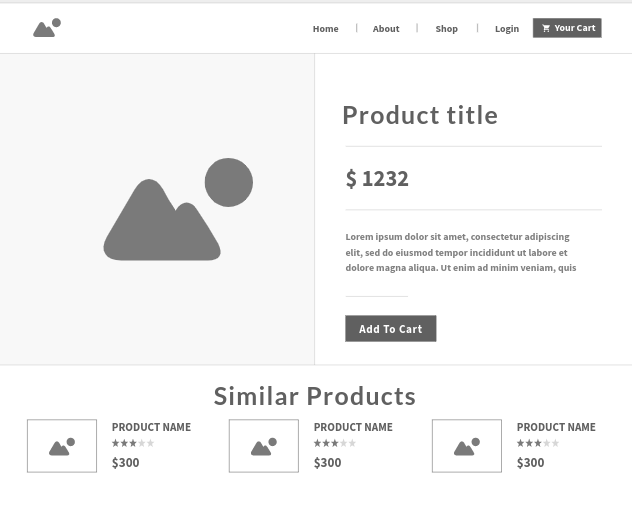


Figure : Product Page

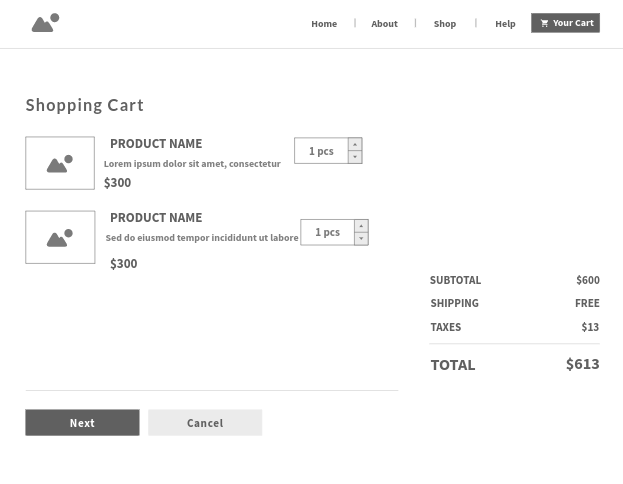


Figure : Shopping Cart

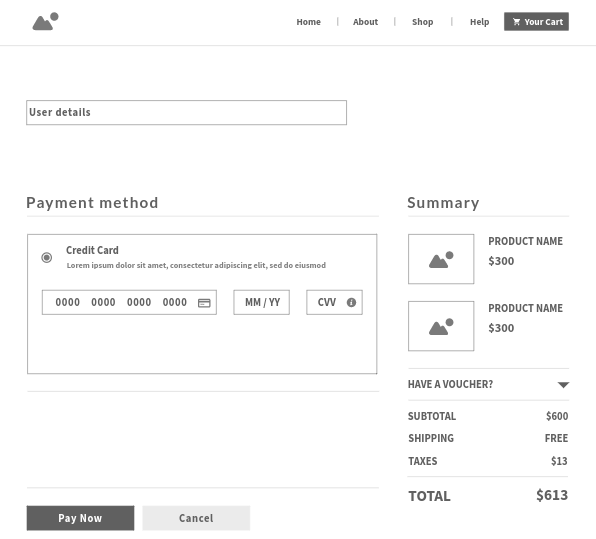


Figure : Checkout form

# 4: Implementation

In the implementation phase, the above system design is transferred into code. For coding I have used HTML, CSS, Bootstrap and JavaScript, jQuery for front end of the system. And for backend a popular PHP framework Laravel is used. Database is maintained in MySQL. Whole system is developed using object-oriented style and MVC pattern. For server Laravel’s artisan command is used, and VSCode was used as the editor to write the code.

All the coding screenshot is available in the Appendix section.

# 5: Testing

In this phase, the code and the function of the system is tested. It helps to assure that the application works as experiment. For testing this application, I have performed Black box testing and Unit testing.

Black box testing is the type of testing that tests the usability of the application by going through the application in the browser. It checks all components in the user interface like buttons, links, redirect links, etc.

|  |  |
| --- | --- |
| **Test ID** | T01 |
| **Test Purpose** | User Registration |
| **Expected Result** | User successfully registered |
| **Actual Result** | User successfully registered |
| **Result** | Test passed |

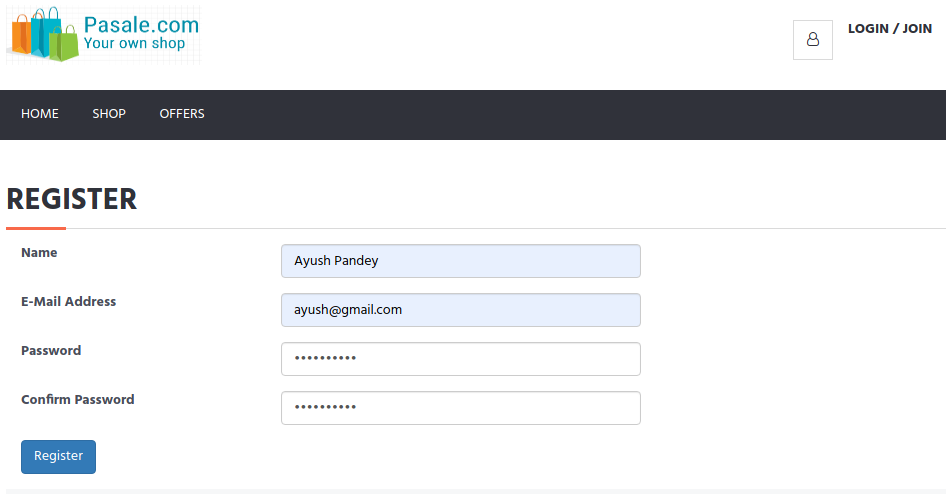


Figure : Test T01

|  |  |
| --- | --- |
| **Test ID** | T02 |
| **Test Purpose** | User Login |
| **Expected Result** | User successfully logged in |
| **Actual Result** | Password incorrect message shown |
| **Result** | Test failed (insert correct password) |

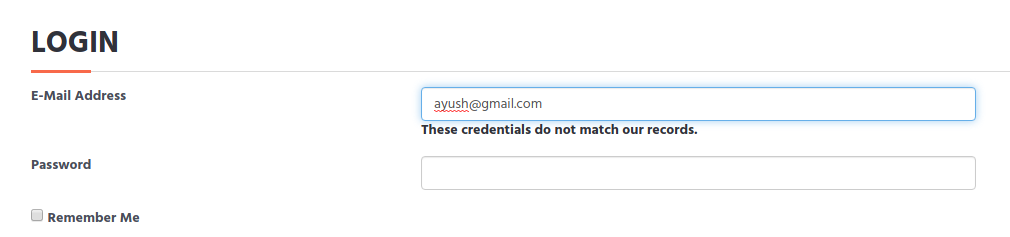


Figure : Test T02



Figure : Test T02 fixed

|  |  |
| --- | --- |
| **Test ID** | T03 |
| **Test Purpose** | Add product to cart |
| **Expected Result** | Product added to cart |
| **Actual Result** | Product added to cart |
| **Result** | Test passed |

