

Saplingo- An Online Portal For Sapling Sale

Mini-Project report submitted to
Kristu Jyoti College of Management and Technology,
Changanacherry

In partial fulfillment of the requirements for the award of the

BACHELOR OF COMPUTER APPLICATION

BY

ARUN SURESH (210021088937)

Under the guidance of
MS. SUNANDHA RAJAGOPAL



DEPARTMENT OF COMPUTER APPLICATIONS

**KRISTU JYOTI COLLEGE OF MANAGEMENT
AND TECHNOLOGY, CHANGANACHERRY
NOVEMBER, 2023**

Kristu Jyoti College of Management and Technology, Changanacherry

DEPARTMENT OF COMPUTER APPLICATION



CERTIFICATE

Certify that the report entitled “*Saplingo- an online portal for sapling sale*” is a bonafide record of the mini-project work done by *Arun Suresh* under our guidance and supervision is submitted in partial fulfillment of the Bachelor's Degree in Computer Applications, awarded by Mahatma Gandhi University Kerala and that no part of this work has been submitted earlier for the award of any other degree.

Ms. Sunandha Rajagopal
Project guide

Mr. Roji Thomas
HOD

External Examiner

Internal Examiner

DECLARATION

I hereby declare that the project work entitled “**SAPLINGO- AN ONLINE PORTAL FOR SAPLING SALE**” submitted to Mahatma Gandhi University in partial fulfillment of requirement for the award of degree of Bachelor of Computer Application from Kristu Jyoti College of Management and Technology, Changanacherry, is a record of bonafide work done under the guidance of **Ms. SUNANDHA RAJAGOPAL, project guide, Kristu Jyoti of Management and Technology, Changanacherry**. This project has not been submitted in partial or fulfillment of any other degree/diploma/fellowship or similar of this university or any other university.

ARUN SURESH (210021088937)

Place: Kottayam

Date: 11 January 2024

ACKNOWLEDGEMENT

I would like to express our thankfulness towards many people who have been instrumental in making this project. First of all, I thank the college management who has been with me and provided the most helpful facilities throughout the completion of the project. I am greatly indebted to Rev. Fr. **Joshy Cheeramkuzhy CMI, Principal Kristu Jyoti College**, who wholeheartedly permitted us and whose advice was a real encouragement. I am deeply indebted to **Ms. Sunandha Rajagopal, Project Guide**, for the valuable discussion and helpful counseling. I am sincerely indebted to **Asst. Prof. Mr. Roji Thomas, HOD, Department of Computer Applications**, for the valuable guidance. Last but not least, I wish to express my gratitude and heartfelt thanks to my friends for their valuable advice, encouragement, and support for the successful completion of the project. Above all, I am thanking God Almighty.

ABSTRACT

Project Topic: Saplingo - An Online Platform For Sapling Sale

“Saplingo” is the complete platform to develop the culture of organic farming and stimulate its growth by providing correct gross profit for the farmers. This application provides a completely user-friendly experience at first glance. A user who has very limited technical knowledge can easily understand the software behaviors and can buy and sell products from this platform.

Coming to the working of this platform I will allow a user login like other social media platforms. Then the user can view the feed of all saplings that are to be sold. If they are interested then he/she can instantly buy any product like other E-Commerce applications. After ordering, the product will be delivered to the respective buyer.

In conclusion, this project delves into the dynamic world of e-commerce, focusing specifically on the online sales of saplings. It explores the intricacies of running a successful online sapling sales platform, addressing key aspects such as website design, marketing strategies, customer engagement, and sustainability practices. The project analyzes the challenges and opportunities within this niche market, considering the growing global interest in reforestation and sustainable living. By examining real-world case studies and employing market research techniques, students will gain valuable insights into the sustainable e-commerce landscape and the potential impact of their actions on the environment.

INDEX

Chapter 1: Introduction	[8-10]
➤ Introduction	
➤ Problem Statement	
➤ Scope and Relevance of the Project	
➤ Objectives	
Chapter 2: System Analysis	[11-16]
➤ Introduction	
➤ Existing System	
o Limitations of Existing System	
➤ Proposed System	
o Advantages of the proposed System	
➤ Feasibility Study	
o Technical Feasibility	
o Operational Feasibility	
o Economic Feasibility	
➤ Software Engineering Paradigm Applied	
Chapter 3: System Design	[17-26]
➤ Introduction	
➤ Database Design o Entity Relationship Model	
➤ Process Design –Dataflow Diagrams	
➤ Object Oriented Design – UML Diagrams	
o Activity Diagram	
o Sequence Diagram	
o Use Case Diagram	
➤ Input Design	
➤ Output Design	

Chapter 4: System Environment	[27-32]
➤ Introduction	
➤ Software Requirements Specification	
➤ Hardware Requirements Specification	
➤ Tools, Platforms	
o Front End Tool o Back End Tool	
o Operating System	
Chapter 5: System Implementation	[33-66]
➤ Introduction	
➤ Coding	
o Sample Codes	
o Code Validation and Optimization	
➤ Unit Testing	
Chapter 6: System Testing	[67-70]
➤ Introduction	
➤ Integration Testing	
➤ System Testing	
o Test Plan & Test Cases	
Chapter 7: System Maintenance	[71-73]
➤ Introduction	
➤ Maintenance	
Chapter 8: Future Enhancement and Scope of further Development	[74-76]
➤ Introduction	
➤ Merits of the System	
➤ Limitations of the System	
➤ Future Enhancement of the System	
Chapter 9: Conclusion	[77-78]
Chapter 10: Bibliography	[79-80]
Appendix	[81-89]

INTRODUCTION

CHAPTER 1

INTRODUCTION

In an era where the preservation of our environment stands as a collective responsibility, "Saplingo" emerges as a beacon of change, introducing a novel approach to foster green living. This project, aptly named "Saplingo: An online portal for sapling sale," addresses the growing need for accessible and convenient avenues to contribute to reforestation and personal green spaces.

"Saplingo" is an innovative online platform that bridges the gap between nature enthusiasts and the beauty of sustainable living. This project is dedicated to simplifying the process of acquiring saplings, making it not just a transaction but a seamless journey towards cultivating life.

In this project report, I delve into the intricacies of developing and implementing "Saplingo," an e-commerce platform that not only facilitates the purchase of saplings but also educates users about the significance of tree cultivation. Through this portal, users can explore a diverse array of saplings, learn about their unique characteristics, and effortlessly bring a piece of nature into their homes.

I will navigate through the key features, functionalities, and technology stack employed in creating "Saplingo," shedding light on the meticulous design choices aimed at ensuring user-friendly interactions. Furthermore, this report will showcase the project's commitment to environmental sustainability, highlighting the incorporation of eco-friendly practices in the packaging and delivery of saplings. The project is executed with two modules:

- User Module
- Admin Module

User Module

A Saplingo user can log in to his account and view the shop, he/she can purchase saplings and can post a product into the shop using the post a product option.

PROBLEM STATEMENT

Since the world is entirely reliant on the internet, and with everyone engrossed in their daily tasks, there is an urgent need to develop software that facilitates users in effortlessly ordering their saplings without unnecessary delays. The traditional sapling acquisition process involves a cumbersome and time-consuming approach. Individuals are required to visit nurseries, investing considerable time contemplating what saplings to acquire, leading to fatigue. At times, they may not even secure their preferred saplings or nursery. The Sapling Ordering Service is a revolutionary software that offers users the convenience of acquiring saplings without enduring lengthy queues or exhausting decision-making processes.

SCOPE AND RELEVANCE

Users can effortlessly purchase sapling products directly without the need to travel or leave their homes. The admin can seamlessly add their saplings with the corresponding category, and customers can easily navigate through the options, confirming their sapling selections independently. The traditional manual process of acquiring saplings involves a time-consuming and cumbersome approach. The Sapling Ordering Service streamlines this process, providing a convenient and efficient way for users to access a variety of saplings without the need for physical presence or elaborate procedures.

OBJECTIVE

The primary goal of this project is to create a software 'sapling ordering system,' offering users the convenience to purchase saplings online without the need to travel or expend unnecessary time.

SYSTEM ANALYSIS

CHAPTER 2

SYSTEM ANALYSIS

INTRODUCTION

System Analysis is a structured process of identifying and solving problems. The performance, management, and documentation of the activities related to the four phases of the life cycle of a business information system.

The analysis involves the requirement determination and specifications. It involves establishing all the system elements and then mapping these requirements to the software form. The analysis is intended to capture and describe all the requirements of the system and to make a model that defines a key domain class in the system. The purpose is to provide an understanding and to enable communication about the system between the developers and the people establishing the requirements. Therefore the analysis is typically in terms of code or programs during this phase. It is the first step towards really understanding the requirements.

EXISTING SYSTEM

This system helps in enhancing our traditional handmade products to publish in the market. The system helps buyers to search for the product in their fingertips. It provides an easy way to buy the product directly from the merchants without the interface of any third party. It also helps rural people to sell their products to the market and to earn money to improve their status.

LIMITATIONS OF EXISTING SYSTEM

- **Limited Customization:** Some systems may have limitations on menu customization, making it challenging for customers to tailor sapling services to their specific preferences or gardening requirements.
- **Lack of Real-Time Updates:** If the system lacks real-time updates or synchronization, there might be issues with inventory management, leading to situations where items are displayed as available when they are actually out of stock.
- **Complexity for Users:** If the user interface is complex, it may pose a challenge for customers trying to navigate the system, potentially leading to user frustration and errors in the ordering process.

- **Inadequate Communication Channels:** If the communication channels between customers and catering providers are limited or inefficient, it could result in misunderstandings, delays, or missed details regarding menu choices, delivery times, or special requests.
- **Scalability Issues:** Some systems may face scalability challenges, struggling to handle a high volume of orders or a large number of concurrent users during peak times. This can lead to slower response times and potential service disruptions.
- **Security Concerns:** Inadequate security measures may expose the system to data breaches, risking the compromise of sensitive customer information, such as personal details and payment data.

PROPOSED SYSTEM

In the envisioned system, we aimed to enhance security compared to the current system. The proposed system addresses numerous drawbacks of the existing system, allowing customers to purchase saplings online without intermediary assistance directly. Here, the admin can seamlessly add saplings with corresponding categories and update them. The system is entirely computerized, ensuring robust security and fostering a more user-friendly environment than its predecessor. This system offers convenience to both users and admins in the realm of sapling management.

ADVANTAGES OF THE PROPOSED SYSTEM

- **User-Friendly Interface:** A well-designed and intuitive interface can enhance the user experience, making it easy for customers to browse menus, place orders, and navigate through the website.
- **Personalized User Accounts:** Implementing user accounts enables customers to save their preferences, order history, and payment details, streamlining the ordering process and providing a more personalized experience.
- **Efficient Order Processing:** Advanced systems can improve order processing efficiency, reducing the time it takes for customers to place orders. This can lead to higher customer satisfaction and increased orders.

- **Real-Time Order Tracking:** The ability for customers to track their orders in real-time provides transparency and helps manage expectations, leading to a better overall customer experience.
- **Menu Customization and Special Requests:** Allowing customers to customize their orders and make special requests ensures that their preferences are accommodated, making the ordering process more flexible and customer-centric.

FEASIBILITY STUDY

In any Project, feasibility analysis is a very important stage, here the project is checked for its feasibility. Any project may face scarcity in resources, time, or workforce. Hence all these are to be studied in detail and a conclusion should be drawn whether the project under consideration is feasible or not. This analysis is a test of the proposed project, regarding its workability, and impact on users and clients. Feasibility and risk involved are inversely related to each other. The main objective of feasibility is to test the technical, operational, and economic feasibility of a project.

Technical Feasibility

Technical Feasibility centers around the existing computer system and to what extent it can support the proposed addition. Technical considerations evaluate existing hardware and software. This involves financial considerations to accommodate technical enhancements. Evaluating the technical feasibility is the trickiest part of a feasibility study. This is because, at this point, not too many detailed designs of the system, making it difficult to access issues like performance, cost, etc. Several issues have to be considered while doing a technical analysis.

Operational Feasibility

Operational Feasibility is considered with the working of the system after its installation. This system can be installed in the client environment and the developers will help in the maintenance of the system in the future. Proposed projects are beneficial only if they can be turned into information systems that will meet the organization's operating requirements, this test of feasibility asks if the system will work when it is developed and installed.

Economic Feasibility

The economic feasibility of Saplingo, an innovative online platform for selling saplings, is robust and promising, driven by its commitment to delivering premium saplings and related accessories to a diverse customer base.

The carefully curated catalog, encompassing a wide range of cutting-edge plant varieties, positions Saplingo as a one-stop destination for gardening enthusiasts and casual plant lovers alike. By focusing on an intuitive and user-friendly interface, Saplingo enhances customer convenience, allowing plant enthusiasts to make well-informed purchasing decisions.

The emphasis on detailed sapling descriptions, expert reviews, and comparison tools ensures that consumers can explore the latest advancements in the gardening industry, leading to increased customer satisfaction and loyalty. With an expanding market of gardening-savvy consumers, Saplingo is well-positioned to capitalize on the growing demand for high-quality plant varieties, making it economically feasible. Moreover, by offering a seamless online shopping experience, Saplingo is poised to attract a significant customer base, contributing to its economic viability and long-term success in the competitive online plant retail landscape.

SOFTWARE ENGINEERING PARADIGM

This establishment and use of sound engineering principles to obtain economically developed software that is reliable and works efficiently on real machines is called software engineering.

