

**A Mini Project Report On**

## “TEMPLE SEVA MANAGEMENT SYSTEM”

**Submitted in partial fulfillment for the academic year 2024-25**

#### Bachelor of Engineering in

**Computer Science and Engineering**

***Submitted by,***

##### ANIL GOWDA S G 22CSE012

***Under the guidance of:***

**Mrs. SAHANA D GOWDA**

**Asst. Professor, Dept. of CSE BGSIT, B G Nagara**

**** 

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING FACULTY OF ENGINEERING, MANAGEMENT & TECHNOLOGY B G S INSTITUTE OF TECHNOLOGY**

**BG NAGARA - 571448 2024-25**



## 

## CERTIFICATE

This is to certify that the Mini Project report entitled **“TEMPLE SEVA MANAGEMENT SYSTEM”** has been successfully carried out by **Mr. ANIL GOWDA S G**, bearing **USN:22CSE012** a bonafide student of BGS Institute of Technology, B.G Nagara in partial fulfillment of requirements of degree of Bachelor of Engineering in Computer Science & Engineering of Adichunchanagiri University, B.G Nagara during the year of 2024-25. It is certified that all corrections/suggestions indicated for the internal assessment have been incorporated in the report deposited in the department library.

|  |  |
| --- | --- |
| **Signature of Guide:** | **Signature of HOD:** |
| **Mrs. Sahana D Gowda** | **Dr. Shashikala S V** |
| Asst. Professor, Dept., of CSE | Professor & HOD, Dept., of CSE |
| BGSIT, B G Nagara | BGSIT, B G Nagara |

**External Viva**

**Name of the Examiners Signature with date**

1.

2.

# ACKNOWLEDGEMENT

I sincerely convey my regards and thanks to **Dr. B N Shobha, Dean & Principal, Faculty of Engineering, Management & Technology, BGS Institute of Technology, BG Nagara, Mandya,** for giving me a chance to carry out and present my mini project work.

My sincere thanks to **Dr. Shashikala S V**, **Prof. and Head of Department, CS&E, BGSIT, B G Nagara, Mandya,** for giving me a chance to carry out and present my project work with all the support and facilities.

I would like to thank **Mrs. Sahana D Gowda, Assistant Professor, Department of CS&E, BGSIT, BG Nagara** our honorable guide who stood as an excellent guide to carry out my work has been always available as an expressive evaluator for the creation and correction of the report towards our work. They have taken pain and time to go through our work when needed.

My heartfelt gratitude to all the teaching and non-teaching faculties of **Computer Science and Engineering Department, BGSIT, BG Nagara, Mandya,** for their support and guidance towards the completion of our mini project work.

Finally, I would also extend our heartfelt thanks to our family members, classmates, friends and well-wishers for their support and encouragement throughout this effort.

**ANIL GOWDA S G 22CSE012**

# ABSTRACT

The Temple Seva Management System is designed to manage temple resources, services, and events using a database. It includes modules for marriage halls, special events, priest details, and sevas (services). The marriage hall module helps manage capacity, amenities, and bookings. The special events module tracks event details like name, description, venue, and time. Priests are managed with their contact details, expertise, and salary, and are linked to ceremonies they perform. The seva module records services, costs, and associated priests. Managers oversee all operations with secure login credentials. The system ensures efficient management and easy retrieval of information with a user-friendly interface.

|  |  |
| --- | --- |
| **Title** | **Page No** |
| **ACKNOWLEDGEMENT** | **i** |
| **ABSTRACT** | **ii** |
| **TABLE OF CONTENTS** | **iii** |
| **LIST OF FIGURES** | **iv** |
| **CHAPTER 1: INTRODUCTION** | **1-3** |
| 1.1 Overview | 1 |
| 1.2 Problem Statement | 1 |
| 1.3 Objectives | 2 |
| 1.4 Existing System | 2 |
| 1.4.1 Disadvantages | 2 |
| 1.5 Proposed System | 3 |
| 1.4.1 Advantages | 3 |
| **CHAPTER 2: LITERATURE SURVEY** | **4-5** |
| **CHAPTER 3: SYSTEM REQUIREMENTS** | **6** |
| 3.1 Hardware Requirements | 6 |
| 3.2 Software Requirements | 6 |
| **CHAPTER 4: SYSTEM ARCHITECTURE** | **7-9** |
| **CHAPTER 5: IMPLEMENTATION** | **10-47** |
| 5.1 Source Code | 10 |
| 5.2 Entity Relationship (ER) Diagram | 44 |
| 5.3 Schema Diagram | 47 |
| **CHAPTER 6: RESULT AND SNAPSHOT** | **48-51** |
| **CONCLUSION** | **52** |
| **FUTURE ENHANCEMENT** | **53** |
| **REFERENCES** | **54** |

|  |  |  |
| --- | --- | --- |
| **Figure No.** | **Figure Name** | **Page No.** |
| 5.2.1 | Entity Relationship Diagram | 44 |
| 5.3.1 | Schema Diagram | 47 |
| 6.1 | Home Page | 48 |
| 6.2 | Manager Sign Up | 48 |
| 6.3 | Login Page | 49 |
| 6.4 | Menu Page | 49 |
| 6.5 | Marriage Hall Details | 50 |
| 6.6 | Priests Details | 50 |
| 6.7 | Insert New Seva | 51 |
| 6.8 | Delete Special Event Record | 51 |

## CHAPTER 1

### OVERVIEW

# INTRODUCTION

The Temple Seva Management System is a database-driven solution aimed at streamlining temple operations and addressing the inefficiencies of traditional manual processes. This system integrates key modules to manage marriage hall bookings, special events, priest assignments, and seva (service) offerings. The marriage hall management module records details such as capacity, amenities, and booking schedules, while the special events module captures comprehensive event details like venue, time, and assigned managers. Priest management maintains crucial information, including contact details, expertise, and schedules, ensuring smooth allocation of responsibilities. The seva management module tracks service offerings, costs, and associated priests, enabling devotees to access clear and accurate information. By implementing role-based access control, the system ensures secure and efficient management while maintaining data integrity. Designed for scalability and operational transparency, this system enhances administrative efficiency and provides a user- friendly interface for administrators and devotees alike. The Temple Management System represents a practical application of database concepts to modernize temple workflows and improve overall resource management.

### PROBLEM STATEMENT

Managing temple operations like marriage hall bookings, special events, priest schedules, and seva offerings using traditional methods is often slow, prone to mistakes, and inefficient. These manual processes make it hard to track bookings, manage resources, organize events, and keep accurate records of priests and their duties. Without a centralized system, communication breaks down, data is repeated unnecessarily, and services are delayed. Devotees may struggle to get clear information about available services and events, while temple administrators face challenges in managing data and maintaining transparency. To solve these problems, a database - driven Temple Management System is proposed. This system will bring everything together - marriage hall management, event planning, priest details, and seva tracking - on a single platform. By giving managers secure access and organizing resources in a structured way, the system will save time, reduce errors, and improve the overall experience for both temple staff and devotees.

### OBJECTIVES

The main objective of the Temple Seva Management System is to simplify and improve the management of temple operations using a centralized, database-driven approach. This system aims to organize and streamline key processes like managing marriage halls, planning special events, assigning tasks to priests, and tracking seva (service) offerings. It focuses on reducing manual work, eliminating errors, and improving the efficiency of resource management. The system ensures secure role-based access for managers, protects sensitive information, and enhances transparency in operations. By providing a structured platform for storing and retrieving information, it minimizes data redundancy and ensures accurate record- keeping. Additionally, the system aims to improve the overall experience for both administrators and devotees by making services easily accessible, ensuring timely communication, and supporting scalability to handle the growing needs of modern temples.

### EXISTING SYSTEM

The current system for managing temple operations is mostly manual, using paper records, spreadsheets, or basic communication tools. Marriage hall bookings, event schedules, and seva offerings are often written in physical registers, which can lead to mistakes, missing records, and difficulties in finding information. Assigning tasks to priests and managing their schedules is done informally, causing miscommunication and overlaps. Tracking financial details, like the costs of sevas and events, is time-consuming and prone to errors. Devotees often struggle to get accurate information about available services, event timings, or priest availability because there is no central system. Security is also a concern, as important details like bookings and resource availability are not well-protected. Overall, the manual system is inefficient, unscalable, and lacks transparency, making it unsuitable for meeting the growing needs of temples. This shows the need for a better, automated Temple Management System to fix these problems.

### DISADVANTAGES

The manual system for managing temple operations has several drawbacks. It is time- consuming and prone to errors, as tasks like managing bookings, schedules, and records are done by hand. Finding specific information is slow and inefficient, and there is a high risk of miscommunication or overlapping duties due to informal task assignments. Sensitive information, like booking details and schedules, is not well-secured and may be accessed by unauthorized people. Physical records can be easily lost or damaged, making data retrieval difficult. The system is also not scalable, meaning it cannot handle the growing needs of a

modern temple. Devotees often face delays or struggle to get accurate information about services and events, while financial tracking is tedious and error-prone. These challenges make the manual system inefficient and unsuitable for managing temple operations effectively.

### PROPOSING SYSTEM

The proposed Temple Seva Management System is an automated, database-driven solution designed to fix the problems of the manual system. It brings together different modules to manage marriage halls, special events, priests, and sevas in one organized platform. The marriage hall module allows secure bookings, keeping detailed records of hall capacity, amenities, and schedules. The special events module helps in planning events by recording details like venue, time, and date under the guidance of managers. Priest management stores important information like contact details, expertise, and salaries, making it easier to assign them to rituals and ceremonies. The seva module tracks services, their costs, descriptions, and the priests involved, giving devotees easy access to information.

The system uses role-based access, so managers can securely manage operations while protecting sensitive information from unauthorized access. By centralizing all operations in a relational database, it avoids duplication, improves data accuracy, and makes retrieval faster. This system also ensures better transparency, scalability, and ease of management. It provides a smooth experience for both temple staff and devotees, simplifies resource management, and makes temple activities more efficient and organized.

### ADVANTAGES

The proposed Temple Seva Management System offers many advantages that improve the efficiency and organization of temple operations. It automates key processes, saving time and reducing manual errors. All data, such as marriage hall bookings, event schedules, priest details, and seva offerings, is stored in a centralized database, making it easy to retrieve and manage information. This system eliminates duplication of data and ensures accuracy, allowing temple administrators to focus on more important tasks.

