|  |
| --- |
| **TIFFINLY**  PROJECT REPORT SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE AWARD OF  **BACHELOR OF COMPUTER APPLICATIONS**  To  **MARIAN COLLEGE KUTTIKKANAM [AUTONOMOUS]**  Affiliated to  **MAHATMA GANDHI UNIVERSITY, KOTTAYAM**  By  **MEKHNA ALPHONS JOBY**  (Reg.No:23UBC240)  Guided By  **Ms. ROSAMMA** **K S**    **DEPARTMENT OF COMPUTER APPLICATIONS**  **MARIAN COLLEGE KUTTIKKANAM (AUTONOMOUS)**  **PEERMADE – 685531**  **OCTOBER, 2025**  1 |

# DECLARATION

I, **MEKHNA ALPHONS JOBY [Reg.no 23UBC240],** certify that the Mini project report entitled **“TIFFINLY”** is an authentic work carried out by me at Marian College Kuttikkanam (Autonomous). The matter embodied in this project work has not been submitted elsewhere for the award of any degree or diploma to the best of our knowledge and belief.

Signature of the Student:

**MEKHNA ALPHONS JOBY**

Date:

# BONAFIDE CERTIFICATE

This is to certify that this project work entitled **“TIFFINLY”** is a bonafide record of work done by **Ms. MEKHNA ALPHONS JOBY [Reg.no 23UBC240]** at Marian College Kuttikkanam (Autonomous) in partial fulfilment for the award of the **Degree of Bachelor of Computer Applications** of **Mahatma Gandhi University, Kottayam.**

This work has not been submitted elsewhere for the award of any degree to the best of our knowledge.

Head of the Department Internal Guide

**Dr Rajimol A Ms. Rosamma K S**

Dept. of Computer Applications Dept. of Computer Applications

Marian College Kuttikkanam Marian College Kuttikkanam

Peermade – 685531 Peermade - 685531

Submitted for the Viva-Voice Examination held on

**DEPARTMENT SEAL**

**EXTERNAL EXAMINER**

# ACKNOWLEDGEMENT

“Gratitude is a feeling which is more eloquent than words, more silent than silence.” In undertaking this project, I needed the direction, assistance and cooperation of various individuals and organizations, which is received in abundance with grace of God, without their unconstrained support, the project could not have been completed. If words are considered as the symbol of approval and token of acknowledgement, then let the following words play the heralding role of expressing my gratitude.

I wish to acknowledge our sincere gratitude to our Manager, **Rev.Fr. Boby Alex** **Mannamplackal** and **Dr. Ajimon George**, Principal **Marian College Kuttikkanam (Autonomous),** for all their efforts and administration in educating me in their premier institution.

I extend my gratitude to **Dr. Rajimol A**, Head of the Department of Computer Application, who is a constant source of inspiration and whose advice helped me to complete this project successfully. I express my deep sense of gratitude to our internal project guide,

**Ms. Rosamma K S**, for her profound guidance for the successful completion of this project. With great enthusiasm I express my gratitude to all the faculty members of Department of Computer Applications for their timely help and support.

Finally, I express my deep appreciation to all my friends and family members for the moral support and encouragement they have given to complete this project successfully.

Mekhna Alphons Joby

# ABSTRACT

## ABSTRACT

The project named **“Tiffinly”** is a web-based meal subscription and delivery management platform that connects customers with affordable food through flexible subscription plans. Designed for students and working professionals, the system allows customers to browse meal options, customize preferences, manage subscriptions, schedule deliveries, and make secure payments.

The platform is role-based:

* **Customers** can browse plans, subscribe, customize meals, track orders, and submit feedback.
* **Admins** manage meals, plans, subscriptions, partners, payments, and respond to inquiries.
* **Delivery Partners** accept and update deliveries, view history, and log issues.

Built with **HTML, CSS, JavaScript, PHP, and MySQL**, Tiffinly supports full CRUD operations and real-time updates via AJAX. Core features include subscription management, cart and checkout, feedback, delivery tracking, and payout handling. It provides an end-to-end digital solution for streamlining the subscription-based meal service system.

|  |
| --- |
| **TABLE OF CONTENTS**      **TITLE……………………………………………………………………. .PAGE NUMBER**  **1.INTRODUCTION ..............................................................................................................10**  **1.1 ABOUT THE PROJECT 10**  **1.2 EXISTING SYSTEM 10**  **1.3 PROPOSED SYSTEM 11**   1. **SYSTEM ANALYSIS .......................................................................................................13**    1. **PROBLEM DEFINITION 13**    2. **FEASIBILITY ANALYSIS 13**    3. **RECOMMENDED IMPLEMENTATIONS 14** 2. **SOFTWARE REQUIREMENT SPECIFICATION ......................................................17**    1. **INTRODUCTION 17**    2. **PURPOSE 17**    3. **SCOPE 18**    4. **TECHNICAL OVERVIEW 18**    5. **FUNCTIONAL REQUIREMENTS 18**    6. **NON-FUNCTIONAL REQUIREMENTS 20**    7. **STATED REQUIREMENTS 21**    8. **EXTERNAL INTERFACE REQUIREMENTS 23** 3. **SYSTEM DESIGN ............................................................................................................27**    1. **INTRODUCTION 27**    2. **DESIGN METHODOLOGY 28**    3. **SYSTEM ARCHITECTURE AND PROCESS FLOW 32**    4. **MODULE DETAILS 45**    5. **PERFORMANCE CONSIDERATIONS 46**    6. **SECURITY CONSIDERATIONS 46**    7. **TABLE DESIGN 46**   **5. CODING .............................................................................................................................52**   * 1. **INTRODUCTION 52**   2. **SELECTION OF SOFTWARE 52**   3. **CODING PHASE 53** |
| **6. TESTING .................................................................................................................... 55**   * 1. **INTRODUCTION 55**   2. **TESTING 55**   3. **TESTING METHODS 56**   4. **IMPLEMENTATION 59**   **7. MAINTENANCE AND ENHANCEMENT .................................................................... 62**  **7.1 MAINTENANCE 62**  **7.2 ENHANCEMENT 62**  **8. CONCLUSION .......................................................................................................... .65** |

# INTRODUCTION

## 1.INTRODUCTION

### 1.1 ABOUT THE PROJECT

**Tiffinly – A Meal Subscription Platform** digitizes the process of ordering and delivering healthy meals through subscription models. It is built for students, professionals, and individuals living away from home.

The system offers **Basic and Premium meal plans** with weekly schedules (Weekdays, Extended Week, Full Week). Customers can customize dietary preference (Basic Plan feature) and meals (Premium Plan feature), add plans to cart, set delivery preferences, and pay securely.

Admins oversee the platform’s operations including meal listings, subscriptions, delivery partner management, and payments. Delivery partners can log in to accept orders, update statuses, and track their earnings.

#### 1.1.1 THE PURPOSE AND SCOPE

The main purpose of the Tiffinly project is to develop a unified and user-friendly web application that digitizes the process of ordering and delivering meals through subscription plans. It bridges the gap between cloud kitchens and customers by streamlining every step—meal browsing, customization, carting, payment, delivery, and feedback.

This platform is intended to cover the complete subscription cycle – meal browsing, customization, carting, delivery scheduling, payment, tracking, feedback, and administration.

The scope of Tiffinly encompasses a complete food subscription cycle. Key functionalities include:

* **For Customers:**
  + Browse and compare meal plans
  + Select a plan and based on it save preferences and activate a subscription
  + Schedule deliveries and make payments
  + Track order history
  + Submit feedback and raise inquiries
* **For Admins:**
  + Add/edit/delete meals and plans
  + Get an overview of customer accounts and subscriptions
  + Assign deliveries if required and monitor delivery process
  + Analyse feedbacks and respond to inquiries
* **For Delivery Partners:**
  + View unassigned orders
  + Accept and complete deliveries
  + Update delivery status and log issues
  + Analyse performance and earnings.

The platform is designed for scalability and supports role-based access, secure transactions, and efficient data management using PHP, MySQL, HTML, CSS, and JavaScript.

### 1.2 EXISTING SYSTEM

In the current scenario, individuals rely on local tiffin providers or food delivery services, which often lack customization, transparency, and consistency. Manual processes lead to miscommunication, delayed deliveries, and no clear feedback mechanism. The absence of digital scheduling and weekly meal planning also causes inefficiencies in kitchen operations and meal preparation.

### 1.3 PROPOSED SYSTEM

The proposed Tiffinly system resolves the limitations of the existing manual and semi-digital services by offering a fully integrated web-based solution. It allows customers to choose meal plans, customize preferences, handle payments and track deliveries through a centralized dashboard. Admins and delivery partners are provided with tools to manage operations efficiently from their dashboards.

# SYSTEM ANALYSIS

## 2. SYSTEM ANALYSIS

## 

### 2.1 PROBLEM DEFINITION

In the existing meal delivery ecosystem, customers face challenges in accessing healthy meals on a consistent basis. Many rely on unstructured local services that offer limited customization, inconsistent quality, and no systematic scheduling or tracking. Manual processes dominate both user requests and meal preparation, leading to missed deliveries, lack of transparency, and absence of real-time feedback.

There is also no provision for customizing plans based on dietary preferences. Administrators and delivery partners work in silos, with little to no automation in monitoring or communication. This results in operational inefficiencies and dissatisfied customers.

#### 2.1.1 ADVANTAGES OF PROPOSED SYSTEM

The proposed Tiffinly platform aims to overcome the limitations of the existing system through a modern, digital interface that connects customers, admins, and delivery agents on a unified platform.

**Advantages include:**

* **Customization:** Customers can select plan based on dietary preference
* **Time Efficiency:** Customers schedule entire subscription duration deliveries in one go, reducing daily ordering stress.
* **Automation:** Subscription management, meal selections, payments, and order tracking are fully digitized.
* **Transparency:** Meal plans, prices, delivery status, and order history are clearly accessible.
* **Feedback Loop:** Built-in rating and review system ensures continuous improvement.
* **Role Separation:** Customers, admins, and delivery partners access the system through dedicated interfaces.

### 2.2 FEASIBILITY ANALYSIS

Feasibility study is a test of a system proposal according to its workability, ability to meet user needs and effective use of resources. The objective of feasibility is not to solve the problem but to acquire a sense of its scope. The main aim of the feasibility study is to test the technical, social and economic feasibility of the system.

The feasibility study can be classified into the following categories:

* Operational Feasibility
* Technical Feasibility
* Economic Feasibility

#### 2.2.1 OPERATIONAL FEASIBILITY

The proposed system is highly user-friendly, with intuitive interfaces for all user roles. End-users (customers) benefit from a smooth ordering experience, while admins and delivery partners are provided with tools to streamline their workflow. Workload is minimized through automation, and system notifications enhance responsiveness.

#### 2.2.2 TECHNICAL FEASIBILITY

Technical feasibility deals with hardware as well as software requirements and to what extend it can support the proposed system. The system uses standard and widely-supported technologies:

* Frontend: HTML, CSS, JavaScript
* Backend: PHP
* Database: MySQL  
  All components are supported by XAMPP, making it easy to develop, test, and deploy the system on any standard computing environment without expensive infrastructure.

The hardware required is an android phone and software is Android Studio. If the necessary requirements are made available with the system, then the proposed system is said to be technically feasible.

#### 2.2.3 ECONOMIC FEASIBILITY

Economic feasibility is an important factor. Since the existing system is manual on the feasibility for wrong data entry is higher and consumes a lot of time and can occur errors. But the proposed system aims at processing of information’s efficiently, thus saving the time. The new system need only a system and which is already available therefore the cost is negligible. Proposed system uses validation check so there are no errors. Even though an initial investment has to be made on the software and the hardware aspects, the proposed system aims at processing of information’s efficiently. Thus, the benefits acquired out of the system are sufficient enough for the project to be undertaken. So, the proposed system is economically feasible.

Since the development is done using open-source technologies and local resources (e.g., laptops running XAMPP), the financial cost is minimal. Once implemented, the automated system significantly reduces human resource dependency and operational costs. It is a cost-effective solution with a high return on investment in terms of reliability and customer satisfaction.

### 2.3 RECOMMENDED IMPLEMENTATIONS

Two principle sources of data are:

* Written documents
* Data from the persons, who are involved in the operation of the system under study.

For successful implementation, the following methods were used during the research and planning phases:

* Questionnaires
* Personal Interviews
* Observations

**Questionnaires**

Questionnaires are one of the best ways to gather opinions and preferences from end customers. In this project, questionnaires were shared with students, working professionals, and homemakers who typically use tiffin services. The aim was to understand their food habits, delivery expectations, and level of satisfaction with current systems. Most customers preferred a weekly meal planning system with customization options. The data collected helped shape the user experience features in Tiffinly.

**Personal Interviews**

Personal interviews are a direct and reliable method to gather detailed information. This method was the primary source of fact-finding for the Tiffinly project. Discussions were held with local tiffin service providers, delivery personnel, and students living away from home. They shared insights about issues in the current manual system, such as miscommunication, forgotten orders, and irregular deliveries. Their suggestions for features like scheduled delivery slots, customizable plans, and order tracking were carefully considered and included in the system.

**Observations**

Observing how local food delivery services operate on a daily basis provided valuable insights. Meals were usually prepared based on calls or fixed routines, with handwritten notes used to track orders. There was no proper system to manage customer preferences, timing, or delivery updates. Watching these manual processes in action helped us understand the gaps and design a digital platform that simplifies operations for admins, enhances convenience for customers, and streamlines delivery for partners. The structure of these observations strongly influenced the database and workflow design of the Tiffinly system.

**SOFTWARE**

**REQUIREMENT SPECIFICATIONS**

## 3. SOFTWARE REQUIREMENT SPECIFICATION

### 3.1 INTRODUCTION

Requirements specification is the starting step for the development activities. It is currently one of the weak areas of software engineering. During requirement specification, the goal is to produce a document of the client’s requirements. This document forms the basis of development and software validation. The basic reason for the difficulty in software requirements specification comes from the fact that there are three interested parties- the client, the end customers and the software developer.

### 3.2 PURPOSE

The origin of most software systems is in the need of a client, who either wants to automate an existing manual system or desires a new software system. The software system itself is created by the developer. Finally, the completed system will be used by the end customers. Thus, there are three major parties interested in a new system: the client, the customers and the developer. A basic purpose of software requirements specification is to bridge the communication gap. SRS is the medium through which the client and user needs are accurately specified. Indeed, SRS forms the basis of software development. A good SRS should satisfy all the parties, something very hard to achieve, and involves trade-offs and persuasion.

Another important purpose of developing an SRS is helping the clients understand their own needs. Advantages are:

* An SRS establishes the basis for agreement between the client and the supplier on what the software product will do
* An SRS provides a reference for validation of the final product
* A high-quality SRS is a prerequisite to high-quality software.
* A high-quality SRS reduces the development cost.

### 3.3 SCOPE

#### 3.3.1 SYSTEM STATEMENT OF SCOPE

The Tiffinly Meal Subscription Platform is developed to provide a complete digital solution for subscribing to meal plans. It offers a user-friendly web interface where customers can register, browse meal options, customize plans, and subscribe to affordable food packages.

The system supports real-time order scheduling and delivery tracking. Customers can view upcoming and past orders, give feedback, and securely manage payments. Admins manage the entire ecosystem including meals, customers, delivery partners, subscriptions, and delivery coordination. Delivery partners interact through their dedicated dashboard to accept orders and update delivery statuses.

All roles access the system through secure, role-based logins. The platform is built with scalability in mind and supports both local and campus-wide use cases.

### 3.4 TECHNICAL OVERVIEW

#### 3.4.1 USER CHARACTERISTICS

The Tiffinly platform supports three user types:

1. **Customer:**
   * Can register and subscribe to meal plans.
   * Can schedule delivery preferences.
   * Can make payments, view history, rate services and raise inquiries.
2. **Admin:**
   * Has full access to all customers and subscription records
   * Manages meal items, subscription plans, customers, and deliveries
   * Analyse feedback and respond to inquiries.
3. **Delivery Partner:**
   * Can view available orders and accept deliveries
   * Update delivery statuses (Out for Delivery, Delivered, Cancelled)
   * Track delivery history
   * Monitor performance and earnings

Each role sees only its relevant dashboard; features and permissions are hidden from non-authorized customers.

