Statement and Confirmation of Own Work

A signed copy of this form must be submitted with every assignment. If the statement is missing your work may not be marked.

Student Declaration

I confirm the following details:

Candidate Name:	**Gagan Raj Dangol**
Candidate ID Number:	00172900
Qualification:	**L5DC**
Unit:	**Computing Project**
Centre:	**Softwarica college of IT and E-commence**
I have read and understood both NCC Education's *Academic Misconduct Policy* and the *Referencing and Bibliographies* document. To the best of my knowledge my work has been accurately referenced and all sources cited correctly. I confirm that this is my own work and that I have not colluded or plagiarised any part of it.	
Candidate Signature:	
Date:	**2019/04/30**

Woodcraft Store Management System

Gagan Raj Dangol

00172900

Computing Project

Level 5 Diploma in Computing

Email: dangolgrozen123@gmail.com

Softwarica College of IT & E-Commerce

Kathmandu, Nepal

01/27/2019

Submitted to: Kiran Rana

Acknowledgement

I would like to thank Module Leader Kiran Rana Sir for the opportunity to set off this project and for his support and guidance throughout the project.

I would also like to thanks to all my supportive friends and teachers. Their teachings and suggestion were really valuable.

Abstract

Woodcraft Store Management System is a web-based application software that focuses on providing its customers with best web-services. Moreover, e-commerce facilities and also other services that facilitates stock management, information gathering, etc.

Contents

List of Figures

Chapter 1: Introduction

Introduction

Woodcraft Store Management System is a web-based application software that focuses on providing its customers with best web-services. Moreover, e-commerce facilities and also other services that facilitates stock management, information gathering, etc.

Laravel is a PHP framework for developing web-base and desktop applications and is object oriented. For the development of Handicraft Store Management system, I chose Laravel.

1.1 Problem Background to the system

The transactions held on the store were based on bills and receipts. Customers order products either by telephone or going into the store. Recordkeeping is paper-based and recorded into registers and files. This led to tenuous working environment. Growing business and customers made it difficult to manage the records.

The sole purpose of this project is to develop a web-application that computerizes above tasks. The application enables customers/users to engage with the store using their phone and computers and held daily transactions, as similar to an e-commerce site. Use of database tools leads to proper management and safety of records. Thus, the project will help in overcoming these difficulties.

1.2 Aims

The major aims of the project are:

- To automate and simplify daily transaction of the store
- Keeping the business relations intact
- Keep the customers/users engaged with the store

1.3 Objectives

To obtain above goals following objectives are to be considered:

- Perform analysis and gather requirements
- Create user friendly interface
- Design database
- Perform testing
- Perform maintenance
- Provide online shopping services
- Produce a well-documented report
- Design logical and physical diagrams
- Perform proper Testing
- Perform Maintenance

1.4 Overview of the project design

Woodcraft and its collection is a hobby for many people. Handmade crafts are valued even more. People are less aware of the varieties of products that the store possesses and most people are not even aware of the traditional values that woodcarving carries along with it. It is within the aim of the store to publicize and promote their products but primarily maintain and manage their daily transactions held with their customers and wholesalers.

This project is ought to help complete this aim. It leads to a web-based application that lets the customers to place product-order, bill-order and others through internet. It also allows customers to instantly get information and share if needed, which can contribute in publication and promotion of woodcarving. It also informs the users/customers and wholesalers about ongoing activities. The project ensures to develop an application that is convenient to use and in a way that the users are motivated to keep engaging with the store.

Chapter 2: Analysis

Analysis:

2.1 Introduction to Analysis

Analysis is the process of identification and documentation of requirement of the proposed system. In the analysis phrase, first step is to feasibility study after that we model system in use case diagram and class diagram which is also called system modelling.

2.1.1 Needs for Analysis

For analysis following things are to be considered before-hand:

1. Identify customer's needs.
2. Evaluate system for feasibility.
3. Perform economic and technical analysis.
4. Allocate functions to system elements.
5. Establish schedule and constraints.
6. Create system definitions.

2.1.2 Object Oriented Analysis

Object Oriented Analysis (OOA) is an approach to analysis of system by applying object-oriented programming, also using visual modeling throughout the development life cycle. It is used to create a model of the system's functional requirements that is independent of implementation constraints. In OOA, data is modeled by ER diagrams and Data flow diagrams and behaviors are modeled by Activity and Sequence Diagrams.

2.1.3 Merits and Pitfalls of the Project

Merits

- Turn services online i.e. remotely accessible
- Avoids bargains and brawls
- Automate and simplify daily transactions

Pitfalls

- Online payment is not available
- Multiple products cannot be ordered at a time
- Services are limited to national borders

2.2 Requirements

Requirements are the conditions/tasks of a project that must be completed to ensure the success of the project. They represent a fine picture of the work that is to be done. On the basis of functionalities, there are two types of requirements: functional and non-functional requirements.

2.2.1 Functional Requirements

Functional requirements are the features and functions of the product that must be implemented to let users to complete their tasks. It determines **what** a system should do.

Non-Functional Requirements

Non-Functional requirements are the behaviors of the system which establish constraints of the system's functionality. In other words it enhances work flow the functionalities. It determines **how** a system is supposed to be.

2.2.2 MoSCoW Prioritization

MoSCoW prioritization method is a widely used prioritization technique used for management over business analysis, project management and software development fields to make stakeholders understand the importance of project requirements with respect to their priorities. It is also known as MoSCoW analysis

MoSCoW stands and defines for following phrases:

M = Must Have

S = Should Have

C = Could Have

W = Won't Have

Each of which explains the weight of the priority that requirements get.

Functional Requirements and their dependencies with MoSCoW prioritization:

S.N.	Requirements	Prioritization MoSCoW	Description	Dependency
F1.	Login	**M**	Log into the site to continue where left off	F2
F2.	Registration	**M**	Create account to further use the functionalities of the application	-
F3.	Insert Item	**M**	Adding items/products to the site to let users/customer order them	F1
F4.	Update Item	**S**	Edit the details or information of items originally provided while adding	F3
F5.	Delete Item	**S**	Delete items that are no longer required or out of stock	F3
F6.	View Item Information	**M**	Retrieve the data of item for all users and non-users	F3

F7.	Search and select Item	**S**	Retrieve the data of specific item for all users and non-users when searched	F3
F8.	Home Page	**M**	Landing page for all visitors also known as index	-
F9.	Edit Profile	**S**	Edit user information originally provided by them	F1
F10.	Order Items	**M**	Users/customer orders the items/products from the site.	F3
F11.	User Review	**C**	Take review from users and let them rate the products or the site	F1
F12.	Log Out	**M**	Log out of the site in purpose of security or to change account in the same device	F1
F13.	Generate Bill	**M**	Produce a bill of products a user just ordered	F10
F14.	Forgot Password	**C**	Edit password	F1
F15.	Contact Information	**S**	Show contact and all sort of necessary information to users	F8
F16.	Visitor Counter	**C**	Counts the number of visitors that visited the website	F8
F17.	Manage Inventory	**S**	Stock management and sales report	F1

Table 1: Functional Requirements

Non-Functional Requirements with MoSCoW prioritization:

S. No	Requirements	MoSCoW	Description
NF1.	Security	**S**	Protection of system and its data from both external and internal threats
NF2.	Performance	**S**	Resources required, response time, transaction rates, throughput, benchmark specifications
NF3.	Availability	**S**	Is the system available at all time, locations and platforms
NF4.	Reliability	**S**	How often does the system fails or doesn't failed at all. Does it work flawlessly when needed
NF5.	Maintainability	**S**	Is system repair possible, if yes, how difficult is it to do so
NF6.	Usability	**S**	How difficult is it to learn and operate the system also counts time
NF7.	Data Integrity	**S**	Is there any chance of data loss and data counterfeit
NF8.	Flexibility	**S**	Effort required to make changes in the software. How easy or hard it is to make changes

Table 1: Non-Functional Requirements

2.3 Natural Language Analysis (NLA)

NLA (Natural Language Analysis) is a process of selecting out nouns, verbs and adjectives to sort classes, methods and attributes respectively from the given scenario. There are several processes for NLA.

Filtering nouns for classes

Following steps are applied for the noun filtration:

I. Listing out nouns
II. Removing repetitive nouns
III. Removing synonyms
IV. Removing predictive nouns
V. Removing nouns that are out of scope

2.4 Initial Class Diagram

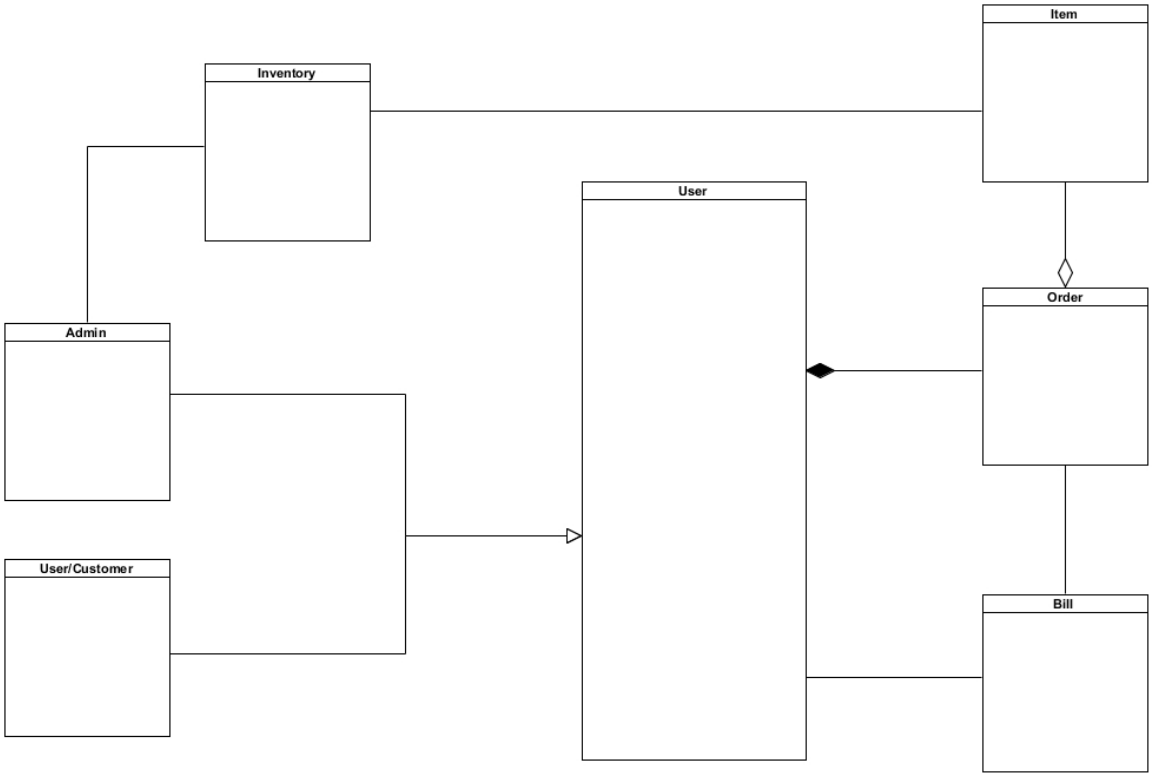


Figure 1: Initial Class Diagram

2.5 Use Case

A use case diagram is the primary form of system requirements for a new software program which is being developed. Simply, it is a graphical representation of the actions that a user and the admin, which are called actors in this context, does with the system. These actions are called use cases.

Justifications:

A use case diagram is usually simple, it does not represent the detail of the use cases. It specifies the expected behavior but not the exact method of making it happen. So, the following use case involves user and admin performing their actions in a rough sketch. The system also functions in few ways. Generating bill in this cse.

Advantages:

- It helps manage complexity and makes it easier to understand the requirements and the features.
- It can help differentiate actual uses of the system to unnecessary functions.
- It encourages designers to envision outcomes before attempting to specify them.

Disadvantages:

- Non-functional requirements are undermined by the use case diagram.
- It is not accurate, it is just a rough sketch of what actually happens.
- There is no interaction between the requirements within it.

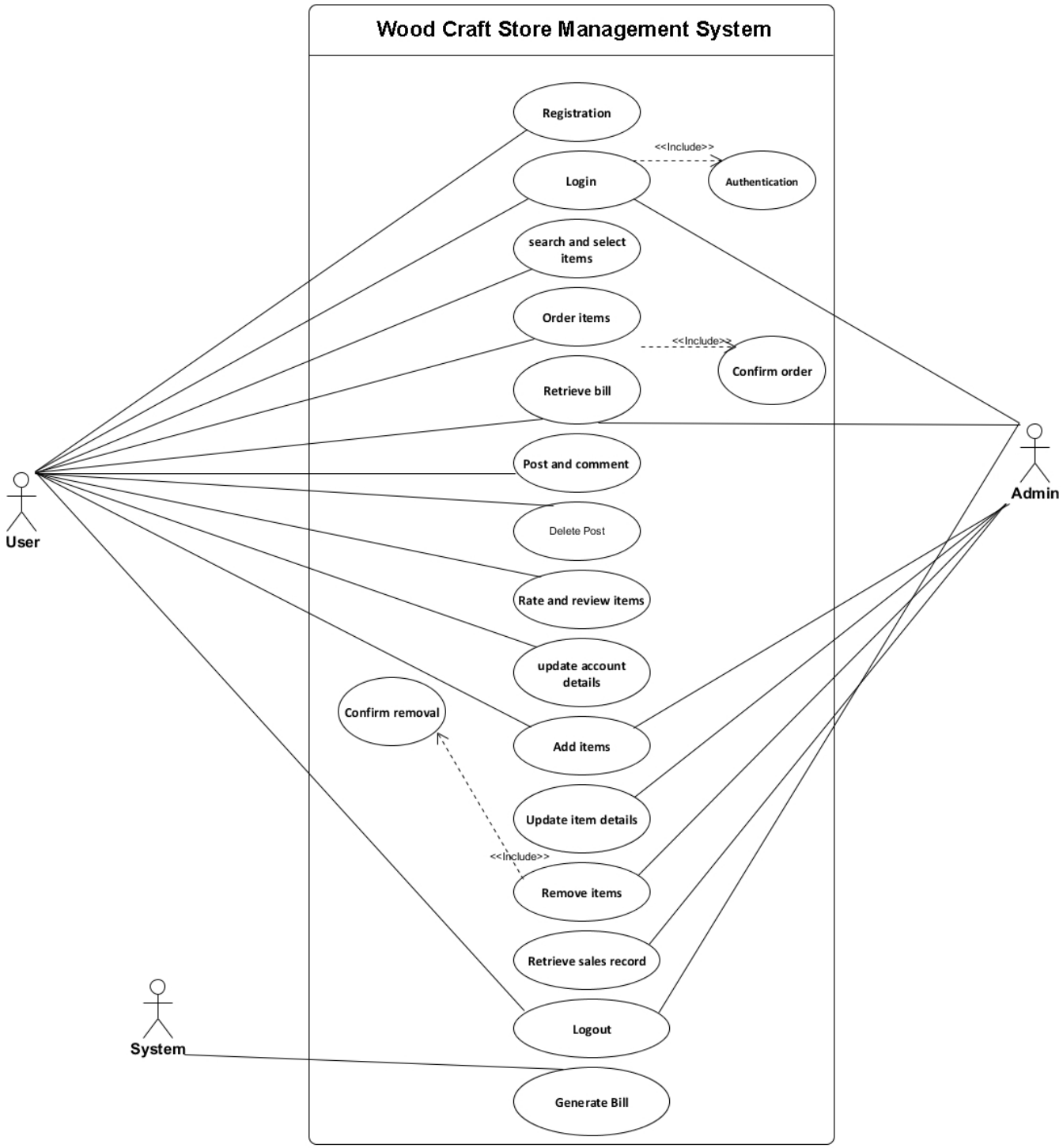


Figure 2: Use Case Diagram

2.6 Development Methodology

2.6.1 Waterfall Model

Waterfall model is a simple means of software development methodology. It is also referred to as a linear-sequential life cycle model (SDLC). In waterfall model, each phase must be completed before the next phase can begin and there is no overlapping in the phases.

This model is simple and easy to understand and use. It is easy to manage due to the rigidity of the model. Each phase has specific deliverables and a review process. This is the reason why I used waterfall model.

Following are the steps involved in the waterfall model:

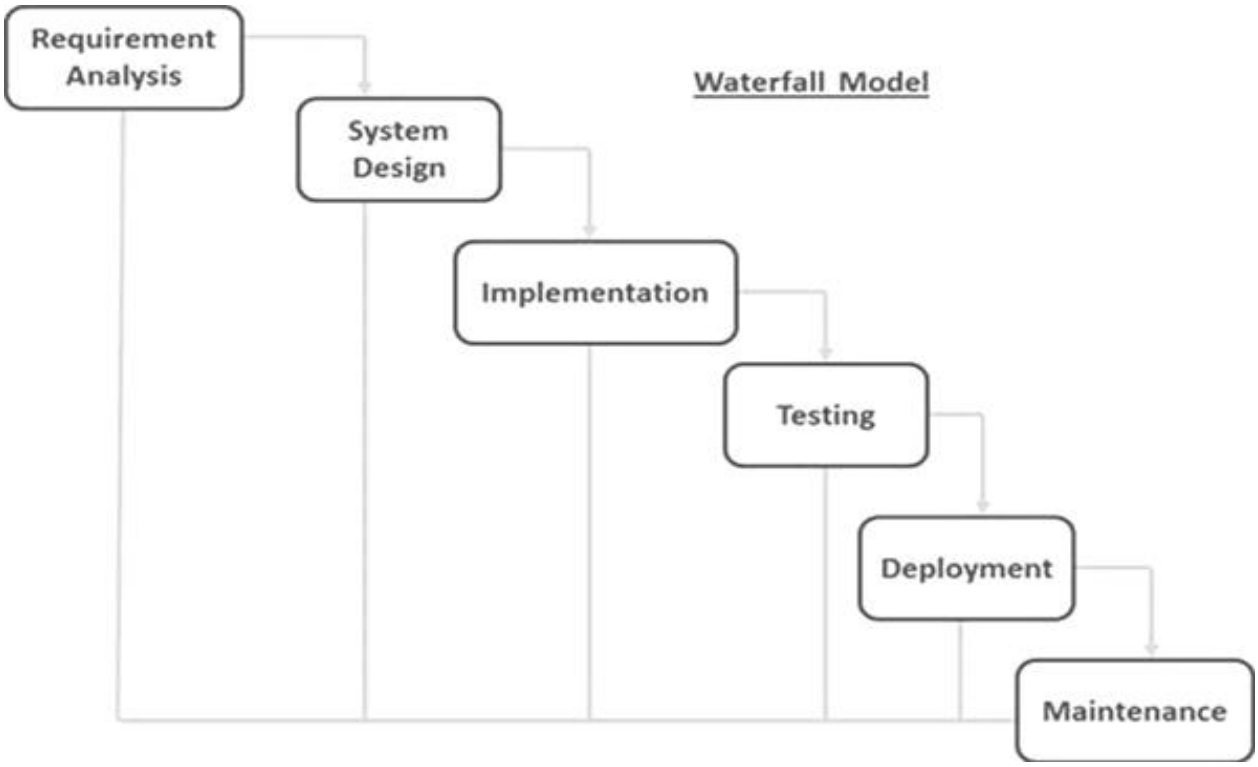


Figure 3: Waterfall Model

- **Requirement Analysis**
  It involves understanding what need to be design and what is its function, purpose etc. Here, the specifications of the input and output or the final product are studied and marked.

- **System Design**
  The requirement specifications from first phase are studied in this phase and system design is prepared. System Design helps in specifying hardware and system requirements and also helps in defining overall system architecture.

- **Implementation**
  With inputs from system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality which is referred to as Unit Testing.

- **Testing**
  All the units developed in the implementation phase are integrated into a system after testing of each unit. The software designed, needs to go through constant software testing to find out if there are any flaw or errors.

- **Deployment**
  Once the functional and non-functional testing is done, the product is deployed in the customer environment or released into the market.

- **Maintenance**
  This step occurs after installation, and involves making modifications to the system or an individual component to alter attributes or improve performance.

2.7 Design Pattern

For this project I chose, Model View Controller (MVC). MVC design pattern specifies that an application consist of a data model, presentation information, and control information. The pattern requires that each of these be separated into different objects. MVC is more of an architectural pattern, but not for complete application.

- The **Model** contains only the pure application data, it contains no logic describing how to present the data to a user.
- The **View** presents the model's data to the user. The view knows how to access the model's data, but it does not know what this data means or what the user can do to manipulate it.
- The **Controller** exists between the view and the model. It listens to events triggered by the view (or another external source) and executes the appropriate reaction to these events. In most cases, the reaction is to call a method on the model. Since the view and the model are connected through a

notification mechanism, the result of this action is then automatically reflected in the view.

Following are the reasons to choose MVC as the design pattern for this application:

- Multiple developers can work simultaneously on the model, controller and views.
- Changes doesn't affect the model.
- Development process is faster with MVC

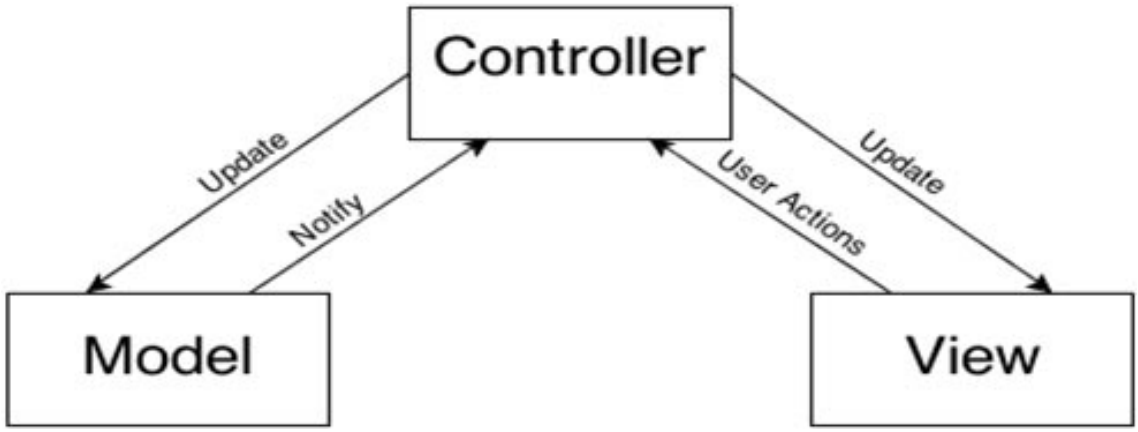


Figure 4: MVC Pattern

2.8 Architecture

A 3-tier architecture is a client-server architecture in which the functional process logic, data access, computer data storage and user interface are developed and maintained as independent modules on separate platforms. Three-tier architecture is a software design pattern and a well-established software architecture.

This architecture is chose for following reasons:

- It helps update one tier's technology stack without affecting other application a reas.
- It helps scale up and out the application.

- It adds reliability and more independence of the underlying servers or services.
- It helps
  cache requests, network usage are minimized and Application and Data Tier l oads are reduced. Any tier can load-balanced if necessary.

