SRI KRISHNA COLLEGE OF TECHNOLOGY
(An Autonomous Institution)
Approved by AICTE | Affiliated to Anna University Chennai|
Accredited by NBA - AICTE| Accredited by NAAC with 'A' Grade
KOVAIPUDUR, COIMBATORE 641042

HALL BOOKING SYSTEM

A PROJECT REPORT

Submitted by

SOUBARNIKA S	727822TUCS226
VAISHNAV S R	727822TUCS249
VAISHNAVI M	727822TUCS250
VISHMITHA K	727822TUCS254

In partial fulfilment for the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

JULY 2024

BONAFIDE CERTIFICATE

Certified that this project report **HALL BOOKING SYSTEM** is the bonafide work of **SOUBARNIKA S 727822TUCS226, VAISHNAV S R 727822TUCS249, VAISHNAVI M 727822TUCS250, VISHMITHA K 727822TUCS254** who carried out the project work under my supervision.

SIGNATURE	*SIGNATURE*
Ms. A. GOMATHY	**Dr. M. UDHAYAMOORTHI**
SUPERVISOR	**HEAD OF THE DEPARTMENT**
Assistant Professor,	Associate Professor,
Department of Computer Science and Engineering,	Department of Computer Science and Engineering,
Sri Krishna College of Technology,Coimbatore-641042	Sri Krishna College of Technology,Coimbatore-641042

Certified that the candidate was examined by me in the Project Work Viva Voce examination held on________________at Sri Krishna College of Technology, Coimbatore-641042.

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

First and foremost we thank the **Almighty** for being our light and for showering his gracious blessings throughout the course of this project.

We express our gratitude to our beloved Principal, **Dr. M.G. Sumithra,** for providing all facilities.

We are grateful to our beloved Head, Computing Sciences **Dr.T. Senthilnathan,** for her tireless and relentless support.

With the grateful heart, our sincere thanks to our Head of the Department **Dr. M. Udhayamoorthi**, Department of Computer Science and Engineering for the motivation and all support to complete the project work.

We thank **Ms. A. Gomathy**, Assistant Professor, Department of Computer Science and Engineering, for his motivation and support.

We are thankful to all the **Teaching and Non-Teaching Staff** of Department of Computer Science and Engineering and to all those who have directly and indirectly extended their help to us in completing this project work successfully.

We extend our sincere thanks to our **family members** and our beloved **friends**, who had been strongly supporting us in all our endeavour

ABSTRACT

The Hall Management System is an advanced and comprehensive web application meticulously crafted to streamline the booking and management of event spaces. By leveraging a sophisticated tech stack that includes React.js for a highly dynamic and responsive user interface, Tailwind CSS for modern, versatile, and customizable styling, Spring Boot for a robust and scalable backend, REST APIs for seamless and efficient communication, and MySQL for dependable and secure data management, the system is designed to elevate both user experience and operational efficiency. Users are provided with a powerful platform that allows them to explore and view a diverse range of halls, filtered by various criteria such as location, size, rating, and amenities. The ability to bookmark preferred halls and make reservations effortlessly enhances the convenience and functionality of the application. The system's intuitive and user-friendly design ensures accessibility for individuals with varying levels of technical expertise, while the backend infrastructure, supported by Spring Boot and REST APIs, guarantees reliable performance, scalability, and efficient data handling. MySQL underpins robust data management and security, ensuring the integrity and protection of user information. In conclusion, the Hall Management System represents a seamless integration of cutting-edge technologies and user-centric design principles, delivering a comprehensive and efficient solution for managing event spaces, ultimately simplifying the process of discovering and booking the ideal venue and significantly enhancing the overall user experience.

TABLE OF CONTENT

LIST OF FIGURES

LIST OF ABBREVIATIONS

Abbreviation	Acronym
VS Code	Visual Studio Code
IDE	Integrated Development Environment
JVM	Java Virtual Machine
SDLC	Software Development Lifecycle

CHAPTER 1

INTRODUCTION

This project aims to offer a seamless and efficient solution for users to manage hall bookings through an online platform. In this chapter, we will explore the problem statement, provide an overview, and outline the main objectives of the hall management system.

1.1 PROBLEM STATEMENT

How can we develop a Hall Management System that allows users to efficiently view, explore, bookmark, and reserve halls, considering factors such as location, size, rating, and amenities, while ensuring an intuitive user interface and minimizing booking conflicts?

1.2 OVERVIEW

In the realm of hall management, users often face challenges such as finding suitable halls based on specific criteria, managing bookings, and accessing accurate information about hall features. Traditional methods can be cumbersome, with issues such as outdated information, inefficient search processes, and difficulties in managing reservations. To address these challenges, we propose the development of a Hall Management System. This system will utilize modern web technologies to deliver a comprehensive, user-friendly platform for viewing, exploring, bookmarking, and reserving halls. By incorporating advanced search functionalities, detailed hall information, and intuitive booking features, the system aims to streamline the reservation process, reduce conflicts, and enhance overall user satisfaction.