### 3.5 FUNCTIONAL REQUIREMENTS

The functional requirements of this website are as follows: \

1. **User Management:**

* User Registration Process:

Based on the selected role,

* Customers register with their name, email, phone number, password and a security question with answer.
* Delivery partners register with their name, email, phone number, password, a security question with answer, vehicle type, vehicle number, license number, license proof, Aadhaar number and availability.
* User Authentication:
* Users log in to their respective dashboards by entering their registered email and password.
* The system verifies credentials against stored data to allow access.
* Users can securely log out, ending their active sessions to prevent unauthorized access.

1. **Meal Plan Management:**

* Customers can view available meal plans.
* Meal details (day-wise) for a week are displayed for available plans.
* Customers can choose dietary preferences (basic plan feature) or customize meals (premium plan feature) and set delivery preferences for the subscription duration based on schedule chosen (Weekdays, Extended Week (including Saturday), Full Week (including Sunday)).

1. **Subscription and Scheduling:**

* Customers can subscribe to meal plans (basic/premium).
* Customers select preferred delivery time slots and address for the subscription duration.
* Admin can overview subscription records.

1. **Order and Delivery Management:**

* Order records store all delivery, meal, and user details.
* Delivery partners can accept new orders.
* Delivery status can be updated (Pending → Out for Delivery → Delivered → Cancelled).

1. **Feedback and Rating System:**

* After delivery, customers can rate meals, service and platform and leave feedback.
* Admins can view all feedback to monitor service quality and respond to inquiries.

1. **Admin Dashboard:**

Admins can:

* Access user records
* Monitor subscription plans data
* Manage Meal plans, categories and items
* Analyse feedback and monitor delivery statuses
* Respond to inquiries
* Manage partner payouts

### 3.6 NON-FUNCTIONAL REQUIREMENTS

The non-functional requirements of this website are as follows:

**1. Usability:**

* Clean, user-friendly interface for all roles (customer, admin, delivery partner).
* Clear navigation and structured layout for easy meal selection and order tracking.

**2. Reliability:**

* The system should perform reliably in over 95% of daily usage scenarios.
* Ensures consistent data processing for subscriptions and deliveries.

**3. Availability:**

* Accessible globally 24/7.
* Backup and recovery mechanisms to restore service within an hour of a failure.

**4. Security:**

* Database backed up hourly to avoid data loss.
* User passwords and sensitive data will be encrypted.
* Role-based access control will prevent unauthorized access.

**5. Performance:**

* Quick response times and minimal latency under normal conditions.
* Efficient handling of large volumes of data.

**6. Scalability:**

* System designed to support growing user base and delivery load.
* Modular architecture supports the addition of new features without major restructuring.

**7. Maintainability:**

* Codebase is clean and modular with proper documentation.
* Regular updates and maintenance tasks will be scheduled to ensure system stability.

### 3.7 STATED REQUIREMENTS

**3.7.1 GENERAL REQUIREMENTS**

The Tiffinly platform consists of three main user modules: Customer, Admin, and Delivery Partner.

**1. Login**

* Only registered customers, delivery partners and admins can log in to the system to avail the services.
* The registered customers, delivery partners and admins use their email ID and password to log in.
* The password must contain an uppercase and lowercase alphabetic characters, numbers, and special characters with a minimum requirement of 8 characters.
* Admins and delivery partners will be redirected to their respective panel when logging in with the predefined email ID and password using the same login form.
* Customers will be directed to the user dashboard of the site when the login is successful.

**2. Sign Up**

* New customers and delivery partners need to register in order to use the platform's services.
* This includes several fields:
  + Full Name
  + Email Address
  + Phone Number
  + Password
  + Confirm Password
  + Security question with answer
  + Vehicle Type, Vehicle Number, License Number, License Proof, Aadhaar Number and Availability (Additional fields to be filled up if registering as a delivery partner)

1. **Admin Panel**

The admin can:

* Access user records
* Monitor subscription plans data
* Manage Meal plans, categories and items
* Analyse feedback and monitor delivery statuses
* Respond to inquiries
* Manage partner payouts

1. **Explore Meal Plans**

* Customers can view available meal plans.
* Meal details (day-wise) for a week are displayed for available plans.
* Customers can choose dietary preferences (basic plan feature) or customize meals (premium plan feature) and set delivery preferences for the subscription duration based on schedule chosen (Weekdays, Extended Week (including Saturday), Full Week (including Sunday)).

1. **Customize & Purchase Subscription**

* Customers can select preferred delivery time slots and addresses and save it.
* Subscriptions are added to cart and paid online.

1. **Order & Delivery Management**

* Customers can monitor order status every day.
* Delivery partners can accept orders and update delivery status.

1. **Rating & Feedback**

* Customers rate meals, services and platform and give feedbacks.
* Admins can view reviews for performance monitoring.

#### 3.7.2 INPUTS

The Tiffinly system takes the following inputs to provide its services:

* **User Registration**:
* Full Name
* Email Address
* Phone Number
* Password, Confirm Password
* Security Question with Answer
* Vehicle Type, Vehicle Number, License Number, License Proof, Aadhaar Number and Availability (Additional fields to be filled up if registering as a delivery partner)
* **Login**:
* Registered Email Address
* Password
* **Meal Customization:**
* Plan Type (basic or premium)
* Schedule selected (Weekdays, including Saturday, including Sunday)
* Dietary preference (veg/non-veg) for basic plan and meal customization for premium plan
* **Subscription Checkout:** 
  + Address and Time Slot,
  + Phone Number and Payment details
* **Delivery Updates:**
* Order ID
* Meal Type and Meal
* Delivery status (Pending / Out for Delivery / Delivered)
* **Feedback:**
* Rating (1–5) for service, meals and platform
* Comment

#### 3.7.3 PROCESSING

The system will perform the following key processing tasks:

* **Data Validation**:
  + Validate email format, contact number and password complexity.
  + Ensure required fields are not empty.
* **Meal Planning & Customization:** 
  + Customers select meals based on plan type and delivery schedule.
* **Subscription Management:**
* Stores and retrieves user subscription data
* Calculates total cost dynamically
* **Payment**:
  + Customers add products to cart, place orders, and track order history.
  + Secure payment processing.
* **Delivery Scheduling:**
* Delivery partners can view available orders and accept deliveries.
* Updates order delivery status.
* **Feedback Management:**
* Captures and stores user feedback.
* Admins can view reviews and respond to inquiries.

#### 3.7.4 OUTPUTS

The system produces the following outputs:

* **Meal Plan Summary:** 
  + Selected meals and delivery schedule
* **Order & Subscription Details:**
* Order ID, user info, meals, and delivery time
* **Order and Payment Confirmation**:
* Order details, payment confirmation.
* **Delivery Tracking:**
* Real-time order status viewable by user and admin.
* **Feedback Reports:**
* Ratings and comments per meal/order.
* **Admin Reports:**
* Summary of subscriptions, user activity, delivery logs.

### 3.8 EXTERNAL INTERFACE REQUIREMENTS

#### 3.8.1 USER INTERFACES

All user interfaces will be GUI interfaces, designed to be clean, accessible, and fast offering high functionality. The interfaces will have a pleasing appearance and intuitive design.

* **Design and Appearance**:
  + The interface will use suitable design elements and pleasing colours to create a comfortable and attractive environment for customers.
  + Consistent design themes will be maintained across all pages.
* **Usability Components**:
* Textboxes, dropdowns, calendar pickers, sliders and buttons will be used to facilitate easy data entry.
* Clear labels and instructions will be provided for all input fields.
  + Intuitive navigation elements will help customers move seamlessly through the platform.
* **Responsive Design**:
* The user interface will be responsive to ensure usability across various devices, including desktops, tablets, and smartphones.

#### 3.8.2 HARDWARE INTERFACES

The system needs a computer or any other smart devices with network availability to access the web application. No other external hardware is required.

**Hardware Specification**

* **Processor**: Intel Pentium or higher
* **RAM**: 256 MB or higher
* **Hard Disk Drive**: 100 MB required on disk
* **Keyboard**: Standard QWERTY keyboard

**Implementation Specification**

* **Operating System**: Windows OS

#### 3.8.3 SOFTWARE INTERFACES

**Software Specification**

* **Operating System**: Windows 11 or equivalent
* **Browser**: Chrome, Firefox, Edge (latest versions)
* **Web Server**: XAMPP with Apache and MySQL
* **Programming Language**: PHP
* **Database**: MySQL
* **Frontend**: HTML, CSS, JavaScript

# SYSTEM DESIGN

## 4. SYSTEM DESIGN

### 4.1 INTRODUCTION

The design phase aims to develop a solution for the problems identified during the analysis phase. This phase marks the transition from understanding the problem to creating a solution. System design details the required features and operations, including screen layouts, business rules, process diagrams, pseudocode, and other relevant documentation.

During this phase, the overall structure and specifics of the software are defined. This includes determining the number of tiers needed for the architecture, designing inputs and outputs, and establishing database and data structure designs. Proper analysis and design are critical to the development cycle since errors in this phase can be costly to fix later. Therefore, careful attention is given during the design phase.

The logical framework of the product and its physical attributes are outlined during this stage. The operating environment is set up, and key resources are identified. Any element that requires user input or approval must be documented and reviewed by the user. The physical aspects of the system are specified, and a detailed design is prepared.

Subsystems identified during design are used to create a detailed system structure. Each subsystem is divided into one or more design units or modules, and detailed logic specifications are created for each module. The module logic is typically described in a high-level design language, which is independent of the final implementation language.

A good design should consider:

* **Promptness**: The design should be straightforward and clear, guiding customers to their desired outcomes intuitively.
* **Memory Load**: Research shows that customers can retain about six words in their short-term memory. The number of choices presented to customers should ideally be four or fewer to prevent confusion and forgetfulness.
* **Service Reachability**: Customers dislike going through many steps to access a service. More than five steps can cause impatience, so minimizing the number of steps helps reduce frustration.
* **Navigation**: Customers should easily navigate back and forth between different steps, allowing them to access various parts of the dialog seamlessly.
* **Phonetic Similarity**: Avoid choices with similar pronunciations to reduce confusion and ensure customers can clearly distinguish between options.
* **Error Handling**: Implementing graceful error handling helps decrease dependency on operators by managing mistakes effectively.
* **User Updates**: Keep customers informed about the ongoing process to maintain their engagement and understanding.

For the general design, one or more potential designs are proposed and broadly sketched. These alternatives are then presented to the customers, who choose the design that best suits their requirements while staying within project constraints.

The detailed design stage specifies the user interface, database, programs, hardware, and training and system documentation. Several structured techniques are used during the design phase. To design the software components, the designer transforms the automated processes in the physical data flow diagram into a program structure chart, which decomposes software processes into detailed modules and shows control paths between modules.

### 4.2 DESIGN METHODOLOGY

#### 4.2.1 INPUT DESIGN

Input design focuses on converting user-oriented inputs into a format recognizable by the computer. Collecting input data is one of the costliest parts of the system in terms of equipment, time, and user involvement. The goal of input design is to make data entry as simple, logical, and error-free as possible.

Input design serves as the bridge between the information system and its customers, transforming transaction data into a form suitable for processing. This process can involve reading data from printed documents or directly keying data into the system. Effective input design controls the amount of input required, minimizes errors, avoids delays and extra steps, and keeps the process straightforward.

System analysis determines the following input design details:

* **What data to input**
* **What medium to use**
* **How the data is arranged and coded**
* **Data items and transactions requiring validation to detect errors**

Activities involved in input design include:

* **Data Recording**: Collecting data for input.
* **Data Verification**: Ensuring the accuracy of the collected data.
* **Data Conversion**: Transforming data into the required format.
* **Data Validation**: Checking data for errors.
* **Data Correction**: Fixing any identified errors.

#### 4.2.1 OUTPUT DESIGN

Output design involves creating necessary outputs tailored to meet the requirements of various customers. It is essential to approach the design of computer outputs in a well-considered manner. Outputs refer to any information generated by the information system, whether in printed or displayed form. Analysts design computer outputs by identifying specific outputs required to fulfil system requirements.

Computers serve as crucial sources of information for customers. Efficient and thoughtful output design enhances the system's interaction with customers and supports decision-making processes.

When designing outputs, system analysts must achieve the following objectives:

* Determine the information to be presented
* Decide whether to display, print, or verbally communicate the information, and select the appropriate output medium
* Format the information in an easily understandable manner

Output design is critical to the success of any system as it bridges the gap between the user and the system's operations. Effective output design includes specifications and procedures for presenting data clearly to customers. Customers should never be left uncertain about system activities, as appropriate error messages and acknowledgment messages are provided.

#### 4.2.3 CODE DESIGN

The coding phase transforms the detailed software design into a programming language. It translates the software's detailed design representation into executable code. Code design aims to minimize the lines of code used while modularizing the implementation. Modules hide complexity by encapsulating executable statements under named functions or procedures. Effective information hiding enhances program understanding at higher abstraction levels. Module names should accurately describe their actions to avoid confusion.

In this software, a modularized approach is employed with different functions created for various operations, each named to reflect its action.

#### 4.2.4 DATABASE DESIGN

Database design identifies relevant data relationships and defines tables using standard methods. Each table's attributes are carefully defined to optimize database performance, ensure data integrity, minimize redundancy, and enhance security.

A database system is a computer representation of an information system designed to handle integrated data efficiently. It minimizes redundancy to provide quick, flexible, and cost-effective information access. Database design considers several specific objectives:

* + Controlled redundancy
  + User-friendly interface
  + Data independence
  + Cost-effective data retrieval
  + Accuracy and integrity
  + Failure recovery
  + Privacy and security
  + Performance optimization

Database design involves creating multiple views of data, including logical and physical views. The logical view represents data independently of its storage, focusing on how customers and programmers interact with it. The physical view describes how data are stored and accessed in physical storage.

Each table in the database typically includes a primary key, a unique column (or combination of columns) that uniquely identifies each record. Primary keys ensure data integrity by enforcing uniqueness and cannot contain null values.

Normalization is employed to organize database data, minimizing redundancy and anomalies during data insertion, update, and deletion operations.