Software engineering is the discipline whose aim is:

1. Production of quality software.
2. Software that is delivered on time.
3. Cost within the budget.
4. Satisfies all requirements.

A software process is how we produce the software. Apart from hiring smart, knowledgeable engineers and buying the latest development tools, an effective software development process is also needed, so that engineers can systematically use the best technical and managerial practices to complete their projects successfully.

A **software life cycle** is the series of identifiable stages that a software product undergoes during its lifetime. A software lifecycle model is a descriptive and diagrammatic representation of the software life cycle.

A life cycle model represents all the activities required to make a software product transit to life cycle phases. It also captures the order in which these activities are to be taken.

LIFE CYCLE MODELS

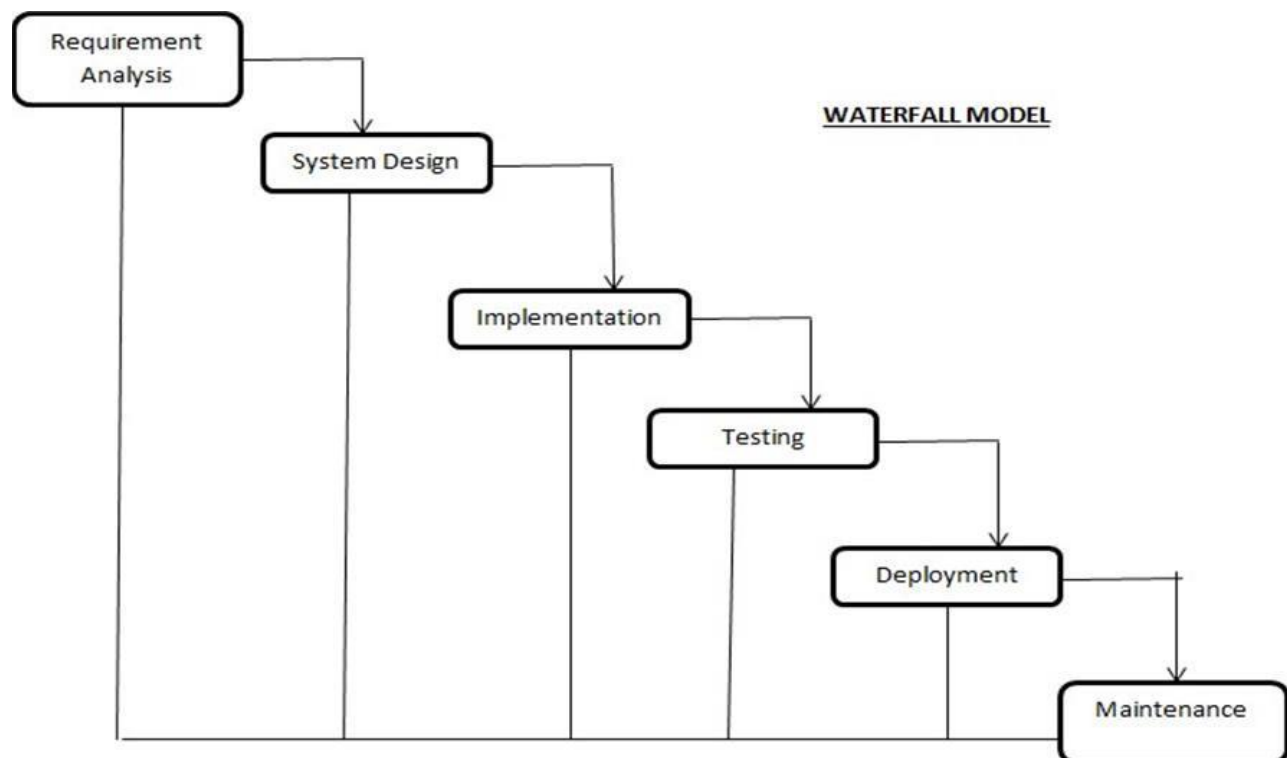
There are various Life Cycle Models to improve the software processes.

1. Waterfall Model
2. Prototype Model
3. Iterative Enhancement Model
4. Evolutionary Model
5. Spiral Model

In this project **waterfall model** is followed.

The waterfall approach was the first SDLC Model to be used widely in software engineering to ensure the success of the project. In the “Waterfall” approach, the whole process of software development is divided into separate phases. In this Waterfall model, typically, the outcome of one phase acts as the input for the next phase sequentially.

The following illustration is a representation of the different phases of the Waterfall Model:



SYSTEM DESIGN

CHAPTER 3

SYSTEM DESIGN

INTRODUCTION

System design is the solution to the creation of a new system. This is an important aspect made up of several steps. System design is the process of developing specifications for a candidate system that meet the criteria established in the system analysis. A major step in system design is the preparation of the input forms and output reports in a form applicable to the users.

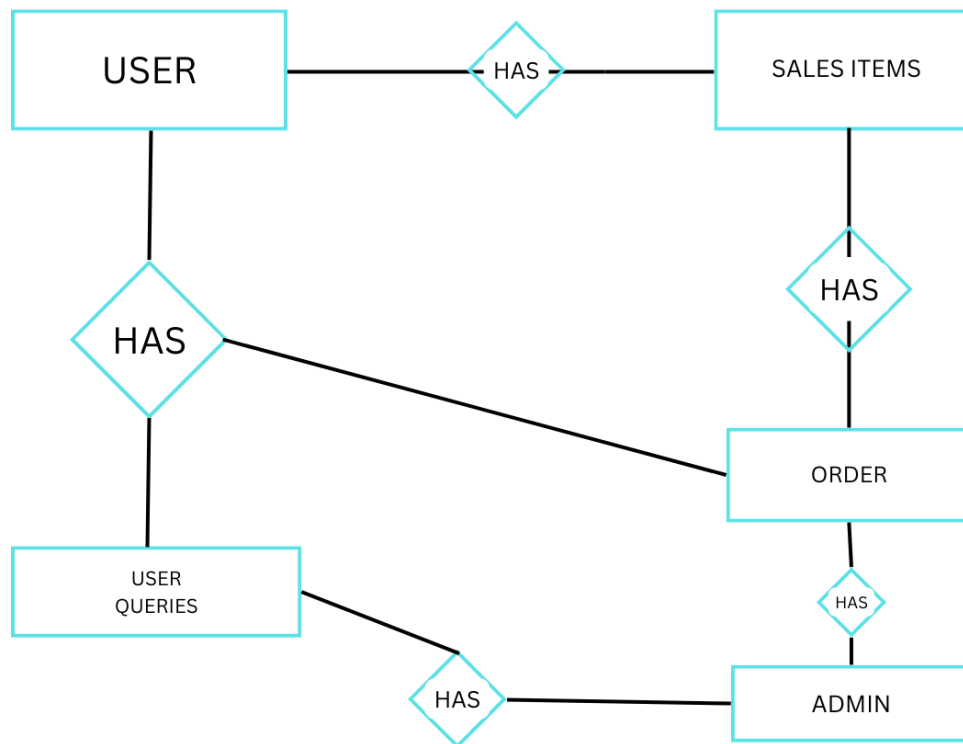
The main objective of system design is to use the package easily by a computer operator. System design is the creative act of invention, developing new inputs, a database, offline files, methods, procedures, and output for processing business to meet an organization's objective. System design is built on information gathered during the system analysis.

The complete, efficient, and successful system should provide the following in successions:

- ✚ From where should we start
- ✚ Where we have to go
- ✚ Where should we stop

If the project is to be successful, we will need to answer these questions. The answer to these questions is schema manner known as system design. A systematic manner will be followed to achieve beneficial results at the end. It involves starting with a vague idea and ultimately developing it into a useful system. The design phase is transition from a user-oriented to document-oriented for the programmers. Software reports can be broken into a series of steps starting with the basic ideas and ending with the finished projects.

DATABASE DESIGN (ENTITY-RELATIONSHIP MODEL)



PROCESS DESIGN (DATA FLOW DIAGRAM)

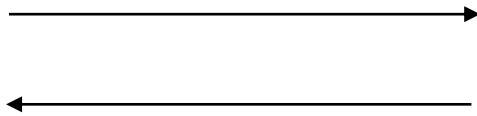
A data flow diagram (DFD) is a graphical representation of the “flow” of data through an information system, modeling its process aspects. Often they are a preliminary step used to create an overview of the system which can later be elaborated. DFDs show what kind of information will be input to and output from the system, where the data will come from and go to, and where the data will be stored. It does not show information about the timing of the processes, or information about whether processes will operate in sequence or in parallel.

DFD SYMBOLS

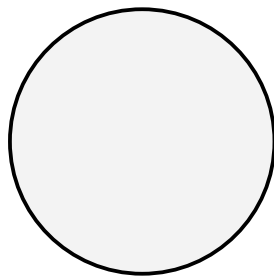
1. A **SQUARE** defines a source or destination of system data.



2. An **ARROW** identifies data flow or data in motion. It is a pipeline through which information flows.



3. A **CIRCLE** or a **BUBBLE** represents a process that transforms incoming data flow into outgoing data flow.



4. An **OPEN RECTANGLE** is a data store or data at rest or a temporary test repository of data.



Note that the DFD describes what data flows rather than how they are processed, so it does not depend on hardware, software data structure, or file organization. One of the tools of structured analysis is the diagram. A data flow diagram is a graphical representation of the system. The analyst can use a data flow diagram to explain this understanding of the system.

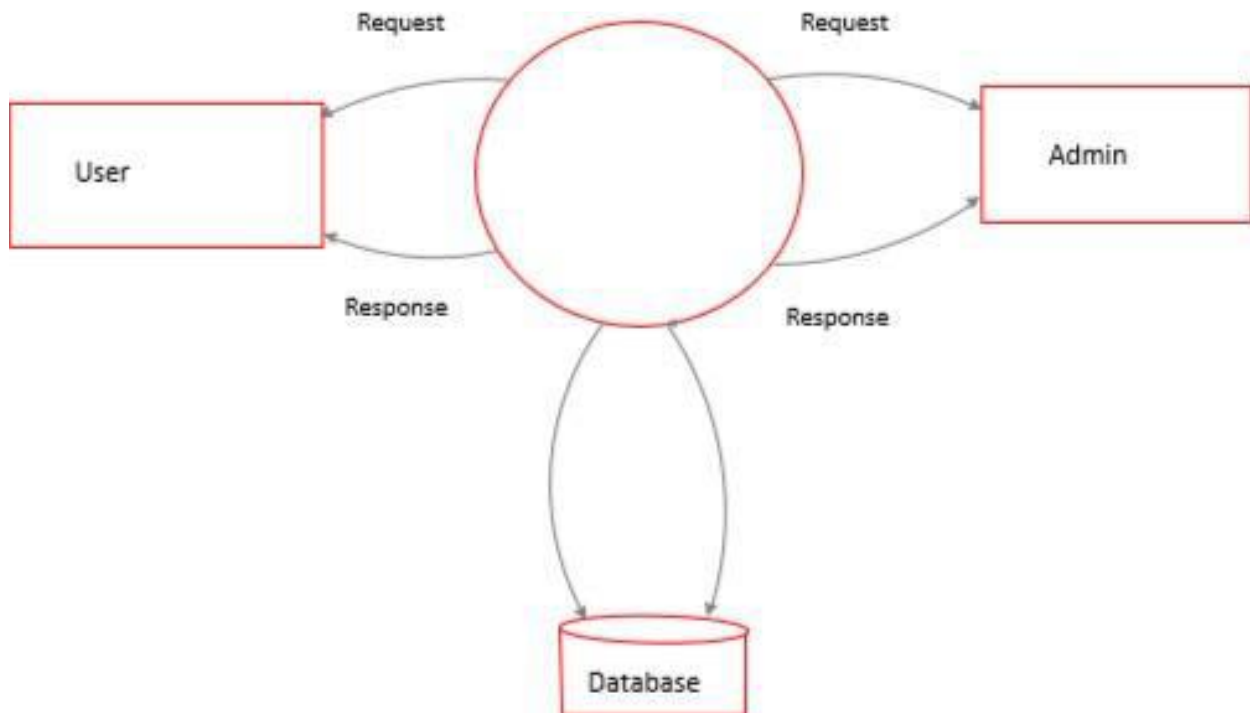
- ☐ Data flows are an initiative way of showing how data is processed by a system.
- ☐ Data flow models are used to show how data flows through a sequence of processing steps.

Four steps are commonly used to construct a DFD:

1. The process should be named and numbered for easy reference.
2. The direction of flow is from top to bottom and from left to right.
3. When a process is exploded into lower level details, they are numbered.
4. The name of data stores, sources and destinations are written in capital letters.