1.3 OBJECTIVE

The objective of the Hall Management System is to develop a comprehensive, user-friendly platform that simplifies the booking process for events, meetings, and parties. The system aims to streamline user interactions by providing an intuitive interface for searching, reserving, and managing hall bookings. It focuses on real-time availability management to ensure accurate and up-to-date information, preventing scheduling conflicts and double bookings.

CHAPTER 2
SYSTEM SPECIFICATION

In this chapter, we are going to see the software that we have used to build the website. This chapter gives you a small description about the software used in the project.

2.1 VS CODE

Visual Studio Code is a source code editor developed by Microsoft for Windows, Linux, and macOS. It includes support for debugging, embedded Git control, syntax highlighting, intelligent code completion, snippets, and code refactoring. It is also customizable, so users can change the editor's theme, keyboard shortcuts, and preferences.

VS Code is an excellent code editor for React projects. It is lightweight, customizable, and has a wide range of features that make it ideal for React development. It has built-in support for JavaScript, JSX, and TypeScript, and enables developers to quickly move between files and view detailed type definitions. It also has a built-in terminal for running tasks. Additionally, VS Code has an extensive library of extensions that allow developers to quickly add features like code snippets, debugging tools, and linting supportto their projects.

2.2 LOCAL STORAGE

Local storage is a type of web storage for storing data on the client side of a web browser. It allows websites to store data on a user’s computer, which can then be accessed by the website again when the user returns. Local storage is a more secure alternative to cookies because it allows websites to store data without having to send it back and forth with each request. Local storage is a key-value pair storage mechanism, meaning it stores data in the form of a key and corresponding value. It is similar to a database table in that it stores data in columns and rows, except that local storage stores the data in the browser rather than in a database. Local storage is often used to store user information such as preferences and settings, or to store data that is not meant to be shared with other websites. It is also used to cache data to improve the performance of a website. Local storage is supported by all modern web browsers, including Chrome,

Firefox, Safari, and Edge. It is accessible through the browser's JavaScript API. Local storage is a powerful tool for websites to store data on the client side. It is secure, efficient, and can be used to store data that does not need to be shared with other websites.

Local Storage is a great way to improve the performance of a website by caching data. Local storage in web browsers allows website data to be stored locally on the user's computer. It is a way of persistently storing data on the client side, which is not sent to the server with each request. This allows users to store data such as preferences, login information, and form data without needing to send it to a server. It is typically stored in a browser's cookie file, but it can also be stored in other locations such as HTML5 Local Storage and Indexed DB. The data stored in local storage is persistent and can be accessed by the website even if the user closes the browser or navigates to another page. It is a great way for websites to store user-specific data, as it is secure, reliable, and fast. It is also a great way for developers to store data that does not need to be sent to the server with each request.

One of the key benefits of using local storage is its reliability. Unlike server-side storage, which can be affected by network outages or other server issues, local storage is stored locally on the user's machine, and so is not affected by these issues. Another advantage of local storage is its speed. Because the data is stored locally, it is accessed quickly, as there is no need to send requests to a server. This makes it ideal for storing data that needs to be accessed quickly, such as user preferences or session data. Local storage is also secure, as the data is stored on the user's machine and not on a server. This means that the data is not accessible by anyone other than the user, making it a good choice for storing sensitive information.

CHAPTER 3

PROPOSED SYSTEM

This chapter gives a small description about the proposed idea behind the development of our website

3.1 PROPOSED SYSTEM

The proposed Hall Booking System is designed to transform the way venues are reserved and managed. The front-end of the system will be developed using React with Vite, providing users with a modern, intuitive interface that simplifies the process of searching for available halls, viewing detailed information, and making reservations. The user experience will be enhanced by a responsive design, ensuring easy navigation and accessibility across various devices.

Users can bookmark their preferred halls, creating a personalized list of potential venues for future events. This feature is particularly useful for users planning multiple events or who wish to compare different options before making a final decision. The reservation process is made straightforward through an intuitive interface that allows users to select available dates and complete their bookings with minimal effort.

On the back-end, the system will utilize Spring Boot to handle real-time availability management, secure transactions, and data processing. This robust framework will ensure the system's scalability and reliability, allowing it to efficiently manage multiple concurrent bookings and maintain accurate records of hall availability. The integration of secure payment processing will facilitate seamless and protected transactions for users. MySQL will be employed as the database solution to store and manage all booking-related data, including user information, hall details, and reservation records. This choice ensures reliable data integrity and quick access to information, supporting the system’s operational efficiency and performance.

In summary, the Hall Management System provides a comprehensive and user-friendly solution for hall bookings. It simplifies the process for users by offering convenient search, bookmarking, and reservation features, while empowering hall owners with efficient request management tools. The integration of these functionalities ensures a smooth and effective booking experience, ultimately leading to increased satisfaction for all stakeholders involved.

3.2 ADVANTAGES

Efficiency: The Hall Management System allows users to quickly and easily search for, bookmark, and reserve halls from any location with internet access. This eliminates the need for time-consuming manual processes, reducing the administrative workload for hall owners and streamlining the booking process for users.