**4.3 SYSTEM ARCHITECTURE AND PROCESS FLOW**

## UML DIAGRAMS

**4.3.1 USE CASES**

### REGISTRATION

|  |  |
| --- | --- |
| Use Case Id: | CL\_UC\_01 |
| Use Case Name: | Registration |
| Created by: | Mekhna |
| Date Created: | 28-06-2025 |
| Description: | Allows new customers to register for an account on the platform. |
| Primary actor: | User, Delivery Partner |
| Secondary actor: | None |
| Precondition: | User navigates to the registration page |
| Postcondition: | User account is successfully created. |
| Main flow: | 1. User navigates to the registration page. 2. User provides required details: full name, email, phone no, and password. 3. User submits the registration form. 4. System validates the information and creates the user account. 5. Use case ends. |

### LOGIN

|  |  |
| --- | --- |
| Use Case Id: | CL\_UC\_02 |
| Use Case Name: | Login |
| Created by: | Mekhna |
| Date Created: | 21-06-2025 |
| Description: | This use case enables customers to access the system by entering their credentials. There are three user roles: admin, delivery partner and customer. The admin logs in using their admin email and password, while the customers and delivery partners uses their own email and password. Once logged in, customers can purchase, delivery partners can view available orders and accept deliveries and admins can add or edit items on the website. |
| Primary actor: | User/admin/delivery partner |
| Secondary actor: | None |
| Precondition: | The user should have a valid account. |
| Postcondition: | The system displays relevant homepage. |
| Main flow: | 1. The user goes to the login page. 2. The user inputs their registered email and password. 3. The user submits the login form. 4. The system checks the user's credentials. 5. If the credentials are correct, the system grants the user access. 6. The use case concludes. |

### VIEW PRODUCTS

|  |  |
| --- | --- |
| Use Case Id: | CL\_UC\_03 |
| Use Case Name: | View Product |
| Created by: | Derin & Sheethal |
| Date Created: | 24-06-2024 |
| Description: | This use case allows customers to view details of products available on the platform. Both admins and customers can browse products. Admins can see options to edit product details, while customers can see product descriptions, prices, and availability. |
| Primary actor: | User/admin |
| Secondary actor: | None |
| Precondition: | The user should be logged into their account. |
| Postcondition: | The system displays the detailed information of the selected product. |
| Main flow: | 1. The user navigates to the product catalog page. 2. The user selects a product they wish to view. 3. The system retrieves the product details. 4. The system displays the product details, including description, price, and availability. 5. If the user is an admin, they also see options to edit the product details. 6. The use case concludes. |

### PURCHASE PRODUCT

|  |  |
| --- | --- |
| Use Case Id: | CL\_UC\_04 |
| Use Case Name: | Purchase Product |
| Created by: | Derin & Sheethal |
| Date Created: | 24-06-2024 |
| Description: | This use case allows customers to purchase products from the platform. Customers select products, proceed to checkout, and complete the payment process. |
| Primary actor: | User |
| Secondary actor: | None |
| Precondition: | The customer must be logged into their account and have selected items to purchase. |
| Postcondition: | The system confirms the purchase, updates the order status, and processes the payment. |
| Main flow: | 1. The customer navigates to the product catalog and selects the items they wish to purchase. 2. The customer adds the selected items to their shopping cart. 3. The customer proceeds to the checkout page. 4. The system displays the order summary and prompts the customer to enter payment information. 5. The customer enters their payment details and submits the payment. 6. The system processes the payment and updates the order status. 7. The system confirms the purchase and provides an order confirmation to the customer. 8. The use case concludes. |

### MANAGE PRODUCTS

|  |  |  |
| --- | --- | --- |
| Use Case Id: | CL\_UC\_05 | |
| Use Case Name: | Manage Products | |
| Created by: | Derin & Sheethal | |
| Date Created: | 24-06-2024 | |
| Description: | This use case allows admins to manage products on the platform. Admins can add new products, edit existing product details, and remove products as needed. This functionality ensures that the product catalog remains up-to-date and accurate. | |
| Primary actor: | Admin | |
| Secondary actor: | None | |
| Precondition: | The admin must be logged into their account with appropriate privileges. | |
| Postcondition: | The product catalog is updated with the new, edited, or removed product details. | |
| Main flow: | 1. The admin navigates to the product management page. 2. The admin selects an option to add, edit, or remove a product.   • **To add a Product:**   1. The admin enters the product details, including name, description, price, and availability. 2. The admin submits the new product information. 3. The system updates the product catalog with the new product. | |
|  | • | **To Edit a Product:**   1. The admin selects the product to be edited from the catalog. 2. The admin modifies the product details as needed. 3. The admin submits the updated product information. 4. The system updates the product catalog with the edited product details. |
|  | • | **To Remove a Product:**   1. The admin selects the product to be removed from the catalog. 2. The admin confirms the removal. 3. The system removes the product from the catalog.   3. The system confirms the action (add, edit, or remove) and updates the product catalog accordingly.  The use case concludes. |

### BOOKING CONSULTATION AND SERVICES

|  |  |
| --- | --- |
| Use Case Id: | CL\_UC\_07 |
| Use Case Name: | Book Consultation and Services |
| Created by: | Derin & Sheethal |
| Date Created: | 24-06-2024 |
| Description: | This use case allows customers to book consultations or services offered on the platform. Customers can select the type of consultation or service, choose a suitable time slot, and confirm their booking.  The system manages and schedules the bookings. |
| Primary actor: | User |
| Secondary actor: | None |
| Precondition: | The user must be logged into their account and have access to available consultations or services. |
| Postcondition: | The booking is confirmed, and the system updates the schedule with the new appointment. |
| Main flow: | 1. The user navigates to the consultations or services booking page. 2. The user selects the type of consultation or service they wish to book. 3. The user chooses a preferred time slot from the available options. 4. The user provides any additional required information (e.g., details about the consultation or service). 5. The user submits the booking request. 6. The system updates the schedule and sends a confirmation to the user. 7. The use case concludes. |

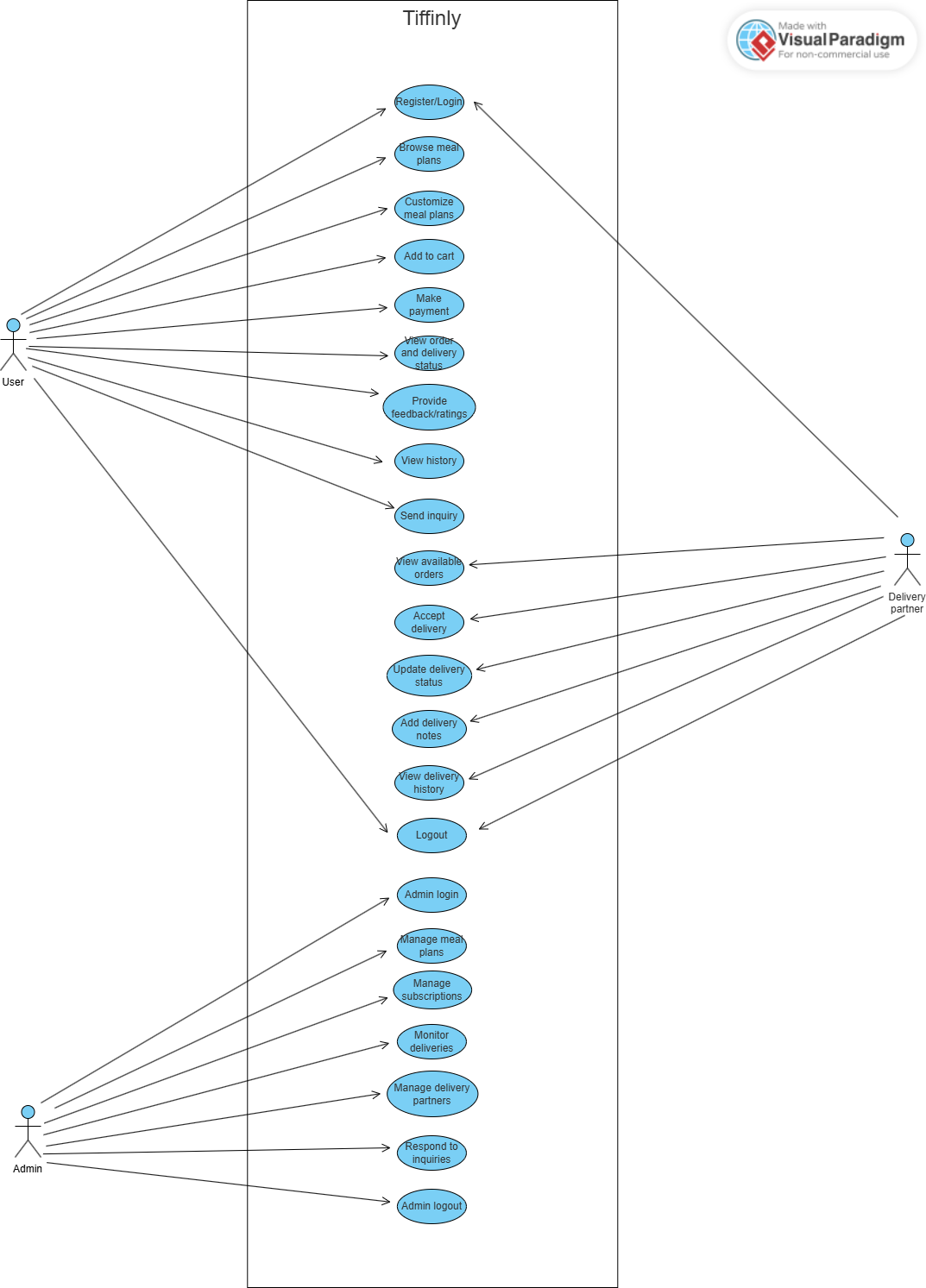
### MANAGE BOOKING AND CONSULTAION

|  |  |
| --- | --- |
| Use Case Id: | CL\_UC\_08 |
| Use Case Name: | Manage Booking and Consultation |
| Created by: | Derin & Sheethal |
| Date Created: | 24-06-2024 |
| Description: | This use case allows admins or service providers to manage bookings and consultations on the platform. They can review, approve, modify, or cancel bookings and consultations to ensure smooth scheduling and service delivery. |
| Primary actor: | Admin |
| Secondary actor: | None |
| Precondition: | The admin or service provider must be logged into their account with appropriate privileges. |
| Postcondition: | The booking and consultation details are updated as per the actions taken  (approved,or canceled). |
| Main flow: | 1. The admin or service provider navigates to the booking and consultation management page. 2. The admin or service provider reviews the list of upcoming and pending bookings and consultations 3. The admin or service provider selects a booking or consultation to manage.   • **To Approve a Booking/consultation:**  1. The admin or service provider reviews the booking details. |
|  | 1. The admin or service provider approves the booking or consultation. 2. The system updates the status of the booking or consultation to "Accepted" and sends a confirmation to the user.   • **To Cancel a Booking/Consultation:**   * 1. The admin or service provider selects the booking or consultation to be canceled.   2. The admin or service provider confirms the cancellation.   3. The system updates the status to "Canceled" and sends a cancellation notice to the user.  1. The system confirms the action   (accept, or cancel) and updates the booking and consultation records accordingly.   1. The use case concludes. |

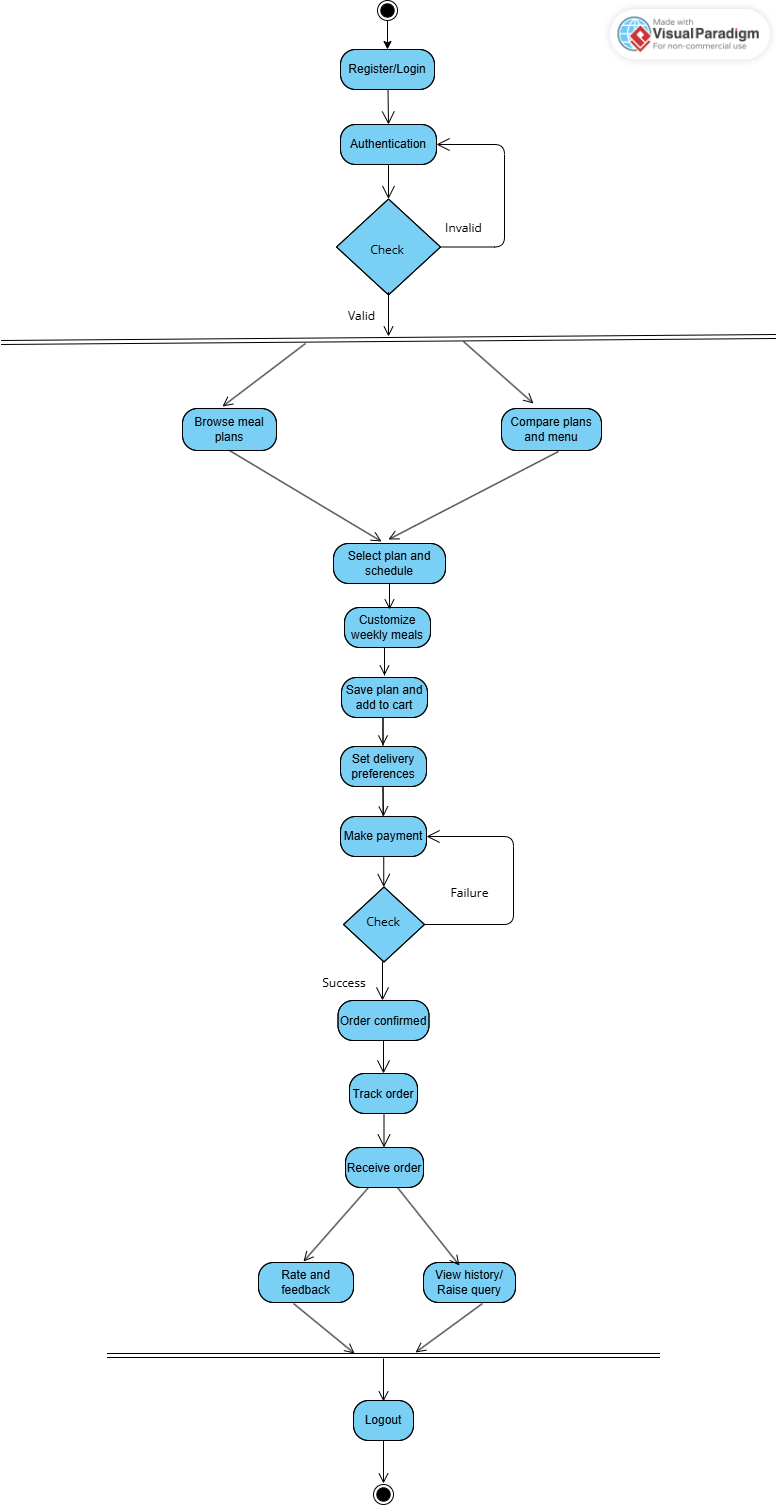
### LOG OUT

|  |  |
| --- | --- |
| Use Case Id: | CL\_UC\_10 |
| Use Case Name: | Log Out |
| Created by: | Derin & Sheethal |
| Date Created: | 24-06-2024 |
| Description: | This use case allows both customers and admins to log out of their accounts on the platform. Logging out ensures that the session is securely ended, and access is restricted until the user or admin logs in again. |
| Primary actor: | User/Admin |
| Secondary actor: | None |
| Precondition: | The user or admin must be logged into their account. |
| Postcondition: | The user or admin must be logged into their account. |
| Main flow: | 1. The user or admin navigates to the log out option. 2. The user or admin selects the option to log out. 3. The system terminates the session of the user or admin. 4. The system redirects the user or admin to the login page. 5. The system confirms that the user or admin has been successfully logged out. 6. Use case ends. |

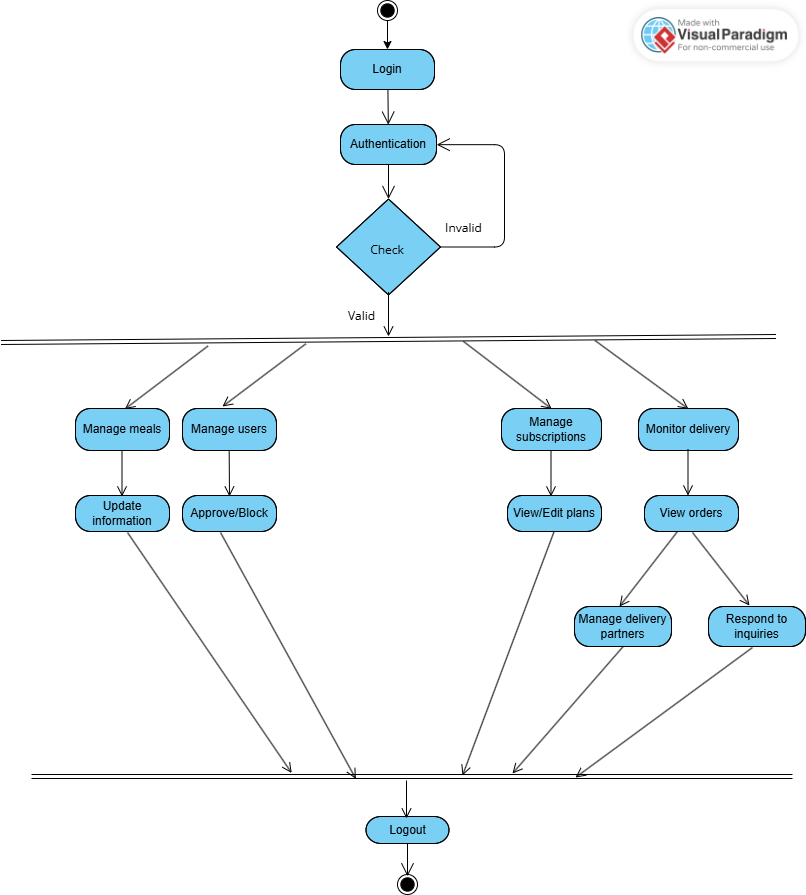
#### 4.3.2 USECASE DIAGRAM



#### 4.3.3 ACTIVITY DIAGRAMS

 USER SIDE

ADMIN SIDE



Manage reviews

DELIVERY PARTNER SIDE

### 

### 4.4 MODULE DETAILS

There are five main modules in this website:

* **Login Module**
* **Sign-Up Module**
* **Product Discovery Module**
* **Consultation Booking Module**
* ➢ **Admin Dashboard Module**

1. **Login Module**

The login module allows registered customers and administrators to securely access the Celestia website. It verifies user credentials (email and password) to grant access to personalized services such as browsing designs, purchasing products, and managing consultations.