Role-based access ensures that only authorized users, like managers, can access sensitive information, enhancing security and privacy. Devotees benefit from the system as they can quickly get accurate information about services, events, and priest availability. The system provides better transparency and accountability by keeping clear records of all operations, including bookings, events, and financial details. It also simplifies financial management by automatically tracking costs and payments, streamlining temple activities, and improving the experience for both administrators and devotees.

**CHAPTER 2**

# LITERATURE SURVEY

##### Existing Manual Systems

Many temples still rely on paper-based records to manage operations such as marriage hall bookings, seva services, event schedules, and priest assignments. This manual approach often leads to problems like misplaced records, data entry errors, and difficulty retrieving information quickly. Temple staff must spend significant time updating registers and managing resources, leading to inefficiency. Devotees face challenges in obtaining timely and accurate information due to the lack of an organized system. As temples grow, the limitations of the manual system become more apparent, highlighting the need for a modern digital solution.

##### Digitalization of Religious Institutions

Digitalization in religious institutions has been explored with positive results, as automation improves the management of resources like event scheduling, resource allocation, and financial tracking. Some temples have adopted digital systems to reduce errors and improve operational efficiency. However, many temples still lack an integrated system that manages all operations in one place. This gap in temple management highlights the importance of implementing a comprehensive Temple Seva Management System to streamline all operations.

##### Database-Driven Management Systems

A database-driven approach to managing temple operations offers several advantages, such as avoiding data redundancy and errors by storing marriage hall bookings, priest schedules, and seva services in a relational database. This allows administrators to quickly retrieve and update information, improving data accuracy and efficiency. Centralizing all temple-related data into a single database makes it easier to manage operations in real-time and improves transparency by logging all actions, ensuring accountability in temple operations.

##### Role-Based Access Control (RBAC)

Role-Based Access Control (RBAC) is an important security measure that allows administrators to assign different levels of access based on roles, ensuring sensitive data is protected. For example, staff managing bookings may access event schedules, while managers can modify financial records. Implementing RBAC ensures that only authorized personnel can access critical data, improving security and simplifying user

management by adjusting access permissions as needed, ensuring both flexibility and protection.

##### User-Centric Design for Devotee Interaction

A user-centric design is essential for ensuring temple administrators and devotees find the system easy to navigate. A user-friendly interface helps devotees access seva schedules, book services, check events, and view priest availability quickly. An intuitive design reduces the learning curve and encourages greater engagement, leading to higher user satisfaction and participation in temple services, improving overall temple functionality and communication with devotees.

##### Scalable Solutions for Growing Needs

As temples expand, they need scalable systems to handle increased data, users, and interactions. A scalable system ensures that as the temple grows, the management system adapts to accommodate more services, events, and resources without compromising performance. Scalable systems, such as relational databases with modular architecture, allow the system to continue functioning efficiently as the temple’s operations grow, ensuring smooth management even with increased complexity.

##### Case Studies in Similar Systems

Case studies of religious organizations or community event management platforms provide valuable insights into successful digital solutions. Church management software and community platforms often include modules for booking services, scheduling events, managing staff, and handling donations. These systems have proven effective in simplifying operations and improving efficiency. By learning from these case studies, temples can adapt these strategies to improve their own management processes.

The literature shows that moving from manual systems to digital solutions will provide significant benefits for temple management. A Temple Seva Management System will improve data accuracy, increase efficiency, and ensure better security for sensitive information. The system will be scalable, allowing it to grow with the temple’s needs, and streamline processes, providing better services for both temple administrators and devotees. This transformation will make temple operations more transparent, efficient, and user-friendly.

**CHAPTER 3**

# SYSTEM REQUIREMENTS

#### Hardware Requirements

The minimum hardware components

* + - Intel Pentium Processor at 500 MHz or Faster
    - PC with 512 GB or more Hard disk
    - Wi-Fi Card or Ethernet Card
    - Mouses or Other Pointing Devices
    - Keyboard

#### Software Requirements

* + - **Operating System:** Windows
    - **Frontend:** HTML, CSS
    - **Backend:** PHP
    - **Database:** MySQL
    - **Code editor:** Visual Studio Code
    - **Browser:** Google Chrome
    - **Xampp Server**

**CHAPTER 4**

# SYSTEM ARCHITECTURE

The architecture of the Temple Management System is organized to ensure that temple activities are managed efficiently. It is divided into different layers, each responsible for specific tasks, all working together to provide a smooth and effective management system. The system is built using a **client-server architecture**, where the client interacts with the system and the server handles the data and business logic. Here’s a breakdown of the architecture:

##### User Interface (Client Layer):

* + The **user interface** is the part of the system where users interact with the software. It is designed to be simple and easy to use for everyone, including temple administrators, staff, and devotees.
  + Users access the system via a **web-based interface**, meaning they can use it through any browser (like Chrome, Firefox, etc.), so it’s accessible on computers, tablets, and smartphones.
  + In this layer, users can do things like:
    - **Book marriage halls**
    - **Check event schedules**
    - **View available seva services**
    - **See priest availability**
  + The interface is designed to be user-friendly, so even people with limited technical knowledge can use it easily.

##### Application Layer:

* + The **application layer** is the heart of the system, where all the **business logic** happens. It is responsible for processing the actions that users take and ensuring everything works correctly.
  + This layer contains different **modules** for managing various temple activities, such as:
    - **Marriage Hall Management**: Handles bookings, capacity, and schedule details.
    - **Event Management**: Manages special events, including details like time, date, and location.
    - **Priest Management**: Manages priest schedules and their assignments

for rituals.

* + - **Seva Management**: Handles the tracking of seva services, including descriptions, costs, and availability.
  + Each time a user performs an action, like booking a hall or checking an event, the application layer processes the request and interacts with the database to store or retrieve data.

##### Database Layer:

* + The **database layer** is where all the important information is stored. This includes all the data about marriage halls, events, priests, and sevas.
  + The database uses a **relational database system**, which means that data is organized into **tables** that are linked to each other. For example, a table may store event details, while another stores priest information. The system can then pull together information from different tables when needed.
  + The data is carefully structured and organized so that it is easy to find and update. For example, when a marriage hall is booked, the system updates the booking status in the database.

##### Security Layer:

* + The **security layer** is very important to protect sensitive information and ensure that only the right people can access certain data.
  + The system uses a method called **Role-Based Access Control (RBAC)**. This means that users have specific **roles**, and based on these roles, they are allowed to access only certain parts of the system.
    - For example, a temple manager might have access to all the information about events and bookings, while a regular user can only view event schedules or book sevas.
  + This helps protect private data, like financial records or personal details of devotees, by making sure only authorized people can see or edit it.

##### Server Layer:

* + The **server layer** is where the system runs and processes all the requests from users.
  + The server is responsible for receiving requests from the **client layer**, processing them, and sending back the appropriate information.
  + It also communicates with the **database layer** to fetch or update data as needed.
  + The server ensures that the system can handle multiple users at once, ensuring everything works smoothly, even when many people are using the system at the same time.

1. **Integration for Efficient Temple Management**

By combining these layers in a **well-integrated system architecture**, the Temple Management System can effectively manage the temple’s resources, events, and services. The **client-server model** allows for easy access and interaction with the system, while the **business logic layer** processes user requests accurately. The **database layer** ensures that all data is securely stored and can be retrieved quickly, while the **security layer** protects sensitive information from unauthorized access. The **server layer** ensures that all requests are processed efficiently, even under high usage.

This architecture provides a comprehensive and **scalable solution** for managing temple activities, allowing the system to grow as the temple expands its operations. By leveraging these layers, the system can provide efficient, secure, and user-friendly services to both temple administrators and devotees, ultimately streamlining temple operations and improving overall user experience.

The system architecture is built with different layers that work together to provide a smooth experience for temple administrators and devotees. It uses a **client-server model**, where users interact with the system through a web interface, the business logic is processed in the application layer, data is stored and managed in a database, and security is ensured through role-based access. The server handles the requests and ensures everything runs smoothly. This setup helps in managing temple activities efficiently, securely, and with ease.

## CHAPTER 5

### SOURCE CODE

1. **SQL**

##### temple.sql

-- Manager Table

# IMPLEMENTATION

CREATE TABLE Manager ( ManagerID INT PRIMARY KEY,

ManagerName VARCHAR(255), ContactNumber VARCHAR(15), Email VARCHAR(255),

Address VARCHAR(255), Username VARCHAR(50), Password VARCHAR(50)

);

-- Priest Table

CREATE TABLE Priest ( PriestID INT PRIMARY KEY,

PriestName VARCHAR(255), ContactNumber VARCHAR(15), Email VARCHAR(255),

Address VARCHAR(255), PoojaExpertise VARCHAR(255), Salary DECIMAL(10, 2)

);

-- Seva Table

CREATE TABLE Seva ( SevaID INT PRIMARY KEY, SevaName VARCHAR(255),

Description TEXT,

Duration INT,

Cost DECIMAL(10, 2),

PriestID INT, Date DATE,

FOREIGN KEY (PriestID) REFERENCES Priest(PriestID)

);

-- Marriage Hall Table

CREATE TABLE MarriageHall ( HallID INT PRIMARY KEY, HallName VARCHAR(255),

Capacity INT, Amenities TEXT, BookingDate DATE, BookingTime TIME, ManagerID INT,

FOREIGN KEY (ManagerID) REFERENCES Manager(ManagerID)

);

-- Special Events Table

CREATE TABLE SpecialEvents ( EventID INT PRIMARY KEY,

EventName VARCHAR(255), Description TEXT,

Date DATE, Time TIME,

Venue VARCHAR(255),

ManagerID INT,

FOREIGN KEY (ManagerID) REFERENCES Manager(ManagerID)

);

##### Php database.php

<?php

$db\_server="localhost";

$db\_user="root";

$db\_pass="";

$db\_name="temple"; try{

$conn=mysqli\_connect($db\_server,$db\_user,$db\_pass,$db\_name);

}

catch(mysqli\_sql\_exception)

{

echo"";

}

?>

##### index.php

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Temple Management System</title>