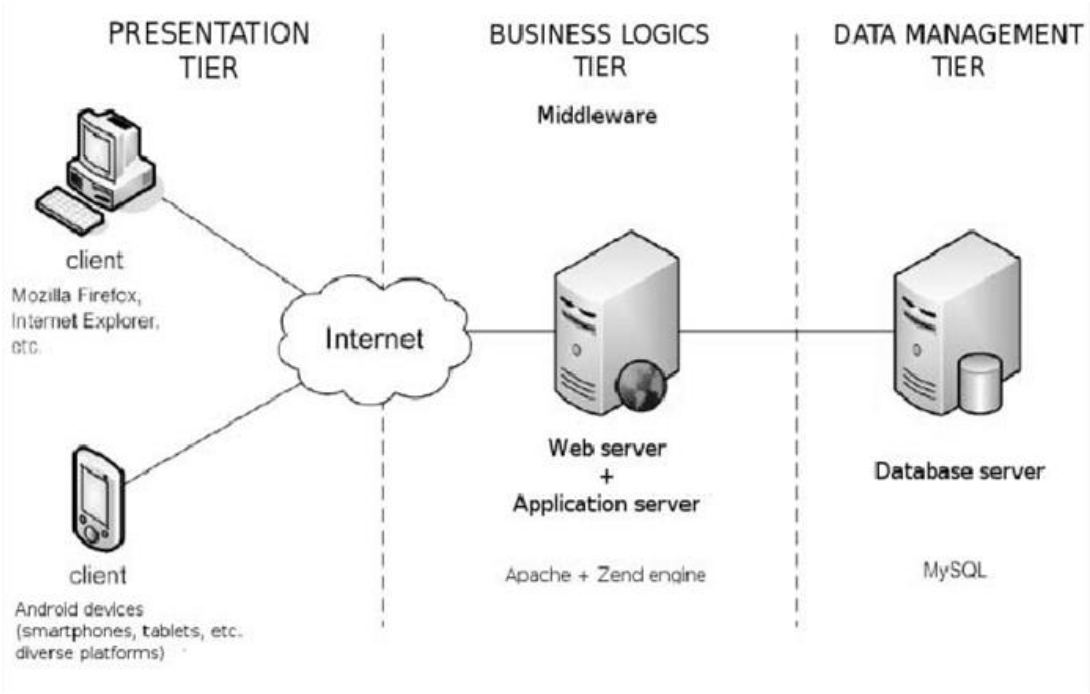


Figure 5: 3-Tier Architecture

Chapter 3: Design

Design

3.1 Justification

The system is designed to meet the requirements identified in the previous ph ases during the design phase.The requirements identified in the Requirements Analy sis Phase are transformed into a system design document which accurately describe s the system design and can be used in the next phase as an input to system develo pment.

The Design Phase's purpose is to transform the requirements into complete a nd detailed specifications for system design. The Development Team starts the Deve lopment Phase once the design is approved. It involves dynamic modeling, structural modeling and database modeling.

3.2 Dynamic Modeling

3.2.1 Sequence Diagram:

Sequence Diagrams are interaction diagrams that detail how operations are carried out. Sequence Diagrams show elements as they interact over time and they are organized according to object (horizontally) and time (vertically). The sequence diagram is used primarily to show the interactions between objects in the sequential order that those interactions occur.

Justifications:

Here, I have drawn sequence diagram to show the high-level interaction between active objects in a system, model the interaction between object instances within a collaboration that realizes a use case, the interaction between objects within a collaboration that realizes an operation and to show all the possible paths through the interaction.

Advantages:

- The sequence diagram may be used in object-oriented analysis to validate class diagrams against use cases, or to show the timing of interactions between entities within the system scope.

Disadvantages:

- A sequence diagram must be defined for each possible scenario. Strictly speaking, a sequence diagram requires a fully defined class model.

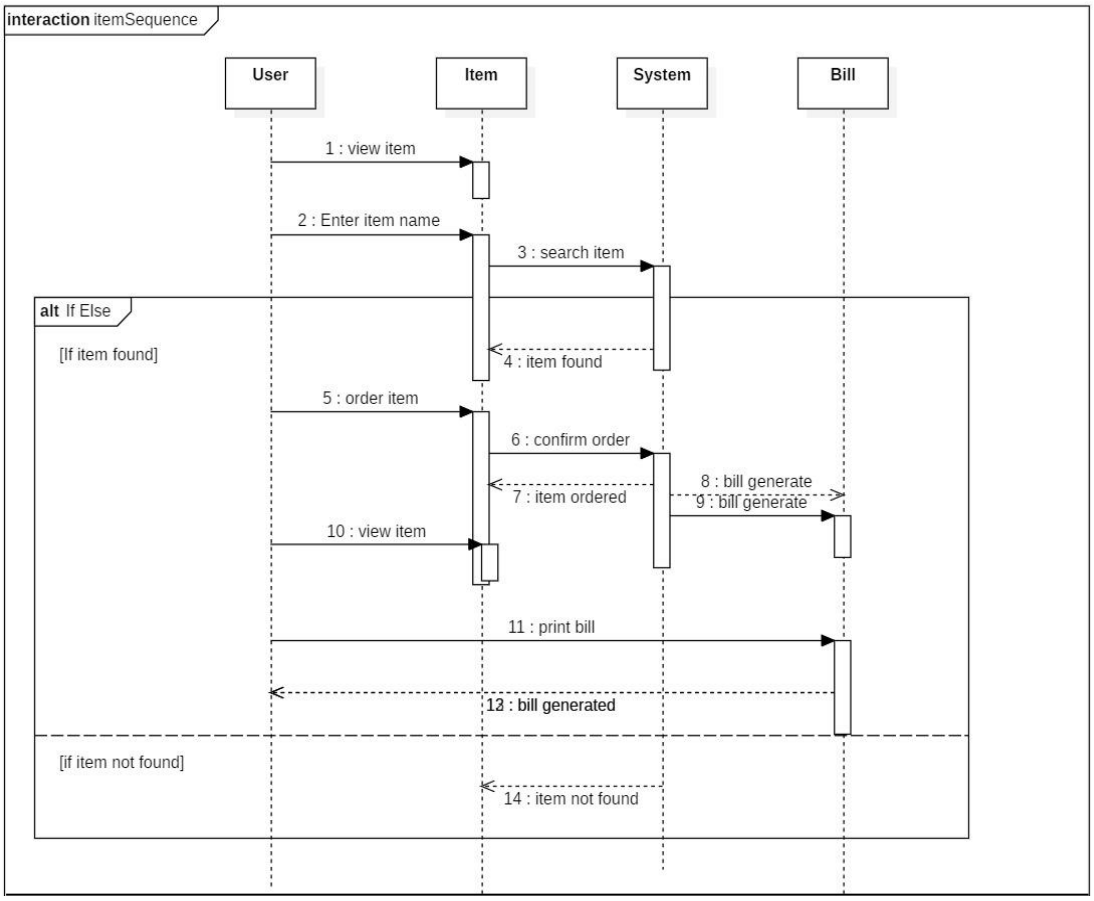


Figure 6: Item Sequence Diagram

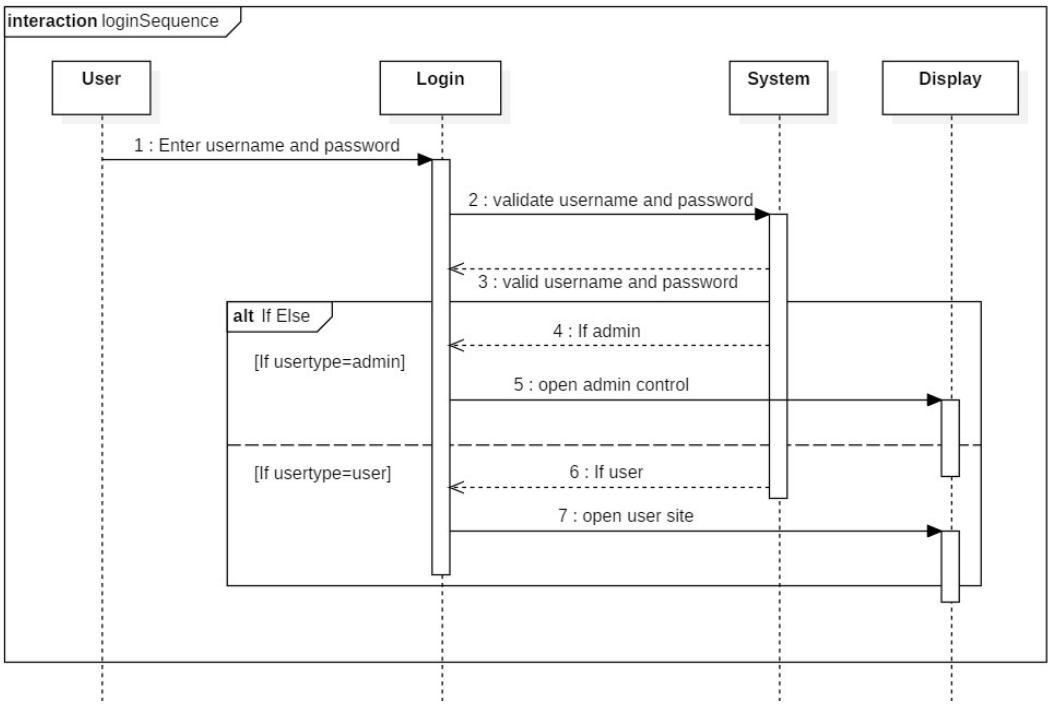


Figure 7: Login Sequence Diagram

3.2.2 Activity Diagram:

An **activity diagram** is a graphical representation that describe dynamic aspects of the system which is generally considered as an advanced version of flow chart that models the flow from one activity to another activity.

Justifications:

The activity diagram here describes how
to coordinate activities to deliver a service that can be at various levels of abstraction
.
Typically, some operations need to achieve an event, especially where the operation is intended to accomplish a number of different things requiring coordination, or how events in a single use case relate to each other, in particular using cases where acti vities may overlap and need coordination..

Advantages:

- UML modeling language included activity diagrams for both analysts and stak eholders are usually easily comprehensible.
- Because they are among the most user-friendly diagrams available, they are generally considered an essential tool in the repertoire of an analyst.
- In addition, as mentioned above, activity diagrams allow an analyst to display multiple conditions and actors through the use of swimlanes within a workflow. However, swimlanes are optional as a single condition or actor is normally sh own without the repertoire of them.

Disadvantages:

- UML modeling language includes activity diagrams that can become overly co mplex because their user-friendly nature can lend itself to an all-inclusive description.

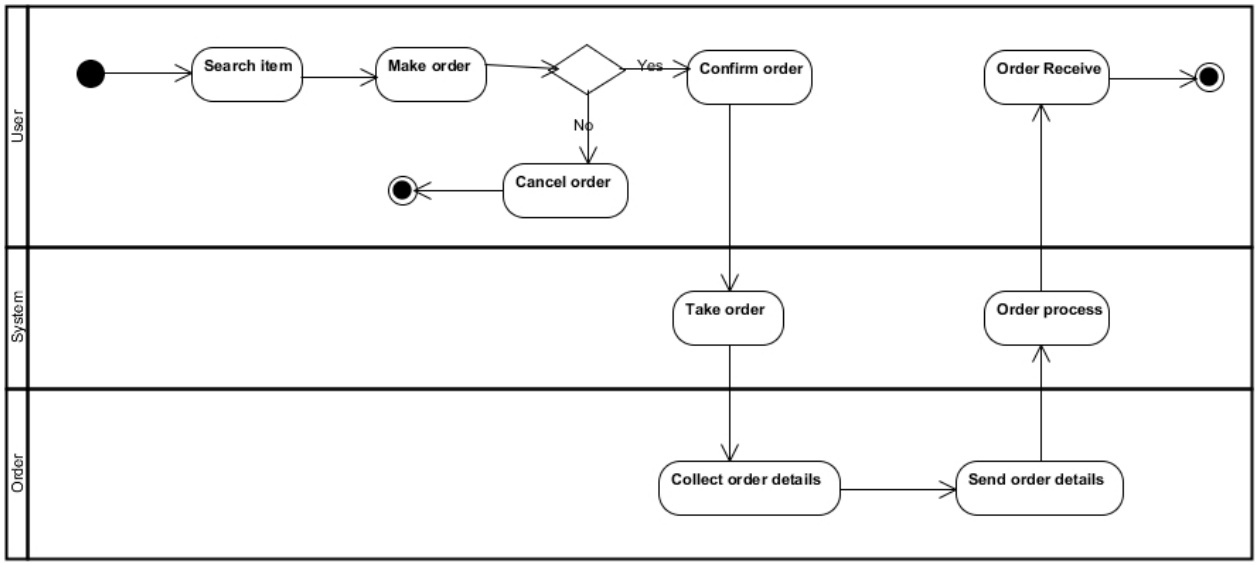


Figure 8: Order Item Activity Diagram

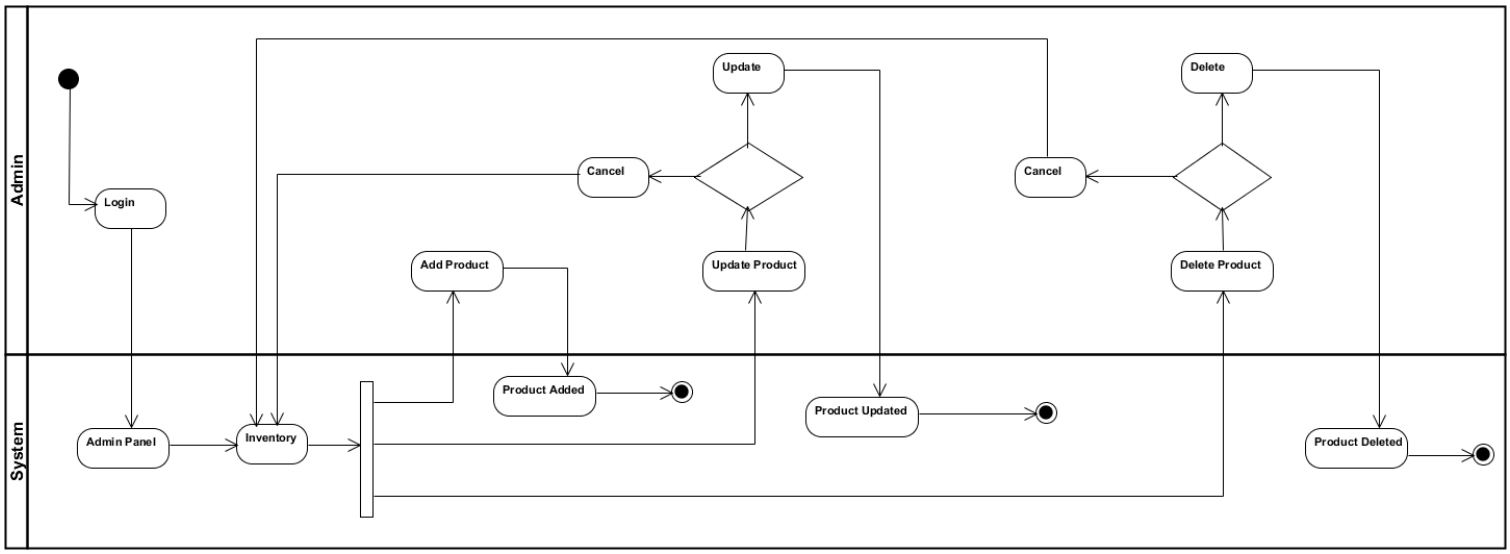


Figure 9: Inventory Activity Diagram

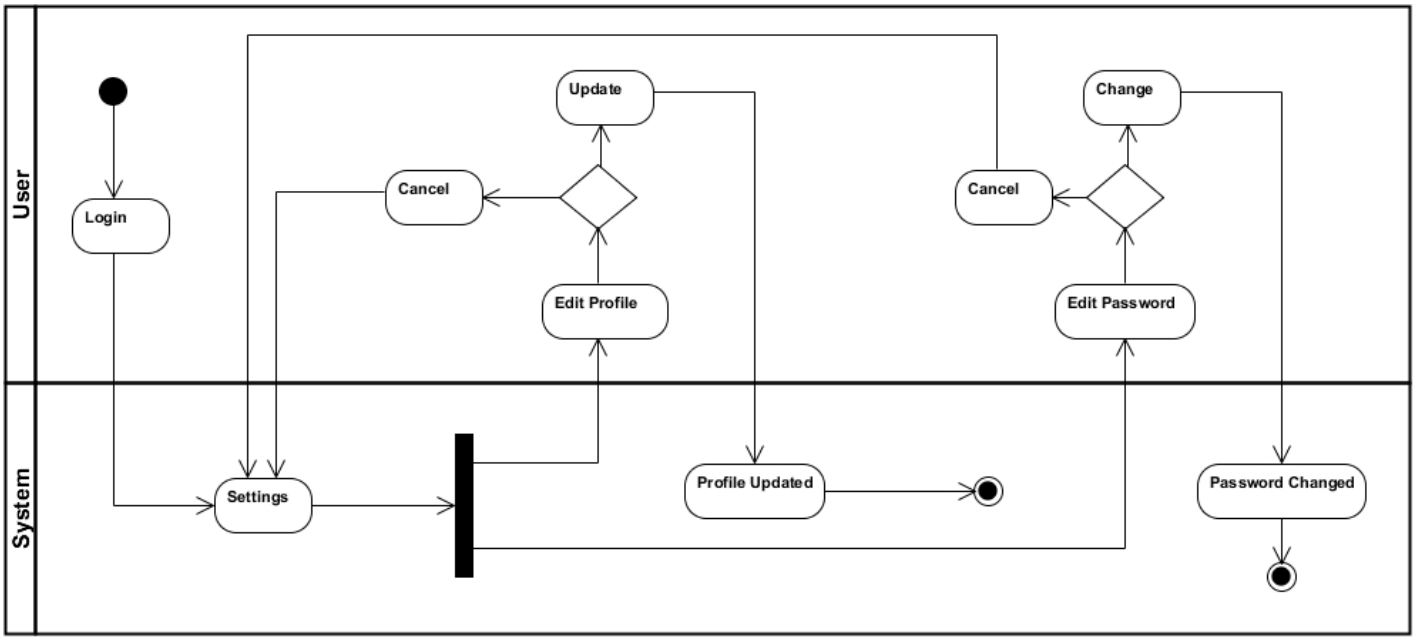


Figure 10: Update Profile Activity Diagram

3.3 Structural Modeling

3.3.1 Class Diagram:

A class diagram is an illustration of the Unified Modeling Language (UML) rela tionships and source code dependencies between classes.
A class defines the methods and variables in an object in this context, which is a spe cific entity in a program or code unit representing that entity. In all forms of object-oriented programming (OOP), class diagrams are useful. The concept is multiple.

Justification:

The classes are arranged in a class diagram in groups that share common fea tures.
A class diagram is similar to a flowchart in which classes are portrayed as box es with three rectangles inside each box.
The top rectangle contains the class names; the middle rectangle contains the class attributes; the lower rectangle contains the class methods, also known as operations.

Advantages:

- It forces the programmer to think about his / her classes ' structure and how they interact with each other before they actually write any code. T his can result in a more robust application.
- It provides a blueprint for maintenance programmers to get an overview of how the application is structured before examining the actual code. This may reduce maintenance time.

Disadvantages:

- The programmer may need to learn UML to build the class diagram in the first place.
- The time spent building the class diagram may add to overall development time.

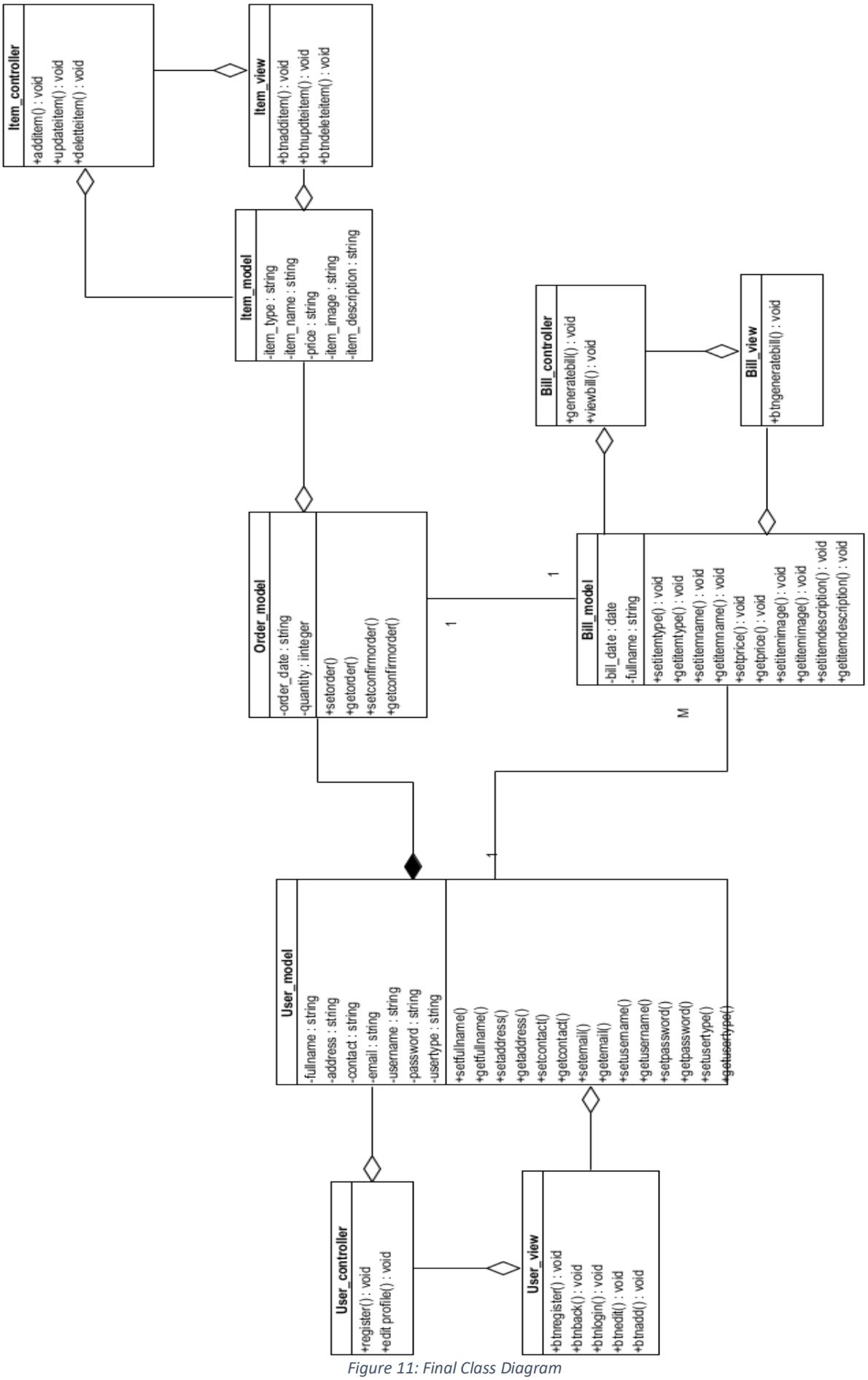


Figure 11: Final Class Diagram

3.4 Database Modeling

3.4.1 ER Diagram:

An **entity-relationship diagram (ERD**) is a data modeling technique that graphically illustrates an information system's entities and the relationships between those entities. An **ERD** is a conceptual and representational model of data used to represent the entity framework infrastructure.

Justification:

An entity relationship diagram (ERD), also referred to as an entity relationship model, is a graphical representation of an information system depicting the rel ationships within that system between people, objects, places, concepts, or ev ents.An ERD is a data modeling technique that can be used as the basis for a relational database to help define business processes.

Advantages:

- ER model is very simple because we can easily draw an ER diagram if we know the relationship between entities and attributes.
- ER model is a diagramm of any logical database structure. We can eas ily understand the relationship between entities and relationship by vie wing the ER diagram.
- For database designer, it is an effective communication tool.

Disadvantages:

- Some information in the ER model is lost or hidden
- ER model represents a limited relationship in comparison with other mo dels of data such as relational model etc.
- Data manipulation in the ER model is hard to show.