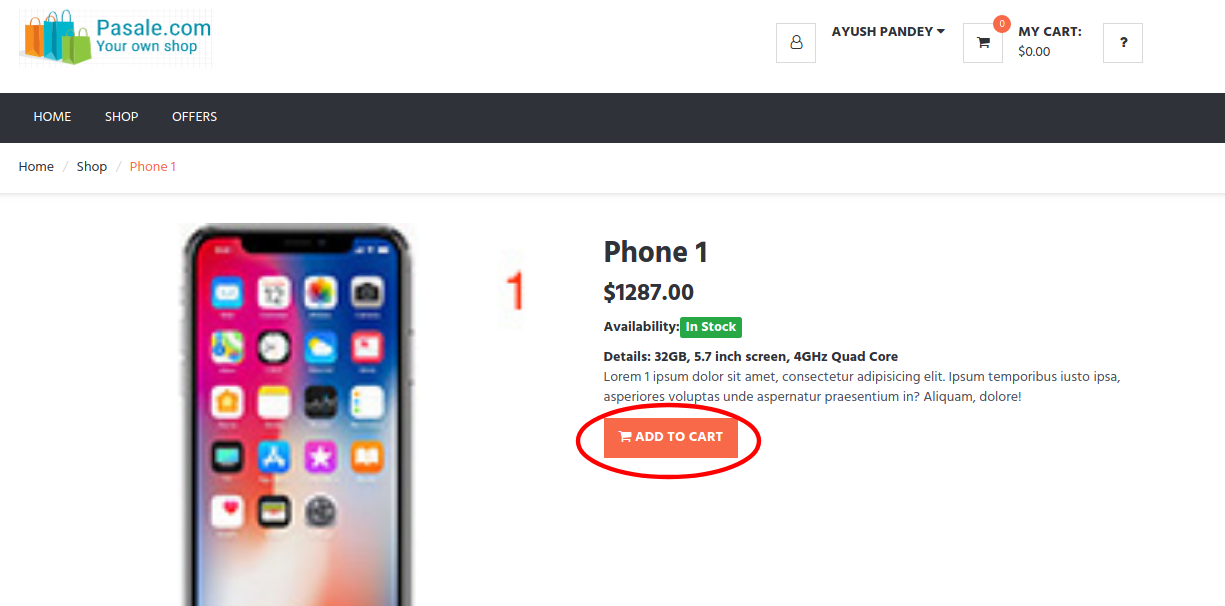


Figure : Test T03

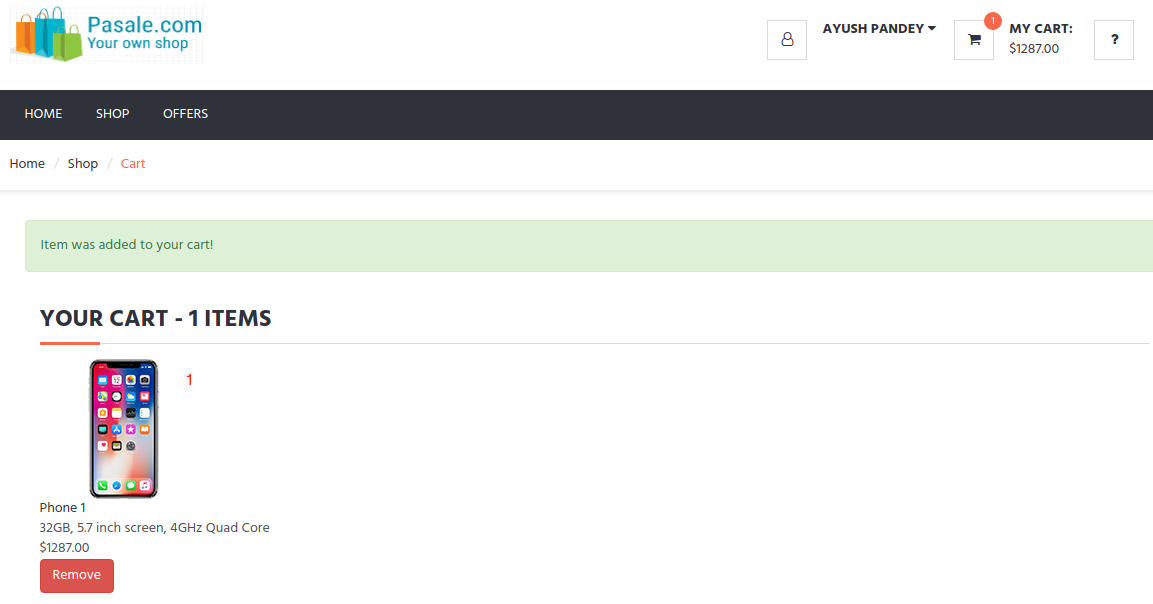
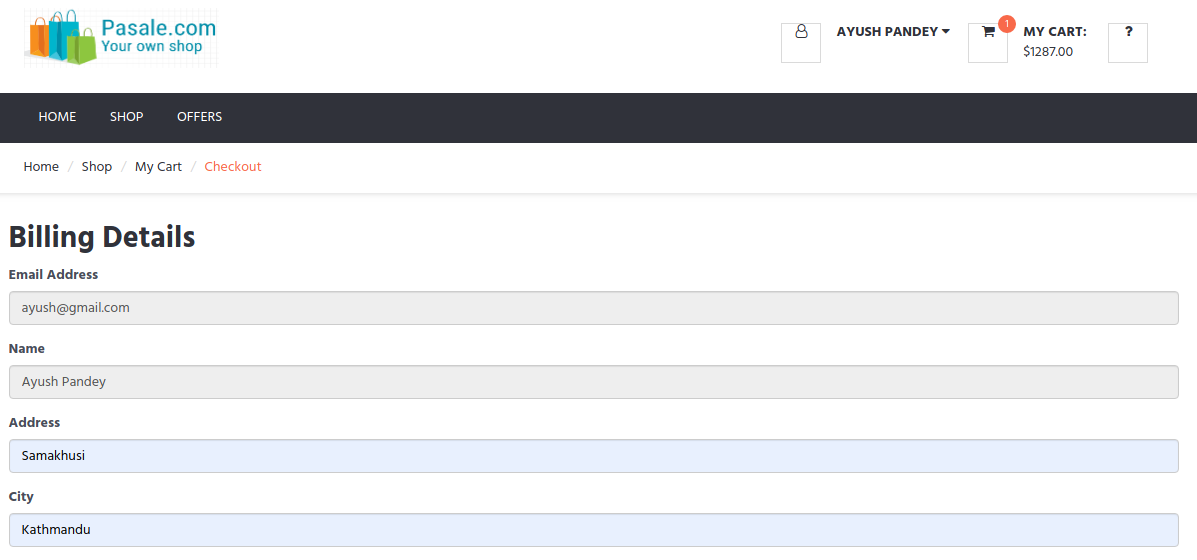


Figure : Test T03 passed

|  |  |
| --- | --- |
| **Test ID** | T04 |
| **Test Purpose** | Product ordered |
| **Expected Result** | Successfully order product |
| **Actual Result** | Products ordered |
| **Result** | Test passed |