<style>

body {

font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif; margin: 0;

padding: 0;

background: linear-gradient(to bottom, #FFF8E7, #FFD699); /\* Subtle gradient \*/

}

header {

background: linear-gradient(to right, #FFA500, #FF8C00); color: #fff;

text-align: center; padding: 20px 0;

box-shadow: 0 4px 6px rgba(0, 0, 0, 0.3);

}

header h1 {

font-size: 3rem;

font-family: 'Georgia', serif; letter-spacing: 2px;

margin: 0;

}

nav {

background-color: #FFD699; display: flex;

justify-content: center; padding: 10px 0;

box-shadow: 0 2px 4px rgba(0, 0, 0, 0.2);

}

nav ul {

list-style: none; margin: 0;

padding: 0; display: flex; gap: 30px;

}

nav ul li { margin: 0;

}

nav ul li a {

color: #FF8C00;

text-decoration: none; font-size: 18px;

font-weight: bold; padding: 10px 15px; border-radius: 5px;

transition: background-color 0.3s, transform 0.3s;

}

nav ul li a:hover {

background-color: #FF8C00; color: #FFD699;

transform: scale(1.1);

}

section#home {

background: url('img/img1.jpg') no-repeat center center; background-size: cover;

height: calc(100vh - 70px); display: flex;

flex-direction: column; justify-content: center; align-items: center;

text-shadow: 2px 2px 5px rgba(0, 0, 0, 0.7); color: #fff;

padding: 20px;

box-shadow: inset 0 0 20px rgba(0, 0, 0, 0.5);

}

section#home .overlay { background: rgba(0, 0, 0, 0.5); padding: 30px;

border-radius: 15px; text-align: center;

}

section#home h2 { font-size: 3.5rem;

margin-bottom: 20px;

font-family: 'Georgia', serif; letter-spacing: 3px;

}

section#home a { display: inline-block; margin: 15px; padding: 15px 30px;

font-size: 1.2rem; font-weight: bold; color: #fff;

background: linear-gradient(to right, #FFA500, #FF8C00); border-radius: 10px;

text-decoration: none;

box-shadow: 0 4px 6px rgba(0, 0, 0, 0.3); transition: transform 0.3s, box-shadow 0.3s;

}

section#home a:hover { transform: scale(1.1);

box-shadow: 0 6px 10px rgba(0, 0, 0, 0.5);

}

footer {

background: linear-gradient(to right, #FFA500, #FF8C00); color: #FFD699;

text-align: center; padding: 15px 0; position: fixed; bottom: 0;

width: 100%;

box-shadow: 0 -4px 6px rgba(0, 0, 0, 0.2);

}

footer a {

color: #FFD699;

text-decoration: none; font-weight: bold; margin: 0 10px;

}

footer a:hover {

text-decoration: underline;

}

</style>

</head>

<body>

<header>

<h1>Sri Adichunchanagiri Kshethra Seva Management</h1>

</header>

<nav>

<ul>

<li><a href="index.php">Home</a></li>

<li><a href="#">About</a></li>

<li><a href="#">Contact</a></li>

</ul>

</nav>

<section id="home">

<div class="overlay">

<h2>Embrace the Divine Journey</h2>

<a href="login.php">Login</a>

<a href="signup.php">Signup</a>

</div>

</section>

<footer>

<a href="index.php">Home</a> | <a href="#">Privacy Policy</a> | <a href="#">Terms of Use</a>

</footer>

</body>

</html>

##### login.php

<?php session\_start();

include("database.php");

if ($\_SERVER['REQUEST\_METHOD'] == "POST") {

$manager\_username = $\_POST['manager\_username'];

$password = $\_POST['password'];

$query = "SELECT \* FROM Manager WHERE Username = '$manager\_username' AND Password = '$password'";

$result = mysqli\_query($conn, $query);

if ($result && mysqli\_num\_rows($result) > 0) {

$\_SESSION['manager\_username'] = $manager\_username; header("Location: menu.php");

exit;

} else {

echo "<script>alert('Invalid username or password');</script>";

}

}

?>

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Login Page</title>

<link rel="stylesheet" href="styles.css">

<style>

body {

background: linear-gradient(to right, #FFA500, #FF6347); /\* Orange gradient \*/ background-image: url('img/img2.jpg'); /\* Add your image path \*/

background-size: cover; background-position: center; background-blend-mode: overlay;

background-color: rgba(255, 165, 0, 0.5); /\* Light overlay for the image \*/ font-family: 'Arial', sans-serif;

display: flex;

justify-content: center; align-items: center; height: 100vh; margin: 0;

}

form {

background-color: rgba(255, 255, 255, 0.9); /\* Light transparent background for the

form \*/

padding: 30px; border-radius: 12px;

box-shadow: 0 10px 20px rgba(0, 0, 0, 0.1);

width: 100%;

max-width: 400px;

}

h1 {

font-size: 2rem; color: #333333; text-align: center;

margin-bottom: 20px;

}

label {

font-size: 1rem; color: #555555;

margin: 10px 0; display: block;

}

input {

width: 100%; padding: 12px; margin-bottom: 15px;

border: 2px solid #ddd; border-radius: 8px; font-size: 1rem;

transition: border-color 0.3s ease;

}

input:focus {

border-color: #FF6347; /\* Focus border color \*/ outline: none;

}

input[type="submit"] {

background-color: #FF6347; /\* Button color \*/ color: #ffffff;

cursor: pointer; font-size: 1.1rem; padding: 12px; border: none; border-radius: 8px; width: 100%;

transition: background-color 0.3s ease;

}

input[type="submit"]:hover {

background-color: #e53e3e; /\* Button hover color \*/

}

p {

text-align: center; color: #555555; margin-top: 20px; font-size: 0.9rem;

}

a {

color: #FF6347;

text-decoration: none; font-weight: bold;

}

a:hover {

text-decoration: underline;

}

@media (max-width: 768px) { form {

padding: 20px; max-width: 90%;

}

h1 {

font-size: 1.5rem;

}

}

</style>

</head>

<body>

<form method="post">

<h1>Login</h1>

<label for="manager\_username">Username:</label>

<input type="text" id="manager\_username" name="manager\_username" required><br>

<label for="password">Password:</label>

<input type="password" id="password" name="password" required><br><br>

<input type="submit" value="Login">

<p>Don't have an account? <a href="signup.php">Sign up here</a></p>

</form>

</body>

</html>

##### signup.php

<?php session\_start();

include("database.php");

if ($\_SERVER['REQUEST\_METHOD'] == "POST") {

$mid = $\_POST['ManagerID'];

$manager\_name = $\_POST['ManagerName'];

$username = $\_POST['Username'];

$password = $\_POST['Password'];

$contact\_number = $\_POST['ContactNumber'];

$email = $\_POST['Email'];

$address = $\_POST['Address'];

if (!empty($username) && !empty($password) && !empty($email) && !empty($address))

{

// Use prepared statements to prevent SQL injection

$stmt = $conn->prepare("INSERT INTO Manager (ManagerID, ManagerName,

ContactNumber, Email, Address, Username, Password)

VALUES (?, ?, ?, ?, ?, ?, ?)");

$stmt->bind\_param("issssss", $mid, $manager\_name, $contact\_number, $email,

$address, $username, $password);

if ($stmt->execute()) {

header("Location: login.php"); // Redirect to login page after successful signup die;

} else {

echo "<script type='text/javascript'>alert('Error: Unable to create account. Please try again.')</script>";

}

$stmt->close();

} else {

echo "<script type='text/javascript'>alert('Please enter all required fields.')</script>";

}

}

?>

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Manager Signup Page</title>

<link rel="stylesheet" href="styles.css">

<style>

\* {

box-sizing: border-box; margin: 0;

padding: 0;

}

body {

font-family: 'Arial', sans-serif; display: flex;

justify-content: center; align-items: center; height: 100vh; background: #FFD699;

}

form {

background-color: #ffffff; padding: 30px;

border-radius: 10px;

box-shadow: 0 5px 15px rgba(0, 0, 0, 0.1);

width: 80%;

max-width: 900px;

}

h1 {

font-size: 2rem; color: #FF6347; margin-bottom: 20px; text-align: center;

}

.form-row { display: flex;

justify-content: space-between; gap: 20px;

margin-bottom: 20px;

}

.form-row .form-column { width: 48%;

}

label {

font-size: 1rem; color: #555;

margin: 10px 0; display: block;

}

input[type="text"], input[type="email"], input[type="password"], input[type="tel"], input[type="number"] {

width: 100%;

padding: 12px; margin-bottom: 15px;

border: 2px solid #ddd; border-radius: 5px; font-size: 1rem;

color: #555;

}

input[type="submit"] { background-color: #FF6347; color: white;

border: none; padding: 15px; font-size: 1.1rem; cursor: pointer; border-radius: 5px; width: 100%;

transition: background-color 0.3s ease;

}

input[type="submit"]:hover { background-color: #FF4500;

}

p {

margin-top: 15px; color: #555;

font-size: 0.9rem; text-align: center;

}

a {

color: #FF6347;

text-decoration: none;

}

a:hover {

text-decoration: underline;

}

@media (max-width: 768px) {

.form-row {

flex-direction: column;

}

.form-row .form-column { width: 100%;

}

}

</style>

</head>

<body>

<form method="post">

<h1>Manager Sign Up</h1>

<div class="form-row">

<div class="form-column">

<label for="ManagerID">Manager ID:</label>

<input type="number" id="ManagerID" name="ManagerID" required><br>

<label for="ManagerName">Manager Name:</label>

<input type="text" id="ManagerName" name="ManagerName" required><br>

<label for="Username">Username:</label>

<input type="text" id="Username" name="Username" required><br>

</div>

<div class="form-column">

<label for="ContactNumber">Contact Number:</label>

<input type="tel" id="ContactNumber" name="ContactNumber" required><br>

<label for="Email">Email:</label>

<input type="email" id="Email" name="Email" required><br>

<label for="Address">Address:</label>

<input type="text" id="Address" name="Address" required><br>

</div>

</div>

<label for="Password">Password:</label>

<input type="password" id="Password" name="Password" required><br>

<input type="submit" value="Submit">

<p>Already have an account? <a href="login.php">Login here</a></p>

</form>

</body>

</html>

##### menu.php

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>MENU</title>

<style>

body {

font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

background: linear-gradient(to right, #FFA500, #FF6347); /\* Orange gradient \*/ margin: 0;

padding: 0; color: #fff;