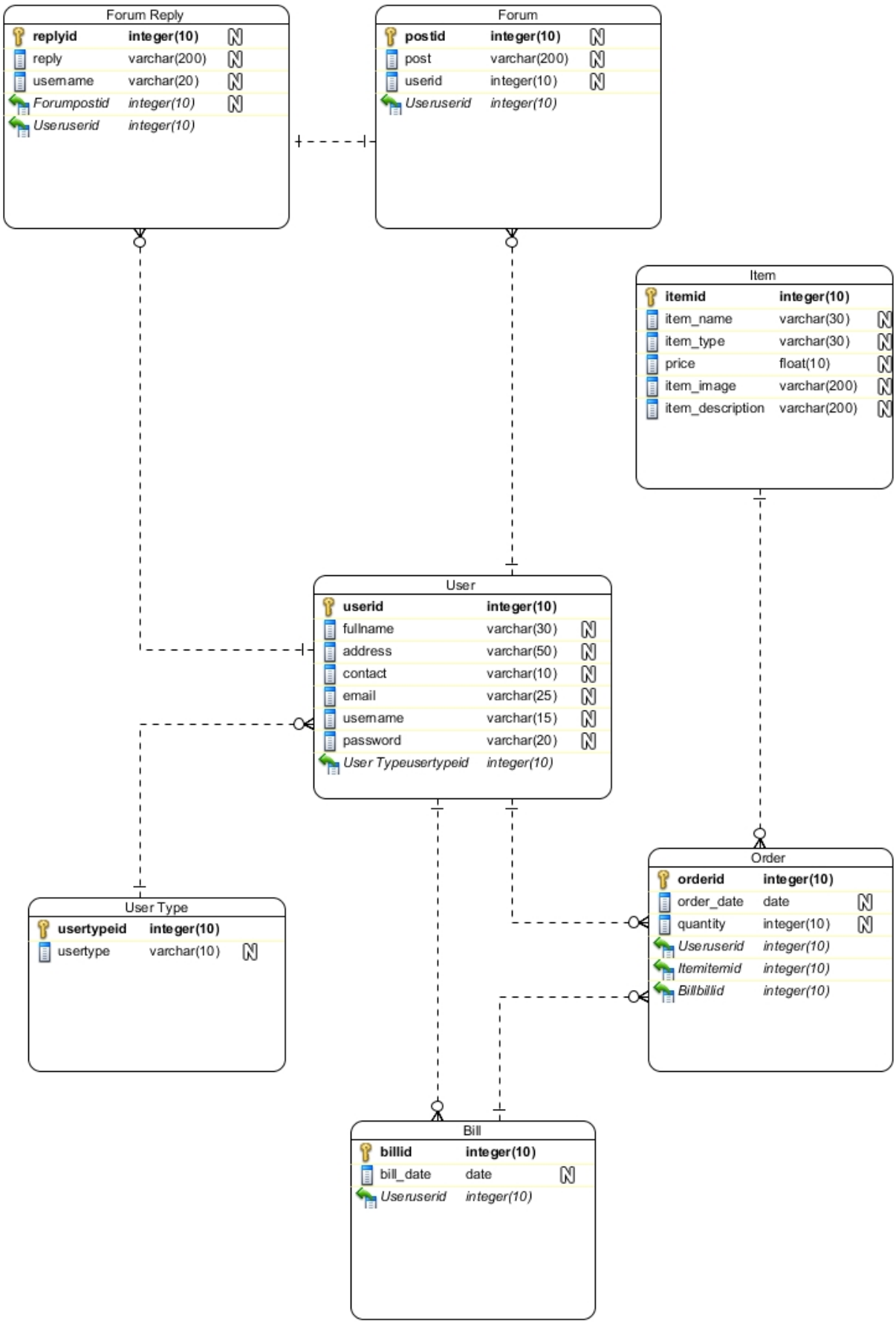


Figure 12: ER Diagram

3.4.2 Data Flow Diagram

A data flow diagram (DFD) illustrates how data is processed by a system in terms of inputs and outputs. As its name indicates its focus is on the flow of information, where data comes from, where it goes and how it gets stored.

Justification:

Essentially, there are two different types of notes for data flow diagrams (Yourdon & Coad or Gane & Sarson) that define different visual representations for processes, d ata stores, data flow and external entities.

Data flow diagrams typed by Yourdon and Coad are usually used for system a nalysis and design, while DFDs typed by Gane and Sarson are more common when viewing information systems.Visually, how processes look is the biggest difference b etween the two ways to draw data flow diagrams. Processes are depicted as circles i n the Yourdon and Coad mode, while the processes are squares with rounded corne rs in the Gane and Sarson diagrams.

Advantages:

- It aids in describing the boundaries of the system.
- It is beneficial for communicating existing system knowledge to the users.
- A straightforward graphical technique which is easy to recognize.
- DFDs can provide a detailed representation of system components.
- It is used as the part of system documentation file.

Disadvantages:

- It make the programmers little confusing concerning the system.
- The biggest drawback of the DFD is that it simply takes a long time to create, so long that the analyst may not receive support from management to complete it.
- Physical considerations are left out.

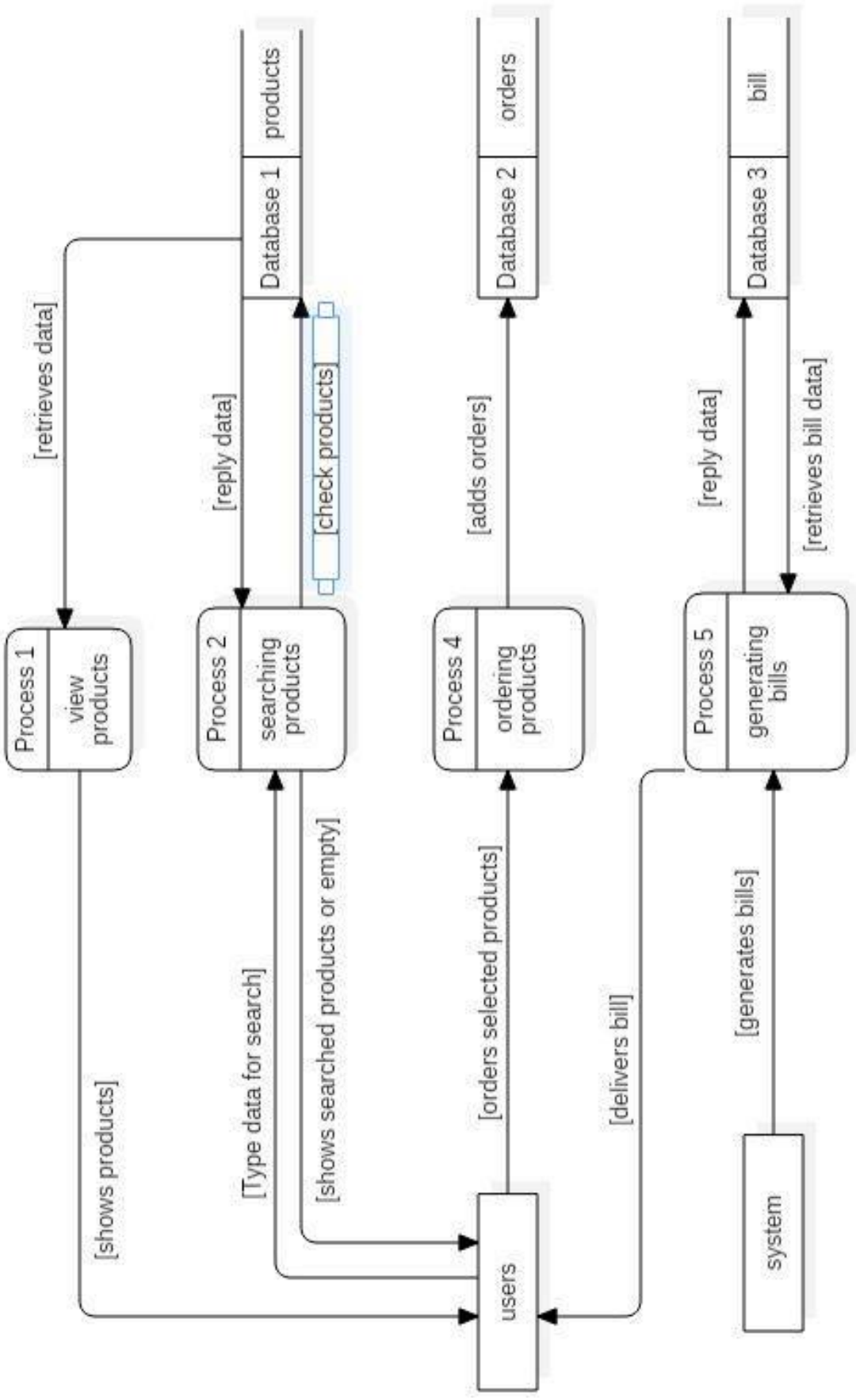


Figure 13: Data Flow Diagram

3.5 Data Dictionary

Data dictionary is a simple yet systematic table that encompasses particular data and information own structures, which is called Metadata. The data dictionary is a crucial component of any relational database. Ironically, because of its importance, it is invisible to most database users. Typically, only database administrators interact with the data dictionary.

Advantages:

- It gives the well-structured and clear information about the database. One can analyze the requirement, any redundancy like duplicate columns, tables etc. Since it provides a good documentation on each object, it helps to understand the requirement and design to the great extent.
- It is very helpful for the administrator or any new DBA to understand the database. Since it has all the information about the database, DBA can easily able to track any chaos in the database.

Disadvantages:

- Creating a new data dictionary is a very big task. It will take years to create one.
- It should be well designed in advance to take all the advantages of it. Otherwise, it will create problems throughout its life.
- The cost of data dictionary will be bit high as it includes its initial build and hardware charges as well as cost of maintenance

User Type					
Column Type	**Type**	**Length**	**Nullable**	**PK/FK**	**Description**
usertypeid	integer	10	No	PK	Unique Identification
usertype	varchar	10	Yes		Types of User

User					
Column Type	**Type**	**Length**	**Nullable**	**PK/FK**	**Description**
userid	integer	10	No	PK	Unique Identification
fullname	Varchar	30	Yes		Fullname of User
address	Varchar	50	Yes		Address of User
contact	Varchar	10	Yes		Contact of User
email	Varchar	25	Yes		Email of User
username	varchar	15	Yes		Username
password	varchar	20	Yes		Password
usertypeid	integer	10	Yes	FK	Unique Identification

Item					
Column Type	**Type**	**Length**	**Nullable**	**PK/FK**	**Description**
itemid	Integer	10	No	PK	Unique Identification
item_name	Varchar	30	Yes		Name of Item
item_type	Varchar	30	Yes		Type of Item
price	Float	10	Yes		Price of Item
item_image	varchar	200	Yes		Image of Item
item_description	varchar	200	Yes		Description of Item

Bill					
Column Type	**Type**	**Length**	**Null**	**PK/FK**	**Description**
billid	Integer	10	No	PK	Unique Identification
bill_date	Date		Yes		Date of Bill Created
orderid	Integer	10	Yes	FK	Unique Identification
userid	Integer	10	Yes	FK	Unique Identification

Order					
Column Type	**Type**	**Length**	**Nullable**	**PK/FK**	**Description**
orderid	Integer	10	No	PK	Unique Identification
order_date	Date		Yes		Date of Order
quantity	Integer	10	Yes		Quantity of Items
userid	Integer	10	Yes	FK	Unique Identification
itemid	Integer	10	Yes	FK	Unique Identification
billid	Integer	10	Yes	FK	Unique Identification

Forum					
Column Type	**Type**	**Length**	**Null**	**PK/FK**	**Description**
postid	Integer	10	No	PK	Unique Identification
post	Date		Yes		Posts by users
usernme	Varchar	30	Yes		Name of the user who posted
userid	Integer	10	Yes	FK	Unique Identification

Forumreply					
Column Type	**Type**	**Length**	**Null**	**PK/FK**	**Description**
replyid	Integer	10	No	PK	Unique Identification
reply	Date		Yes		replies by users
usernme	Varchar	30	Yes		Name of the user who replied
postid	Integer	10	Yes	FK	Unique Identification
userid	Integer	10	Yes	FK	Unique Identification

3.6 Prototype

A prototype is an original model, form or an instance that serves as a basis for other processes. In software technology, the term prototype is a working example through which a new model or a new version of an existing product can be derived.

1. Login

Chapter 4: Implementation

Implementation

4.1 Framework

It is the post-design phase where the project plan will begin to work on those plans. The project's entire design is now changing in coding.
Accommodation planning will be coded for all projects using PHP programming language. PHP stands for Preprocessor Hypertext (no, the acronym does not follow the name). It is an open source, server-side, and
language of scripting used for web application development.

By scripting language, we mean a program written for task automation based on scri pts (code lines). Laravel is the framework that is used for this project.
Laravel is an expressive, elegant web application framework.

4.2 User Interface (UI)

The user interface (UI) is everything designed into an information device with which a person may interact. This can include display screens, keyboards, mouse and the appearance of a desktop. It is also the way through which a user interacts with an application or a website. The growing dependence of many companies on web applications and mobile applications has led many companies to place increased priority on UI in an effort to improve the user's overall experience.

The screenshots of working application/UI are included in this report (see Appendix) along with the screenshots of its Laravel and HTML codes, respectively.

Chapter 5: Testing

Testing

Introduction

Software testing is an investigation conducted to provide stakeholders with inf ormation on the quality of the software product or service being tested.[1] Software t esting can also provide an objective, independent view of the software to enable the company to appreciate and understand the risks of implementing software.
Testing techniques include the execution of a program or

Test case or Test plan: It is the documentation of specific test with objectives, resource and process of testing.

Unit Testing

UNIT TESTING is a software test level where a software tests individual units / comp onents.
The purpose is to validate the performance as designed by each unit of the software.
A unit is any software's smallest testable part.
Usually it has one or several inputs and usually one output.

Advantages of unit testing;

- ✓ Fix bugs in early development cycle.
- ✓ Improves the quality of code.
- ✓ Reduces defects and bugs.
- ✓ Less costly.

Black box testing

BLACK BOX TESTING is a software testing method in which the tester does not kno w the internal structure / design / implementation of the item being tested. These test s, though usually functional, may be functional or non-functional.Advantages of black box testing;

- ✓ Validates the functional requirement.
- ✓ Test can be done through end user’s point of view.
- ✓ No need of technical person for testing.

Test case no.	T1
Purpose of the case	Exploring Home Page
Test data	url: 127.0.0.1:8000
Class name	UnitTest
Function name	testURL
Expected result	Open home page
Actual result	Open home page (index)
Conclusion	Actual results matches the expected results

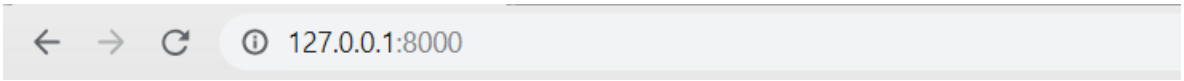
5.1 URL

Unit Testing

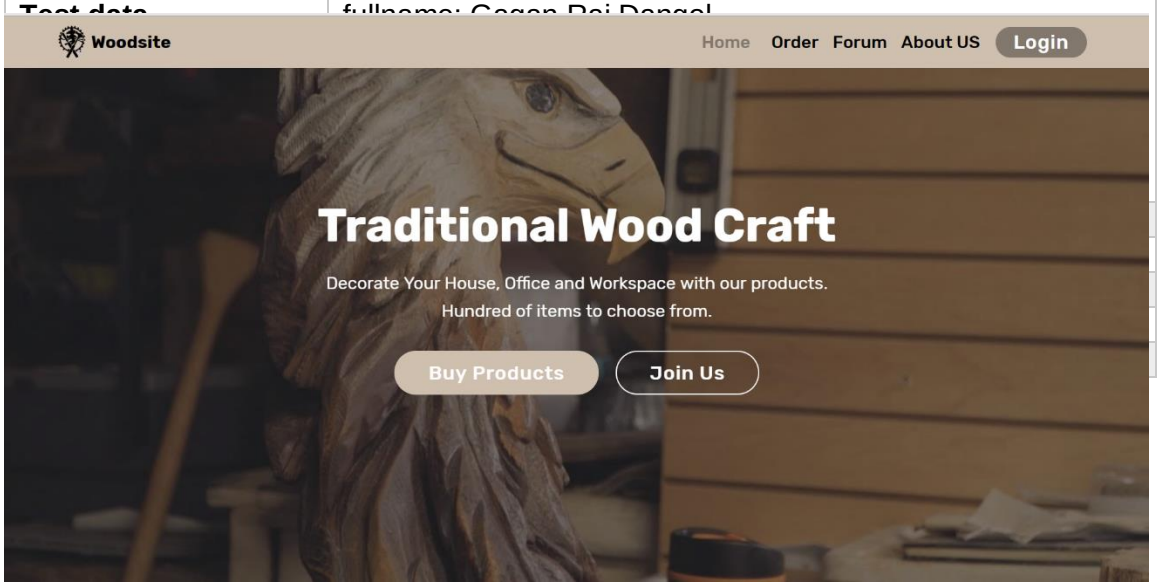
```
public function testURL()
{
    $response = $this->call('GET','/');
    $this->assertEquals(200,$response->status());
}
```

```
C:\xampp\htdocs\wcms>vendor\bin\phpunit --filter testURL
PHPUnit 7.5.2 by Sebastian Bergmann and contributors.

Runtime:       PHP 7.2.5
Configuration: C:\xampp\htdocs\wcms\phpunit.xml

.                                                                   1 / 1 (100%)

Time: 474 ms, Memory: 10.00MB

OK (1 test, 1 assertion)
```

Black box Testing



Test case no.	T2
Purpose of the case	Registering new user
Test data	fullname: Gagan Raj Dangol



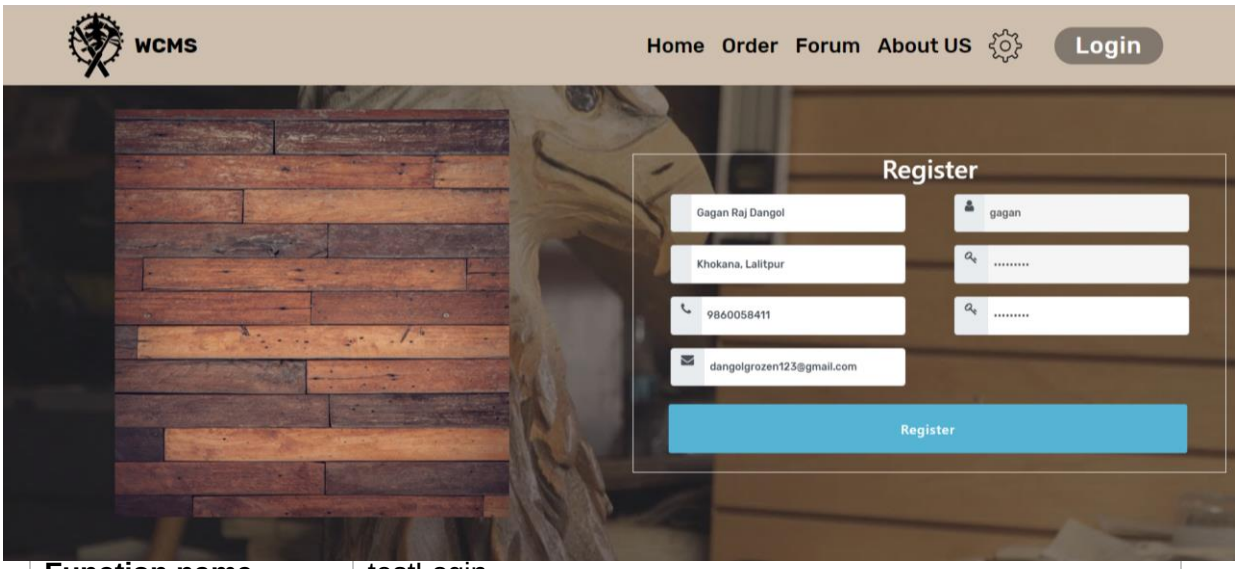
5.2 Registration

Unit Testing

```php
public function testRegister()
{
    $response = $this->call("GET",'/register',[
        'fullname' => "Gagan Raj Dangol",
        'address' => "Khokana, Lalitpur",
        'contact' => "9860058411",
        'email' => "dangolgrozen123@gmail.com",
        'username' => "gagan",
        'password' => "gagrangol",
        'usertypeid' => 2
    ]);

    $this->assertEquals(200,$response->status());
}
```

```
C:\xampp\htdocs\wcms>vendor\bin\phpunit --filter testRegister
PHPUnit 7.5.2 by Sebastian Bergmann and contributors.

Runtime:       PHP 7.2.5
Configuration: C:\xampp\htdocs\wcms\phpunit.xml

.

Time: 4.18 seconds, Memory: 12.00MB

OK (1 test, 1 assertion)
```

Black box Testing



Function name	testLogin
Expected result	Login and go to index
Actual result	Login and go to index
Conclusion	Actual results matches the expected results

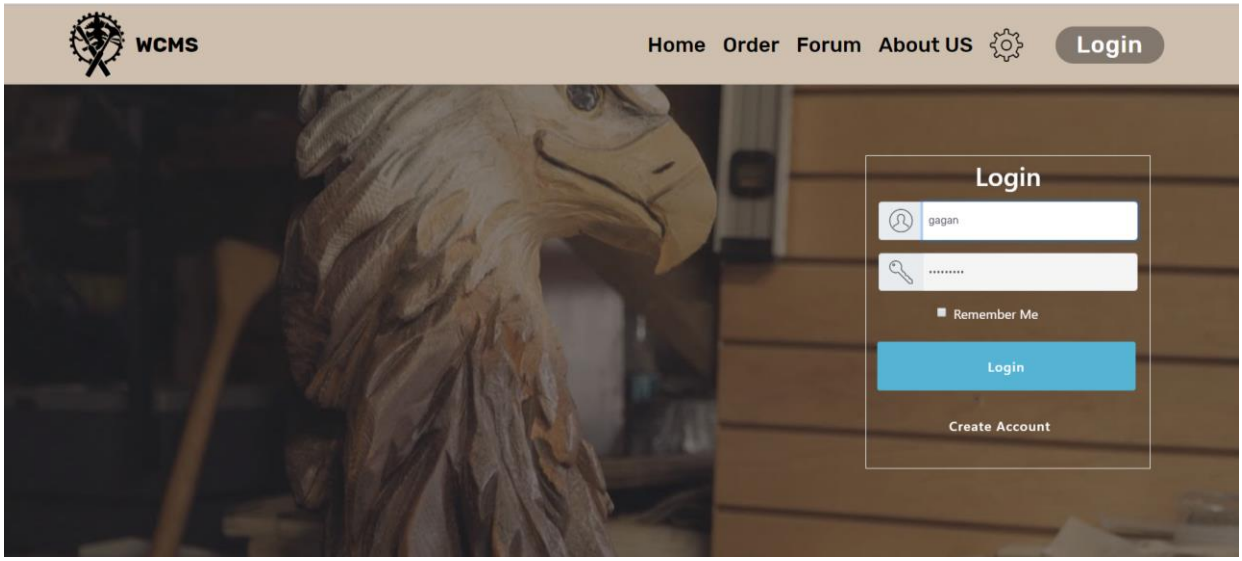
5.3 Login

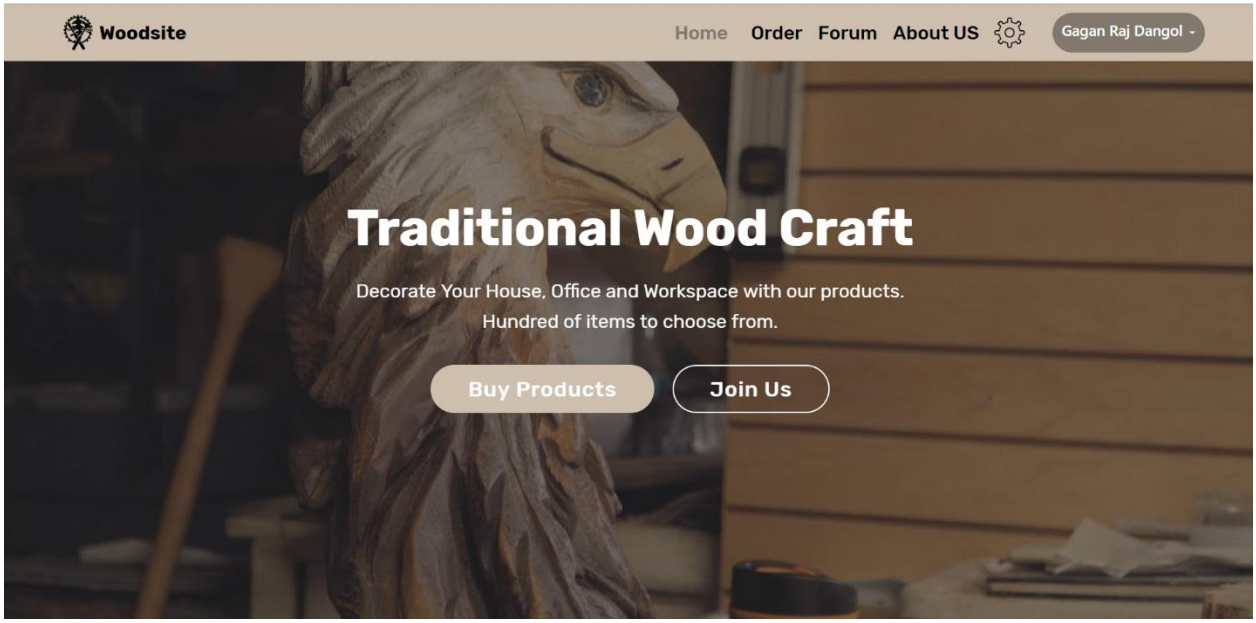
Unit Testing

```
public function testLogin()
{

    $username="gagan";
    $password="gagrangol";

    $response = $this->call("GET",'wcms/login',[
        'username'=>$username,
        'password' => $password
    ]);

    $this->assertEquals(200,$response->status());
}
```

```
C:\xampp\htdocs\wcms>vendor\bin\phpunit --filter testLogin
PHPUnit 7.5.2 by Sebastian Bergmann and contributors.

Runtime:       PHP 7.2.5
Configuration: C:\xampp\htdocs\wcms\phpunit.xml

.                                                                   1 / 1 (100%)

Time: 843 ms, Memory: 10.00MB

OK (1 test, 1 assertion)
```

