

1. SYNOPSIS

1.1 Title

“Countryside Cultivators co”

1.2 Introduction:

Countryside cultivators Co is a web based fictitious company that specializes in selling agricultural products and equipment to farmers and other agricultural enthusiasts. The company aims to provide high – quality products and equipment to its customers to enhance their farming experience and yield. The company operates primarily through its website, which serves as an e-commerce platform for its products.

1.3 Objective:

- The main objective of this web application is to computerize the manual system and reduce the time consumption.
- To help farmers to receive tools and equipment’s in a more reasonable price.
- It reduces the error scope and no paper work required.
- Administrator of the application has the authority to manage all the users.

1.4 Project Category:

Web Based online shopping and database management system.

1.5 Hardware and Software Requirements:

1.5.1 Hardware Requirements:

- Processor: Intel Core i3-2340UE or any higher version
- Processor Speed: Minimum 3.70 GHz
- RAM: 8GB
- Hard Disk: 512GB or above

1.5.2 Software Requirements:

- Operating System: Windows 11.
- Text Editor: VS code editor.
- Server: Xampp server with MySQL database.
- Browser: Chrome, Mozilla Firefox or any other.

1.6 Languages used:

Front end: HTML, CSS, JavaScript, Bootstrap

Back end: PHP, MySQL

1.7 Operating System Used:

Windows 11 Operating System.

1.8 Modules:

1.8.1 Admin side:

- Login
- Logout
- Create product
- Products
- Help
- Contact info
- Dashboard
- orders

1.8.2 User side:

- Login
- Signup
- Shop
- Register
- Cart
- Checkout
- Profile
- Home
- Contact us
- Other products
- Logout
- Payment options

1.9 Scope:

1. The system is developed as a web application. This system can also be accessed by handheld devices like cell phones through browser in the future.
2. New features such as purchase of seeds, fertilizers and other farming products will be added.
3. Settings option will be added in the future updates.
4. Online payment options will be added in future updates.
5. Social media platforms will be added in the upcoming updates.

2. SOFTWARE REQUIREMENT ANALYSIS AND SPECIFICATION

2.1 Introduction:

Software Requirement Specification (SRS) is a document, which describes completely the external behaviour of the software. First and foremost the work of a software developer is to study the system to be developed and specify the user requirements before going for the designing phase. This document will let us know how this system behaves and responds.

2.1.1 Purpose:

The main purpose of SRS is to translate the idea in the mind of a client into a formal document, through SRS the client clearly describes what it expects from the proposed system and the developer clearly understands what capabilities are required to build the system.

2.1.2 Scope:

This SRS describes the requirement of the system. It is meant for use of the developer that will be the basis for validating the final delivered system; any changes made to the requirement in future will have to go through the formal change approval process.

2.1.3 Definition, Acronyms and Abbreviations:

TCP/IP: Transmission Control Protocol/Internet Protocol.

DFD: Data Flow Diagram.

E-RD: Entity-Relationship Diagram.

SRS: Software Requirement Analysis and Specification.

SQL: Structured Query Language.

RAM: Random Access Memory.

XAMPP: Cross-Platform Apache MariaDB/MySQL PHP and Perl.

CSS: Cascading Style Sheet.

CFD: Context Flow Diagram.

2.1.4 Refences:

Internet

https://en.wikipedia.org/wiki/Software_requirements_specification

<https://youtube.com/@learnwithdevil/>

<https://drawio.com>

2.1.5 Overview:

The Countryside Cultivators Co is a web-based application. This application provides enough security through the user login so that unauthorized person can't access personal information and update wrong information.

2.2 Overall Description:

2.2.1 Product perspective:

Countryside cultivators co is a web based fictious company that specializes in selling agricultural products and equipment to farmers and other agricultural enthusiasts. The company aims to provide high – quality products and equipment to its customers to enhance their farming experience and yield. The company operates primarily through its website, which serves as an e-commerce platform for its products.

- Good and easy user interfaces which makes the system user friendly.
- Active workspaces for the users and the admin with many functions.
- Secured and easy access.

2.2.2 Product Functions:

- The user interface for this website is easy to use.
- It can be accessed by admin and the users at any time in anyplace.
- The user can view all the products and services.
- The admin can manage all the users of the application.

2.2.3 User Characteristics:

The system is designed with the intention to provide easy to use simple system so that no cumbersome and elaborate training for operation is needed.

Countryside Cultivators Co website has 2 levels of users.

Users:

- The users can login to the system with the email and passwords provided by the user at the time of registration.
- The users can update the contact number and the address details anytime they want.
- The users can login to the system with the email and passwords provided by the user at the time of registration.
- The users can update the contact details anytime they want.

Admin:

- The users can login to the system with the email and passwords.
- The admin can view all the details and the users who have registered.
- The admin can delete any of the users if invalid details are found.

2.2.4 General Constraints:

- User should be familiar with basic Computer knowledge.
- The system is completely dependent on Internet connection.
- The information provided by the user the assumed to be genuine.

2.2.5 Assumption and Dependencies:

- The user should be familiar with the primary usage of internet.
- The developed system can run on platform (web application).

2.3 Specific Requirements:

This section describes all the details that the system developer needs to know for designing and developing this system, these requirements can be organized by the modes of operation user class object, features and functional hierarchies.

2.3.1 External Interface Requirements:

It specifies all the interface of the system: to the people, other system, hardware and other system.

2.3.1.1 User Interfaces:

- The user can access the site through a web browser.
- Home page which has links to other pages.
- He can navigate to the various icons and view the products on the system.
- A start validation is provided to the login form. On successful validation, the permission to use the system is provided.

2.3.1.2 Hardware Interfaces:

Windows device which has a web browser installed in it.

2.3.1.3 Software Interfaces:

In software Interfaces the system requirements needed for the Project to run on a system is specified.

- Any web browser

2.3.1.4 Communication Interfaces:

- TCP/IP

2.3.2 Functional Requirements:

Functional requirements which output should be produced for the given inputs. All inputs are entered according to the data types and no blanks are allowed for mandatory fields. Invalid inputs are not allowed in the system. It prompts to re-enter the data. Appropriate error messages are displayed.

USERSIDE:

Login:

- Users have to authenticate with email and password in order to access the web application.

Registration Process:

- User need to be registered themselves before login to the Website. Add address and contact.
- User has to add their address and their contact no, which can be update later on.

Logout:

- Users can logout of the website closing the session for safety.

ADMIN SIDE:

Login:

Admin has to authenticate with email and password in order to access the admin side web application.

Manage Users:

The admin can view all the users and delete their account if found fraud.

View All Details:

The admin can view all the details updated b the users and can edit or delete these updates for the gentility.

Logout:

Logout closes the session of the admin and directs the admin to the login page where he/she will have to login again to access the admin page.

2.3.3 Non-Functional Requirements:

2.3.3.1 Performance requirements:

The performance of the overall system should be faster and errorfree, with built in error checking and correction facilities. In order to run this application, we require:

- An Internet with minimum 56 kbps bandwidth.
- To access this page, we require IE6 or any higher version browser.

3.3.3.2Design Constraints

- Email field to be entered with text in email format itself.
- The user email and password are verified and then only user can access the product.
- Requires to specify the information for all the mandatory fields.
- The application shall display error messages to the user when an error is detected.

2.3.3.3 Safety Requirements:

- The system is safe to use. The data is stored in the central database and it is made sure that thereis minimum data loss in case of any mishaps. In case the user forgets password, forgot passwordfunctionality helps to set a new password.
- Authorization helps to check the permission level of the user accessing the site.

2.3.3.4 Security Requirements:

The system is protected by the use of login system that is maintained by the admin of the system with certain set of constraints. The users can access the home page, he can go through the shop and purchase any products after logging in or registering with valid email and password. Admin have all permissions.

2.3.3.5 Software Quality attributes:

- This section of the Countryside Cultivators Co SRS describes the applications, attributes and properties.
- Security: “Countryside Cultivators Co” shall be managed by the administrator via predetermined roles.
- Maintainability: During maintenance stage, the SRS can be referred for the validation.
- Portability: Since it is a system, it is portable.
- Timeliness: The system carries out all the operations with consumption of very less time.
- Usability: The user interface of the application is very user friendly that the users can use the system without any confusion. But the user should have basic knowledge about the computer and internet.

3. SYSTEM ANALYSIS & DESIGN

3.1 Introduction:

System design is the process or art of defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements. One could see it as the application of systems theory to product development. There is some overlap with the disciplines of systems analysis, systems architecture and systems engineering. In System design focus is on deciding which modules are needed for the system, the specifications of these modules should be interconnected is called System Design.

System design is also called top-level design. Here we consider a system to be set of components with clearly defined behavior that interact with each other in a fixed manner to produce some behavior. In a system design, the design consists of module definitions, with each module supporting a functional abstraction.

3.2 Content Flow Diagram:

It is common practice to draw the context-level data flow diagram first, which shows the interaction between the system and external agents which acts as data source and data sinks. On the context diagram the system's interactions with the outside world are modelled purely in terms of data flows across the system boundary. The context diagram shows the entire systems the single process, and gives no clues as to its internal organization.


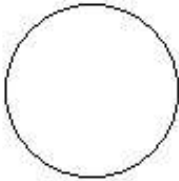




3.3 Data Flow Diagram:

Data Flow Diagram: A Data Flow Diagram (DFD) is a graphical representation of the “flow” of data through an information system. A data flow diagram can also be used for the visualization of data processing. It is common practice for a designer to draw a context-level DFD first which shows the interaction between the system and outside entities.

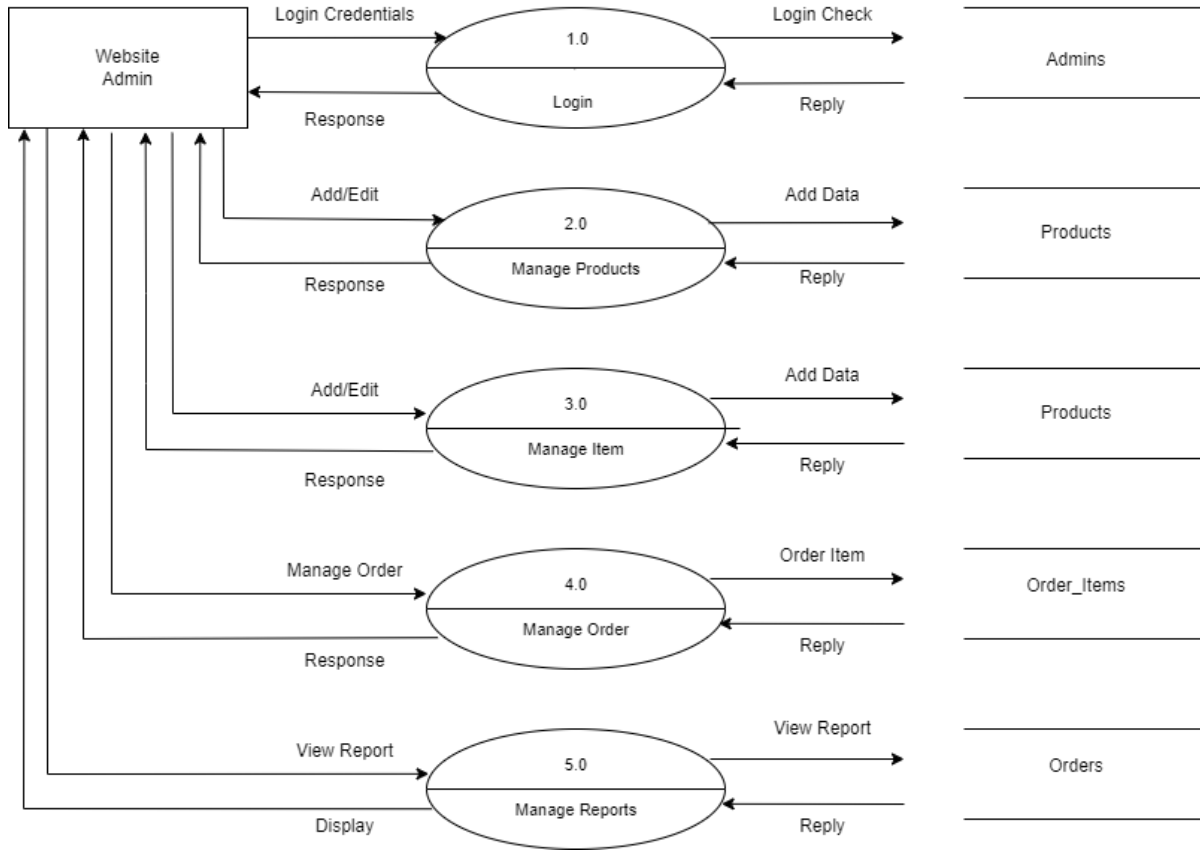
A DFD represents flow of data through a system. Data flow diagrams are commonly used during problem analysis. It views a system as a function that transforms the input into desired output. A DFD shows movement of data through the different transformations or processes in the system.

Dataflow diagrams can be used to provide the end user with a physical idea of where the data they input ultimately has an effect upon the structure of the whole system from order to dispatch to stock how any system is developed can be determined through a dataflow diagram. The appropriate register saved in database and maintained by appropriate authorities. In the normal convention, logical DFD can be completed using some notations.

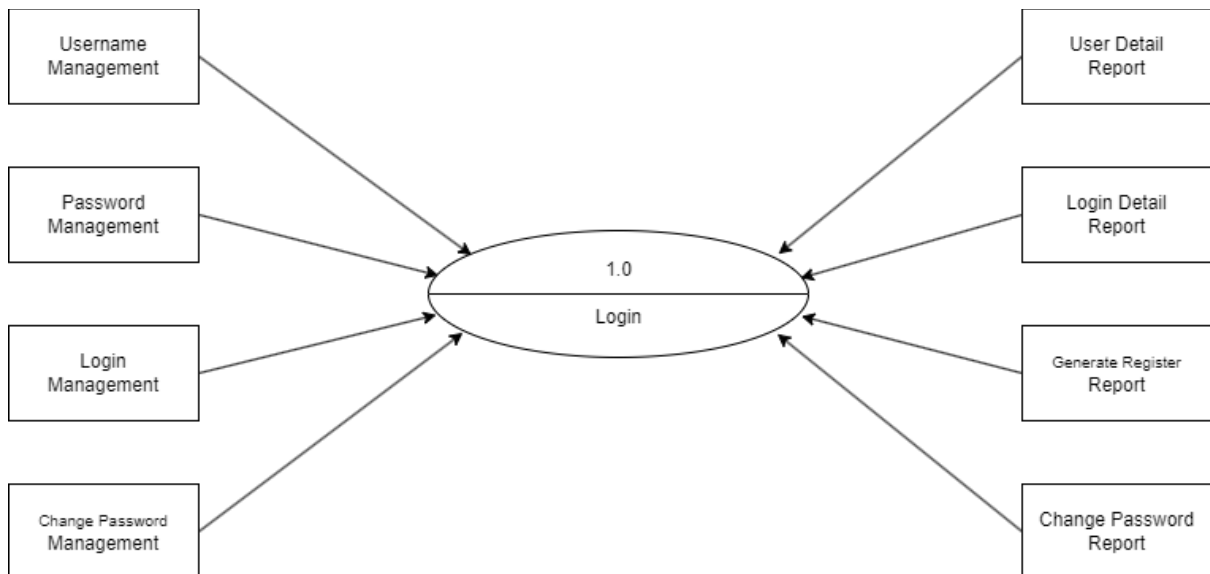
DIAGRAM	DESCRIPTION
	Represents Source or destination of data
	Represents a process that transforms Incoming data into Outgoing flows
	Represents data flow
	Represents data stores

3.3.1 Levels Of DFD

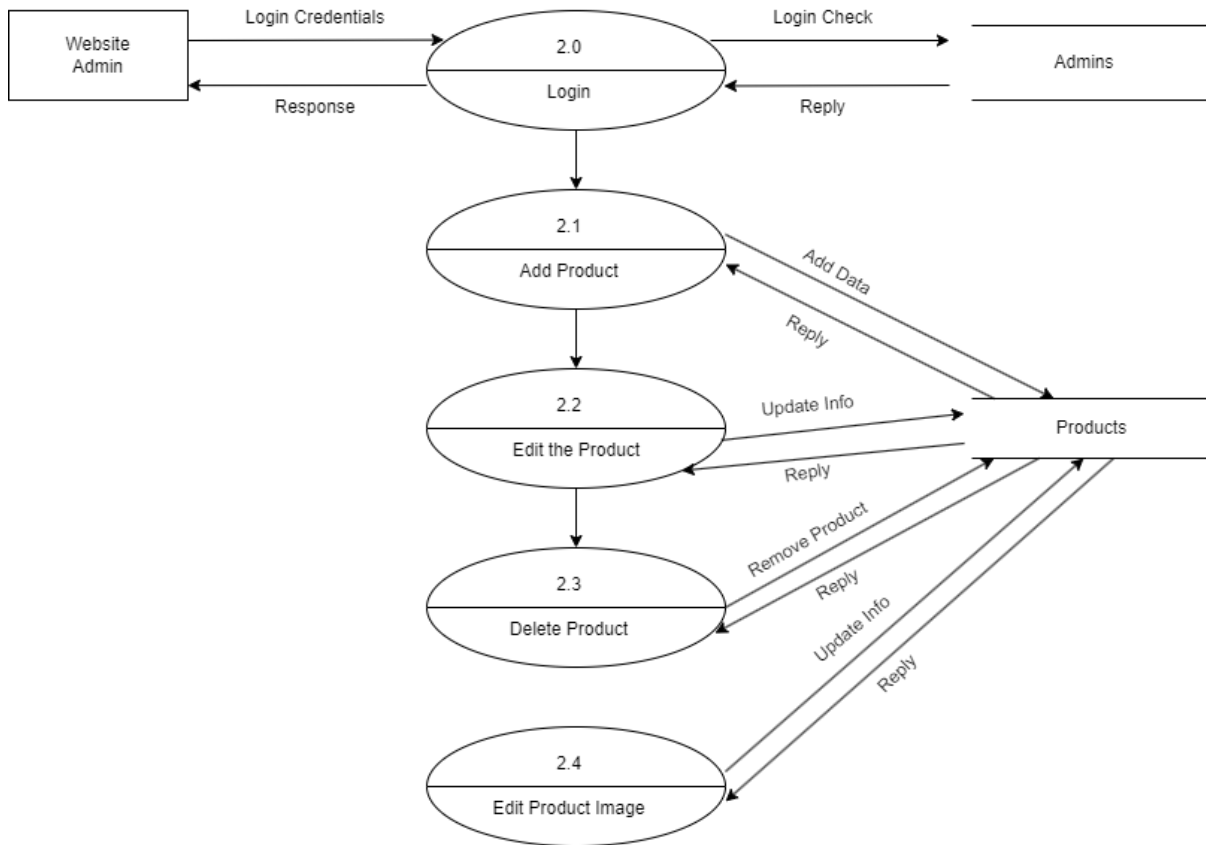
0th Level Admin Side DFD



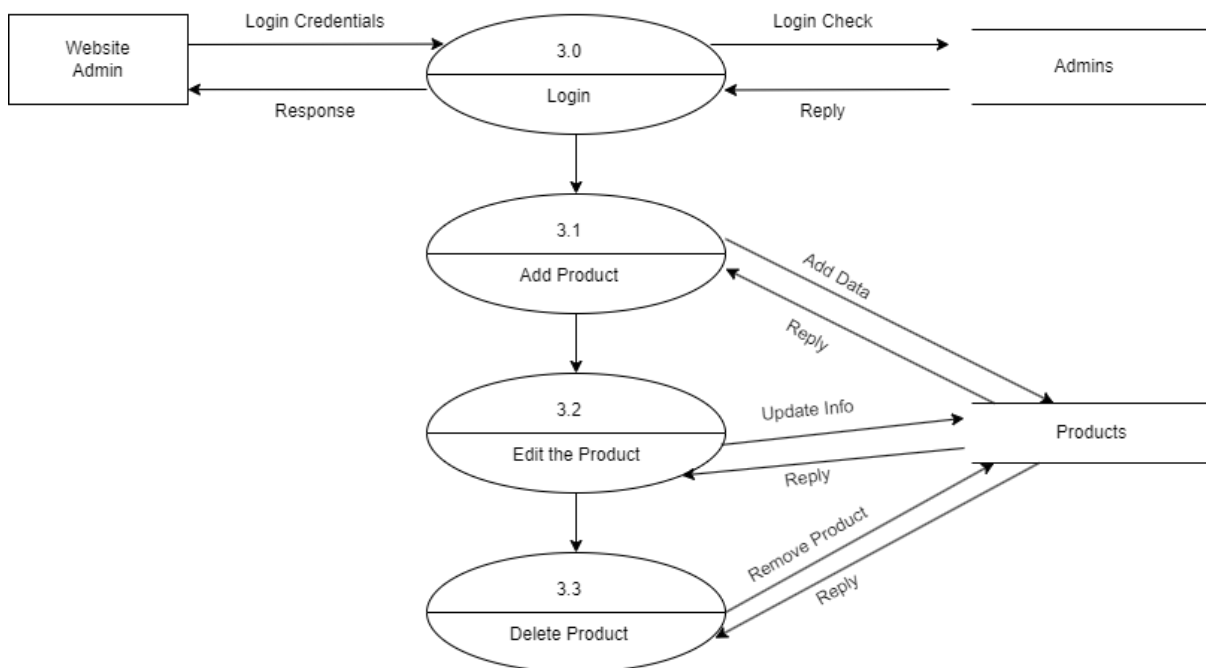
1st Level Admin Side DFD (1.0)