}

.header {

background-color: #FF6347; /\* Tomato red \*/ color: white;

text-align: center; padding: 5px; font-size: 2rem;

letter-spacing: 2px;

text-shadow: 2px 2px 4px rgba(0, 0, 0, 0.5);

}

.container {

max-width: 1000px; margin: 0px auto; padding: 0px;

background-image: url('img/imgb.JPG'); /\* Add your image URL here \*/ background-size: cover;

background-position: center; /\* Align image at the top \*/ background-repeat: no-repeat;

background-attachment: fixed; /\* Fixed background to avoid scrolling with content \*/ border-radius: 12px;

box-shadow: 0 12px 24px rgba(0, 0, 0, 0.1);

}

.btn-container { display: flex;

flex-direction: column; gap: 15px;

align-items: center; padding: 20px 0;

}

.btn {

background-color: #FF6347; /\* Tomato red \*/ color: white;

padding: 15px 30px; border: none;

border-radius: 8px; cursor: pointer; font-size: 1.1rem; text-align: center;

transition: all 0.3s ease; width: 80%;

max-width: 400px; text-decoration: none;

}

.btn:hover {

background-color: #e53e3e; /\* Slightly darker red on hover \*/ transform: scale(1.05);

}

.home-btn { position: absolute; top: 20px;

right: 20px;

background-color: #333; padding: 10px 20px; border-radius: 8px; color: #fff;

font-size: 1rem;

text-decoration: none;

transition: background-color 0.3s;

}

.home-btn:hover { background-color: #444;

}

/\* Responsive adjustments \*/ @media (max-width: 768px) {

.container { padding: 15px;

}

.btn-container { padding: 10px;

}

.btn {

width: 90%; padding: 12px;

}

}

/\* Styling for images \*/

.image-gallery { display: flex;

justify-content: center; gap: 20px;

margin-top: 20px;

flex-wrap: nowrap; /\* Prevent wrapping \*/

}

.image-gallery a {

text-align: center; display: block;

width: 300px; /\* Set width to make images equal size \*/

}

.image-gallery img { width: 100%; height: auto; border-radius: 8px;

box-shadow: 0 4px 10px rgba(0, 0, 0, 0.2);

}

.image-gallery figcaption { margin-top: 10px;

color: #fff;

font-size: 1.2rem; font-weight: bold;

text-shadow: 1px 1px 2px rgba(0, 0, 0, 0.5);

}

</style>

</head>

<body>

<div class="header">

<h1>Sri Adichunchanagiri Kshethra</h1>

</div>

<div class="container">

<a href="index.php" class="btn home-btn">LOG OUT</a>

<div class="btn-container">

<a href="hall.php" class="btn">MARRIAGE HALL DETAILS</a>

<a href="priest.php" class="btn">PRIEST DETAILS</a>

<a href="seva.php" class="btn">SEVA DETAILS</a>

<a href="special.php" class="btn">SPECIAL EVENT DETAILS</a>

</div>

</div>

<!-- Image Gallery Section -->

<div class="image-gallery">

<figure>

<a href="hall.php">

<img src="img/marriage.jpg" alt="Marriage Hall">

<figcaption>Marriage Hall Details</figcaption>

</a>

</figure>

<figure>

<a href="priest.php">

<img src="img/priests.jpg" alt="Priests">

<figcaption>Priest Details</figcaption>

</a>

</figure>

<figure>

<a href="seva.php">

<img src="img/seva.jpg" alt="Seva">

<figcaption>Seva Details</figcaption>

</a>

</figure>

<figure>

<a href="special.php">

<img src="img/events.jpg" alt="Special Events">

<figcaption>Special Event Details</figcaption>

</a>

</figure>

</div>

</body>

</html>

##### hall.php

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Marriage Hall Management</title>

<style>

body {

font-family: 'Arial', sans-serif; margin: 0;

padding: 0;

background-color: #FFF8E1; /\* Light Cream \*/ display: flex;

}

/\* Sidebar Navigation \*/ nav {

background-color: #FF6F00; /\* Bright Orange \*/ color: white;

width: 200px; /\* Reduced width \*/ height: 100vh;

padding-top: 20px; position: fixed; top: 0;

left: 0;

box-shadow: 2px 0 8px rgba(0, 0, 0, 0.2);

}

nav a {

color: white;

text-decoration: none; padding: 15px; display: block;

font-size: 18px;

border-bottom: 1px solid #FF8F00; transition: background-color 0.3s ease;

}

nav a:hover {

background-color: #FF8F00; /\* Darker orange on hover \*/

}

/\* Content Section \*/

.content {

margin-left: 220px; /\* Adjusted margin for smaller sidebar \*/

width: calc(100% - 220px); /\* Adjusted to make more space for table \*/ padding: 40px;

display: flex;

flex-direction: column; align-items: center;

}

/\* Header \*/ header {

background-color: #FF8F00; /\* Darker Orange \*/ color: white;

padding: 15px; width: 100%; display: flex;

justify-content: center; position: fixed;

top: 0;

left: 220px; /\* Adjusted header position \*/ z-index: 10;

}

h1 {

margin: 0;

font-size: 24px;

}

/\* Search Bar and Button \*/

.search-container {

margin-top: 80px; /\* Space below header \*/ margin-bottom: 30px;

display: flex;

justify-content: center; gap: 15px;

}

input[type="text"] { padding: 12px;

font-size: 16px; width: 300px; border-radius: 8px;

border: 1px solid #bbb; outline: none;

transition: border-color 0.3s ease;

}

input[type="text"]:focus { border-color: #FF8F00;

}

button {

padding: 12px 20px;

background-color: #FF8F00; /\* Darker Orange \*/ color: white;

border: none; border-radius: 8px; font-size: 16px; cursor: pointer;

transition: background-color 0.3s ease;

}

button:hover {

background-color: #FF6F00; /\* Darker shade on hover \*/

}

/\* Table Styling \*/ table {

width: 95%; /\* Increased table width \*/ margin-top: 40px;

border-collapse: collapse; background-color: #fff; border-radius: 10px;

box-shadow: 0 4px 12px rgba(0, 0, 0, 0.1);

}

th, td {

padding: 15px;

text-align: left; font-size: 16px;

border-bottom: 1px solid #ddd;

}

th {

background-color: #FF8F00; /\* Darker Orange \*/ color: white;

}

tr:nth-child(even) {

background-color: #f9f9f9; /\* Light gray for even rows \*/

}

tr:hover {

background-color: #f1f1f1;

}

/\* No Results Text \*/

.no-results {

text-align: center; font-size: 18px; color: #FF6F00; margin-top: 20px;

}

</style>

</head>

<body>

<!-- Sidebar -->

<nav>

<a href="menu.php">Menu</a>

<a href="u3.php">Update</a>

<a href="del3.php">Delete</a>

<a href="in3.php">Insert</a>

</nav>

<!-- Main Content -->

<div class="content">

<header>

<h1>Marriage Hall Management</h1>

</header>

<!-- Search Bar -->

<div class="search-container">

<input type="text" id="hallSearchInput" placeholder="Search by Hall Name">

<button onclick="searchHall()">Search</button>

</div>

<?php include("database.php");

$sql = "SELECT \* FROM MarriageHall";

$result = $conn->query($sql); if ($result->num\_rows > 0) {

echo "<table>

<tr>

<th>Hall ID</th>

<th>Hall Name</th>

<th>Capacity</th>

<th>Amenities</th>

<th>Booking Date</th>

<th>Booking Time</th>

<th>Manager ID</th>

</tr>";

while($row = $result->fetch\_assoc()) { echo "<tr>

<td>".$row["HallID"]."</td>

<td>".$row["HallName"]."</td>

<td>".$row["Capacity"]."</td>

<td>".$row["Amenities"]."</td>

<td>".$row["BookingDate"]."</td>

<td>".$row["BookingTime"]."</td>

<td>".$row["ManagerID"]."</td>

</tr>";

}

echo "</table>";

} else {

echo "<p class='no-results'>No halls found.</p>";

}

$conn->close();

?>

</div>

<script>

function searchHall() {

var input, filter, table, tr, td, i, txtValue;

input = document.getElementById("hallSearchInput"); filter = input.value.toUpperCase();

table = document.querySelector("table"); tr = table.getElementsByTagName("tr");

for (i = 1; i < tr.length; i++) {

td = tr[i].getElementsByTagName("td")[1]; // Hall Name column if (td) {

txtValue = td.textContent || td.innerText;

if (txtValue.toUpperCase().indexOf(filter) > -1) { tr[i].style.display = "";

} else {

tr[i].style.display = "none";

}

}

}

}

</script>

</body>

</html>

##### priest.php

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Priest Management</title>

</head>

<body>

<!-- Sidebar -->

<nav>

<a href="menu.php">Menu</a>

<a href="up1.php">Update</a>

<a href="del1.php">Delete</a>

<a href="in1.php">Insert</a>

</nav>

<!-- Main Content -->

<div class="content">

<header>

<h1>Priest Management</h1>

</header>

<!-- Search Bar -->

<div class="search-container">

<input type="text" id="priestSearchInput" placeholder="Search by Priest Name">

<button onclick="searchPriest()">Search</button>

</div>

<?php include("database.php");

$sql = "SELECT \* FROM Priest";

$result = $conn->query($sql); if ($result->num\_rows > 0) {

echo "<table>

<tr>

<th>Priest ID</th>

<th>Priest Name</th>

<th>Contact Number</th>

<th>Email</th>

<th>Address</th>

<th>Pooja Expertise</th>

<th>Salary</th>

</tr>";

while($row = $result->fetch\_assoc()) { echo "<tr>

<td>".$row["PriestID"]."</td>

<td>".$row["PriestName"]."</td>

<td>".$row["ContactNumber"]."</td>

<td>".$row["Email"]."</td>

<td>".$row["Address"]."</td>

<td>".$row["PoojaExpertise"]."</td>

<td>".$row["Salary"]."</td>

</tr>";

}

echo "</table>";

} else {

echo "<p class='no-results'>No priests found.</p>";