Black box Testing





Test case no.	T4
Purpose of the case	Inserting new item
Test data	item_name: Ganesh item_type: C price: 12000 item_description: Idol
Class name	UnitTest
Function name	testAddItem
Expected result	Add item and go to inventory
Actual result	Add item and go to inventory
Conclusion	Actual results matches the expected results

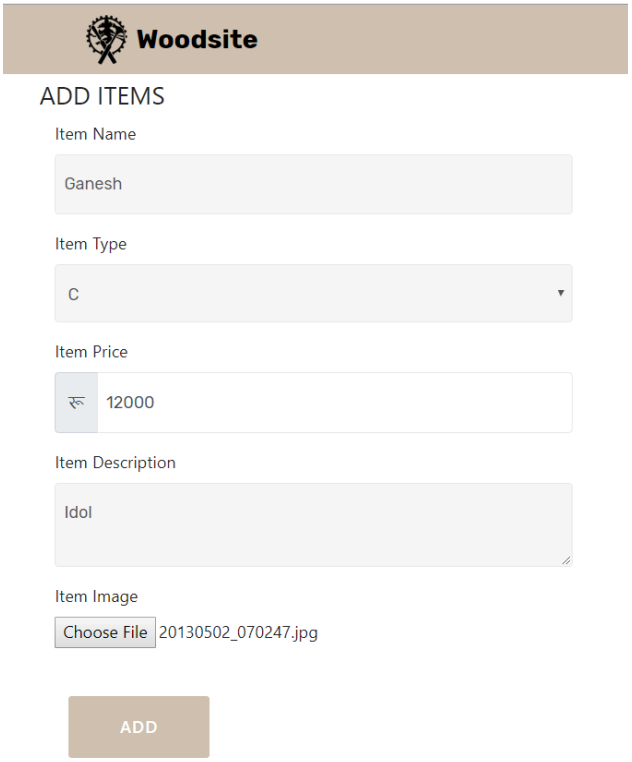
5.4 Add Item

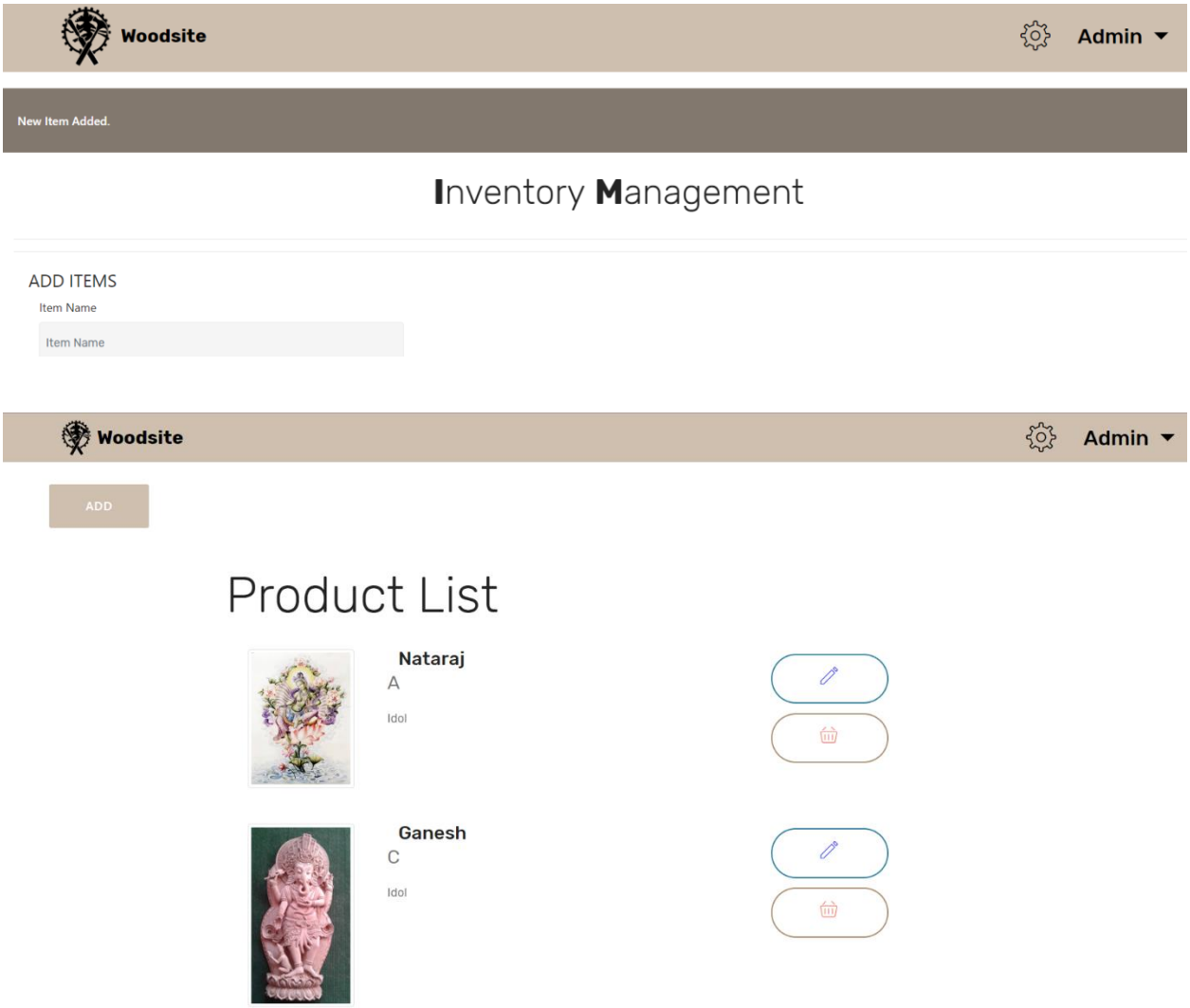
Unit Testing

```
public function testAddItem()
{
    $response = $this->call('PUT', '/inventory',[
        'itemid' => '3',
        'item_name' => 'Bhimsen',
        'item_type' => 'D',
        'price' => 10000,
        'item_description' => 'qwertyuioplkjhgfdzxcvbn'
    ]);

    $this->assertEquals(500,$response->status());
}
```

```
C:\xampp\htdocs\wcms>vendor\bin\phpunit --filter testAddItem
PHPUnit 7.5.2 by Sebastian Bergmann and contributors.

Runtime:       PHP 7.2.5
Configuration: C:\xampp\htdocs\wcms\phpunit.xml

.

Time: 458 ms, Memory: 12.00MB

OK (1 test, 1 assertion)
```

Black Box Testing





Test case no.	T5
Purpose of the case	Deleting an item
Test data	Itemid: 2
Class name	UnitTest
Function name	testDeleteItem
Expected result	Delete item
Actual result	Delete item
Conclusion	Actual results matches the expected results

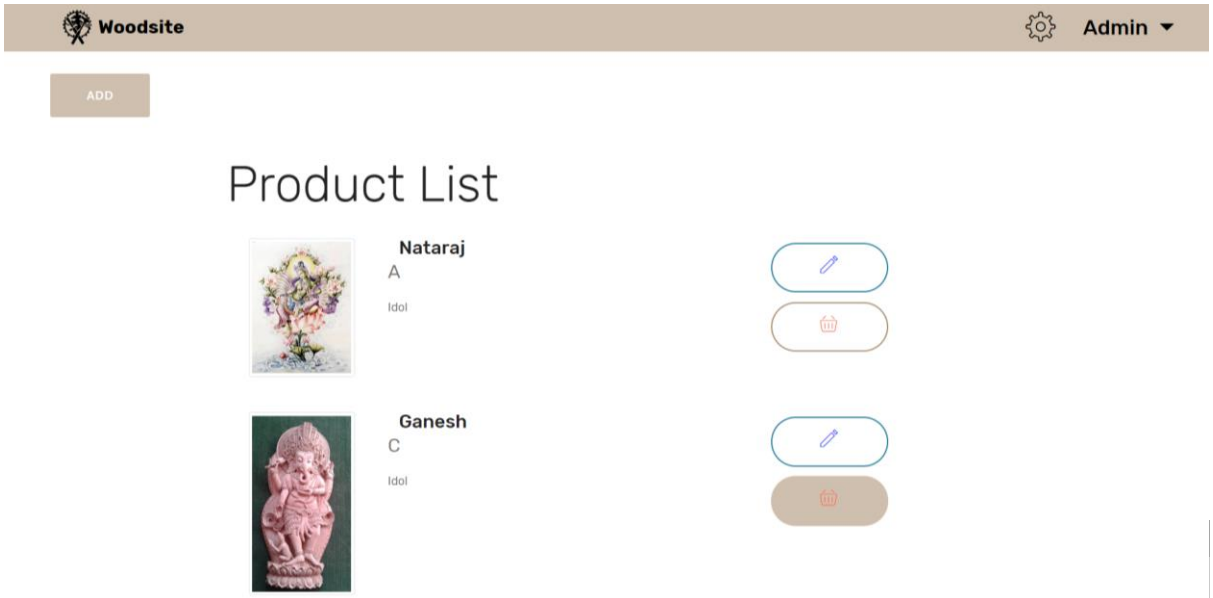
5.5 Delete Item

Unit testing

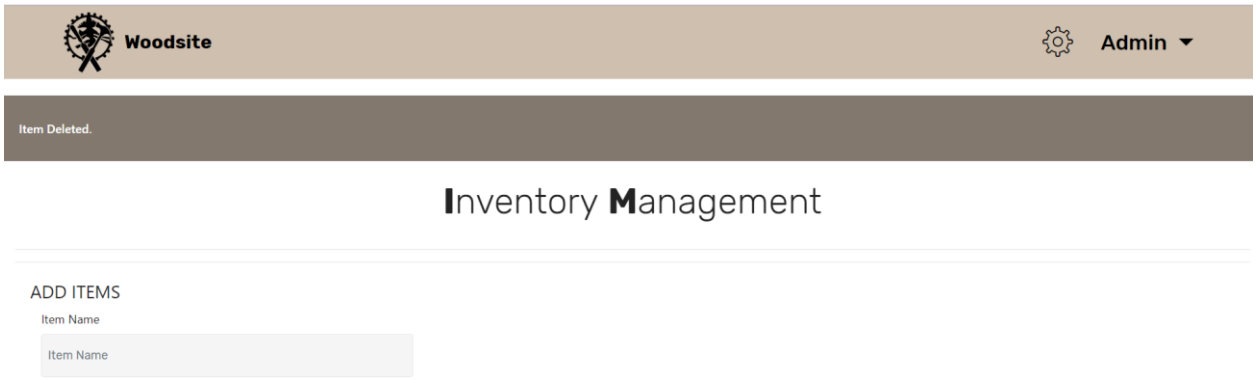
```
public function testItemDelete()
{
    $response = $this->call('DELETE', '/additems/1', [
        'itemid' => 2
    ]);

    $this->assertEquals(302,$response->status());
}
```

```
                    vendor\bin\phpunit --filter testItemDelete
PHPUnit 7.5.2 by Sebastian Bergmann and contributors.

Runtime:       PHP 7.2.5
Configuration: C:\xampp\htdocs\wcms\phpunit.xml

.

Time: 4.2 seconds, Memory: 12.00MB

OK (1 test, 1 assertion)

C:\xampp\htdocs\wcms>
```

Black Box Testing



Test data	No data
Class name	UnitTest
Function name	testItems
Expected result	Display item data
Actual result	Display item data
Conclusion	Actual results matches the expected results



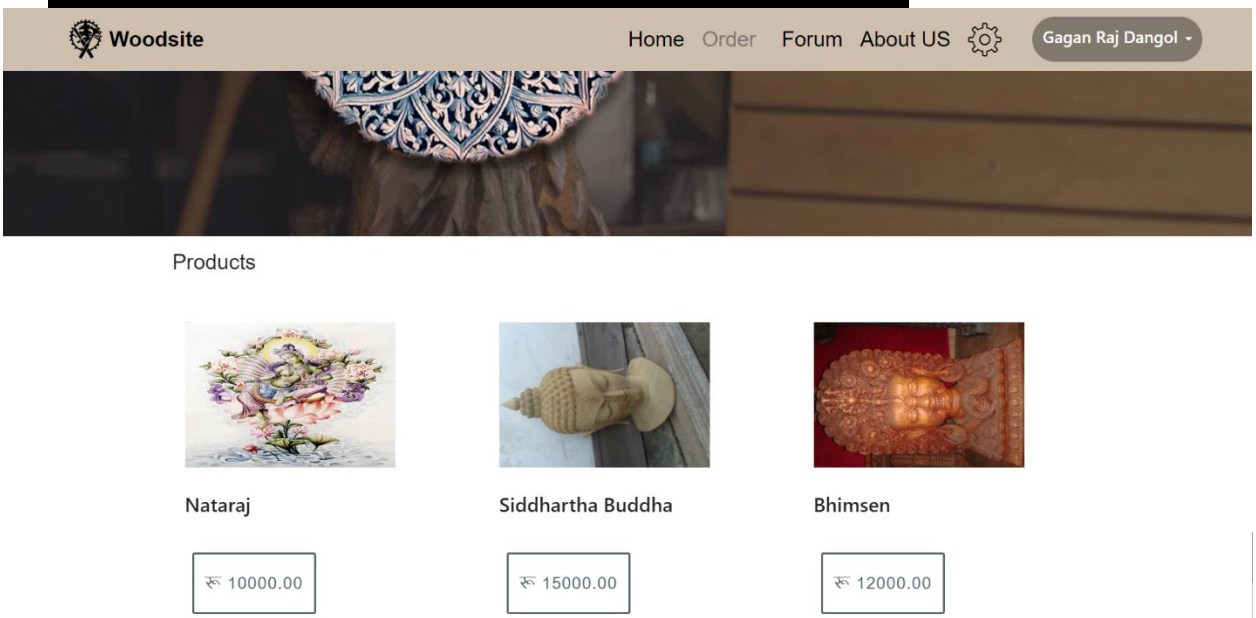
5.6 Items

Unit testing

```
public function testItems()
{
    $response = $this->call('GET', '/order2');

    $this->assertEquals(200,$response->status());
}
```

```
C:\xampp\htdocs\wcms>vendor\bin\phpunit --filter testItems
PHPUnit 7.5.2 by Sebastian Bergmann and contributors.

Runtime:       PHP 7.2.5
Configuration: C:\xampp\htdocs\wcms\phpunit.xml

.

Time: 1.83 seconds, Memory: 14.00MB

OK (1 test, 1 assertion)
```

Black Box Testing



Test data	order_date: 2019-04-09 quantity: 3 userid: 2 Itemid: 1
Class name	UnitTest
Function name	testOrderProcess
Expected result	insert into table order
Actual result	Insertion failed
Conclusion	Actual results didn't match the expected results

5.7 Order

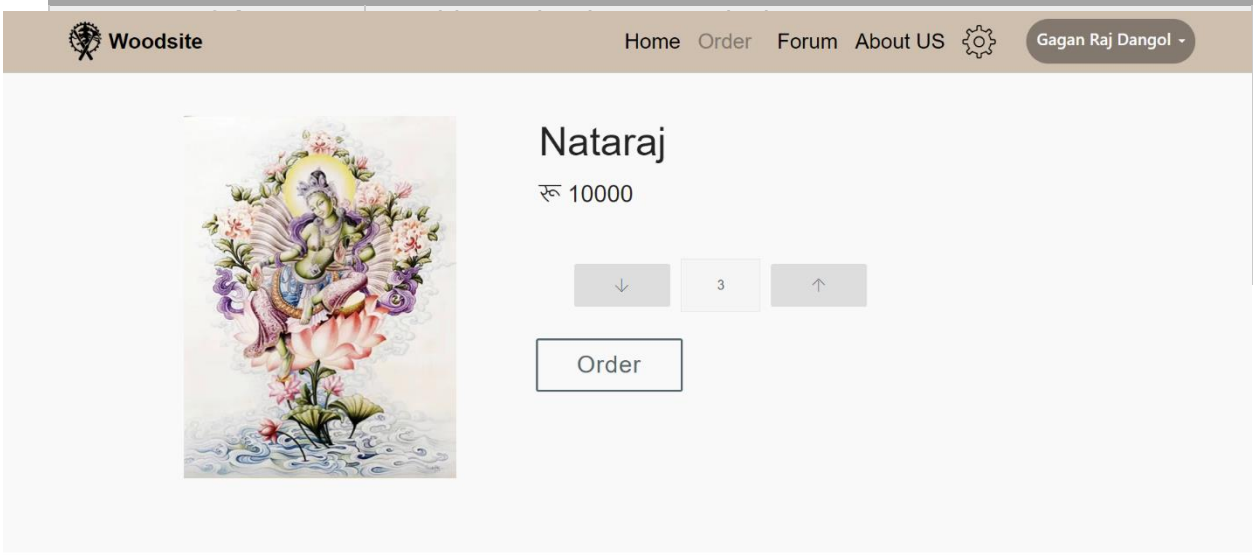
Unit testing

```php
public function testOrder()
{
    $response = $this->call('GET', '/orderprocess/1');
    $this->assertEquals(200,$response->status());
}
```

Expected result	fetch item data
Actual result	fetch item data
Conclusion	Actual results matches the expected results

```
C:\xampp\htdocs\wcms>vendor\bin\phpunit --filter testOrderprocess
PHPUnit 7.5.2 by Sebastian Bergmann and contributors.

Runtime:       PHP 7.2.5
Configuration: C:\xampp\htdocs\wcms\phpunit.xml

R                                                                   1 / 1 (100%)

Time: 1.68 seconds, Memory: 14.00MB

There was 1 risky test:

1) Tests\Unit\UnitTest::testOrderprocess
This test did not perform any assertions

C:\xampp\htdocs\wcms\tests\Unit\UnitTest.php:105

OK, but incomplete, skipped, or risky tests!
Tests: 1, Assertions: 0, Risky: 1.

C:\xampp\htdocs\wcms>
```

Test case no.	T8



Black Box Testing

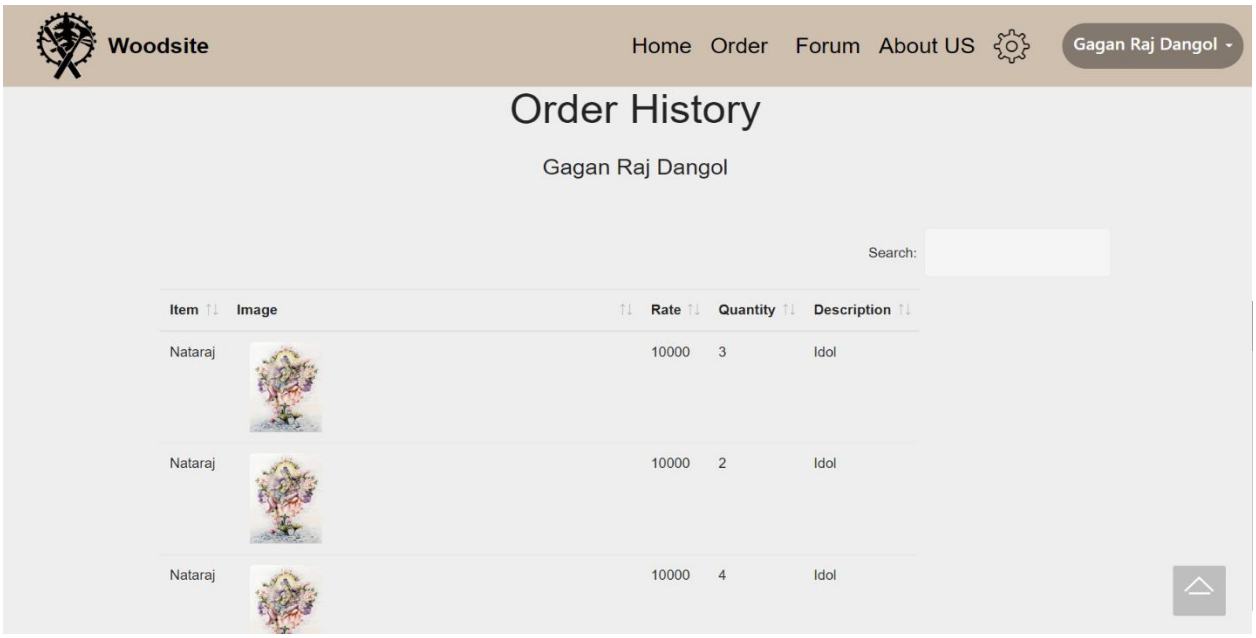
5.8 Order List

Unit Testing

```
public function testOrderlist()
{
    $response = $this->call('GET', '/orderlist', [
        'userid' => 2,
        'itemid' => 1
    ]);

    $this->assertEquals(500,$response->status());
}
```

```
C:\xampp\htdocs\wcms>vendor\bin\phpunit --filter testOrderlist
PHPUnit 7.5.2 by Sebastian Bergmann and contributors.

Runtime:       PHP 7.2.5
Configuration: C:\xampp\htdocs\wcms\phpunit.xml

.

Time: 442 ms, Memory: 14.00MB

OK (1 test, 1 assertion)
```

Black Box Testing



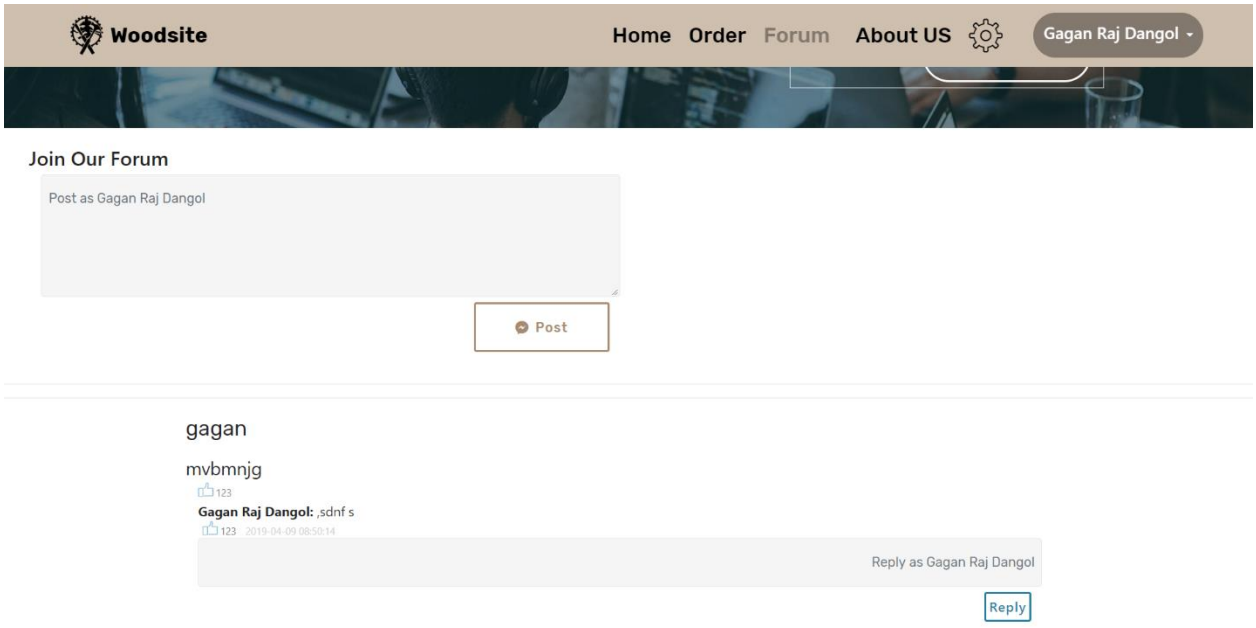
5.9 Forum

Unit Testing

```php
public function testForum()
{
    $response = $this->call('GET', '/post',[
        'postid' => 1
    ]);
    $this->assertEquals(200,$response->status());
}
```

```
C:\xampp\htdocs\wcms>vendor\bin\phpunit --filter testForum
PHPUnit 7.5.2 by Sebastian Bergmann and contributors.

Runtime:       PHP 7.2.5
Configuration: C:\xampp\htdocs\wcms\phpunit.xml

.

Time: 392 ms, Memory: 14.00MB

OK (1 test, 1 assertion)
```

Black Box Testing



Chapter 6: Other Project Issues

Other Project Issues

6.1 Risk Management

Risk management refers to the process of identifying potential risks in advance, analyzing them and taking precautionary steps to reduce the risk.