1. **Sign Up Module**

The sign-up module enables new customers to create accounts on Celestia. Customers provide personal details including name, email, phone number, and password to register and access features like saving favourite designs, tracking orders, and booking consultations.

1. **Product Discovery Module**

The product discovery module allows customers to explore trendy design ideas and browse products offered on Celestia. Customers can view detailed product information, and add items to their shopping cart for purchase.

1. **Consultation Booking Module**

The consultation booking module enables customers to schedule virtual or in-person consultations with design experts through Celestia and receive confirmation of their consultation bookings.

1. **Admin Dashboard Module**

The admin dashboard module provides administrators with tools to oversee consultation schedules, update website content. Admins can ensure smooth platform operation and maintain data integrity.

### 4.5 PERFORMANCE CONSIDERATIONS

**Hardware Requirements**

The system is designed to perform optimally with a minimum of 4GB RAM and is compatible with Windows OS versions and higher.

### 4.6 SECURITY CONSIDERATIONS

**Access Control**

* **Authorized Access**: Only customers with valid usernames and passwords are allowed to access the Celestia Interior and Exterior Design platform.
* **Login Security**: The login process includes robust security measures to authenticate customers and prevent unauthorized access.

### 4.7 TABLE DESIGN

**1.Table name: users**

|  |  |  |  |
| --- | --- | --- | --- |
| Sl. No. | Type | Constraint | Description |
| 1 | int(11) | PK | User ID |
| 2 | varchar(100) | NOT NULL | Full name |
| 3 | varchar(100) | UNIQUE, NOT NULL | User email |
| 4 | varchar(15) | UNIQUE, NOT NULL | Phone number |
| 5 | varchar(255) | NOT NULL | Hashed password |
| 6 | enum('user','admin','delivery') | DEFAULT 'user' | Role type |
| 7 | varchar(255) | NULL | Security question for reset |
| 8 | varchar(255) | NULL | Security answer |
| 9 | timestamp | DEFAULT current\_timestamp() | Account created |
| 10 | timestamp | DEFAULT current\_timestamp() ON UPDATE | Last update |

**2.Table name: auth\_tokens**

|  |  |  |  |
| --- | --- | --- | --- |
| Sl. No. | Type | Constraint | Description |
| 1 | INT | PK, AUTO\_INCREMENT, NOT NULL | Unique token record ID |
| 2 | INT | FK → users(user\_id), NOT NULL, ON DELETE CASCADE | References the user who owns the token |
| 3 | VARCHAR(32) | NOT NULL, INDEX | Public identifier for the token (used for quick lookup) |
| 4 | VARCHAR(64) | NOT NULL | Secret validator part of the token (stored hashed for security) |
| 5 | DATETIME | NOT NULL | Expiration date and time for the token |

**3.****Table name: addresses**

|  |  |  |  |
| --- | --- | --- | --- |
| Sl. No. | Type | Constraint | Description |
| 1 | int(11) | PK, NOT NULL | Unique ID for address |
| 2 | int(11) | FK → users(user\_id) | Reference to user |
| 3 | enum('home','work') | NOT NULL | Type of address |
| 4 | varchar(255) | NOT NULL | Primary address line |
| 5 | varchar(255) | NULL | Secondary address line |
| 6 | varchar(100) | NOT NULL | City |
| 7 | varchar(100) | NOT NULL | State |
| 8 | varchar(10) | NOT NULL | Postal code |
| 9 | varchar(255) | NULL | Landmark details |
| 10 | tinyint(1) | DEFAULT 1 | Marks default address |

**4.Table name: delivery\_partner\_details**

|  |  |  |  |
| --- | --- | --- | --- |
| Sl. No. | Type | Constraint | Description |
| 1 | int(11) | PK | Partner ID (linked to user\_id) |
| 2 | varchar(50) | NOT NULL | Vehicle type |
| 3 | varchar(20) | NOT NULL | Registration number |
| 4 | varchar(30) | NOT NULL | License no. |
| 5 | varchar(255) | NULL | Uploaded license proof |
| 6 | varchar(20) | NULL | Aadhaar ID |
| 7 | enum('Part-time','Full-time') | NOT NULL | Work type |

**5.Table name: deliveries**

|  |  |  |  |
| --- | --- | --- | --- |
| Sl. No. | Type | Constraint | Description |
| 1 | int(11) | PK, NOT NULL | Delivery record ID |
| 2 | int(11) | FK → subscriptions(subscription\_id) | Linked subscription |
| 3 | date | NOT NULL | Scheduled date |
| 4 | enum('scheduled','out\_for\_delivery','delivered','cancelled') | DEFAULT 'scheduled' | Delivery status |
| 5 | enum('unpaid','paid') | DEFAULT 'unpaid' | Payment state |
| 6 | time | NULL | Scheduled time |

**6.Table name: delivery\_issues**

|  |  |  |  |
| --- | --- | --- | --- |
| Sl. No. | Type | Constraint | Description |
| 1 | int(11) | PK | Issue ID |
| 2 | int(11) | FK | Related assignment |
| 3 | int(11) | FK | Subscription |
| 4 | int(11) | FK | Delivery partner |
| 5 | varchar(50) | NULL | Meal affected |
| 6 | varchar(100) | NULL | Type of issue |
| 7 | enum('open','resolved') | DEFAULT 'open' | Resolution state |
| 8 | text | NULL | Details |

**7.Table name: delivery\_assignments**

|  |  |  |  |
| --- | --- | --- | --- |
| Sl. No. | Type | Constraint | Description |
| 1 | int(11) | PK | Assignment ID |
| 2 | int(11) | FK | Subscription linked |
| 3 | date | NOT NULL | Assigned date |
| 4 | int(11) | FK → delivery\_partner\_details(partner\_id) | Delivery partner |
| 5 | timestamp | DEFAULT current\_timestamp() | Assignment timestamp |
| 6 | enum('pending','out\_for\_delivery','delivered','cancelled') | DEFAULT 'pending' | Delivery progress |
| 7 | varchar(20) | NOT NULL | Breakfast/Lunch/Dinner |
| 8 | int(11) | FK → meals(meal\_id) | Assigned meal |
| 9 | int(11) | FK → partner\_payments(payment\_id) | Payment record |
| 10 | enum('unpaid','paid') | DEFAULT 'unpaid' | Partner payment |

**8.Table name: delivery\_preferences**

|  |  |  |  |
| --- | --- | --- | --- |
| Sl. No. | Type | Constraint | Description |
| 1 | int(11) | PK | Preference ID |
| 2 | int(11) | FK | User |
| 3 | enum('breakfast','lunch','dinner') | NOT NULL | Meal choice |
| 4 | int(11) | FK → addresses(address\_id) | Delivery address |
| 5 | varchar(50) | NULL | Preferred time |
| 6 | timestamp | DEFAULT current\_timestamp() | Created |
| 7 | timestamp | DEFAULT current\_timestamp() ON UPDATE | Updated |

**9.Table name: feedback**

|  |  |  |  |
| --- | --- | --- | --- |
| Sl. No. | Type | Constraint | Description |
| 1 | int(11) | PK | Unique feedback ID |
| 2 | int(11) | FK → users(user\_id) | Feedback given by user |
| 3 | enum('meal','service','platform') | NOT NULL | Type of feedback |
| 4 | tinyint(4) | NOT NULL, CHECK (1–5) | Rating value |
| 5 | text | NULL | Additional feedback |
| 6 | varchar(255) | NULL | Details of meal (if meal feedback) |
| 7 | date | NULL | Related delivery date (if service feedback) |
| 8 | timestamp | DEFAULT current\_timestamp() | Created timestamp |
| 9 | timestamp | DEFAULT current\_timestamp() ON UPDATE | Last updated |

**10.Table name: inquiries**

|  |  |  |  |
| --- | --- | --- | --- |
| Sl. No. | Type | Constraint | Description |
| 1 | int(11) | PK | Inquiry ID |
| 2 | int(11) | FK → users(user\_id), NULL allowed | User making inquiry |
| 3 | text | NOT NULL | Inquiry content |
| 4 | varchar(50) | DEFAULT 'general' | Category (general, technical, etc.) |
| 5 | text | NULL | Admin response |
| 6 | enum('pending','responded','closed') | DEFAULT 'pending' | Inquiry state |
| 7 | timestamp | DEFAULT current\_timestamp() | Logged timestamp |

**11.Table name: meals**

|  |  |  |  |
| --- | --- | --- | --- |
| Sl. No. | Type | Constraint | Description |
| 1 | int(11) | PK | Unique meal ID |
| 2 | int(11) | FK → meal\_categories(category\_id) | Meal category |
| 3 | varchar(100) | NOT NULL | Name of meal |
| 4 | text | NULL | Meal description |
| 5 | varchar(255) | NULL | Meal image path |
| 6 | tinyint(1) | DEFAULT 1 | Availability flag |

**12. Table name: meal\_categories**

|  |  |  |  |
| --- | --- | --- | --- |
| Sl. No. | Type | Constraint | Description |
| 1 | int(11) | PK | Category ID |
| 2 | varchar(50) | NOT NULL | Category name |
| 3 | En meal\_categories  enum('Breakfast','Lunch','Dinner') | NOT NULL | Meal type |
| 4 | enum('veg','non\_veg') | NOT NULL | Veg/Non-veg option |
| 5 | enum('breakfast','lunch','dinner') | NULL | Delivery slot |

**13. Table name: meal\_plans**

|  |  |  |  |
| --- | --- | --- | --- |
| Sl. No. | Type | Constraint | Description |
| 1 | int(11) | PK | Plan ID |
| 2 | varchar(50) | NOT NULL | Plan name |
| 3 | text | NULL | Plan description |
| 4 | enum('basic','premium') | NOT NULL | Type of plan |
| 5 | tinyint(1) | DEFAULT 1 | Availability |
| 6 | decimal(8,2) | NOT NULL | Base price per day |

**14. Table name: partner\_payments**

|  |  |  |  |
| --- | --- | --- | --- |
| Sl. No. | Type | Constraint | Description |
| 1 | int(11) | PK | Payment ID |
| 2 | int(11) | FK → delivery\_partner\_details(partner\_id) | Partner paid |
| 3 | int(11) | FK → subscriptions(subscription\_id) | Related subscription |
| 4 | date | NULL | Delivery date |
| 5 | decimal(10,2) | NOT NULL | Payment amount |
| 6 | int(11) | DEFAULT 0 | Number of deliveries covered |
| 7 | varchar(50) | NOT NULL | Method (e.g., Razorpay) |
| 8 | enum('success','failed','pending') | DEFAULT 'success' | Payment state |
| 9 | varchar(100) | NULL | Transaction reference |
| 10 | timestamp | DEFAULT current\_timestamp() | Created timestamp |

**15. Table name: payments**

|  |  |  |  |
| --- | --- | --- | --- |
| Sl. No. | Type | Constraint | Description |
| 1 | int(11) | PK | Payment ID |
| 2 | int(11) | FK → users(user\_id) | User making payment |
| 3 | int(11) | FK → subscriptions(subscription\_id) | Related subscription |
| 4 | decimal(10,2) | NOT NULL | Paid amount |
| 5 | varchar(50) | NOT NULL | Payment method |
| 6 | enum('pending','success','failed') | DEFAULT 'pending' | Status |
| 7 | varchar(100) | NULL | Transaction reference |
| 8 | timestamp | DEFAULT current\_timestamp() | Created timestamp |

**16. Table name: plan\_features**

|  |  |  |  |
| --- | --- | --- | --- |
| Sl. No. | Type | Constraint | Description |
| 1 | int(11) | PK | Feature ID |
| 2 | int(11) | FK → meal\_plans(plan\_id) | Linked meal plan |
| 3 | varchar(255) | NOT NULL | Description of feature |
| 4 | int(11) | DEFAULT 0 | Ordering |

**17. Table name: plan\_images**

|  |  |  |  |
| --- | --- | --- | --- |
| Sl. No. | Type | Constraint | Description |
| 1 | int(11) | PK | Image ID |
| 2 | int(11) | FK → meal\_plans(plan\_id) | Linked meal plan |
| 3 | varchar(255) | NOT NULL | Path to image |
| 4 | int(11) | DEFAULT 0 | Ordering |

**18. Table name: subscription\_meals**

|  |  |  |  |
| --- | --- | --- | --- |
| Sl. No. | Type | Constraint | Description |
| 1 | int(11) | PK | Record ID |
| 2 | int(11) | FK → subscriptions(subscription\_id) | Linked subscription |
| 3 | date | NOT NULL | Date of delivery |
| 4 | enum('breakfast','lunch','dinner') | NOT NULL | Meal type |
| 5 | int(11) | FK → meals(meal\_id) | Actual meal delivered |
| 6 | enum('scheduled','cancelled','delivered') | DEFAULT 'scheduled' | Meal delivery status |

**19. Table name: plan\_meals**

|  |  |  |  |
| --- | --- | --- | --- |
| Sl. No. | Type | Constraint | Description |
| 1 | int(11) | PK | Record ID |
| 2 | int(11) | FK → meal\_plans(plan\_id) | Related plan |
| 3 | enum('MONDAY','TUESDAY','WEDNESDAY','THURSDAY','FRIDAY','SATURDAY','SUNDAY') | NOT NULL | Day of week |
| 4 | enum('Breakfast','Lunch','Dinner') | NOT NULL | Type of meal |
| 5 | int(11) | FK → meals(meal\_id) | Meal linked |

**20. Table name: plan\_schedule\_options**

|  |  |  |  |
| --- | --- | --- | --- |
| Sl. No. | Type | Constraint | Description |
| 1 | int(11) | PK | Unique option ID |
| 2 | int(11) | FK → meal\_plans(plan\_id) | Associated meal plan |
| 3 | enum('weekday','extended','full\_week') | NOT NULL | Schedule choice (Mon–Fri, Mon–Sat, Mon–Sun) |
| 4 | decimal(5,2) | DEFAULT 1.00 | Price multiplier applied for this schedule |
| 5 | varchar(255) | NULL | Explanation of the option |

**21. Table name:** **popular\_meals**

|  |  |  |  |
| --- | --- | --- | --- |
| Sl. No. | Type | Constraint | Description |
| 1 | int(11) | PK | Record ID |
| 2 | int(11) | FK → meals(meal\_id) | Linked meal |
| 3 | int(11) | DEFAULT 0 | Score/rank based on orders/feedback |
| 4 | timestamp | DEFAULT current\_timestamp() | Added timestamp |

**22. Table name: subscriptions**

|  |  |  |  |
| --- | --- | --- | --- |
| Sl. No. | Type | Constraint | Description |
| 1 | int(11) | PK | Subscription ID |
| 2 | int(11) | FK → users(user\_id) | User who subscribed |
| 3 | int(11) | FK → meal\_plans(plan\_id) | Subscribed meal plan |
| 4 | date | NOT NULL | Start date |
| 5 | date | NOT NULL | End date |
| 6 | enum('active','paused','cancelled','expired') | DEFAULT 'active' | Subscription state |
| 7 | timestamp | DEFAULT current\_timestamp() | Created timestamp |
| 8 | timestamp | DEFAULT current\_timestamp() ON UPDATE | Last updated |

# CODING

## 5.CODING

### 5.1 INTRODUCTION

Coding section is where the magic happens. All the planning and the designing done in the previous sections come to life in this section. After this section can only the programmer enjoy the result of his/her hard work when he/she runs the program for the first time.

### 5.2 SELECTION OF SOFTWARE

**PHP**

PHP, an acronym for Hypertext Preprocessor, is a versatile server-side scripting language that falls under the broader category of software development. It is widely recognized for its pivotal role in web development and boasts several essential features that make it a preferred choice for building dynamic websites and web applications. Here are some of its key features:

* Open Source
* Database Integration
* Embedded in HTML
* Cross-Platform Compatibility
* Security

**MYSQL**

MySQL is an open-source relational database system, widely used for web development task like data storage, manipulation, and retrieval. It seamlessly integrates into web applications, eliminating the need for complex setup. MySQL is embedded within web development environments, making administrative tasks effortless. It operates as an SQL-based database, storing data in text files on the device. Unlike systems like JDBC, MySQL simplifies data access with its broad range of relational database features. Its features are

* Zero configuration
* Server less
* Stable cross platform database file
* Less memory
* Self-contained
* Transactional

### 5.3 CODING PHASE

The goal of the 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 coding phase does not affect the structure of the system; it has great impact on the internal structure of modules, which affects the testability of the system. The goal of the coding phase is to produce clear simple programs. The aim is not to reduce the coding effect, but to program in a manner so that testing and maintenance costs are reduced.

