**README: Project Structure and Important Files**

This system is implemented as an online web-based application and a responsive web application that enables access via the mobile devices also. Proposed system is developed according to the Model-View-Controller (MVC) architecture. MVC is a popular software architectural pattern which splits the different responsibilities among different layers. Here, the model is handle database functions and related business logics. View is responsible for UI related functions. Controller executes the user’s commands and also bridges the communication between Model and View. Technologies such as Apache web server, PHP, HTML, CSS, MySQL is used to develop this web application.

Web server

Application server

Controller

View

Model

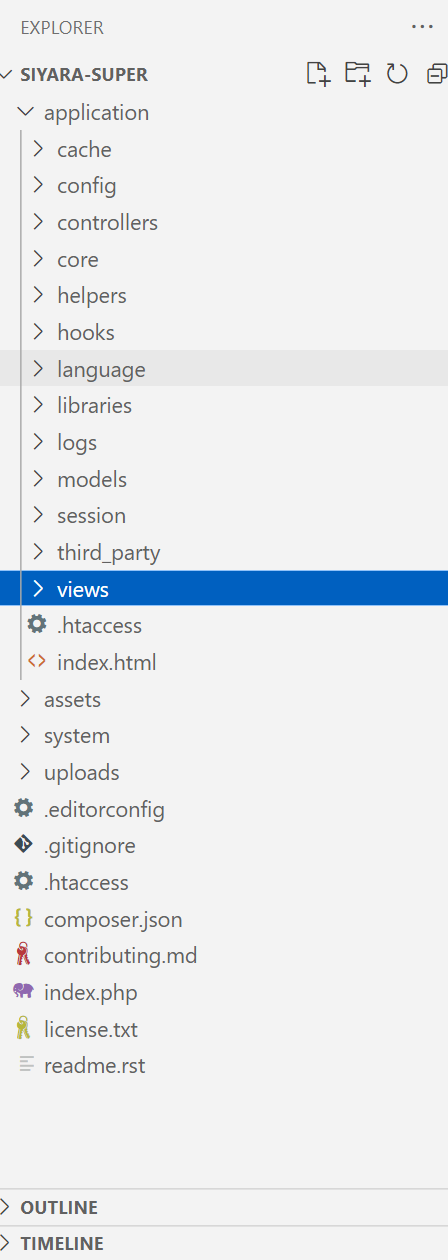
Data base

Request

Response

### *Figure 1: System Architecture*

This code is developed by following the MVC architecture. Figure 2 shows how Model\_View\_controller appears in the code structure section.



*Figure 2:Appear of MVC architecture in the project structure*

## Hardware Requirements

|  |  |
| --- | --- |
| Type of computer | ThinkPad |
| Processor type | Intel(R) Core(TM) i7 |
| Processor speed | 2.6 Ghz |
| RAM | 16GB |
| Hard disk | 256GB |

Table 1: Hardware environments

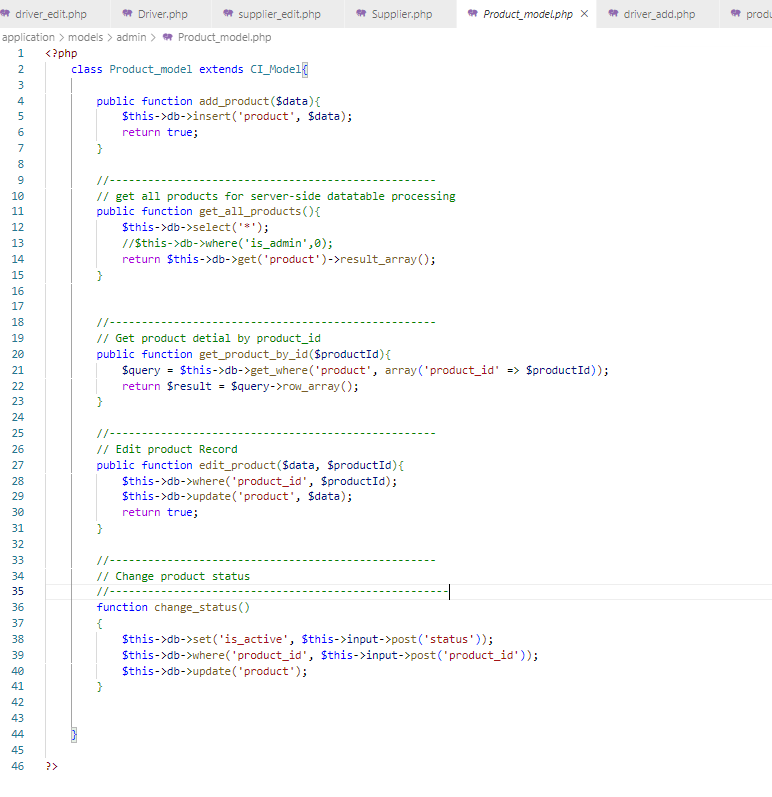
## Software Reqirements

|  |  |
| --- | --- |
| Operating System | Windows 10 |
| Development Framework | Codeignator |
| Development Language | PHP , JavaScript, BootStrap,CSS |
| IDE | Visual studio code |
| Web Server | Apache |
| Database | MYSQL |

Table 2: Software environments

According to the MVC architecture the system can be implemented as given below. As the sample code segment we can take Product\_Model ( shown in Figure 3), Product\_Controller (Figure 4. Figure 5,Figure 6,Figure 7, and Figure 8) Product\_view (Figure 9 and Figure 10) to take insight regarding functionality of the system.