Risk management is held in following ways:

- Risk Identify
- Risk Analyze
- Risk Rank or Evaluate
- Risk Treatment
- Risk Review and Monitor

The impacts that can cause on the project due to the risks is calculated as:

Impact = Likelihood * Consequence

Risk Likelihood values are shown as follows

Likelihood	Value
Low	1
Medium	2
High	3

Table 2: Likelihood measurement

Risk Consequence values are shown below

Consequence	Value
Very low	1
Low	2
Medium	3
High	4
Very High	5

Table 3: Consequences measurement

Risk Management Chart

S. N	Risks	Likelihood	Consequences	Impact	Solution
1	Change in Requirements	3	4	12	thorough requirement analysis and contract negotiation
2	Database Failure	1	5	5	Proper data backup in external mediums
3	Denial of Service attack (DOS)	2	4	8	Installation of reliable security system and firewall
4	Ransomware	1	5	5	Proper data backup and strong system
5	Viruses and Malwares	1	3	3	Installation of strong anti-virus and anti-malware software.
6	Lack of Resources	2	3	6	operation in limited resources or extension of resources
7	Natural Disasters	3	4	12	Data backup, weather news follow up and insurance

6.2 Configuration Management

Configuration Management is a system engineering process designed to establish and maintain the consistency of the p erformance, functional and physical attributes of a product with its requirements, design and operational information throughout its lifetime. The given figures shows the configuration management process of this project.

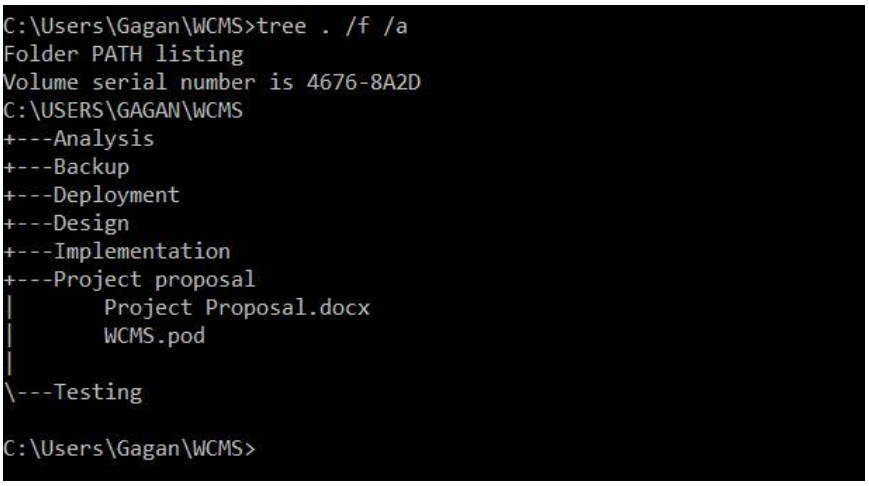


Figure 14: Tree Chart

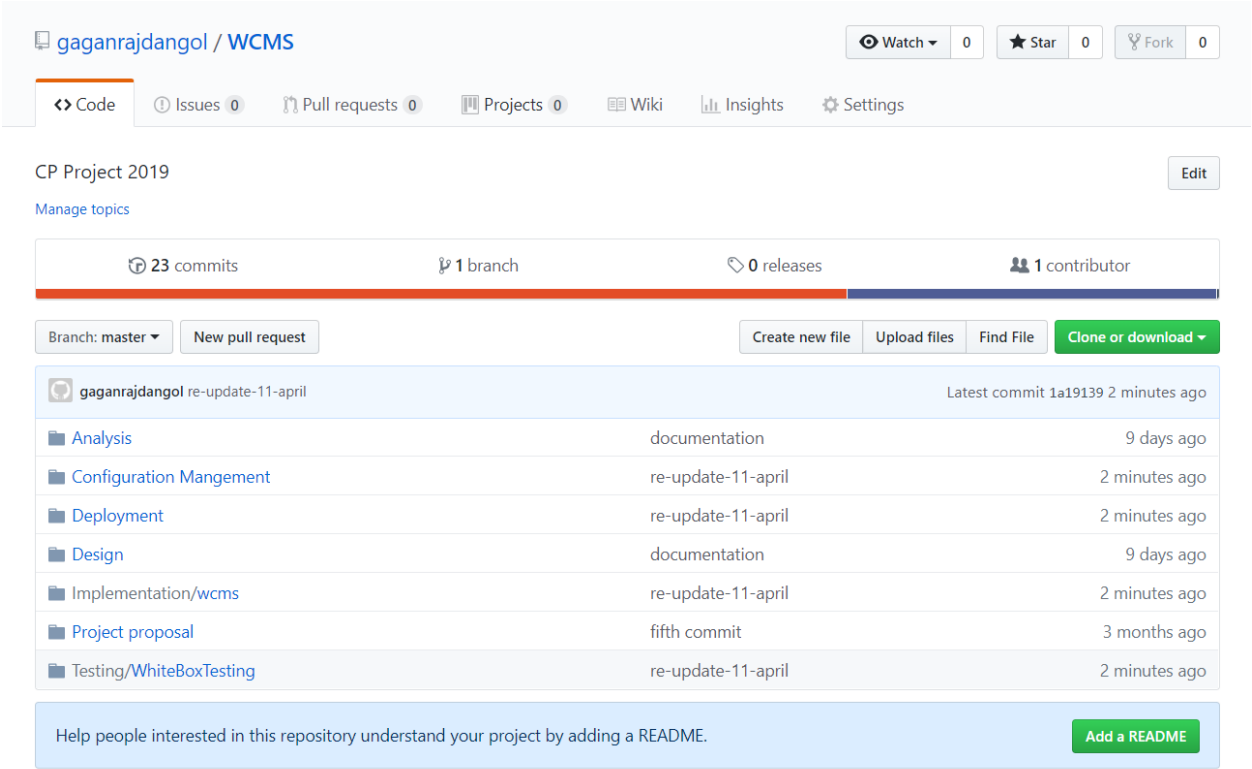


Figure 15: GitHub Repository

GitHub link: https://github.com/gaganrajdangol/WCMS

6.3 Time Scheduling

	Name	Duration	Start	Finish	Predecessors	Resource Names
1	**Project Management**	15 days	**12/21/18 8:00 AM**	**1/4/19 5:00 PM**		
2	Risk Management	5 days	12/21/18 8:00 AM	12/25/18 5:00 PM		
3	Work Breakdown Structure(V	4 days	12/26/18 8:00 AM	12/29/18 5:00 PM	2	
4	Configuration Management	4 days	12/30/18 8:00 AM	1/2/19 5:00 PM	3	
5	Proposal Submission	2 days	1/3/19 8:00 AM	1/4/19 5:00 PM	4	
6	**Analysis**	25 days	**1/5/19 8:00 AM**	**1/29/19 5:00 PM**	**1**	
7	Reuirement Analysis	5 days	1/5/19 8:00 AM	1/9/19 5:00 PM		
8	Use Case	6 days	1/10/19 8:00 AM	1/15/19 5:00 PM	7	
9	Architecture	8 days	1/16/19 8:00 AM	1/23/19 5:00 PM	8	
10	Analysis Specification	6 days	1/24/19 8:00 AM	1/29/19 5:00 PM	9	
11	**Design**	32 days	**1/30/19 8:00 AM**	**3/2/19 5:00 PM**	**6**	
12	Structural Model	7 days	1/30/19 8:00 AM	2/5/19 5:00 PM		
13	Behavioral Model	7 days	2/6/19 8:00 AM	2/12/19 5:00 PM	12	
14	UI DEsign	10 days	2/13/19 8:00 AM	2/22/19 5:00 PM	13	
15	Database Design	8 days	2/23/19 8:00 AM	3/2/19 5:00 PM	14	
16	**Implementation**	32 days	**3/3/19 8:00 AM**	**4/3/19 5:00 PM**	**11**	
17	Build Database	10 days	3/3/19 8:00 AM	3/12/19 5:00 PM		
18	Coding	22 days	3/13/19 8:00 AM	4/3/19 5:00 PM	17	
19	**Testing**	10 days	**4/4/19 8:00 AM**	**4/13/19 5:00 PM**	**16**	
20	Unit Testing	4 days	4/4/19 8:00 AM	4/7/19 5:00 PM		
21	Integration Testing	3 days	4/8/19 8:00 AM	4/10/19 5:00 PM	20	
22	Black Box Testing	1 day	4/11/19 8:00 AM	4/11/19 5:00 PM	21	
23	White Box Testing	2 days	4/12/19 8:00 AM	4/13/19 5:00 PM	22	
24	**Deployment**	10 days	**4/14/19 8:00 AM**	**4/23/19 5:00 PM**	**19**	
25	User Training	6 days	4/14/19 8:00 AM	4/19/19 5:00 PM		
26	Final Report	4 days	4/20/19 8:00 AM	4/23/19 5:00 PM	25	

Figure 16: Time Scheduling

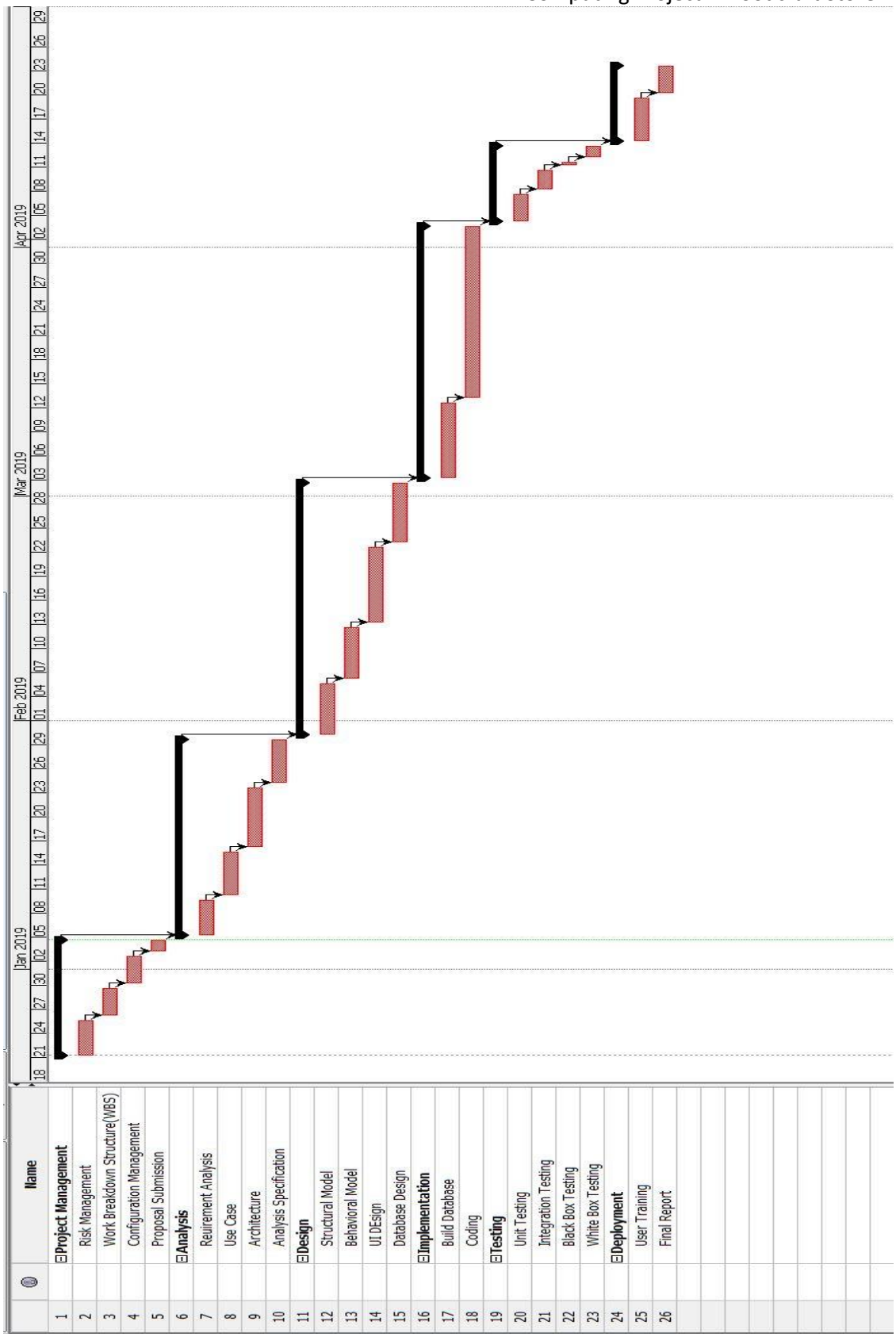


Figure 17: Gantt Chart

6.4 User Manual

User Manual is a readable document that encompasses guides and walkthroughs on how to conveniently and properly use an application. It includes processes of the functionalities of the application and carries any other important notes that directly or indirectly effect the application.

Basic Information:

The system dashboard displays all system information such as login, signup, feature d profiles, etc.

Authorized users and Non-Authorized:

Authorized users can use all the features and can search and post profiles and add t hem to them. But the unauthorized person can search for the user only.

View Profile:

Users can view other users ' profile by sending them friend requests and adding the m to favorites and sending them the message as well. By clicking the add button and adding to favorites, the user can send the friend request by clicking add to favorites.

The User Manual contains all essential information for the user to use the informatio n system to its full extent. This manual includes a description of the functions and ca pabilities of the system, contingencies and alternative operating modes, and step-by-step system access and use procedures.

Unregistered users may view the products, but they may not book or order the produ cts, while registered users may book or order the products. In order to get full service s on the system, users must first register their account.

Order Item:

By clicking the Confirm order buttons on the Bookings page, registered users can or der items.
Users can view their orders on the Orders page after the item is ordered.If the user w ants by clicking on the print icon, the bill can be generated and printed, and they can cancel the order by clicking on the order cancel button on the order page.

Edit Profile:

Users can change / update their profiles whenever they want by clicking the Edit Prof ile button on the drop-
down button inside the navbar (username). Users must complete the fields they want to change / update after moving to the Edit Profile page and then click the update bu tton.

Search Products:

Users can do their job more efficiently and effectively. The search service will help ful fill the desired user. By typing in the Products page on the search textbox, users can search for products by category, name, date, price etc.

6.5 Limitations

Even though the project is ought to solve several problems, it carries few limitations of itself. The Limitations of the project are given below:

- This project doesn't allow online payment services.
- Even though the application is publicly view and used internationally, goods cannot be shipped or delivered internationally.
- Most of the services are limited to domestic regions.

6.6 Future Work

The project has few scopes that have not been encompassed or better yet, they are to be completed once the project have been deployed and operated.

- Content Writing
- User-Admin Chat box
- Featured Item (display at the top of the gallery)
- Site Usage Graph
- Sales Report

Chapter 7: Conclusion

Conclusion

In this documentation I have discussed over the key concepts, themes and skills related to the application. Furthermore, the contents in this document will help to develop the fundamental knowledge, understanding, and analysis and synthesis skills that is needed to develop fit-for-purpose software in an organizational context by taking a practice-based approach.

In conclusion, considering all the factors, the project completed and it encompasses and accomplishes all the requirements. All necessary features are built as well.

Chapter 8: References

References

Anon., 2019. *Tutorial point.* [Online]
Available at: https://www.tutorialspoint.com/software_testing/

Anon., n.d. *stack overflow.* [Online]
Available at: https://stackoverflow.com/questions/16475979/what-is-the-difference-between-functional-and-non-functional-requirement

Anon., n.d. *Waterfall Model.* [Online]
Available at: http://toolsqa.com/software-testing/waterfall-model/

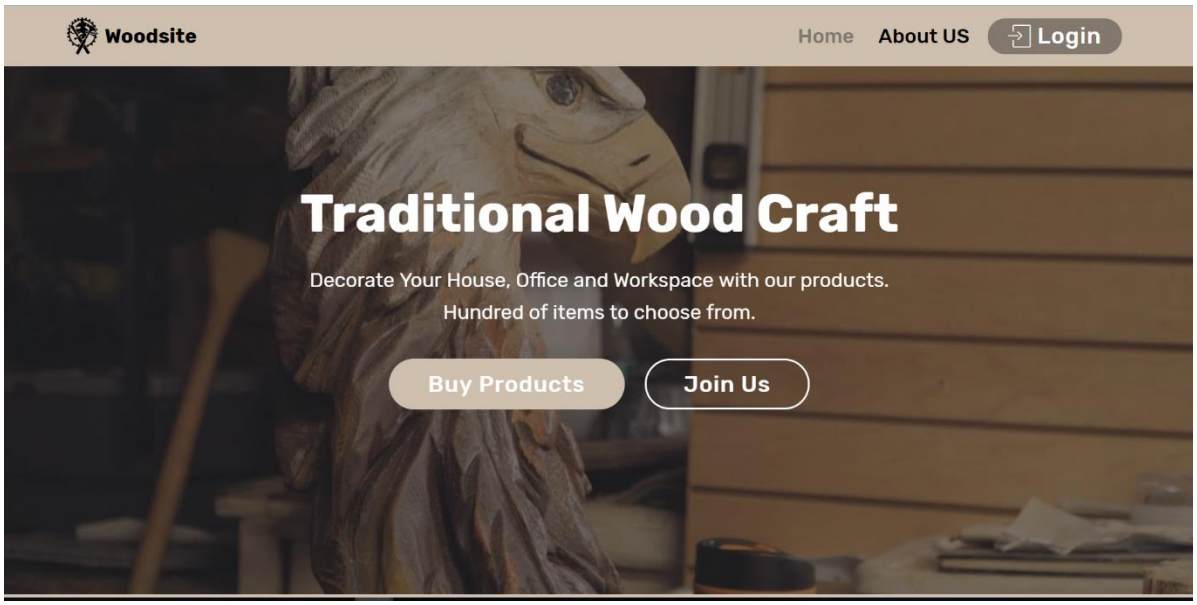
Chapter 9: Appendix

Appendix

The screenshots of the user interfaces are given below. The HTML and Laravel codes are given with respect to their interface. The codes of the controller follows the UI.

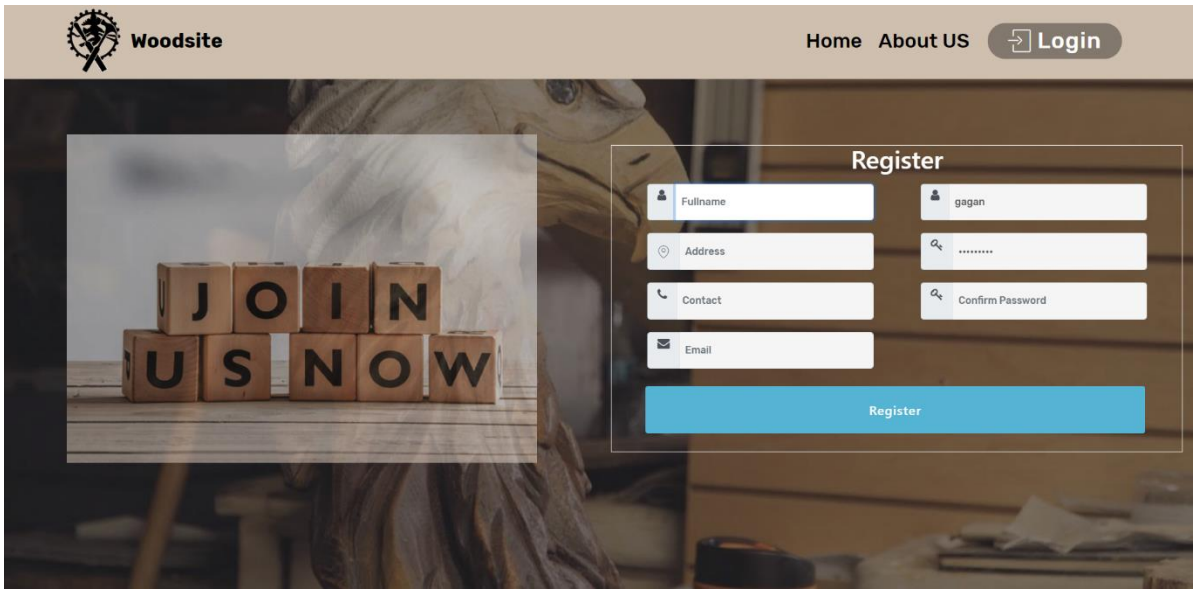
9.1 User Interface (UI)

9.1.1 Home Page



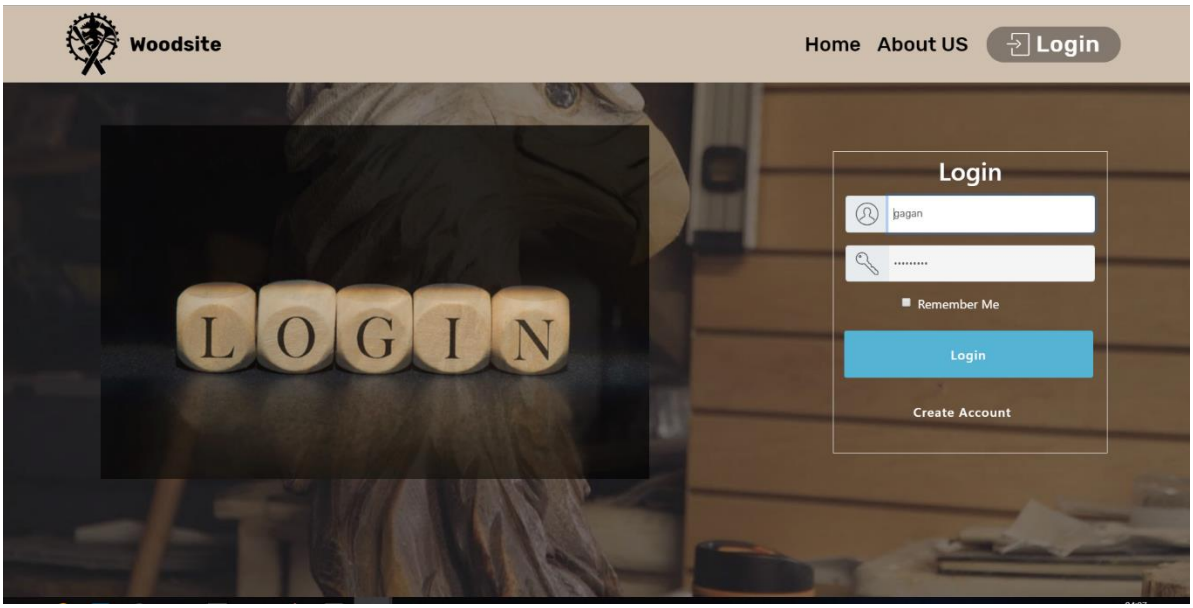

```
<nav class="navbar navbar-expand beta-menu navbar-dropdown align-items-center navbar-fixed-top navbar-toggleable-sm">
    <button class="navbar-toggler navbar-toggler-right" type="button" data-toggle="collapse" data-target="
    #navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle
    navigation">
        <div class="hamburger">
            <span></span>
            <span></span>
            <span></span>
            <span></span>
        </div>
    </button>
    <div class="menu-logo">
        <div class="navbar-brand">
            <span class="navbar-logo">
                <a href="{!! url('wcms') !!}">
                    <img src="assets/images/shiva2-122x145.png" alt="wcms" title="" style="height: 5.1rem;">
                </a>
            </span>
            <span class="navbar-caption-wrap"><a class="navbar-caption text-black display-5" href="index.html">
                    Woodsite</a></span>
        </div>
    </div>
    <div class="collapse navbar-collapse" id="navbarSupportedContent">
        <ul class="navbar-nav nav-dropdown" data-app-modern-menu="true">
          <li class="nav-item">
            <a class="nav-link link text-black display-5" style="color: #55b4d4;" href="{!! url('wcms') !!}"><span
            style="color:#82786e;">Home</span></a>
          </li>
          <li class="nav-item">
            <a class="nav-link link text-black display-5" href="{!! url('wcms/aboutus2') !!}">About US</a>
          </li>
        </ul>
```