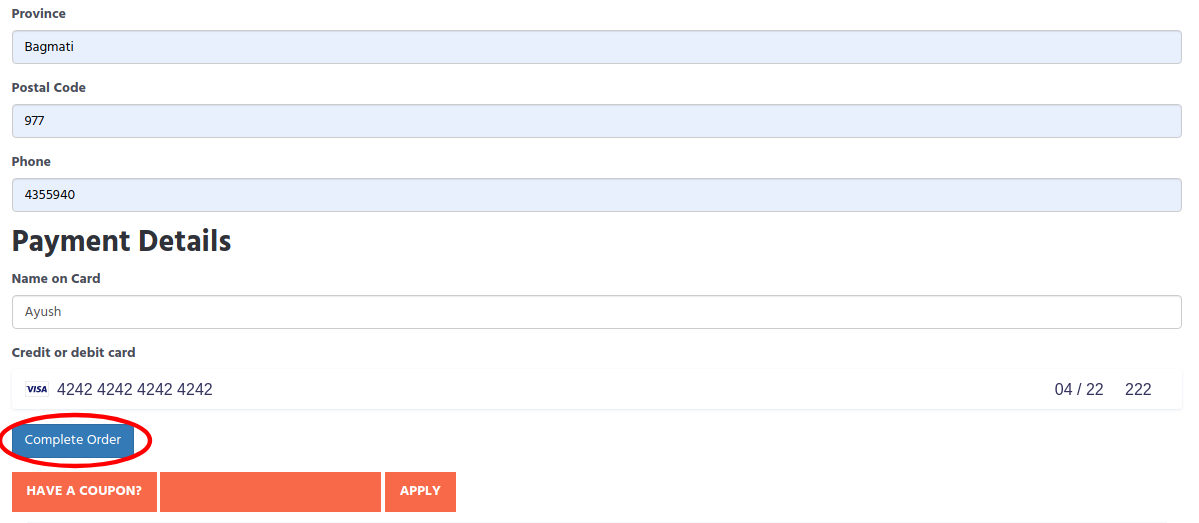


Figure : Test T04



Figure : Test T04 passed

|  |  |
| --- | --- |
| **Test ID** | T05 |
| **Test Purpose** | Delete product from cart |
| **Expected Result** | product successfully deleted |
| **Actual Result** | product successfully deleted |
| **Result** | Test passed |

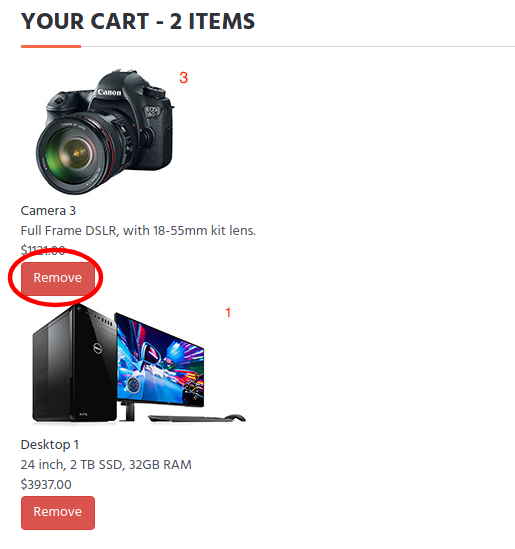


Figure : Test T05

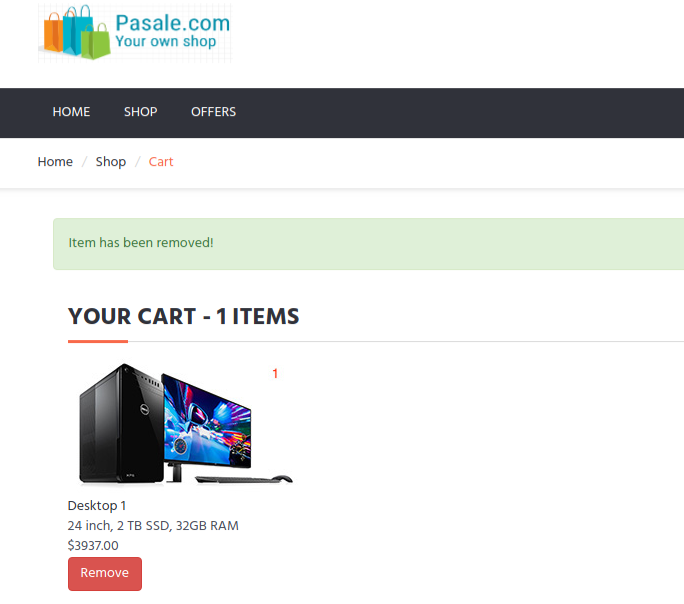


Figure : Test T05 passed

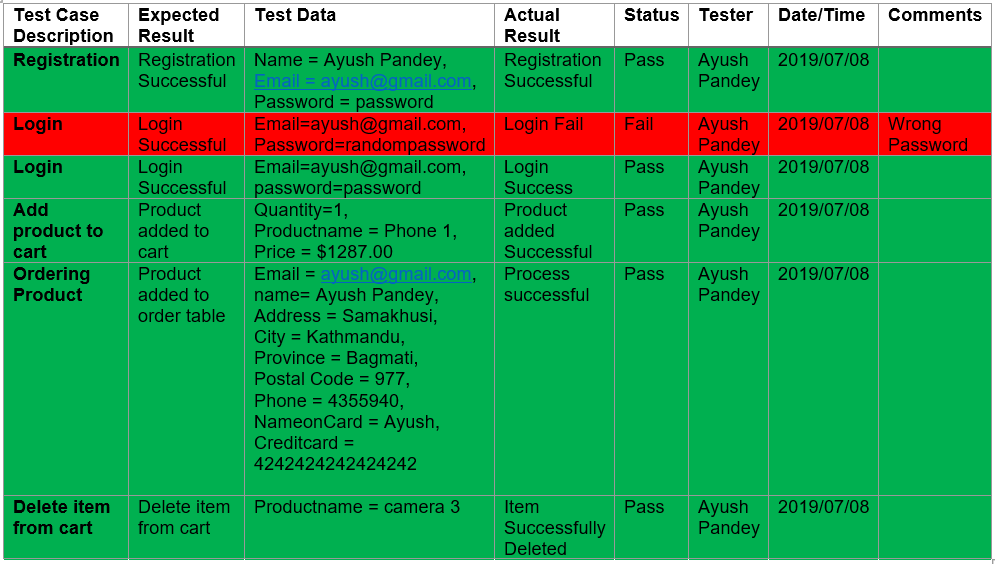


Figure : Testing

Unit Testing is also done to check if the functions of the application works or not. It makes sure the code does what it is supposed to do. For unit testing PHPUnit is used. It tests all unit test functions at once and shows passing of test or errors while testing. In PHPUnit, a green colored notation is used to show passed test and red notation is used to show failed test.

|  |  |
| --- | --- |
| **Test ID** | U01 |
| **Test Class** | LandingPageController |
| **Test Purpose** | Landing page loads |
| **Expected Result** | Landing page successfully loads |
| **Actual Result** | Landing page successfully loaded |
| **Result** | Passed |

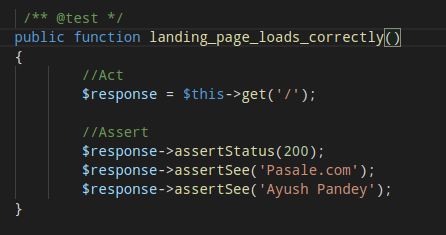


Figure : Test U01

|  |  |
| --- | --- |
| **Test ID** | U02 |
| **Test Class** | ProductController |
| **Test Purpose** | Add a product |
| **Expected Result** | products successfully added |
| **Actual Result** | Product not added; name is not unique |
| **Result** | Failed |

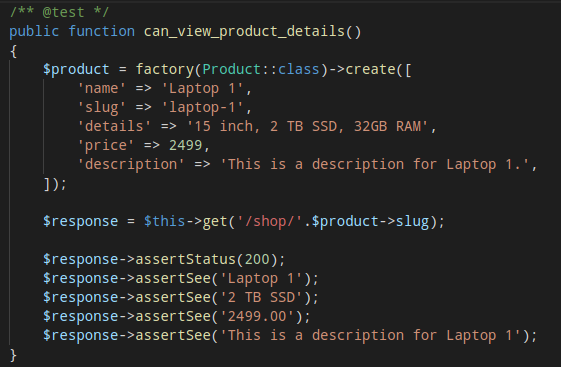


Figure : Test U02

|  |  |
| --- | --- |
| **Test ID** | U03 |
| **Test Class** | CategoryController |
| **Test Purpose** | Category show on the page |
| **Expected Result** | Categories gets loaded |
| **Actual Result** | Categories loaded |
| **Result** | Passes |