**Product\_Model Code Segment**

****

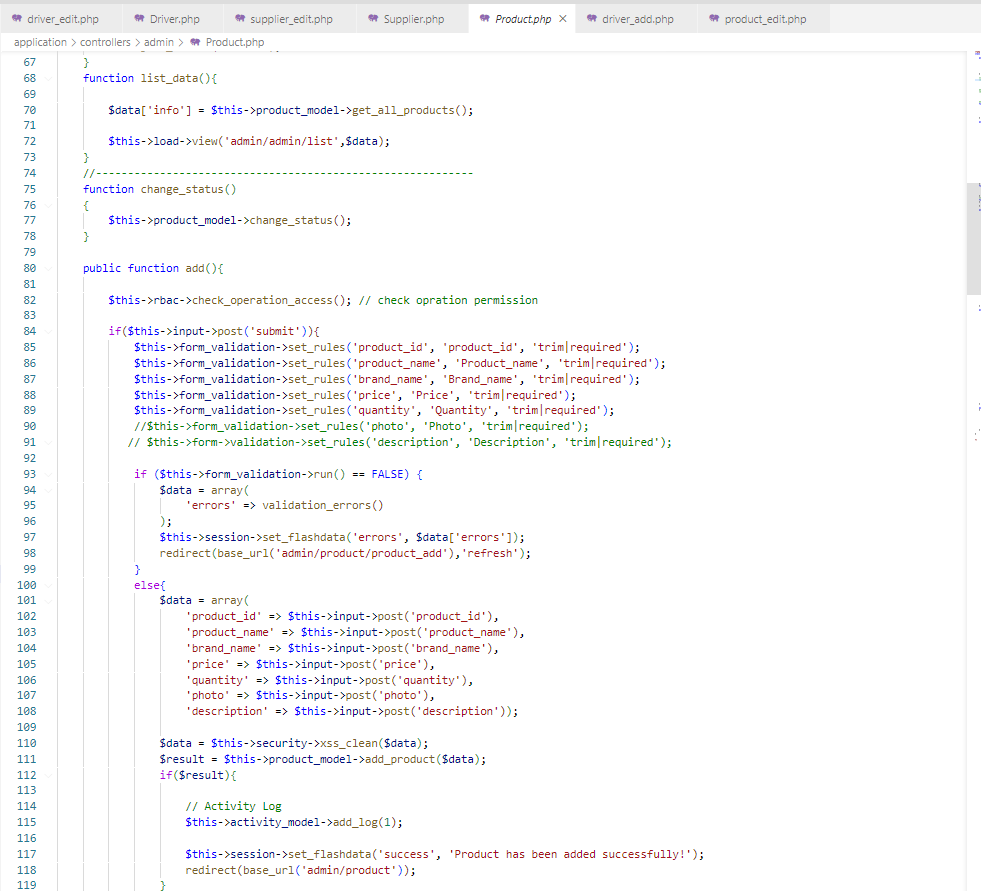
### *Figure 3: Product\_Model code segment*

## Product\_Controller Code Segment

### *Figure 4: Product\_Controller code segment (Part 1)*



### *Figure 5: Product\_Controller code segment (Part 2)*

**

### *Figure 6: Product\_Controller code segment (Part 3)*

### 

### *Figure 7: Product\_Controller code segment (Part 4)*

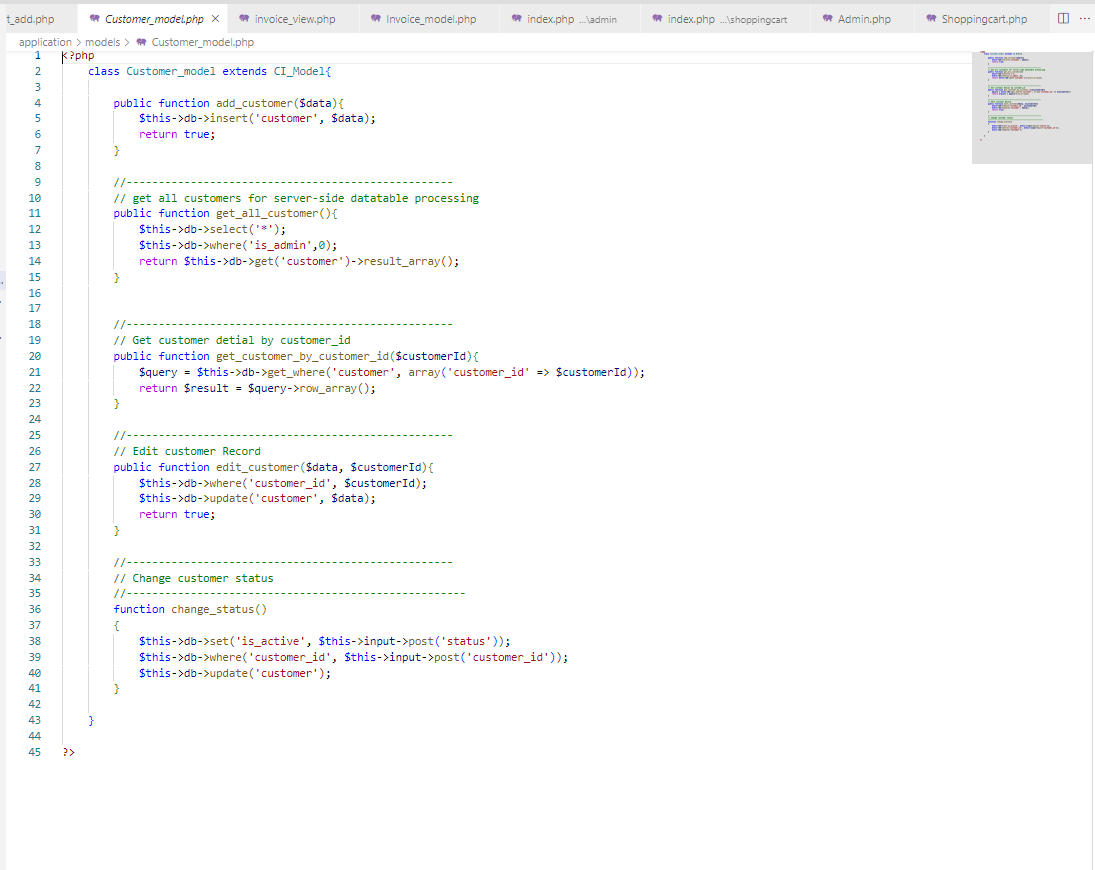
### *Figure 8: Product\_Controller code segment (Part 5)*