Programs should not be constructed so that they are easy to write; they should be easy to read and understand. Reading programs is a much more common activity than writing programs. Hence, the goal of the coding phase is to produce simple programs that are clear to understand and modify.

#### 5.3.1 CODING STANDARDS

The standard used in the development of the system is Microsoft Programming standards. It includes naming conversations of variables, constants and objects, standardized formats for labelling and commenting code, spacing, formatting and indenting.

**Naming Conventions**

The controls are prefixed to indicate their functions. The frames are prefixed with frm, textboxes are prefixed with txt, command buttons with cmd, label boxes with lbl, list boxes with lst, comboboxes with cmb, Date Time Pickers with DTP, Grid with grid and so on.

**Labels and Comments**

The functions of each control are labelled clearly in the GUI. The code also includes comments so that other developers using the source code in future might understand the module functions better.

**TESTING**

**&**

# IMPLEMENTATION

## 6. TESTING

### 6.1 INTRODUCTION

Software testing is a critical element of software quality assurance and represents the ultimate review of specifications design and coding. Testing presents an interesting anomaly for the software. Testing is a quality measure process, which reveals the errors in the program. During testing, the program 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 it is expected. Testing plays a very critical role in determining the reliability and efficiency of the software and it is a very important stage in software development.

### 6.2 TESTING

System testing is actually a series of different tests whose primary purpose is to fully exercise the computer-based systems. Although each test has a different purpose, all work to verify that all system elements have been properly integrated and perform allocated functions.

System testing is done in order to ensure that the system developed doesn’t fail at any point. Before implementations, the system is tested with experimental data to ensure that it will meets the specified requirements, special tests data are input for processing and results examined.

#### 6.2.1 TEST PLAN

**Preparation of test data**

Taking various kinds of test data does the testing. 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 correction are also noted for future use. Two kinds of test data were collected and used:

**Using live test data**

Live test is those that are actually extracted from organization files. After a system is partially constructed, programmers or analyst often ask customers to key in a set of data fom their normal activities. Then, the system person uses this data as a way to partially test the system. In order instance, programmers or analysts extract a set of live data from the files and have entered themselves.

**Using artificial test data**

Artificial test data are created solely for test purpose, since they can be generated to test all combinations of formats and values. In other words, the artificial data, which can quickly be prepared by a data generating utility program in the information system 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 those who wrote the program.

In this project invalid data was entered to test whether the program would break or not. These invalid data entries were randomly generated using random people. Many people were given the software for testing the program. They use gibberish values to test if every validation holds strong.

### 6.3 TESTING METHODS

Testing is generally done at two levels-testing of individual modules and testing of the entire system. During system testing, the system is used experimentally to ensure that the software does not fail that is, that it will run according to its specifications and the results examined. A limited number of uses may be allowed to use the system so analysis can see whether they use it in unforeseen ways. It is preferable to discover any surprise before the organization implements the system and depends on it.

Testing is done throughout system development at various stages. It is always a good practice to test the system at many different levels at various intervals, that is, sub systems, program modules as work progresses and finally the system as a whole. During testing the major activities are concentrated on the examination ad modification of the source code. Usually, this testing is to be performed by the person other than the person who has really coded it. This is done in order to ensure more complete and unbiased testing for making the software more reliable.

There are two types of testing:

* Black box testing
* White box testing

#### 6.3.1 WHITE BOX TESTING

In white box testing, the internal logic of the modules is considered. Following levels of testing are performed for the developed project:

**6.3.1.1 Unit Testing**

This involves the tests carried out on modules programs, which make up a system. This is also called as a program testing. The units in a large system many modules at different levels are needed. Unit testing focuses on the modules, independently of one another, to locate errors. The program should be tested for correctness of logic applied and should detect errors in coding.

Before proceeding one must make sure that all the programs are working independently.

#### 6.3.2 BLACK BOX TESTING

The concept of the black box is used to represent a system who’s inside workings are not available for inspection. In a black box, the test item is treated as “black”, since its logic is unknown; all that is known is what goes in and what comes out, or the input and output.

**6.3.2.1 System Testing**

The system testing is conducted on a complete, integrated system to evaluate the system’s compliance with its specified requirement. It falls within scope of black box testing so no knowledge of inner design or logic is needed. As a rule, system testing takes, as its input, all of the integrated software components that have passed integration testing and also the software system itself integrated with any applicable hardware system. The purpose of the integration testing is to detect any inconsistencies between software units.

System testing is the stage of implementation, which is aimed at ensuring that the system works accurately and efficiently before live operation commence. The logical design and the physical design should be thoroughly and continually examined on paper ensure that they will work when implemented.

**6.3.2.2 Integration Testing**

Integration testing is a systematic technique for constructing the program structure, while at the same time conducting tests to uncover errors associated with interfacing. This is the program is constructed and tested in small segments, which makes it easier to isolate and the following common types of integration problems may be observed:

* Version mistakes
* Data integrity violations
* Overlapping function
* Resource problems especially in memory handling
* Wrong type of parameter in function calls

**6.3.2.3 Validation Testing**

At the culmination of the integration testing, the software was completely assembled as a package, interfacing errors have been uncovered and corrected and a final series of software validation testing began.

In validation testing we test the system functions in a manner that can be reasonably expected by customer, the system was tested against system requirement specification. Different unusual inputs that the customers may use were assumed and the outputs were verified for such unprecedented inputs. Deviation or errors discovered at this step are corrected prior to the completion of this project with the help of user by negotiating to establish a method for resolving deficiencies.

Thus, the proposed system under consideration has been tested by using validation testing and found to be working satisfactorily. Validation checking is performed on the: -

**Numeric Field:** - The numeric field can contain only numbers from 0 to 9. An entry of any character flashes an error message. The individual modules are checked for accuracy and what it has to perform. Each module is subjected to test run along with sample data. The individually tested module are integrated into a single system.

**Character Field:** - This field can only contain letters from A-Z and a-z. It is useful for name, address fields and so on.

**Check Null Fields:** - Before entering values into the database or when updating, a validation is done to check whether any NULL fields are present.

**Email Fields:** - A email only field with a limit of characters. All the necessary validation checks were verified to see if invalid data ever enters into the database. Null values in fields were also treated as invalid values.

**Password Fields:** - - A password only field with a limit of characters. All the necessary validation checks were verified to see if invalid data ever enters into the database. Null values in fields were also treated as invalid values.

#### 6.3.3 OUTPUT TESTING

After performing validation test, the next phase is output test of the system, since no system could be useful if it does not produce the desired output in the desired format. By consideration the format of the report/output was generated or displayed and was tested. Here output format was considered in one way: on the display screen.

#### 6.3.4 USER ACCEPTANCE TESTING

User acceptance test of a system is the key factor for the success of the system. The system under consideration was listed for user acceptance by keeping constant touch with the perspective user of the system at the time of design, development and making changes whenever required. This was done with the regards of the following points: -

* Input screen design
* Output design

Customers from each of the 2 user types (Admin, user) were selected for user acceptance testing. The Admin was given the software for testing with his username and password. The admin actions are performed and see whether all details are entering into the database and working properly as expected... The customers side is tested using a customer name and password by registering to the system and see he can post property from the website.

### 6.4 IMPLEMENTATION

Implementation is the stage of the project when the theoretical design is turned into a working system. The implementation stage is a system project in its own right. It includes careful planning, investigation of current system and its constraints on implementation, design of methods to achieve the changeover, training of the staff in the changeover procedure and evaluation of the changeover method.

The first task in implementation is planning deciding on the methods and time scale to be adopted. Once the planning has been completed the major effort is to ensure that the programs in the system are working properly when the user has been trained.

The complete system involving both computer and user can be executed effectively. Thus, the clear plans are prepared for the activities.

Successful implementation of the new system design is a critical phase in the system life cycle. Implementation means the process of converting a new or a revised system design into an operational one.

**MAINTENANCE**

**&**

# ENHANCEMENT

## 7. MAINTENANCE AND ENHANCEMENT

### 7.1 MAINTENANCE

This software can be modified as need occurs. Maintenance includes all the activities after installation of the software that is performed to keep the system operational. The process of maintenance involves:

* Understanding the existing software
* Understand the effect of change
* Test for satisfaction

This software requires little to no maintenance. During the testing phase most maintenance duties are performed. If a maintenance requirement occurs, it can be solved with ease

### 7.2 ENHANCEMENT

The Celestia website is built with a modular architecture, allowing for easy expansion and additional functionalities. As the business grows and customer demands evolve, the platform can seamlessly integrate new features to enhance the user experience.

Future enhancements to the Celestia website could include:

* **Virtual Design Consultation**: Integrate virtual design consultations using video conferencing tools, allowing clients to discuss their requirements with designers directly from the platform.
* **3D Landscape Visualization**: Offer customers 3D landscape modeling for garden and interior designs, providing a detailed view of how the project will look before implementation.
* **Booking Reminder Alerts**: Implement SMS or email alerts for upcoming bookings, along with calendar integrations for scheduled appointments, helping customers stay informed.
* **Advanced Customer Profiles**: Implement a profile system where customers can track their service history, view consultations, and manage upcoming bookings all from a personalized dashboard.
* **AR Home Design**: Implement Augmented Reality (AR) functionality that allows customers to visualize interior design changes or outdoor projects directly on their property using their mobile device camera.

These future developments will help enhance Celestia’s service offerings, attract new customers, and provide a seamless, user-friendly experience that stays competitive as technology advances.

# CONCLUSION

## 8. CONCLUSION

In today’s dynamic design and landscaping industry, technology plays a pivotal role in shaping customer experiences and elevating service delivery. Traditional methods of offering landscaping, pool building, and interior design services are being transformed by innovative digital solutions. The Celestia website exemplifies this evolution, providing a seamless bridge between customers and service providers. With a user-friendly, interactive platform, Celestia harnesses modern web technologies to allow customers to explore, book, and engage with services from the comfort of their homes, eliminating the barriers of traditional consultations.

The primary goal of Celestia has always been to empower customers with easy access to high quality services in garden development, pool building, interior design, and more. Celestia streamlines service inquiries, enhances transparency, and fosters trust within the service community, setting new standards for customer interaction in the design and landscaping sectors.

As the industry continues to evolve, Celestia remains committed to further enhancements, embracing new technologies, and meeting the growing needs of its customers. Looking ahead, Celestia aims to continue revolutionizing service delivery by integrating technology and design services to create even more seamless and impactful customer experiences.

# BIBLIOGRAPHY

**BIBLIOGRAPHY**

## WEBSITES

* Chatgpt
* www.wikipedia.org
* [www.tutorialpoint.com](http://www.tutorialpoint.com/)
* [www.pinterest.com](http://www.pinterest.com/)
* [www.pexels.com](http://www.pexels.com/)
* www.w3schools.com