Figure : Test U03

|  |  |
| --- | --- |
| **Test ID** | U04 |
| **Test Class** | Helpers |
| **Test Purpose** | Add ‘$’ in price |
| **Expected Result** | Show $100 instead on 100 in price of product |
| **Actual Result** | $ sign shown |
| **Result** | Passed |

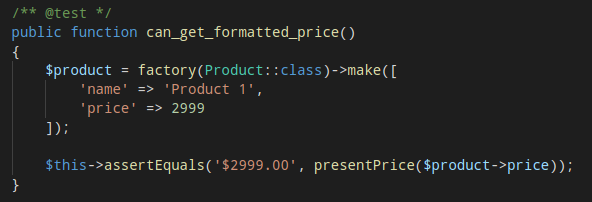


Figure : Test U04

# 6: Other Project Issues

## 6.1: Limitation of the Project

Limitations are the activities that cannot be performed currently, but might be added for the future iteration of development. Some limitations of my project are listed below:

* User can only buy one quantity of product in any particular instance.
* Only one payment system is integrated.
* Product searching is not robust.
* Product sorting is not added.

## 6.2: Future Work

As it is an agile methodology, only the most important feature of the application is added. More features and models can be added in the future to make the system more accessible, usable, etc.

The features that can be added to the application in future is listed below:

* Implement an auction system
* Compare products
* Make the search more robust and add sorting of products on the different basis
* Enhanced security features

## 6.3: Risk Management

Risk is any unwanted event that interrupts the flow of the system. In this project, many risks might occur. So, to make the development process as smooth as possible, potential risks are first identified and then with the help of risk management they are reduced or mitigated.

Risk Management includes four steps that are to be performed. They are:

* Identify Risk
* Access the impact of risk
* Mitigate critical risks
* Control risk

|  |  |
| --- | --- |
| Likelihood | Value |
| Low | 1 |
| Medium | 2 |
| High | 3 |

|  |  |
| --- | --- |
| Consequences | Value |
| Very Low | 1 |
| Low | 2 |
| Medium | 3 |
| High | 4 |
| Very High | 5 |

By taking into account above two tables, the impact of any risk can be derived as Likelihood \* Consequences.

So, based on quantitative risk measurement and guidelines of conducting such assessment following a risk management matrix for this project is presented.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Risk Type | Risk | Likelihood | Consequence | Impact | Action Type | Action |
| Non-Technical | Scope creep – uncontrolled growth of project scope after project begins | 1 | 5 | 5 | Avoidance | Strictly follow the design models developing the system |
| Natural Disaster | 1 | 4 | 4 | Contingency | Regularly backup project files |
| Time Management | 2 | 3 | 6 | Contingency & Avoidance | Since the project is in Agile method, feature that are not complete can be added for next iteration |
| Technical | Hardware Failure | 2 | 4 | 8 | Contingency & Avoidance | Keep backup, regularly maintain the system |
| Performance Degradation | 3 | 5 | 15 | Contingency | Upgrading the hardware |
| algorithmic complexity | 3 | 4 | 12 | Deflection | Get advice from faculty members, research on we for packages to do the complex work |

## 6.4: Configuration Management

All project files are in 3 places, one in local machine, one in GitHub and one in OneDrive as a backup. The project files from local machine and GitHub is synced. The GitHub repository where the project files resides is: <https://github.com/reaperayush7/L5DC-CP-Project-Pasale.com->

The folder structure is based on major SDLC phases. There is a backup folder for local machine backup. All analysis and design files are stored in their respective folder. The main project code is kept in the Implementation folder. Finally, whole project is backed up in OneDrive.

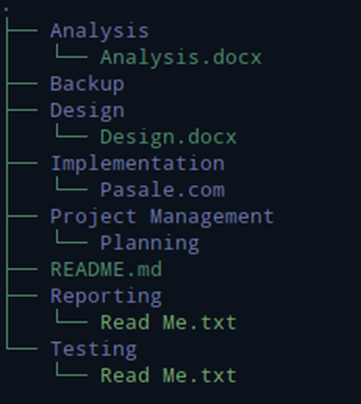


Figure : Project folder tree

## 6.5: User Manual

A User manual is also created in order to help the less technically skilled people to access and use the application properly. The user manual of the project is:

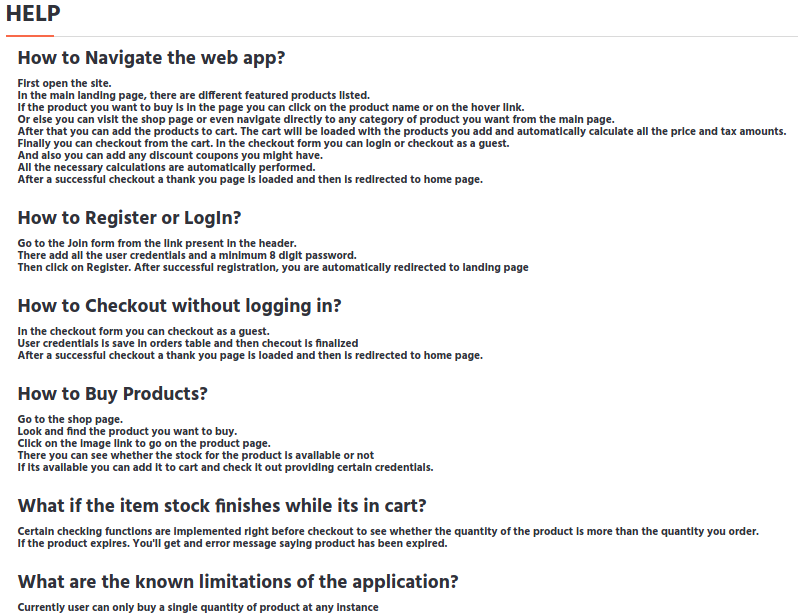


Figure : User Manual/Help

# Conclusion

This documentation has stated all activities that was to be carried out. First the introduction to the project is given. Here, the background of the system is provided along with the justification of the project like where it will be used, how it would benefit the users, etc. The aim and objective of the project was also written.

Further in the next part, the analysis of the project was done. This includes dataflow diagram, functional requirements, nonfunctional requirements, methodology to be used in the development process, feasibility study, and moscow prioritization of the requirements. The use case diagrams of the system were also drawn and explained.

Chapter 3 started with designing and explaining all the static and behavioral models. Then the database was also designed stating all data dictionary. And finally, the applications UI prototype was also created.

In chapter 4 all the models from the previous chapter was implemented into actual code. All code snippets are shown in the appendix section. A help/user manual was also integrated in the application to help users to navigate the web app.

Chapter 5 documented all the test of the project. Black box and white box testing were done extensively to fulfill all the needs.

Finally, the project ended with stating the issues of the project, the limitations, and what features can be added in the future to make the system more useful. Risk management was also done in the project to minimize the risks. In this part, it also explains how the project has been configures proving the GitHub repository URL for easy access to project files.

# References

Buenaflor, L., 2017. *ISO 9126 Software Quality Characteristics.* [Online]   
Available at: https://medium.com/@leanardbuenaflor/iso-9126-software-quality-characteristics-a25a26e7d046  
[Accessed 14 July 2019].