## Product\_View Code Segment

### *Figure 9: Product\_View code segment (Part 1)*

**Customer\_Model Class**

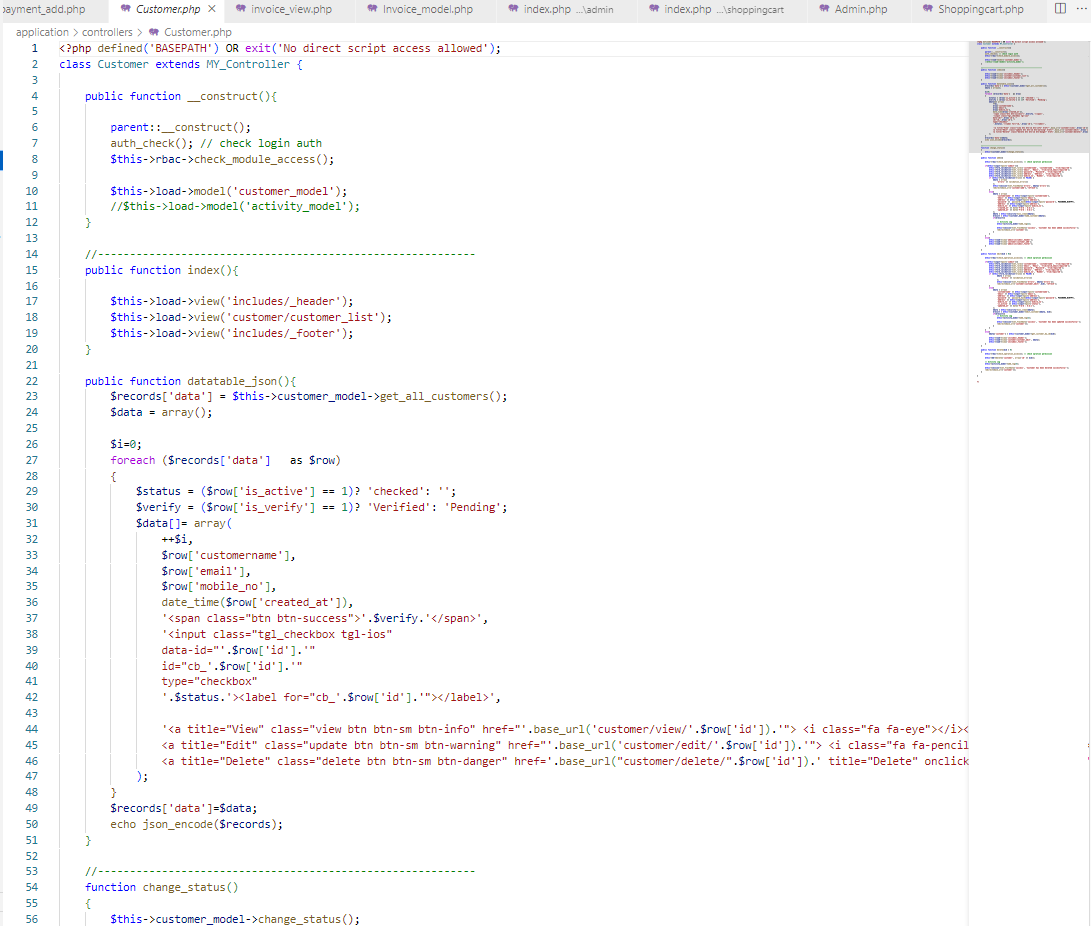
Figure 10 shows the coding section for customer\_model class.