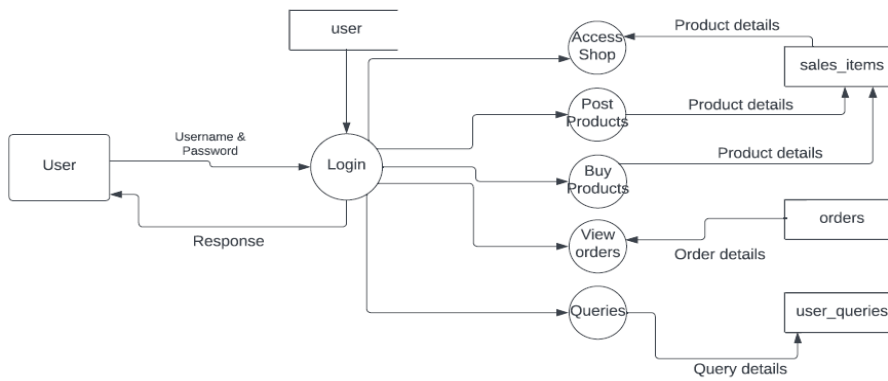
DATA FLOW DIAGRAM

LEVEL 0



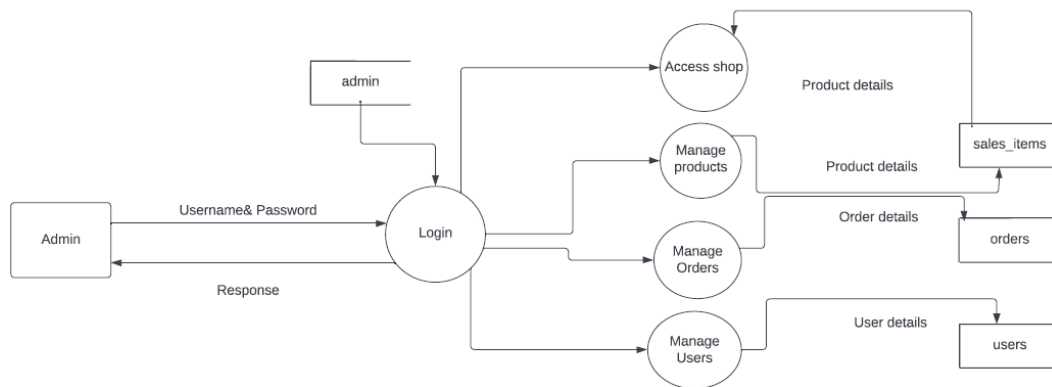
LEVEL 1

USER

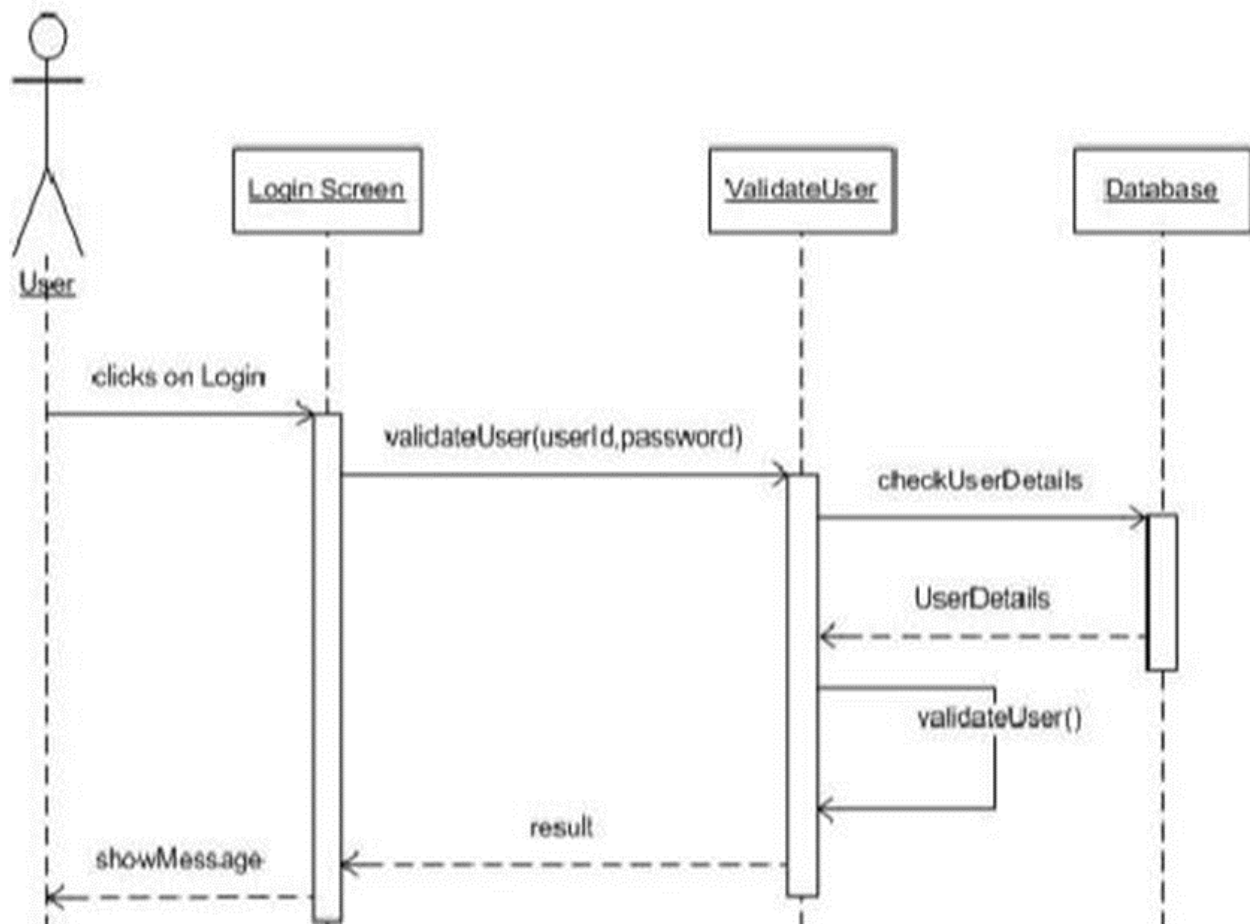


LEVEL 2

ADMIN



SEQUENCE DIAGRAM



TABLES

ORDER

Name	Data Type	Constraints
o_id	int	Primary Auto increment
Item_id	int	Reference key
uid	int	Reference key
firstname	text	
lastname	text	
location	text	
phone	bigint	
email	varchar	
totalprice	double	

status	text	
dateOfOrder	date	
timeOfOrder	date	

SALES_ITEMS

Name	Data Type	Constraints
SI_id	int	Auto increment Primary key
uid	int	Reference key
Simage	long blob	
quantity	int	
description	longtext	
unit_price		

USER

Name	Data Type	Constraints
uid	int	Auto increment Primary key
name	text	
email	varchar	
username	varchar	
password	varchar	
status	int	

ADMIN

Name	Data type	Constraints
Aid	int	Auto increment Primary key
username	varchar	
password	varchar	

USER QUERIES

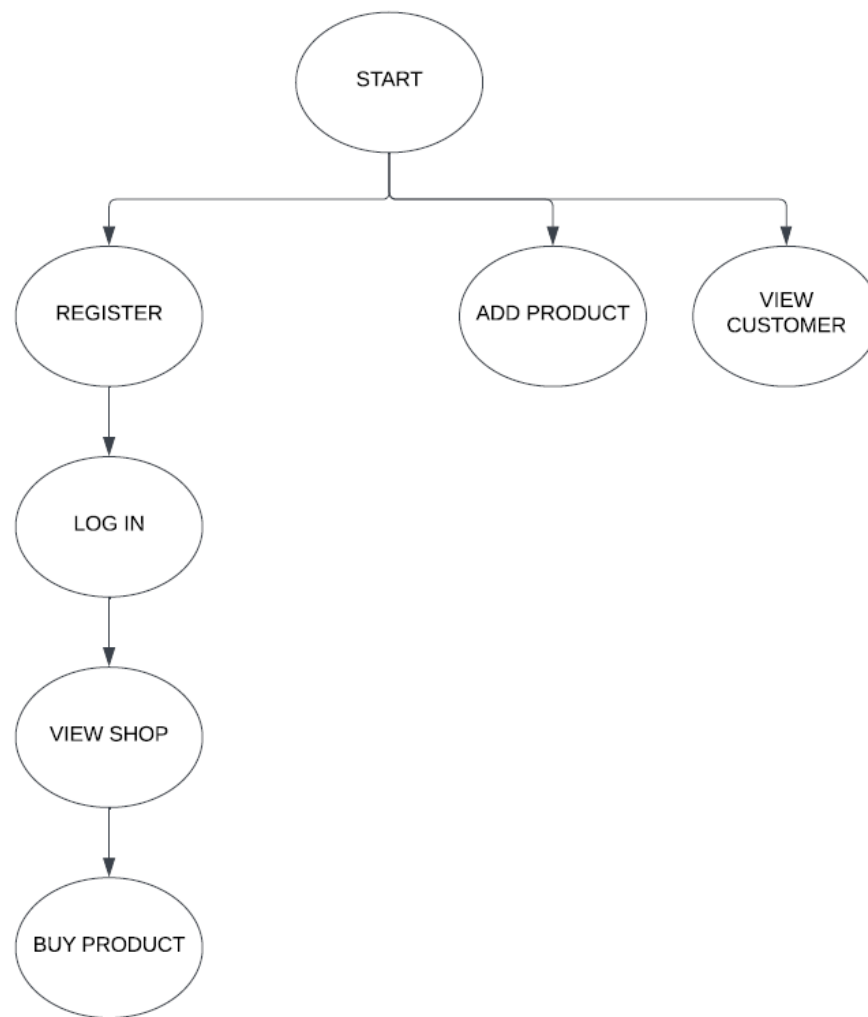
Name	Data Type	Constraints
name	text	
email	varchar	
mobile_no	int	
queries	varchar	
q_id	int	Auto increment Primary key

OBJECT ORIENTED DESIGN (UML DIAGRAM)

UML, short for Unified Modeling Language, is a standardized modeling language consisting of an integrated set of diagrams, developed to help system and software developers specify, visualize, construct, and document the artifacts of software systems, as well as for business modeling and other non-software systems. The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems. The UML is a very important part of developing object-oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects. Using the UML helps project teams communicate, explore potential designs, and validate the architectural design of the software.

Activity Diagram

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration, and concurrency. It describes the flow of control of the target system, such as exploring complex business rules and operations, describing the use case also the business process.



INPUT DESIGN

Input design is the processing of converting the user-oriented description of the inputs of the system. The goal of designing input data is to make data entry as easy logical and free from errors as possible. When we approach input data design, we design source documents that capture the data and then select the media used to enter them into the computer.

OUTPUT DESIGN

The primary consideration in the design of all output is the information requirement and another objective of the users. It is the most important and direct source of information to the user. A major form of output is hardcopy. Print output should be designed around the output requirement of the user. Each output should be given a specific name or title. The output data is displayed on the visual display unit and output can be redirected to printers and or sorted in a file for later use.

SYSTEM ENVIRONMENT

CHAPTER 4

SYSTEM ENVIRONMENT

An important matter in building an application is selecting hardware and software. The hardware drives the software to facilitate solutions. Factors like cost performance and reliability etc are taken into consideration during the purchase of the hardware component for any computerized system.

INTRODUCTION

In computer software, an operating environment or integrated applications environment is the environment in which users run application software. The environment consists of a user interface provided by an applications manager and usually an application programming interface (API) to the applications manager.

An operating environment is usually not a full operating system but is a form of middleware that rests between the OS and the application. For example, the first version of Microsoft Windows, Windows 1.0, was not a full operating system, but a GUI laid over DOS albeit with an API of its own. Similarly, the IBM U2 system operates on both Unix/Linux and Windows NT. Usually, the user interface is text-based or graphical, rather than a command-line interface (e.g., DOS or the Unix shell), which is often the interface of the underlying operating system.

SOFTWARE REQUIREMENT SPECIFICATION

The system requirement specification is a technical specification of requirements for the product. The goal of the system requirement definition is to completely specify the technical requirements for the software product concisely and unambiguously. A software requirements specification (SRS) is a description of a software system to be developed. It lays out functional and non-functional requirements and may include a set of use cases that describe user interactions that the software must provide.

Software requirements:

Operating System: Microsoft Windows XP / above

Platform: Visual Studio 2022

Front End : PHP

Back End : MYSQL Server

The application must inform the user with a proper message when there is an error in data entry or when an activity is successfully completed.

HARDWARE REQUIREMENT SPECIFICATION

This describes the logical and physical characteristics of each interface between the software product and hardware components of the system.

Processor: Intel Pentium Dual Core / above

Hard Disk Space: 80 GB or more

Ram: 1GB or more so that data access can be made faster.

Display: 14.1 Color Monitor (LCD, CRT or LED)

Clock Speed: 2.24 GHz

TOOLS, PLATFORM

The front-end tool used is PHP and the back-end tool used is My SQL, PHP is a scripting language.

FRONT END TOOL

PHP is a dynamically typed general-purpose scripting language that can be embedded in HTML pages. It was designed in 1995 for implementing dynamic web pages, its name initially standing for Personal Home Pages. PHP now stands for the recursive acronym PHP: Hypertext Preprocessor. While predominantly used for server-side scripting PHP can be used for writing command-line scripts and client-side GUI applications. It may also be embedded into host applications to provide them with scripting functionality. The main implementation of PHP is free open-source software. This provides the de facto definition of the syntax and semantics for the language since there is no formal specification. While the end user interface and extensions API for the PHP interpreter are well documented, scan documentation exists for the internals of the parser and the Zend engine (which interprets scripts). The lexical analyzer is defined using a Flex description of 2000 lines, and the parser has a Bison specification of 900 lines. It is difficult to extract any formal specification from either source file because of unnecessary redundancy and poor structuring of the grammar convoluted rules which attempt to enforce static checks embedded semantic actions trying to ensure associativity is appropriately maintained at certain points during parsing and the requirement for tight coupling between the lexical analyzer and parser due to in-string syntax which is not amenable to processing using the traditional lexical analysis and parser interface.