}

$conn->close();

?>

</div>

<script>

function searchPriest() {

var input, filter, table, tr, td, i, txtValue;

input = document.getElementById("priestSearchInput"); filter = input.value.toUpperCase();

table = document.querySelector("table"); tr = table.getElementsByTagName("tr"); for (i = 1; i < tr.length; i++) {

td = tr[i].getElementsByTagName("td")[1]; // Priest Name column if (td) {

txtValue = td.textContent || td.innerText;

if (txtValue.toUpperCase().indexOf(filter) > -1) { tr[i].style.display = "";

} else {

tr[i].style.display = "none";

}

}

}

}

</script>

</body>

</html>

##### seva.php

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Priest Management</title>

</head>

<body>

<!-- Sidebar -->

<nav>

<a href="menu.php">Menu</a>

<a href="up2.php">Update</a>

<a href="del2.php">Delete</a>

<a href="in2.php">Insert</a>

</nav>

<!-- Main Content -->

<div class="content">

<header>

<h1>Seva Management</h1>

</header>

<!-- Search Bar -->

<div class="search-container">

<input type="text" id="sevaSearchInput" placeholder="Search by Seva Name">

<button onclick="searchSeva()">Search</button>

</div>

<?php include("database.php");

$sql = "SELECT \* FROM Seva";

$result = $conn->query($sql); if ($result->num\_rows > 0) {

echo "<table>

<tr>

<th>Seva ID</th>

<th>Seva Name</th>

<th>Description</th>

<th>Duration</th>

<th>Cost</th>

<th>Priest ID</th>

<th>Date</th>

</tr>";

while($row = $result->fetch\_assoc()) { echo "<tr>

<td>".$row["SevaID"]."</td>

<td>".$row["SevaName"]."</td>

<td>".$row["Description"]."</td>

<td>".$row["Duration"]."</td>

<td>".$row["Cost"]."</td>

<td>".$row["PriestID"]."</td>

<td>".$row["Date"]."</td>

</tr>";

}

echo "</table>";

} else {

echo "<p class='no-results'>No sevas found.</p>";

}

$conn->close();

?>

</div>

<script>

function searchSeva() {

var input, filter, table, tr, td, i, txtValue;

input = document.getElementById("sevaSearchInput"); filter = input.value.toUpperCase();

table = document.querySelector("table"); tr = table.getElementsByTagName("tr"); for (i = 1; i < tr.length; i++) {

td = tr[i].getElementsByTagName("td")[1]; // Seva Name column if (td) {

txtValue = td.textContent || td.innerText;

if (txtValue.toUpperCase().indexOf(filter) > -1) { tr[i].style.display = "";

} else {

tr[i].style.display = "none";

}

}

}

}

</script>

</body>

</html>

##### insert4.php

<?php

include("database.php"); // Assuming you have a database connection file

// Check if the form is submitted

if ($\_SERVER["REQUEST\_METHOD"] == "POST") {

// Collect form data

$eventID = $\_POST["EventID"];

$eventName = $\_POST["EventName"];

$description = $\_POST["Description"];

$date = $\_POST["Date"];

$time = $\_POST["Time"];

$venue = $\_POST["Venue"];

$managerID = $\_POST["ManagerID"];

// Use prepared statements to prevent SQL injection

$insertSQL = "INSERT INTO SpecialEvents (EventID, EventName, Description, Date, Time, Venue, ManagerID)

VALUES (?, ?, ?, ?, ?, ?, ?)";

$stmt = $conn->prepare($insertSQL);

$stmt->bind\_param("issssss", $eventID, $eventName, $description, $date, $time, $venue,

$managerID);

// Execute the query

if ($stmt->execute()) {

echo "New record has been inserted successfully.";

} else {

echo "Error inserting record: " . $stmt->error;

}

// Close the prepared statement

$stmt->close();

}

// Close the database connection

$conn->close();

?>

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Insert New Special Event</title>

</head>

<body>

<h2>Insert New Special Event</h2>

<form method="post" action="<?php echo $\_SERVER["PHP\_SELF"]; ?>">

<label for="EventID">Event ID:</label>

<input type="text" name="EventID" required>

<label for="EventName">Event Name:</label>

<input type="text" name="EventName" required>

<label for="Description">Description:</label>

<textarea name="Description" required></textarea>

<label for="Date">Date:</label>

<input type="date" name="Date" required>

<label for="Time">Time:</label>

<input type="time" name="Time" required>

<label for="Venue">Venue:</label>

<input type="text" name="Venue" required>

<label for="ManagerID">Manager ID:</label>

<input type="text" name="ManagerID" required>

<button type="submit">Insert Record</button>

</form>

<div class="redirect-btn">

<a href="special.php">Special Events Page</a>

</div>

</body>

</html>

##### del4.php

<?php

include("database.php"); // Assuming you have a database connection file if ($\_SERVER["REQUEST\_METHOD"] == "POST") {

// Check if EventID is provided

if (isset($\_POST["EventID"]) && !empty($\_POST["EventID"])) {

$eventID = $\_POST["EventID"];

// SQL to delete the record based on EventID

$deleteSQL = "DELETE FROM SpecialEvents WHERE EventID = $eventID";

// Execute the query

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

echo "Record with EventID $eventID has been deleted successfully.";

} else {

echo "Error deleting record: " . $conn->error;

}

} else {

echo "EventID is required.";

}

}

// Close the database connection

$conn->close();

?>

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Delete Special Event Record</title>

</head>

<body>

<header>

<h1>Delete Special Event Record</h1>

<div class="top-right">

<a href="special.php" class="btn">Seva Page</a>

</div>

</header>

<div class="container">

<form method="post" action="">

<label for="EventID">Enter EventID to delete:</label>

<input type="number" name="EventID" required>

<button type="submit">Delete Record</button>

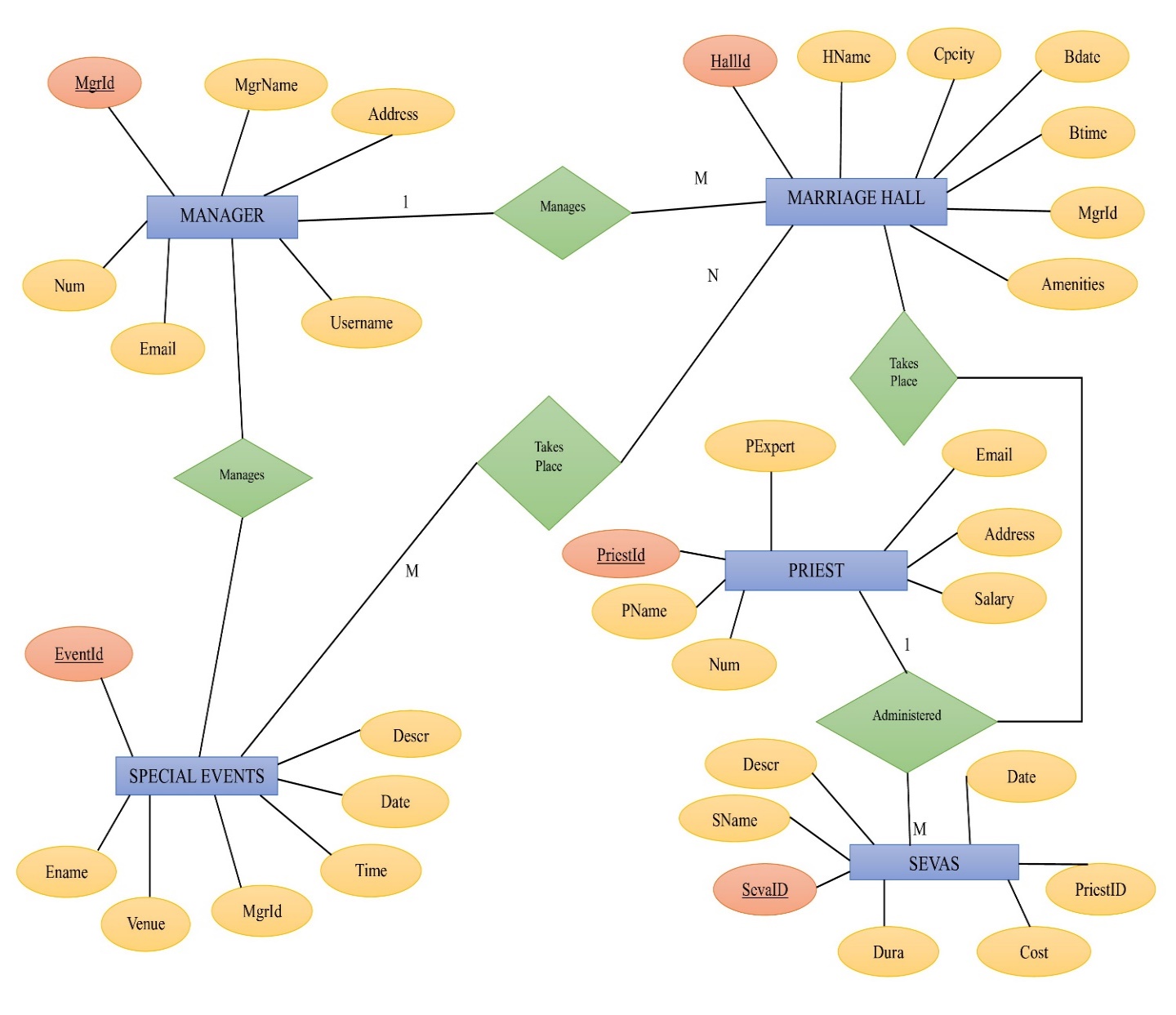
</form>

</div>

</body>

</html>

### ENTITY RELATIONSHIP DIAGRAM

****

**Fig 5.2.1: Entity Relationship Diagram**

##### Entities and Attributes:

* + - **Manager:**
      * **ManagerID:** A unique identifier for each manager.
      * **name:** The manager's full name.
      * **num:** Likely a contact number or employee ID.
      * **password:** For login and security purposes.
      * **uname:** Username for login.