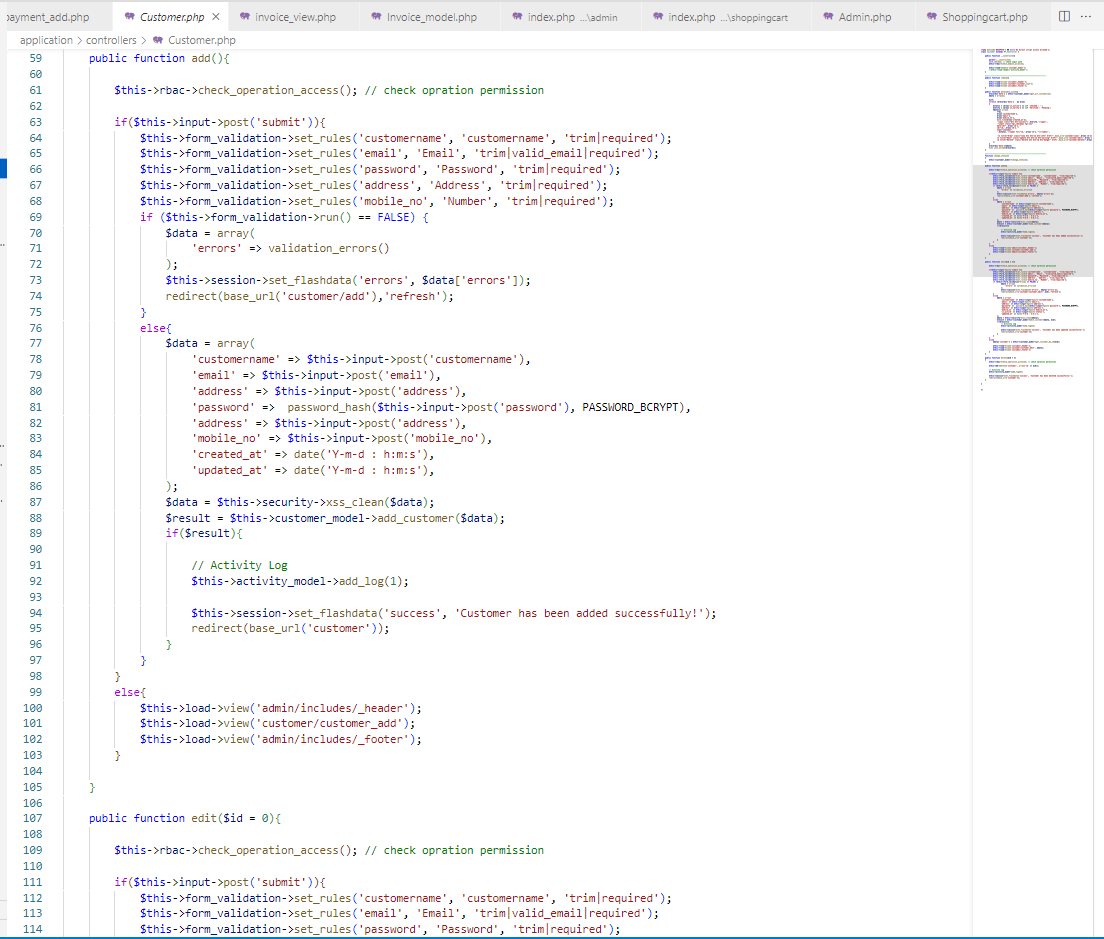
*Figure 10 : Customer\_Model class*

**Customer\_Controller Class**

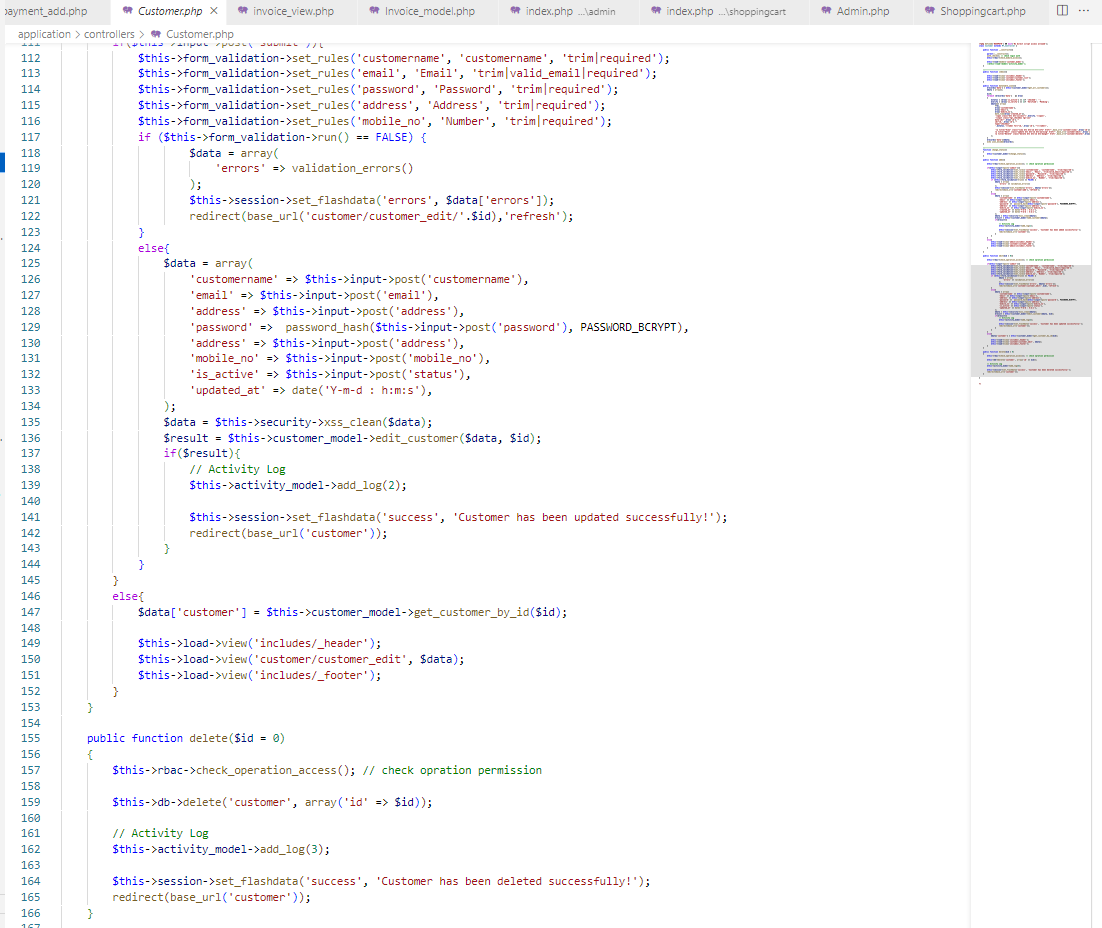
Figure 11, Figure 12, and Figure 13 show the coding sections for customer\_controller class.



*Figure 11 : Customer\_Controller class (part 1)*



*Figure 12: Customer\_Controller class (part 2)*

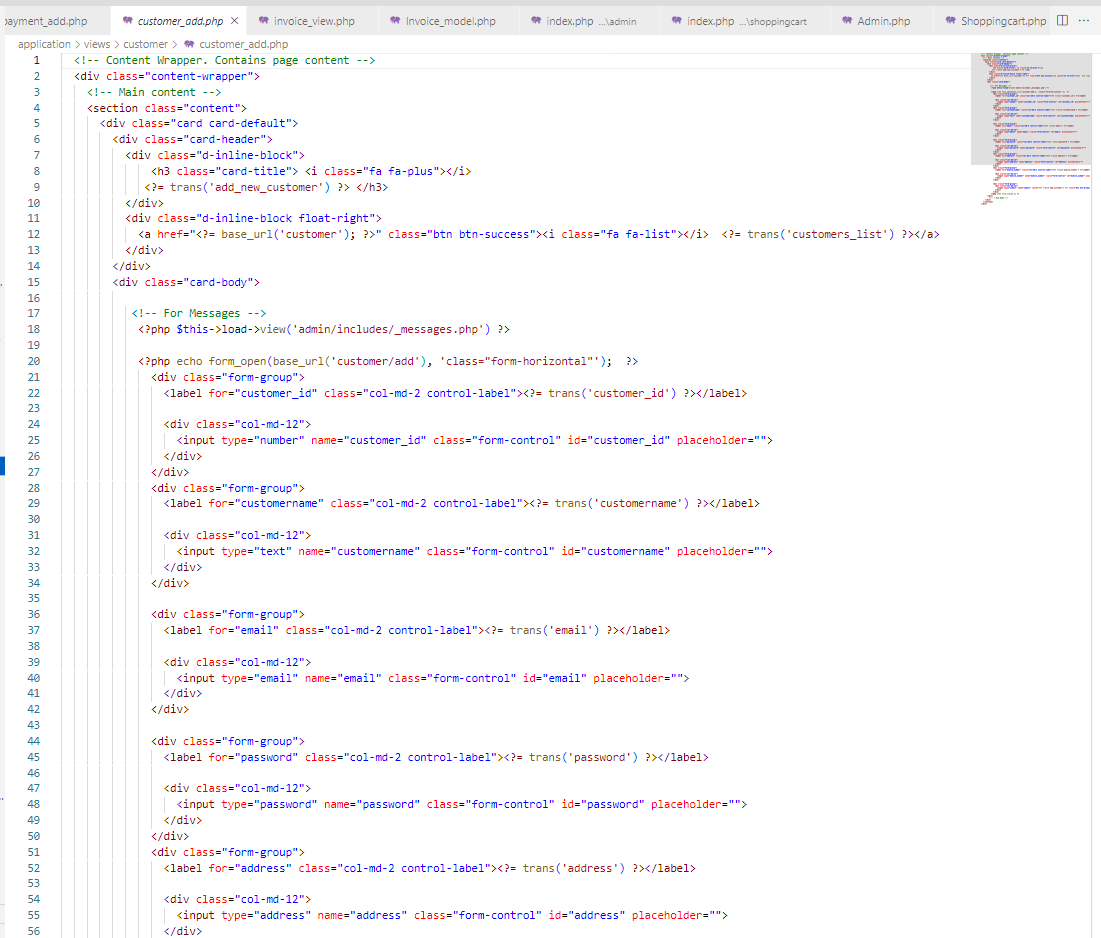


*Figure 13 : Customer\_Controller class (part 3)*

**Customer\_View Class**

**Coding for Customer\_add Section**

Figure 14, and Figure 15 shows coding section for customer\_add.