Inflectra, 2018. *What are System Requirements Specifications/Software (SRS)?.* [Online]   
Available at: https://www.inflectra.com/ideas/topic/requirements-definition.aspx  
[Accessed 17 April 2019].

Madsen, S., 2017. *How to Prioritize with the MoSCoW Technique.* [Online]   
Available at: https://www.projectmanager.com/training/prioritize-moscow-technique  
[Accessed 18 April 2019].

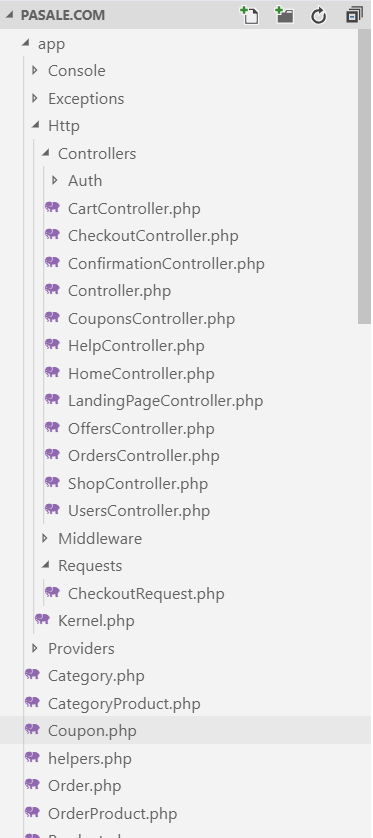
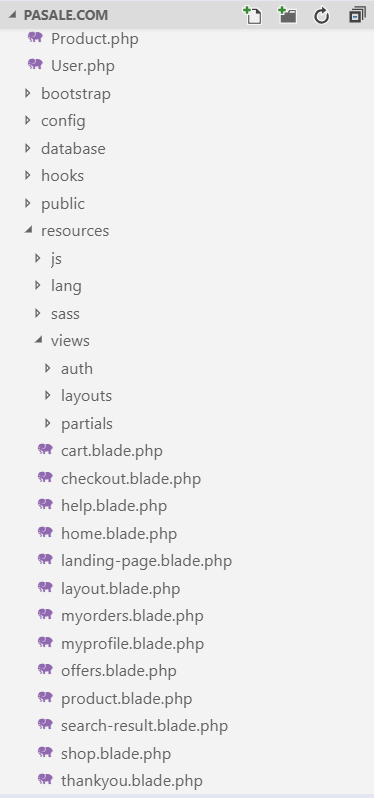
ReQtest, 2018. *Requirements Analysis – Understanding the basics.* [Online]   
Available at: https://reqtest.com/requirements-blog/requirements-analysis/  
[Accessed 14 April 2019].

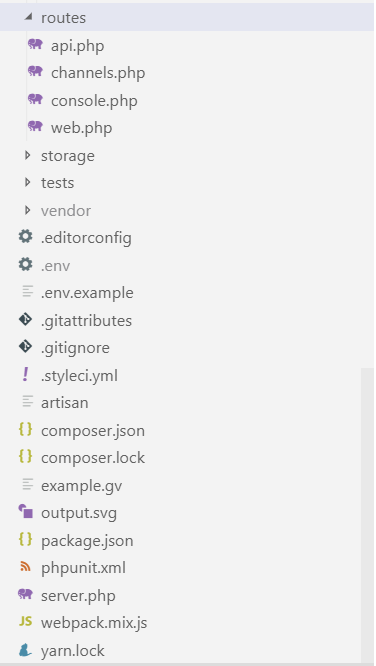
Rouse, M., 2007. *use case.* [Online]   
Available at: https://searchsoftwarequality.techtarget.com/definition/use-case  
[Accessed 17 April 2019].

UKEssays, 2017. *Hard and Soft System Methodology.* [Online]   
Available at: https://www.ukessays.com/essays/management/hard-soft-system-methodology-8121.php  
[Accessed 15 April 2019].

# Appendix

Code directory structure:





Checkout Controller:



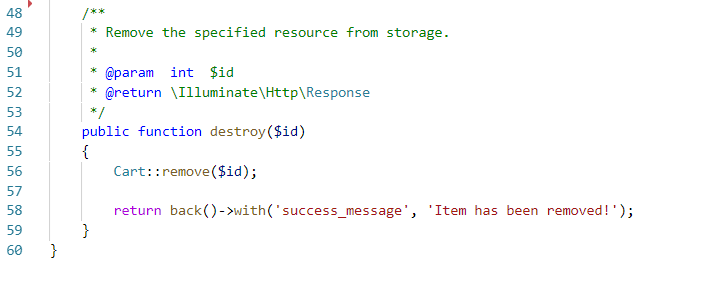






Cart Controller:

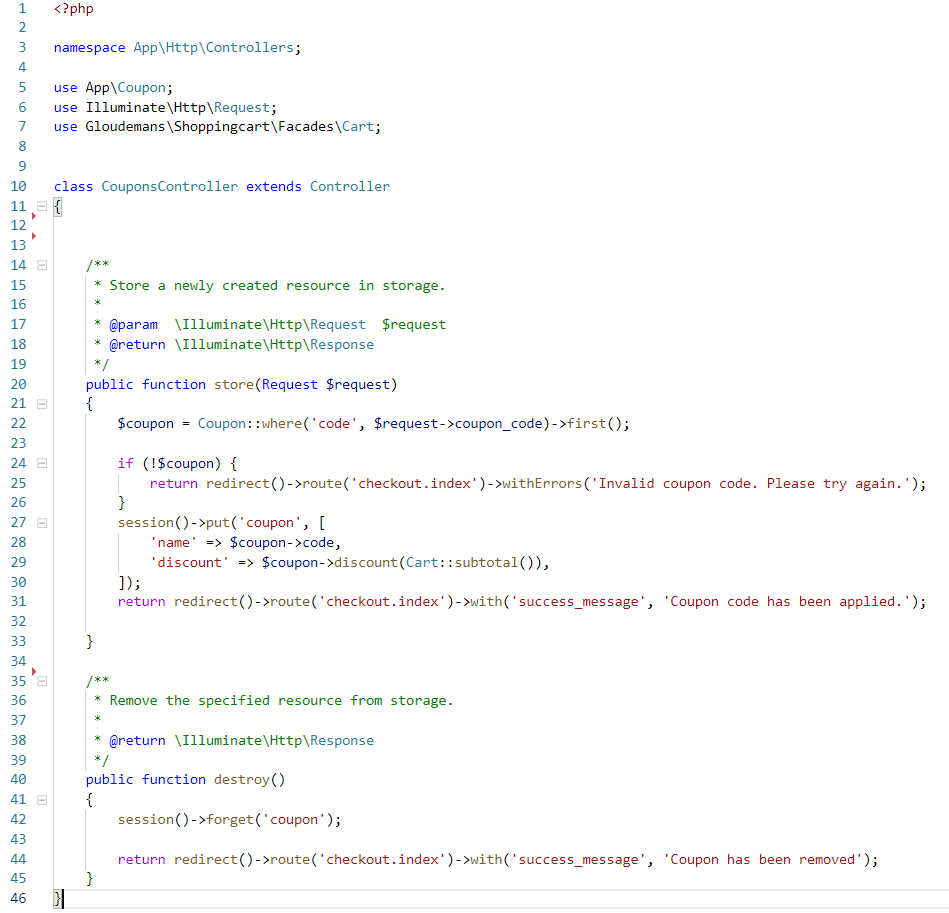




Confirmation Controller:



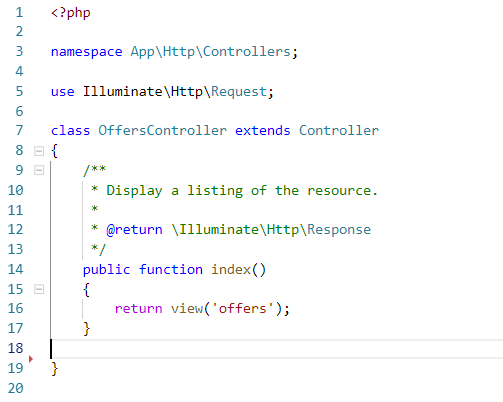
Coupons Controller:



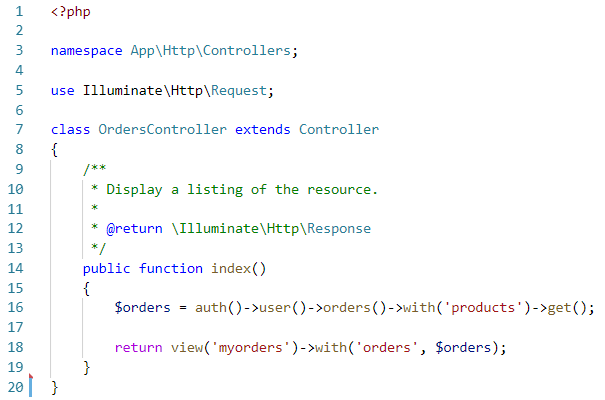
Landing page Controller:



Offers Controller:



Orders Controller:



Shop Controller:



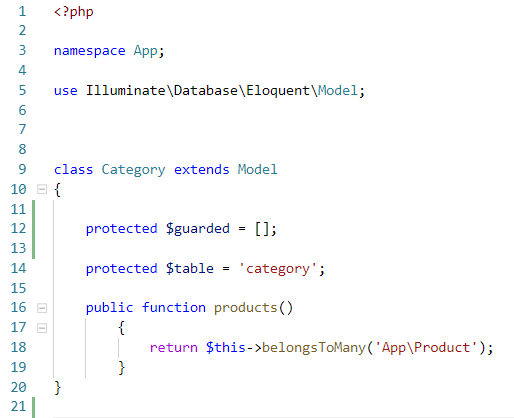


Users Controller:

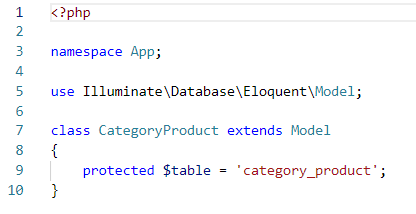


Above diagram shows all the controllers of my project. Now moving onto Models,

Category Model:



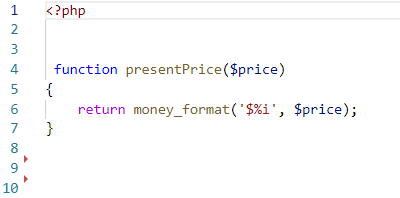
CategoryProduct Model:



Coupon Model:



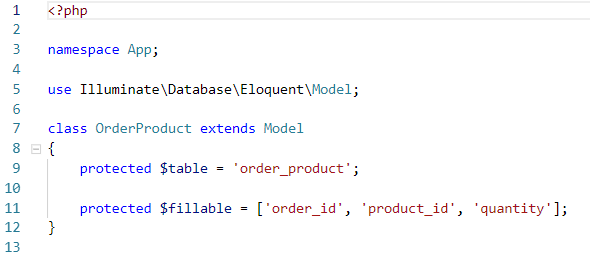
Helpers Model:



Order Model:



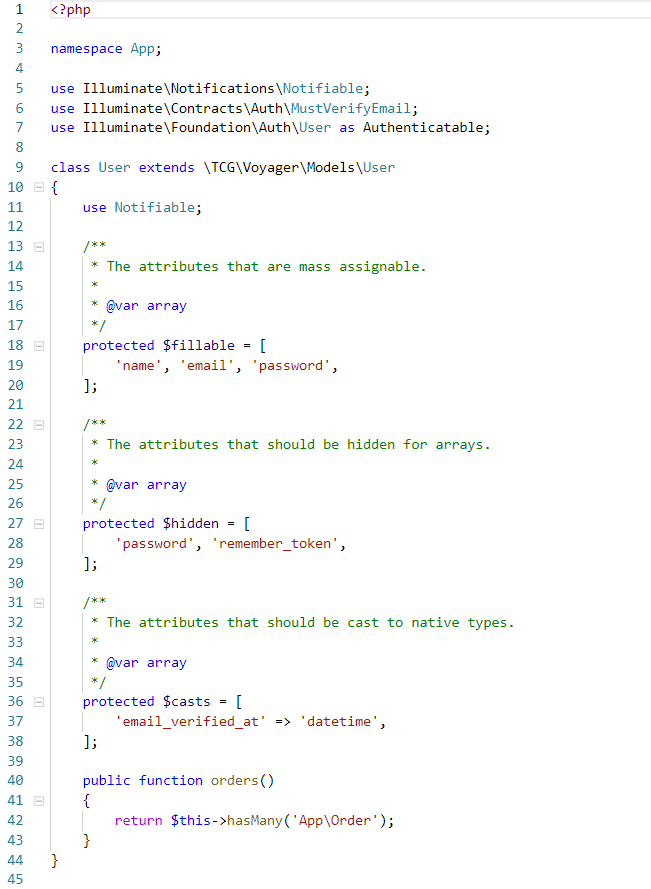
OrderProduct Model



Product Model:



User Model:



Above are all the models that I made. For cart, I used a Laravel package as it reduced the technical burden for me.

The views of the system are provided below:

Layout.blade.php



Cart.blade.php







Checkout.blade.php

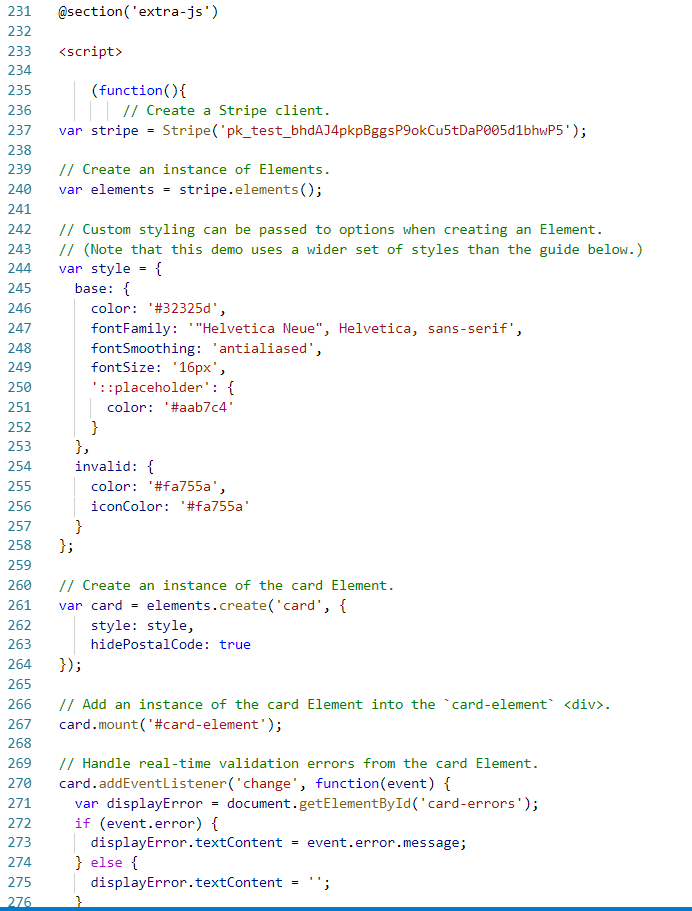


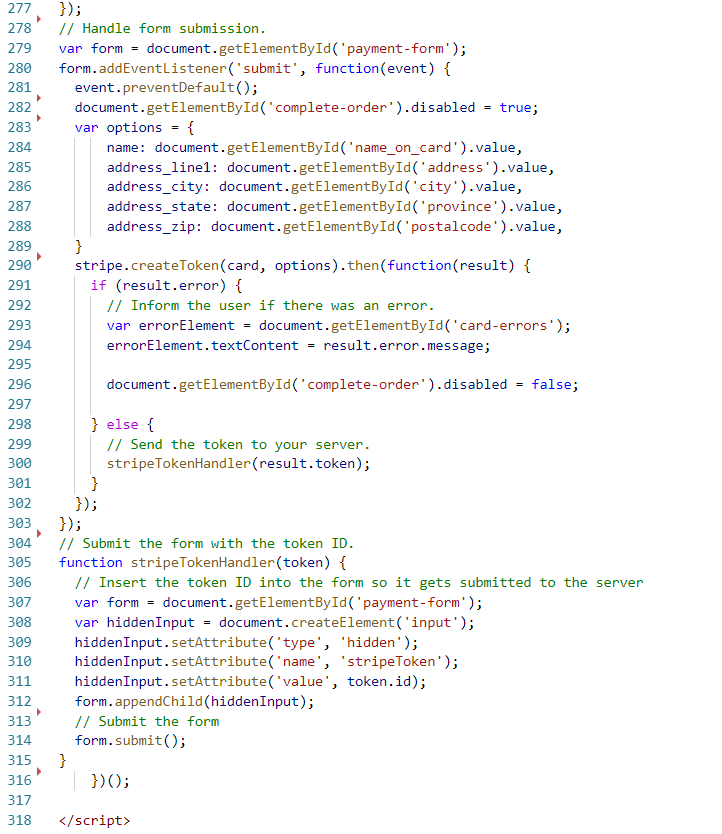






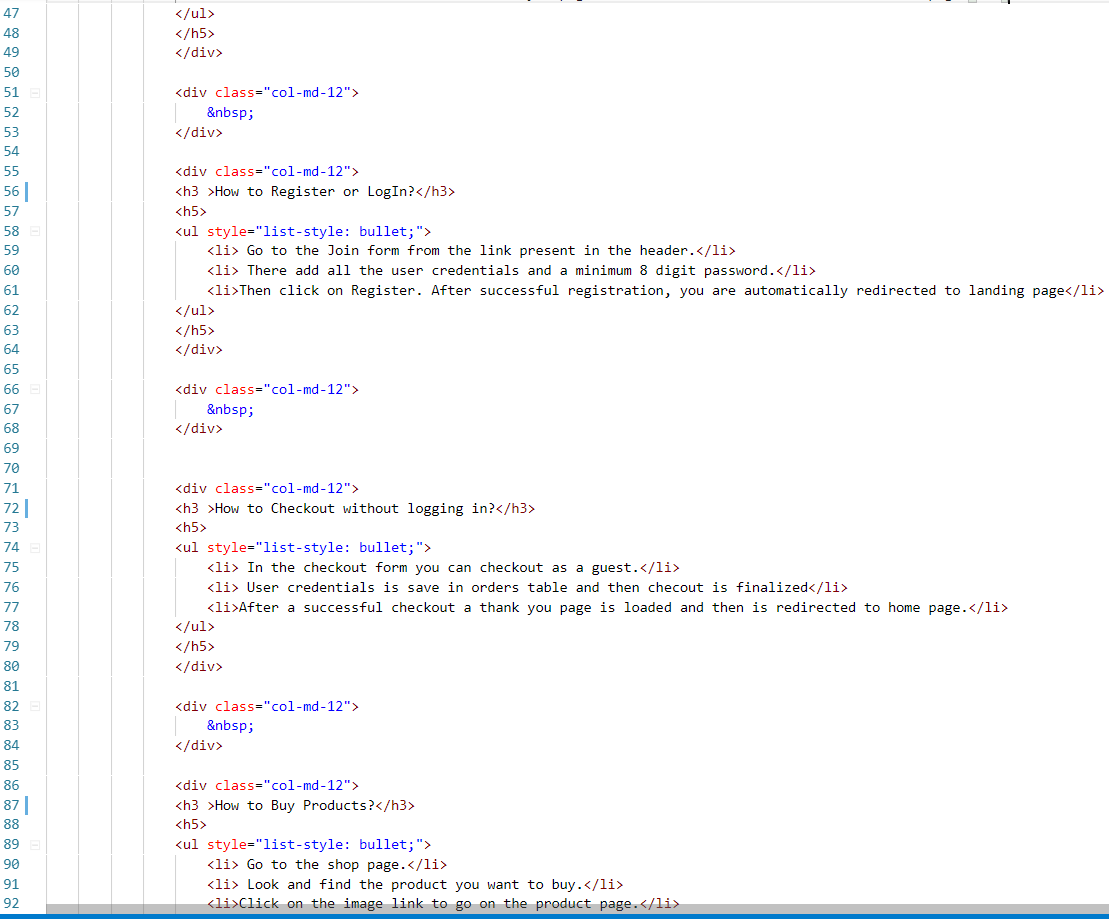


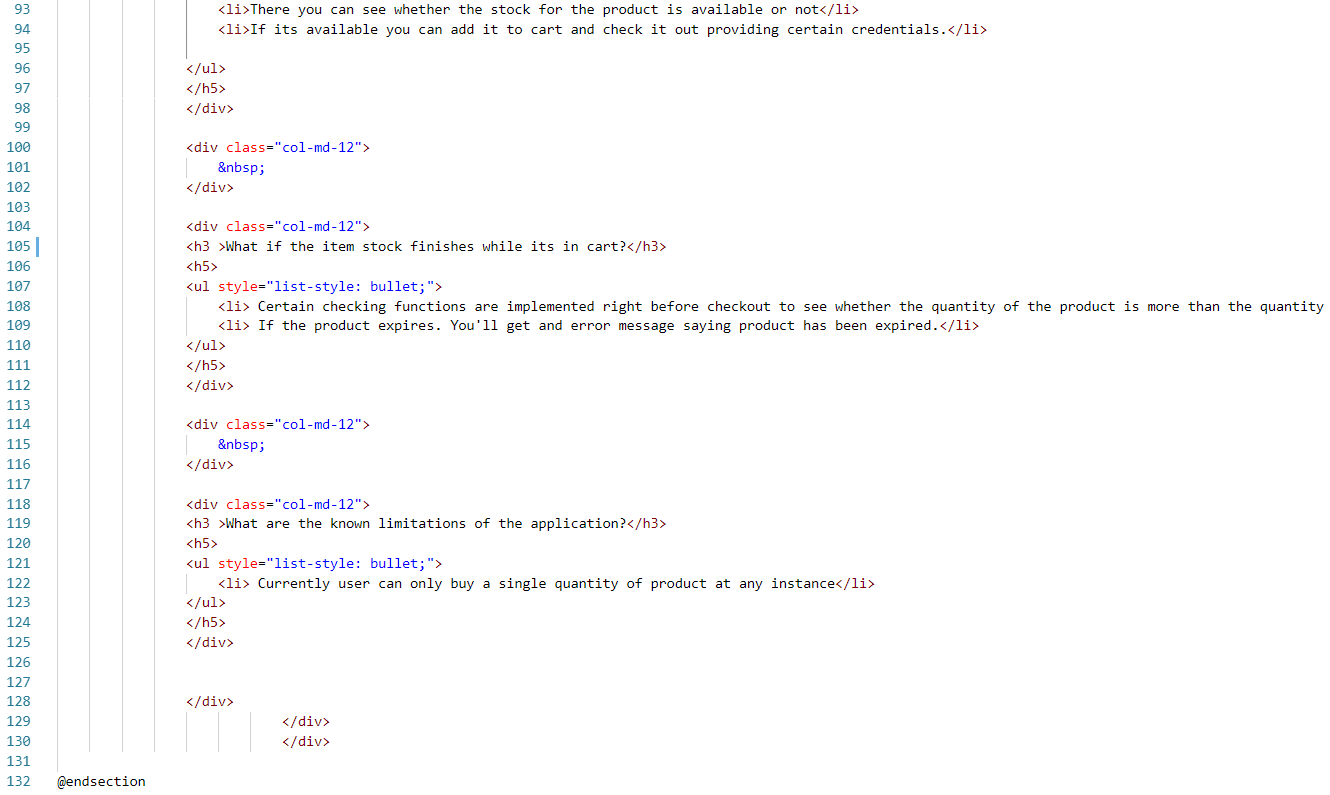




Help.blade.php







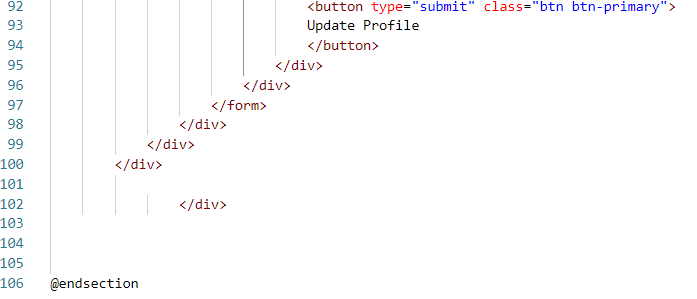
MyOrders.blade.php



MyProfile.blade.php







Offers.blade.php

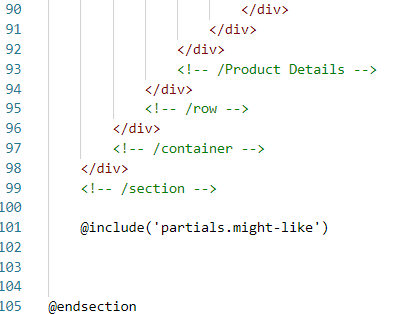




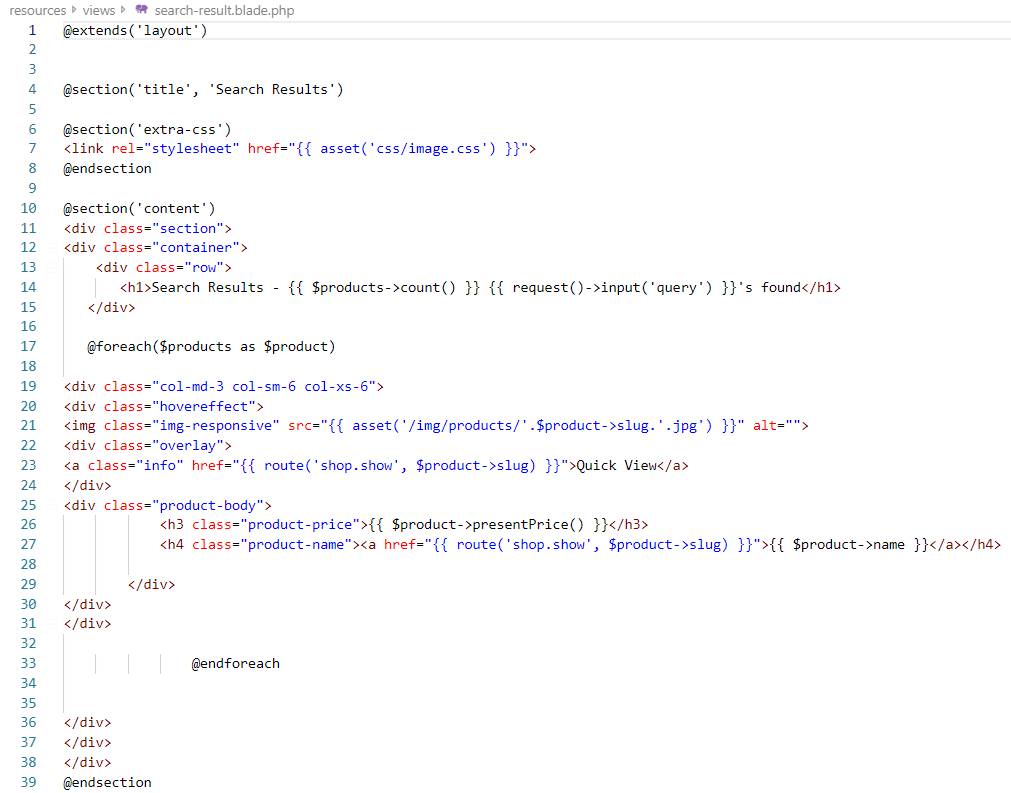
Product.blade.php







Search-Result.blade.php