BACK END TOOL

MS-SQL SERVER

Microsoft SQL Server features include:

Internet Integration

The SQL Server 2008 database engine includes integrated XM support. It has the scalability, availability, and security features required to operate as the data storage component of the largest Web sites.

Scalability and Availability

The same database engine can be used across platforms ranging from laptop computers running Microsoft Windows 98 to large, multiprocessor servers running Microsoft Windows 2008 Data Center Edition. SQL Server 2008 Enterprise Edition supports features such as federated servers, indexed views, and large memory support that allow it to scale to the performance levels required by the largest Web sites.

Enterprise-Level Database Features

The SQL Server 2008 relational database engine supports the features required to support demanding data processing environments. The database engine protects data integrity while minimizing the overhead of managing thousands of users concurrently modifying the database.

Ease of installation, deployment, and use of SQL Server 2008 includes a set of administrative and development tools that improve upon the process of installing, deploying, managing, and using SQL Server across several sites. SQL Server 2008 also supports a standards-based programming model integrated with the Windows DNA, making the use of SQL Server databases and data warehouses a seamless part of building powerful and scalable systems.

Data Warehousing

SQL Server 2008 includes tools for extracting and analyzing summary data for online analytical processing. Server also includes tools for visually designing databases and analyzing data using English-based questions.

Advantages of SQL Server 2008 as a Database Server

Microsoft SQL Server 2008 is capable of supplying the database services needed by extremely large systems. Large servers may have thousands of users connected to an instance of SQL

Server 2008 at the same time. SQL Server 2008 has full protection for these environments, with safeguards that prevent problems, such as having multiple users trying to update the same piece

of data at the same time. SQL Server 2008 also allocates the available resources effectively, such as memory, network bandwidth, and disk I/O, among the multiple users.

Extremely large Internet sites can partition their data across multiple servers, spreading the processing load across many computers, and allowing the site to serve thousands of concurrent users.

Multiple instances of SQL Server 2008 can be run on a single computer. SQL Server 2008 applications can run on the same computer as SQL Server 2008.

Information Representation

In SQL Server, data is represented in terms of rows and columns of a table. Data stored as a table can be easily visualized.

Unique definition of rows

In SQL Server we can make each row of a table unique by using a feature called constraints.

Guaranteed access

In SQL Server, data that is stored across tables in one or more databases can be combined using the query.

Systematic treatment of null values

SQL Server like most RDBMS treats Null values, zeros, and blanks differently. While creating a table, one can specify whether a field allows Null values or not.

High-level update, insert and delete

In SQL Server if a record is updated or deleted in the Master table, the corresponding record in the table is also updated or dropped.

Features of SQL Server 2008

Manage any data, any place, and any time. Store data from structured, semi-structured, and unstructured documents, such as images, and rich media, directly within the database. SQL Server 2008 delivers a rich set of integrated services that enable to do more with data such as query, search, synchronize, report, and analyze. Easy installation: SQL SERVER 2008 provides a set of administrative and development tools for easy installation, deployment, and management.

Integration with internet: SQL SERVER has a database engine which is a core integral part of the RDBMS and stores data in a table. Scalability and Availability. Support for client/server Model: SQL SERVER is designed to follow the client/server model. Operating System Compatibility:

Since SQL SERVER 2008 is a Microsoft product, it is compatible with Windows NT Server 4, Windows 2000, Windows 7, Windows XP, etc. Business service: This layer is known as the application server tier, which is a layer between the user interface and database. Data Services: This layer deals with interacting with the actual data source. The four functional areas in a data service layer are: retrieving data, inserting data, updating data, and deleting data.

SYSTEM IMPLEMENTATION

CHAPTER 5

SYSTEM IMPLEMENTATION

INTRODUCTION

Implementation is the stage of the project where the theoretical design is turned into a working system. At this stage, the main workload, the greatest upheaval, and the major impact on the existing system shifts to the user department. If the implementation is not carefully planned and controlled it can confuse. Implementation includes all those activities that take place to convert from the old system to the new system. The new system may be new, replacing an existing manual or automated system or it may be a major modification to an existing system. Proper implementation is essential to provide a reliable system to meet the organization's requirements. Successful implementation may not guarantee improvement in the organization using the new system, but improper installation will prevent it.

The implementation stage involves the following tasks:

- ✚ Careful planning
- ✚ Investigation of the system and constraints
- ✚ Design of methods to achieve the changeover phase
- ✚ Training of staff in the changeover phase
- ✚ Evaluation of the changeover method, the method of implementation and the time scale to be adopted are found out initially. Next, the system is tested properly and at the same time users are trained in the new procedures.

CODING

CODING STANDARDS

Once the system was tested, the implementation phase started. A crucial phase in the system development life cycle is the successful implementation of a new system design. Implementation simply means converting a new system design into operation. This is the moment of truth the first question that strikes everyone's mind is whether the system will be able to give all the desired results as accepted by the system. Before starting the project implementation process project must have completed the project evaluation process and the project has been approved for implementation.

The project evaluation process includes performing a needs analysis and architecture review. The implementation phase of the software design consists of different tasks to be done sequentially to obtain the desired result.

IMPLEMENTATION OF METHODOLOGY

An implementation methodology is a collection of practices, procedures, and rules that must be applied to perform a specific operation to provide deliverables at the end of each stage. The eight principles listed below are built from a collection of procedures to establish an effective implementation methodology framework. This framework provides flexibility to react and adapt to the unique requirements of every project, incorporating the principles of:

- ✚ Project management and planning
- ✚ Scope and requirements specification
- ✚ Risk and issues management
- ✚ Communication and training
- ✚ Quality management
- ✚ Post-implementation review
- ✚ Documentation
- ✚ Experience

SAMPLE CODES

Shop.php

```
<?php
// Your database connection code
$conn = mysqli_connect("localhost", "root", "", "saplingo");

// Check the database connection
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // Assume you have retrieved the product ID, name, price, and quantity from the form
```

```

$productId = mysqli_real_escape_string($conn, $_POST['id']);
$productName = mysqli_real_escape_string($conn, $_POST['name']);
$productPrice = mysqli_real_escape_string($conn, $_POST['price']);
$purchaseQuantity = mysqli_real_escape_string($conn, $_POST['quantity']);

// Your code to process the purchase and update the quantity in the database
// Example SQL query:
$updateQuantityQuery = "UPDATE sales_items SET quantity = quantity - $purchaseQuantity
WHERE SI_id = '$productId'";

if (!mysqli_query($conn, $updateQuantityQuery)) {
    die("Error updating quantity: " . mysqli_error($conn));
}
// The rest of your checkout logic goes here
}

// Your HTML and form for checkout go here

$sql = "SELECT * FROM sales_items WHERE quantity > 0"; // Filter out items with zero quantity
$result = mysqli_query($conn, $sql);

// Check the query result
if (!$result) {
    die("Error: " . mysqli_error($conn));
}

if (mysqli_num_rows($result) > 0) {
    echo '<div class="row">'; // Open the row container
    while ($row = mysqli_fetch_assoc($result)) {
        echo '<div class="col-md-4">'; // Each product takes 4 columns
        echo '<div class="product" style="border-radius: 30px; border: 1px solid rgb(24, 179, 226);
text-align: center;">';
        echo '<h2><a href="product.php?id=' . $row['SI_id'] . '">'. $row['item_name'] . '</a></h2>';
    }
}

```

```
echo '<a href="product.php?id=' . $row['SI_id'] . '"></a>';
```

```
// Display the product price
```

```
if (isset($row['unit_price'])) {  
    echo '<p>Price: Rs. ' . number_format($row['unit_price'], 2) . '</p>';  
} else {  
    echo '<p>Price: Not available</p>';  
}
```

```
// Fetch the updated quantity from the database
```

```
$productId = $row['SI_id'];  
$quantityQuery = "SELECT quantity FROM sales_items WHERE SI_id = '$productId'";  
$quantityResult = mysqli_query($conn, $quantityQuery);
```

```
if ($quantityResult) {  
    $quantityRow = mysqli_fetch_assoc($quantityResult);  
    $updatedQuantity = isset($quantityRow['quantity']) ? $quantityRow['quantity'] : 'Not available';
```

```
    echo '<p>Quantity: ' . $updatedQuantity . '</p>';  
} else {  
    echo '<p>Quantity: Not available</p>';  
}
```

```
echo '</div>';
```

```
echo '</div>';
```

```
}
```

```
echo '</div>'; // Close the row container
```

```
} else {  
    echo '<div class="col-md-12 text-center">';  
    echo '<p>No products found.</p>';  
    echo '</div>';  
}
```

```
// Close the database connection
```

```
mysqli_close($conn);
```

```
?>
```

Product.php

```
<?php
```

```
// Database connection setup (same as your existing code)
```

```
$db_host = 'localhost';
```

```
$db_username = 'root';
```

```
$db_password = '';
```

```
$db_name = 'saplingo';
```

```
$conn = mysqli_connect($db_host, $db_username, $db_password, $db_name);
```

```
if (!$conn) {
```

```
    die("Connection failed: " . mysqli_connect_error());
```

```
}
```

```
// Get the product ID from the query parameter
```

```
if (isset($_GET['id'])) {
```

```
    $product_id = $_GET['id'];
```

```
// Retrieve product details based on the ID
```

```
$sql = "SELECT * FROM sales_items WHERE SI_id = $product_id"; // Update column names
```

```
$result = mysqli_query($conn, $sql);
```

```
if (!$result) {
```

```
    echo 'Error fetching product details: ' . mysqli_error($conn);
```

```
} else {
```

```
    if (mysqli_num_rows($result) > 0) {
```

```
        $row = mysqli_fetch_assoc($result);
```

```
        // You can remove the code that displays product details here
```

```
    } else {
```

```
        echo 'Product not found';
```

```
    }
```

```
}
```

```

} else {

    echo 'Invalid product ID';
}
?>

<?php
// Display the product image here
echo '<a href="' . $row['Simage'] . '" class="image-popup"></a>';
?>

</div>

<div class="col-lg-6 product-details pl-md-5 ftco-animate">

<?php
// Display product details here with updated column names
echo '<h3 class="mb-4">' . $row['item_name'] . '</h3>';
echo '<p class="price"><span id="displayPrice">Rs.' . number_format($row['unit_price'], 2) .
'</span></p>';
echo '<p class="mb-4">' . $row['description'] . '</p>';
?>

<div class="col-md-12">

<?php
    echo '<p style="color: #000;">' . $row['quantity'] . ' Units available</p>';
?>

<!-- Quantity input and buttons -->
<div class="quantity-section">
    <button onclick="decreaseQuantity()" class="quantity-btn">-</button>
    <input disabled type="number" id="quantityInput" name="quantity_enterd" value="1"
min="0" oninput="updatePrice()">
    <button onclick="increaseQuantity()" class="quantity-btn">+</button>
</div>
</div>

<?php

```

```

// Assuming $row is the current product data fetched from the database
$productId = $row['SI_id'];
$productName = $row['item_name'];
$productPrice = $row['unit_price'];
$productQuantity = $row['quantity'];

echo '<p class="mt-4"><a id="buyNowBtn" href="#" onclick="checkQuantity(' . $productId .
', \' . $productName . '\', ' . $productPrice . ', 1);" class="btn btn-black py-3 px-5">Buy
Now</a></p>';
?>

<script>
function checkQuantity(productId, productName, productPrice, quantity) {
    if (<?php echo $productQuantity; ?> > 0) {
        // Redirect to the next page
        window.location.href = 'checkout.php?id=' + productId + '&name=' + productName +
'&price=' + productPrice + '&quantity=' + quantity;
    } else {
        // Display an alert if the quantity is zero
        alert('Sorry, ' + productName + ' is out of stock.');
```



```
}
```

```
function decreaseQuantity() {
```

```
    var quantityInput = document.getElementById('quantityInput');
```

```
    var currentQuantity = parseInt(quantityInput.value, 10);
```

```
    // Optionally, you can update the displayed price based on the new quantity  
    updatePrice();
```

```
    if (currentQuantity > 1) {
```

```
        // Decrease the input value by 1
```

```
        quantityInput.value = currentQuantity - 1;
```

```
    }
```

```
}
```

```
function updatePrice() {
```

```
    var quantity = document.getElementById('quantityInput').value;
```

```
    var unitPrice = <?php echo json_encode($row['unit_price']); ?>;
```

```
    var totalPrice = quantity * unitPrice;
```

```
    document.getElementById('displayPrice').innerText = 'Rs.' + totalPrice.toFixed(2);
```

```
    var buyNowBtn = document.getElementById('buyNowBtn');
```

```
    buyNowBtn.href = 'checkout.php?id=<?php echo htmlspecialchars($row['SI_id']);  
>>&name=<?php echo urlencode($row['item_name']); >>&price=' + totalPrice + '&quantity=' +  
quantity;
```

```
    }
```

```
</script>
```