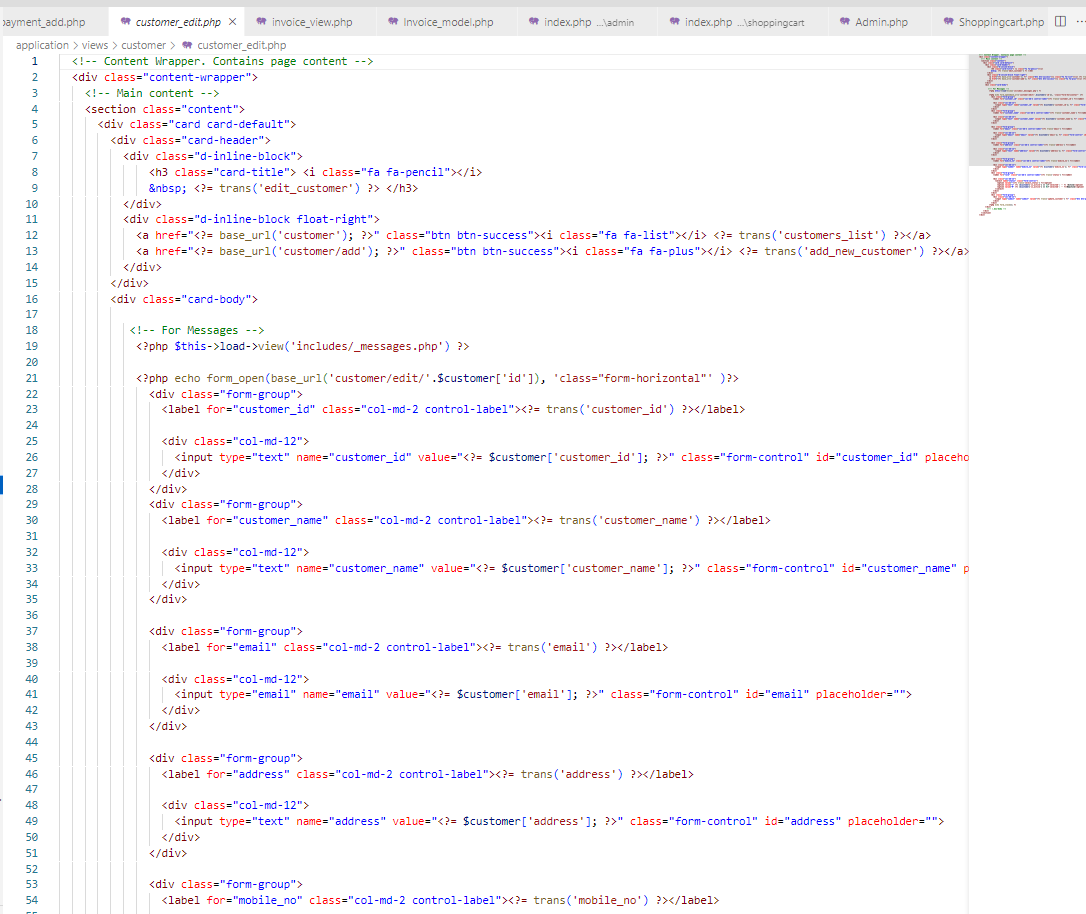
*Figure 14 : Customer\_Add (part 1)*



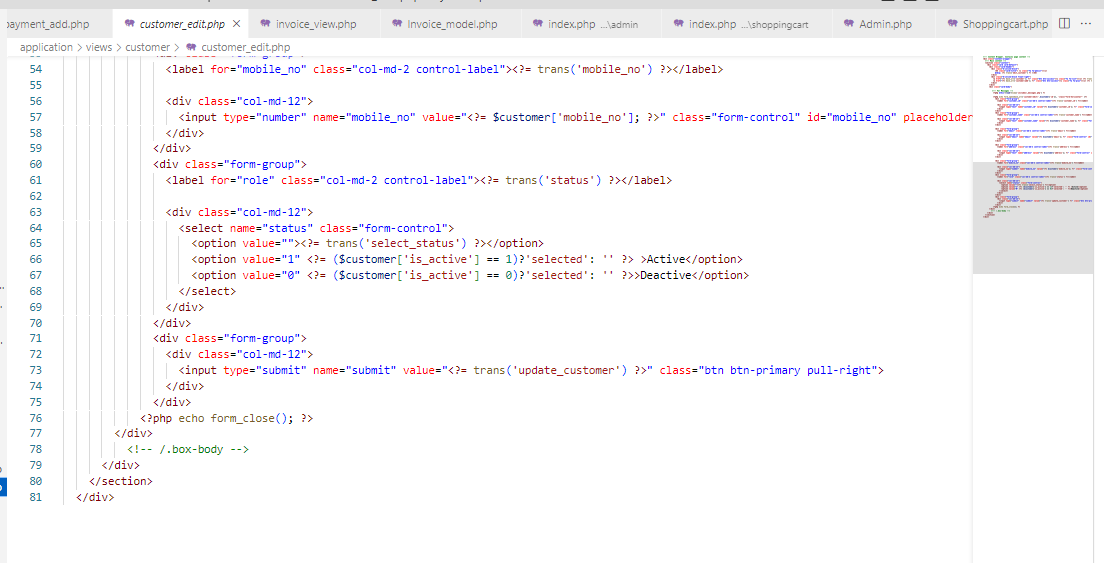
*Figure 15: Customer\_Add (part 2)*

**Coding for Customer\_Edit Section**

Figure 16, and Figure 17 shows the coding section for customer\_edit.