##### Marriage Hall:

* + - * **HallID:** A unique identifier for each hall.
      * **HallName:** The name of the hall.
      * **Capacity:** The number of people it can accommodate.
      * **Amenities:** A list or description of amenities offered (e.g., AC, parking, catering).
      * **DateofBook:** The date of booking for an event (could be redundant).
      * **BookingTime:** The time of booking.
      * **Taken:** Indicates if the hall is booked or available.
      * **Place:** The location of the hall.

##### Special Events:

* + - * **Event\_id:** A unique identifier for each event.
      * **Event\_name:** The name of the event (e.g., "Wedding of John and Jane").
      * **Desc:** A brief description of the event.
      * **Date:** The date of the event.
      * **Venue:** The location of the event (could be a specific hall or another location).
      * **time:** The start time of the event.
      * **cost:** The estimated or actual cost of the event.

##### Priest:

* + - * **priestid:** A unique identifier for each priest.
      * **Priest\_name:** The priest's name.
      * **Pooja\_expert:** Indicates the type of religious ceremonies the priest specializes in.
      * **salary:** The priest's salary or fees.
      * **contact:** The priest's contact information.
      * **email:** The priest's email address.

##### Sevas:

* + - * **Seva\_ID:** A unique identifier for each service.
      * **Seva\_name:** The name of the service (e.g., "Decoration," "Catering," "Photography").
      * **Desc:** A brief description of the service.
      * **Date:** The date the service is required.
      * **cost:** The cost of the service.
      * **Priestid:** The ID of the priest associated with the service (if applicable).

##### Relationships:

Relationships define how entities interact with each other. There are three main types of relationships:

1. **One-to-One (1:1):** One instance of an entity is linked to only one instance of another.
2. **One-to-Many (1:N):** One instance of an entity is linked to many instances of another.
3. **Many-to-Many (M:N):** Many instances of an entity can be linked to many instances of another.

##### Relationships in this System:

1. **Manages:**
   * Relationship: A **Manager** manages one or more **Marriage Halls**.
   * Type: One-to-Many (1:N).
   * Example: Manager A may manage Hall 1 and Hall 2, but Hall 1 is managed by only Manager A.

##### Takes Place:

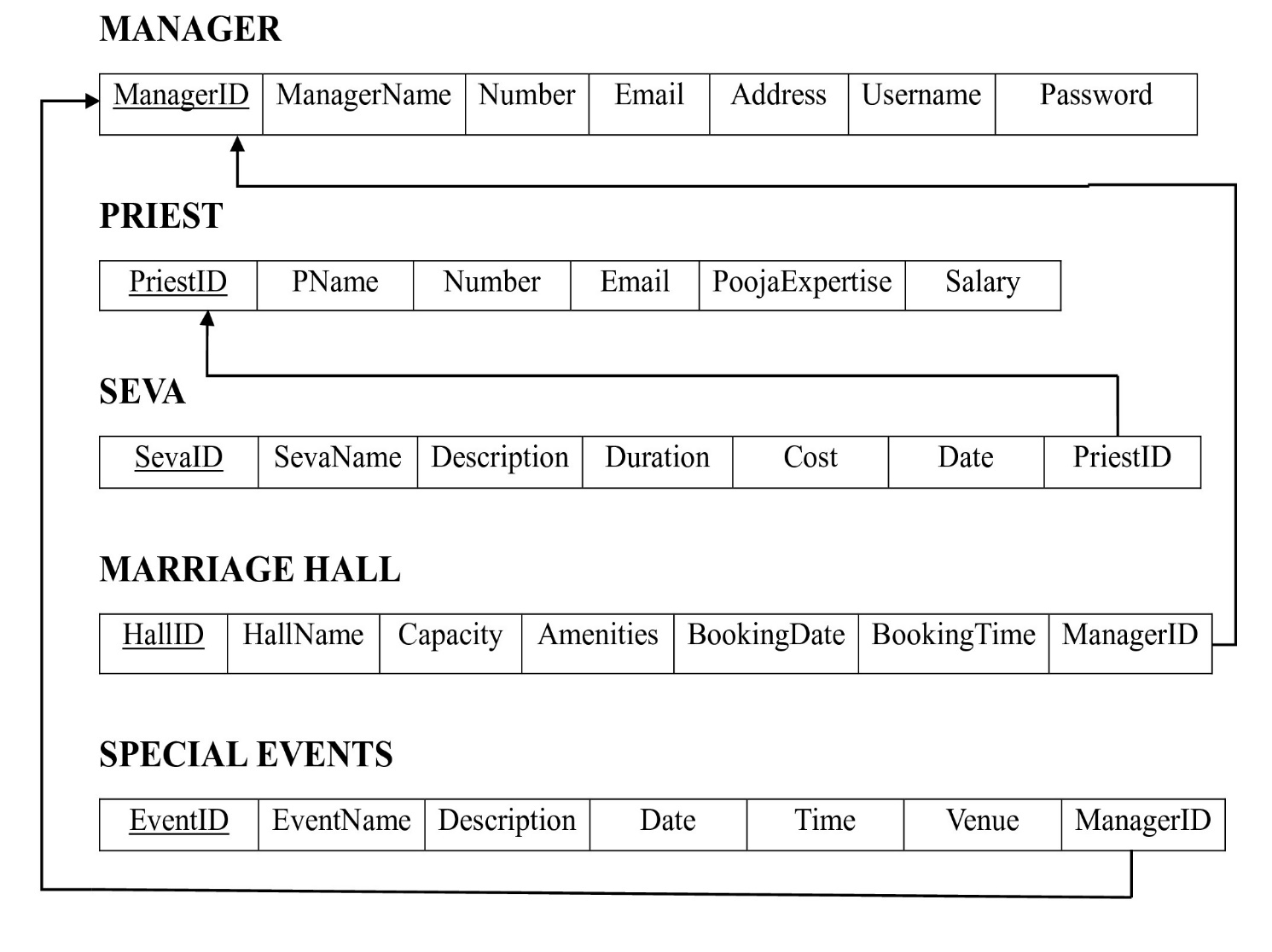
* + Relationship: A **Special Event** can take place in one or more **Marriage Halls**, and a hall can host multiple events.
  + Type: Many-to-Many (M:N).
  + Example: A wedding and a reception can happen in the same hall at different times, and a wedding may involve multiple halls.

##### Administered:

* + Relationship: A **Priest** can administer many **Sevas** (services).
  + Type: One-to-Many (1:N).
  + Example: Priest X can provide services for Decoration, Catering, and Religious Ceremonies.

This ER diagram provides a foundation for managing events, particularly weddings. It allows for tracking managers, their assigned halls, event details, service requirements, and the involvement of priests. By refining the relationships and adding missing attributes, the diagram can be further enhanced to capture more information and improve the database's efficiency.

### SCHEMA DIAGRAM

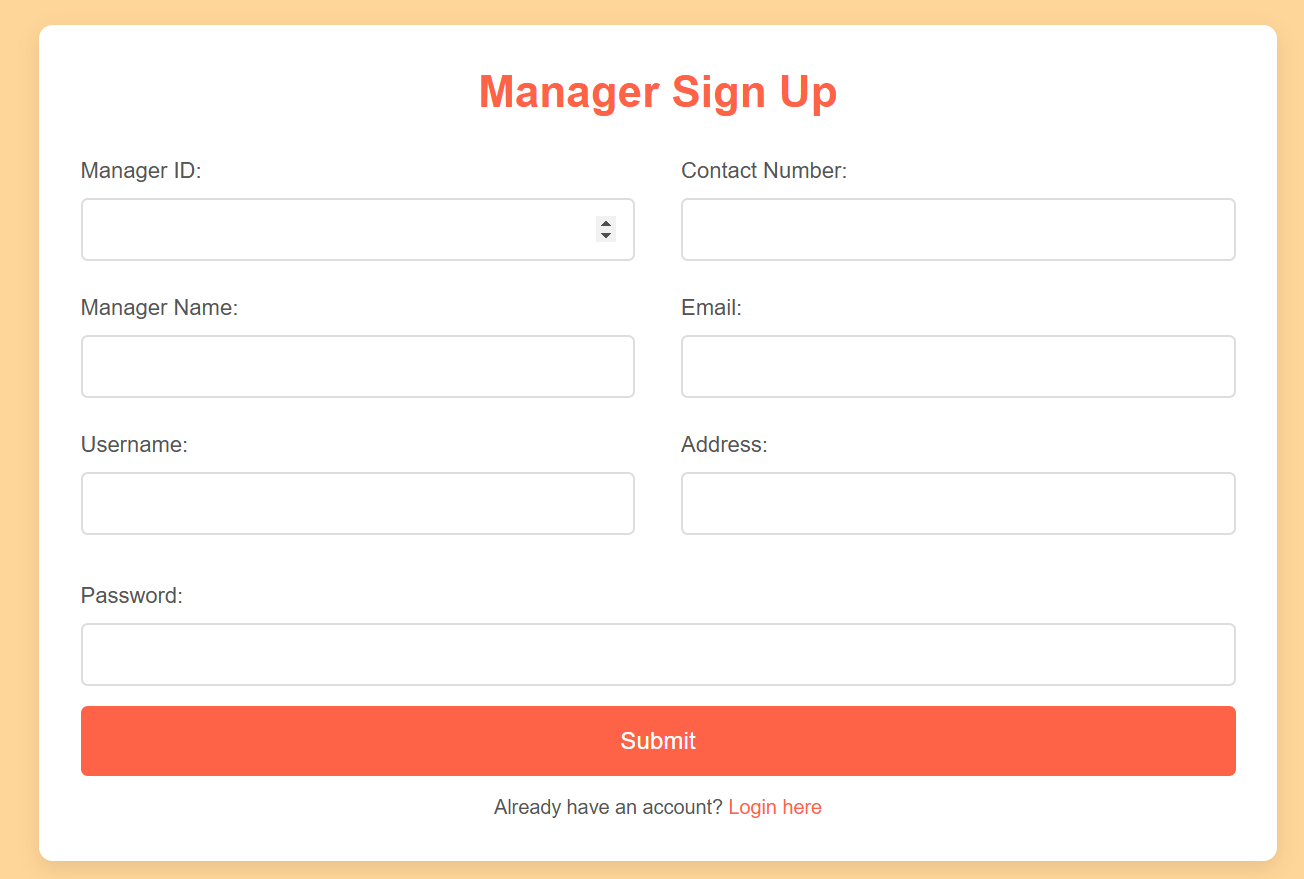
The term "schema" refers to the organization of data as a blueprint of how the database is constructed. The formal definition of a database schema is a set of formulas called integrity constraints imposed on a database. A relational schema shows references among fields in the database. When a primary key is referenced in another table in the database, it is called a foreign key. This is denoted by an arrow with the head pointing at the referenced key attribute. A schema diagram helps organize values in the database. The following diagram shows the schema diagram for the database.