Flexibility: Users can explore a variety of halls, check availability, and make reservations at their convenience. This flexibility accommodates different event planning needs and preferences, improving user satisfaction and ensuring that hall bookings align with their schedules.

Conflict Resolution: The system automatically manages booking availability and prevents double-bookings by ensuring that halls are reserved only for available dates. This feature minimizes the chances of scheduling conflicts and ensures smooth operation.

Accessibility: Both users and hall owners can access the Hall Management System from any location with internet access. This accessibility facilitates remote management of bookings and reservations, enhancing coordination and planning without geographical constraints.

Transparency: The system provides clear and detailed information about hall availability, features, and reservations. This transparency helps users make informed decisions and allows hall owners to efficiently manage and track reservation requests, reducing confusion and improving communication.

Convenience: Users can effortlessly browse through a comprehensive list of available halls, select their desired options, and complete the booking process from anywhere with internet access. This eliminates the need for in-person visits to booking offices, saving time and effort for busy individuals and event organizers.

CHAPTER 4

METHODOLOGIES

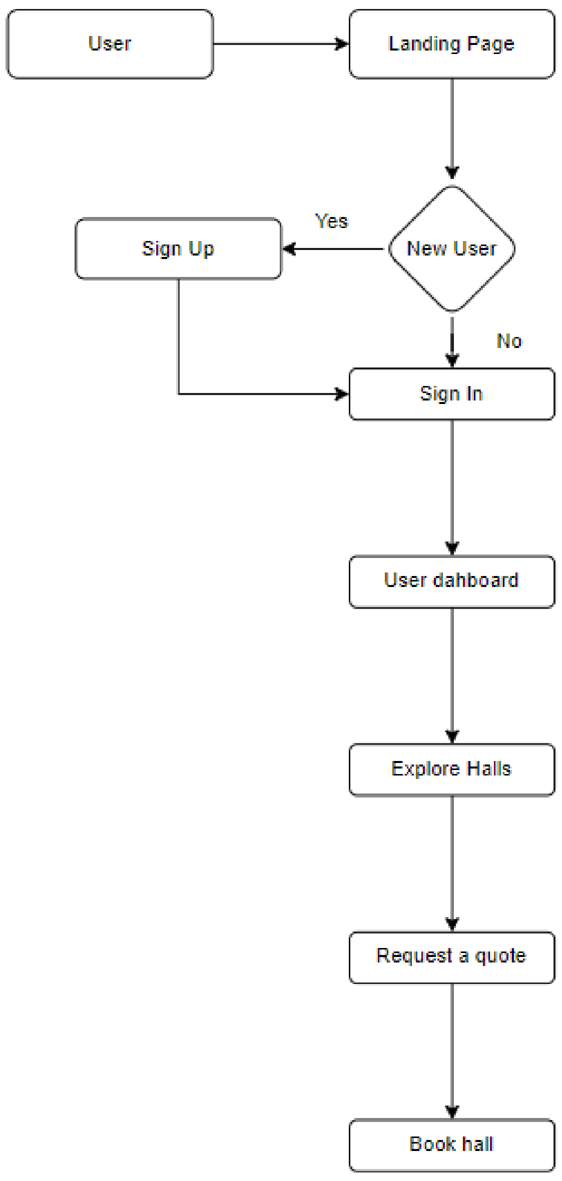


Fig 4.1.Process flow diagram

1. **User Registration:**

   Users can register on the platform by providing essential details such as name, email address, and password. Optional information like address and contact number may be included to facilitate communication and booking confirmations.

2. **Hall Exploration:**

   Upon registration/login, users can explore available halls by browsing through categories or using search functionality. Each hall listing includes details such as hall name, description, capacity, location, and availability.

3. **Booking Process:**

   Users can select their desired hall(s) and proceed to the booking process. During booking, users may need to provide additional information such as event details, booking duration, and special requirements.

4. **Payment Processing:**

   The platform securely processes exam fees using various payment methods, such as credit/debit cards, net banking, or digital wallets.

   Candidates receive a confirmation of successful payment upon completion.

5. **Booking Confirmation:**

   Upon successful payment, the platform generates a booking confirmation with a unique booking ID. The system notifies the hall administrators about the new booking for processing and preparation.

6. **Booking Review:**

   The hall administrators review the booking details and verify the information provided by the user. The system ensures that all requirements and terms are met before confirming the booking.

7. **Booking Confirmation Ticket:**

   After the booking is confirmed, users receive a booking confirmation ticket with details such as hall location, date, time, and any special instructions. The confirmation ticket serves as proof of booking and entry into the hall.