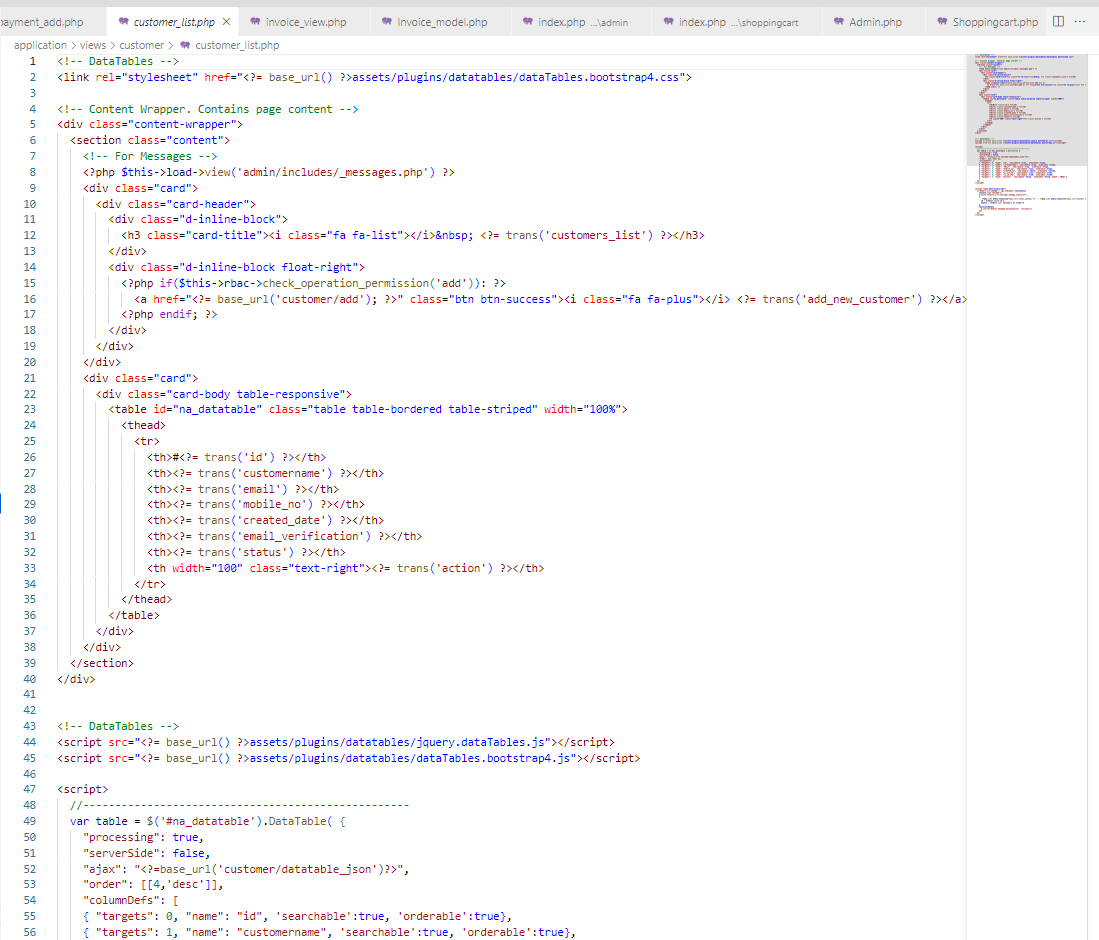
*Figure 16: Customer\_Edit (part 1)*



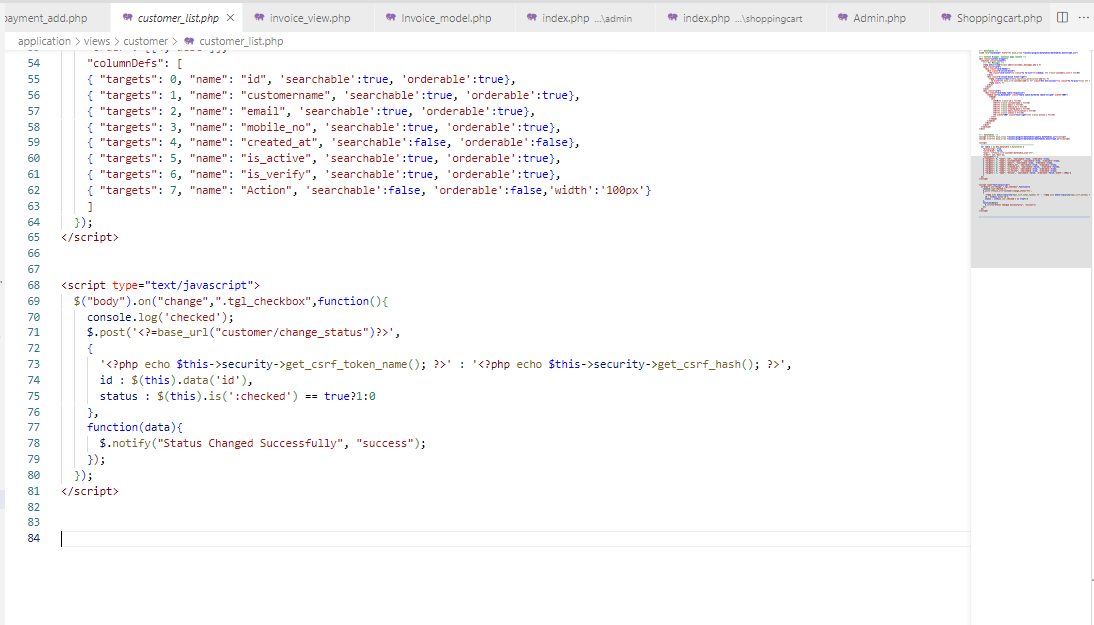
*Figure 17 : Customer\_Edit (part 2)*

**Coding for Customer\_List Section**

Figure 18, and Figure 19 shows the coding section for customer\_list.