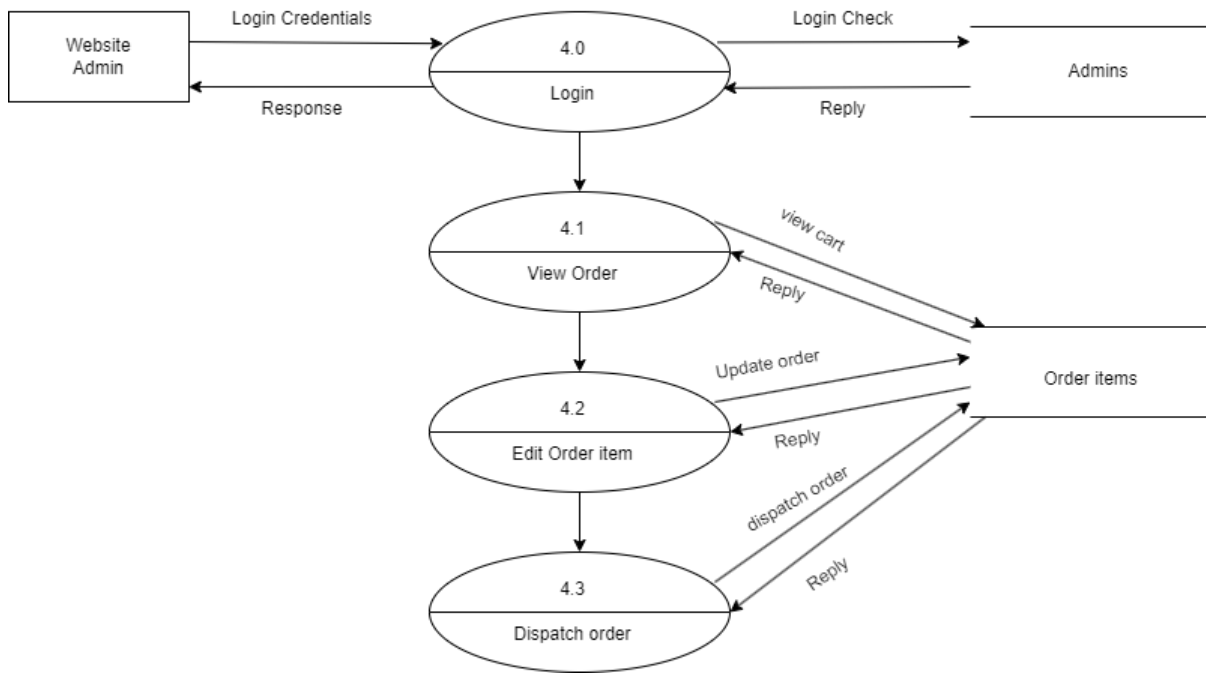
1st Level Admin Side DFD Manage Products and Edit Image (2.0)



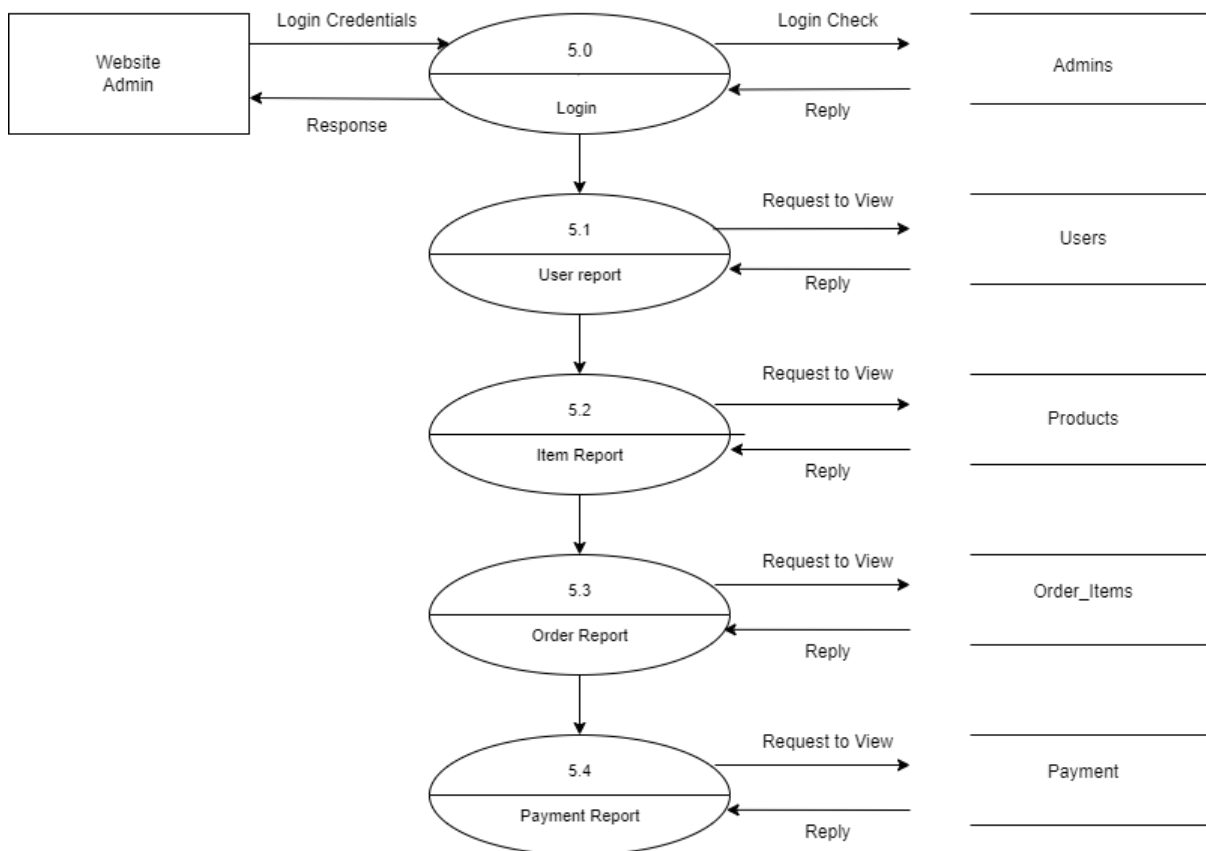
1st Level Admin Side DFD Manage Products (3.0)



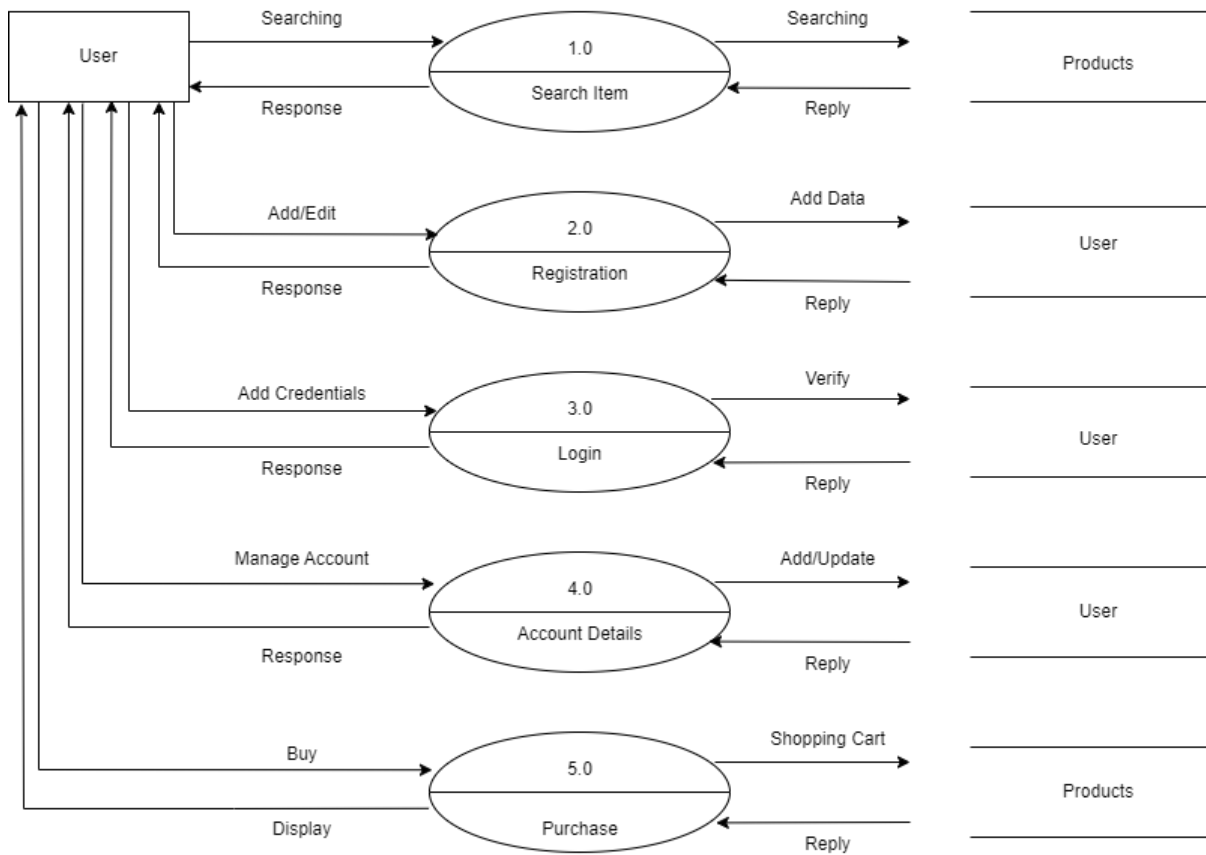
1st Level Admin Side DFD Manage Order (4.0)



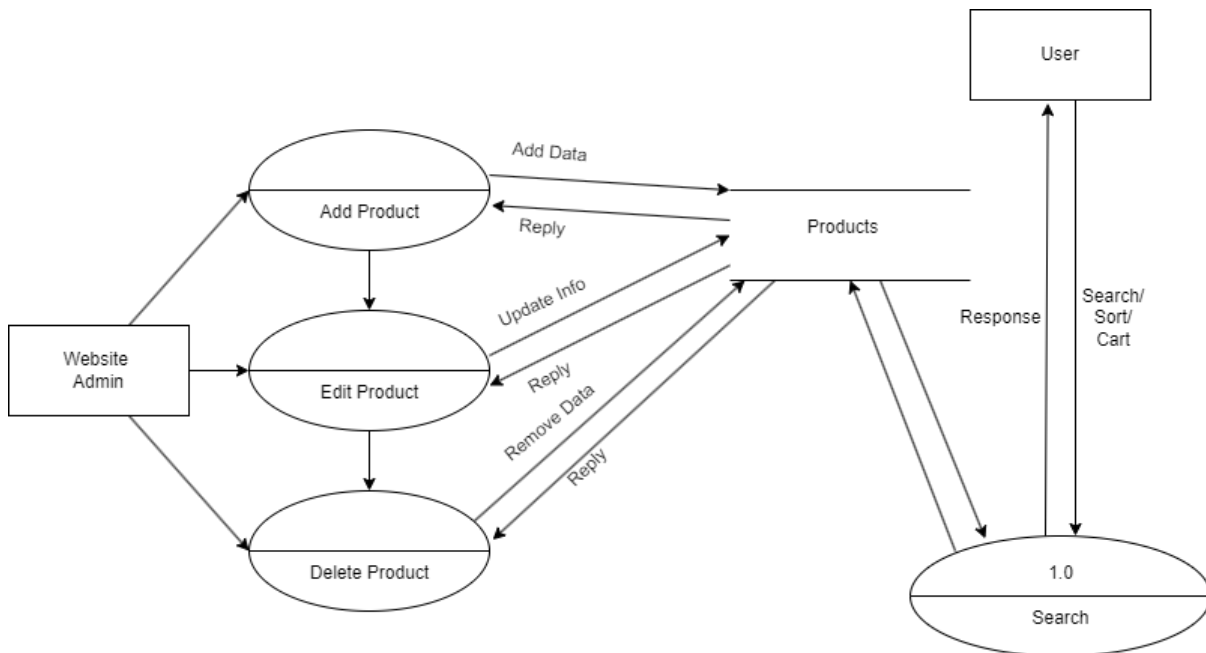
1st Level Admin Side DFD Manage Report (5.0)



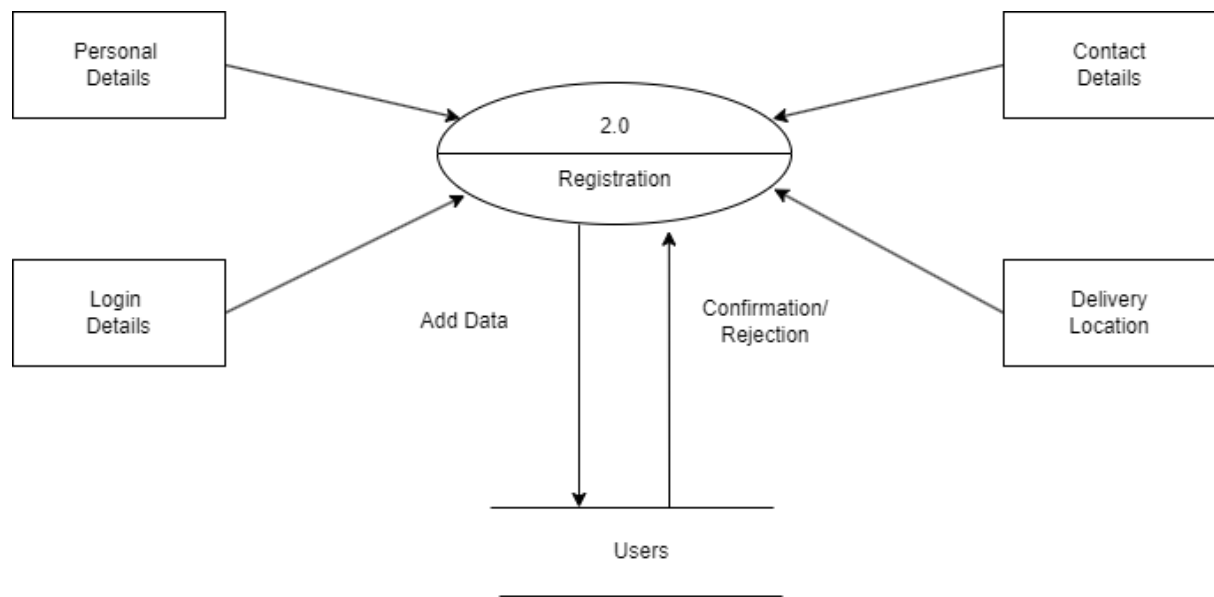
0th Level User side Data flow Diagram



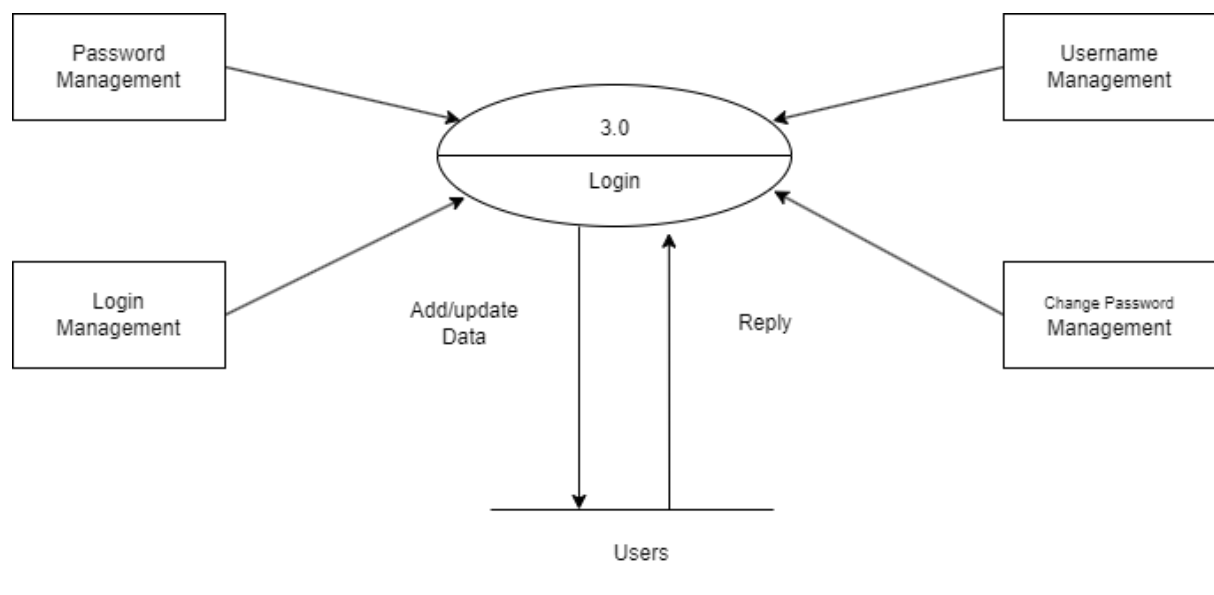
1st Level User side DFD Search (1.0)



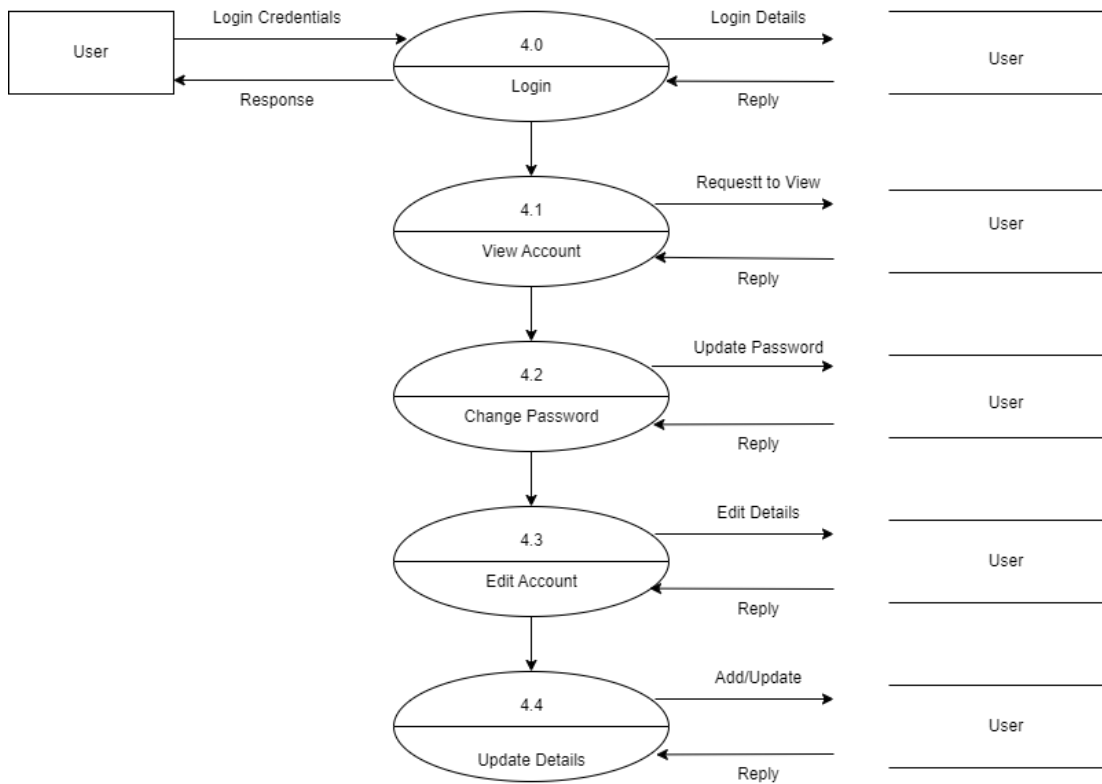
1st Level User side DFD Registration (2.0)



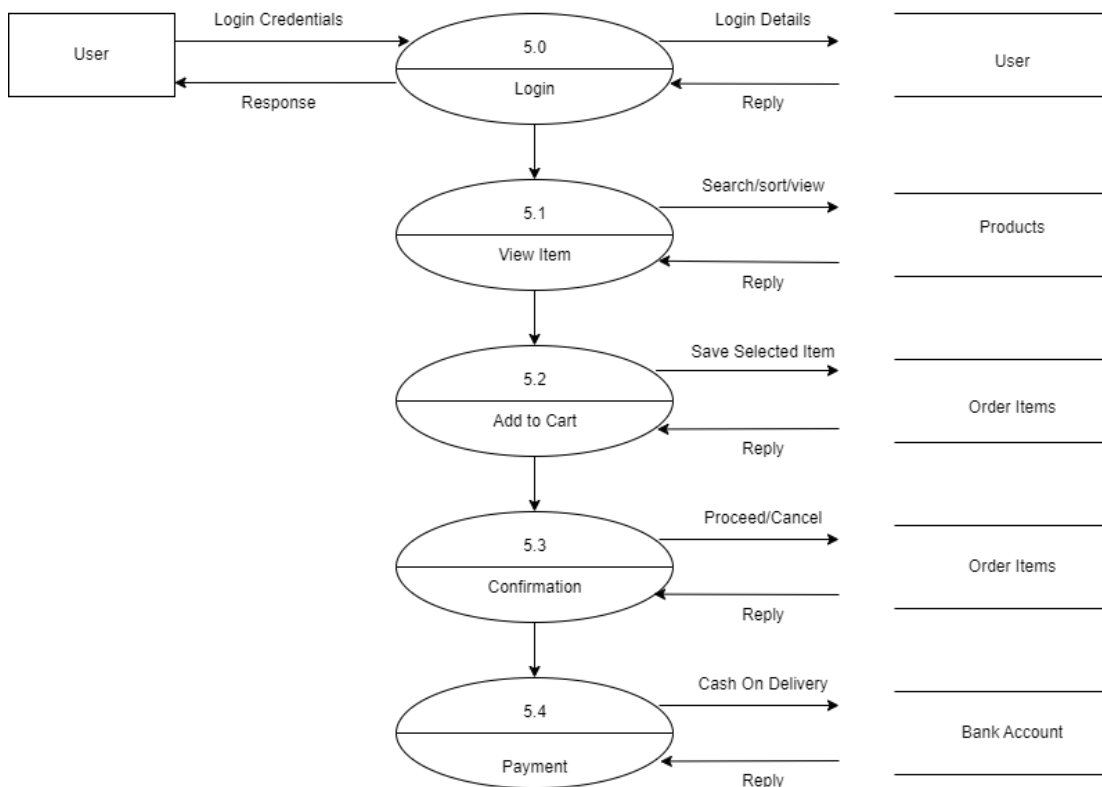
1st Level User side DFD Login (3.0)



1st level – User side DFD Account Details (4.0)



1st level – User side DFD Account Purchase (5.0)



3.4 Entity Relationship Diagram:

An entity – relationship (ER) diagram is a specialized graphic that illustrates the interrelationships between entities in a database. ER diagrams often use symbols to represent three different types of information. Boxes are commonly used to represent entities. Diamonds are normally used to represent relationships and ovals are used to represent attributes.