8. **Event Day:**

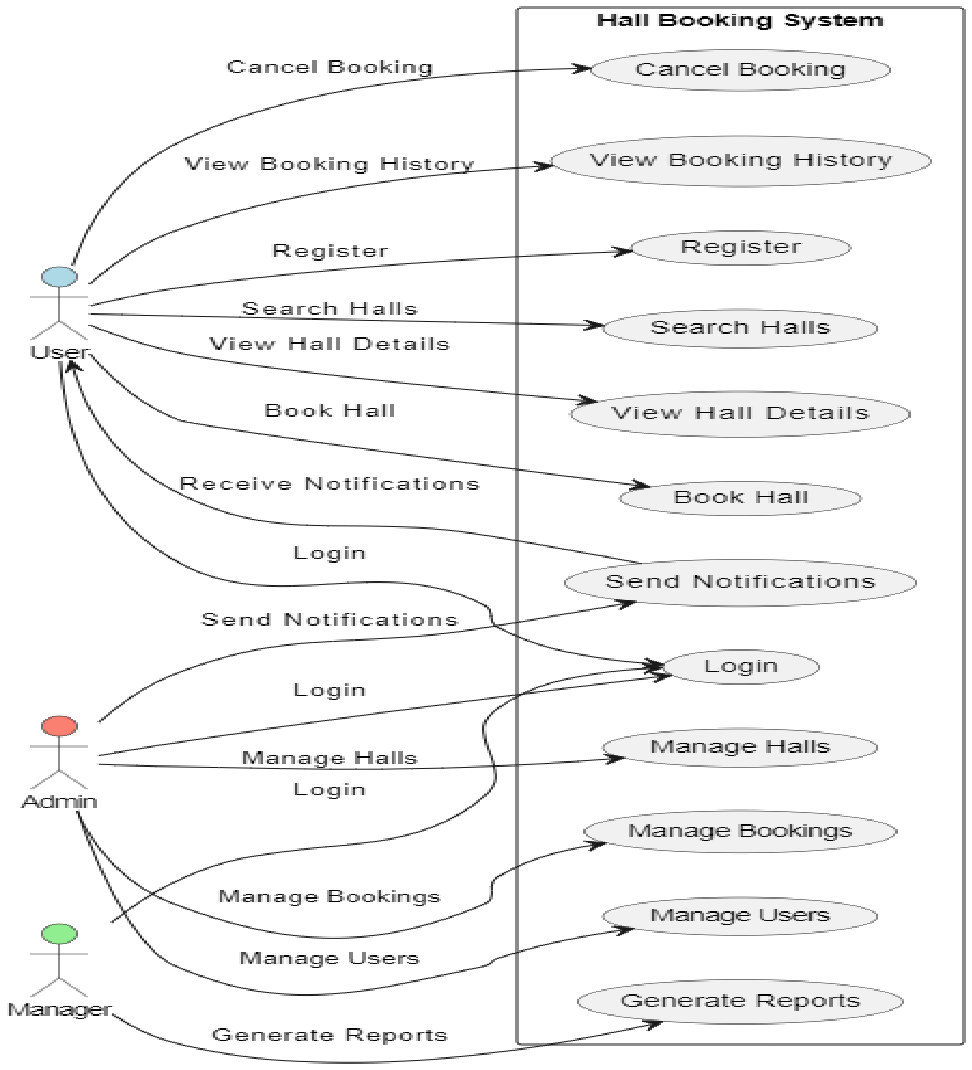
   On the event day, users present their confirmation ticket and identification for verification. Hall supervisors verify user identities and admit them into the hall.

9. **Post-Event Review:**

   After the event, users receive notifications about post-event procedures through the platform. The platform may also request feedback on the hall booking experience to ensure continued improvement and customer satisfaction.

Use Case Diagram:

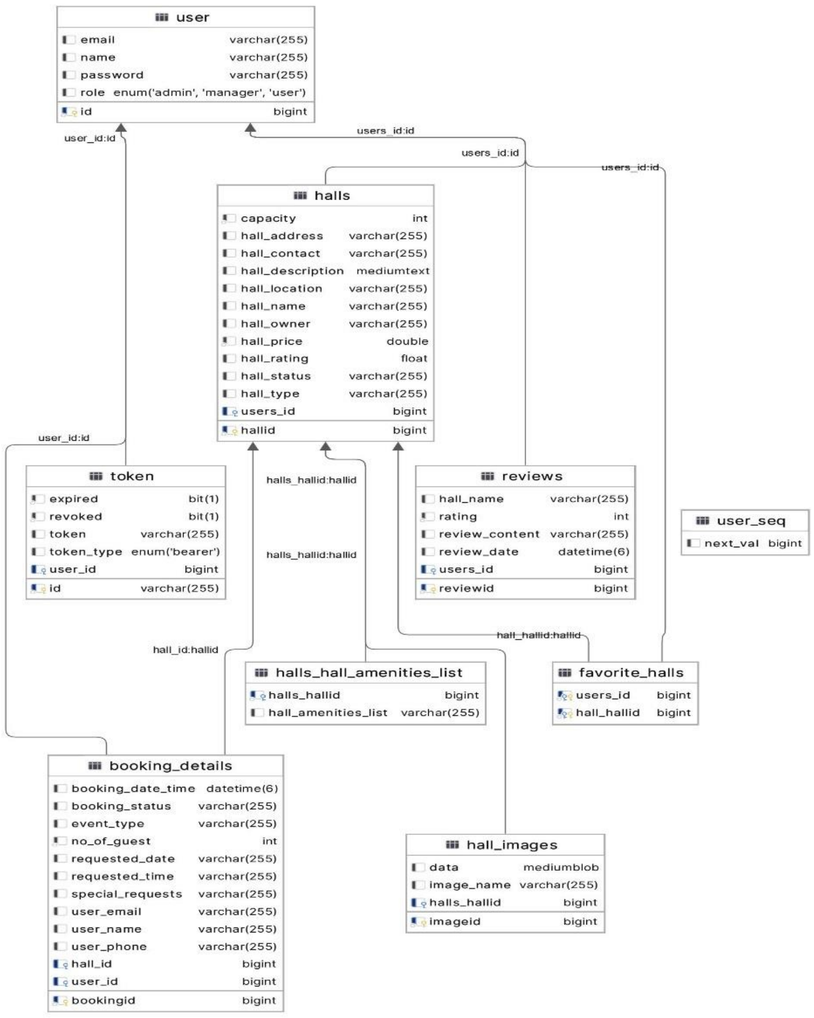
A use case diagram is a visual representation that illustrates the interactions between users and the Hall Management System. It captures the functional requirements of the system by detailing the various scenarios in which users engage with the system. In this diagram, different types of users, such as event organizers and hall owners, are depicted as actors interacting with the system's use cases. For instance, an event organizer may search for available halls, view detailed information, bookmark preferred options, and make reservations, while a hall owner manages reservation requests, approves or rejects bookings, and updates hall availability. The use case diagram provides a clear and structured overview of how different users interact with the system, helping to ensure that all functional requirements are considered and integrated into the design of the Hall Management System.