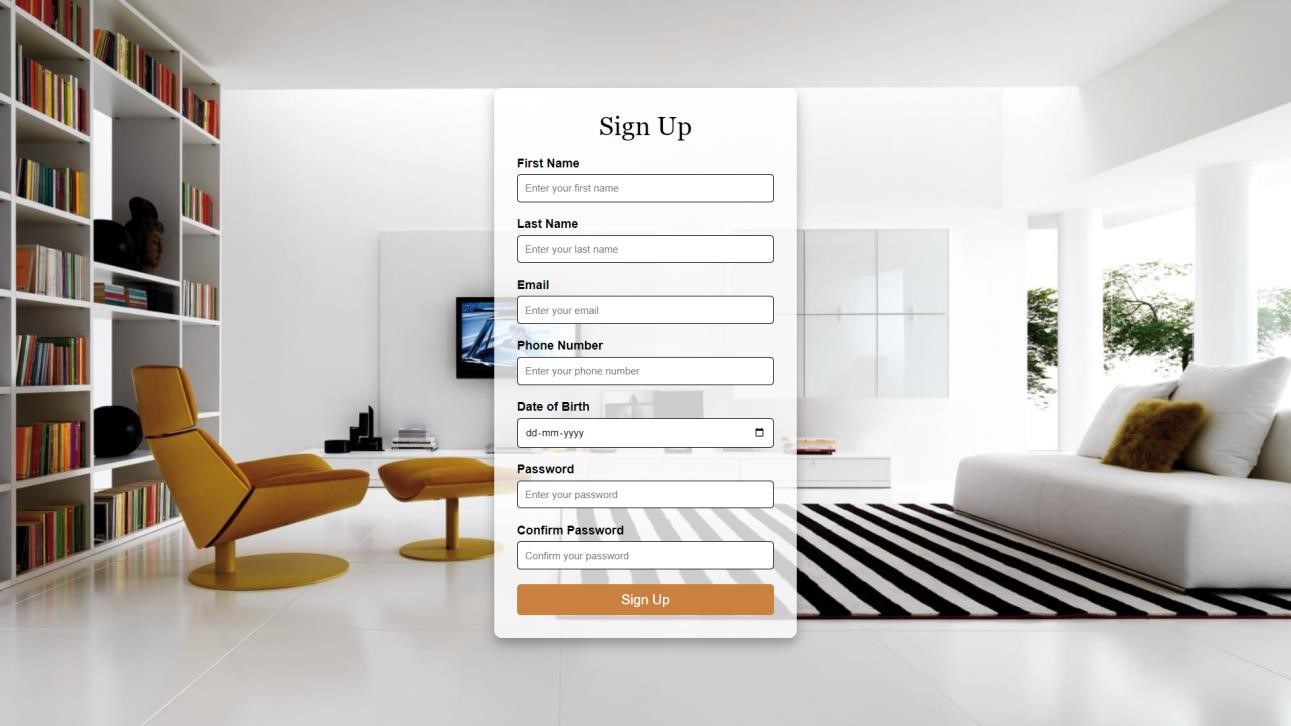
# APPENDIX

## SCREENSHOTS

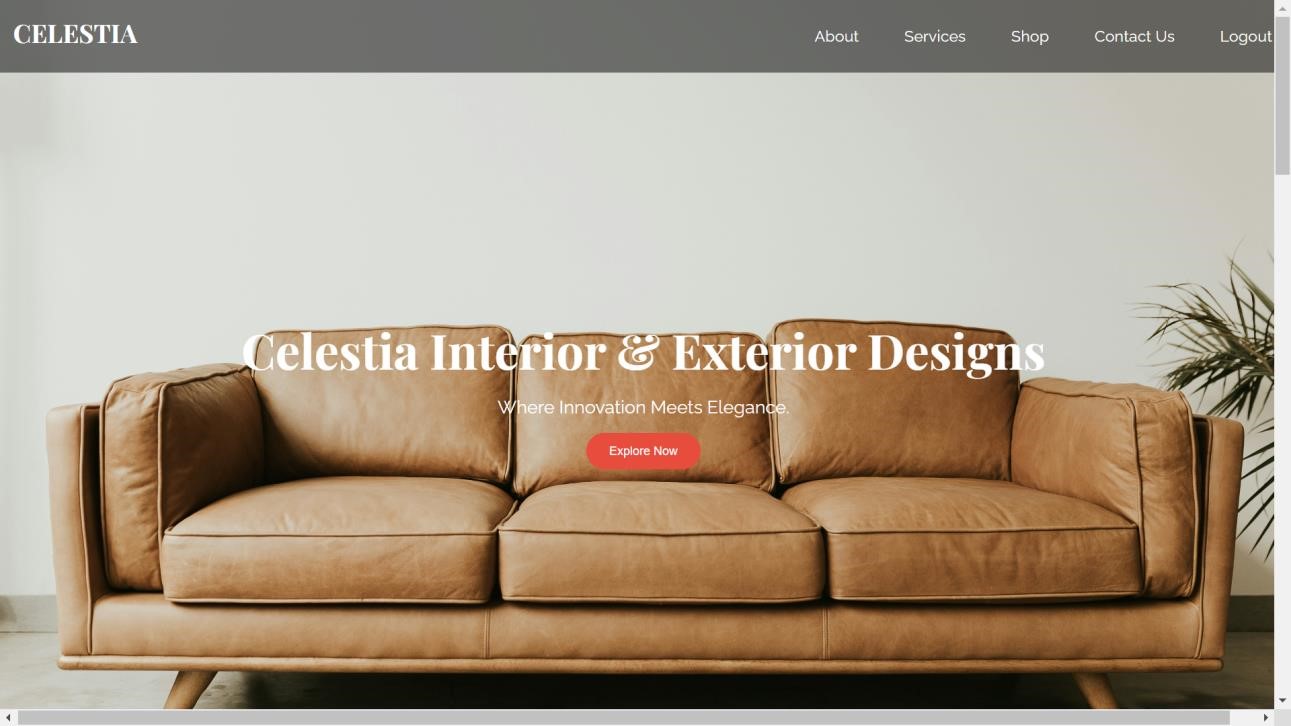
1.LOGIN PAGE

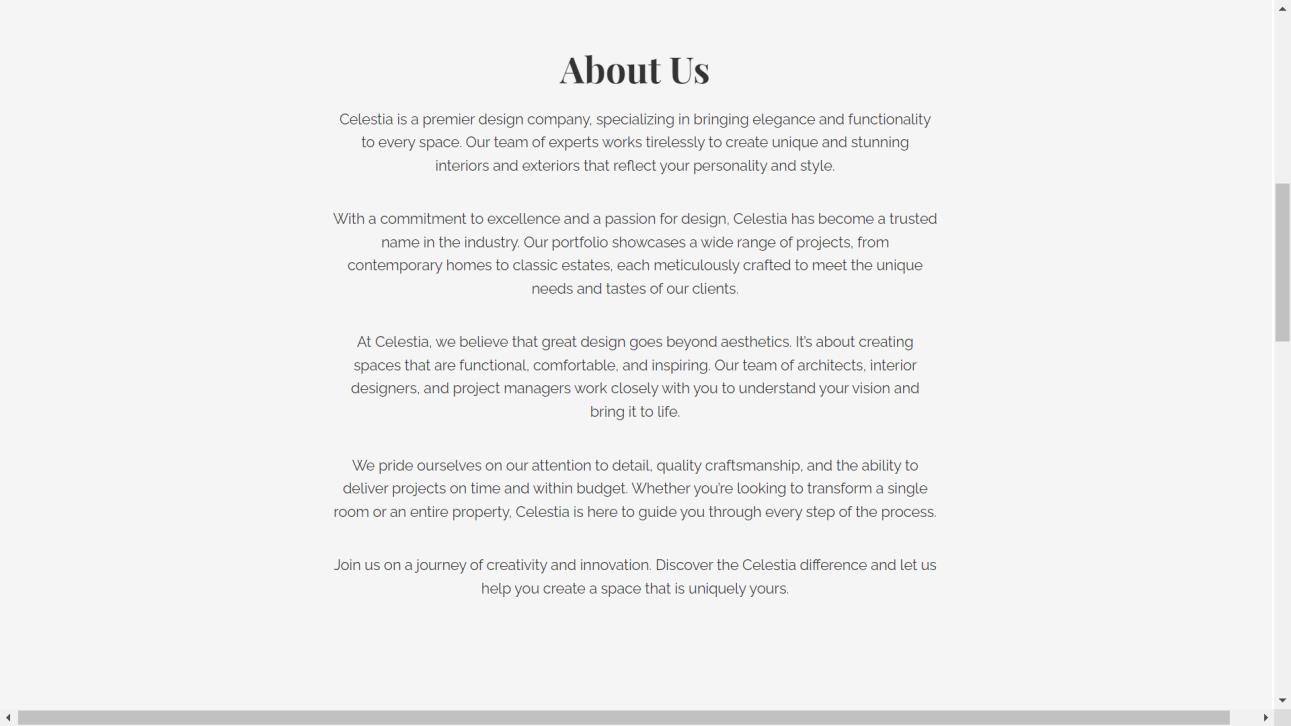


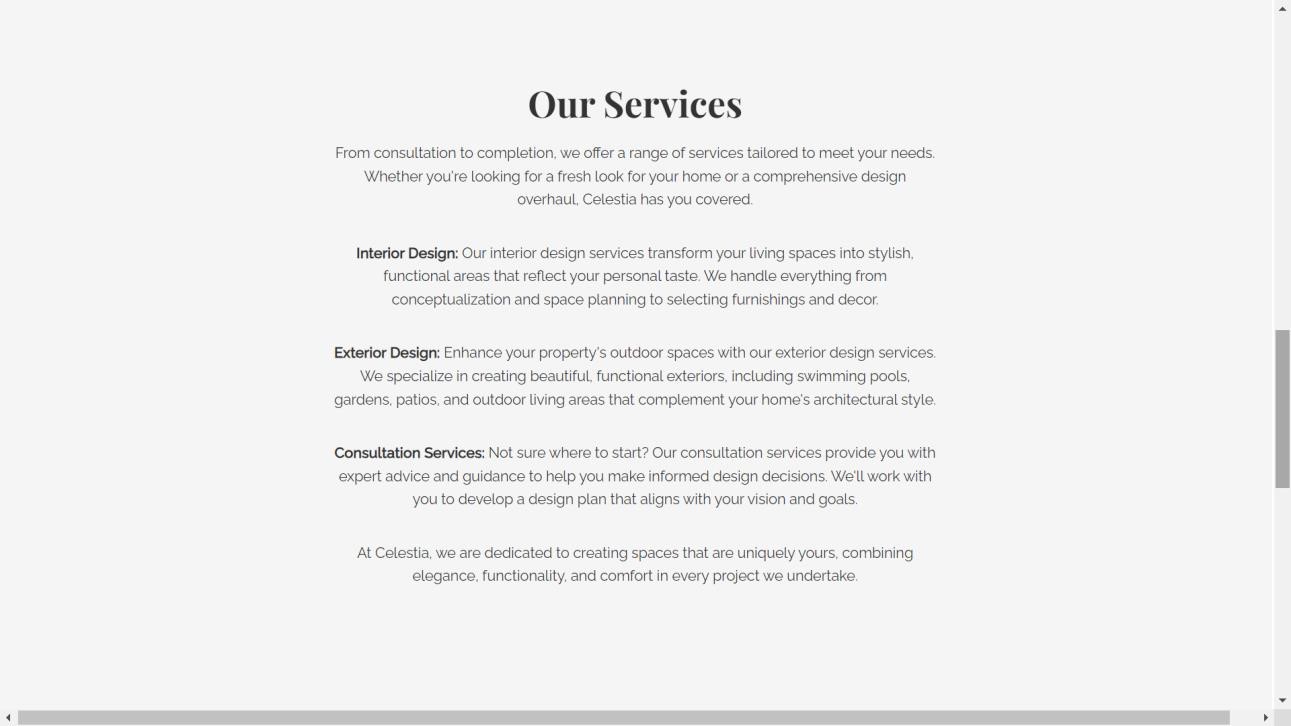
2.REGISTRATION PAGE

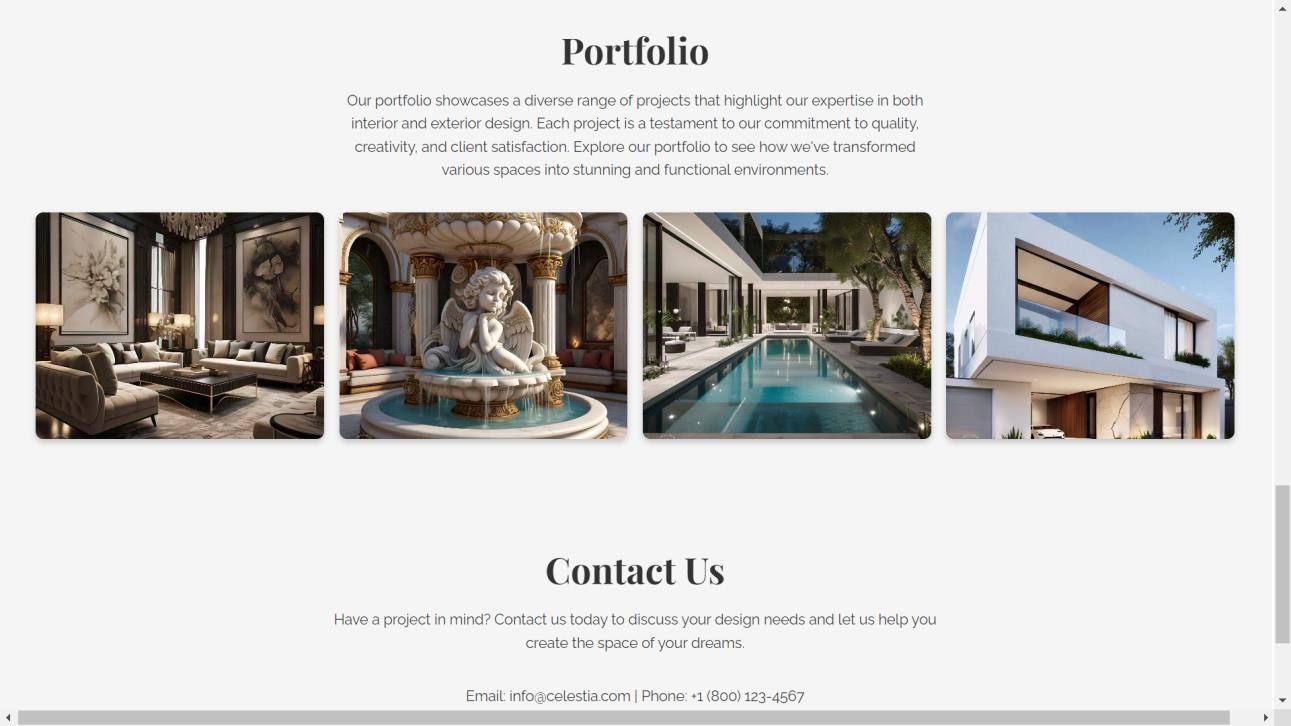


3.HOME PAGE

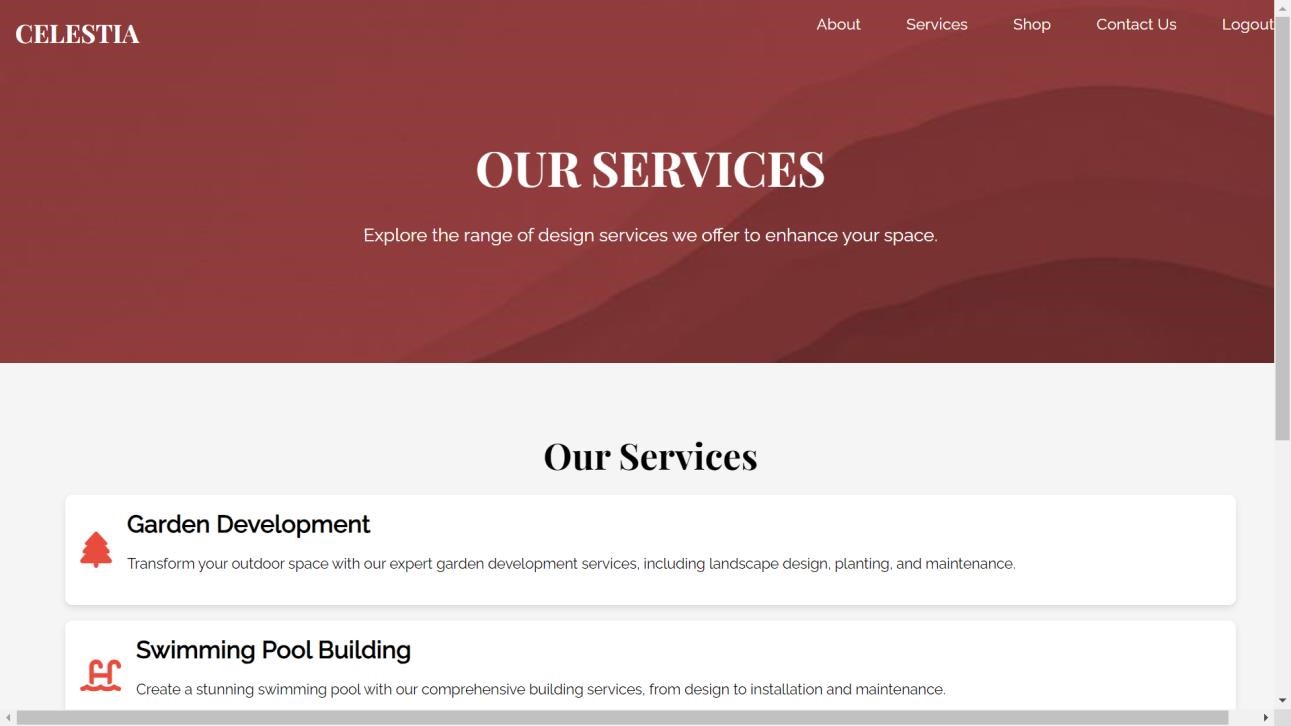


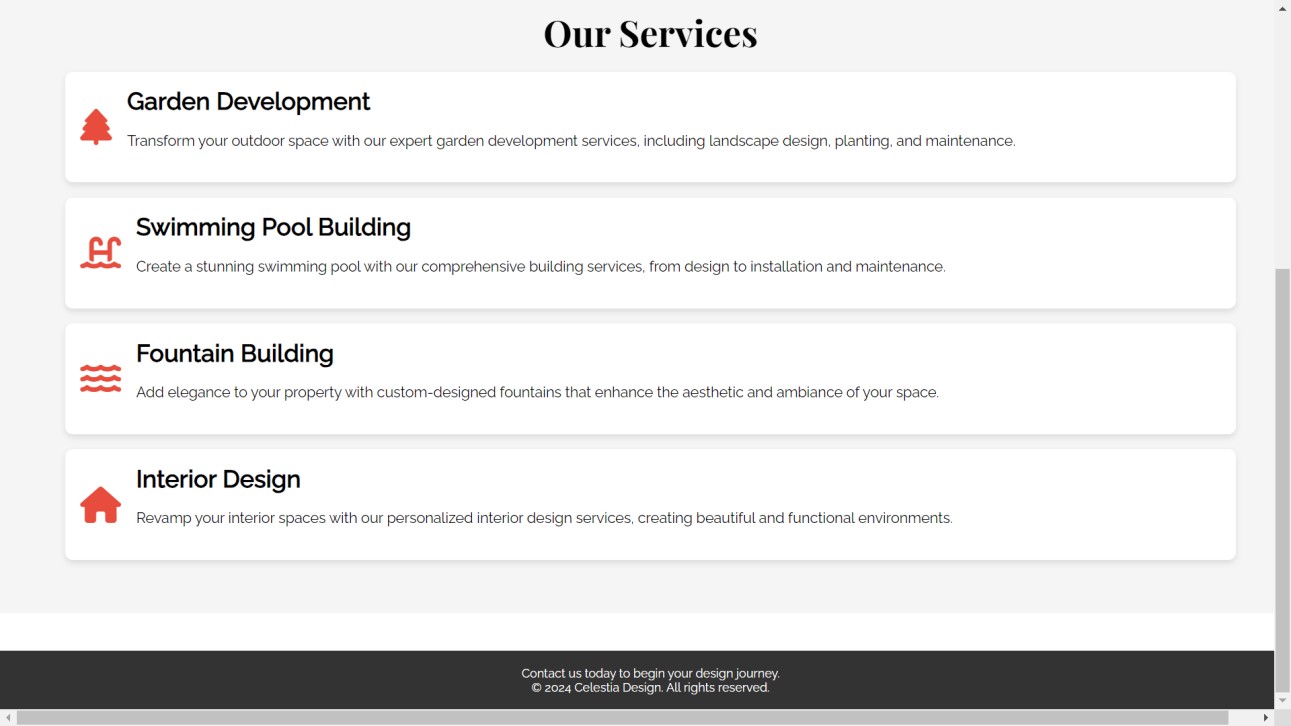