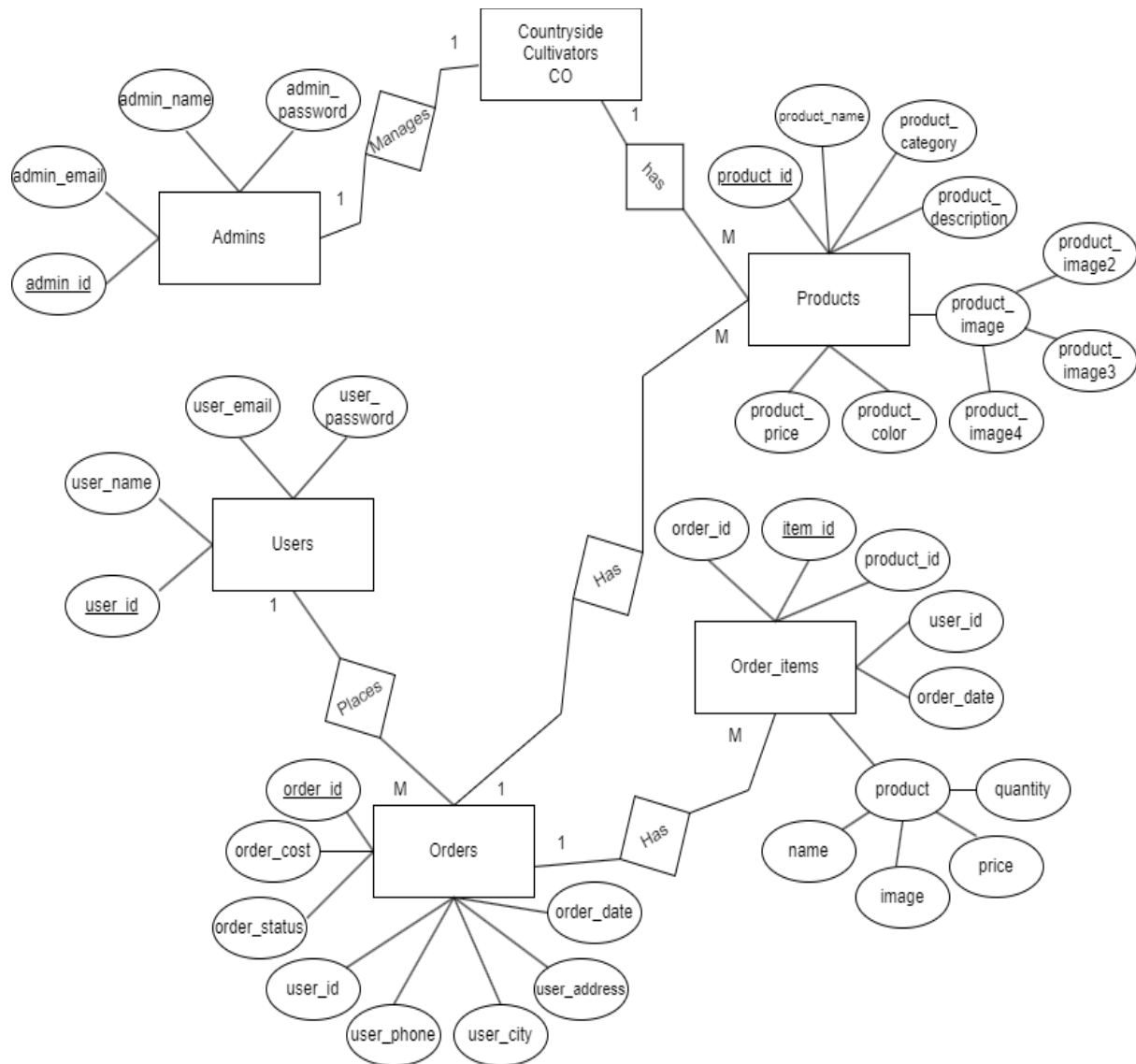
An entity-relationship model (ERM) in software engineering is an abstract and conceptual representation of data. Entity–relationship modelling is a relational schema database modelling method, used to produce a type of conceptual schema or semantic data model of a system, often a relational database, and its requirements in a top-down fashion.

An entity may be defined as a thing which is recognized as being capable of an independent existence and which can be uniquely identified. An entity is an abstraction from the complexities of some domain. When we speak of an entity, we normally speak of some aspect of the real world which can be distinguished from other aspects of the real world.

An entity may be a physical object such as a house or a car, an event such as a house sale or car service, or a concept such as a customer transaction or order. Although the term entity is the one most commonly used, following Chen we should really distinguish between an entity and an entity type. There are usually many instances of an entity type. Because the term entity type is somewhat cumbersome, most people tend to use the term entity as a synonym for this term.

A relationship captures how two or more entities are related to one another. Relationships can be thought of as verbs, linking two or more nouns. Examples: an owns relationship between a company and a computer, a supervises relationship between an employee and a department, a performs relationship between an artist and a song, a proved relationship between a mathematician and a theorem. Relationships are represented as diamonds, connected by lines to each of the entities in the relationship.

Entity Relationship Diagram:



4. DATABASE DESIGN

4.1 Introduction:

A Database is a collection of related data, which can be of any size and complexity, by using the concept of Database, we can easily store and retrieve data. The major purpose of a database is to provide the information, which utilizes it with the information that the system needs according to its own requirements. Database design is done before building it to meet needs of end users within a given Information system that a database intended to support to stop the database defines the needed data and data structures that such a database comprises Manila major physical implemented using SQL server.

Along with documenting your database design from the most important design wall you should have his to create a table structure theories table has a primary key and foreign key primary key should meet the following goals.

- Each record is unique within a table and no other record within the table has all of its columns equal to any other.
- For record to be unique all the columns are necessary that is data in one column should not be repeated anywhere else in the table regarding the second goal the column that has completely unique data throughout the table is known as a primarykey field a foreign key field is a field that links one table to another.

4.2 Normalized Tables:

Admins Table: Displays all the admins who controls the website.

Attribute	Type	Length	Constraint	Description
admin_id	Int	11	Primary key	Admin's id
admin_email	Text	255	Not null	Admin's email
admin_name	Text	255	Not null	Admin's name
admin_password	Text	255	Not null	Admin's password

Users Table: Displays all the users who have registered to the website.

Attribute	Type	Length	Constraint	Description
user_id	Int	11	Primary key	user's id
user_email	Text	255	Not null	user's email
user_name	Text	255	Not null	user's name
user_password	Text	255	Not null	user's password

Orders Table: Displays all the orders placed by users.

Attribute	Type	Length	Constraint	Description
Order_id	Int	11	Primary key	Orders id
order_cost	Decimal	8,2	Not null	Cost of order
order_status	Varchar	255	Not null	Status of order
user_id	Varchar	255	Not null	User's id
user_phone	Varchar	255	Not null	User's phone
user_city	Varchar	255	Not null	User's city
user_address	Varchar	255	Not null	User's address
order_date	Date and time		Not null	Date of ordering

Order_items Table: Displays the type of product ordered by user.

Attribute	TYPE	LENGTH	CONSTRAINT	DESCRIPTION
item_id	Int	11	Primary key	Items id
order_id	Int	11	Not null	Orders id
product_id	Int	11	Not null	Products id
product_name	varchar	255	Not null	Name of product
product_image	text	255	Not null	Product image
product_price	varchar	255	Not null	Price of product
product_quantity	Varchar	255	Not null	Product quantity
user_id	Int	11	Not null	User's id
order_date	Date and time		Not null	Date of ordering

Products Table: Displays all the products present in the website.

Attribute	Type	Length	Constraint	Description
product_id	Int	11	Primary key	Products id
product_name	Varchar	255	Not null	Name of product
product_category	Varchar	255	Not null	Product category
product_description	Varchar	255	Not null	Product description
product_image	Text	255	Not null	Product image
product_image2	Text	255	Not null	Product image 2
product_image3	Text	255	Not null	Product image 3
product_image4	Text	255	Not null	Product image 4
product_price	Decimal	8,2	Not null	Product price
product_color	Varchar	255	Not null	Colour of product

5. DETAILED DESIGN

5.1 Introduction:

This document is the design report for countryside cultivators co. This is mainly about ‘How to do’ and will help provide an insight to the whole system design and implementation of the e-commerce website.

Detailed design is the second level of the design process. During detailed design, we specify how the module in the system interacts with each other and the internal logic of each of the modules specified during system design is decided, hence it is also called as logic design.

Detailed design essentially expands the system design and database design to contain a more detailed description of the processing logic and data structures so that the design is sufficiently complete for coding.

5.2 Application Documents:

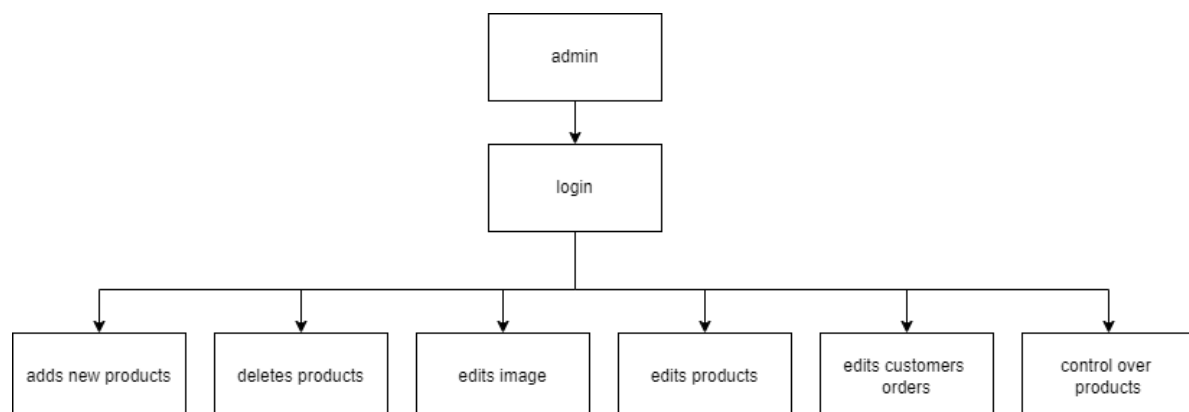
- Synopsis document for “CC.CO (Countryside cultivators co)”.
- Software Requirement Specification for “CC.CO (Countryside cultivators co)”.
- Database Design document for “CC.CO (Countryside cultivators co)”.

5.3 Structure of the software program:

The functional components are:

- Admin module
- Client module

Structure chart for Admin:



Modules of Admin:

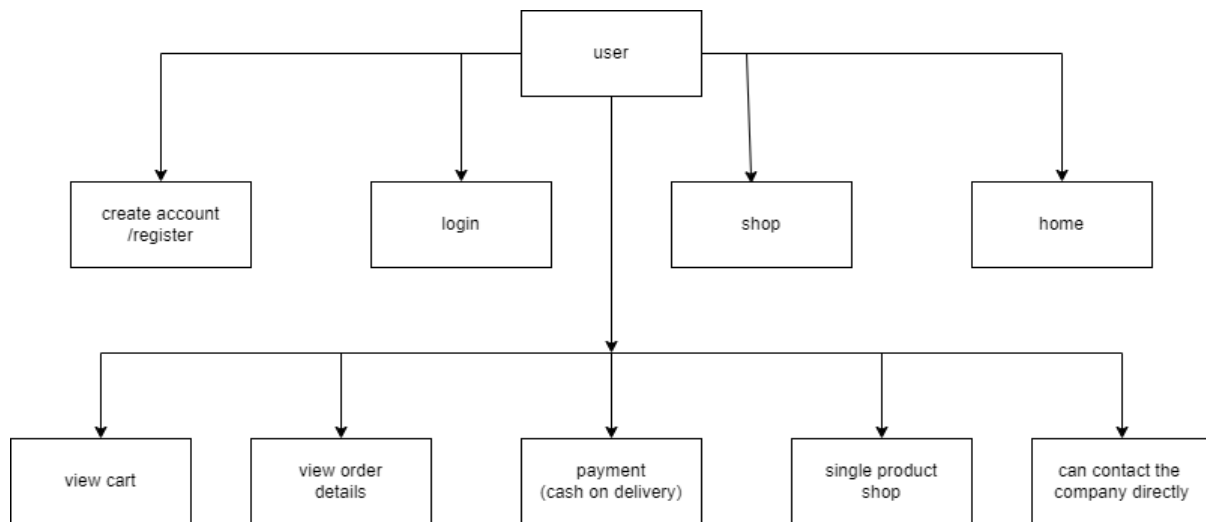
- Login
- Account
- Add product
- Edit images
- Edit orders
- Edit product
- Help
- Dashboard
- Products
- Orders

Module Description of Components

Description of the component:

- Login: The user needs to login to the system with the username and password which admin logs into the application and is headed to the home page.
- Account: Gives the admin details about the person using it with admin email, id and sl.no.
- Add product: Allows the admin to create new products.
- Edit images: Allows admin to edit the images.
- Edit orders: Allows admin to edit any orders placed by customer.
- Edit product: Admin can edit any product in the website.
- Help: Shows the email and number of admin for help.
- Dashboard: It is the main index of the admin side.
- Products: The admin can change, edit and delete any products and can add them also.
- Orders: Used to track, edit and delete orders.

Structured chart of users:



Modules of users:

- Login
- Register
- Account
- Cart
- Shop
- Home
- Single product
- Contact info
- Checkout
- Payment

- Login: The user can login after he registers to the company.
- Register: If user does not Register with details, he will not be able to log in and purchase products in the website.
- Account: The user can view his account details in the account page.
- Cart: The user can add items to the cart and can check out.
- Shop: The user can go through the shop and add items to the cart.
- Home: It is the index that displays products and equipment etc.
- Single Products: Once the user decides to purchase the product, he will be redirected to the single product page where he can add items to the cart.
- Contact: The companies contacts are present in this page.
- Checkout: After Selecting products and user wishes to purchase then he must checkout with his user login details.

6. CODING

6.1 Introduction:

The goal of coding or programming phase is to translate the design of the system produced during the design phase into code in a given programming language, which can be executed by a computer and that performs the computation specified by the design. The coding phase affects both testing and maintenance profoundly. The goal during this phase is not to simplify the job of the programmer. Rather, the goal should be simplifying the job of the tester and the maintainer.

There are many different criteria for judging a program, including readability, size of the program, execution time and required memory. Having readability and understanding as a clear objective of the coding activity can itself help in producing software that is more maintainability.

6.2 Programming Style:

It is impossible to provide an exhaustive list of what to do and what not to do produce simple readable code. Being able to do this will amount to providing an algorithm for writing good code. next we will list some general rules that usually apply.

Names:

Selecting module and variable names often not considered important by novice programmers. Only when one starts reading programs written by other where the variable names are cryptic and not representative does one realize the importance of selecting proper names. Most variables in a program reflect some entity in the problem domain, and the modules reflect some process. Variable names should be closely related to the entity they represent, and module name should reflect their activity. It is bad practice to choose cryptic names just to avoid typing or totally unrelated names. It is also bad practice to use the same name for multiple purposes.

Control Constructs:

It is desirable that as much as possible single-entry, single exit constructs be used. It is also desirable to use the few standard control constructs rather than using a wide variety of constructs, just because they are available in the language.

Information Hiding:

Information hiding should be supported where possible. Only the access function for the datastructures should be made visible while hiding the data structure behind these functions.

User-Defined Types:

Modern languages allow users to define types like the Enumerated type. When such facilities are available, they should be exploited where applicable. For example, when working with dates, the type can be defined for the day of the week.

Nesting:

The different control constructs, the if-then-else, can be nested. If the nesting becomes too deep then the programs become harder to understand. In case of deeply nested if-then-else, it is often difficult to determine then if statement to which a particular else clause is associated. Deep nesting should be avoided, even if it means a little inefficiency.

6.3 Module Size:

We discussed this issue during system design. A programmer should carefully examine any routine with very few statements (say fewer than 5) or with too many statements (say more than 50). Large modules often will not be functionally cohesive and too small modules might incur unnecessary overhead. There is can be no hard-and-fast rule about module sizes the guiding principle should cohesion and coupling.

Module Interface:

A module with a complex interface should be carefully examined. Such modules might not be functionally cohesive and might be implementing multiple functions. As a rule of thumb, any module whose interface has more than five parameters should be carefully examined and broken into multiple modules with simpler interface if possible.

Program Layout:

How the program is organized and presented can have great effect on the readability of it. Proper indentation, blank spaces, and parentheses should be used to enhance the readability of programs. Automated tools are available to “pretty print” a program, but it is good practice to have a clear layout of programs.

Side Effects:

When a module is invoked, it sometimes has side effect of modifying the program state beyond the modifications of parameters listed in the module interface definition, for example, modifying global variables. Such side effects should be avoided where possible, and if a module has side effects, they should be properly documented.

Robustness:

A program is robust if it does something planned even for exceptional conditions. A program might encounter exceptional conditions in such forms as incorrect input, the incorrect value of some variable, and overflow. In general, a program should check for validity inputs, were possible, and should check for possible overflow of the data structures. If such situations do arise, the program should not just “crash” it should produce some meaningful message and exit gracefully.

6.4 Source code:

User side:

index.php: It is the home page of the website.

```
<?php include('layouts/header.php');?>
<!--home-->
<section id="home">
  <div class="container">
    <h2> New Equipments </h2>
    <h1>BEST PRICES!</h1>
    <h3><p><span>Quality Products<br>For the most affordable prices!</span>
    </p></h3>
  </div>
</section>
<!--brand-->
<section id="brand" >
  <div class="row">
```

```

        
        
        
        
    </div>
</section>
<!--featured-->
<section id="featured" class="my-5 pb-5">
    <div class="container text-center mt-5 py-5">
        <h3>Our Featured Products</h3>
        <hr>
        <p>Here you can check out our new featured products</p>
    </div>
    <div class="row mx-auto container-fluid">

        <?php include('server/get_featured_products.php'); ?>
        <?php while($row = $featured_products->fetch_assoc()){ ?>

            <div class="product text-center col-lg-3 col-md-4 col-sm-12">
                
                <h5 class="p-name"><?php echo $row['product_name']; ?></h5>
                <h4 class="p-price">Rs.<?php echo $row['product_price'];?></h4>
                <a href="<?php echo "single_product.php?product_id=".$row['product_id'];?>">
                <button class="btn buy-btn">buy now</button></a>
            </div>
            <?php } ?>
        </div>
    </section>

<!--banner-->
<section id="banner" class="my-5 py-5">
    <div class="container">
        <h4 class="text-center">Affordable Farming Equipments</h4>
        <hr class="mx-auto">
        <h1 class="text-center">We have a wide range of Equipments and Other
        Products</h1>
    </div>
</section>

<!--equipment-->
<section id="equipment" class="my-5">
    <div class="container text-center mt-5 py-5">
        <h3>EQUIPMENT</h3>
        <hr>
        <p>Here you can check out our Farming Equipments</p>
    </div>

```

```

<div class="row mx-auto container-fluid">

    <?php include('server/get_equipment.php');?>
    <?php while($row = $equipment->fetch_assoc()){ ?>

        <div class="product text-center col-lg-3 col-md-4 col-sm-12">
            
            <h5 class="p-name"><?php echo $row['product_name'];?></h5>
            <h4 class="p-price">Rs.<?php echo $row['product_price'];?></h4>
            <a href="<?php echo "single_product.php?product_id=" . $row['product_id'];?>">
            . $row['product_id'];?>"> <button class="buy-btn">Buy Now </button></a>
        </div>
    <?php } ?>
</div>
</section>
<!--fittings-->
<section id="fittings" class="my-5">
    <div class="container text-center mt-5 py-5">
        <h3>FITTINGS</h3>
        <hr>
        <p>Here you can check out some Fittings</p>
    </div>
    <div class="row mx-auto container-fluid">

        <?php include('server/get_fittings.php');?>
        <?php while($row=$fittings->fetch_assoc()){ ?>

            <div class="product text-center col-lg-3 col-md-4 col-sm-12">
                
                <h5 class="p-name"><?php echo $row['product_name'];?></h5>
                <h4 class="p-price">Rs.<?php echo $row['product_price'];?></h4>
                <a href="<?php echo "single_product.php?product_
                id=" . $row['product_id'];?>">
                <button class="buy-btn">Buy Now</button></a>
            </div>
        <?php } ?>
    </div>
</section>

<!--tools-->
<section id="tools" class="my-5">
    <div class="container text-center mt-5 py-5">
        <h3>TOOLS</h3>
        <hr>
        <p>Here you can check out some TOOLS</p>
    </div>

```



```

<div class="row mx-auto container-fluid">

    <?php include('server/get_tools.php');?>
    <?php while($row=$tools->fetch_assoc()){?>

        <div class="product text-center col-lg-3 col-md-4 col-sm-12">
            
            <h5 class="p-name"><?php echo $row['product_name'];?></h5>
            <h4 class="p-price">Rs.<?php echo $row['product_price'];?></h4>
            <a href="<?php echo "single_product.php?product_id=" .
            $row['product_id'];?>">
                <button class="buy-btn">buy now </button></a>
        </div>
    <?php }?>
</div>
</section>
<!--other-->
<section id="other" class="my-5">
    <div class="container text-center mt-5 py-5">
        <h3>OTHER PRODUCTS</h3>
        <hr>
        <p>Here you can check out some Other Products</p>
    </div>
    <div class="row mx-auto container-fluid">
        <?php include('server/get_other.php');?>
        <?php while($row=$other->fetch_assoc()){?>
            <div class="product text-center col-lg-3 col-md-4 col-sm-12">
                
                <h5 class="p-name"><?php echo $row['product_name'];?></h5>
                <h4 class="p-price">Rs.<?php echo $row['product_price'];?></h4>
                <a href="<?php echo "single_product.php?product_id=" . $row['product_id'];?>">
                    <button class="buy-btn">Buy Now</button></a>
            </div>
        <?php }?>
    </div>
</section>
<?php include('layouts/footer.php');?>

```

login.php: Here the user can login to the website with his credentials.

```

<?php
session_start();
include('server/connection.php');

if(isset($_SESSION['logged_in'])){
    header("location:account.php");
    exit;
}

```

```

}
if(isset($_POST['login_btn'])){
$email = $_POST['email'];
$password = md5($_POST['password']);

$stmt = $conn->prepare("SELECT user_id,user_name,user_email,user_password FROM
users WHERE user_email=? AND user_password=? LIMIT 1");
$stmt->bind_param('ss',$email,$password);

if($stmt->execute()){
$stmt->bind_result($user_id,$user_name,$user_email,$user_password);
$stmt->store_result();

if($stmt->num_rows() == 1){
$stmt->fetch();
    $_SESSION['user_id'] = $user_id;
    $_SESSION['user_name'] = $user_name;
    $_SESSION['user_email'] = $user_email;
    $_SESSION['logged_in'] = true;

    header('location: account.php?message=logged in successfully');
}else{
    header('location: login.php?error=could not verify your account');
}
}else{
//error
    header('location: login.php?error=something went wrong');
}
}
?>

```

```

<?php include('layouts/header.php');?>
<!--login-->
<section class="my-5 py-5">
    <div class="container text-center mt-3 pt-5">
        <h2 class="form-weight-bold">LOGIN</h2>
        <hr class="mx-auto">
    </div>
    <div class="mx-auto container">
        <form id="login-form" method="POST" action="login.php" >
            <p style="color:red" class="text-center">
                <?php if(isset($_GET['error'])){ echo $_GET['error'] ;}?></p>
            <div class="form-group">
                <label>Email</label>
                <input type="text" class="form-control" id="login-email" name="email"
                placeholder="Enter Your Email" required/>
            </div>
            <div class="form-group">