4.2 Use Case Diagram

Class Diagram:

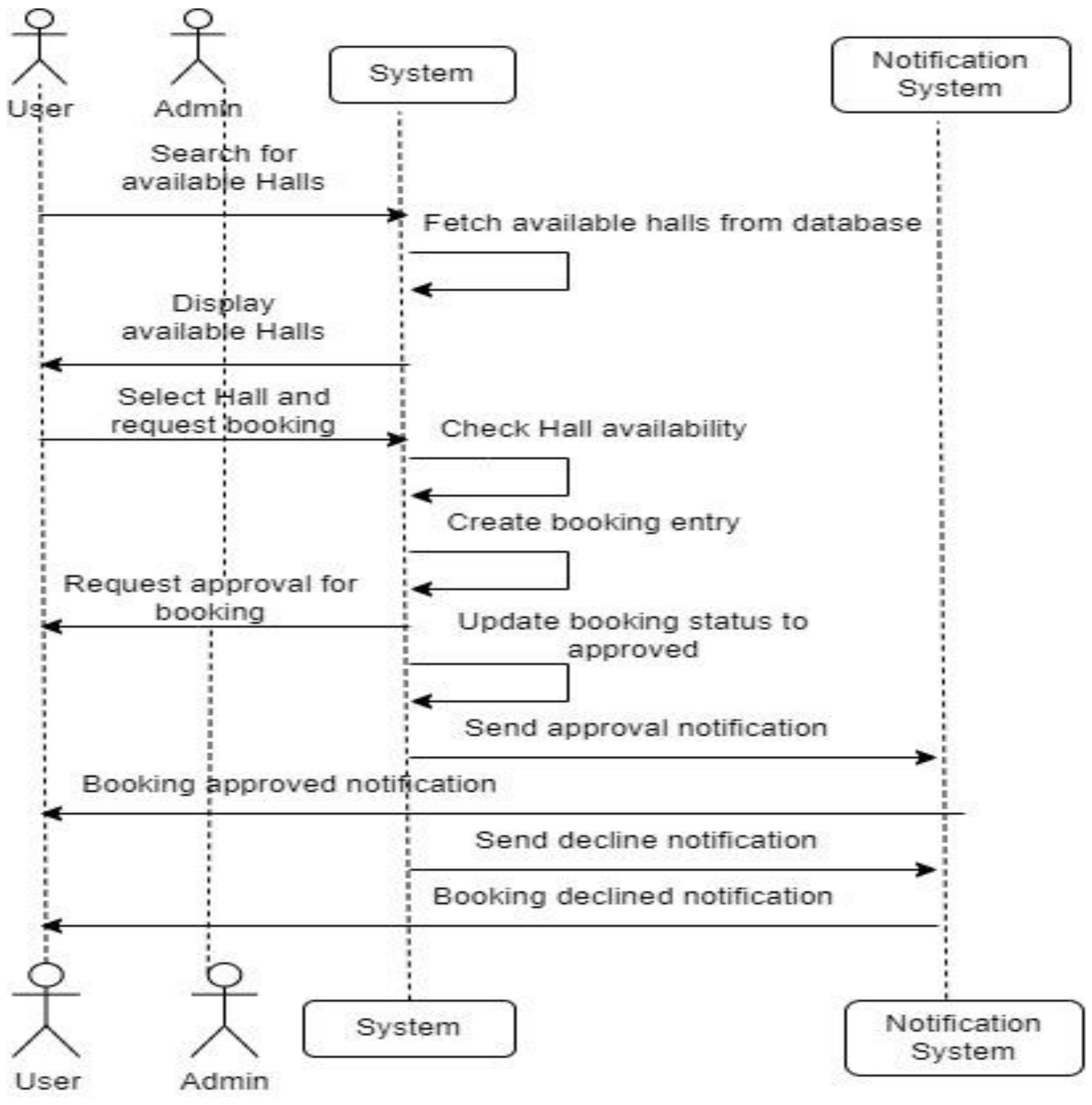
The class diagram for the Hall Management System visually represents the system’s structure and the interactions between various entities. Key classes include **BookingDetails**, which manages reservation information; Halls, which details venue attributes; and User, which captures user information and roles. Additionally, **FavoriteHalls** tracks user’s favorite venues, **Reviews** handles user feedback, **HallImages** manages images of halls, and **HallsHallAmenitiesList** lists amenities. The Token class deals with authentication. The diagram outlines how these classes are interconnected, providing a clear blueprint for the system’s architecture and development.