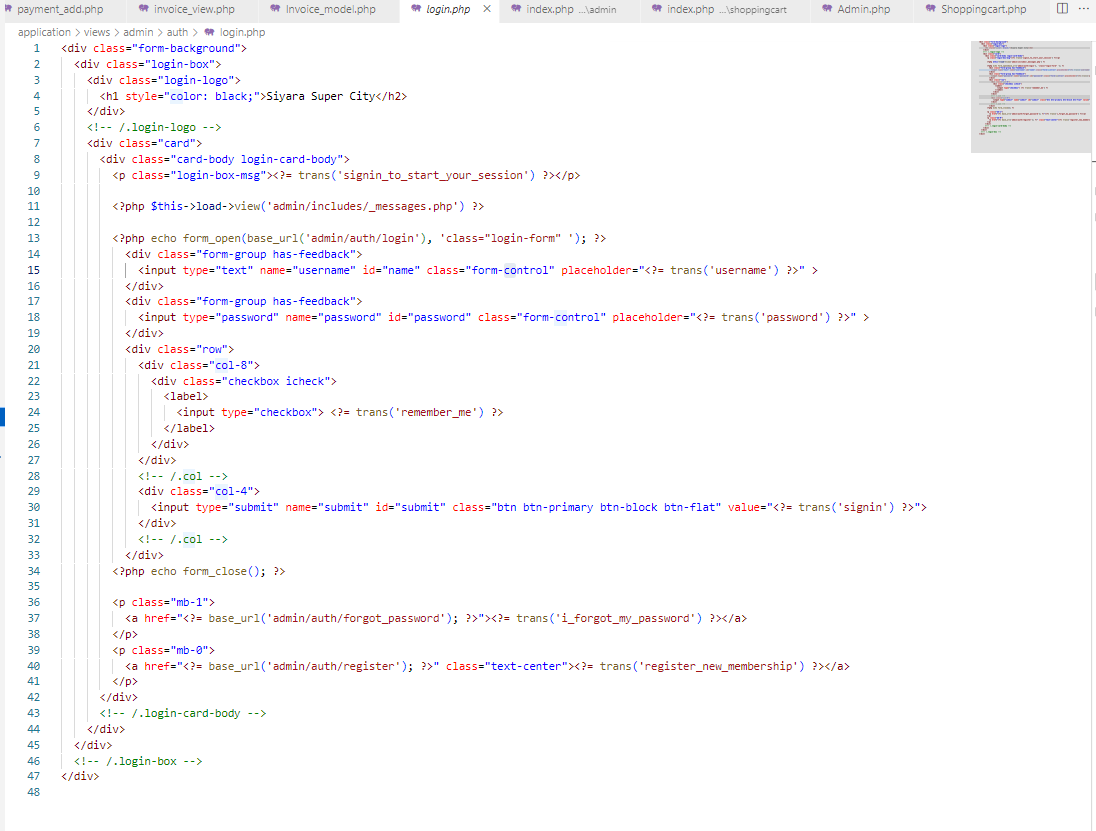
*Figure 18 : Customer\_List (part 1)*



*Figure 19 : Customer\_List (part 2)*

**Code Section For Login Page**

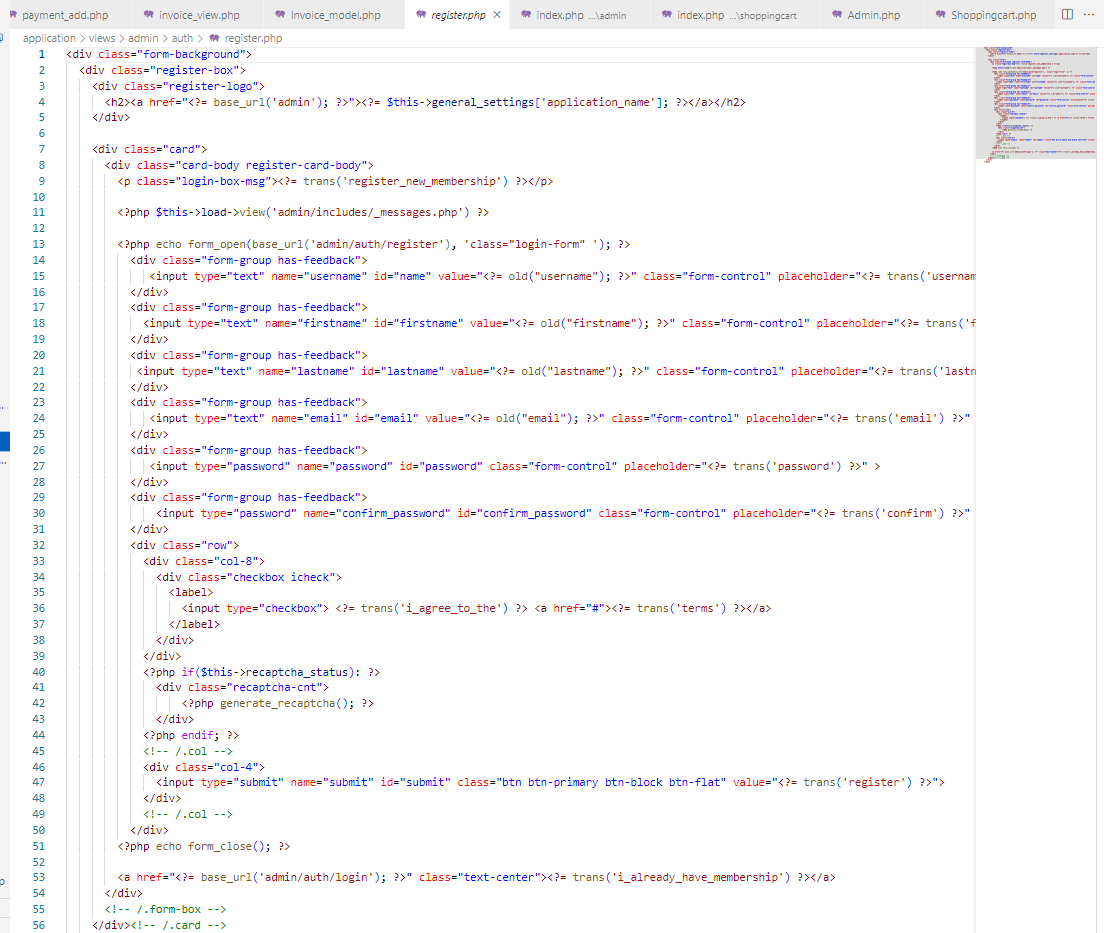
Figure 20 shows the coding section for the login page.