```

```

        <label>Password</label>
        <input type="password" class="form-control" id="login-password"
            name="password" placeholder="Enter Your Password" required/>
    </div>
    <div class="form-group">
        <input type="submit" class="btn" id="login-btn" name="login_btn"
            value="Login"/>
    </div>
    <div class="form-group">
        <a id="register-url" href="register.php" class="btn">Dont have an account?
            Register</a>
    </div>
</form>
</div>
</section>
<?php include('layouts/footer.php');?>

```

account.php: The user can view his account information and the orders.

```

<?php
session_start();
include('server/connection.php');

if(!isset($_SESSION['logged_in'])){
    header("location:login.php");
    exit;
}
if(isset($_GET['logout'])){
    if(isset($_SESSION['logged_in'])){
        unset($_SESSION['logged_in']);
        unset($_SESSION['user_email']);
        unset($_SESSION['user_name']);
        header("loaction: login.php");
        exit;
    }
}
if(isset($_POST['change_password'])){
    $password = $_POST['password'];
    $confirmpassword = $_POST['confirmpassword'];
    $user_email = $_SESSION['user_email'];
    //if passwors dont match

    if($password !== $confirmpassword){
        header("location: account.php?error=passwords do not match");

        //if passwords less than 6 characters
    }else if(strlen($password) < 6 ){
        header("location: account.php?error=password must be at least 6 characters");
    }else{

```

```

$stmt = $conn->prepare("UPDATE users SET user_password=?Where user_email=?");
$stmt->bind_param('ss',md5($password),$user_email);

if($stmt->execute()){
    header("location: account.php?message=password has been updated successfully");
}else{
    header("location:account.php?error=could not update password");
}
}
}

//get orders
if(isset($_SESSION['logged_in'])){
    $user_id = $_SESSION['user_id'];
    $stmt = $conn->prepare("SELECT * FROM orders WHERE user_id=? ");
    $stmt->bind_param('i',$user_id);
    $stmt->execute();
    $orders = $stmt->get_result();//array
}
?>

<?php include('layouts/header.php');?>
<!--account-->
<section class="my-5 py-5">
    <div class="row container mx-auto">
        <div class="text-center mt-3 pt-5 col-lg-6 col-md-12 col-sm-12">
            <h3 class="font-weight-bold">ACCOUNT INFO</h3>
            <hr class="mx-auto">
            <div class="account-info">
                <p>User ID : <span><?php if(isset($_SESSION['user_id'])){echo
                $_SESSION['user_id'];}?></span></p>

                <p>User Name : <span><?php if(isset($_SESSION['user_name'])){
                echo $_SESSION['user_name'];}?></span></p>

                <p>User Email : <span><?php if(isset($_SESSION['user_email'])){
                echo $_SESSION['user_email'];}?></span></p>

                <p><a href="#orders" id="orders-btn">Your Orders</a></p>
                <p><a href="account.php?logout=1" type="submit" class="btn"
                id="logout-btn">Logout</a></p>
            </div>
        </div>
    </div>
    <div class="col-lg-6 col-md-12 col-sm-12">
        <form id="account-form" method="POST" action="account.php">
            <p class="text-center" style="color:red"><?php if(isset($_GET['error'])){echo
            $_GET['error'];}?></p>

            <p class="text-center" style="color:green"><?php if(isset($_GET['message'])){

```

```

        echo $_GET['message']; }?></p>
        <h3>Change Password</h3>
        <hr class="mx-auto">
        <div class="form-group">
            <label>Password</label>
            <input type="password" class="form-control" id="account-password"
                name="password" placeholder="Password" required/>
        </div>
        <div class="form-group">
            <label>Confirm Password</label>
            <input type="password" class="form-control" id="account -confirm-
                password" name="confirmpassword" placeholder="Confirm Password"
                required/>
        </div>
        <div class="form-group">
            <input type="submit" value="change password" name="change_password"
                class="btn" id="change-pass-btn">
        </div>
    </form>
</div>
</div>
</section>

<!--orders-->
<section id="orders" class="orders container my-5 py-3">
    <div class="container mt-2">
        <h2 class="font-weight-bold text-center">YOUR ORDERS</h2>
        <hr class="mx-auto">
    </div>
    <table class="mt-5 pt-5">
        <tr>
            <th>Order ID</th>
            <th>Order Cost</th>
            <th>Order Status</th>
            <th>Order Date and Time</th>
            <th>Order Details</th>
        </tr>
        <?php while($row = $orders->fetch_assoc() ){ ?>
            <tr>
                <td>
                    <span><?php echo $row['order_id'];?></span>
                </td>
                <td>
                    <span><?php echo $row['order_cost'];?></span>
                </td>
                <td>
                    <span><?php echo $row['order_status'];?></span>
                </td>
            </tr>
        </tr>
    </table>

```

```

        <td>
            <span><?php echo $row['order_date'];?></span>
        </td>
    </td>

    <form method="POST" action="order_details.php"><input type="hidden"
        value="<?php echo $row['order_status'];?>" name="order_status">

    <input type="hidden" value="<?php echo $row['order_id'];?>"
        name="order_id">

        <input class="btn order-details-btn" name="order_details_btn"
            type="submit" value="details">
    </form>
    </td>
</tr>
<?php }?>
</table>
</section>
<?php include('layouts/footer.php');?>

```

cart.php: The user can view his cart items here.

```

<?php
session_start();

if(isset($_POST['add_to_cart'])){
    //if user has already added product to cart
    if(isset($_SESSION['cart'])){
        //array will return product id
        $products_array_ids = array_column($_SESSION['cart'], "product_id");

        //if product has been added or not to the cart
        if(!in_array($_POST['product_id'], $products_array_ids) ){

            $product_id = $_POST['product_id'];
            $product_array = array(
                'product_id' => $_POST['product_id'],
                'product_name' => $_POST['product_name'],
                'product_price' => $_POST['product_price'],
                'product_image' => $_POST['product_image'],
                'product_quantity' => $_POST['product_quantity']
            );

            //product id acts as unique id for product array which has array stored inside array. eg-
            > [2=>[],3=>[],5=>[]]

            $_SESSION['cart'][$product_id] = $product_array;
            //product has already been added
        }else{

```

```

        echo '<script>alert("product was already added to cart"); </script>';
    }

    //if this is the first product
} else {
    $product_id = $_POST['product_id'];
    $product_name = $_POST['product_name'];
    $product_price = $_POST['product_price'];
    $product_image = $_POST['product_image'];
    $product_quantity = $_POST['product_quantity'];

    $product_array = array(
        'product_id' => $product_id,
        'product_name' => $product_name,
        'product_price' => $product_price,
        'product_image' => $product_image,
        'product_quantity' => $product_quantity
    );

    //product id acts as unique id for product array which has array stored inside array. eg-
    > [2=>[], 3=>[], 5=>[]]

    $_SESSION['cart'][$product_id] = $product_array;
}

    //calculate total
    calculatetotalcart();

//remove from cart
} else if (isset($_POST['remove_product'])) {
    $product_id = $_POST['product_id'];
    unset($_SESSION['cart'][$product_id]);

    //calculate total
    calculatetotalcart();

} else if (isset($_POST['edit_quantity'])) {
    //we get id and quantity from the form
    $product_id = $_POST['product_id'];

    $product_quantity = $_POST['product_quantity'];
    //get product array from session
    $product_array = $_SESSION['cart'][$product_id];
    //update product quantity
    $product_array['product_quantity'] = $product_quantity;
    //return the array back
    $_SESSION['cart'][$product_id] = $product_array;
    //calculate total
    calculatetotalcart();
} else {

```

```

<td>
    <form method="POST" action="cart.php">
        <input type="hidden" name="product_id" value="<?php echo
        $value['product_id'];?>">

        <input type="number" name="product_quantity" value="<?php echo
        $value['product_quantity'];?>">

        <input type="submit" class="btn edit-btn" value="edit" name="edit_quantity"/>
    </form>
</td>
<td>
    <span></span>
    <span class="product-price">Rs.<?php echo $value['product_quantity'] *
    $value['product_price'];?></span>
</td>
</tr>
<?php }?>
<?php }?>
</table>
<div class="cart-total">
<table>
<tr>
<td>Total</td>
<?php if(isset($_SESSION['cart'])){ ?>
<td>Rs.<?php echo $_SESSION['total'];?>
</td>
<?php }?>
</tr>
</table>
</div>
<div class="checkout-container">
    <form method="POST" action="checkout.php">
        <input type="submit" class="btn checkout-btn" value="checkout"
        name="checkout">
    </form>
</div>
</section>
<?php include('layouts/footer.php'); ?>

```

checkout.php: The user can checkout using his account credentials.

```

<?php
session_start();

if(!empty($_SESSION['cart'])){
    //let user in or
    //redirect to home page
}else{

```

```

        header('location:index.php');
    }
?>

<?php include('layouts/header.php');?>
<!--checkout-->
<section class="my-5 py-5">
    <div class="container text-center mt-3 pt-5">
        <h2 class="form-wiight-bold">CHECK OUT</h2>
        <hr class="mx-auto">
    </div>
    <div class="mx-auto container">
        <form id="checkout-form" method="POST" action="server/place_order.php">

            <p class="text-center" style="color:red;">
                <?php if(isset($_GET['message'])) { echo $_GET['message']; }?>

                <?php if(isset($_GET['message'])) { ?>
                    <a href="login.php" class="btn btn-primary">Login</a>
                <?php }?>
            </p>
            <div class="form-group checkout-small-element">
                <label>Name</label>
                <input type="text" class="form-control" id="checkout-name" name="name"
                placeholder="Enter Name" required/>
            </div>
            <div class="form-group checkout-small-element">
                <label>Email</label>
                <input type="text" class="form-control" id="checkout-email" name="email"
                placeholder="Enter Email Address" required/>
            </div>
            <div class="form-group checkout-small-element">
                <label>Phone</label>
                <input type="tel" class="form-control" id="checkout-phone" name="phone"
                placeholder="Enter Phone Number" required/>
            </div>
            <div class="form-group checkout-small-element">
                <label>City Name</label>
                <input type="text" class="form-control" id="checkout-city" name="city"
                placeholder="Enter City Name" required/>
            </div>
            <div class="form-group checkout-large-element">
                <label>Shipping Address</label>
                <input type="text" class="form-control" id="checkout-address"
                name="address" placeholder="Enter Shipping Address" required/>
            </div>
            <div class="form-group checkout-btn-container">
                <p>Total Amount:Rs<?php echo $_SESSION['total'];?></p>

```

```

        <input type="submit" class="btn" id="checkout-btn" name="place_order"
        value="place order"/>
    </div>
</form>
</div>
</section>

<?php include('layouts/footer.php');?>

```

complete_payment.php: The user can complete the payment of products.

```

<?php
session_start();

include('server/connection.php');
//change order_status to paid
if(isset($_GET['order_id']) && isset($_GET['user_id'])){

    $order_id = $_GET['order_id'];
    $order_status = "paid";
    $user_id = $_SESSION['user_id'];
    //change order_status to paid
    $stmt = $conn->prepare("UPDATE orders SET order_status=? Where
        order_id=?");
    $stmt->bind_param('si',$order_status,$order_id);
    $stmt->execute();
    $stmt->store_result();
    //store payment info in database
    $stmt1 = $conn->prepare("INSERT INTO payments (order_id,user_id)
    VALUES (?,?);");

    $stmt1->bind_param('ii',$order_id,$user_id);
    $stmt1->execute();
    $stmt1->store_result();

    //go to user account
    header("location: ../account.php?payment_message=paid successfully,thanks for
    your purchase");
} else{
    header("location:index.php");
    exit;
}
?>

<div class="mx-auto container text-left">
    <form method = "POST" >
        <h1>Thank you for your Order</h1>
        <hr>
        <h1>Order Placed Successfully</h1>

```

```

        <h2>You will recive your order within 10Days</h2>
        <h2>If you have any Questions or want to Cancel the Order</h2>
        <h2>Please Contact us either by our Email or Phone Number</h2>
    </form>
</div>
</div>
</section>
<?php include('layouts/footer.php');?>

```

contact.php: All the contact information of the website is available.

```

<?php include('layouts/header.php');?>
<!--contact-->
<section id="contact" class="container my-5 py-5">
    <div class="container text-center mt-5">
        <h3>CONTACT US</h3>
        <hr class="mx-auto">
        <p class="w-50 mx-auto">
            Phone Number: 8296568412
        </p>
        <p class="w-50 mx-auto">
            Email Address: ccco28@gmail.com
        </p>
        <p class="w-50 mx-auto">
            24x7 Service
        </p>
        <p class="w-50 mx-auto">
            We provide Cash on Delivery at the time.
        <br>
            Online Payment will be accepted soon.<br>Sorry for the Inconvenience.
        </p>
    </div>
</section>
<?php include('layouts/footer.php');?>

```

order_details.php: User can view the order details in account page.

```

<?php
    /*keywords
        not paid
        delivered
        shipped
    */
include('server/connection.php');

if(isset($_POST['order_details_btn']) && isset($_POST['order_id'])){
    $order_id = $_POST['order_id'];
    $order_status = $_POST['order_status'];
}

```

```

$stmt = $conn->prepare("SELECT * FROM order_items WHERE order_id=?");
    $stmt->bind_param('i',$order_id);
    $stmt->execute();
    $order_details = $stmt->get_result();
    $order_total_price = calculatetotalorderprice($order_details);
} else {
    header("location:account.php");
    exit;
}

function calculatetotalorderprice($order_details){
    $total = 0;

    foreach($order_details as $row){

        $product_price = $row['product_price'];
        $product_quantity = $row['product_quantity'];
        $total = $total + ($product_price * $product_quantity);
    }
    return $total;
}
?>

```

```

<?php include('layouts/header.php');?>
<!--order details-->
<section id="orders" class="orders container my-5 py-3">
    <div class="container mt-5">
        <h2 class="font-weight-bold text-center">ORDER DETAILS</h2>
        <hr class="mx-auto">
    </div>
    <table class="mt-5 pt-5 mx-auto">
        <tr>
            <th>Product</th>
            <th>Price</th>
            <th>Quantity</th>
        </tr>
        <?php foreach($order_details as $row){?>
            <tr>
                <td>
                    <div class="product-info">
                        
                        <div>
                            <p><?php echo $row['product_name'];?></p>
                        </div>
                    </div>
                </td>
                <td>
                    <span>Rs.<?php echo $row['product_price'];?></span>
                </td>
            </tr>
        </?php?>
    </table>

```

```

        <td>
            <span><?php echo$row['product_quantity'];?>.nos</span>
        </td>
    </tr>
<?php }?>
</table>
<?php if($order_status == "not paid"){ ?>

    <form style="float:right;" method="POST" action="payment.php">
        <input type="hidden" name="order_id" value="<?php echo $order_id;?>">

        <input type="hidden" name="order_total_price" value="<?php echo
        $order_total_price ;?>">

        <input type="hidden" name="order_status" value="<?php echo
        $order_status;?>">
        <br>
        <input type="submit" name="order_pay_btn" class="btn btn-primary"
        value="cash on delivery">
    </form>
<?php }?>
</section>
<?php include('layouts/footer.php');?>

```

payment.php: The user can confirm the payment.

```

<?php
session_start();
if(isset($_POST['order_pay_btn'])){
    $order_status = $_POST['order_status'];
    $order_total_price = $_POST['order_total_price'];
}
?>

<?php include('layouts/header.php');?>
<!--payment-->
<section class="my-5 py-5">
    <div class="container text-center mt-3 pt-5">
        <h2 class="form-weight-bold">PAYMENT</h2>
        <hr>
    </div>
<div class="mx-auto container text-center">
    <form method="POST" action="complete_payment.php">
        <?php if(isset($_SESSION['total']) && $_SESSION['total'] != 0){?>
        <?php $amount = strval($_SESSION['total']);?>
        <p>Total Payment:Rs.<?php echo $_SESSION['total'];?></p>
        <p>Cash On Dilevery</p>
        <input class="btn btn-primary" type="submit" name="pay" value="cash on
        dilevery">
    </form>

```

```

</form>

<form method="POST" action="complete_payment.php">
    <?php }else if(isset($_POST['order_status']) && $_POST['order_status'] == "not
    paid"){?>
        <?php $amount = strval($_POST['order_total_price']);?>
        <p>Total Payment:Rs.<?php echo $_POST['order_total_price'];?></p>
        <p>Cash On Dilevery</p>
        <input class="btn btn-primary" type="submit" name="pay" value="cash on
        dilevery">
    </form>

    <?php }else{ header("location:payment.php?error=something went wrong") ?>
        <p>You dont have an Order</p>
    <?php } ?>
</div>
</section>
<?php include('layouts/footer.php');?>

```

register.php: The user can register to the website using his credentials.

```

<?php
session_start();
include('server/connection.php');

    //if user is already registered take to account page
    if(isset($_SESSION['logged_in'])){
        header('location:account.php');
        exit;
    }
    if(isset($_POST['register'])){

        $name = $_POST['name'];
        $email = $_POST['email'];
        $password = $_POST['password'];
        $confirmpassword = $_POST['confirmpassword'];
        //if passwors dont match
        if($password !== $confirmpassword){
            header('location:register.php?error=passwords do not match');
        }
        //if passwords less than 6 characters
        }else if(strlen($password) < 6 ){
            header('location:register.php?error=password must be at least 6 characters');
        }
        //if there is no error
        }else{
            //check whether user exists
            $stmt1 = $conn->prepare("SELECT count(*) FROM users where user_email=?");
            $stmt1->bind_param('s',$email);
            $stmt1->execute();
            $stmt1->bind_result($num_rows);
            $stmt1->store_result();

```

```

$stmt1->fetch();
//user already exists with existing email
if($num_rows != 0){
    header('location:register.php?error=user with this email already exists');
}else{
    //creates new user
    $stmt = $conn->prepare("INSERT INTO (user_name,user_email,user_password)
        VALUES (?, ?, ?)");
    $stmt->bind_param('sss',$name,$email,md5($password));
    //md5 hashes the password
    //if account is created successfully
    if($stmt->execute()){
        $_SESSION['user_email'] = $email;
        $_SESSION['user_name'] = $name;
        $_SESSION['logged_in'] = true;
        header('location:account.php?message=you have registered successfully');
        //account could not be created
    }else{
        header('location:register.php?error=couldnt create an account at the moment');
    }
}
}
}
?>

```

```

<?php include('layouts/header.php');?>
<!--register-->
<section class="my-5 py-5">
    <div class="container text-center mt-3 pt-5">
        <h2 class="form-weight-bold">REGISTER</h2>
        <hr class="mx-auto">
    </div>
    <form id="register-form" method="POST" action="register.php">
    <p style="color:red;"><?php if(isset($_GET['error'])){ echo $_GET['error']; }?></p>
    <div class="mx-auto container">
    <form id="register-form">

    <div class="form-group">
        <label>Name</label>
        <input type="text" class="form-control" id="register-name" name="name"
            placeholder="Enter Your Name" required/>
    </div>
    <div class="form-group">
        <label>Email</label>
        <input type="text" class="form-control" id="register-email" name="email"
            placeholder="Enter Your Email" required/>
    </div>
    <div class="form-group">

```



```

        <label>Password</label>
        <input type="password" class="form-control" id="register-password"
        name="password" placeholder="Enter Your Password" required/>
    </div>
    <div class="form-group">
        <label>Confirm Password</label>
        <input type="password" class="form-control" id="register-confirm-password"
        name="confirmpassword" placeholder="Confirm Your Password" required/>
    </div>
    <div class="form-group">
        <input type="submit" class="btn" id="register-btn" name="register"
        value="Register"/>
    </div>
</div>
<div class="form-group">
    <a id="login-url" href="login.php" class="btn">Do you have an Account? Login</a>
</div>
</form>
</section>

<?php include('layouts/footer.php');?>

```

shop.php: The user can view all the products available in the website.

```

<?php

include('server/connection.php');
$stmt = $conn->prepare("SELECT * FROM products");
$stmt->execute();
$products = $stmt->get_result();

?>

<?php include('layouts/header.php');?>
<!--shop-->
<section id="shop" class="my-5 py-5">
    <div class="container text-center mt-9 py-5">
        <h3>OUR PRODUCTS</h3>
        <hr>
        <p>Here you can check out Our Products</p>
    </div>
    <div class="row mx-auto container">
        <?php while($row = $products->fetch_assoc()){ ?>

            <div onclick="window.location.href='single_product.php';" class="product
            text-center
            col-lg-3 col-md-4 col-sm-12">
                
                <h5 class="p-name"><?php echo $row['product_name'];?></h5>

```

```

        <h4 class="p-price">$<?php echo $row['product_price'];?></h4>
        <a href="<?php echo "single_product.php?product_id=". $row['product_id'];?>">
        <button class="btn buy-btn">Buy Now</button></a>
    </div>
<?php } ?>
</div>
</section>
<?php include('layouts/footer.php');?>

```

single_product.php: The user on clicking any product is redirected to this page where he can view a single product.

```

<?php
    include('server/connection.php');

    if(isset($_GET['product_id'])){

        $product_id = $_GET['product_id'];
        $stmt = $conn->prepare("SELECT * FROM products WHERE product_id = ?");
        $stmt->bind_param("i",$product_id);
        $stmt->execute();
        $products = $stmt->get_result();//array
        //no product id was found
    }else{
        header('location: index.php');
    }
?>

<?php include('layouts/header.php');?>
<!--single product-->
<section class="container single-product my-5 pt-5">
    <div class="row mt-5">

        <?php while($row = $products->fetch_assoc()){ ?>

            <div class="col-lg-5 col-md-6 col-md-12">
                
                <div class="small-img-group">
                    <div class="small-img-col">
                        
                    </div>
                    <div class="small-img-col">
                        
                    </div>
                </div>
            </div>

```

```

        <div class="small-img-col">
            
        </div>
        <div class="small-img-col">
            
        </div>
    </div>
</div>
<div class="col-lg-5 col-md-12 col-12">
    <h4>Tools</h4>
    <h3 class="py-4"><?php echo $row['product_name']; ?></h3>
    <h2><?php echo $row['product_price']; ?></h2>
    <form method="POST" action="cart.php">
        <input type="hidden" name="product_id" value="<?php echo
        $row['product_id']; ?>"/>

        <input type="hidden" name="product_image" value="<?php echo
        $row['product_image']; ?>"/>

        <input type="hidden" name="product_name" value="<?php echo
        $row['product_name']; ?>"/>

        <input type="hidden" name="product_price" value="<?php echo
        $row['product_price']; ?>"/>

        <input type="number" name="product_quantity" value="1"/>

        <button class="btn buy-btn" type="submit" name="add_to_cart">
            Add to Cart</button>
    </form>
    <h4 class="mt-5 mb-5">Product Details</h4>
    <span><?php echo $row['product_description']; ?></span>
</div>
<?php } ?>
</div>
</section>
<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js"
integrity="sha384MrcW6ZMFYlzcLA8Nl+NtUVF0sA7MsXsP1UyJoMp4YLEuNSf
AP+JcXn/tWtIaxVXM" crossorigin="anonymous"></script>
<script>
    var mainimg = document.getElementById("mainimg");
    var smallimg = document.getElementsByClassName("small-img");
    for(let i=0; i<4; i++){