9.1.2 Register




```
<section class="cid-rg6iDVUTMv mbr-fullscreen mbr-parallax-background" id="header2-d">


    <div class="mbr-overlay" style="opacity: 0.5; background-color: rgb(35, 35, 35);"></div>


    <div class="container mbr-align-left col-md-6" style="opacity: 0.9;">
        <div class="media-container-row"style="opacity: 0.9;">
            <img src="{{ URL::to('assets/images/joinus.jpg') }}" alt="" style="opacity: 0.7;">
        </div>

        </div>

    <div class="container align-right col-md-6">
            <div class="card">


                <div class="container align-center" style="border: 1px solid white">
                    <form method="POST" action="{{ route('register') }}">
                        @csrf

                        <div class="form-group h2  mbr-white">Register</div>


                            <div class="col-md-6" style="float:left;">

                            <div class="form-group col-md-12">
                                <div class="input-group">
                                <div class="input-group-prepend">
                                <span class="input-group-text fa fa-user"></span>
                            </div>
                                <input id="fullname" type="text" style="font-size:0.8rem;" class="form-control h4 {{
                                $errors->has('name') ? ' is-invalid' : '' }}" name="fullname" value="{{ old('name') }}"
                                placeholder="Fullname" required autofocus>

                                @if ($errors->has('name'))
                                    <span class="invalid-feedback" role="alert">
                                        <strong>{{ $errors->first('name') }}</strong>
                                    </span>
                                @endif
                            </div>
                        </div>

                        <div class="form-group col-md-12">
                                <div class="input-group">
                                <div class="input-group-prepend">
                            <span class="input-group-text mbri-pin"></span>
                        </div>


                                <input id="address" type="text" style="font-size:0.8rem;" class="form-control h4 {{
                                $errors->has('address') ? ' is-invalid' : '' }}" name="address" value="{{ old('address')
                                }}" placeholder="Address" required autofocus>

                                @if ($errors->has('address'))
                                    <span class="invalid-feedback" role="alert">
                                        <strong>{{ $errors->first('address') }}</strong>
                                    </span>
                                @endif
                            </div>
                        </div>

                    <div class="form-group col-md-12">
                            <div class="input-group">
                            <div class="input-group-prepend">
                        <i class="input-group-text fa fa-phone"></i>
```

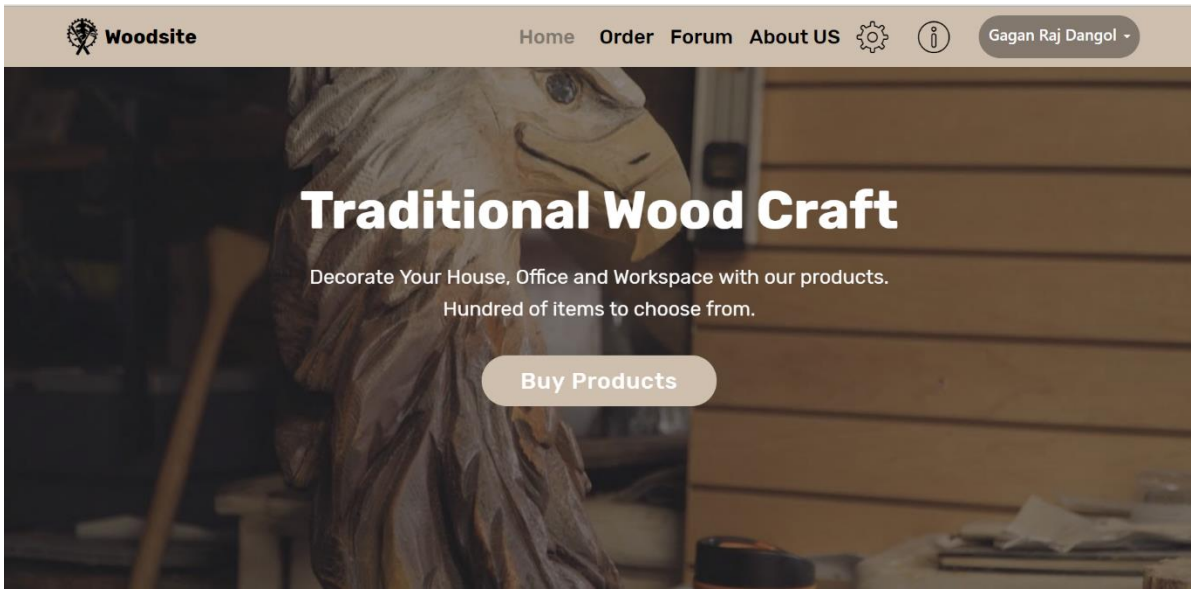
9.1.3 Login




```
                ;" placeholder="Username" required autofocus>

            @if ($errors->has('username'))
                <span class="invalid-feedback" role="alert">
                    <strong>{{ $errors->first('username') }}</strong>
                </span>
            @endif
        </div>
    </div>

    <div class="form-group col-md-12">
            <div class="input-group">
            <div class="input-group-prepend">
                <span class="input-group-text mbri-key display-4"></span>
            </div>


            <input id="password" type="password" class="form-control{{ $errors->has('password') ? '
            is-invalid' : '' }}" name="password" style="font-size:0.8rem;" placeholder="Password"
            required>

            @if ($errors->has('password'))
                <span class="invalid-feedback" role="alert">
                    <strong>{{ $errors->first('password') }}</strong>
                </span>
            @endif
        </div>
    </div>

    <div class="form-group row col-md-12 align-center">
            <div class="form-check col-md-12">
                <input class="form-check-input {{ old('remember') ? 'checked' : '' }}" type="checkbox"
                 name="remember" id="remember">
<se
                <label class="form-check-label mbr-white" for="remember">
                    {{   ('Remember Me') }}
```

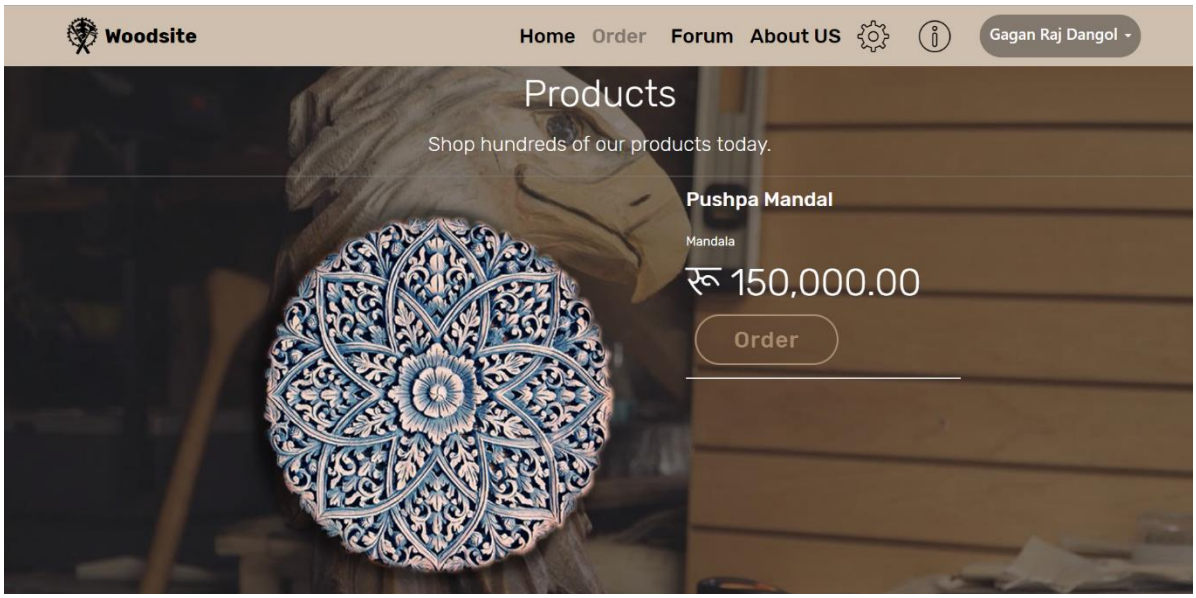
```
    <div class="mbr-overlay" style="opacity: 0.5; background-color: rgb(35, 35, 35);"></div>

    <div class="container mbr-align-left col-md-6" style="opacity: 0.9;">
        <div class="media-container-row"style="opacity: 0.9;">
            <img src="{{ URL::to('assets/images/login.jpg') }}" alt="" style="opacity: 0.9;">

        </div>
    </div>

    <div class="container align-right col-md-3">
            <div class="card">


                <div class="container align-center col-md-12" style="border: 1px solid white">
<!-- form -->

                    <form method="POST" action="{{ route('login') }}">
                        @csrf

                        <div class="form-group h2 mbr-white mt-2">Login</div>

                        <div class="row" style="float:left;">

                            <div class="form-group col-md-12">
                                <div class="input-group">
                                <div class="input-group-prepend">
                                    <span class="input-group-text mbri-user display-4"></span>
                                </div>
                                <input id="username" type="username" class="form-control{{ $errors->has('username') ? '
                                is-invalid' : '' }}" name="username" value="{{ old('username') }}" style="font-size:0.8rem
                                    ;" placeholder="Username" required autofocus>
```

9.1.4 User Home Page




```
<section class="cid-rg6iDVUTMv mbr-fullscreen mbr-parallax-background" id="header2-d">


    <div class="mbr-overlay" style="opacity: 0.5; background-color: rgb(35, 35, 35);"></div>

    <div class="container align-center">
        <div class="row justify-content-md-center">
            <div class="mbr-white col-md-10">
                <h1 class="mbr-section-title mbr-bold pb-3 mbr-fonts-style display-1">
                    Traditional Wood Craft</h1>

                <p class="mbr-text pb-3 mbr-fonts-style display-5">
                    Decorate Your House, Office and Workspace with our products.<br>Hundred of items to choose from.</p>
                <div class="mbr-section-btn">
                  <a class="btn btn-md btn-secondary display-4" href="{!! url('/order2') !!}" title="Order Products">Buy
                  Products</a>
                    <!-- <a title="Create an Account" class="btn btn-md btn-white-outline display-4" href="{{ url('wcms/
                    register') }}">Join Us</a> -->
                </div>
            </div>
        </div>
    </div>

</section>




    <div class="countdown1 cid-rp3V6ZK5S1" id="countdown1-17" style="background-color: transparent;">
      <div style="box-shadow: 5px 10px 18px black;" >

    <div class="countdown1 cid-rp3V6ZK5S1" id="countdown1-17" style="background-color: transparent;">
      <div style="box-shadow: 5px 10px 18px black;" >

    <div class="container ">
        <h2 class="mbr-section-title pb-3 align-center mbr-fonts-style display-2">
            COUNTDOWN</h2>
        <h3 class="mbr-section-subtitle align-center mbr-fonts-style display-5">
            Get Ready For A New Product Launch And A Major Update</h3>

    </div>
    <div class="container countdown-cont align-center">
        <div class="daysCountdown" title="Days"></div>
        <div class="hoursCountdown" title="Hours"></div>
        <div class="minutesCountdown" title="Minutes"></div>
        <div class="secondsCountdown" title="Seconds"></div>
        <div class="countdown pt-5 mt-2" data-due-date="2020/04/30">
        </div>
    </div>
  </div>
  </div>


<section class="features8 cid-rp3UMjXgEn mbr-parallax-background" id="features8-15" style="background-image: url('{{
    url::to('assets/images/mbr-1920x1280.jpg') }}');">



    <div class="mbr-overlay" style="opacity: 0.6; background-color: rgb(35, 35, 35);">
    </div>

    <div class="container">
        <div class="media-container-row">
```

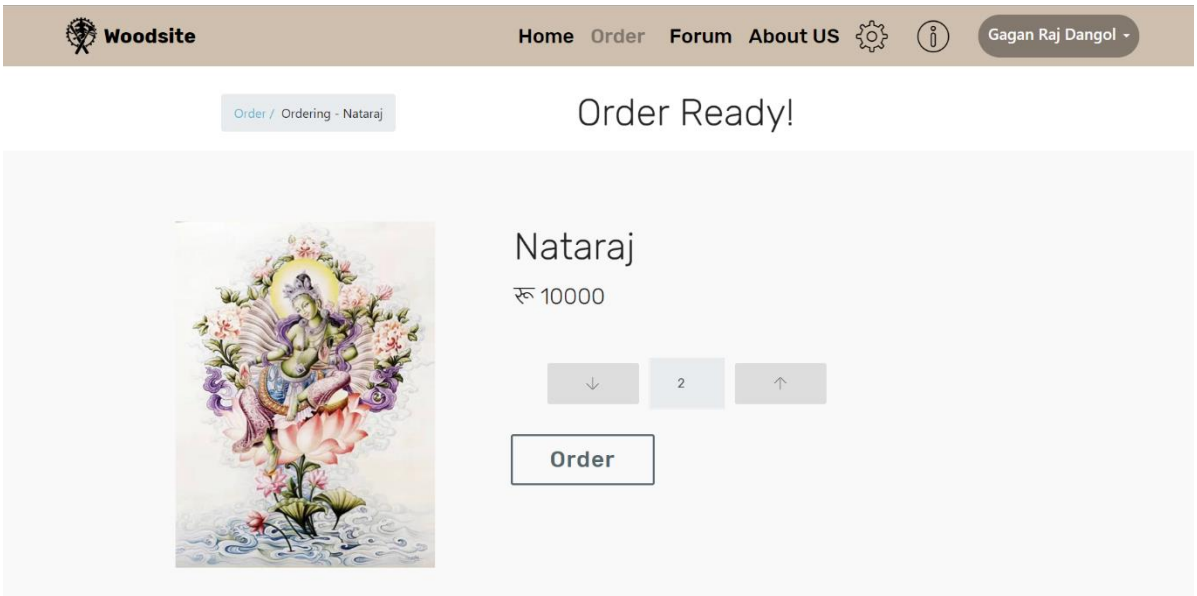
9.1.5 Order




```
<section class="cid-rg6iDVUTMv mbr-parallax-background" id="header2-d" style="height: 10%;">
    <div class="mbr-overlay" style="opacity: 0.5; background-color: rgb(35, 35, 35);"></div>
    <div class="container mbr-white">
        <div class="media-container-row">
            <div class="title col-12 col-md-8 mt-5">
                <h2 class="align-center pb-3 mbr-fonts-style display-2">
                    Products</h2>
                <h3 class="mbr-section-subtitle align-center mbr-light mbr-fonts-style display-5">Shop hundreds of our
                products today.</h3>

            </div>
        </div>
    </div>

    <hr style="background-color: white;">
    <!--Container-->
    <div class="container mb-5 mbr-white">
        <div class="row">
            <div class="card-wrapper media-container-row media-container-row">

                        <div class="col-12 col-md-6">
                            <div class="wrapper">
                                <!--Image-->
                                <div class="img-fluid" style="width: 100%;">
                                    <img src="{{ URL::to('itemImages/mandala.png') }}" alt="" title="" style="width:
                                      90%;">
                                </div>
                            </div>
                        </div>
                        <div class="col-12 col-md-7 col-lg-4">
                            <div class="wrapper col-left">
                                <!--Title-->
                                <h4 class="card-title mbr-fonts-style display-5">Pushpa Mandal</h4>
                                <!--Subtitle-->
                                <p class="mbr-text mbr-fonts-style pt-3 display-7">
                                </p>
                                <!--Btn-->
                                <div class="mbr-section-btn col-md-12"><a href="" class="btn btn-secondary-outline
                                m-0 display-4">
                                        Order</a></div>
                                        </div>
                        <hr style="background-color: white;">

                        </div>


            <!--Card-2-->

            <!--Card-3-->

        </div>
    </div>
</div>
</section>
<br>
    <div style="position: sticky;" class="align-right">
        <form>
            <input type="text" class="col-md-10 form-control align-center" name="search" style="border-radius: 40px; float
              : left;" placeholder="Search Products By Name Or Type" id="searchproducts">
            <button type="submit" class="btn btn-secondary-outline pt-2" value="" style="border-radius: 40px; height: 55px
              ;width: 220px;" id="btnsearch"><h2><span class="mbri-search"></span></h2></button>

        </form>
    </div>

<script src="{{ asset('js/app.js') }}"></script>
        <script type="text/javascript">
            $(document).ready(function(){
                                    // $('#selectType').change(function(e){
                                    //    e.preventDefault();
```

```
<!--Container-->
<div class="container" id="searchdata">
    <div class="row justify-content-center">
        <!--Titles-->
        <div class="title pb-5 col-12">

            <h3 class="mbr-section-subtitle mbr-light mbr-fonts-style display-5">
                Products</h3>
        </div>
        <!--Card-1-->
        @if($item->count())
    @foreach($item as $items)

     <form action="{!! url('/orderprocess', $items->itemid) !!}" class="col-md-4">
        <div class="card col-12 col-md-6 col-lg-3 mb-5">
            <div class="card-wrapper">
                <div class="card-img">
                    <img src="/{{ $items->item_image }}" alt="" title="" style="width: 210px;height: 270px;">
                </div>
                <div class="card-box" style="width: 210px;">
                    <h4 class="card-title mbr-fonts-style">{{ $items->item_name }}</h4>

                    <!--Btn-->
                    <div class="mbr-section-btn align-left mb-1">
                        <input type="submit" style="font-size: 20px; height: 75px; width: 150px;" class="btn
                        btn-secondary-outline display-7 p-0" value="रू {{ $items->price }}.00">
                    </div>
                </div>
            </div>
        </div>
    </form>
        @endforeach
    @else

        <h2 colspan="4"> No record found</h2>12
```

9.1.6 Order Processing

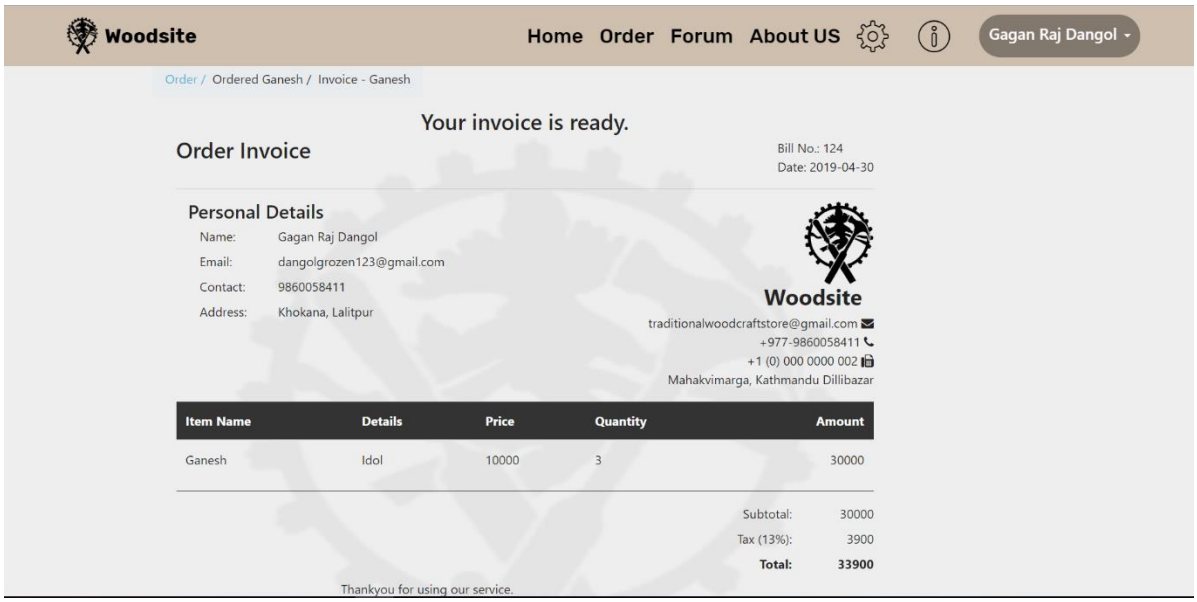


```
<section class="mbr-section content4 cid-rg9atgb20I" id="content4-21">



    <div class="container">
        <div class="media-container-row">
          @foreach($orders as $order)
          <div class="">
            <ul class="breadcrumb">
  <li><a href="{!! url('/order2') !!}">Order /  </a></li>
  <li> Ordering - {{$order->item_name}}</li>
</ul>
          </div>
          @endforeach
            <div class="title col-12 col-md-8">
                <h2 class="align-center pb-3 mbr-fonts-style display-2">
                    Order Ready!</h2>


            </div>
        </div>
    </div>
</section>

<section class="services2 cid-rg9kyIxrUr" id="services2-11">
    <!---->

    <!---->
    <!--Overlay-->

    <!--Container-->
    @foreach($orders as $order)
    <form method="POST" action="{!!url('/order',$order->itemid)!!}">
    @csrf
    <div class="container">
        <div class="col-md-12">
```

```
                    <input type="number" id="quantity" name="quantity" class="form-control
                    input-number align-center" value="1" min="1" max="100" requried readonly>

                    <span class="input-group-btn">

                        <button type="button" class="quantity-right-plus btn btn-defeat btn-number"
                        data-type="plus" data-field="">

                            <span class="mbri-up"></span>

                        </button>


                    </span>

                </div>

            </div>

    </div>

</div>
```

```
                </div></div></div><p></p>
            </div>
            <!--Btn-->
            <div class="mbr-section-btn pt-3 align-left">
              <button type="submit" class="btn btn-warning-outline display-4" onclick="if (!confirm('You are
                about to order this item. Are you sure?')) { return false }">
                    Order</button></div>
        </div>
    </div>
</div>
</form>
@endforeach
```