4.3 Class Diagram

Sequence Diagram:

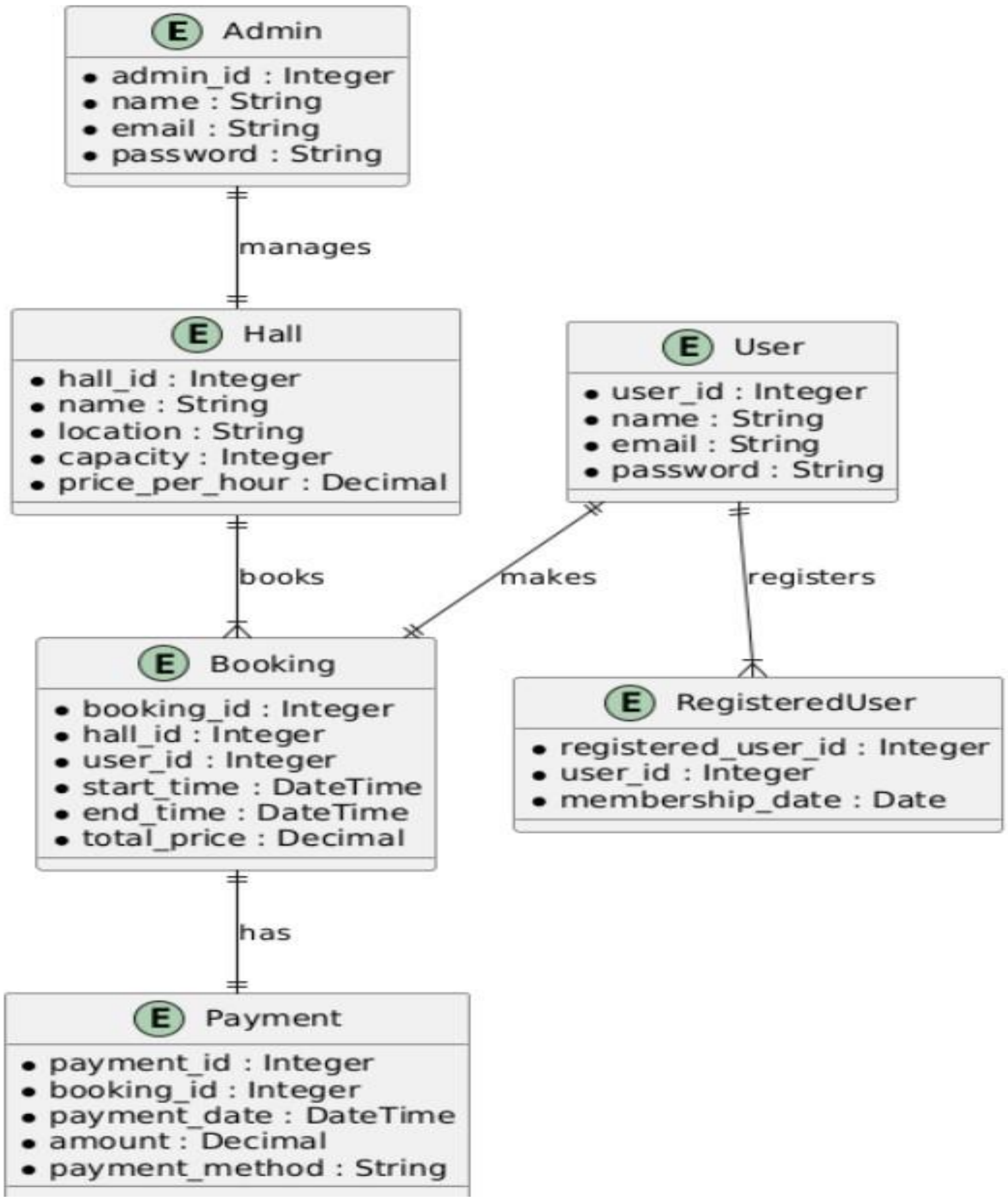
The Hall Booking Sequence Diagram illustrates the process for a user to request a hall booking. It details the interaction between the User, Hall Management System, and Hall Owner components. The diagram shows how the user searches for available halls, selects a preferred hall, and submits a booking request. The Hall Management System processes the request by checking hall availability and sending the booking details to the Hall Owner. The Hall Owner then reviews the request, approves or rejects it, and updates the system with the booking status. This sequence ensures that the user is informed of the booking outcome and that the hall reservation process is managed efficiently.