```

```

        smallimg[i].onclick = function(){
            mainimg.src = smallimg[i].src;
        }
    }
</script>
<?php include('layouts/footer.php');?>

```

Server folder

place_order.php: This code is used to place an order given by the user.

```

<?php
session_start();
include('connection.php');

//if user is not logged in
if(!isset($_SESSION['logged_in'])){
    header("location: ../checkout.php?meaasge=please login/register to placean order");
    exit;
    //code below will not work
    //if user is logged in
}else{

    if(isset($_POST['place_order'])){
        //1.get user info and store in database
        $name = $_POST['name'];
        $email = $_POST['email'];
        $phone = $_POST['phone'];
        $city = $_POST['city'];
        $address = $_POST['address'];
        $order_cost = $_SESSION['total'];
        $order_status = "not paid";
        $user_id = $_SESSION['user_id'];
        $order_date = date('y-m-d H:i:s');
        $stmt = $conn->prepare("INSERT INTO orders
        (order_cost,order_status,user_id,user_phone,user_city,user_address,order_date)
        VALUES (?,?,?,?,?,?);");

        $stmt->bind_param
        ('isiiss',$order_cost,$order_status,$user_id,$phone,$city,$address,$order_date);
        $stmt_status = $stmt->execute();

        if(!$stmt_status){
            header("location:index.php");
            exit;
        }
        //2.issue new order and store order info in db
    }
}

```

```

$order_id = $stmt->insert_id;
//3.get products from cart (from session)
foreach($_SESSION['cart'] as $key => $value){

    $product = $_SESSION['cart'][$key];//returs array
    $product_id = $product['product_id'];
    $product_name = $product['product_name'];
    $product_image = $product['product_image'];
    $product_price = $product['product_price'];
    $product_quantity = $product['product_quantity'];
    //4.store each item in order_items in db
    $stmt1 = $conn->prepare("INSERT INTO order_items
(order_id,product_id,product_name,product_image,product_price,product_quantity,
user_id,order_date)
VALUES (?, ?, ?, ?, ?, ?, ?, ?)");

    $stmt1->bind_param
('iissiiis',$order_id,$product_id,$product_name,$product_image,$product_price,$product_quantity,
$user_id,$order_date);
    $stmt1->execute();
}
//5.remove everything from cart -> delay until payment is done
//unset($_SESSION['cart']);
$_SESSION['order_id'] = $order_id;
//6.inform user weather everything is ok or not
header("location: ../payment.php?order_status=order placed successfully");
}
}
?>

```

connection.php: This code is used to connect the user to the database.

```

<?php
$conn = mysqli_connect("localhost","root","","php_project")
or die("couldn't connect to database");
?>

```

get_equipment.php: Displays the equipment in home page.

```

<?php
include('connection.php');
$stmt = $conn->prepare("SELECT * FROM products WHERE
product_category='equipment' LIMIT 4");
$stmt->execute();
$equipment = $stmt->get_result();//array
?>

```

get_featured_products.php: Displays featured products in home page.

```

<?php

```

```
include('connection.php');
$stmt = $conn->prepare("SELECT * FROM products LIMIT 4");
$stmt->execute();
$featured_products = $stmt->get_result();//array
?>
```

get_fittings.php: Displays the fittings in home page.

```
<?php
include('connection.php');
$stmt = $conn->prepare("SELECT * FROM products WHERE
product_category='fittings' LIMIT 4");
$stmt->execute();
$fittings = $stmt->get_result();//array
?>
```

get_other.php: Displays other products in home page.

```
<?php
include('connection.php');
$stmt = $conn->prepare("SELECT * FROM products WHERE product_category='other'
LIMIT 4");
$stmt->execute();
$other = $stmt->get_result();//array
?>
```

get_tools.php: Displays the tools in home page.

```
<?php
include('connection.php');
$stmt = $conn->prepare("SELECT * FROM products WHERE product_category='tools'
LIMIT 4");
$stmt->execute();
$tools = $stmt->get_result();//array
?>
```

Admin Side

account.php: Displays account information of the admin.

```
<?php
session_start();
include('../server/connection.php');

if(!isset($_SESSION['admin_logged_in'])){
    header('location:login.php');
    exit();
}
?>
```

```
<?php include('header.php'); ?>
```

```
<h2>ACCOUNT</h2>
```

```

<?php if(isset($_GET['order_updated'])) { ?>
    <p class="text-center" style="color:green;">
<?php echo $_GET['order_updated']; ?></p>
    <?php } ?>

<?php if(isset($_GET['order_failed'])) { ?>
    <p class="text-center" style="color:red;">
    <?php echo $_GET['order_failed']; ?></p>
    <?php } ?>

<div class="table-responsive">
    <div class="container">
        <p>Admin ID : <?php echo $_SESSION['admin_id']; ?></p>
        <p>Admin Name : <?php echo $_SESSION['admin_name']; ?></p>
        <p>Admin Email : <?php echo $_SESSION['admin_email']; ?></p>
    </div>
</div>
</div>
</div>
</div>
</body>
</html>

```

add_product.php: Allows admin to add products to the website.

```

<?php
session_start();
include('../server/connection.php');
?>

<?php include('header.php'); ?>
<h2>CREATE PRODUCT</h2>
<div class="table-responsive">
    <div class="mx-auto container">
        <form id="create-form" method="POST" enctype="multipart/form-data"
        action="create_product.php">
            <p style="color :red;"><?php if(isset($_GET['error'])) {
            echo $_GET['error']; } ?></p>

            <div class="form-group mt-2">
                <input type="hidden" name="product_id" value="<?php echo
                $product['product_id']; ?>" >
                <label>Title</label>
                <input type="text" class="form-control" id="product-name" name="name"
                placeholder="Enter Product Name" required>
            </div>
            <div class="form-group mt-2">
                <label>Description</label>
                <input type="text" class="form-control" id="product-
                desc" name="description" placeholder="Enter Product Description" required>

```

```

</div>
<div class="form-group mt-2">
  <label>Price</label>
  <input type="text" class="form-control" id="product-price" name="price"
    placeholder="Enter Product Price" required>
</div>

<div class="form-group mt-2">
  <label>Color</label>
  <input type="text" class="form-control" id="produc-color" name="color"
    placeholder="Enter Product Color" required>
</div>
<div class="form-group mt-2">
  <label>Category</label>
  <select class="form-select" required name="category">
    <option value="fittings">Fittings</option>
    <option value="fittings">Equipment</option>
    <option value="tools">Tools</option>
    <option value="other">Other</option>
  </select>
</div>
<div class="form-group mt-2">
  <label>Image1</label>
  <input type="file" class="form-control" id="image1" name="image1"
    placeholder="Enter Image 1" required>
</div>
<div class="form-group mt-2">
  <label>Image2</label>
  <input type="file" class="form-control" id="image2" name="image2"
    placeholder="Enter Image 2" required>
</div>
<div class="form-group mt-2">
  <label>Image3</label>
  <input type="file" class="form-control" id="image3" name="image3"
    placeholder="Enter Image 3" required>
</div>
<div class="form-group mt-2">
  <label>Image4</label>
  <input type="file" class="form-control" id="image4" name="image4"
    placeholder="Enter Image 4" required>
</div>
<div class="form-group mt-3">
  <input type="submit" class="btn btn-primary" name="create_product"
    value="Create">
</div>
</form>
</div>
</div>

```



```

</div>
</div>
</div>
</body>
</html>

```

create_product.php: Allows admin to create new products.

```

<?php
session_start();
include('../server/connection.php');
?>

<?php
if(isset($_POST['create_product'])){

    $product_name = $_POST['name'];
    $product_description = $_POST['description'];
    $product_price = $_POST['price'];
    $product_category = $_POST['category'];
    $product_color = $_POST['color'];

    $image1 = $_FILES['image1']['tmp_name'];
    $image2 = $_FILES['image2']['tmp_name'];
    $image3 = $_FILES['image3']['tmp_name'];
    $image4 = $_FILES['image4']['tmp_name'];
    //file_name=$_FILES['image1']['name'];

    $image_name1 = $product_name."1";
    $image_name2 = $product_name."2";
    $image_name3 = $product_name."3";
    $image_name4 = $product_name."4";

    move_uploaded_file($image1,"../assets/imgs/".$image_name1);
    move_uploaded_file($image2,"../assets/imgs/".$image_name2);
    move_uploaded_file($image3,"../assets/imgs/".$image_name3);
    move_uploaded_file($image4,"../assets/imgs/".$image_name4);

    $stmt = $conn->prepare("INSERT INTO
products(product_name,product_description,product_price,product_image,
product_image2,product_image3,product_image4,product_category,product_color)
VALUES(?,?,?,?,?,?,?,?)");

    $stmt->bind_param
('ssssssss',$product_name,$product_description,$product_price,$image_name1,
$image_name2,$image_name3,$image_name4,$product_category,$product_color);

    if($stmt->execute()){

```

```

        header('location:products.php?product_created=product has been created
successfully');
    }else{
        header('location:products.php?product_failed=error occured try again');
    }
}
?>

```

delete_orders.php: Allows admin to delete orders placed by user.

```

<?php
session_start();
include('../server/connection.php');
?>

<?php
if(!isset($_SESSION['admin_logged_in'])){
    header('location:login.php');
    exit;
}
if(isset($_GET['order_id'])){

    $order_id = $_GET['order_id'];
    $stmt = $conn->prepare("DELETE FROM orders where order_id=?");
    $stmt->bind_param('i',$order_id);

    if($stmt->execute()){
        header('location:index.php?deleted_successfully=order has been deleted
successfully');
    }else{
        header('location:index.php?deleted_failure=could not delete order');
    }
}
?>

```

delete_products.php: Allows admin to delete products of the website.

```

<?php
session_start();
include('../server/connection.php');
?>

<?php
if(!isset($_SESSION['admin_logged_in'])){//remove ! if it dosent work
    header('location:login.php');
    exit;
}
if(isset($_GET['product_id'])){
    $product_id = $_GET['product_id'];
    $stmt = $conn->prepare("DELETE FROM products where product_id=?");
    $stmt->bind_param('i',$product_id);
    if($stmt->execute()){

```

```

        header('location:products.php?deleted_successfully=product has been deleted
        successfully');
    }else{
        header('location:products.php?deleted_failure=could not delete product');
    }
}
?>

```

edit_images.php: Allows admin to edit the images of the product.

```

<?php
session_start();
include('../server/connection.php');
?>
<?php
if(isset($_GET['product_id'])){
    $product_id = $_GET['product_id'];
    $product_name = $_GET['product_name'];
}else{
    header('location:products.php');
}
?>
<?php include('header.php')?>
<h2>EDIT IMAGE</h2>
<div class="table-responsive">
    <div class="mx-auto container"><form id="edit-image-form"method="POST"
    enctype="multipart/form-data" action="update_images.php">
        <p style="color: red;"><?php if(isset($_GET['error'])){echo
        $_GET['error'];} ?></p>

        <input type="hidden" name="product_id" value="<?php echo
        $product_id;?>" >
        <input type="hidden" name="product_name" value="<?php echo
        $product_name;?>" >
        <div class="form-group mt-2">
            <label>Image1</label>
            <input type="file" class="form-control" id="image1" name="image1"
            placeholder="image1" required>
        </div>
        <div class="form-group mt-2">
            <label>Image2</label>
            <input type="file" class="form-control" id="image2" name="image2"
            placeholder="image2" required>
        </div>
        <div class="form-group mt-2">
            <label>Image3</label>
            <input type="file" class="form-control" id="image3" name="image3"
            placeholder="image3" required>
        </div>
        <div class="form-group mt-2">

```

```

<label>Image4</label>
  <input type="file" class="form-control" id="image4" name="image4"
    placeholder="image4" required>
</div>
<div class="form-group mt-3">
  <input type="submit" class="btn btn-primary" name="update_images"
    value="edit">
</div>
</form>
</div>
</div>
</div>
</div>
</body>
</html>

```

```
<?php
session_start();
include('../server/connection.php');
?>

<?php

    if(isset($_GET['order_id'])){

        $order_id=$_GET['order_id'];
        $stmt = $conn->prepare("SELECT * FROM orders WHERE order_id= ?");
        $stmt->bind_param('i',$order_id);
        $stmt->execute();
        $order = $stmt->get_result();//array

    }elseif(isset($_POST['edit_order'])){

        $order_status = $_POST['order_status'];
        $order_id = $_POST['order_id'];
        $stmt = $conn->prepare("UPDATE orders SET order_status=? where order_id=?");
        $stmt->bind_param('si',$order_status,$order_id);

        if($stmt->execute()){
            header('location:index.php?order_updated=order has been updated successfully');
        }else{
            header('location:index.php?order_failed=error occurred try again');
        }
    }else{
        header('location:index.php');
        exit;
    }
}
```

?>

```

<?php include('header.php')?>
<h2>EDIT ORDER</h2>
<div class="table-responsive">
  <div class="mx-auto container">
    <form id="edit-order-form" method="POST"
      action="edit_order.php">
      <?php foreach($order as $r){ ?>

        <p style="color:red;"><?php if(isset($_GET['error'])){ echo
$_GET['error'];} ?></p>
        <div class="form-group my-3">
          <label>order-id</label>
          <p class="my-4"><?php echo $r['order_id']?></p>
        </div>
        <div class="form-group mt-3">
          <label>order-price</label>
          <p class="my-4"><?php echo $r['order_cost']?></p>
        </div>
        <input type="hidden" name="order_id" value="<?php echo
$r['order_id'];?>">
        <div class="form-group my-3">
          <label>order-status</label>
          <select class="form-select" required name="order_status">

            <option value="not paid" <?php if($r['order_status']=='not paid'){
echo"selected";} ?>>Not-Paid</option>

            <option value="paid" <?php if($r['order_status']=='paid'){ echo
"selected";} ?>>Paid</option>
          </select>
        </div>
        <div class="form-group my-3">
          <label>Order-Date</label>
          <p class="my-4"><?php echo $r['order_date']?></p>
        </div>
        <div class="form-group mt-3">
          <input type="submit" class="btn btn-primary" name="edit_order"
            value="edit">
        </div>
      <?php } ?>
    </form>
  </div>
</div>
</div>
</div>
</div>

```

```
</body>
</html>
```

edit_products.php: Allows admin to edit the product details.

```
<?php
session_start();
include('../server/connection.php');

if(isset($_GET['product_id'])){

$product_id=$_GET['product_id'];
$stmt = $conn->prepare("SELECT * FROM products WHERE product_id= ?");
$stmt->bind_param('i',$product_id);
$stmt->execute();
$products = $stmt->get_result();//array

}else if(isset($_POST['edit_btn'])){

$product_id = $_POST['product_id'];
$title = $_POST['title'];
$description = $_POST['description'];
$price = $_POST['price'];
$color = $_POST['color'];
$category = $_POST['category'];

$stmt = $conn->prepare("UPDATE products SET product_name=?,
product_description=?, product_price=?,product_color=?, product_category=?
where product_id=?");
$stmt->bind_param
('ssssi',$title,$description,$price,$color,$category,$product_id);

if($stmt->execute()){
    header('location:products.php?edit_success_message=product has been updated
successfully');
}else{
    header('location:products.php?edit_failure_message=error occured try again');
}
}else{
    header('products.php');
    exit;
}
?>

<?php include('header.php')?>

<h2>EDIT PRODUCT</h2>
<div class="table-responsive">
    <div class="mx-auto container">
```

```

<form id="edit-form" method="POST" action="edit_product.php">
  <p style="color :red;"><?php if(isset($_GET['error'])){echo
    $_GET['error'];}></p>
  <div class="form-group mt-2">
    <?php foreach($products as $product){?>
    <input type="hidden" name="product_id" value="<?php echo
    $product['product_id'];?>" >
    <label>Title</label>
    <input type="text" class="form-control" id="product-name" value="
    <?php echo $product['product_name'];?>" name="title" placeholder="title"
    required>
  </div>
  <div class="form-group mt-2">
    <label>Description</label>
    <input type="text" class="form-control" id="product-desc" value="
    <?php echo $product['product_description'];?>" name="description"
    placeholder="description" required>
  </div>
  <div class="form-group mt-2">
    <label>Price</label>
    <input type="text" class="form-control" id="product-price" value="
    <?php echo $product['product_price'];?>" name="price" placeholder="price"
    required>
  </div>
  <div class="form-group mt-2">
    <label>color</label>
    <input type="text" class="form-control" id="produc-color" value="
    <?php echo $product['product_color'];?>" name="color" placeholder="color"
    required>
  </div>
  <div class="form-group mt-2">
    <label>Category</label>
    <select class="form-select" required name="category">
      <option value="fittings">Fittings</option>
      <option value="equipment">Equipment</option>
      <option value="tools">Tools</option>
      <option value="other">Other</option>
    </select>
  </div>
  <?php }?>
  <div class="form-group mt-3">
    <input type="submit" class="btn btn-primary" name="edit_btn" value="edit">
  </div>
</form>
</div>
</div>
</div>

```

```

    </div>
</body>
</html>

```

help.php: Displays the contact information of the admin.

```

<?php
session_start();
include('../server/connection.php');

if(!isset($_SESSION['admin_logged_in'])){
    header('location:login.php');
    exit;
}

$stmt = $conn->prepare("SELECT * FROM orders");
$stmt->execute();
$orders = $stmt->get_result();//array
?>

<?php include('header.php')?>

<h2>HELP</h2>

<div class="table-responsive">
    <div class="container mt-3">
        <p>Please contact admin@gmail.com</p>
        <p>Please call 1234567890</p>
    </div>
</div>
</div>
</div>
</div>
</body>
</html>

```

index.php: It is the admin Dashboard where all the orders are tracked.

```

<?php
session_start();
include('../server/connection.php');

if(!isset($_SESSION['admin_logged_in'])){
    header('location:login.php');
    exit;
}

$stmt = $conn->prepare("SELECT * FROM orders");
$stmt->execute();
$orders = $stmt->get_result();//array
?>

```



```

<?php include('header.php')?>
<h2>DASHBOARD</h2>
<?php if(isset($_GET['order_updated'])) { ?> <p class="text-center"
    style="color:green;"><?php echo $_GET['order_updated'];?></p>
<?php } ?>

<?php if(isset($_GET['order_failed'])) { ?> <p class="text-center"
    style="color:red;"><?php echo $_GET['order_failed'];?></p>
<?php } ?>

<?php if(isset($_GET['order_deleted'])) { ?> <p class="text-center"
    style="color:green;"><?php echo $_GET['order_deleted'];?></p>
<?php } ?>

<?php if(isset($_GET['order_deleted_failed'])) { ?> <p class="text-center"
    style="color:red;"><?php echo $_GET['order_deleted_failed'];?></p>
<?php } ?>

<div class="table-responsive">
<table class="table">
<thead>
<tr>
<th>Order ID</th>
<th>Order Status</th>
<th>User ID</th>
<th>Order Date and Time</th>
<th>user Phone</th>
<th>User Address</th>
<th>Edit</th>
<th>Delete</th>
</tr>
</thead>
<tbody>
<?php foreach($orders as $order) { ?>
<tr>
<td><?php echo $order['order_id'];?></td>
<td><?php echo $order['order_status'];?></td>
<td><?php echo $order['user_id'];?></td>
<td><?php echo $order['order_date'];?></td>
<td><?php echo $order['user_phone'];?></td>
<td><?php echo $order['user_address'];?></td>

<td><a class="btn btn-primary" href="edit_order.php?order_id=<?php echo
$order['order_id'];?>">Edit</a></td>

<td><a class="btn btn-danger" href="delete_orders.php?order_id=<?php echo
$order['order_id'];?>">Delete</a></td>
</tr>

```

```

        <?php }?>
    </tbody>
</table>
</div>
</div>
</div>
</div>
</body>
</html>

```

login.php: Admin can login to the dashboard using this code.

```

<?php
session_start();
include('../server/connection.php');

if(isset($_SESSION['admin_logged_in'])){
    header('location:login.php');
    exit();
}

if(isset($_POST['login_btn'])){

    $email = $_POST['email'];
    $password = md5($_POST['password']);

    $stmt = $conn->prepare("SELECT admin_id, admin_name, admin_email,
admin_password FROM admins WHERE admin_email=? AND
admin_password=? LIMIT 1");
    $stmt->bind_param('ss',$email,$password);

    if($stmt->execute()){
        $stmt->bind_result
        ($admin_id,$admin_name,$admin_email,$admin_password);
        $stmt->store_result();
        if($stmt->num_rows() == 1){
            $stmt->fetch();

            $_SESSION['admin_id'] = $admin_id;
            $_SESSION['admin_name'] = $admin_name;
            $_SESSION['admin_email'] = $admin_email;
            $_SESSION['admin_logged_in'] = true;

            header('location: index.php?message=logged in successfully');
        }else{
            header('location: login.php?error=could not verify your account');
        }
    }else{
        //error
    }
}

```

```

        header('location: login.php?error=something went wrong');
    }
}
?>

```

logout.php: Logs the admin out of the dashboard.

```

<?php
session_start();
include('../server/connection.php');

if(isset($_GET['logout']) && $_GET['logout'] == 1) {
    if(isset($_SESSION['admin_logged_in'])) {
        unset($_SESSION['admin_logged_in']);
        unset($_SESSION['admin_email']);
        unset($_SESSION['admin_name']);
        header ('loaction: login.php');
        exit;
    }
}
?>

```

orders.php: The admin can also view the orders in this page.

```

<?php
session_start();
include('../server/connection.php');

if(!isset($_SESSION['admin_logged_in'])){
    header('location:login.php');
    exit;
}
$stmt = $conn-> prepare ("SELECT * FROM orders");
$stmt-> execute ();
$orders = $stmt->get_result();//array
?>

<?php include('header.php')?>

<h2>ORDERS</h2>
<?php if(isset($_GET['order_updated'])){?>
    <p class="text-center" style="color:green;"><?php echo
        $_GET['order_updated'];?></p>
<?php }?>

<?php if(isset($_GET['order_failed'])){?>
    <p class="text-center" style="color:red;"><?php echo $_GET['order_failed'];?></p>
<?php }?>

<?php if(isset($_GET['order_deleted'])){?>