9.1.7 Bill

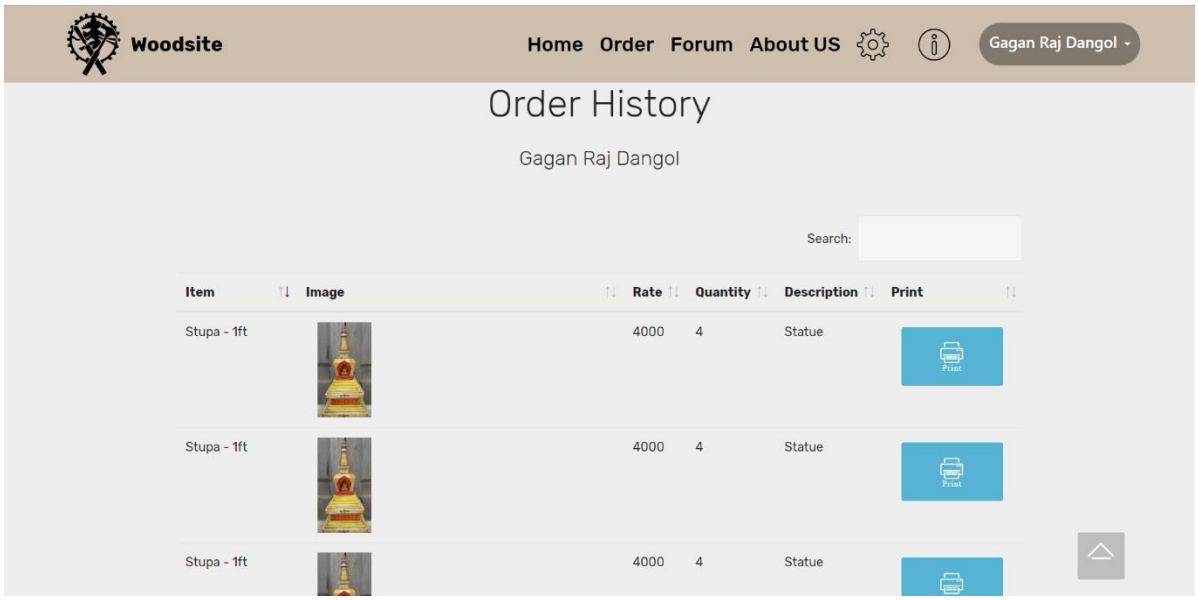



```
<section>
  <br><br>
  <br><br>
  <br>
        <div class="container">
          <div class="row">
            <div id="breadcrumb">
            <ul class="breadcrumb">
  <li><a href="{!! url('/order2') !!}">Order /  </a></li>
  <li>Ordered {{$getBill->item_name}} /  </li>
  <li>Invoice - {{$getBill->item_name}}</li>
</ul>
          </div>
            <div class="col-lg-10 col-centered public-view">
              <div class="container">
                <div class="" style="background-color:#333"></div>

                    <div class="align-center col-md-12" id="info1"><h3>Your invoice is ready.</h3></div>

                <div class="invoice-detail-body" id="printthisdiv" style="background-image: url('{{ url::to('assets/im
                  ages/shiva-translucent.png') }}');background-repeat: no-repeat;background-size: cover;;z-index: -1;"
                >
                  <div class="invoice-detail-title content-block">
                    <div class="d-flex justify-content-between">
                      <div class="invoice-title">

                        <h3>Order Invoice</h3>
                      </div>
                      <div class="invoice-logo"><div class="photo-drop">
                        <div class="photo-drop-preview">
                        <div>Bill No.: {{$getBill->orderid}}</div>
                        <div>Date: <?php echo date('Y-m-d')?></div>
                        </div>
                      </div>
                    </div>
      </div>
      <div class="align-left aside-content">
          <h2 class="mbr-title pt-2 mbr-fonts-style display-2">
              {{$order->item_name}}</h2>
          <div class="mbr-section-text">
              <p class="mbr-text text1 pt-2 mbr-light mbr-fonts-style display-4">
                  रू {{$order->price}}
              </p>
              <p class="mbr-text text2 pt-4 mbr-light mbr-fonts-style display-7"></p><div class="container">
              <div class="container">
                <div class="row">
                <div class="container">

<div class="row">

                <div class="col-lg-8">

                    <div class="input-group">

                <span class="input-group-btn">

                    <button type="button" class="quantity-left-minus btn btn-default btn-number"
                    data-type="minus" data-field="">

                      <span class="mbri-down"></span>

                    </button>

                </span>

                <input type="number" id="quantity" name="quantity" class="form-control
                input-number align-center" value="1" min="1" max="100" requried readonly>

                <span class="input-group-btn">
```

```
      </div>
    </div>
  </div>
  <hr class="divider lite margin">
  <div class="col-md-6" style="float: left;">
    <div class="row">
      <div class="col-md invoice-address invoice-address-company">
        <h4>Personal Details</h4>
        <div class="col-md-12 input-details1">
          <div class=""><label style="width:100px;">Name: </label><span>{{Auth::user()->fullname}}</span
          ></div>
          <div class=""><label style="width:100px;">Email: </label><span>{{Auth::user()->email}}</span>
          </div>
          <div class=""><label style="width:100px;">Contact: </label><span>{{Auth::user()->contact}}</
          span></div>
          <!-- <div class=""><label style="width:100px;">Zip Code: </label><span>44600</span></div> -->
          <div class=""><label style="width:100px;">Address: </label><span>{{Auth::user()->address}}</
          span></div>
        </div>
      </div>
    </div>
  </div>
  <div class="align-right mb-3">
    <img style="width: 10%;height:10%;" src="{{ url('assets/images/shiva-162x193.png') }}" alt="
    company-logo" title=""/>
    <div class="col-md-12 align-right">
    <h3><b>Woodsite</b></h3>
  </div>
  <div>traditionalwoodcraftstore@gmail.com <span class="fa fa-envelope"></span></div>
  <div>+977-9860058411 <i class="fa fa-phone"></i></div>
  <div>+1 (0) 000 0000 002 <i class="fa fa-fax"></i></div>
  <div>Mahakvimarga, Kathmandu Dillibazar <i class="fas fa-map-marker-alt"></i></div>
  </div>
        <div class="invoice-item-list content-block">
          <table class="table invoice-table invoice-table-view">
            <thead class="thead">
```

```
<div>Mahakvimarga, Kathmandu Dillibazar <i class="fas fa-map-marker-alt"></i></div>
</div>
      <div class="invoice-item-list content-block">
        <table class="table invoice-table invoice-table-view">
          <thead class="thead">
            <tr>
              <th class="invoice-detail-name" style="background-color:#333!important;color:#fff">
              Item Name</th>
              <th class="invoice-detail-summary align-left" style="background-color:#333!important;
                color:#fff">
          Details
        </th>
              <th class="invoice-detail-rate" style="background-color:#333!important;color:#fff">
              Price
            </th>
            <th class="invoice-detail-quantity" style="background-color:#333!important;color:#fff">
            Quantity
          </th>
          <th class="invoice-detail-total align-right" style="background-color:#333!important;color:#
            fff">
          Amount
        </th>

      </tr>
    </thead>
    <tbody class="invoice-items">
      <tr class="item-row item-row-1">
        <td class="item-row-summary" data-label="Item #1"><span class="item-row-name"></span>
          <span class="item-row-description">
            {{$getBill->item_name}}
        </span>
      </td>
      <td class="item-row-summary" data-label="Item #1"><span class="item-row-name"></span>
          <span class="item-row-description">
            {{$getBill->item_description}}
        </span>
```

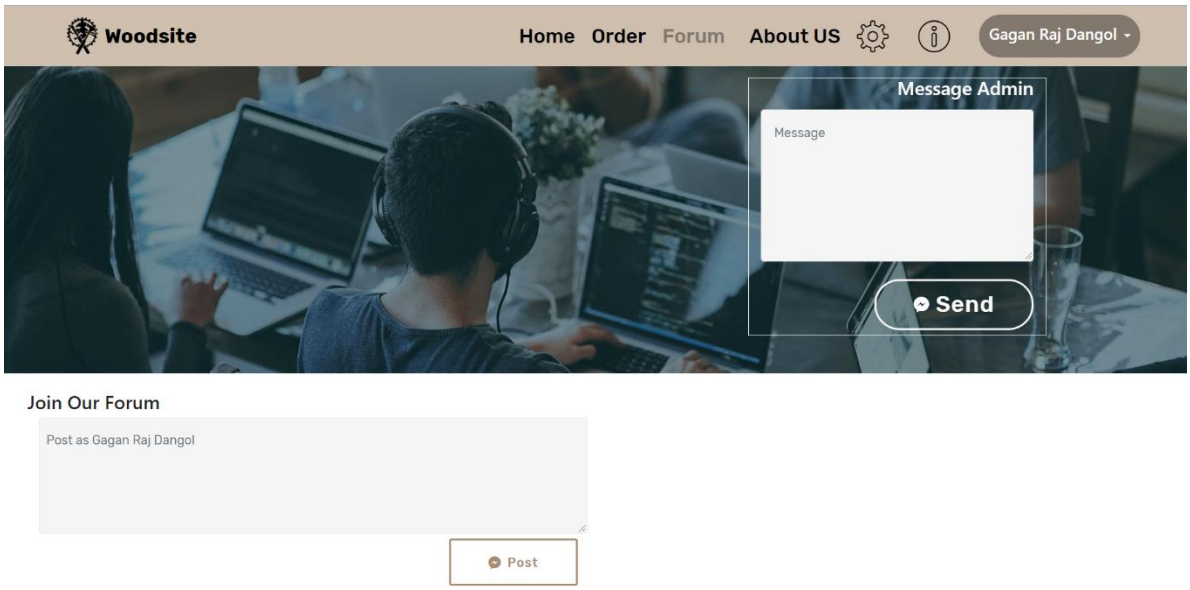
9.1.8 Orderlist




```
<section class="section-table cid-rmEhkIRgdt mt-5" id="table1-4">


<div class="container container-table mt-5">
    <h2 class="mbr-section-title mbr-fonts-style align-center pb-3 display-2 mt-5">
        Order History
    </h2>
    <h3 class="mbr-section-subtitle mbr-fonts-style align-center pb-5 mbr-light display-5">
    {{Auth::user()->fullname}}</h3>
    <div class="table-wrapper">
      <div class="container">
        <div class="row search">
          <div class="col-md-6"></div>
          <div class="col-md-6">
              <div class="dataTables_filter">
                <label class="searchInfo mbr-fonts-style display-7">Search:</label>
                <input class="form-control input-sm" disabled="No">
              </div>
          </div>
        </div>
      </div>

      <div class="container scroll">
        <table class="table isSearch" cellspacing="0">
          <thead>
            <tr class="table-heads ">
              <th class="head-item mbr-fonts-style display-7">Item</th>
              <th class="head-item mbr-fonts-style display-7">Image</th>
              <th class="head-item mbr-fonts-style display-7">Rate</th>
              <th class="head-item mbr-fonts-style display-7">Quantity</th>
              <th class="head-item mbr-fonts-style display-7">Description</th>
              <th class="head-item mbr-fonts-style display-7">Print</th>
            </tr>
          </thead>
```

```
            <th class="head-item mbr-fonts-style display-7">Item</th>
            <th class="head-item mbr-fonts-style display-7">Image</th>
            <th class="head-item mbr-fonts-style display-7">Rate</th>
            <th class="head-item mbr-fonts-style display-7">Quantity</th>
            <th class="head-item mbr-fonts-style display-7">Description</th>
            <th class="head-item mbr-fonts-style display-7">Print</th>
          </tr>
        </thead>

        <tbody>
        @if($order->count())
@foreach($order as $orders)
        <tr>

          <td class="body-item mbr-fonts-style display-7">{{ $orders->item_name }}</td>
          <td class="body-item mbr-fonts-style display-7"><img class="col-md-3" src="/{{ $orders->item_image }}"
          alt=""></td>
          <td class="body-item mbr-fonts-style display-7">{{ $orders->price }}</td>
          <td class="body-item mbr-fonts-style display-7">{{ $orders->quantity }}</td>
          <td class="body-item mbr-fonts-style display-7">{{ $orders->item_description }}</td>
          <td class="body-item mbr-fonts-style display-7"><button class="btn btn-primary" title="Print the bill">
              <span class="mbri-print display-4"><div style="font-size: 10px;">Print</div></span>
            </button></td>

        </tr>
        @endforeach
@else

        <h2 colspan="4">You haven't ordered any products yet. <a href="{{ ('/order2') }}">Start Ordering</a></h2>

@endif
      </tbody>
      </table>
</div>
<div class="container table-info-container">
  <div class="row info">
```

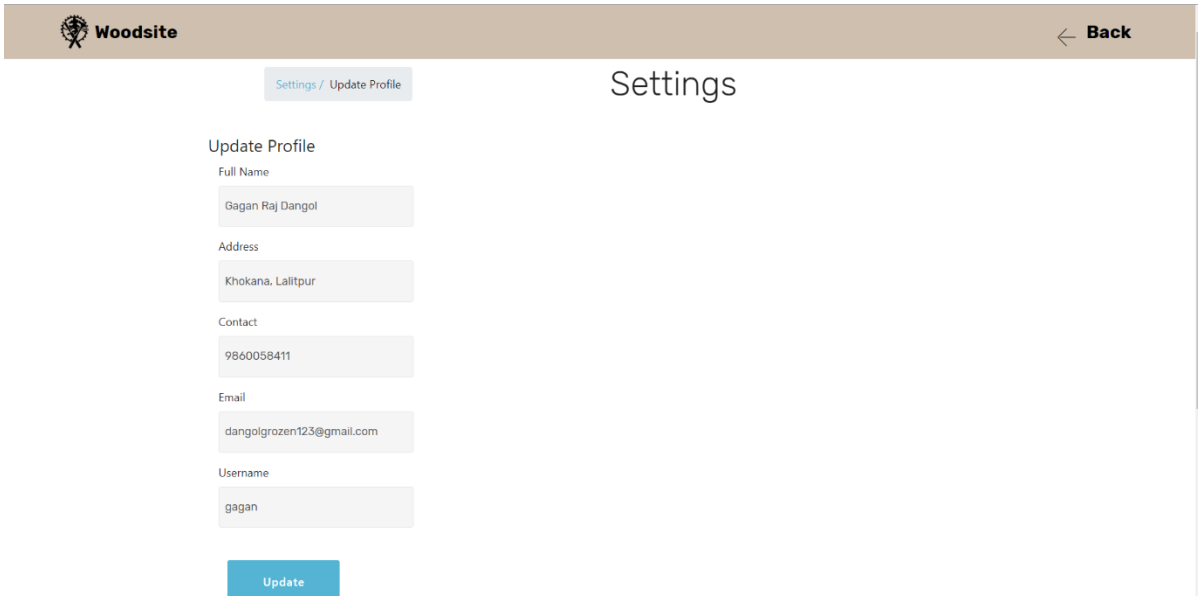
9.1.9 Forum




```
<section class="header15 cid-rjUYoC1zhK mbr-parallax-background mt-5" id="header15-2" style="background-image: url(
  {{ url::to('assets/images/background4.jpg') }}'); ">


    <div class="mbr-overlay" style="opacity: 0.4; background-color: rgb(7, 59, 76);"></div>

    <div class="container align-right">
<div class="row">
    <div class="mbr-white col-lg-8 col-md-7 mt-5 mb-5 content-container">

    </div>

    <div class="col-lg-4 col-md-5 mt-5 mb-5" style="border-style:solid; border-color: white; border-width: 1px;">
    <div class="form-container">
        <div class="media-container-column" data-form-type="formoid">
            <div data-form-alert="" hidden="" class="align-center">Message Sent!!</div>
            <form class="mbr-form"  method="post"><input type="hidden" name="email" data-form-email="true" value="">
                <div data-for="name">
                    <div class="form-group align-right">
                        <h4 class="mbr-white">Message Admin</h4>
                    </div>
                </div>


                <div class="form-group" data-for="message">
                    <textarea type="text" class="form-control px-3" name="message" rows="7" placeholder="Message"
                    data-form-field="Message" id="message-header15-2"></textarea>
                </div>
                <span class="input-group-btn"><button href="" type="submit" class="btn btn-form btn-white-outline
                display-4"><span class="fab fa-facebook-messenger mbr-iconfont mbr-iconfont-btn" style="font-size: 2
                  px;"></span>Send</button></span>
            </form>
        </div>
```

```
        @if($message = Session::get('success'))
  <div class="alert alert-info">
   <p><h6>{{$message}}</h6></p>
  </div>
  @endif
    </div>


<div class="container col-md-12">
  <div class="mt-4 mb-4">
  </div>
  <div class="container col-md-12">
    <!-- <div class="col-md-3">
      <img src="{{ url::to('assets/images/noimage-240x149.png') }}" class="rounded float-left img-fluid
      img-thumbnail" alt="">
    </div> -->
    <h4>Join Our Forum</h4>
    <form class="form-horizontal" action="{!! url('/forum') !!}" method="post">
       @csrf
                        {{method_field('put')}}
                        <fieldset>
    <div class="form-group align-right col-md-6" data-for="message">
      <textarea type="text" class="form-control" name="post" rows="5" required placeholder="Post as
      {{Auth::user()->fullname}}" value="" id="message-header15-2"></textarea>
                    <input type="text" name="userid" value="{{Auth::user()->userid}}" class="form-control px-3" style=
                    "display: none;">
                    <input type="text" name="usernme" value="{{Auth::user()->fullname}}" class="form-control px-3"
                    style="display: none;">
    <button href="" type="submit"  name="btnpost" class="btn btn-secondary-outline display-7"><span class="fab
    fa-facebook-messenger mbr-iconfont mbr-iconfont-btn" style="font-size: 18px;"></span>Post</button>
    </div>
    </fieldset>
  </form>
</div>
  </div>
```

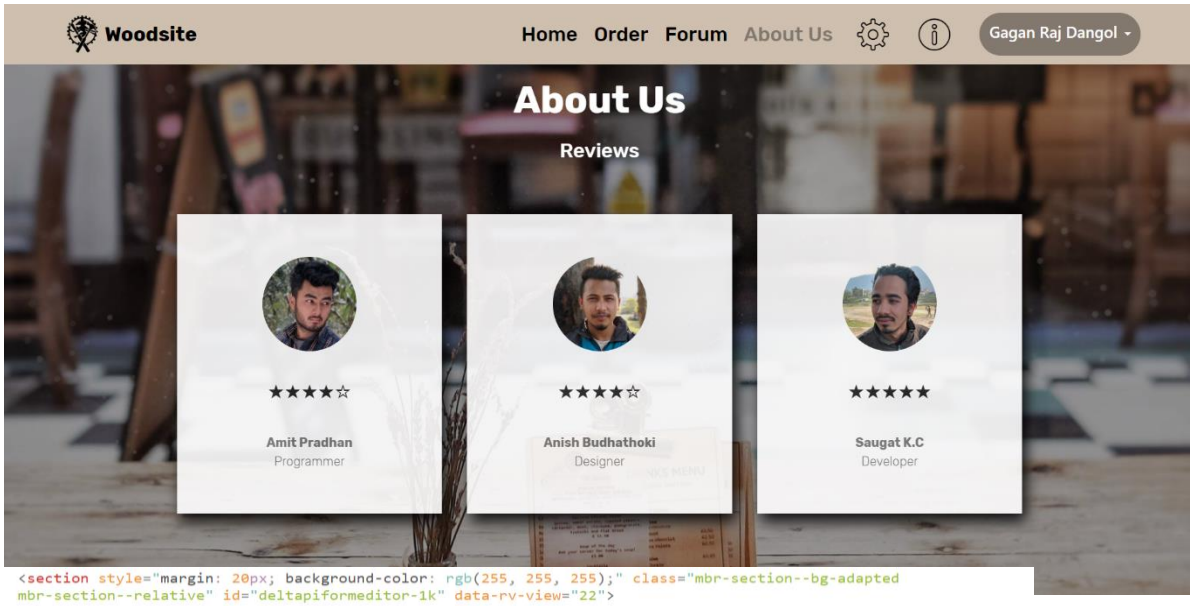
9.1.10 Update Profile



```
<section class="mbr-section content4 cid-rg8zY9aF7t" id="content4-1j">


    <div class="container">
        <div class="media-container-row">
          <div class="">
            <ul class="breadcrumb">
  <li><a href="{!! url('/order2') !!}">Settings /  </a></li>
  <li> Update Profile</li>
</ul>
          </div>
          <div class="title col-12 col-md-8">
              <h2 class="align-center pb-3 mbr-fonts-style display-2">
                  Settings</h2>


          </div>
        </div>
    </div>
</section>
 <div>
        @if($message = Session::get('success'))
  <div class="alert alert-info">
   <p><h6>{{$message}}</h6></p>
  </div>
  @endif
    </div>
```

```
<section style="margin: 20px; background-color: rgb(255, 255, 255);" class="mbr-section--bg-adapted
mbr-section--relative" id="deltapiformeditor-1k" data-rv-view="22">

    <div class="dpform">
```

9.1.11 About Us



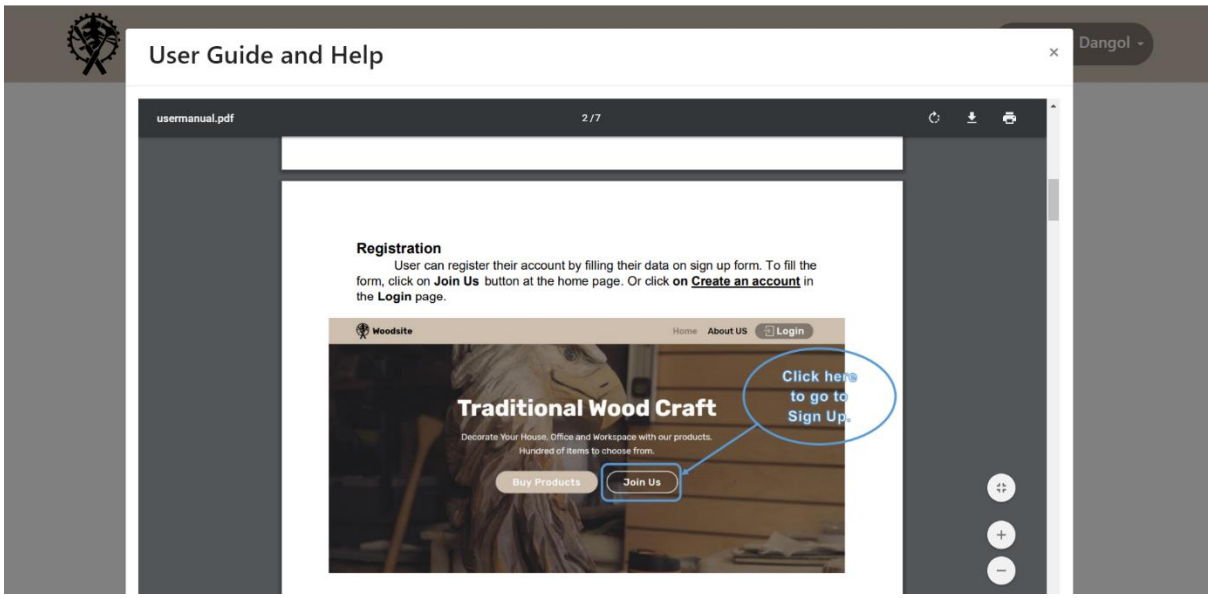

```
<section style="margin: 20px; background-color: rgb(255, 255, 255);" class="mbr-section--bg-adapted
mbr-section--relative" id="deltapiformeditor-1k" data-rv-view="22">

    <div class="dpform">
        <div class="container">
            <div class="row">
                <div class="col-md-10 col-md-offset-1">


<style>.dpform{text-align:left;}</style>

<div style="background-color: rgb(255, 255, 255);">
<div style="background-color: rgb(255, 255, 255);">


  <form method="post" class="form-horizontal" action="{!!url('/updateprofile', Auth::user()->id)!!}" enctype="
  multipart/form-data">

        @csrf
        {!!method_field('put')!!}

<fieldset>

<!-- Form Name -->
<legend>Update Profile</legend>

<!-- Text input-->
<div class="form-group">
  <label class="col-md-4 control-label" for="fullname">Full Name</label>
  <div class="col-md-4">
  <input id="fullname" name="fullname" type="text" placeholder="Full Name" class="form-control input-md" value="
  {!!(Auth::user()->fullname)!!}">

  </div>
</div>
```

```
<section class="testimonials1 cid-rg8ER7VnFx mbr-parallax-background" id="testimonials1-1q" style="background-image:
    url('{{ url::to('assets/images/izzie-r-418251.jpg') }}'); ">

    <div class="mbr-overlay" style="opacity: 0.4; background-color: rgb(5, 5, 5);"></div>


    <div class="container">
        <div class="media-container-row">
            <div class="title col-12 align-center">
                <h2 class="pb-3 mbr-fonts-style display-2 mbr-white">
                    <b>About Us</b></h2>
                <h3 class="mbr-section-subtitle mbr-white pb-3 mbr-fonts-style display-5" style="color: white;">
                Reviews</h3>
            </div>
        </div>
    </div>

    <div class="container pt-3 mt-2">
        <div class="media-container-row">
            <div class="mbr-testimonial p-3 align-center col-12 col-md-6 col-lg-4" style="opacity: 0.9;">
                <div class="panel-item p-3" style="box-shadow: 5px 10px 18px black;">
                    <div class="card-block">
                        <div class="testimonial-photo">
                            <img src="{{ URL::to('assets/images/programmer.jpg') }}" alt="" title="">
                        </div>
                        <p class="mbr-text mbr-fonts-style display-5">★★★★☆</p>
                    </div>
                    <div class="card-footer">
                        <div class="mbr-author-name mbr-bold mbr-fonts-style display-7">
                            Amit Pradhan</div>
                        <small class="mbr-author-desc mbr-italic mbr-light mbr-fonts-style display-7">
                            Programmer</small>
                    </div>
                </div>
            </div>


    <div class="mbr-testimonial p-3 align-center col-12 col-md-6 col-lg-4" style="opacity: 0.9;">
        <div class="panel-item p-3" style="box-shadow: 5px 10px 18px black;">
            <div class="card-block">
                <div class="testimonial-photo">
                    <img src="{{ URL::to('assets/images/designer.jpg') }}" alt="" title="">
                </div>
                <p class="mbr-text mbr-fonts-style display-5">★★★★☆<br></p>
            </div>
            <div class="card-footer">
                <div class="mbr-author-name mbr-bold mbr-fonts-style display-7">
                    Anish Budhathoki</div>
                <small class="mbr-author-desc mbr-italic mbr-light mbr-fonts-style display-7">Designer</small>
            </div>
        </div>
    </div>

    <div class="mbr-testimonial p-3 align-center col-12 col-md-6 col-lg-4" style="opacity: 0.8;">
        <div class="panel-item p-3" style="box-shadow: 5px 10px 18px black;">
            <div class="card-block">
                <div class="testimonial-photo">
                    <img src="{{ URL::to('assets/images/developer.jpg') }}" alt="" title="">
                </div>
                <p class="mbr-text mbr-fonts-style display-5">★★★★★<br></p>
            </div>
            <div class="card-footer">
                <div class="mbr-author-name mbr-bold mbr-fonts-style display-7">
                    Saugat K.C</div>
                <small class="mbr-author-desc mbr-italic mbr-light mbr-fonts-style display-7">
                    Developer</small>
            </div>
```

9.1.12 Inventory


```
<section class="mbr-section content4 cid-rg8kFJXlk1" id="content4-1a">
 <div>
       @if($message = Session::get('success'))
  <div class="alert alert-info">
   <p><h6>{{$message}}</h6></p>
  </div>
 @endif
    </div>


    <div class="container">
        <div class="media-container-row">
            <div class="title col-12 col-md-8">
                <h2 class="align-center pb-3 mbr-fonts-style display-2"><strong>
                    I</strong>nventory <strong>M</strong>anagement</h2>


            </div>
        </div>
    </div>
</section>

        <div class="col-md-12 col-md-offset-1"><hr><hr></div>


<section style="margin: 20px; background-color: rgb(255, 255, 255);" class="mbr-section-bg-adapted
mbr-section-relative" id="deltapiformeditor-1e" data-rv-view="13">
    <!-- Add Form -->

<div class="col-md-6">

                    <form class="form-horizontal" action="{!! url('/inventory') !!}" method="post" enctype="multipart/
                    form-data">
                        @csrf
```

9.1.13 Help




```
                                px;">
                                <a class="dropdown-item mbr-white" href="{{ route('logout') }}" onclick="event.
                                    preventDefault();
                                                  document.getElementById('logout-form').submit();"><span class='
                                                  mbri-logout display-5"></span><h5> Logout</h5></a>
                                                  <form id="logout-form" action="{{ route('logout') }}" method="
                                                  POST" style="display: none;">
                                    @csrf
                                </form>
                                <a href="{!! url('/orderlist') !!}" class="mbr-white"><span class="
                                mbri-shopping-cart display-5"></span><h5> Orderlist</h5></a>
                            </div>
                            </li>


                    </ul>


                </div>
            </nav>
        </section>

        <div class="modal fade col-md-12" id="modalHelp">
          <div class="modal-dialog modal-dialog-center modal-lg col-md-10">
            <div class="modal-content">
              <div class="modal-header">
                   
                <h2 class="text-center" id="">User Guide and Help</h2>
                <button type="button" class="close" data-dismiss="modal">&times;</button>
              </div>
              <div class="modal-body" style="height: 700px;">
                <embed src="{{ url('docs/usermanual.pdf') }}" type="application/pdf" width="100%" height="100%" />
```

9.2 Controller

Instead of defining all of your request handling logic as Closures in route files, you may wish to organize this behavior using Controller classes. Controllers can group related request handling logic into a single class. Controllers are stored in the app/Http/Controllers directory.