Checkout_process.php

```
<?php
```

```
session_start();
```

```
// Assuming you have a database connection, replace the placeholders with your actual database  
connection details
```

```
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "saplingo";
$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

// Retrieve form data
$productId = $_POST['productId'];
$firstname = $_POST['firstname'];
$lastname = $_POST['lastname'];
$location = $_POST['pickuppoint'];
$phone = $_POST['phone'];
$email = $_POST['email'];
$totalPrice = $_POST['totalPrice'];

$uid = $_SESSION['uid']; // Assuming you have a 'uid' key in your session.

$productId = $_POST['productId'];
$quantity = $_POST['quantity'];
$sql = "UPDATE sales_items SET quantity = quantity - $quantity WHERE SI_id = '$productId'";
mysqli_query($conn, $sql);

// Insert data into the 'order' table
$sql = "INSERT INTO `order` ( uid, item_id, firstname, lastname, location, phone, email,
totalprice, status)
VALUES ('$uid', '$productId', '$firstname', '$lastname', '$location', '$phone', '$email',
'$totalPrice','placed')";

if ($conn->query($sql) === TRUE) {
```

```
header("Location: index.php?order_posted=1");
exit(); // Ensure that no further code is executed after the redirect
} else {

echo "Error: " . $sql . "<br>" . $conn->error;
    echo '<br><a href="shop.php"><button>Back To Shop</button></a>';
}
$conn->close();
?>
```

U_ProcessLogin.php

```
<?php
// Start a session (this should be at the top of every PHP page where you use sessions).
session_start();
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "saplingo";

$conn = new mysqli($servername, $username, $password, $dbname);

// Check the connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

// Get the input from the form
$username = $_POST['uname'];
$password = $_POST['pswd'];

// Use prepared statements to prevent SQL injection
$stmt = $conn->prepare("SELECT uid, username FROM user WHERE username = ? AND
password = ?");
$stmt->bind_param("ss", $username, $password);
```

```

if ($stmt->execute()) {
    $result = $stmt->get_result();

    if ($result->num_rows == 1) {
        // Login successful
        $row = $result->fetch_assoc();

        $_SESSION['loggedin'] = true;
        $_SESSION['uid'] = $row['uid']; // Save the uid in the session
        $_SESSION['username'] = $row['username']; // Save the username in the session
        header("Location: index.php");
        exit();
    } else {
        // Login failed
        echo "<script>";
        echo "alert('Login failed. Please check your username and password.');"
        echo "window.location.href='ULogin.php';" // Redirect back to the login page
        echo "</script>";
    }
} else {
    // Database query error
    echo "Error: " . $stmt->error;
}

// Close the database connection
$stmt->close();
$conn->close();
?>

```

Signup process.php

```

<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // Database connection information
    $servername = "localhost"; // Change to your database server
    $username = "root"; // Change to your database username

```

```

$password = ""; // Change to your database password
$dbname = "saplingo"; // Change to your database name

// Create a connection to the database
$conn = new mysqli($servername, $username, $password, $dbname);

// Check the connection

if ($conn->connect_error) {

die("Connection failed: " . $conn->connect_error);
}

// Retrieve user data from the form
$newUsername = mysqli_real_escape_string($conn, $_POST['username']);
$newPassword = mysqli_real_escape_string($conn, $_POST['password']);
$newName = mysqli_real_escape_string($conn, $_POST['name']);
$newEmail = mysqli_real_escape_string($conn, $_POST['email']);

// Check for duplicate username or email
$check_duplicate_sql = "SELECT * FROM user WHERE username = '$newUsername' OR
email = '$newEmail'";
$duplicate_result = $conn->query($check_duplicate_sql);

if ($duplicate_result->num_rows > 0) {
    echo '<script>alert("Username or email already exists.");window.location.href =
"usersignup.php";</script>';
} else {
    // SQL query to insert a new user record using prepared statements
    $insert_sql = $conn->prepare("INSERT INTO user (username, password, status, email,
name)
VALUES (?, ?, 1, ?, ?)");

    // Bind parameters

```

```
$insert_sql->bind_param("ssss", $newUsername, $newPassword, $newEmail, $newName);
```

```
if ($insert_sql->execute()) {
```

```
    $account_created = true; // Variable to track successful account creation
```

```
} else {
```

```
    echo "Error adding user record: " . $insert_sql->error;
```

```
}
```

```
$insert_sql->close();
```

```
}
```

```
// Close the database connection
```

```
$conn->close();
```

```
}
```

```
// Print success message outside the form submission block
```

```
if (isset($account_created) && $account_created) {
```

```
    echo '<script>alert("Account created successfully.");
```

```
    window.location.href = "Ulogin.php"; // Redirect to the sign-in page
```

```
</script>';
```

```
    header("location: sign.html");
```

```
    exit();
```

```
}
```

```
?>
```

My_profile.php

```
div class="full" style="margin-left:560px;">
```

```
<div class="contents">
```

```
<div class="heading">
```

```
<h2>Welcome <?php echo $username; ?> </h2>
```

```
</div>
```

```
<div class="form">
```

```
<label>Username: <?php echo $username; ?></label>
```

```
<label>User Id: <?php echo $uid; ?></label>
```

```
</div>
```

```
</div>
```

```
</div>
```

```
<?php
```

```
// Replace $uid with the actual user ID you want to fetch orders for
```

```
$uid = $_SESSION['uid']; // Example user ID
```

```
// SQL query to fetch orders and item details for the given user ID
```

```
$sql = "SELECT o.o_id, o.totalprice, o.location, s.item_name, s.quantity
```

```
FROM `order` o
```

```
JOIN `sales_items` s ON o.item_id = s.SI_id
```

```
WHERE o.uid = ?";
```

```
// Use prepared statement to prevent SQL injection
```

```
$stmt = $conn->prepare($sql);
```

```
// Check if the statement was prepared successfully
```

```
if ($stmt === false) {
```

```
    die('Error: ' . $conn->error);
```

```
}
```

```
$stmt->bind_param("i", $uid); // Assuming $uid is an integer, adjust the type if needed
```

```
// Check if the binding was successful
```

```
if ($stmt === false) {
```

```
    die('Bind Param Error: ' . $stmt->error);
```

```
}
```

```
// Execute the statement
```

```

$stmt->execute();

// Get the result set
$result = $stmt->get_result();

// Close the prepared statement to free up resources
$stmt->close();

// Check if there are any orders
if ($result->num_rows > 0) {

?>
<?php
    // Output data of each row
    while ($row = $result->fetch_assoc()) {
        echo "<tr>
            <td>{$row['o_id']}</td>
            <td>{$row['totalprice']}</td>
            <td>{$row['location']}</td>
            <td>{$row['item_name']}</td>
            <td>{$row['quantity']}</td>
            <td><button                                class='cancel-btn'
onclick='cancelProduct({$row['o_id']})>Cancel</button></td>
            </tr>";
        }
    ?>
</table>

<script>
    function cancelProduct(orderId) {
        // You can implement the cancel logic using JavaScript/jQuery
        // For example, you might want to make an AJAX request to the server to cancel the
product.

        // Here, we'll simply display an alert for demonstration purposes.

```



```
alert('Product with Order ID ' + orderId + ' is canceled.');
```

```
    }
```

```
</script>
```

```
</body>
```

```
</html>
```

```
<?php
```

```
} else {
```

```
    echo "<br><center>No Orders Yet!</center>";
```

```
}
```

```
// Close the database connection
```

```
$conn->close();
```

```
?>
```

```
<br>
```

```
<?php
```

```
// Database connection setup (replace with your credentials)
```

```
$db_host = 'localhost';
```

```
$db_username = 'root';
```

```
$db_password = '';
```

```
$db_name = 'saplingo';
```

```
$conn = new mysqli($db_host, $db_username, $db_password, $db_name);
```

```
if ($conn->connect_error) {
```

```
    die("Connection failed: " . $conn->connect_error);
```

```
}
```

```
// Check if the delete button is clicked
```

```
if (isset($_POST['delete_product'])) {
```

```
    $delete_product_id = mysqli_real_escape_string($conn, $_POST['delete_product']);
```

```
// Perform delete action (you may want to add additional validation and error handling)
$delete_sql = "DELETE FROM sales_items WHERE SI_id = ?";
$delete_stmt = mysqli_prepare($conn, $delete_sql);

if ($delete_stmt) {
    mysqli_stmt_bind_param($delete_stmt, "i", $delete_product_id);
    mysqli_stmt_execute($delete_stmt);
} else {
    echo "Error preparing delete statement: " . mysqli_error($conn);
}

mysqli_stmt_close($delete_stmt);
}

// Fetch product data from the database for the logged-in user
$sql = "SELECT SI_id, Simage, item_name, quantity, description, unit_price FROM sales_items
WHERE uid = ?";
$stmt = mysqli_prepare($conn, $sql);

if ($stmt) {
    $uid = $_SESSION['uid'];
    mysqli_stmt_bind_param($stmt, "i", $uid);
    mysqli_stmt_execute($stmt);
    $result = mysqli_stmt_get_result($stmt);

    // Check if the user has posted any products
    if (mysqli_num_rows($result) > 0) {
        // Output table with CSS styles
        echo "<style>
            table {
                width: 95%;
                border-collapse: collapse;
                margin-bottom: 20px;
```

```
margin-left:20px;
    border-radius:10px;
    border:1px solid black;
}
```

```
th, td {
    border: 1px solid #dddddd;
    text-align: left;
    padding: 8px;
}
```

```
th {
    background-color: #f2f2f2;
}
p{
    margin-left:30px;
    font-size:15px;
}
button{
    border:1px solid gray;
    border-radius:5px;
    margin-right:20px;
}
</style>";
```

```
echo "<form method='post' action='".$_SERVER['PHP_SELF']."'>";
```

```
echo "<table>
```

```
<br><p><b>Your Products</b></p>
```

```
<tr>
```

```
<th>Sales Item No</th>
```

```
<th>Image</th>
```

```
<th>Item Name</th>
```

```

<th>Quantity</th>
    <th>Description</th>
    <th>Unit Price</th>
    <th>Action</th>
</tr>";

// Output data of each row
// delete and update button when click the button go to process_post.php
while ($row = mysqli_fetch_assoc($result)) {
    echo "<tr>
        <td>{$row['SI_id']}</td>
        <td><img src='{$row['Simage']}' alt='Product Image' style='width: 50px; height:
50px;'></td>

        <td>{$row['item_name']}</td>
        <td>{$row['quantity']}</td>
        <td>{$row['description']}</td>
        <td>{$row['unit_price']}</td>
        <td><button                type='submit'                name='delete_product'
value='{$row['SI_id']}'>Delete</button>
        <button    type='submit'    name='update_product'    value='{$row['SI_id']}'
formaction='postedit.php'>Update</button></td>
        </tr>";
    }

    echo "</table>";
    echo "</form>";
} else {
    echo "<br><center>You haven't posted any products yet!</center>";
}

mysqli_stmt_close($stmt);
} else {
    echo "Error preparing statement: " . mysqli_error($conn);
}

```

```
mysqli_close($conn);
```

```
?>
```

Contact.php

```
function numbercheck() {  
    // Get form field values  
    var nameValue = document.getElementsByName('name1')[0].value;  
    var emailValue = document.getElementsByName('email')[0].value;  
    var mobileNumber = document.getElementsByName('mobilenumber')[0].value;  
    var queriesValue = document.getElementsByName('queries')[0].value;  
  
    // Check if any of the fields are empty  
    if (nameValue === "" || emailValue === "" || mobileNumber === "" || queriesValue === "") {  
  
alert("Please fill in all fields.");  
        return false; // Prevent form submission  
    }  
  
    // Check if the mobile number has more than 10 digits or contains non-numeric characters  
    if (mobileNumber.length > 10 || !/^\\d+$/ .test(mobileNumber)) {  
        alert("Please enter a valid 10-digit mobile number.");  
        return false; // Prevent form submission  
    }  
  
    // If all conditions are met, the form will be submitted  
    return true;  
}
```

Process Order.php

```
<?php  
if ($_SERVER['REQUEST_METHOD'] == 'POST') {  
    // Get the form data  
    $firstname = $_POST['firstname'];  
    $lastname = $_POST['lastname'];  
    $pickupPoint = $_POST['pickuppoint'];
```

```

$phone = $_POST['phone'];
$email = $_POST['email'];
if (isset($_GET['totalPrice'])) {
    $totalPrice = floatval($_GET['totalPrice']);
    // Now, you can use $totalPrice in your process_order.php script.
    echo 'Total Price: $' . number_format($totalPrice, 2);
} else {
    echo 'Total Price not found in the URL.';
}
$quantity = isset($_POST['count']) ? intval($_POST['count']) : 1;

// Calculate total price
// $totalPrice = ($productPrice * $quantity) + 20.00; // Assuming $0 discount

// Replace these with your actual database connection details
$db_host = 'localhost';
$db_username = 'root';
$db_password = "";
$db_name = 'saplingo';

// Create a database connection
$conn = new mysqli($db_host, $db_username, $db_password, $db_name);

if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

$sql = "INSERT INTO `order` (firstname, lastname, quantity, location, phone, email,totalprice)
VALUES ('$firstname', '$lastname','$quantity','$pickupPoint','$phone',
'email','$totalPrice')";

if ($conn->query($sql) === TRUE) {

```

```

        echo "Order placed successfully!";
    } else {
        echo "Error: " . $sql . "<br>" . $conn->error;
    }

    $conn->close();
} else {
    echo "Invalid request.";
}
?>

```

Connection page

```

<?php

if (isset($_POST['submit'])) {
    // Connect to the database

    $servername = "localhost";
    $username = "root";
    $password = "";
    $dbname = "saplingo";

    $conn = new mysqli($servername, $username, $password, $dbname);

    if ($conn->connect_error) {
        die("Connection failed: " . $conn->connect_error);
    }
}
?>