```

```

        <p class="text-center" style="color:green;"><?php echo
        $_GET['order_deleted'];?></p>
    <?php }?>

    <?php if(isset($_GET['order_deleted_failed'])){?>
    <p class="text-center" style="color:red;"><?php echo
    $_GET['order_deleted_failed'];?></p>
    <?php }?>

    <div class="table-responsive">
    <table class="table">
    <thead>
    <tr>
    <th>Order ID</th>
    <th>Order Status</th>
    <th>User ID</th>
    <th>Order Date and Time</th>
    <th>user Phone</th>
    <th>User Address</th>
    <th>Edit</th>
    <th>Delete</th>
    </tr>
    </thead>
    <tbody>
    <?php foreach($orders as $order){?>
    <tr>
    <td><?php echo $order['order_id'];?></td>
    <td><?php echo $order['order_status'];?></td>
    <td><?php echo $order['user_id'];?></td>
    <td><?php echo $order['order_date'];?></td>
    <td><?php echo $order['user_phone'];?></td>
    <td><?php echo $order['user_address'];?></td>

    <td><a class="btn btn-primary" href="edit_order.php?order_id=<?php echo
    $order['order_id'];?>">Edit</a></td>

    <td><a class="btn btn-danger" href="delete_orders.php?order_id=<?php echo
    $order['order_id'];?>">Delete</a></td>
    </tr>
    <?php }?>
    </tbody>
    </table>
    </div>
    </div>
    </div>
    </div>
    </body>
    </html>

```

products.php: the admin can view and edit all the products of the website.

```

<?php
session_start();
include('../server/connection.php');

    if(!isset($_SESSION['admin_logged_in']))
        header('location:login.php');
        exit;
    }
    $stmt = $conn->prepare("SELECT * FROM products");
    $stmt->execute();
    $products = $stmt->get_result();//array
?>

<?php include('header.php')?>

<h2>PRODUCTS</h2>
<?php if(isset($_GET['product_created'])) { ?>

<p class="text-center" style="color:green;"><?php echo
$_GET['product_created'];?></p>
<?php }?>

<?php if(isset($_GET['product_failed'])) { ?>
<p class="text-center" style="color:red;"><?php echo $_GET['product_failed'];?></p>
<?php }?>

<?php if(isset($_GET['edit_success_message'])) { ?>
<p class="text-center" style="color:green;"><?php echo
$_GET['edit_success_message'];?></p>
<?php }?>

<?php if(isset($_GET['edit_failure_message'])) { ?>
<p class="text-center" style="color:red;"><?php echo
$_GET['edit_failure_message'];?></p>
<?php }?>

<?php if(isset($_GET['deleted_successfully'])) { ?>
<p class="text-center" style="color:green;"><?php echo
$_GET['deleted_successfully'];?></p>
<?php }?>

<?php if(isset($_GET['deleted_failure'])) { ?>
<p class="text-center" style="color:red;"><?php
echo$_GET['deleted_failure'];?></p>
<?php }?>

<?php if(isset($_GET['images_updated'])) { ?>

```

```

<p class="text-center" style="color:green;"><?php echo
$_GET['images_updated'];?></p>
<?php }?>

<?php if(isset($_GET['images_failed'])) { ?>
<p class="text-center" style="color:red;"><?php echo$_GET['images_failed'];?></p>
<?php }?>

<div class="table-responsive">
<table class="table">
<thead>
<tr>
<th>Product ID</th>
<th>Product Image</th>
<th>Product Name</th>
<th>Product Price</th>
<th>Product Category</th>
<th>Product Color</th>
<th>Edit Images</th>
<th>Edit</th>
<th>Delete</th>
</tr>
</thead>
<tbody>
<?php foreach ($products as $product) { ?>

<tr>
<td><?php echo $product['product_id'];?></td>
<td>"
style="width:70px;height:70px; "></td>
<td><?php echo $product['product_name'];?></td>
<td><?php echo "Rs." . $product['product_price'];?></td>
<td><?php echo $product['product_category'];?></td>
<td><?php echo $product['product_color'];?></td>

<td><a class="btn btn-warning" href="<?php
echo"edit_images.php?product_id="
$product['product_id']."&product_name=" . $product['product_name'];?>">
Edit images</a></td>

<td><a class="btn btn-primary" href="edit_product.php?product_id=<?php echo
$product['product_id'];?>">Edit</a></td>
<td><a class="btn btn-danger" href="delete_product.php?product_id=<?php echo
$product['product_id'];?>">Delete</a></td>
</tr>
<?php }?>
</tbody>
</table>

```

```

</div>
</div>
</div>
</div>
</body>
</html>

```

update_image.php: The admin can change the images of existing product.

```

<?php
session_start();
include('../server/connection.php');
?>
<?php
    if(isset($_POST['update_images'])){

        $product_id = $_POST['product_id'];
        $product_name = $_POST['product_name'];

        $image1 = $_FILES['image1']['tmp_name'];
        $image2 = $_FILES['image2']['tmp_name'];
        $image3 = $_FILES['image3']['tmp_name'];
        $image4 = $_FILES['image4']['tmp_name'];

        //$file_name=$_FILES['image1']['name'];
        $image_name1 = $product_name."1.jpeg";
        $image_name2 = $product_name."2.jpeg";
        $image_name3 = $product_name."3.jpeg";
        $image_name4 = $product_name."4.jpeg";

        move_uploaded_file($image1,"../assets/imgs/".$image_name1);
        move_uploaded_file($image2,"../assets/imgs/".$image_name2);
        move_uploaded_file($image3,"../assets/imgs/".$image_name3);
        move_uploaded_file($image4,"../assets/imgs/".$image_name4);

        $stmt = $conn->prepare("UPDATE products SET
        product_image=?,product_image2=?,product_image3=?,product_image4=? WHERE
        product_id=?");
        $stmt->bind_param
        ('ssssi',$image_name1,$image_name2,$image_name3,$image_name4,$product_id);

        if($stmt->execute()){
            header('location:products.php?images_updated=images has been updated
            successfully');
        }else{
            header('location:products.php?images_failed=error occurred try again');
        }
    }
?>

```

TESTING

7.1 Introduction:

Software testing is the process used to help identify the correctness, completeness, security and quality of developed computer software. This includes the process of executing the program or application with the intent of finding error. Testing phase computes the state and behavior of the product against a specification.

7.2 Testing Criteria:

The testing phase consists of evaluating the software that has been developed in order to confirm that it produces the output required in a safe and efficient manner. In this phase inherent errors that occur, have to be handled and the user should be informed so that he/she can follow the guidelines and instructions and get around the error and obtain the output.

7.2.1 Objective of Testing:

During testing, the program to be tested is executed with a set of test cases and the output of the program for the test cases is evaluated to determine if the program is performing as expected. Clearly, the success of testing in revealing error in programs depends critically on the test cases. Because code is the only product that can be executed and whose actual behavior can be observed, testing is the phase where the errors remaining from all the previous phases is detected.

7.3 Testing Methodologies:

Following are the testing methodologies:

7.3.1 Unit Testing:

Unit testing focuses efforts on the smallest unit of software design. This is known as module testing. The modules are tested separately during programming stage itself. In this step, each module is found to be working satisfactory as regards to the expected output from the module.

7.3.2 Integration Testing:

Data can be lost across an interface. One module can have an adverse effect on another, sub functions, may not be linked in desired manner in major functions. Integration testing is a systematic approach for constructing the programmer structure, while at the same time conducting test to uncover errors associated within the interface. The objective is to take unit tested module and build program structure. All the modules are combined and tested as a whole.

Top-Down Integration:

This method is an incremental approach to the construction of program structure. Modules are integrated by moving downward through the control hierarchy, beginning with the main program module. The module subordinates to the main program module are incorporated into the structure in earlier a depth first or breadth first name.

Bottom-Up Integration:

This method begins the construction and testing with modules at the lowest levelling the program structure. Since the modules are integrated from the bottom-up processing required for the modules subordinated to a given level is always available and need for stubs is eliminated.

Validation:

At the culmination of the integration testing, software is completely assembled as a package. Interfacing errors have been uncovered and corrected and a final series of software test begin in validation testing. Validation test can be defined in many ways, but a simple definition is that the validation succeeds when the software functions in a manner that is expected by the customer. After validation test has been conducted, one of the three possible conditions exists.

- a) The function or performance characteristics confirm to specification and are accepted.
- b) A deviation from specified is uncovered and a deficiency list is created.
- c) Proposed system under consideration has been tested by using validation test and found to be working satisfactory.

- **Output testing:**

After performing the validation testing, the next step is output testing of the proposed system, since no system could be useful if it does not produce the required output in a specific format. The output format on the screen is found to be correct; the format was designed in the system design time according to the user needs. For the hard copy also, the output comes as per the specified requirements by the user. Hence output testing did not result in any correction for the system.

- **User acceptance testing:**

User acceptance of a system is the key factor for the success of any system. The system under consideration is tested for the user acceptance by constantly keeping in touch with the perspective system user at the time of developing and making changes whenever required.

This is done regard to the following point –

- i. Input screen design
- ii. Output screen design
- iii. Format of reports and other output.

7.3.3 Preparation of Test Data:

The above testing is done by taking various kind of test data. Preparation of test data plays a vital role in the system testing. After preparing the test data the system under study is tested using that test data. While testing the system by using test data errors are again uncovered and corrected by using above testing steps and corrections are also noted for future use.

Live test data are those that are actually extracted from organization files. After a system is partially constructed, programmers or analysts often ask users to key in a set of data from either normal activity. Then the system person uses this data as a way to partially test the system. In other instances, programmers or analysts extract a set of live data from the files and have them entered themselves. It is difficult to obtain live data in sufficient amounts to conduct extensive testing. And although it is realistic data that will show how the system will perform for the typical processing requirements, assuming that the live data entered is fact typical, such data generally will not test combinations or formats that can enter the system. This bias towards typical values then does not provide a true system test and in fact ignores the cases most likely to cause the system failure.

7.3.4 Using artificial test data:

Artificial test data are created solely for test purposes, since they can be generated to test all combination of formats and values. In other words, the artificial data, which can quickly be prepared by a data generating utility program in the information systems department, make possible the testing of all login and control paths through the program.

The most effective test program uses artificial test data generated by person other than who wrote the program.

7.4 System Testing:

System Testing is the testing of a complete and fully integrated software product. Usually, software is only one element of a larger computer-based system. Ultimately, software is interfaced with other software/hardware systems. System Testing is actually a series of different tests whose sole purpose is to exercise the full computer-based system.

System testing tables:

Sl. No.	Test condition	Test report
1.	System loading	Successful
2.	System run procedure	Successful
3.	File I/O operation	Successful
4.	Database communication	Successful
5.	Server/client interaction	Successful
6.	Memory usage	Normal
7.	System processor usage	Normal
8.	Authentication/Authorization	Successful

7.5 Test Cases:

Login: This test case is observed when the user logs in to the website.

No.	Test Case	Expected Output	Observed Output
1.	Required fields are empty.	Required fields cannot be empty.	As Expected,
2.	Incorrect email and password.	These credentials don't match our record.	As Expected,
3.	Valid email and password combination entered.	Redirection to home page.	As Expected,
4.	Authentication.	The user's email and password are checked and verified.	As Expected,

User Shop Page: This test case is observed when the user clicks on the shop button.

No.	Test Case	Expected output	Observed output
1.	When user shop page is loaded.	Display navigation bar with All products displayed with footer.	As Expected,
2.	Home button on clicks.	Redirect to home page.	As Expected,
3.	Shop button on clicks.	Redirect to shop page.	As Expected,
4.	Contact button on click.	Redirect to contact page.	As Expected,
5.	Cart button on click.	Redirect to cart page.	As Expected,
6.	Account button on click.	Redirect to account page.	As Expected,
7.	Click on any product button.	Redirect single product Page.	As Expected,

User Home Page: This test case is observed when the user clicks on home button.

No.	Test Case	Expected output	Observed output
1.	When user home page is loaded.	Display navigation bar with home, shop, contact, account And cart options with products displayed and footer.	As Expected,
2.	Home button on clicks.	Redirect to home page	As Expected,
3.	Shop button on clicks.	Redirect to shop page	As Expected,
4.	Contact button on click.	Redirect to contact page	As Expected,
5.	Cart button on click.	Redirect to cart page	As Expected,
6.	Account button on click.	Redirect to account page	As Expected,
7.	Click on any product button.	Redirect single product page	As Expected,
8.	Select Logout.	Redirect to login page	As Expected,

User account Page: This test case is observed when the user clicks on account button.

No.	Test Case	Expected output	Observed output
1.	When user account is loaded.	Display navigation bar with With account info, change password and your orders options with footer.	As Expected,
2.	Home button on clicks.	Redirect to home page.	As Expected,
3.	Shop button on clicks.	Redirect to shop page.	As Expected,
4.	Contact button on click.	Redirect to contact page.	As Expected,
5.	Cart button on click.	Redirect to cart page.	As Expected,
6.	Account button on click.	Redirect to account page.	As Expected,
7.	Click on any product button.	Redirect single product page.	As Expected,

User cart Page: This test case is observed when the user clicks on cart button.

No.	Test Case	Expected output	Observed output
1.	When user shop cart is loaded.	Display navigation bar with Your cart that contains the orders with details and checkout button with footer.	As Expected,
2.	Home button on clicks.	Redirect to home page.	As Expected,
3.	Shop button on clicks.	Redirect to shop page.	As Expected,
4.	Contact button on click.	Redirect to contact page.	As Expected,
5.	Cart button on click.	Redirect to cart page.	As Expected,
6.	Account button on click.	Redirect to account page.	As Expected,
7.	Click on any product button.	Redirect single product page.	As Expected,

Admin Login: This test case is observed when the admin logs in to the website.

No.	Test Case	Expected Output	Observed Output
1.	Required fields are empty.	Required fields cannot be empty.	As Expected,
2.	Incorrect email and password.	These credentials don't match our record.	As Expected,
3.	Valid email and password combination entered.	Redirection to home page.	As Expected,
4.	Authentication.	The admin's email and password are checked and verified.	As Expected,

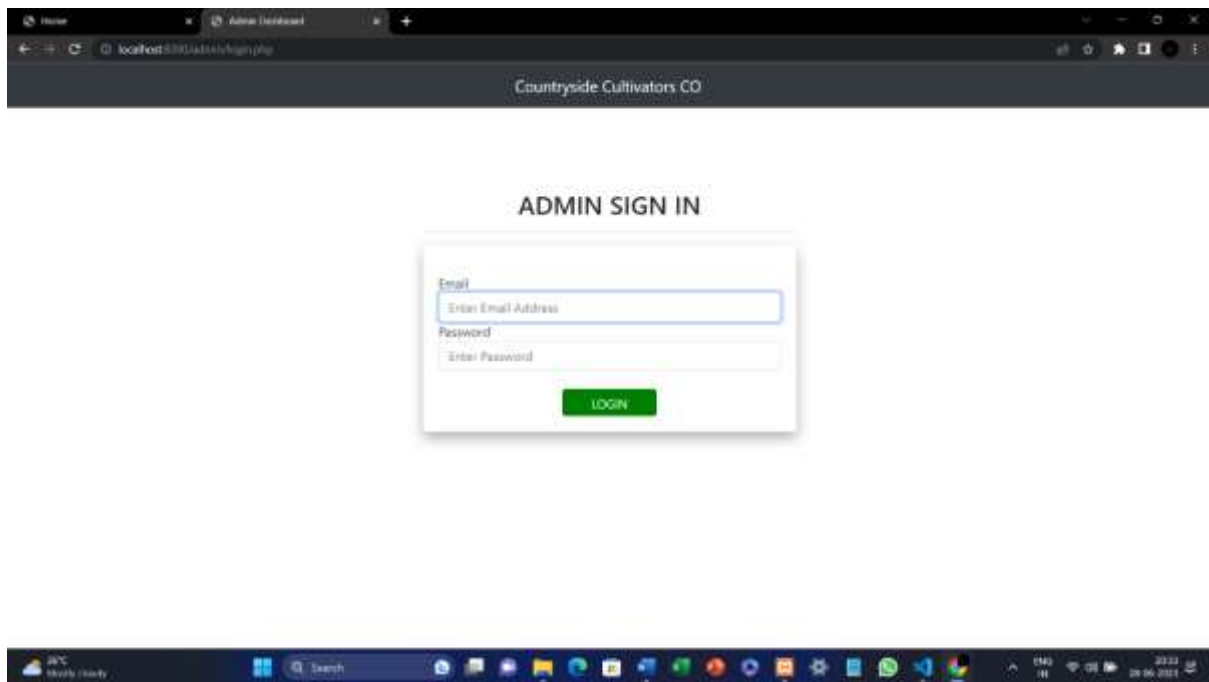
Admin Home Page: This test case is observed when admin logs in.

No.	Test Case	Expected output	Observed output
1.	When Admin home page is loaded.	Display side bar and Dashboard with all the orders.	As Expected,
2.	Select add new product.	Redirect to create product page.	As Expected,
3.	Select orders option.	Displays all the orders placed by users with edit and delete button.	As Expected,
4.	Select products option.	Displays all the products in the website with edit images edit product and delete product option.	As Expected,
5.	Select Admin account option.	Redirects to admin account info page.	As Expected,
6.	Select help option.	Redirect to help page.	As Expected,
7.	Select Logout.	Redirect to login page.	As Expected,

8. SNAPSHOTS

Admin Model

Admin Login: This is the admin sign in page.



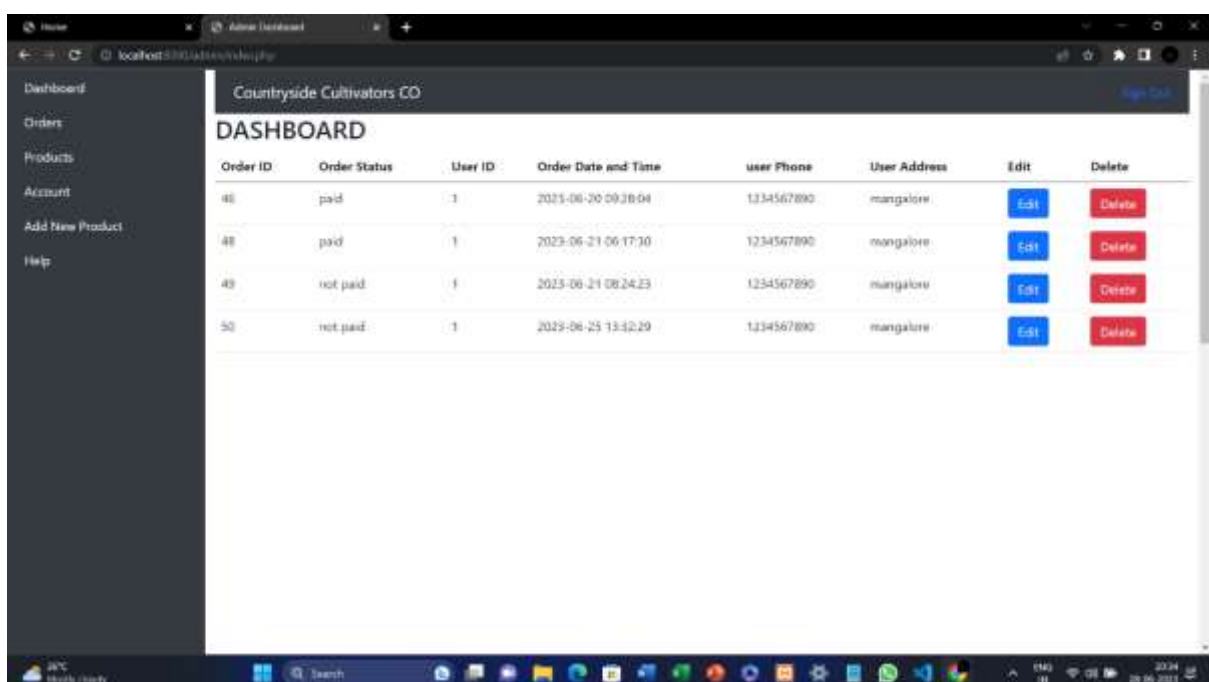
ADMIN SIGN IN

Email
Enter Email Address

Password
Enter Password

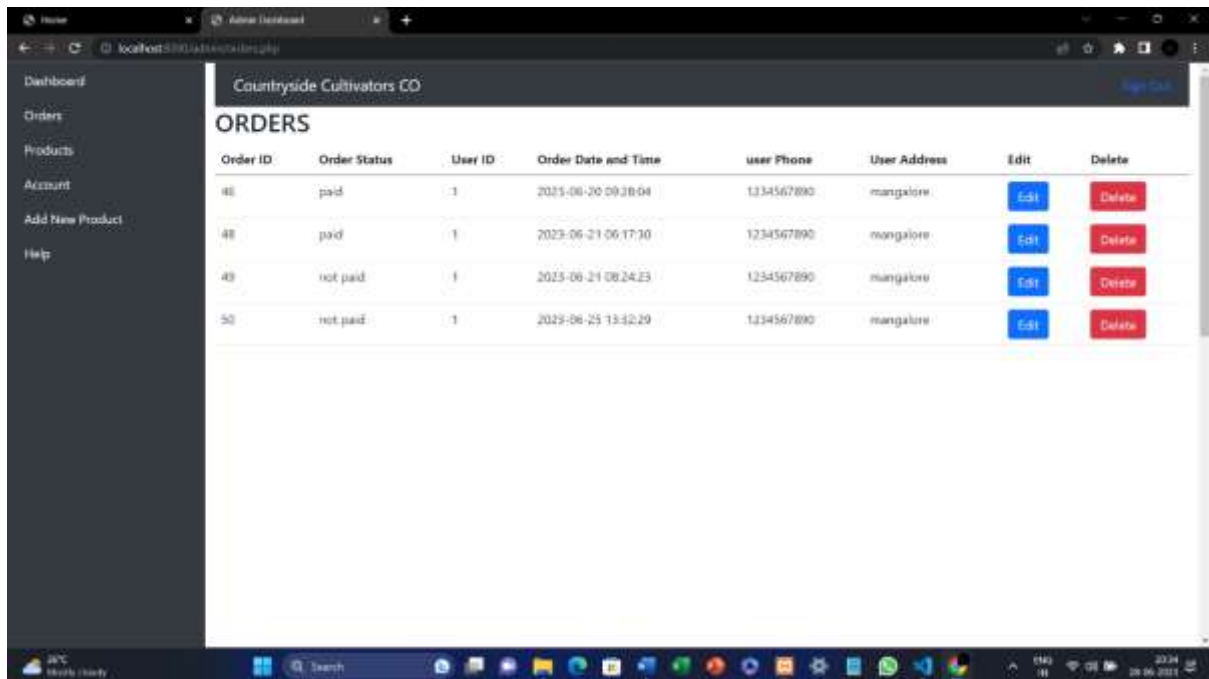
LOGIN

Admin Dashboard: The home page or dashboard for the Admin.