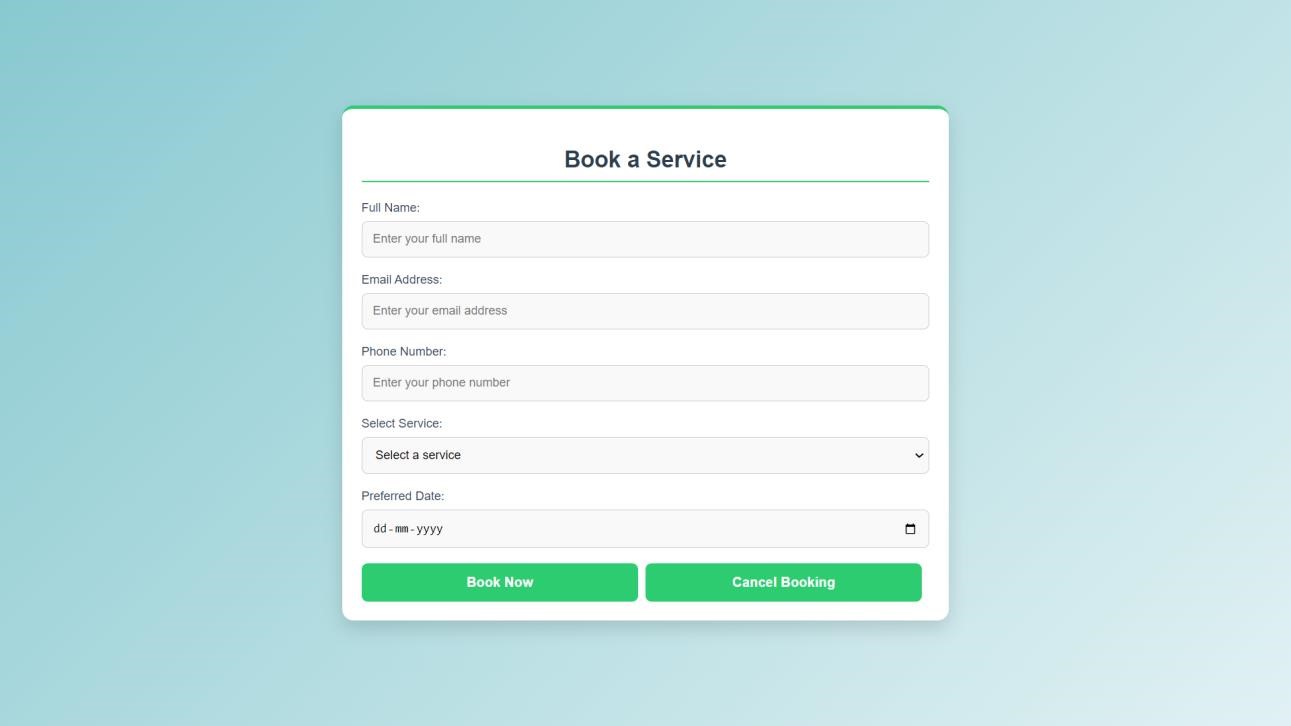


4.SERVICES PAGE

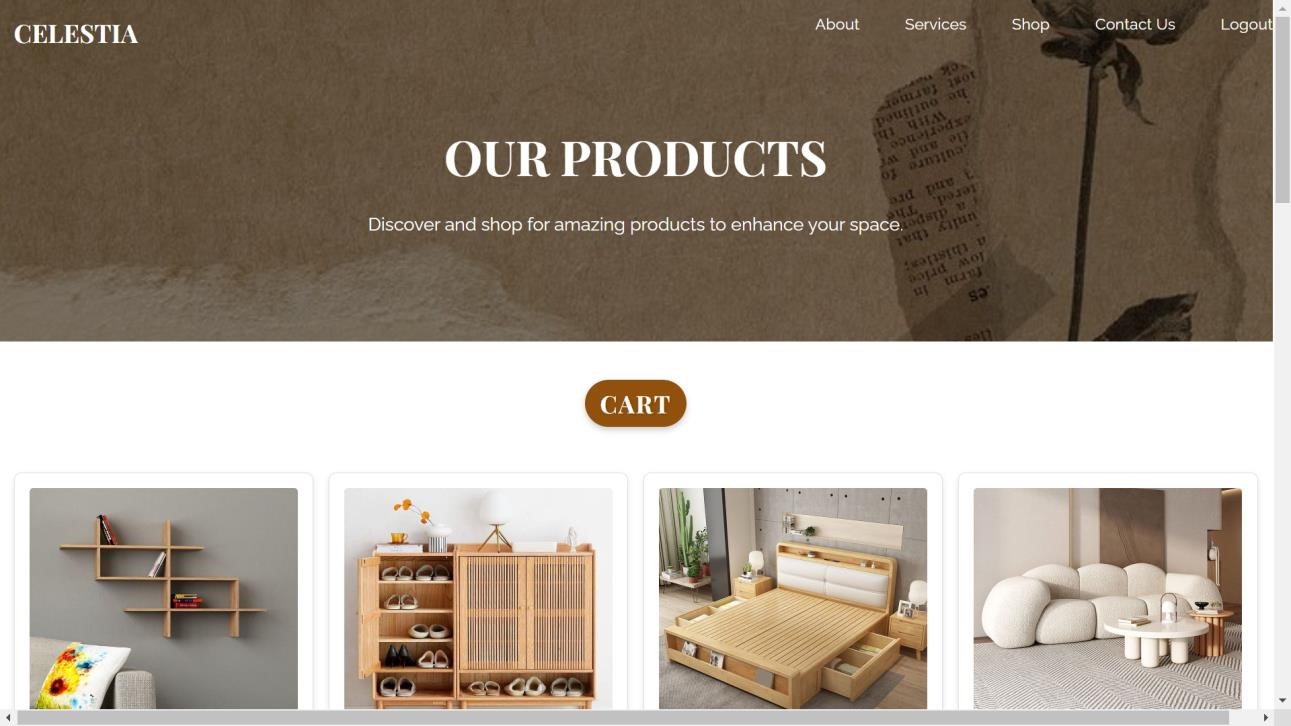


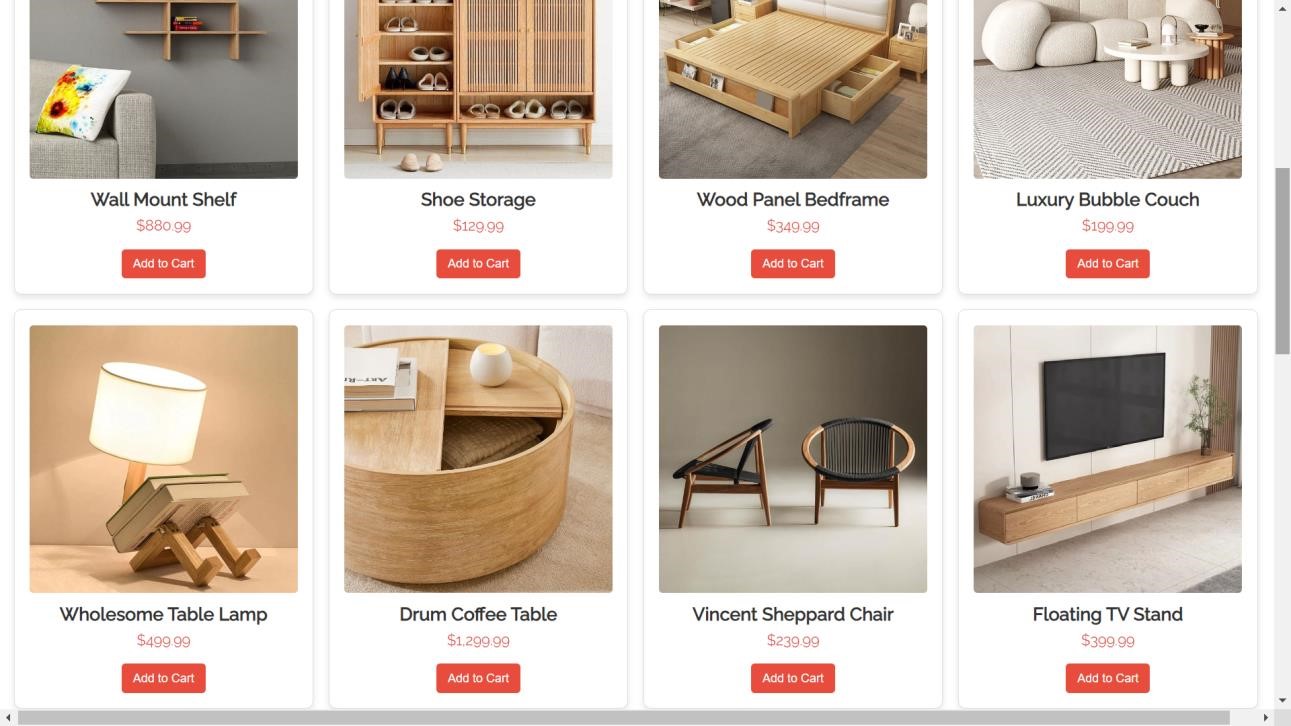


5.SERVICE BOOKING PAGE

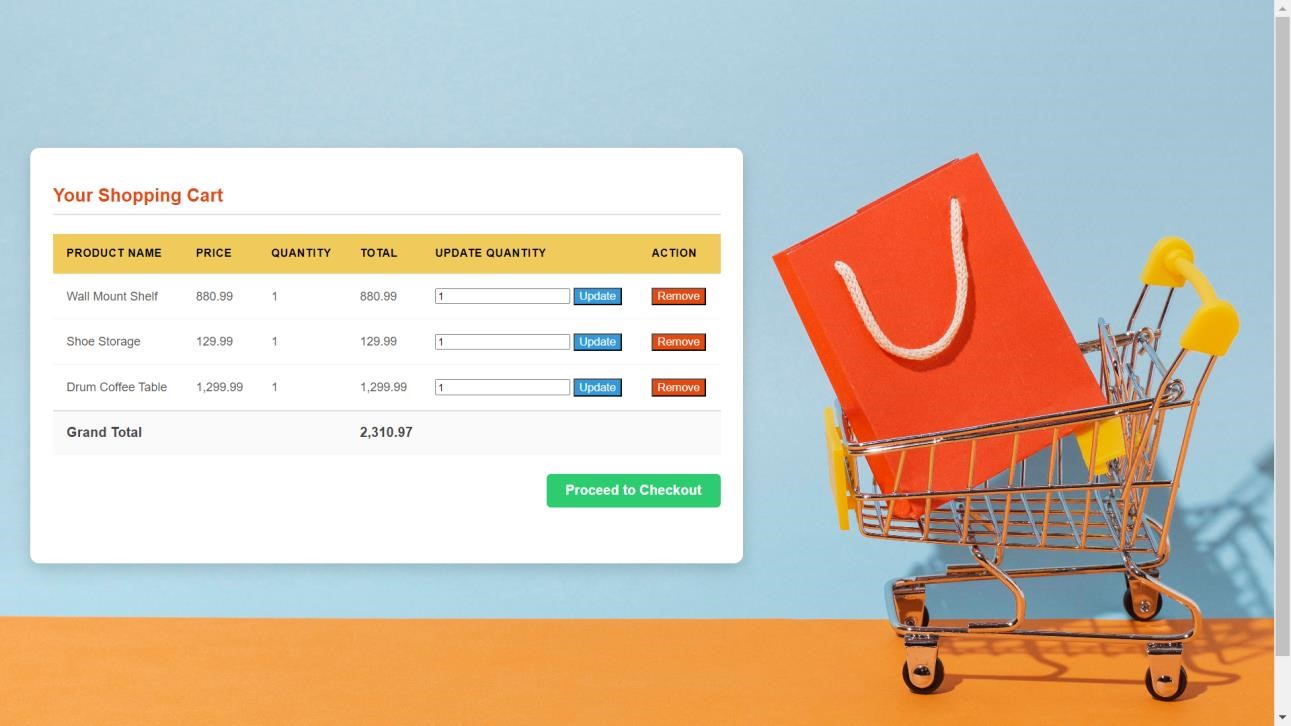


6.SHOP PAGE

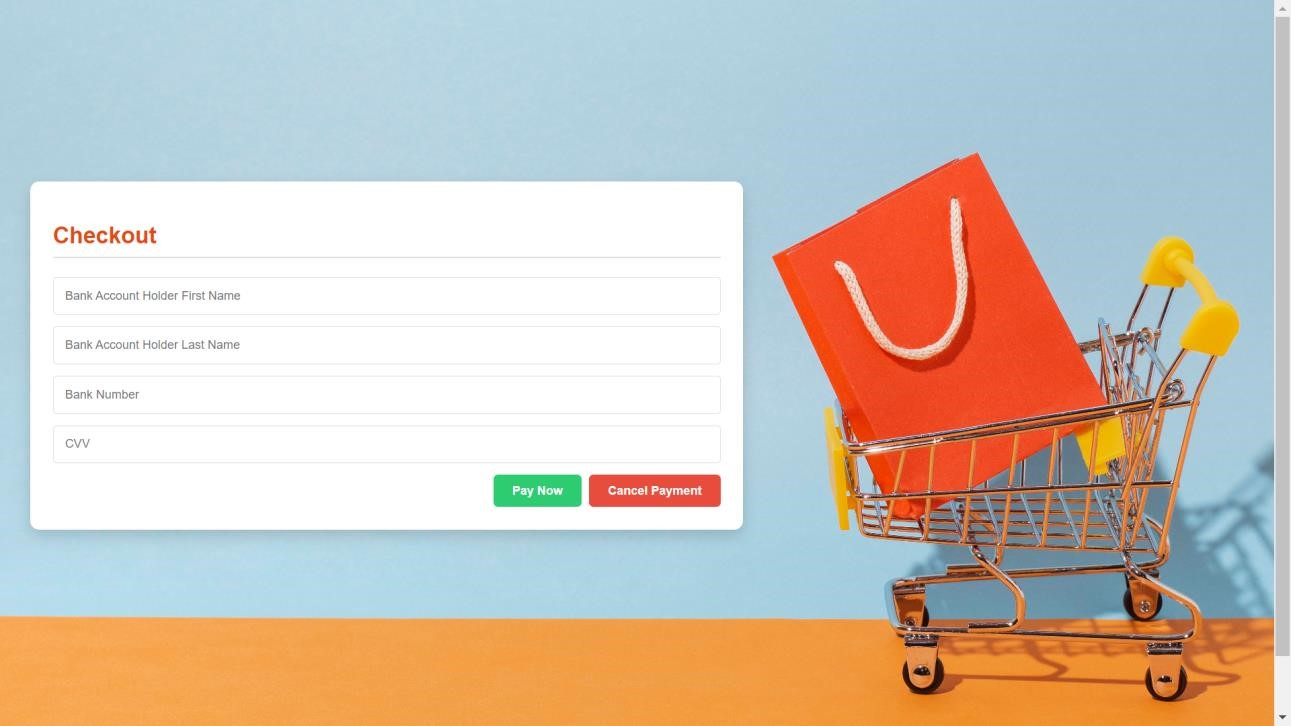




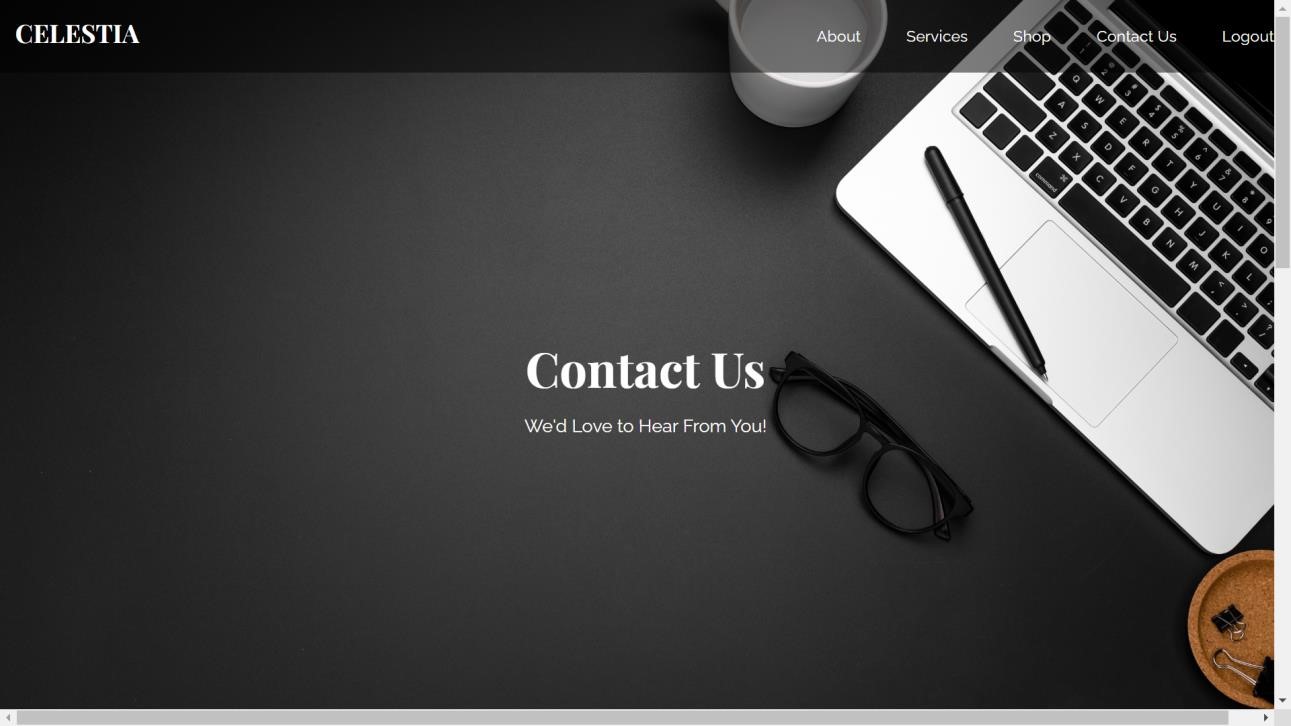
6.CART PAGE

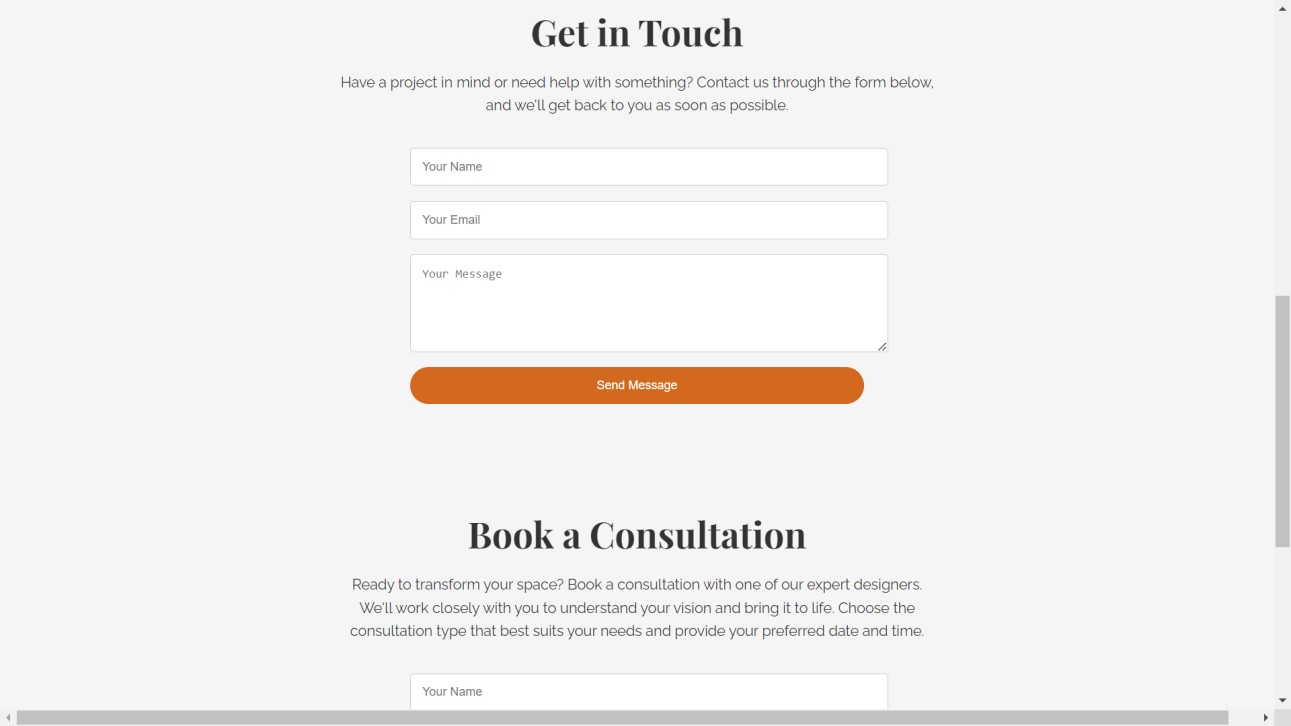