Following are the controllers for the aapplication:

9.2.1 Wcmscontroller

```
namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Validator;
use Auth;

class Wcmscontroller extends Controller
{


    function index()
    {
     return view('woodcraft.index');
    }

    function index2()
    {
     return view('woodcraft.index2');
    }

    function forum()
    {
     return view('woodcraft.forum');
    }

    function order2()
    {
     return view('woodcraft.order2');
    }

    function orderprocess()
    {
     return view('woodcraft.orderprocess');
    }
```

```
public function order()
{

    return view('woodcraft.order');
}

public function aboutus()
{

    return view('woodcraft.aboutus');
}



public function inventory()
{

    return view('woodcraft.inventory');
}

 public function login2()
{

    return view('woodcraft.login2');
}

public function login()
{

    return view('woodcraft.login');
}

public function register()
{
```

```
public function register()
{

    return view('woodcraft.register');
}

public function settings()
{

    return view('woodcraft.settings');
}

public function admin()
{

    return view('woodcraft.admin');
}
public function aboutus2()
{
    return view('woodcraft.aboutus2');
}
public function receipt()
{
    return view('woodcraft.receipt');
}
public function orderlist()
{
    return view('woodcraft.orderlist');
}
```

9.2.2 LoginController

```
/**
 * Create a new controller instance.
 *
 * @return void
 */
public function __construct()
{
    $this->middleware('guest')->except('logout');
}

public function redirectTo()
{
    switch(Auth::user()->usertypeid)
    {
        case '1':
        return '/additems';
        break;
        default:
        return 'wcms/index2';
        break;
    }
}
```

9.2.3 RegisterController

```
     */
    protected function validator(array $data)
    {
        return Validator::make($data, [
            'fullname' => ['required', 'string', 'max:255'],
            'address' => ['required', 'string', 'max:255'],
            'contact' => ['required', 'string', 'max:255'],
            'username' => ['required', 'string', 'max:255'],
            'email' => ['required', 'string', 'email', 'max:255', 'unique:users'],
            'password' => ['required', 'string', 'min:6', 'confirmed'],
        ]);
    }

    /**
     * Create a new user instance after a valid registration.
     *
     * @param  array  $data
     * @return \App\User
     */
    protected function create(array $data)
    {
        return User::create([
            'fullname' => $data['fullname'],
            'address' => $data['address'],
            'contact' => $data['contact'],
            'email' => $data['email'],
            'username' => $data['username'],
            'password' => bcrypt($data['password']),
            'usertypeid' => 2,
        ]);
    }
```

9.2.4 ForumController

```
namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Forum;
use App\ForumReply;
use DB;


class ForumController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index()
    {
        $forum = DB::table("forum")->get();

        foreach ($forum as $forums) {
            $post_id = $forums->postid;

            $forumreply =  DB::table("forumreply")
             ->where('forumreply.postid','=',$post_id)
             ->get()
             ->toArray();
            return view('woodcraft.forum',compact(
        'forum','forumreply'));
        }


    }
     * @return \Illuminate\Http\Response
     */
    public function create()
    {
        return view('woodcraft.forum');
    }

    /**
     * Store a newly created resource in storage.
     *
     * @param  \Illuminate\Http\Request  $request
     * @return \Illuminate\Http\Response
     */
    public function store(Request $request)
    {
        $forum = new Forum;
        $forum ->post=$request->post;
        $forum ->reply=$request->reply;
        $forum ->userid=$request->userid;
        $forum ->usernme=$request->usernme;
        $forum->save();
        return redirect()->to('/post')->with('success','Post Created');
    }

    /**
     * Display the specified resource.
     *
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function show($id)
    {
        //
    }
```

9.2.5 ForumReplyController

```
use Illuminate\Http\Request;
use App\ForumReply;

class ForumReplyController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index()
    {
        $user=auth()->user();
        $forumreply = new ForumReply();
        $forumreplies = $forumreply->where('userid',$user->userid)->get();
        return view('woodcraft.forum',[
        'forumreply' => $forumreplies
    ]);


    $order = new Order();

        }

    /**
     * Show the form for creating a new resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function create()
    {
        return view('woodcraft.forum');
    }
```

```
     */
    public function create()
    {
        return view('woodcraft.forum');
    }

    /**
     * Store a newly created resource in storage.
     *
     * @param  \Illuminate\Http\Request  $request
     * @return \Illuminate\Http\Response
     */
    public function store(Request $request, $id)
    {
        $forumreply = new ForumReply;
        $forumreply ->reply=$request->reply;
        $forumreply ->postid=$id;
        $forumreply ->userid=$request->userid1;
        $forumreply ->username=$request->username1;
        $forumreply->save();
        return redirect()->to('/post');
    }

    /**
     * Display the specified resource.
     *
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function show($id)
    {
        //
    }
```

9.2.6 ItemController

```
namespace App\Http\Controllers;

use Illuminate\Http\Request;

use App\Item;

class ItemController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index()
    {
        $item = new Item();
        $items = $item->get();
        return view('woodcraft.inventory',[
        'item' => $items
    ]);
    }

    /**
     * Show the form for creating a new resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function create($id)
    {
        return view('woodcraft.inventory');
    }

    /**
```

```
     * Store a newly created resource in storage.
     *
     * @param  \Illuminate\Http\Request  $request
     * @return \Illuminate\Http\Response
     */
    public function store(Request $request)
    {
        $item = new Item;
        $pictureInfo = $request->file('item_image');

        $picName = $pictureInfo->getClientOriginalName();

        $folder = "itemImages/";

        $pictureInfo->move($folder,$picName);

        $picUrl = $folder.$picName;
        if(Item::where('item_image', '=', $picUrl)->exists())
        {
            return redirect('/additems');
        }


        $item ->item_name=$request->item_name;
        $item ->item_type=$request->item_type;
        $item ->price=$request->price;
        $item ->item_description=$request->item_description;

        $item->item_image=$picUrl;

        $item->save();
        return redirect()->to('/additems')->with('success','New
    }
```

```
    }

    /**
     * Update the specified resource in storage.
     *
     * @param  \Illuminate\Http\Request  $request
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function update(Request $request, $id)
    {
        //
    }

    /**
     * Remove the specified resource from storage.
     *
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function destroy($id)
    {
        $items = Item::find($id);
        $items->delete();
        return redirect()->to('/additems')->with('success', 'Item Deleted.');
    }
```

9.2.7 OrderController

```
use App\Item;
use DB;
use App\Order;
use Carbon;

class OrderController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index()
    {
        $item = new Item();
        $items = $item->get();
        return view('woodcraft.order2',[
        'item' => $items
    ]);
    //
    }
public function orderlist()
{
    $user=auth()->user();
    $order = new Order();
        $orders = DB::table('order')
        ->join('item','item.itemid','=','order.itemid')
        ->select('order.*','item.*')
        ->where('order.userid','=',$user->userid)
        ->get();
        return view('woodcraft.orderlist',[
        'order' => $orders
    ]);

}
```

```
public function store(Request $request)
{
    //
}

/**
 * Display the specified resource.
 *
 * @param  int  $id
 * @return \Illuminate\Http\Response
 */
public function show($id)
{
    $orders = DB::table('item')->get()->toArray();
    return view('woodcraft.order2', compact('item'));


}
public function show2($id)
{

     $orders = DB::table('item')->where('itemid',$id)->get()->toArray();
    return view('woodcraft.orderprocess', compact('orders'));

}

/**
 * Show the form for editing the specified resource.
 *
 * @param  int  $id
 * @return \Illuminate\Http\Response
 */
public function edit($id)
{
```

```
/**
 * Show the form for creating a new resource.
 *
 * @return \Illuminate\Http\Response
 */
public function create(Request $request, $id)
{
        $user=auth()->user();
        $order = new Order();
         $order->order_date=(Carbon\Carbon::now('Asia/Kathmandu')->toDateString('Y-m-d'));
        $order->quantity=$request->quantity;
         $order->userid=$user->userid;
        $order->itemid=$id;
        $order->save();

        $getBill = DB::table('order')
        ->join('item','item.itemid','=','order.itemid')
        ->select('order.*','item.*')
        ->get()->last();


    return view('woodcraft.receipt',compact('getBill'));
}
```

9.2.8 Settings

```
 * @param  int  $id
 * @return \Illuminate\Http\Response
 */
public function update(Request $request, $id)
{
    $user=User::find($id);
    $user->fullname=$request->fullname;
    $user->address=$request->address;
    $user->contact=$request->contaact;
    $user->email=$request->email;
    $user->username=$request->username;
       $user->save();
       return view('woodcraft.inventory')->with('success','Profile Updated');

}

/**
 * Remove the specified resource from storage.
 *
```

9.3 Migrations

Migrations are like version control for your database, allowing your team to easily modify and share the application's database schema. Migrations are typically paired with Laravel's schema builder to easily build your application's database schema. If you have ever had to tell a teammate to manually add a column to their local database schema, you've faced the problem that database migrations solve.

Following are the migrations of the application:

9.3.1 usertype

```
use Illuminate\Support\Facades\Schema;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateUsertype extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('usertype', function (Blueprint $table) {
            $table->increments('usertypeid');
            $table->string('usertype',10);
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('usertype');
    }
}
```

9.3.2 user

```
class CreateUser extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('users', function (Blueprint $table) {
            $table->increments('userid');
            $table->string('fullname',30);
            $table->string('address',50);
            $table->string('contact',10);
            $table->string('email',25);
            $table->string('username',15)->unique();
            $table->string('password');

            $table->unsignedInteger('usertypeid');
            $table->foreign('usertypeid')->references('usertypeid')->on('usertype');
            $table->rememberToken();

            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('user');
    }
}
```

9.3.3 item

```php
use Illuminate\Support\Facades\Schema;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateItem extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('item', function (Blueprint $table) {
            $table->increments('itemid');
            $table->string('item_name',30);
            $table->string('item_type',30);
            $table->float('price');
            $table->string('item_description',200);
            $table->string('item_image');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('item');
    }
}
```

9.3.4 order

```php
use Illuminate\Support\Facades\Schema;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateOrder extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('order', function (Blueprint $table) {
            $table->increments('orderid');
            $table->date('order_date');
            $table->integer('quantity');

            $table->unsignedInteger('userid');
            $table->foreign('userid')->references('userid')->on('users');


            $table->unsignedInteger('itemid');
            $table->foreign('itemid')->references('itemid')->on('item');
            $table->rememberToken();


            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('order');
    }
```

9.3.5 forum

```
.
use Illuminate\Support\Facades\Schema;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateForum extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('forum', function (Blueprint $table) {
            $table->increments('postid');
            $table->string('post',200);
            $table->string('reply',200);
            $table->string('usernme',50);
            $table->unsignedInteger('userid');
            $table->foreign('userid')->references('userid')->on('users');
            $table->rememberToken();
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('forum');
    }
}
```

9.3.6 forumreply

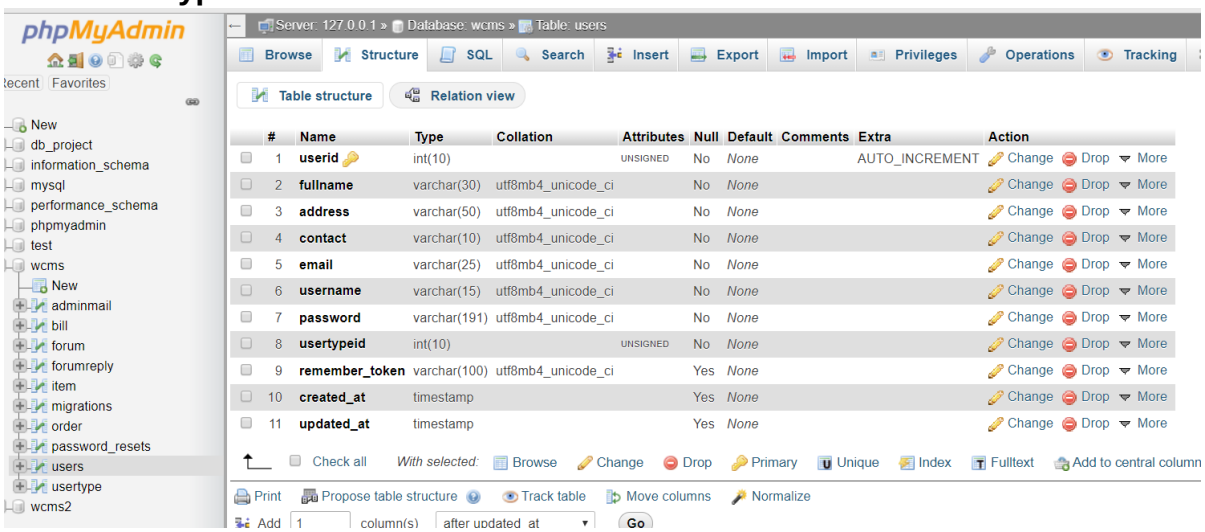
```
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateForumreply extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('forumreply', function (Blueprint $table) {
            $table->increments('replyid');
            $table->string('reply',200);
            $table->string('username',50);

            $table->unsignedInteger('postid');
            $table->foreign('postid')->references('postid')->on('forum');

            $table->unsignedInteger('userid');
            $table->foreign('userid')->references('userid')->on('users');
            $table->rememberToken();
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('forumreply');
    }
}
```

9.4 Database in SQL Server

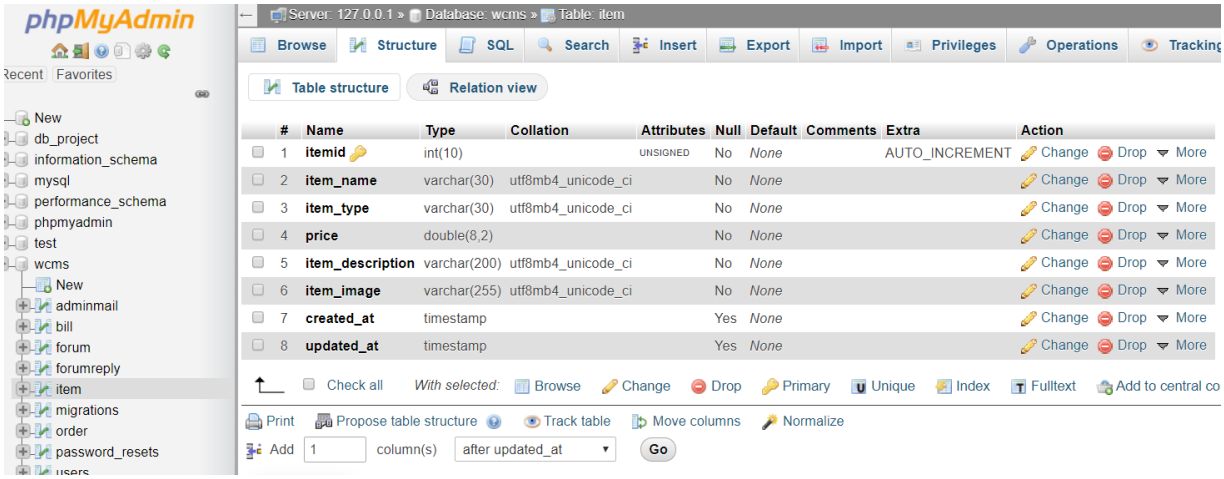
SQL Server database is of two types; System databases are created automatically when we install MS SQL Server. Users. System User databases are created by users (Administrators, developers, and testers who have access to create databases).

Following are the SQL databases for this application:

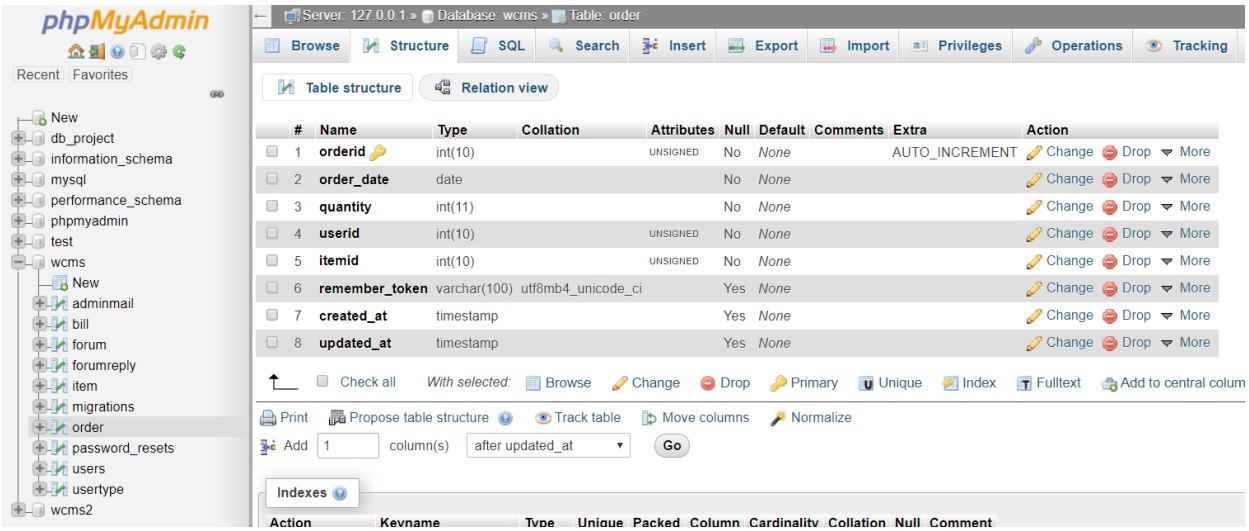
9.3.1 usertype



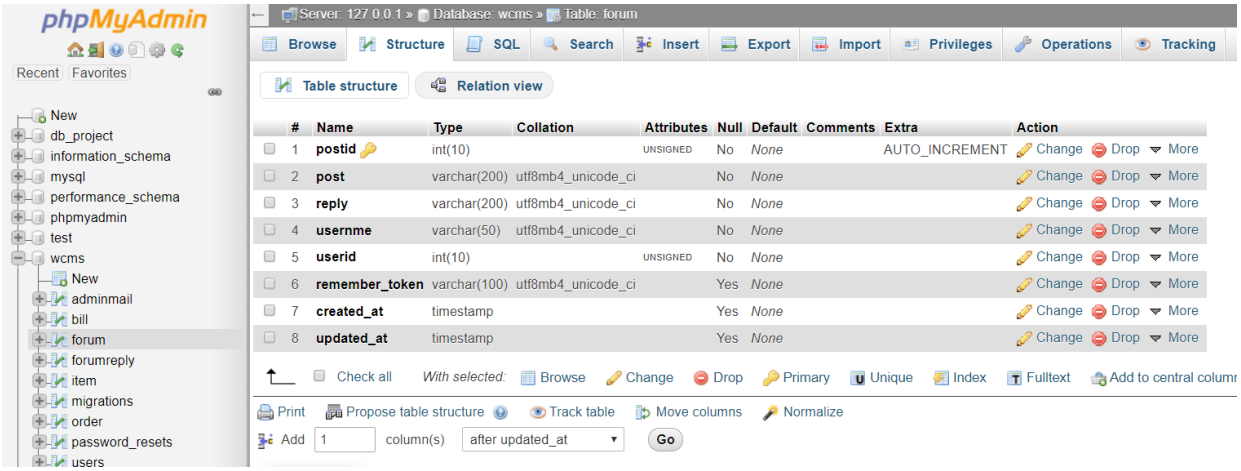
9.3.2 user



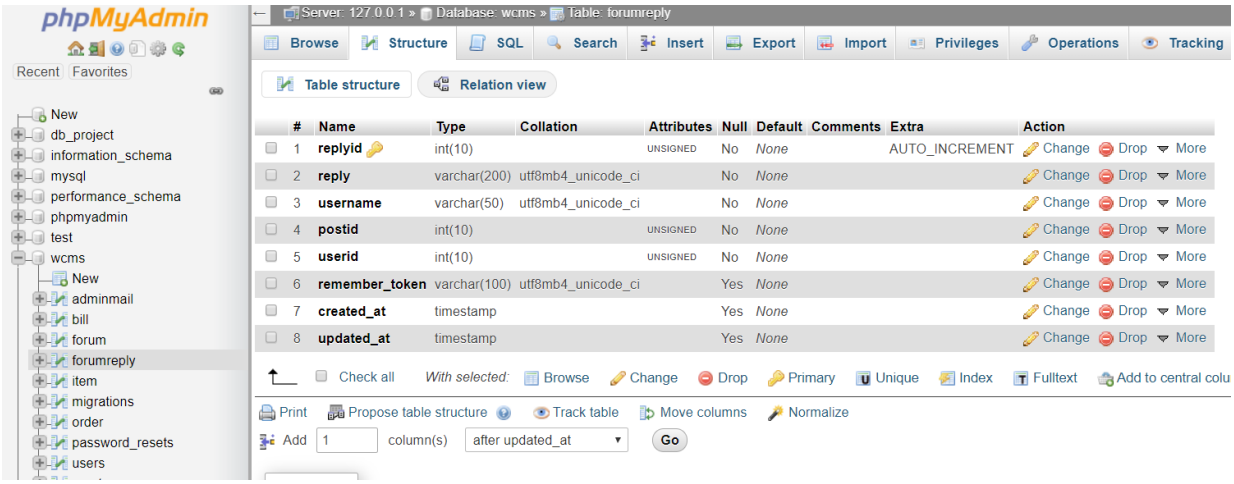
9.3.3 item



9.3.4 order



9.3.5 forum



9.3.6 forumreply