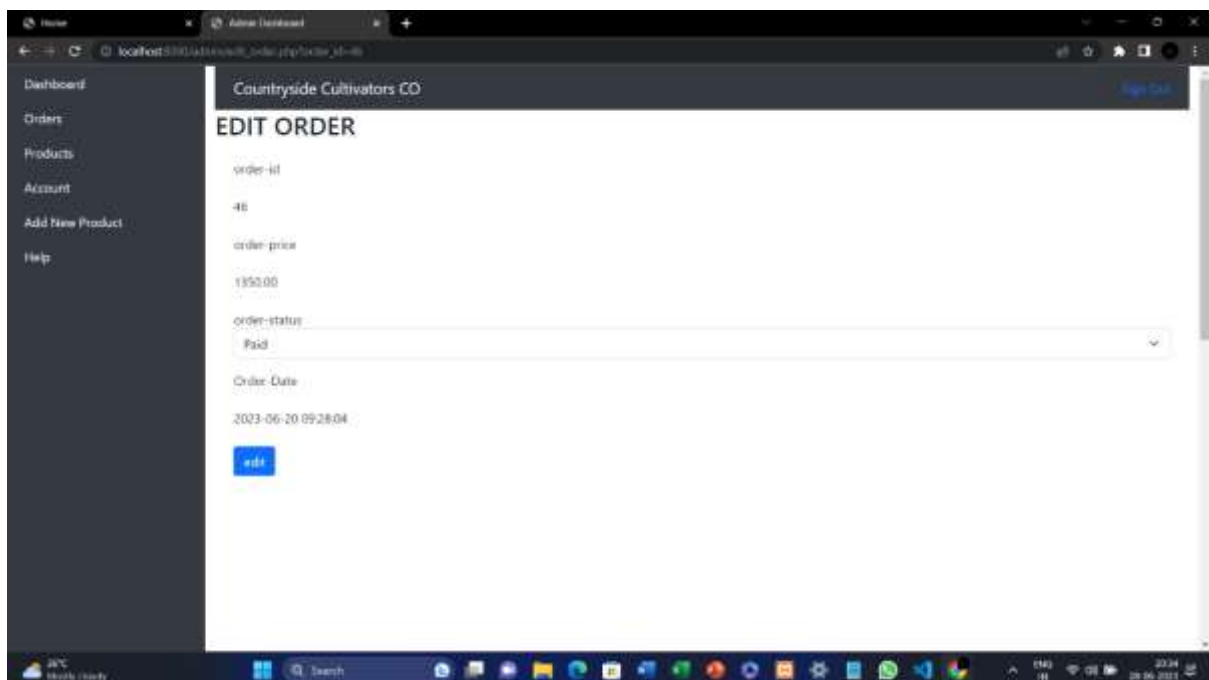


Order ID	Order Status	User ID	Order Date and Time	user Phone	User Address	Edit	Delete
46	paid	1	2023-06-20 09:38:04	1234567890	mangalore	Edit	Delete
48	paid	1	2023-06-21 06:17:10	1234567890	mangalore	Edit	Delete
49	not paid	1	2023-06-21 08:24:23	1234567890	mangalore	Edit	Delete
50	not paid	1	2023-06-25 13:10:29	1234567890	mangalore	Edit	Delete

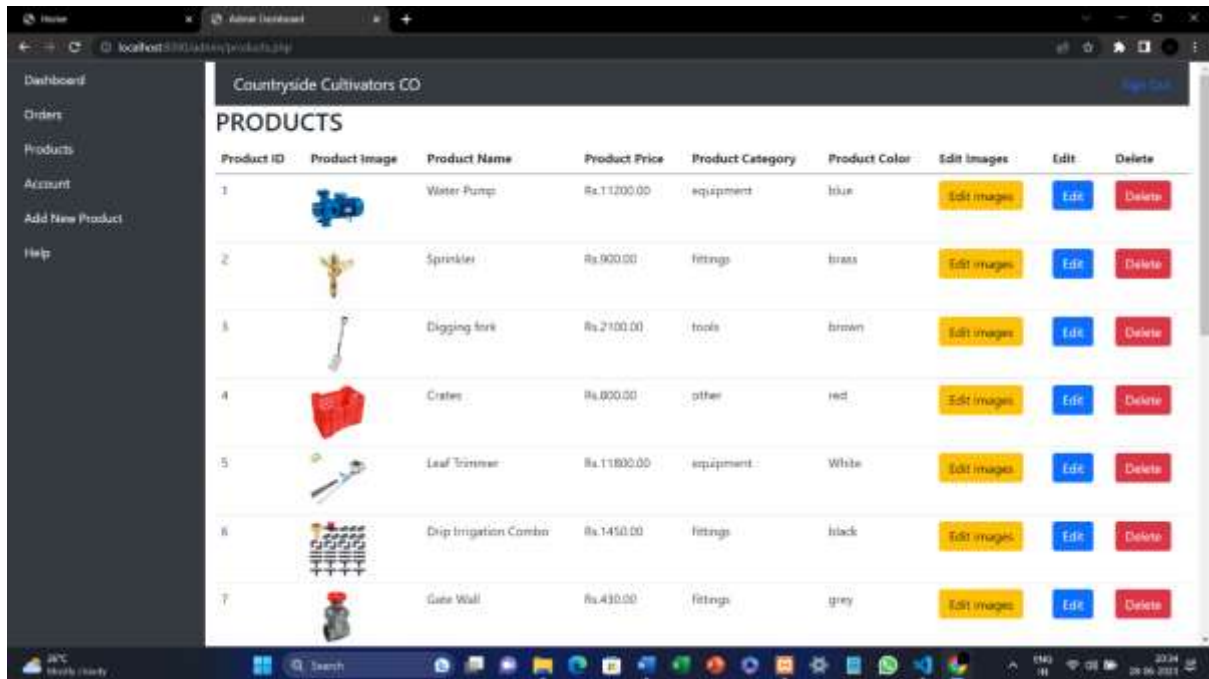
Orders: All the orders placed by users are displayed in this page.



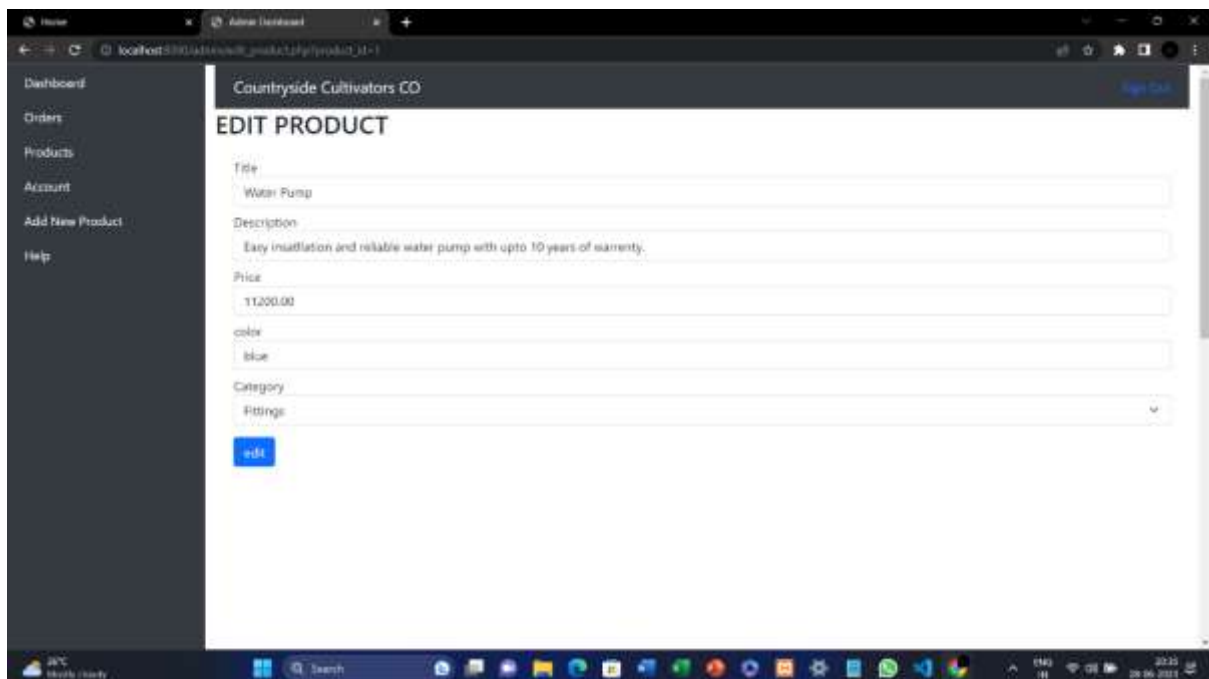
Edit Order: Orders placed by users can be edited in this page.



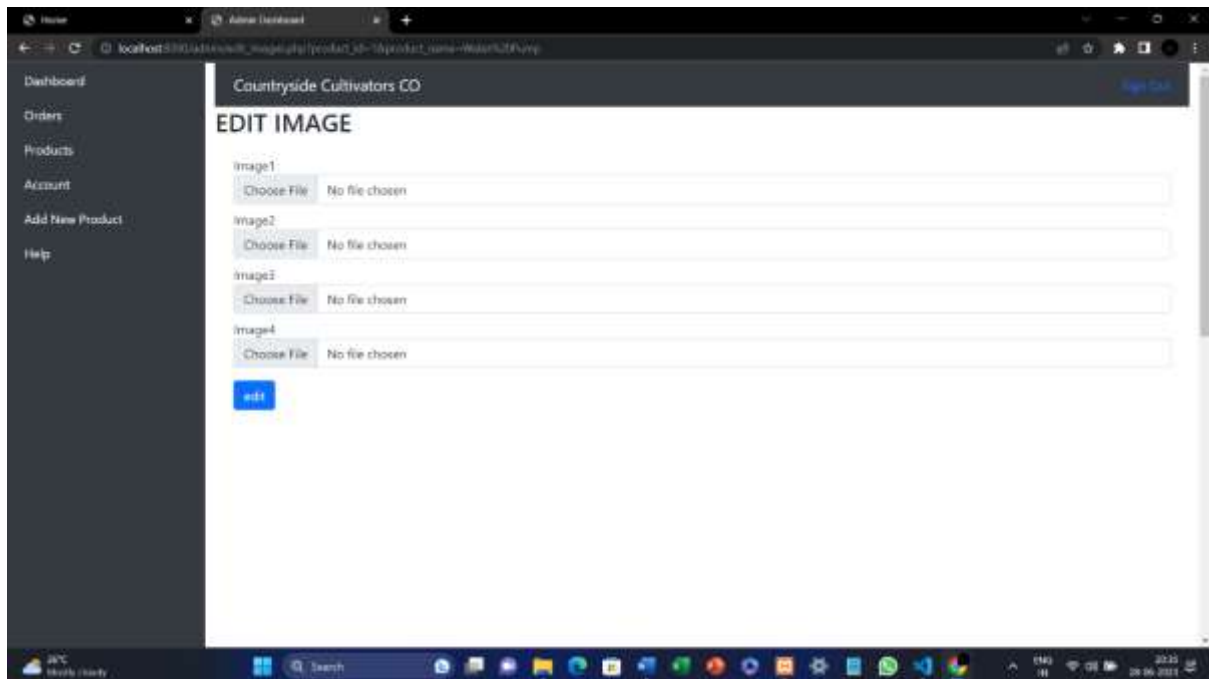
Products: All the products in the website are displayed in this page. The product images and details can be edited and products can be deleted here.



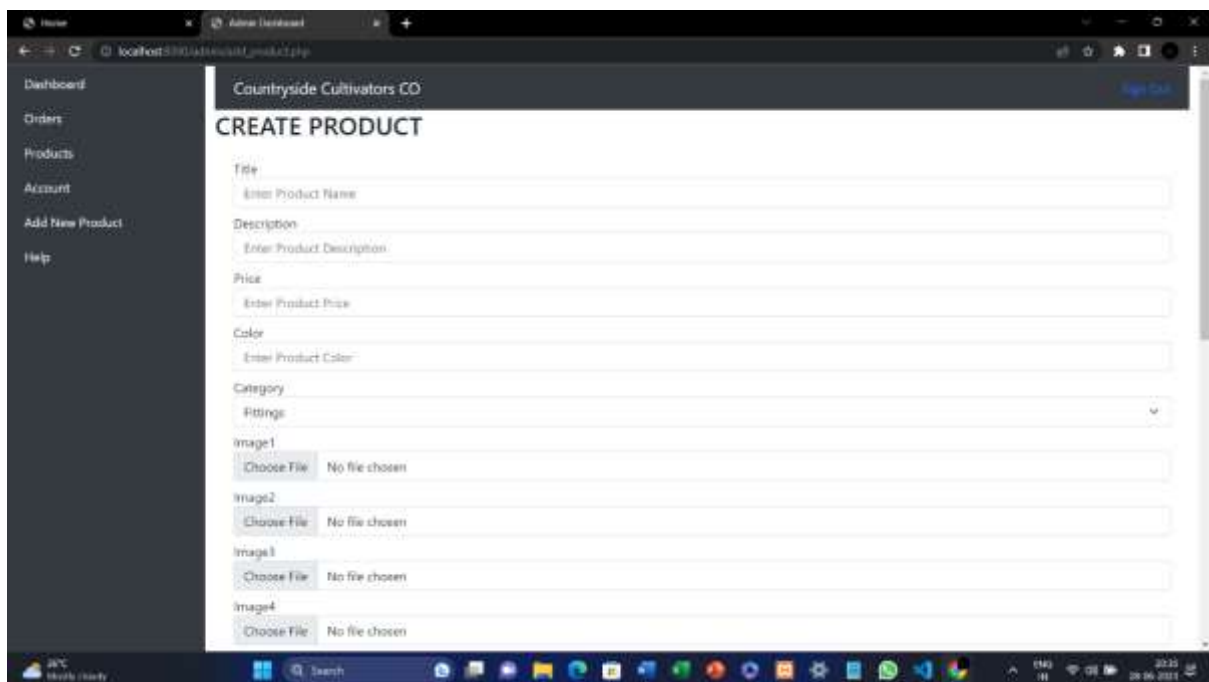
Edit Products: The product details can be edited in this page.



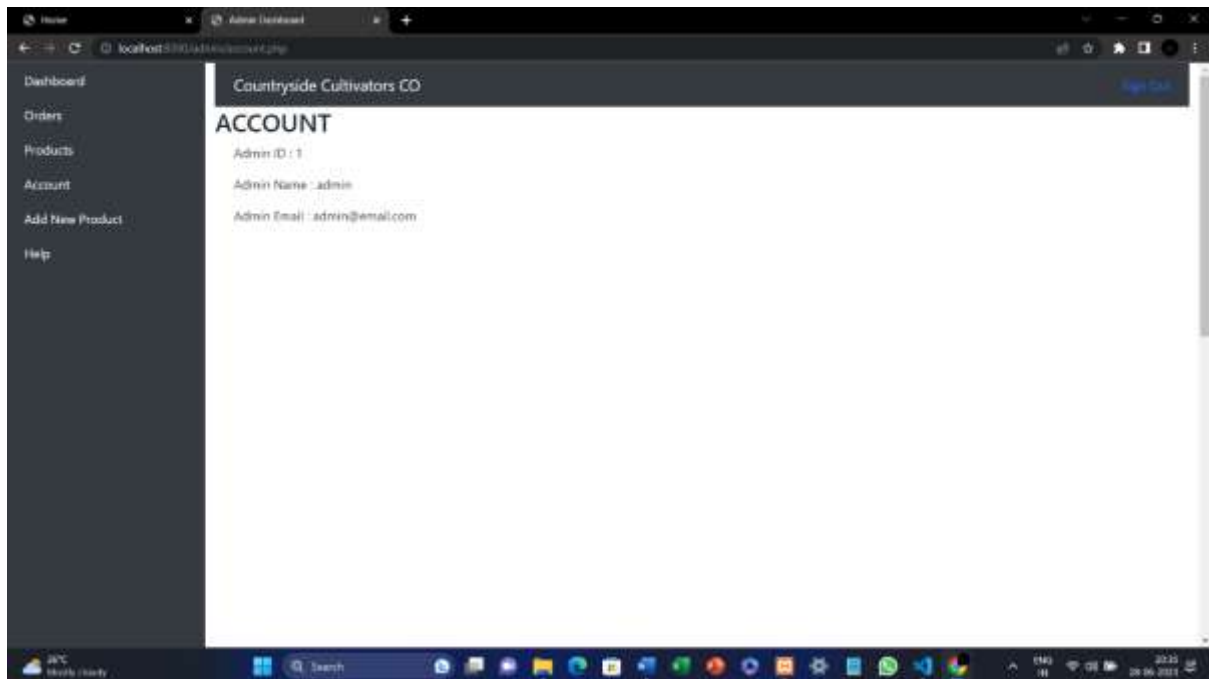
Edit Images: The product image can be edited in this page.



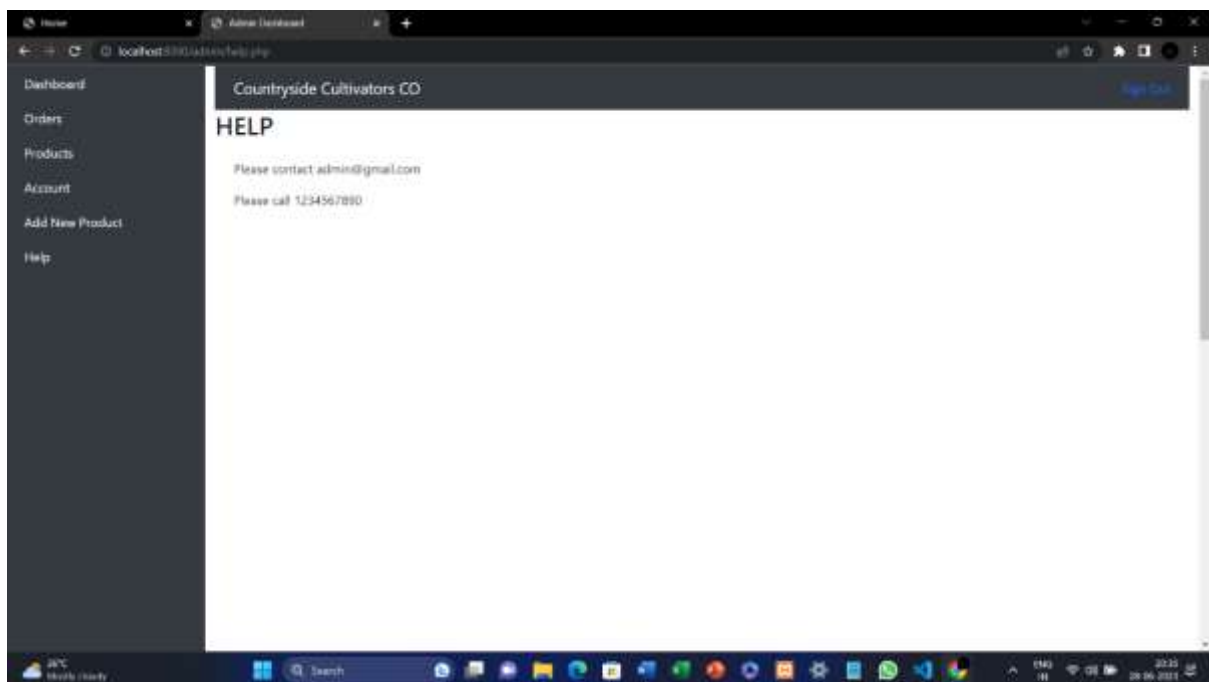
Create Product: A new product can be created in this page.



Account: Admin can view his account details in this page.

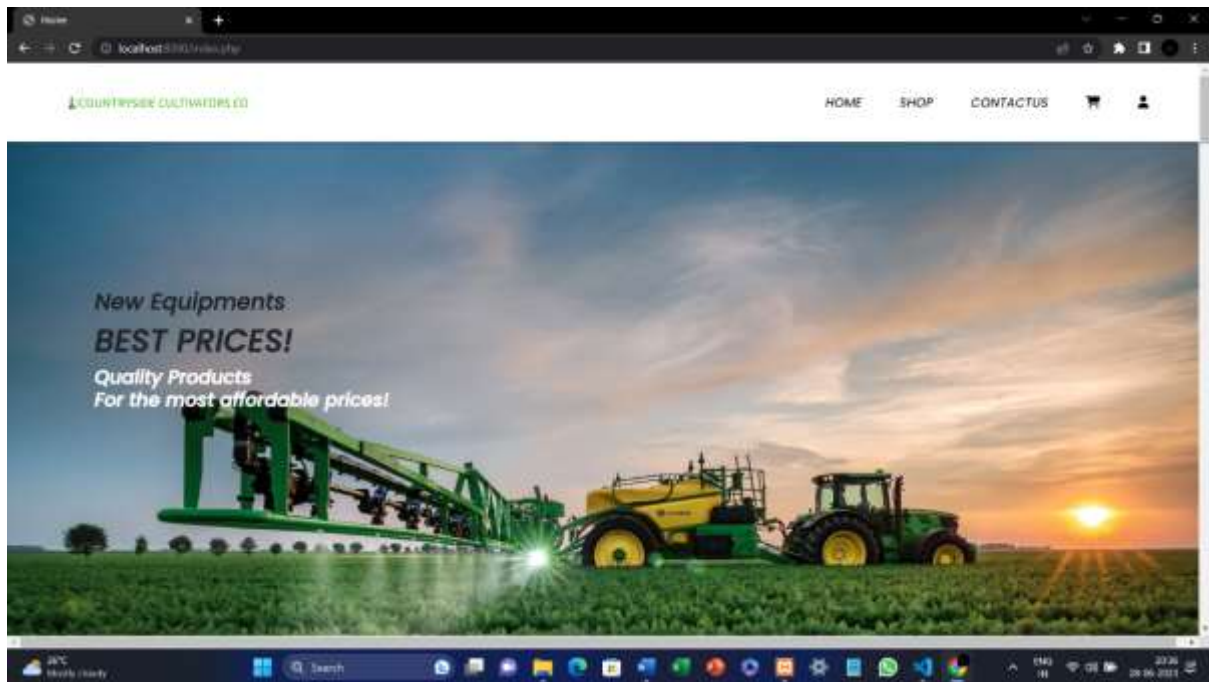


Help: The contact details of admin are present in this page.