7.PAYMENT PAGE



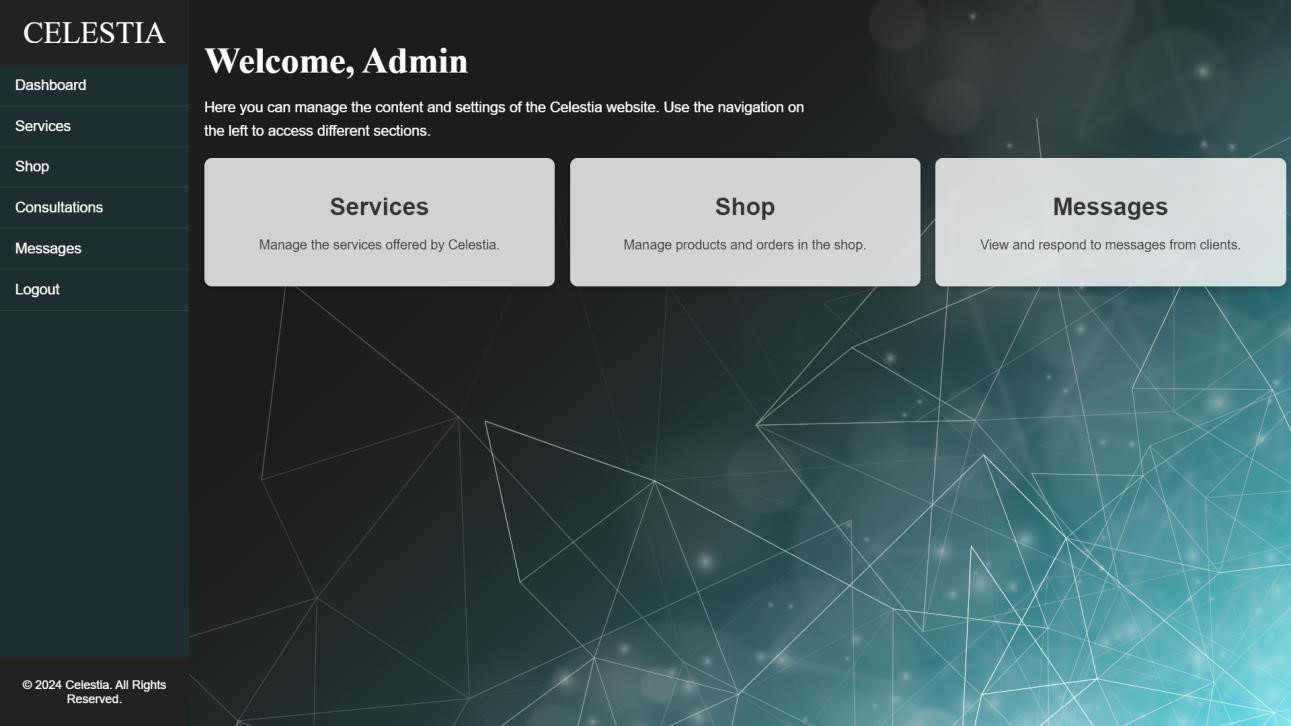
8.CONTACT PAGE



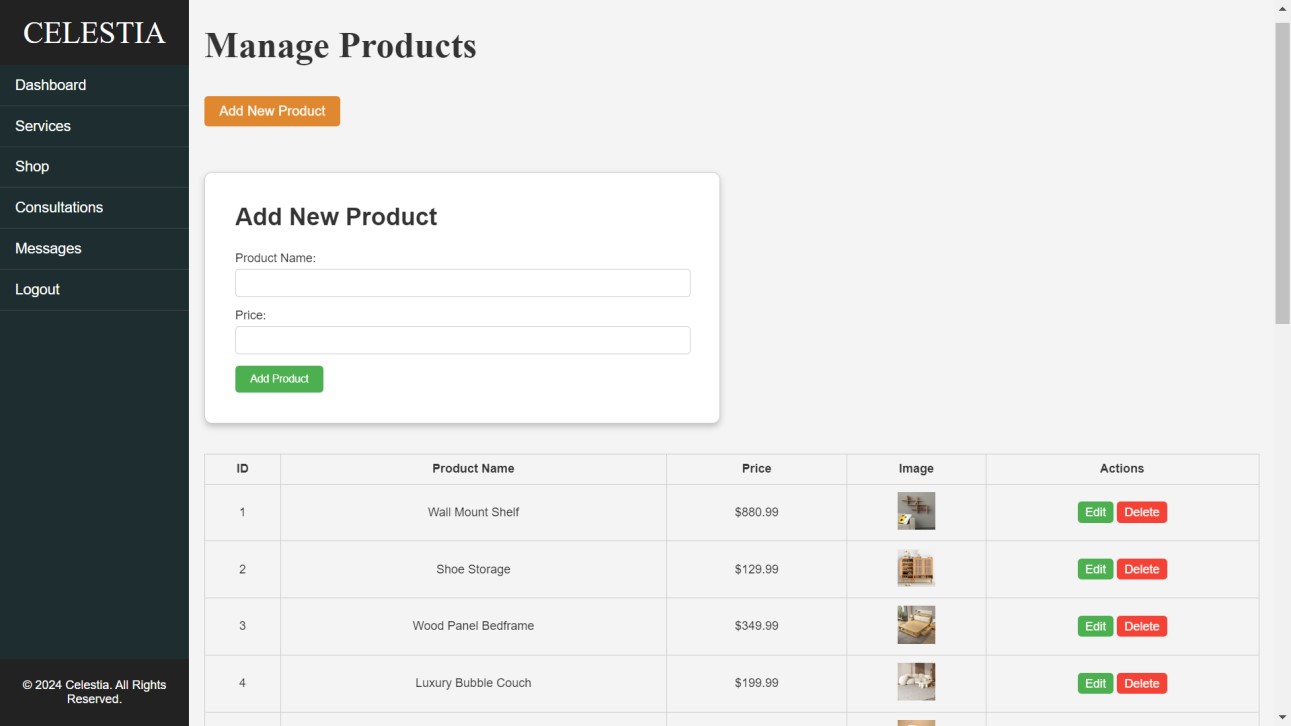


## ADMIN MODULE

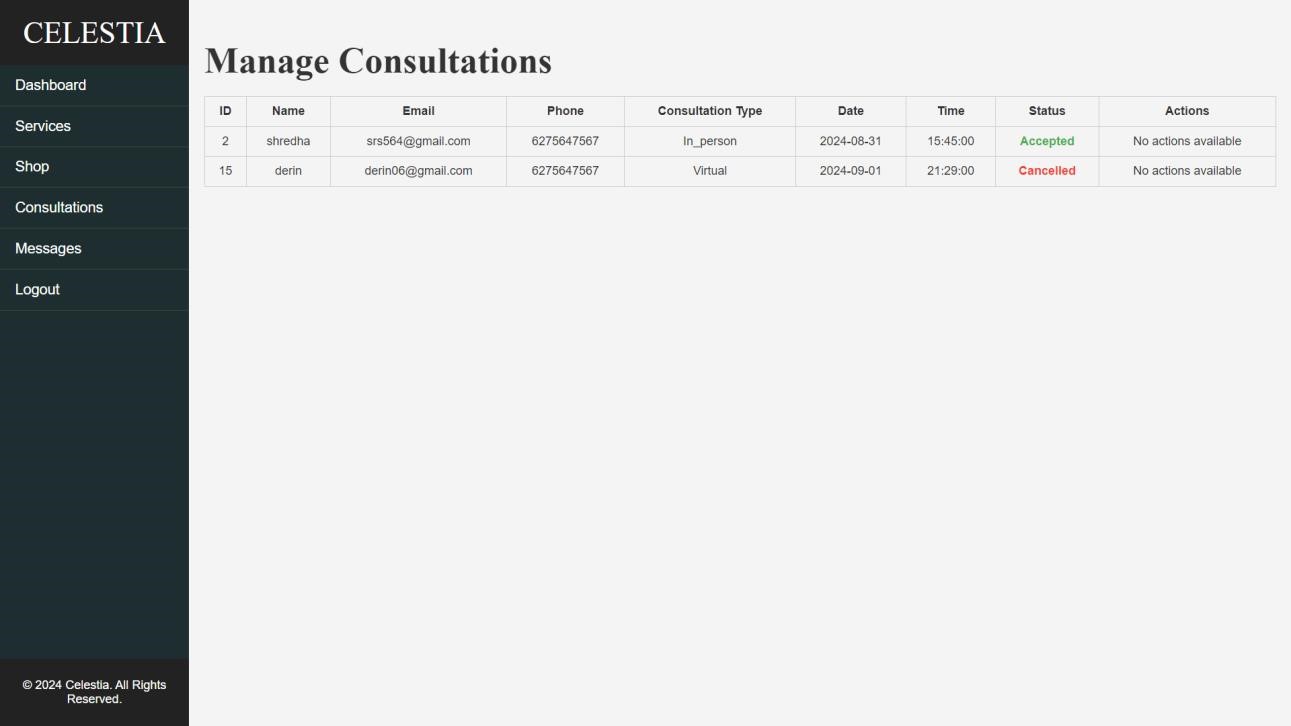
1.DASHBOARD PAGE



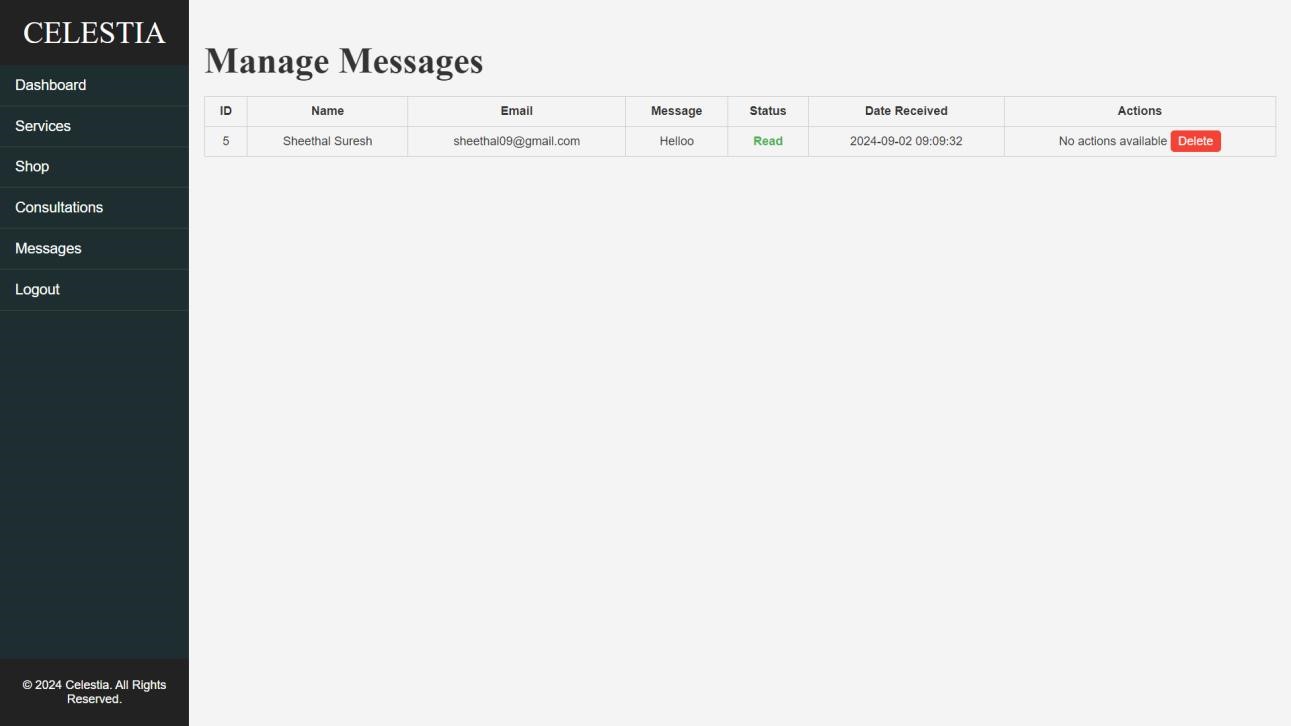
2.MANAGE SHOP PAGE



3.MANAGE CONSULTATIONS PAGE



4.MANAGE MESSAGES PAGE



5.MANAGE SERVICES PAGE