**Fig 5.3.1: Schema Diagram**

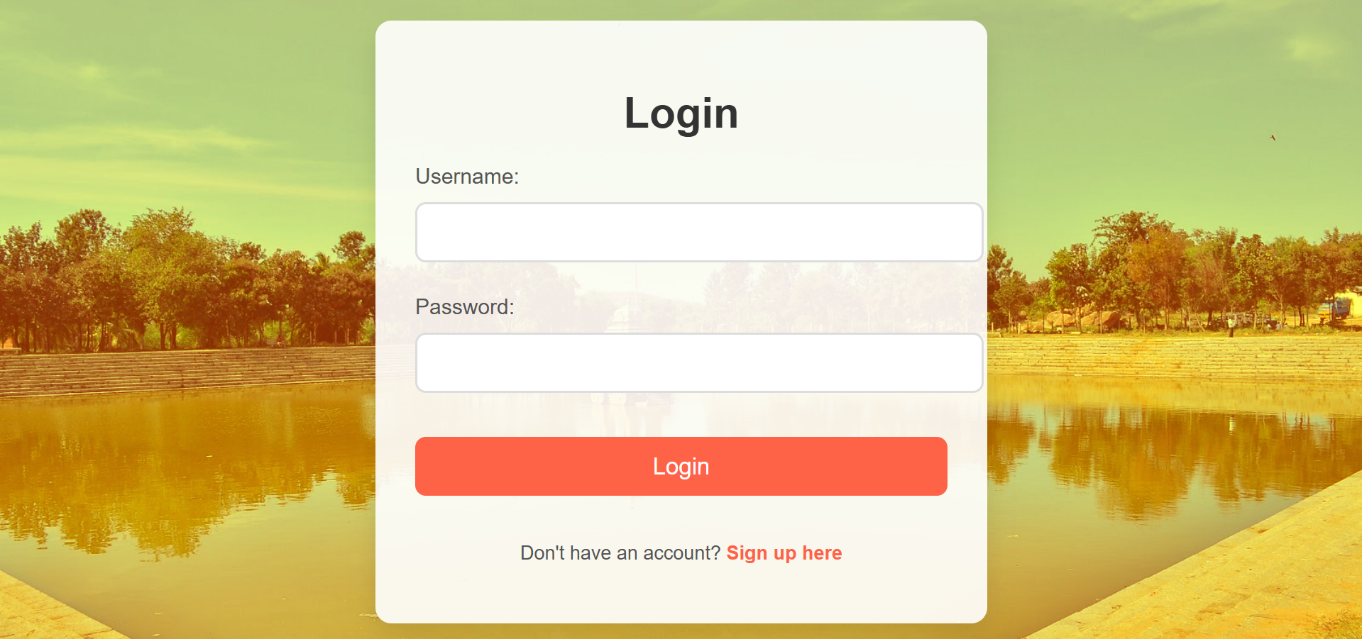
## CHAPTER 6

# RESULT AND SNAPSHOT

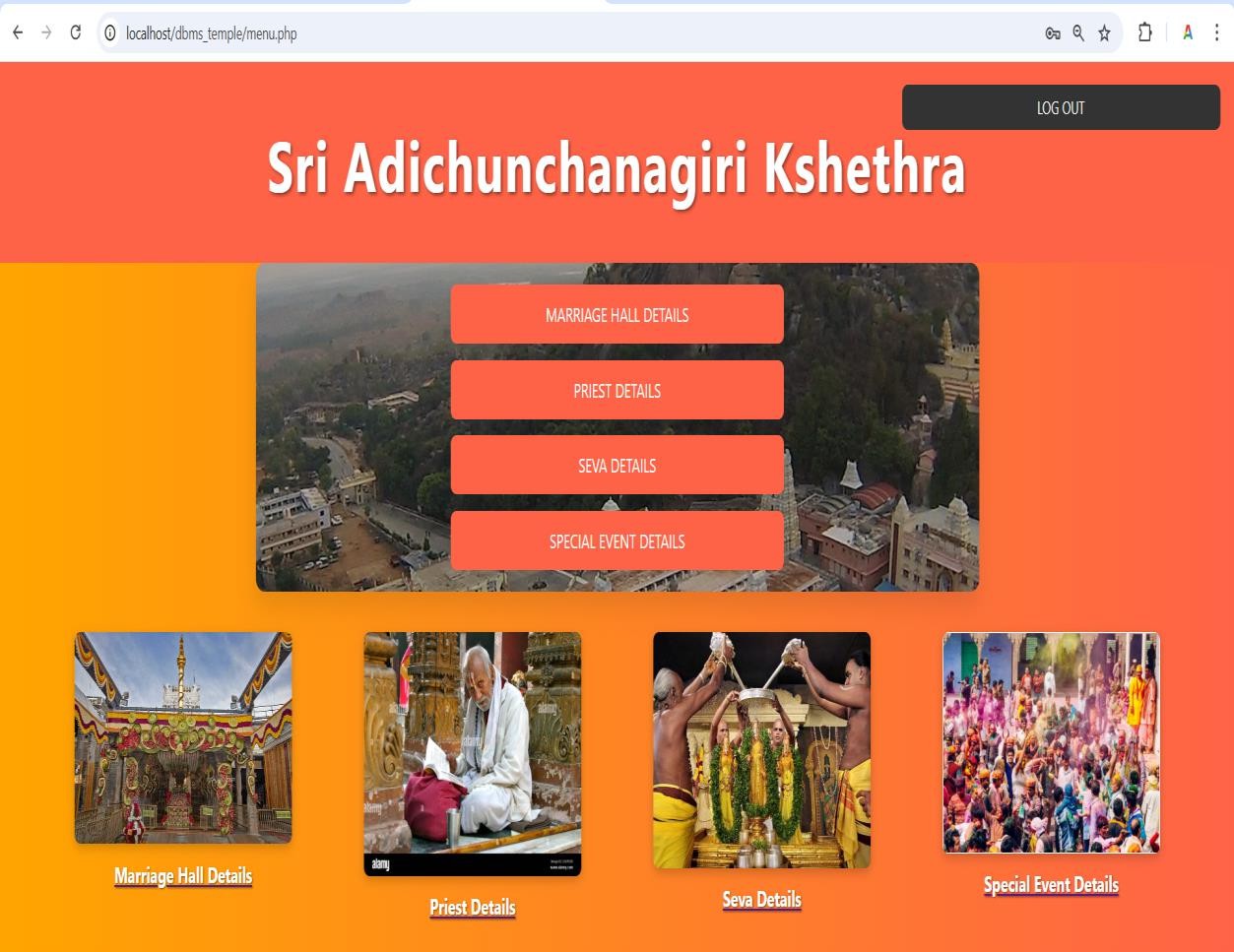
**Fig 6.1: Home Page**

****

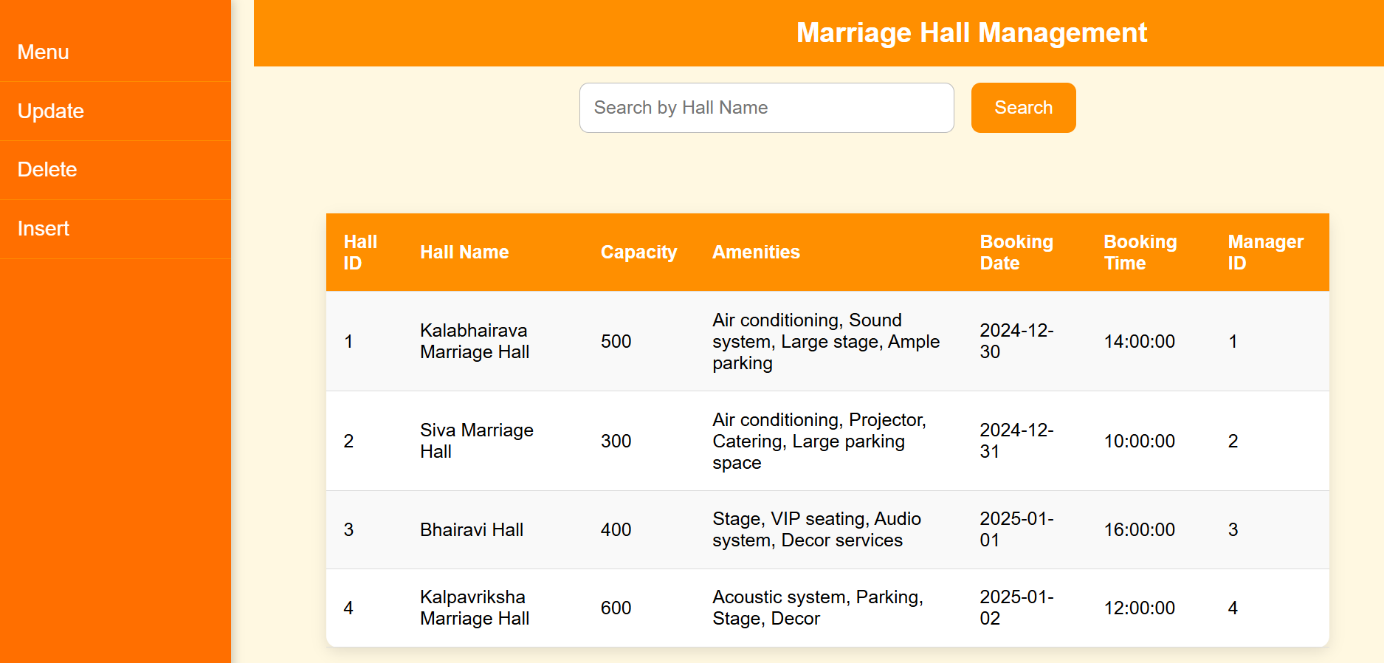
**Fig 6.2: Manager Sign Up**



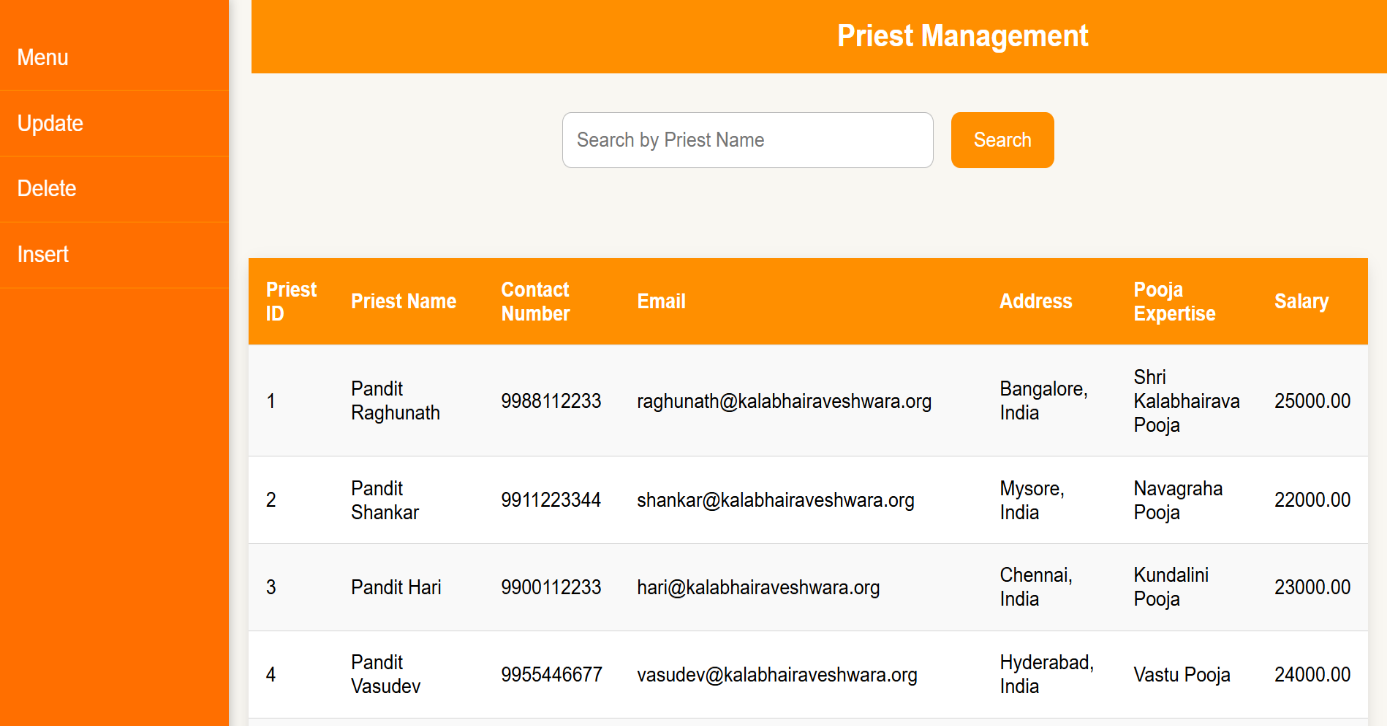
**Fig 6.3: Login Page**

****

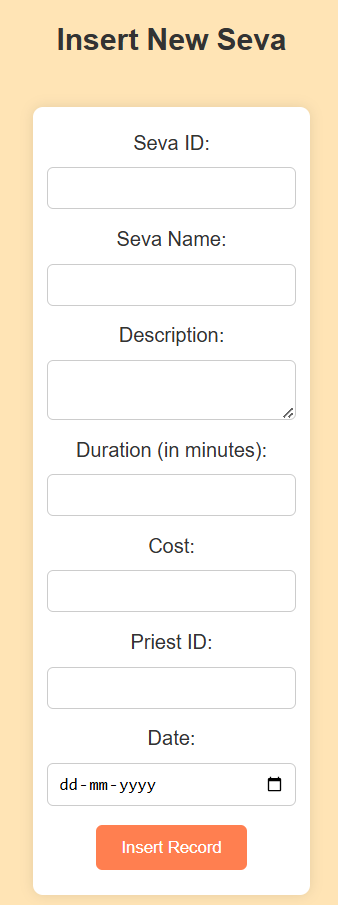
**Fig 6.4: Menu Page**



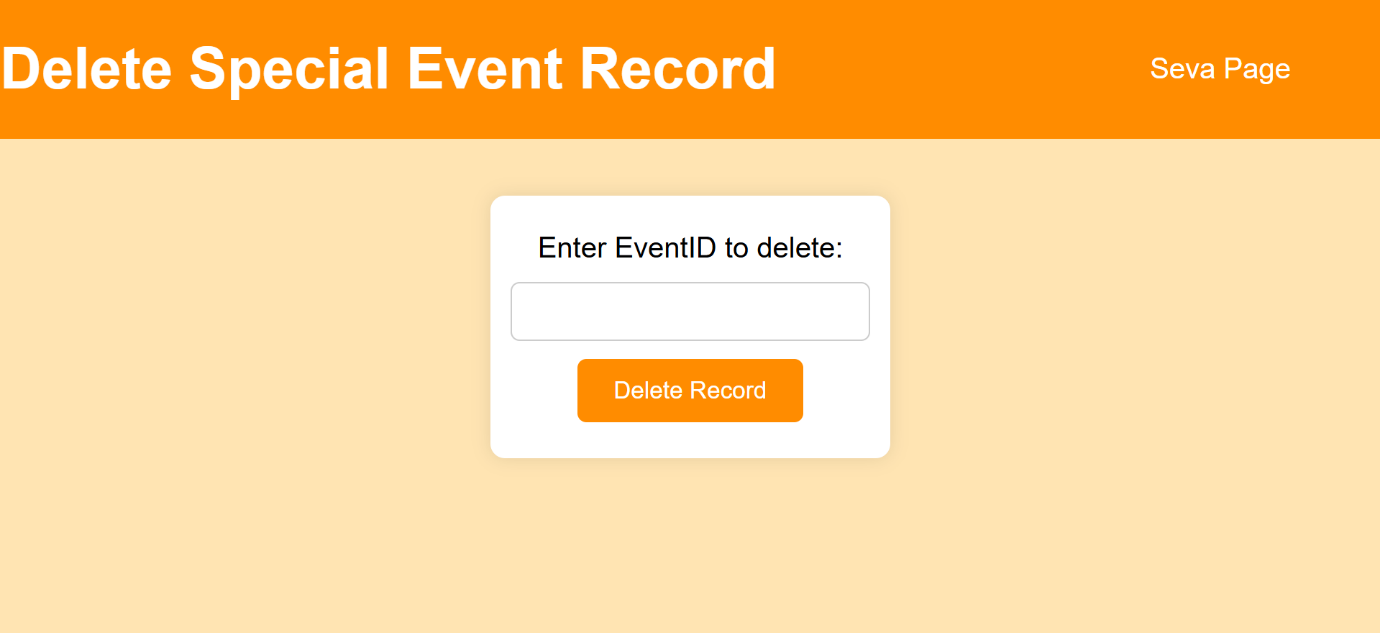
**Fig 6.5: Marriage Hall Details**

****

**Fig 6.6: Priests Details**



**Fig 6.7: Insert New Seva**

****

**Fig 6.8: Delete Special Event Record**

# CONCLUSION

In conclusion, the Temple Seva Management System offers an efficient, organized, and user- friendly solution to address the challenges of traditional manual processes. By integrating modules for managing marriage halls, special events, priest assignments, and seva services, the system ensures streamlined operations and accurate record-keeping. Role-based access enhances data security and accountability, while the centralized database eliminates redundancies and improves information retrieval. The system simplifies administrative tasks such as tracking bookings, organizing events, and assigning priests, making it easier for temple staff to manage day-to-day operations. It also enhances the experience for devotees by providing easy access to information and services, ensuring timely updates and smooth coordination. This modernized approach not only improves transparency and operational efficiency but also provides a scalable framework to accommodate future needs. Overall, the Temple Seva Management System serves as a valuable tool to foster better resource utilization and organized workflow in temple activities.