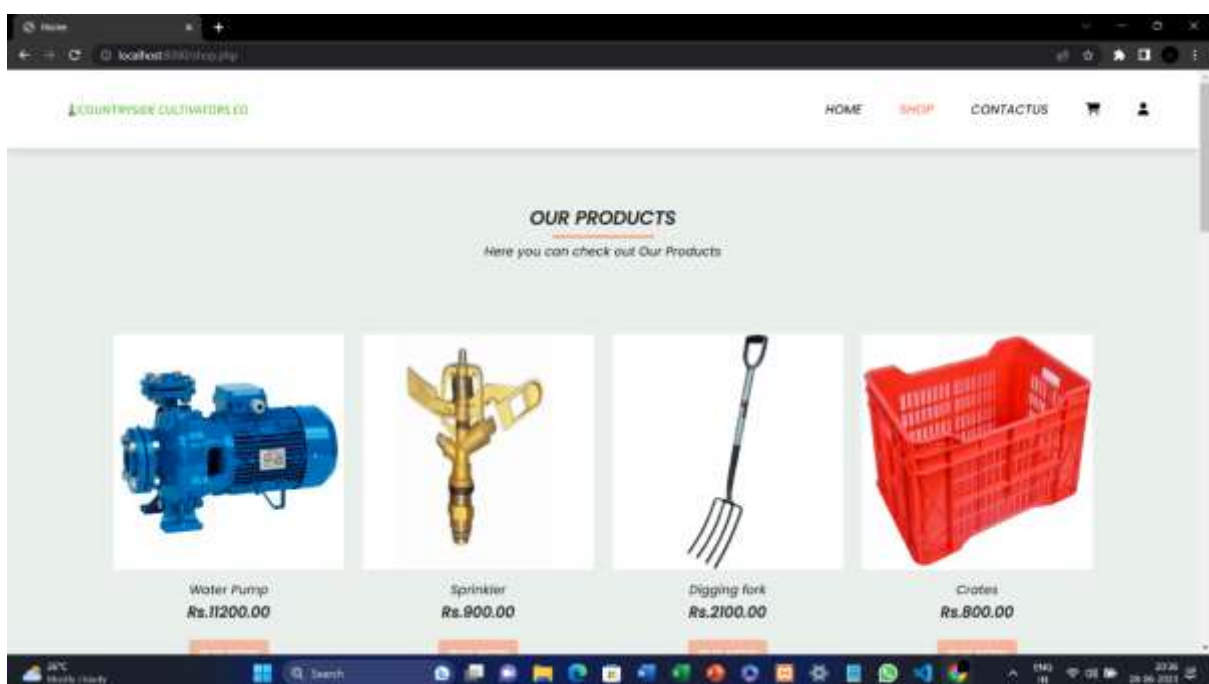


User Model

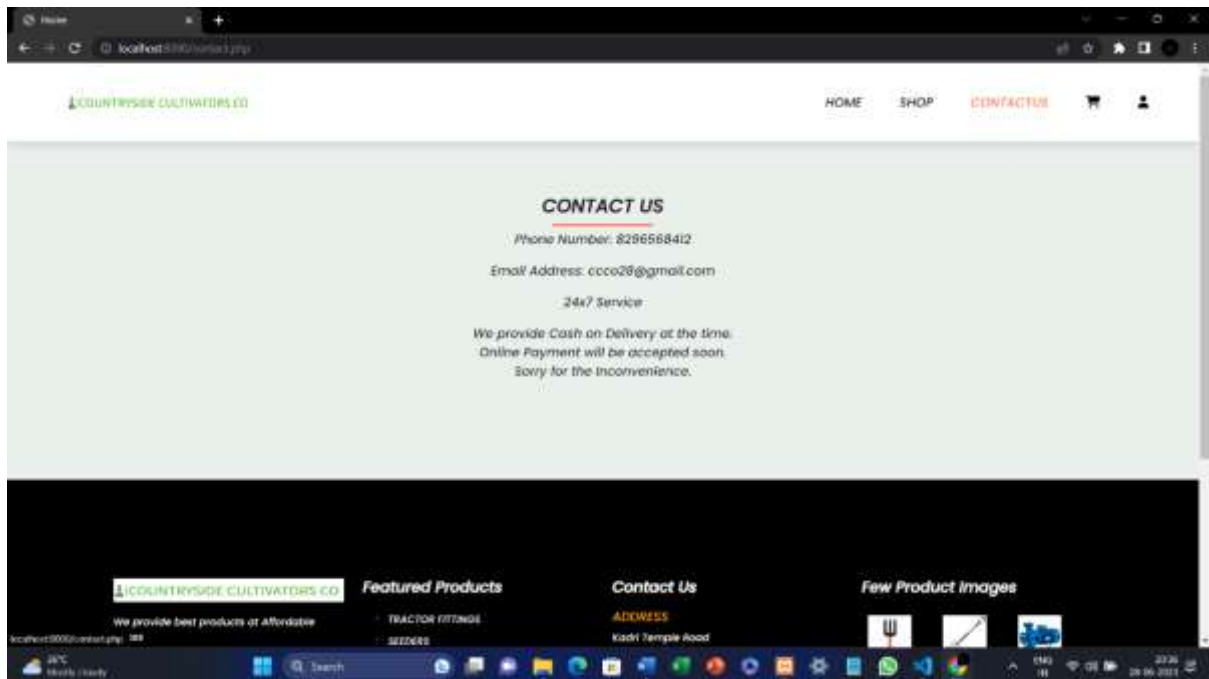
Home: The main home page of user side model.



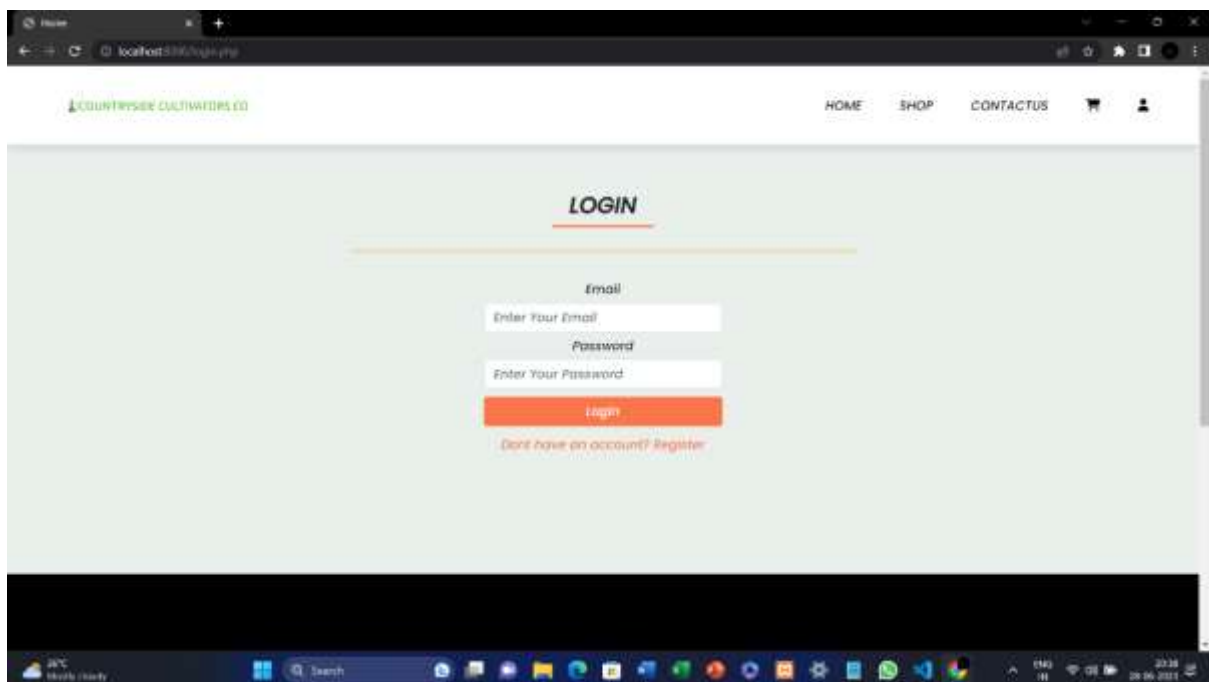
Shop: The shop page displays all the products in the website.



Contact Us: Displays all the contact details of the website.

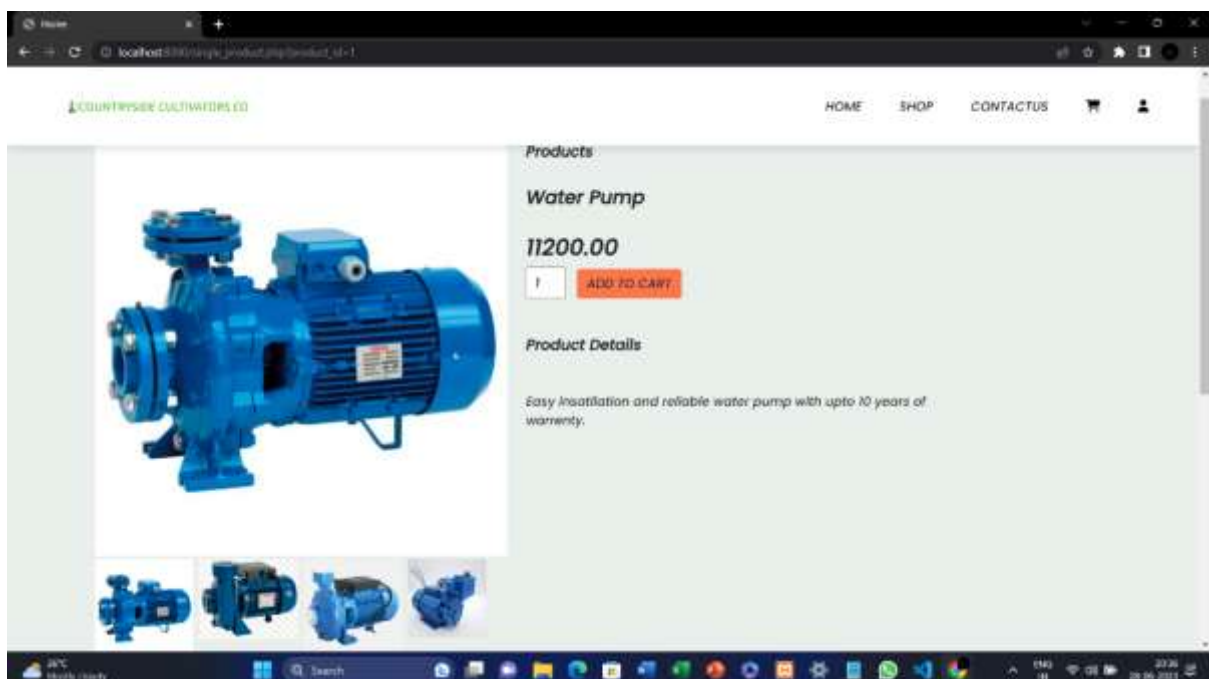


Login: The login page logs in the user to the website to purchase a product.

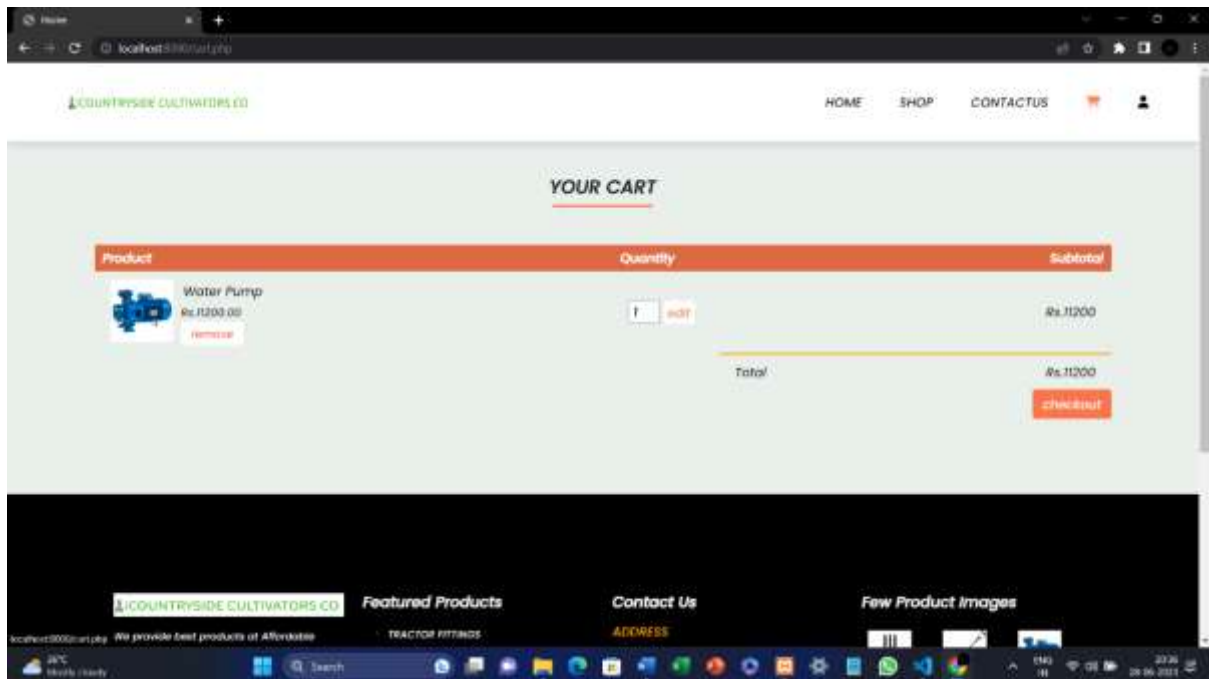


Register: If the user does not have an account he can create one by registering using the required details needed to create an account.

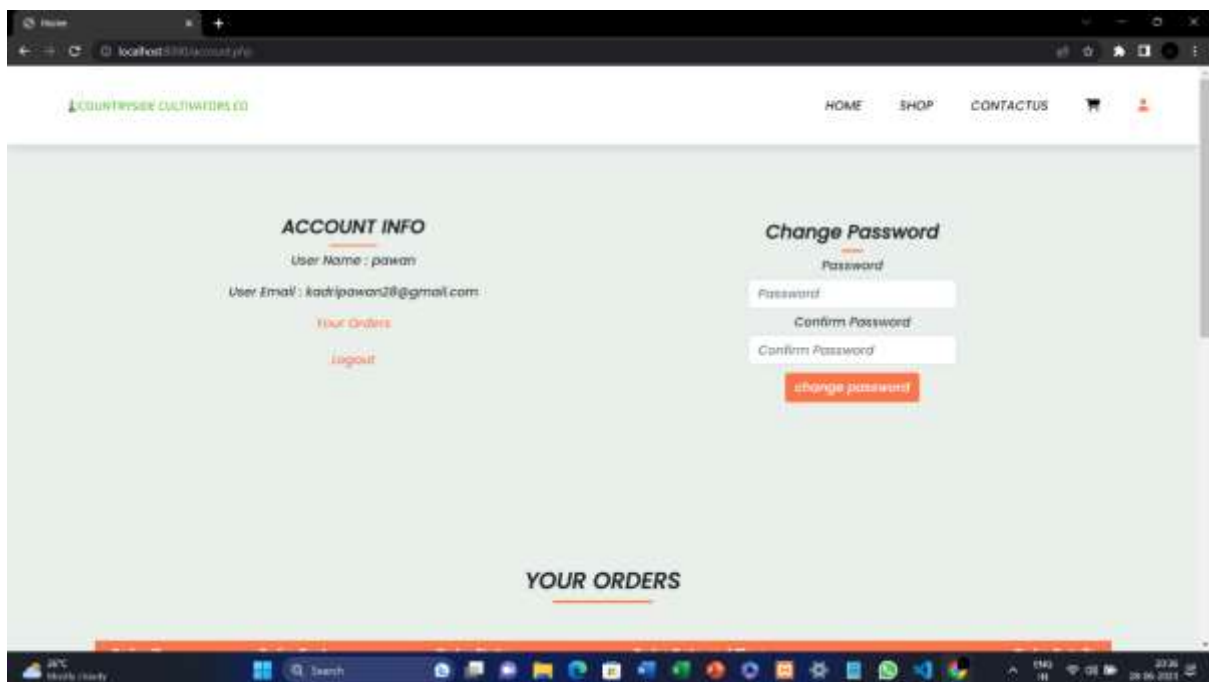
Single Product: One single product with its description and other images are displayed in this page.



Cart: Product added from shop is available in cart and user can check out.



Account: All details and orders of user is displayed in this page.



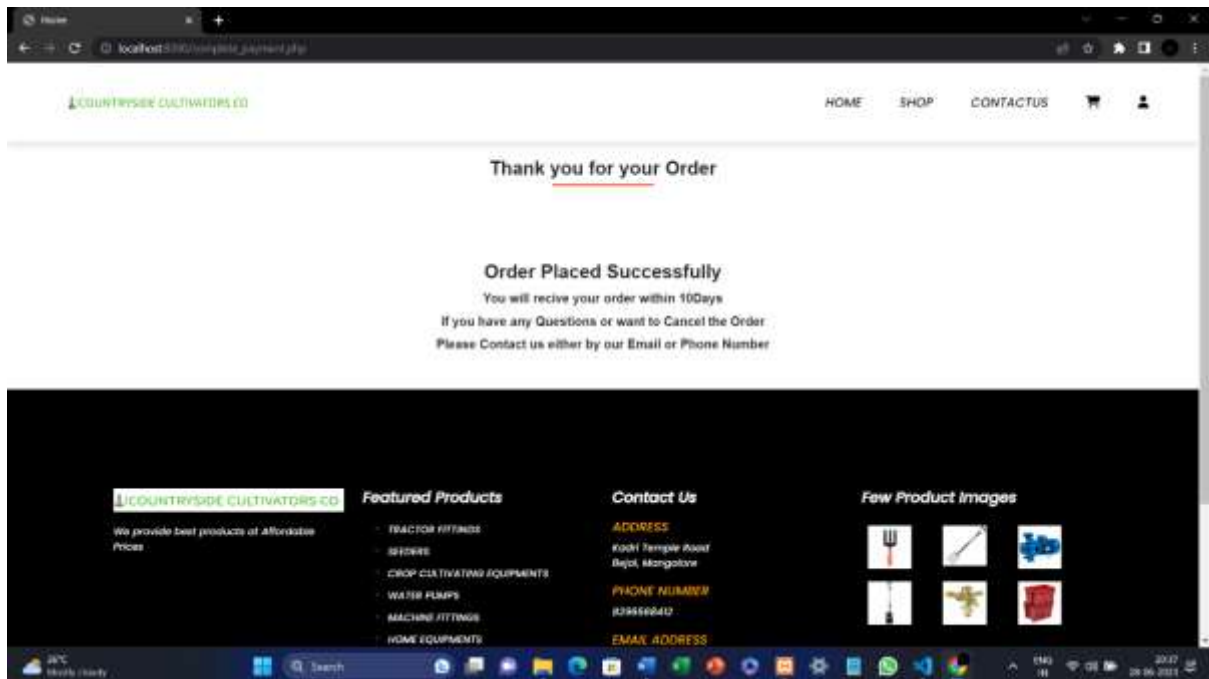
Checkout: The user can confirm his order by checking out by placing the order.

The screenshot shows a web browser window displaying the 'CHECK OUT' page of the Countryside Cultivators Co website. The page has a light green background. At the top, there is a navigation bar with links for HOME, SHOP, and CONTACTUS, along with a shopping cart icon and a user profile icon. The main content area is titled 'CHECK OUT' in bold, underlined text. Below the title, there are five form fields arranged in two columns: 'Name' (with a placeholder 'Enter Name'), 'Email' (with a placeholder 'Enter Email Address'), 'Phone' (with a placeholder 'Enter Phone Number'), 'City Name' (with a placeholder 'Enter City Name'), and 'Shipping Address' (with a placeholder 'Enter Shipping Address'). To the right of these fields, the 'Total Amount Rs.1200' is displayed. At the bottom right of the form area, there is an orange button labeled 'place order'. The browser's address bar shows the URL 'localhost:5173/checkout.php'. The Windows taskbar at the bottom indicates the system time as 22:26 on 28-10-2021.

Payment: The user can confirm payment by clicking cash on delivery button.

The screenshot shows a web browser window displaying the 'PAYMENT' page of the Countryside Cultivators Co website. The page has a light green background. At the top, there is a navigation bar with links for HOME, SHOP, and CONTACTUS, along with a shopping cart icon and a user profile icon. The main content area is titled 'PAYMENT' in bold, underlined text. Below the title, the 'Total Payment Rs.1200' is displayed. Underneath, the text 'Cash On Delivery' is shown, followed by a blue button labeled 'cash on delivery'. The footer of the page contains four sections: 'COUNTRYSIDE CULTIVATORS CO' with the tagline 'We provide best products at Affordable Prices', 'Featured Products' with a list of items (TRACTOR FITTINGS, SEEDERS, CROP CULTIVATING EQUIPMENTS, WATER PUMPS), 'Contact Us' with 'ADDRESS' (Kadi Temple Road, Rajol, Mangalore) and 'PHONE NUMBER', and 'Few Product Images' with three small images of agricultural equipment. The browser's address bar shows the URL 'localhost:5173/payment.php?order_id=1200&order_id=1200&order_id=1200'. The Windows taskbar at the bottom indicates the system time as 22:27 on 28-10-2021.

Complete Payment: The order confirmation message is displayed here.



9. CONCLUSION AND FUTURE ENHANCEMENT

9.1 Conclusion:

The idea behind this project is to provide farmers and daily home gardeners reliable products. The E-commerce System website allows the user to login to the website, if the user does not have an account, he can register and the login to the website to purchase any products. The user can add products to his cart, change the password of his account etc.

On the other side admin (Superuser) he can add products, edit the products, edit the images, delete the products and orders.

This application will only provide quality products to the farmers for a very reasonable price.

9.2 Future Enhancement:

The project currently is developed as a web application. Although it can be accessed by any portable devices like mobile, tablets etc. through browsers. In future we are planning to add online payment methods, provide different types of seeds and even provide various types of fertilizer sprays and we will be available on social media platforms.

Bibliography

1. Title: "Computer Networks"
Author: Andrew S. Tanenbaum and David J. Wetherall

2. Title: "Database Management Systems"
Author: Raghu Ramakrishnan and Johannes Gehrke

3. Title: "Web Technologies: HTML, CSS, JavaScript, XML, AJAX, and PHP"
Author: P. J. Deitel, H. M. Deitel, and T. R. Nieto

Websites:

- <https://drawio.com>
- <https://youtube.com/@learnwithdevil>
- <https://www.w3schools.com/>
- <https://github.com/>
- <https://stackoverflow.com/>
- <https://www.geeksforgeeks.org/>