*Figure 20 : Coding section for login page*

**Code Section For Register Page**

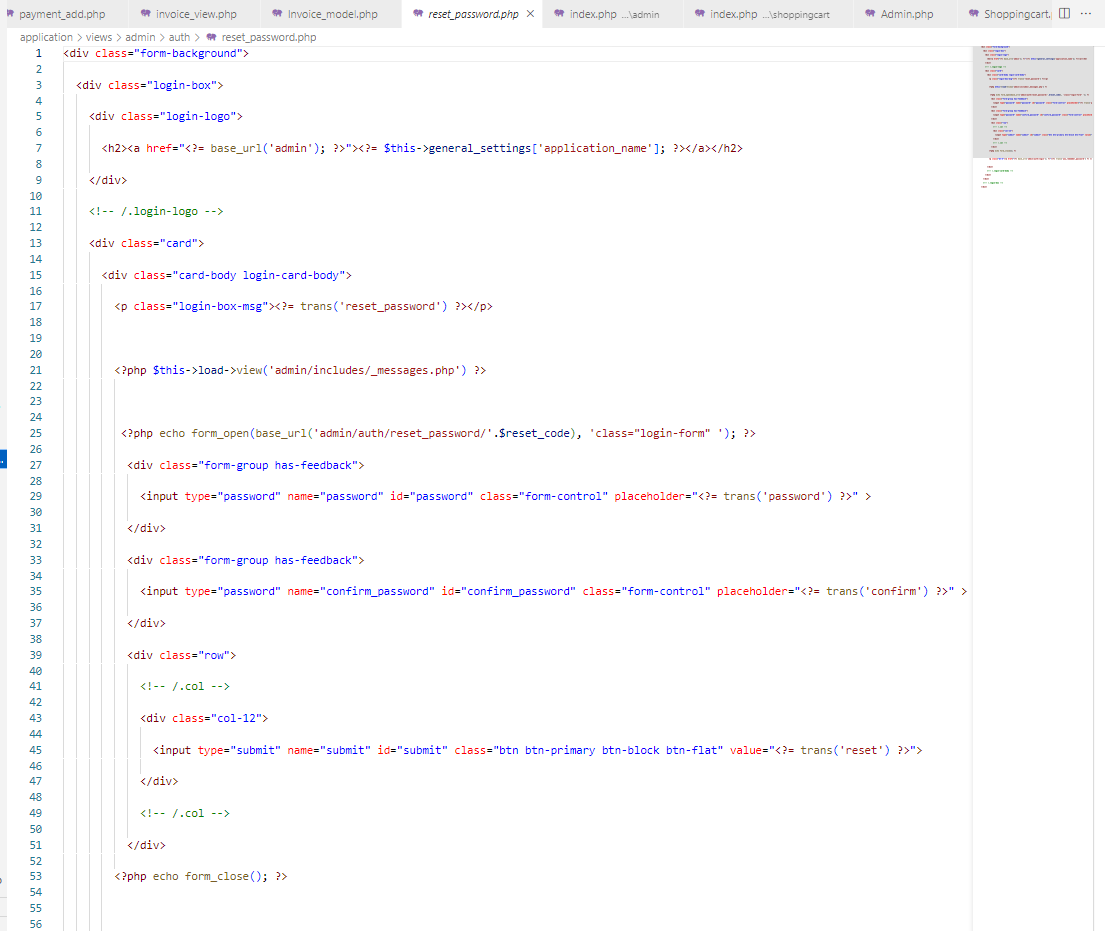
Figure 21 shows the coding section for register page.



*Figure 21: coding section for register page*

**Code Section For Reset Password**

Figure 22 shows the coding section for reset password.



*Figure 22 : coding section for reset password*

**Relational Database Design**

Databases helps us to persist system data for longer term. In a web application, generally, data is kept in variables and as objects. Those data should be converted and persist on data bases to store them permanently. Those stored data then can be retrieved, modified, and deleted using CRUD operations performed via web app. Table 3.1 shows how those identified entities and their relationships in EER diagram will be implemented as tables in the proposed relational database schema of the web application. Each table will keep collection of related records. Each table certain key constraints will be implemented. Primary key stands for the field that is unique for all the record occurrences and foreign key stands for the field use to set relation between tables. Normalization helps to avoid redundancy in the tables.

|  |  |  |  |
| --- | --- | --- | --- |
| **Table name** | **Field name** | **Data type** | **Key constraint** |
| **User**  (Information about users) | User\_id | int(10) | PRIMARY KEY - NOT NULL |
| User name | varchar(50) | NOT NULL |
| Password | varchar(20) | NOT NULL |
| E-mail | varcha(30) | NOT NULL |
| Address | varcha(70) | NOT NULL |
| Mobile number | int(10) | NOT NULL |
| **Customer**  (Information about customer) | Customer\_id | int(10) | PRIMARY KEY – NOT NULL |
| Customer name | varchar(50) | NOT NULL |
| Password | varchar(20) | NOT NULL |
| E-mail | varchar(30) | NOT NULL |
| Address | varchar(70) | NOT NULL |
| Mobile number | int(10) | NOT NULL |
| **Order** | Order\_id | int(10) | PRIMARY KEY – NOT NULL |
| Customer\_id | int(10) | FOREIGN KEY – NOT NULL |
| Total amount | int(10) | NOT NULL |
| Discount percent | int(10) | NOT NULL |
| Discount amount | int(10) | NOT NULL |
| Payable amount | int(10) | NOT NULL |
| Billing address | varchar(70) | NOT NULL |
| Delivery address | varchar(70) | NOT NULL |
| Date | varchar(10) | NOT NULL |
| **Order info** (Order information) | Order info\_id | int(10) | PRIMARY KEY – NOT NULL |
| Order\_id | int(10) | FOREIGN KEY – NOT NULL |
| Product\_id | int(10) | FOREIGN KEY – NOT NULL |
| Quantity | int(10) | NOT NULL |
| Product price | int(10) | NOT NULL |
| **Product** | Product\_id | int(10) | PRIMARY KEY – NOT NULL |
| Product name | varchar(30) | NOT NULL |
| Price | int(10) | NOT NULL |
| Quantity | int(10) | NOT NULL |
| Brand name | varchar(30) | NOT NULL |
| Photo | varchar(500) | NOT NULL |
| Description | varchar(500) | NOT NULL |
| **Payment** | Payment\_id | int(10) | PRIMARY KEY – NOT NULL |
| Customer\_id | int(10) | FOREIGN KEY – NOT NULL |
| Order\_id | int(10) | FOREIGN KEY – NOT NULL |
| Total amount | int(10) | NOT NULL |
| Date | varchar(10) | NOT NULL |
| **Transaction report** | Transaction\_id | int(10) | PRIMARY KEY – NOT NULL |
| Customer\_id | int(10) | FOREIGN KEY – NOT NULL |
| Order\_id | int(10) | FOREIGN KEY – NOT NULL |
| Product\_id | int(10) | FOREIGN KEY – NOT NULL |
| Payment\_id | int(10) | FOREIGN KEY – NOT NULL |
| date | vatchar(10) | NOT NULL |
| **Deliveries** | Delivery\_id | int(10) | PRIMARY KEY – NOT NULL |
| Customer\_id | int(10) | FOREIGN KEY – NOT NULL |
| Order\_id | int(10) | FOREIGN KEY – NOT NULL |
| Product\_id | int(10) | FOREIGN KEY – NOT NULL |
| Vehicle\_id | int(10) | FOREIGN KEY – NOT NULL |
| Date | varchar(10) | NOT NULL |
| **Vehicle** | Vehicle\_id | int(10) | PRIMARY KEY – NOT NULL |
|  | Type | varchar(10) | NOT NULL |

* + 1. Table 2: Relational database design

-----------------------------END-------------------------------