User
Admin
System
Notification System
Search for available Halls
Fetch available halls from database
Display available Halls
Select Hall and request booking
Check Hall availability
Create booking entry
Request approval for booking
Update booking status to approved
Send approval notification
Booking approved notification
Send decline notification
Booking declined notification
User
Admin
System
Notification System

4.4 Sequence Diagram

Entity Relationship Diagram:

The User entity represents individuals who book halls. Each user is uniquely identified by a UserID, and their personal information includes their Name, Email, Password, and optional details such as Address and ContactNumber.The Hall entity describes the halls available for booking. Each hall is identified by a unique HallID and includes attributes such as Name, Location, Capacity, and Amenities available in the hall.



4.5 ER Diagram

CHAPTER 5
IMPLEMENTATION AND RESULT

This chapter gives a description about the output that we produced by developing the website of our idea.

5.1 LANDING PAGE

When a user visits the landing page of our website, they are greeted with a welcoming interface that provides an overview of the available features. The landing page allows users to quickly access key functionalities, such as searching for available halls, exploring detailed information about various venues.

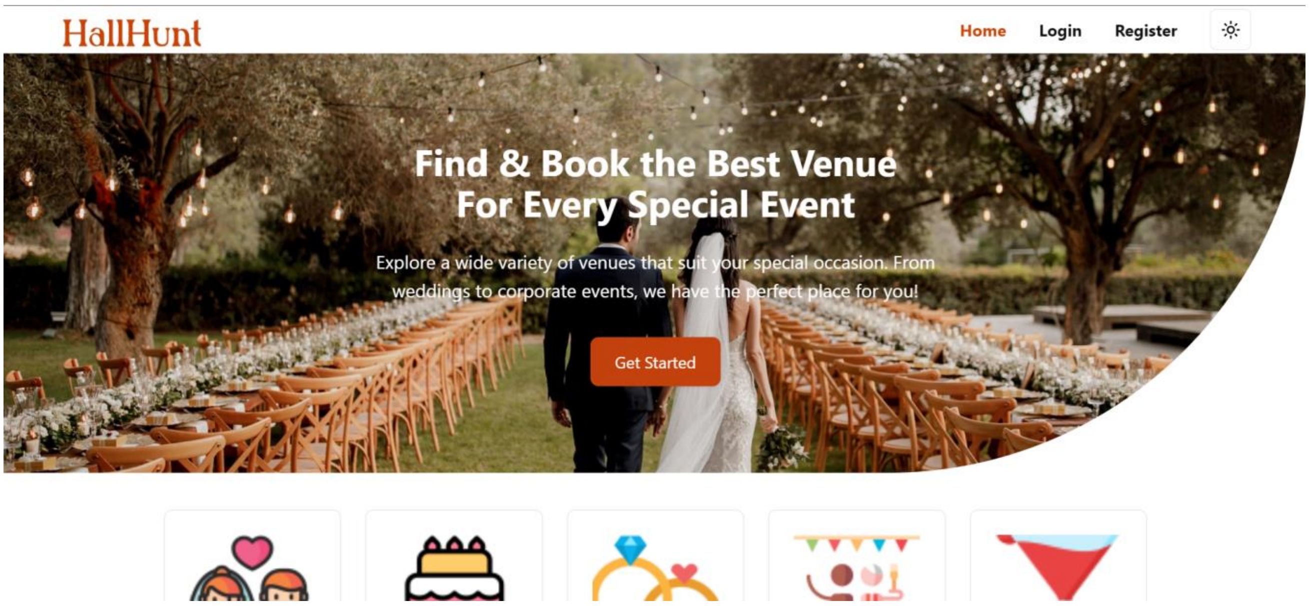
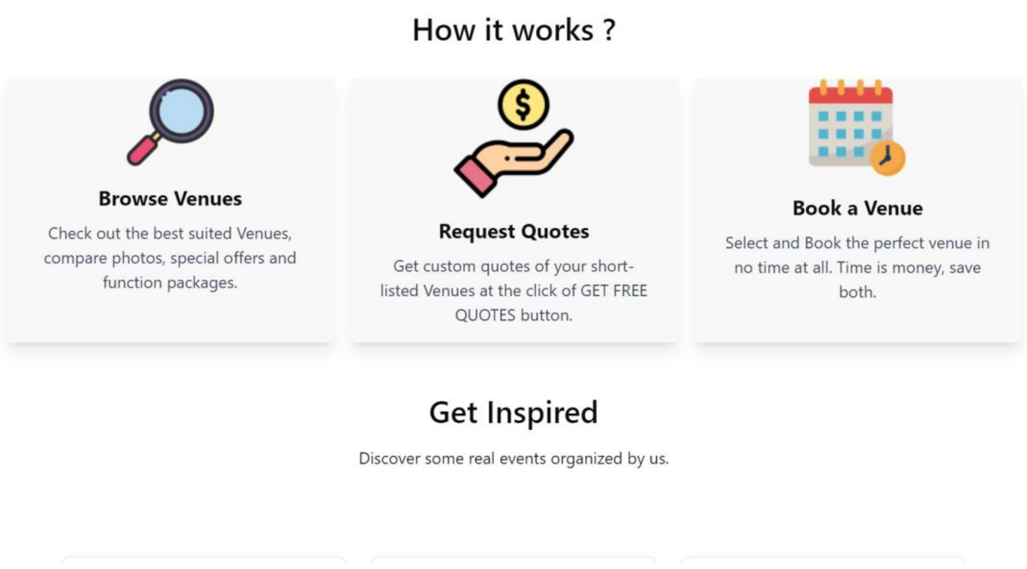