```

Product post.php

```

<h1 class="heading">Post Your Product</h1><br>
<div class="form">
    <form action="process_post.php" method="post" enctype="multipart/form-data">
        <label for="title">Product Title:</label>
        <input type="text" name="item_name" required><br><br>
    </form>
</div>

```

```
<label for="description">Product Description:</label><br>
<textarea name="description" rows="4" cols="50" required></textarea><br><br>

<label for="price">Product Price:</label>
<input type="number" name="unit_price" step="0.01" required><br><br>

<label for="price">Quantity:</label>
<input type="number" name="quantity" min="1" required><br><br>

<label for="image">Upload Product Image:</label>
<input type="file" name="Simage" accept="image/*" required><br><br>

<input type="submit" name="submit" value="Post Product">
<a href="index.php"><input type="button" value="Go Home"></a>
</form>
```

```
<?php
session_start();

if (isset($_POST['submit'])) {
    // Database connection setup (replace with your credentials)
    $db_host = 'localhost';
    $db_username = 'root';
    $db_password = '';
    $db_name = 'saplingo';

    $conn = mysqli_connect($db_host, $db_username, $db_password, $db_name);

    if (!$conn) {
        die("Connection failed: " . mysqli_connect_error());
    }

    // Get user input
```



```

$item_name = mysqli_real_escape_string($conn, $_POST['item_name']); // Sanitize input
$description = mysqli_real_escape_string($conn, $_POST['description']);
$unit_price = floatval($_POST['unit_price']); // Ensure it's a float
$quantity = intval($_POST['quantity']); // Ensure it's an integer

$uid = $_SESSION['uid']; // Assuming you have a 'uid' key in your session.

// Check for file upload
$target_dir = "uploads/";
$target_file = $target_dir . basename($_FILES["Simage"]["name"]);

if (move_uploaded_file($_FILES["Simage"]["tmp_name"], $target_file)) {
    // Insert product data into the database using prepared statement
    $sql = "INSERT INTO sales_items (Simage, item_name, quantity, description, unit_price,
uid) VALUES (?, ?, ?, ?, ?, ?)";
    $stmt = mysqli_prepare($conn, $sql);

    if ($stmt) {
        mysqli_stmt_bind_param($stmt, "ssssdi", $target_file, $item_name, $quantity,
$description, $unit_price, $uid);

        if (mysqli_stmt_execute($stmt)) {
            // Add a header to redirect to shop.php after successful posting
            header("Location: shop.php?product_posted=1");
            exit(); // Ensure that no further code is executed after the redirect
        } else {
            echo "Error: " . mysqli_stmt_error($stmt);
        }

        mysqli_stmt_close($stmt);
    } else {
        echo "Error preparing statement: " . mysqli_error($conn);
    }
}

```

```
} else {  
    echo "Error uploading image."  
}
```

```
mysqli_close($conn);  
}  
?>
```

Process Order.php

```
<?php  
if ($_SERVER['REQUEST_METHOD'] == 'POST') {  
    // Get the form data  
    $firstname = $_POST['firstname'];  
    $lastname = $_POST['lastname'];  
    $pickupPoint = $_POST['pickuppont'];  
    $phone = $_POST['phone'];  
    $email = $_POST['email'];  
    if (isset($_GET['totalPrice'])) {  
        $totalPrice = floatval($_GET['totalPrice']);  
  
        // Now, you can use $totalPrice in your process_order.php script.  
        echo 'Total Price: $' . number_format($totalPrice, 2);  
    } else {  
        echo 'Total Price not found in the URL.';  
    }  
    $quantity = isset($_POST['count']) ? intval($_POST['count']) : 1;  
  
    // Calculate total price  
    //$totalPrice = ($productPrice * $quantity) + 20.00; // Assuming $0 discount  
  
    // Replace these with your actual database connection details  
    $db_host = 'localhost';  
    $db_username = 'root';  
    $db_password = '';  
    $db_name = 'saplingo';
```

```

// Create a database connection
$conn = new mysqli($db_host, $db_username, $db_password, $db_name);

if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

$sql = "INSERT INTO `order` (firstname, lastname, quantity, location, phone, email,totalprice)
VALUES      ('$firstname',      '$lastname','$quantity','$pickupPoint','$phone',
'email','$totalPrice')";

if ($conn->query($sql) === TRUE) {
    echo "Order placed successfully!";
} else {
    echo "Error: " . $sql . "<br>" . $conn->error;
}

$conn->close();
} else {
    echo "Invalid request.";
}
?>

```

Admin.php

```

<?php
// Database connection information
$servername = "localhost"; // Change to your database server
$username = "root"; // Change to your database username
$password = ""; // Change to your database password
$dbname = "saplingo"; // Change to your database name

// Create a connection to the database
$conn = new mysqli($servername, $username, $password, $dbname);

```

```

// Check the connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

// SQL query to get the total number of users
$sql = "SELECT COUNT(*) as total_users FROM user";

// Execute the query
$result = $conn->query($sql);

// Check for query execution errors
if (!$result) {
    die("Query failed: " . $conn->error);
}

if ($result->num_rows > 0) {
    // Fetch the total number of users
    $row = $result->fetch_assoc();
    $totalUsers = $row['total_users'];
} else {
    $totalUsers = 0;
}

// Close the database connection
$conn->close();
?>

<p><?php echo $totalUsers;?><br><span>Users</span></p>

<i class="fa fa-users box-icon"></i>

</div>

</div>

<div class="col-div-3">

<?php

```

```

// Database connection information
$servername = "localhost"; // Change to your database server
$username = "root"; // Change to your database username
$password = ""; // Change to your database password
$dbname = "saplingo"; // Change to your database name

// Create a connection to the database
$conn = new mysqli($servername, $username, $password, $dbname);

// Check the connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

// SQL query to count rows in the "sales_items" table
$count_sql = "SELECT COUNT(*) AS item_count FROM sales_items";

$result = $conn->query($count_sql);

if ($result->num_rows > 0) {
    $row = $result->fetch_assoc();
    $itemCount = $row['item_count'];
} else {
    $itemCount = 0;
}

// Close the database connection
$conn->close();
?>

<div class="box">
    <p><?php echo $itemCount; ?><br/><span>Products</span></p>
    <i class="fa fa-shopping-bag box-icon"></i>
</div>

```

```

        </div>
        <div class="col-div-3">
        <?php
// Database connection information
$servername = "localhost"; // Change to your database server
$username = "root"; // Change to your database username
$password = ""; // Change to your database password
$dbname = "saplingo"; // Change to your database name

// Create a connection to the database
$conn = new mysqli($servername, $username, $password, $dbname);

// Check the connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

// SQL query to count rows in the "order" table
$count_sql = "SELECT COUNT(*) AS order_count FROM `order`"; // Use backticks around table
name `order`

$result = $conn->query($count_sql);

if ($result->num_rows > 0) {
    $row = $result->fetch_assoc();
    $orderCount = $row['order_count'];
} else {
    $orderCount = 0;
}

// Close the database connection
$conn->close();
?>

```

```
<div class="box">
    <p><?php echo $orderCount; ?><br><span>Orders</span></p>
    <i class="fa fa-list box-icon"></i>
</div>
```

```
</div>
```

```
<div class="clearfix"></div>
```

```
<br/><br/>
```

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

```
    <div class="box-8">
```

```
        <div class="content-box">
```

```
            <p>All Users </p>
```

```
            <br/>
```

```
            <?php
```

```
// Database connection information
```

```
$servername = "localhost"; // Change to your database server
```

```
$hostname = "root"; // Change to your database username
```

```
$password = ""; // Change to your database password
```

```
$dbname = "saplingo"; // Change to your database name
```

```
// Create a connection to the database
```

```
$conn = new mysqli($servername, $hostname, $password, $dbname);
```

```
// Check the connection
```

```
if ($conn->connect_error) {
```

```
    die("Connection failed: " . $conn->connect_error);
```

```
}
```

```
// SQL query to retrieve data from the "user" table
```

```
$sql = "SELECT uid, username, status FROM user";
```

```
// Execute the query
```

```
$result = $conn->query($sql);
```

```

if ($result->num_rows > 0) {
    // Output HTML table headers
    echo '<table>';
    echo '<tr><th>User ID</th><th>Username</th><th>User Status</th></tr>';

    // Output data from each row
    while ($row = $result->fetch_assoc()) {
        echo '<tr>';
        echo '<td>' . $row['uid'] . '</td>';
        echo '<td>' . $row['username'] . '</td>';
        echo '<td>' . $row['status'] . '</td>';
        echo '</tr>';
    }

    // Close the table
    echo '</table>';
} else {
    echo "No data found in the 'user' table.";
}

// Close the database connection
$conn->close();
?>

```

Adminuseradd.php

```

<?php if(isset($_GET['error'])){?>
    <p class="error" style="color:white;border-radius:5px;background:none;margin-left:50px;"><?php echo $_GET['error'];?></p>
    <?php }?>
    <label for="username">Username:</label>
    <input type="text" name="username" required placeholder="Enter username"><br>

    <label for="name">Name:</label>
    <input type="text" name="name" required placeholder="Enter name"><br>

```



```
<label for="password">Password:</label>
    <input type="password" name="password" required placeholder="New password"><br>

<label for="email">Email:</label>
<input type="email" name="email" required placeholder="Enter Email"><br>

<input id="b1" type="submit" value="Submit">
<br><a id="link2" href="admin.php">Back to Admin</a>

</form>
<br> <form method="get" action="admin.php">
```

CODE VALIDATION & OPTIMIZATION

Validation is the process of evaluating software at the end of software development to ensure compliance with software requirements. Testing is a common method for validation. Validation succeeds when the software functions in a manner that can be reasonably expected by the customer. Form data validation comes in a couple of different forms. Data can be validated at the field level when it is entered by the user, and it can be validated at the form level (i.e. all fields) when the form is submitted or printed. These different types of validation have different, complementary purposes and for a complete form design, it's a good practice to use a combination of the two methods.

FIELD LEVEL VALIDATION

The purpose of Field Level Validation is to verify that the input to a single field is entered correctly. For example, for an e-mail field, the job of the validation script is to make sure the entered text matches the standard email format, i.e., two sets of strings separated by an "@" symbol. The most common way to implement a text pattern test like this is to use a regular expression. Most of the time validation scripts are used to match input text against a pattern using a regular expression.

FORM LEVEL VALIDATION

Form level validation is used to ensure all the required form data is filled in, and or to make sure that any data dependencies between fields are met before the form is submitted.

Different validators are:

Required Field Validator

It is the simplest type of validation. If the user enters a value in any field, then it is valid.

Compare Validator

Compare validator compares the user's entity with a constant value.

Range Validator

Range validator checks that the user entry between specified lower and upper boundaries.

UNIT TESTING

Unit testing is a software development process in which the smallest testable parts of an application, called units, are individually and independently scrutinized for proper operation. Unit testing can be done manually but is often automated. In computer programming, unit testing is a software testing method by which individual units of source code, sets of one or more computer programs.

Modules together with associated control data, usage procedures, and operating procedures, are tested to determine whether they are fit for use.

The goal of unit testing is to isolate each part of the program and show that the individual parts are correct. A unit test provides a strict, written contract that the piece of code must satisfy.

TEST PLAN & TEST CASES

A test plan is a document detailing the objectives, target market, internal beta team, and processes for a specific beta test for a software or hardware product. The plan typically contains a detailed understanding of the eventual workflow.

SYSTEM TESTING

CHAPTER 6

SYSTEM TESTING

INTRODUCTION

Software testing is a critical element of software quality assurance and represents the ultimate review of specification, design, and coding, testing presents an interesting anomaly for the software engineer.

Testing Objectives include:

- Testing is a process of executing a program with the intent of finding an error.
- A good test case has a probability of finding an as-yet-undiscovered error.
- A successful test uncovers an undiscovered error.

Testing Principles:

- All tests should be traceable to end-user requirements.
- Tests should be planned long before testing begins.
- Testing should begin on a small scale and progress towards testing on a large.
- Exhaustive testing is not possible.

The testing phase of the system development life cycle will test the system design. Testing of the system decides whether the newly designed system works properly or not. After the development of documentation manually about the system this stage is checked. And if the system working properly then it will be considered for implementation and if isn't then the system analyst is informed to find out generated errors or problems and to find out its solutions. This process is known as debugging.

Black Box Testing

Black Box testing also called functional testing, focuses on the functional requirements of the software. Knowing the specified function that a product designed to perform the test can be conducted to ensure that each function is fully operational. Black Box tests are carried out to test that input to function is properly accepted and output is correctly produced. Black Box Testing treats the software as a black box without any knowledge of internal implementation

For the proposed system the black box testing ensures the following requirements.

Menus are displayed correctly for each user. The login checking is ensured. Black Box testing is used to detect errors of the type of incorrect or missing functions, interface errors, errors in data structures or external database access, initialization, and termination errors. Black Box testing methods include equivalence partitioning, Boundary value analysis, and specification testing.

White Box Testing

This type of testing is also called structural testing, involves checking the actual code following the control flow and ensuring the validity of the logic involved in the implementation. White Box Testing is when the tester has access to the internal data structures and algorithms including the code that implements these.

Testing was done on:

- Programming portion: For optimizing the speed of execution, the unwanted portion of the code is removed.
- The structure of every web page of the system test cases.
- Execute each independent part of the code at least once.
- Check the validity of control statements and their boundary values.
- Check the validity of the data structures used to check the validity and the effectiveness of the table design.