Thankyou.blade.php



Shop.blade.php

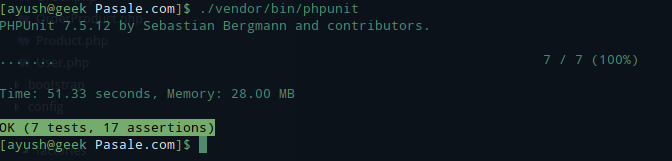




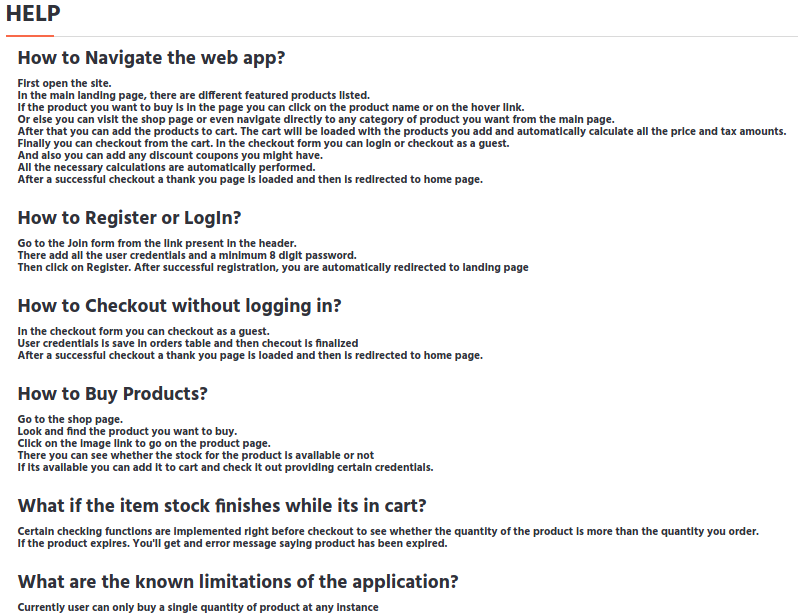
Routes (web.php)



Test scripts is given in the Testing part above in the documentation. The result of the testing is:



User Manual



Further code includes Laravel packages for admin and cart management. This was not written by me. So, the dependency packages are not included here. You can find more in: <https://github.com/reaperayush7/L5DC-CP-Project-Pasale.com->