# FUTURE ENHANCEMENT

The Temple Seva Management System can be enhanced with several future features to improve its functionality and user experience. Integrating secure online payment gateways will enable devotees to book sevas, marriage halls, and events seamlessly from anywhere. Developing a mobile application for Android and iOS platforms will increase accessibility and allow real-time notifications. AI-based recommendations can personalize the experience by suggesting sevas and events based on devotees past preferences. Multi-language support will cater to a diverse user base, while live streaming of temple events and rituals will engage devotees unable to attend in person. An advanced analytics dashboard can provide administrators with insights into bookings, revenue, and attendance, aiding informed decision-making. A feedback system will allow devotees to share their experiences, fostering continuous improvement. The addition of inventory management for temple resources and a dedicated festival management module will streamline operations further. Social media integration can help share updates and announcements, broadening the temple's outreach. These enhancements will make the system more robust, efficient, and future-ready, ensuring a better experience for both administrators and devotees.

# REFERENCES

1. Database Management System by Prof. Date, A Kannan, S Swaminathan
2. Fundamentals of Database System, Rome Z Elmasari C And Shomkont B. Navathe, 7th Edition 2017, Pearson
3. "National Government Service Portal," [Online]. Available: [https://services.india.gov.in/service/detail/integrated-temple-management](https://services.india.gov.in/service/detail/integrated-temple-management%20systemand40itmsand41-tamil-nadu-1) [systemand40itmsand41-tamil-nadu-1](https://services.india.gov.in/service/detail/integrated-temple-management%20systemand40itmsand41-tamil-nadu-1) [Accessed 18 5 2023].
4. Microsoft, "Visual Studio Code: Code Editing. Redefined.," [Online]. Available: [https://code.visualstudio.com/.](https://code.visualstudio.com/) [Accessed 18 05 2023].
5. <http://php.net/>
6. https://www.<http://en.wikipedia.org/wiki/PHP>
7. W3 Schools (HTML, CSS, JS and PHP) – <https://www.w3schools.com/>
8. The MySQL Documentation – <https://dev.mysql.com/doc/>