INTEGRATION TESTING

Through each program work individually, they should work after linking together. This is referred to as interfacing. Data may be lost across the interface one module can have an adverse effect on the other subroutines after linking may not do the desired function expected by the main routine. Integration testing is the systematic technique for constructing the program structure while at the same time conducting tests to uncover errors associated with the interface. Using an integrated test plan prepared in the design phase of the system development as a guide, the integration test was carried out. All the errors found in the system were corrected for the next testing step.

SYSTEM TESTING

System testing is a level of software testing where a complete and integrated software is tested. The purpose of this test is to evaluate the system's compliance with the specified requirements. System testing of software or hardware is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements.

System testing falls within the scope of black-box testing, and as such, should require no knowledge of the inner design of the code or logic.

TEST PLANNING AND CASES

A test plan is a document detailing the objectives, target market, internal beta team, and processes for a specific beta test for a software or hardware product. The plan typically contains a detailed understanding of the eventual workflow.

Test Case 1

LOGIN PAGE

Name of Form: Login

Objective: This page can be used for the login into the system.

Test Case 2

Module Name: Admin Name of Form: Login

Objective: This page can be used for login of the administrator

SYSTEM MAINTENANCE

CHAPTER 7

SYSTEM MAINTENANCE

INTRODUCTION

Software maintenance in software engineering is the modification of a software product after delivery to correct faults, improve performance, or other attributes. A common perception of maintenance is that it merely involves fixing defects. An integral part of the software is maintenance one, which requires an accurate maintenance plan to be prepared during the software development. It should specify how users will request modifications or report problems. The budget should include resources and cost estimates. A new decision should be made for the development of every new system feature and its quality objectives. The software maintenance, which can last for 5–6 years (or even decades) after the development process, calls for an effective plan that can address the scope of software maintenance, the tailoring of the post-delivery/deployment process, the designation of who will provide maintenance, and an estimate of the life-cycle costs. The selection of proper enforcement of standards is a challenging task right from an early stage of software engineering which has not got definite importance by the concerned stakeholders.

SYSTEM MAINTENANCE

Software maintenance denotes any changes made to the software product after it has been delivered to the customer. Maintenance is inevitable for almost any kind of product. However, most products need maintenance due to the wear and tear caused by use. On the other hand software products do not need maintenance on this account, but need maintenance to correct errors, enhance features, port to new platform, etc.

The requirements of Software maintenance arise on account of three main reasons:

1. **Corrective:** Corrective Maintenance of software products becomes necessary to rectify the bugs observed while the system is in use.
2. **Adaptive:** A software product might need maintenance when the customers need the product to run on new platforms, or new operating systems, or when they need the product to be interfaced with new hardware or software.
3. **Perfective:** A software product needs maintenance to support the new features that users want to support, to change different functionalities of the system according to customer demands, or to enhance the performance of the system.

The system developed should be secured and protected against possible hazards. Security measures are provided to prevent unauthorized access to the database at various levels. Password protection and simple procedures to prevent unauthorized access are provided to the users. The system allows the user to enter the system only through a proper username and password.

FUTURE ENHANCEMENT AND
SCOPE OF FURTHER
DEVELOPMENT

CHAPTER 8

FUTURE ENHANCEMENT AND SCOPE OF FURTHER DEVELOPMENT

INTRODUCTION

The purpose of the project entitled "Saplingo: The Sapling Ordering Website" is to develop user-friendly, efficient, and fast software that enables users in India to easily order food with corresponding categories. The system comprises two key modules: the User module and the Admin module.

In the User module, individuals can create their accounts by providing essential details such as a username and password. Upon logging in, users can explore a variety of food items categorized for easy navigation. Ordering is simplified by inputting details such as name, address, email, and contact information. All user-related data is securely stored in the database, ensuring privacy and enabling swift data processing.

On the other hand, the Admin module allows administrators to log in, view all user orders, and efficiently manage them. The system employs PHP and MySQL for development, ensuring a robust and scalable solution.

Saplingo is an online platform dedicated to simplifying the process of purchasing saplings. The meticulous protection of user data adds an extra layer of security, enhancing the overall reliability of the system.

MERITS OF THE SYSTEM

The merits of the system are,

- Convenience for Customers
- Increased Sales
- Cost Savings
- Health and Safety
- Easy to access and use the system

LIMITATIONS OF THE SYSTEM

The limitations of the system are,

- System is not accessible without internet
- High Competition
- User Interface Challenges

FUTURE ENHANCEMENT OF THE SYSTEM

The current system is developed and designed in such a way that any further enhancement can be done with ease and new modules can be added to the existing system with less effort. Anyway, in the future we can provide the facility to chat within the application itself.

CONCLUSION

CHAPTER 9

CONCLUSION

The user creates an account by providing essential details like username, full name, and password. Upon logging in with the username and password, they can explore and select saplings based on corresponding categories, and proceed to place an order by providing details such as name, address, phone number, and email. Users have the option to cancel the order before the admin updates it to "delivered."

The Admin module allows administrators to view all sapling orders from users and efficiently manage them. The admin can update the status of orders and view them by user. If a particular sapling is unavailable, the admin can mark it as "NO," and users won't be able to see that item.

This way, Saplingo serves as an online platform for the sale of saplings, offering users a seamless experience in exploring, ordering, and managing their sapling purchases.

BIBLIOGRAPHY

CHAPTER 10

BIBLIOGRAPHY

BOOK REFERENCES

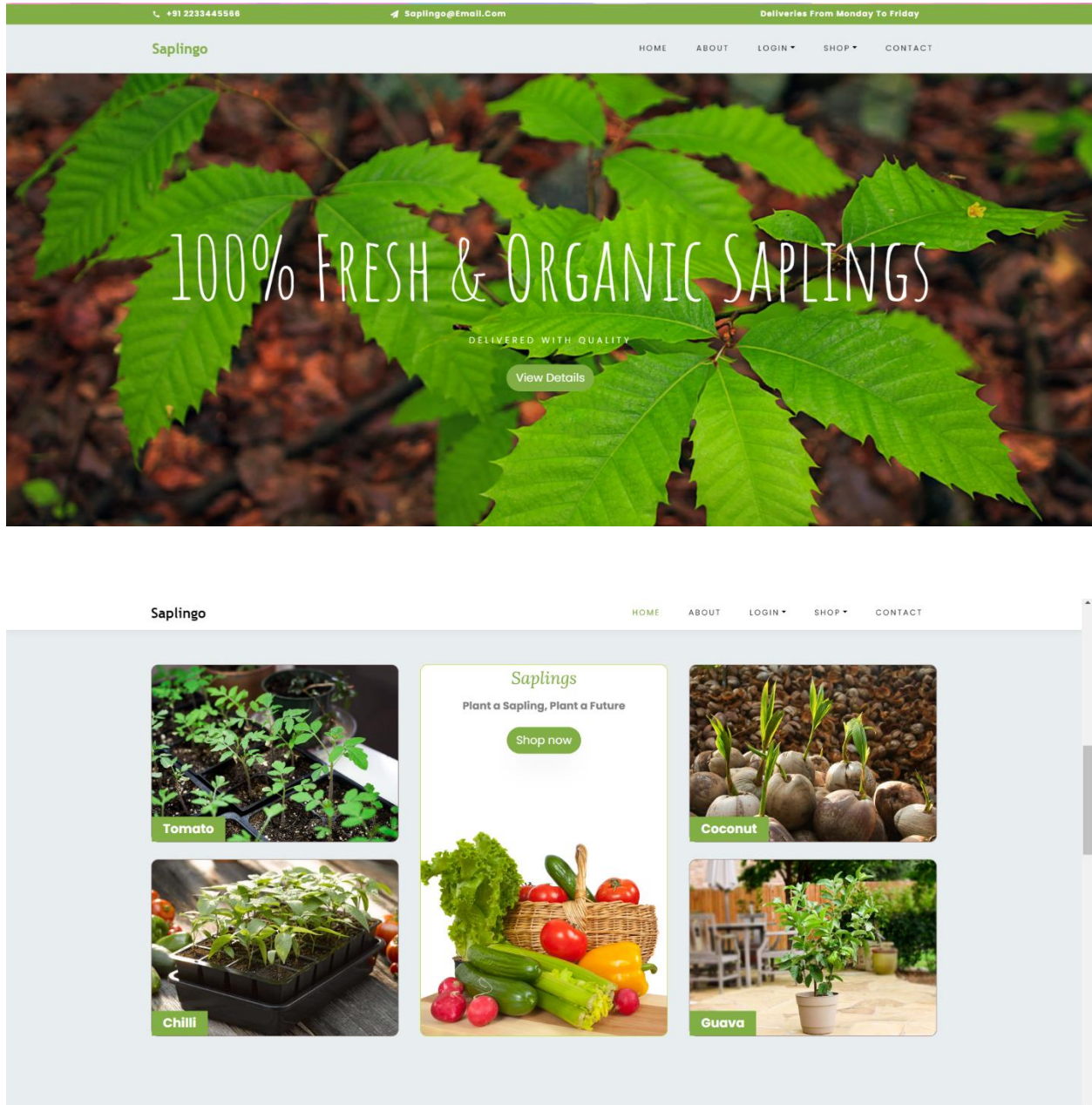
- Web programming using PHP, Binu M.B & Amaladevi V.C
- T. A. Powell, *HTML: the complete reference*. Berkeley, CA: McGraw-Hill, 2000.
- M. E. Davis and J. A. Phillips, *Learning PHP and MySQL: Step-by-step Guide to Creating Database-driven Web Sites*. Shroff Publishers and Distributers, 2006.
- R. Ramakrishnan, *Database management system*, 3rd ed. New York, NY: McGraw-Hill, 2011.
- H. E. Williams and D. Lane, *Web database applications with PHP & MySQL*. New Delhi: Shroff Publishers and Distributors, 2010. D. Maidasani, *Straight to the point: PHP*. Banga-lore: Firewall Media, 2007.
- J. Duckett, *Beginning HTML, XHTML, CSS, and JavaScript*. New Delhi: Wiley, 2014.
- R. Bangia, *Learning Java Script*, 4th ed. New Delhi: Khanna publications, 2009.

WEBSITE REFERENCE

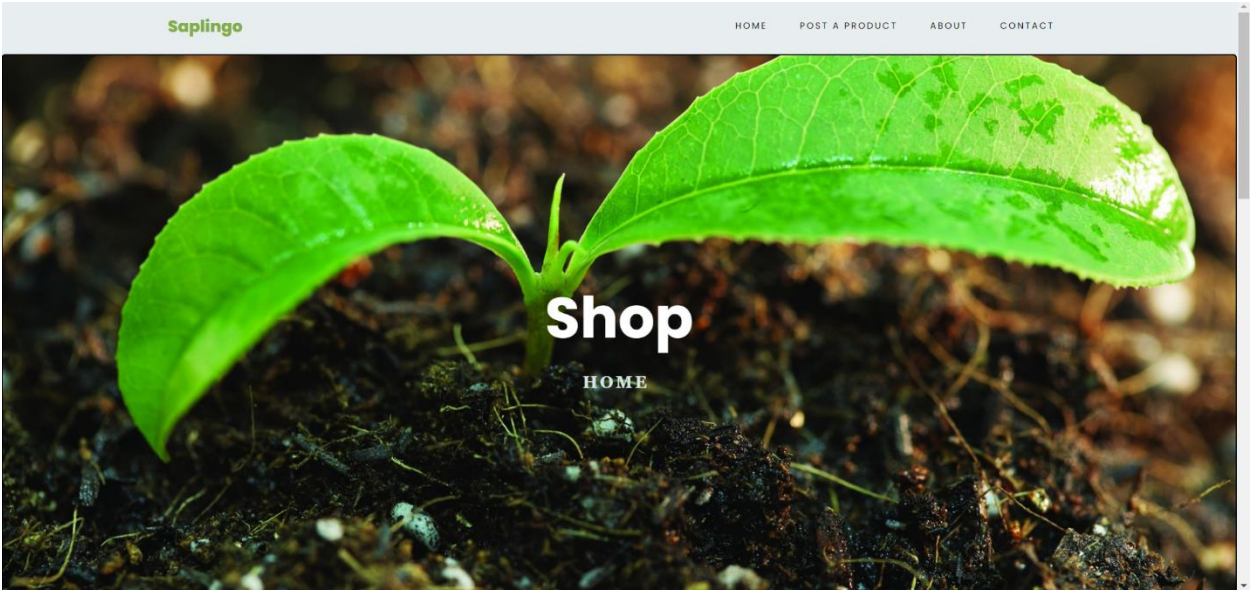
- Mysql.com. (2019). *MySQL*. [online] Available at: <https://www.mysql.com/>.
- WampServer.(2019). *WampServer*. [online] Available at : <http://www.wampserver.com/en/>.
- Php.net. (2019). *PHP: Hypertext Preprocessor*. [online] Available at: <https://www.php.net/>.
- Javascript.com. (2019). *Free JavaScript training, resources and examples for the community*. [online] Available at: <https://www.javascript.com/>.
- MDN Web Docs. (2019). *CSS: Cascading Style Sheets*. [online] Available at: <https://devel-oper.mozilla.org/en-US/docs/Web/CSS>.

APPENDIX

HOME PAGE



SHOP




Saplingo

HOME POST A PRODUCT ABOUT CONTACT

All Products


marigold



Price: Rs. 230.00

Quantity: 10


rose



Price: Rs. 100.00

Quantity: 4

Ceverin Candrian




Price: Rs. 1,499.00

Quantity: 20

PRODUCT CHECKOUT

Saplingo

HOMEABOUTCONTACT



Ceverin Candrian

Rs.1,499.00

So beautiful, so elegant

20 Units available

-

1

+

Buy Now

Billing Details

First Name

Last Name

Shipping Address:

Phone

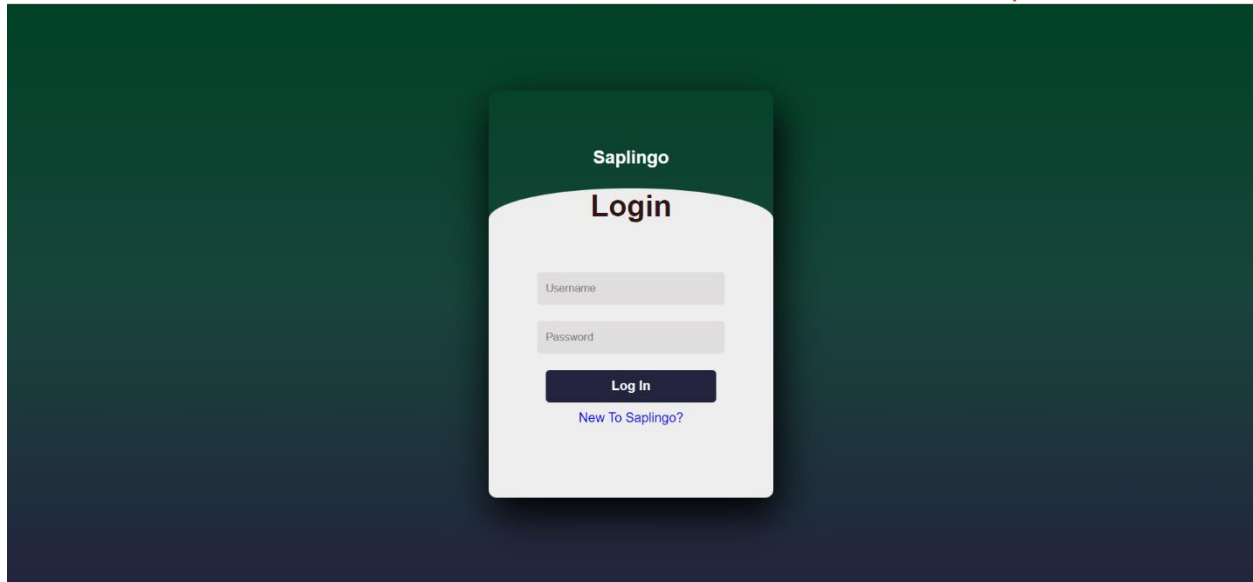
Enter your phone number

Email Address

Enter your email address

Product Name: Ceverin Candrian

USER LOGIN



The login form is centered on a dark green background. It features a white card with the 'Saplingo' logo and the word 'Login' in bold. Below the title are two input fields for 'Username' and 'Password'. A dark blue 'Log In' button is positioned below the password field. A link for 'New To Saplingo?' is located at the bottom of the card.

Saplingo
Login

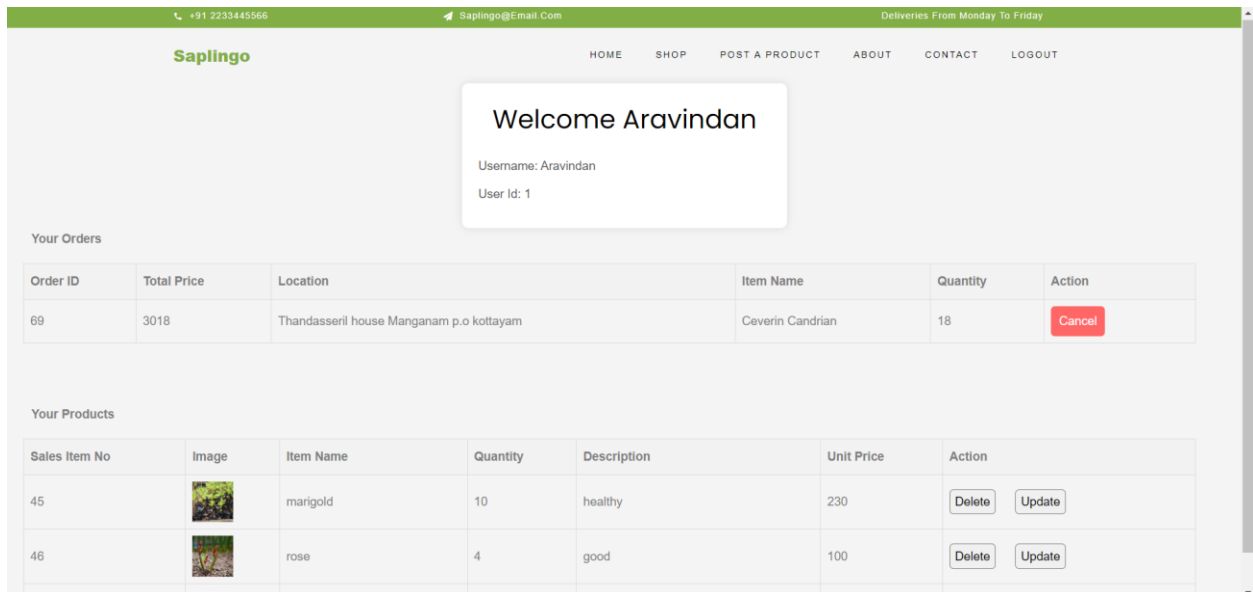
Username

Password

Log In

[New To Saplingo?](#)

USER PROFILE



The user profile dashboard has a green header bar with contact information and a navigation menu. A central white box displays a welcome message and user details. Below this, there are two sections: 'Your Orders' with a table of order data and 'Your Products' with a table of product data.

+91 2233445566 Saplingo@Email.Com Deliveries From Monday To Friday

Saplingo HOME SHOP POST A PRODUCT ABOUT CONTACT LOGOUT



Welcome Aravindan

Username: Aravindan
User Id: 1

Your Orders

Order ID	Total Price	Location	Item Name	Quantity	Action
69	3018	Thandasseril house Manganam p.o kottayam	Ceverin Candrian	18	<button>Cancel</button>

Your Products

Sales Item No	Image	Item Name	Quantity	Description	Unit Price	Action
45		marigold	10	healthy	230	<button>Delete</button> <button>Update</button>
46		rose	4	good	100	<button>Delete</button> <button>Update</button>

PRODUCT POST PAGE FOR USER

SaplingoHOMEABOUTCONTACTPROFILE ▾

Post Your Product

Product Title:

Product Description:

Product Price:

Quantitv:

CONTACT

SAPLINGOHOMEABOUTSHOP ▾CONTACT

Address:

Saplingo

Changanacherry Near Mc

Road

Website

<https://www.Saplingo.com>

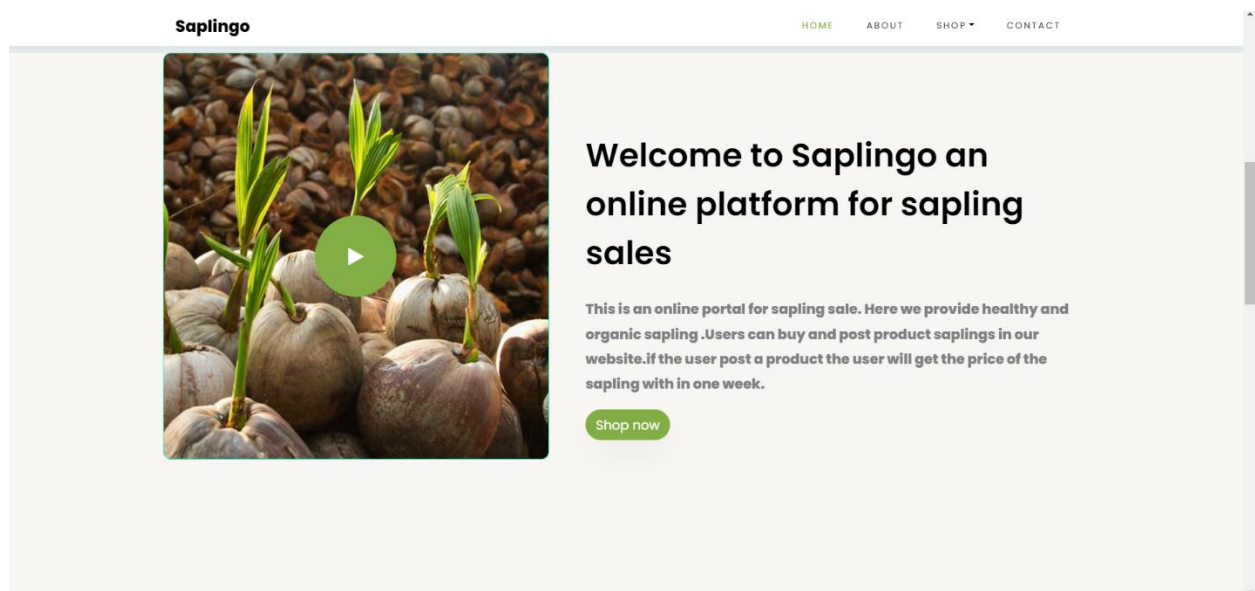
Email:

saplingoservice@gmail.com

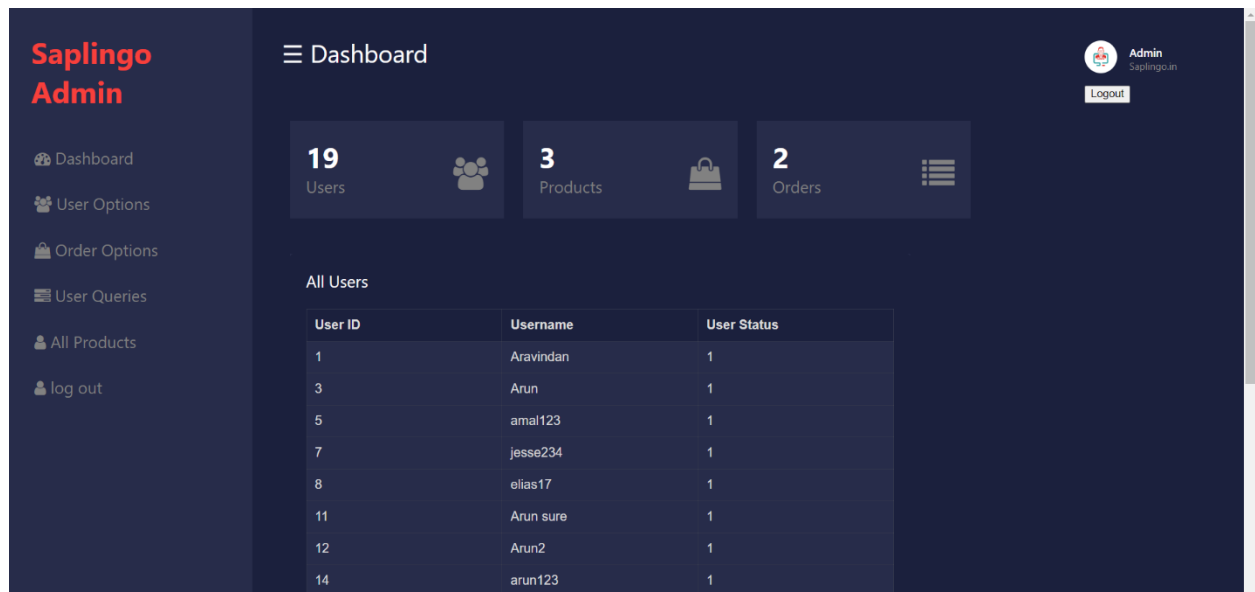
Phone: +91 2233445566

You Can Ask Your Queries

ABOUT US PAGE



ADMIN DASHBOARD



OTHER ADMIN PAGES

Saplingo
Admin

Dashboard

User Options

Order Options

User Queries

All Products



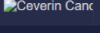
log out

Product Details

Admin
Saplingo.in

Logout

All products

Sales Item...	Item Name	Image	Unit Price	Quantity	Action
45	marigold		230.00	10	Delete
46	rose		100.00	4	Delete
49	Ceverin Ca...		1,499.00	18	Delete

Saplingo
Admin

Dashboard

User Options

Order Options

User Queries

All Products

log out

Order Options

Admin
Saplingo.in

Logout


Orders

Order ID	Item ID	Location	First Name	Last Name	Action
67	46	Thandasse...	Aravind	B	Order Placed
69	49	Thandasse...	George	Washington	Order Placed

Saplingo Admin

- Dashboard
- User Options
- Order Options
- User Queries
- All Products
- log out

User Options

 Admin
Saplingo.in
[Logout](#)

All Users

User ID	Username	User Status	Action
1	Aravindan	1	Delete
3	Arun	1	Delete
5	amal123	1	Delete
7	jesse234	1	Delete
8	elias17	1	Delete
11	Arun sure	1	Delete
12	Arun2	1	Delete
14	arun123	1	Delete
15	abin123	1	Delete
16	arun@123	1	Delete
17	arun@1234	1	Delete
18	arun@1234	1	Delete