HallHunt
Home
Login
Register
Find & Book the Best Venue For Every Special Event
Explore a wide variety of venues that suit your special occasion. From weddings to corporate events, we have the perfect place for you!
Get Started

Fig 5.1 LANDING PAGE



How it works ?

Browse Venues

Check out the best suited Venues, compare photos, special offers and function packages.

Request Quotes

Get custom quotes of your short-listed Venues at the click of GET FREE QUOTES button.

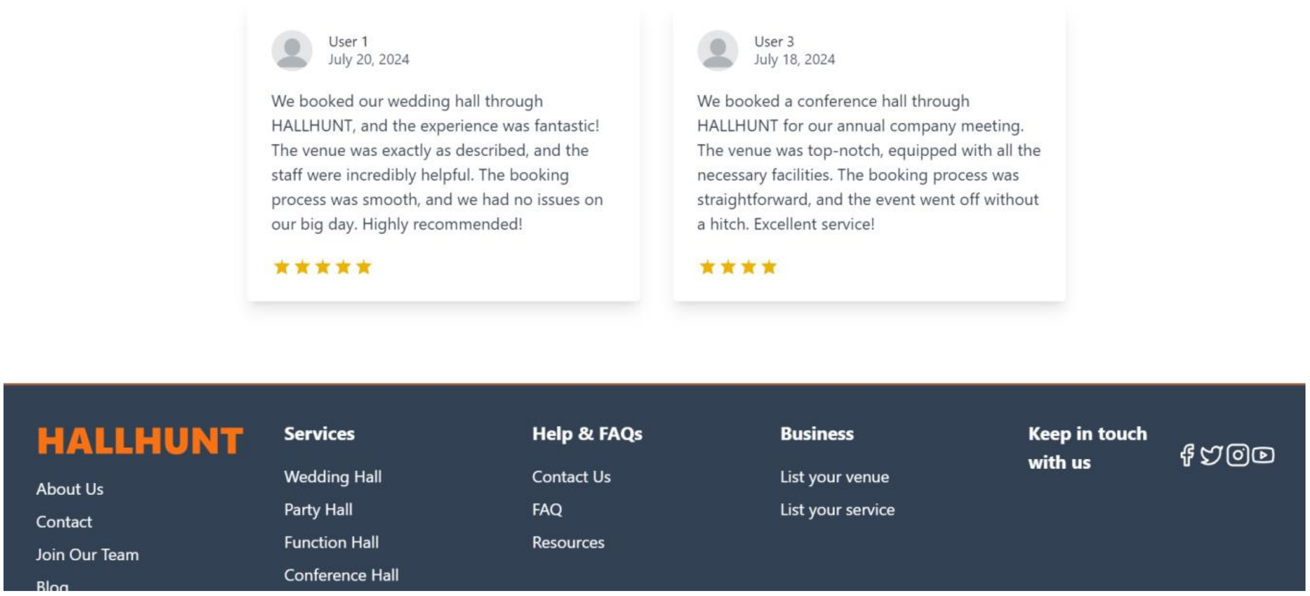
Book a Venue

Select and Book the perfect venue in no time at all. Time is money, save both.

Get Inspired

Discover some real events organized by us.

Fig 5.1 LANDING PAGE



User 1
July 20, 2024

We booked our wedding hall through HALLHUNT, and the experience was fantastic! The venue was exactly as described, and the staff were incredibly helpful. The booking process was smooth, and we had no issues on our big day. Highly recommended!

★★★★★

User 3
July 18, 2024

We booked a conference hall through HALLHUNT for our annual company meeting. The venue was top-notch, equipped with all the necessary facilities. The booking process was straightforward, and the event went off without a hitch. Excellent service!

★★★★

HALLHUNT

About Us
Contact
Join Our Team
Blog

Services
Wedding Hall
Party Hall
Function Hall
Conference Hall

Help & FAQs
Contact Us
FAQ
Resources

Business
List your venue
List your service

Keep in touch with us

Fig 5.1 LANDING PAGE

5.2 LOGIN

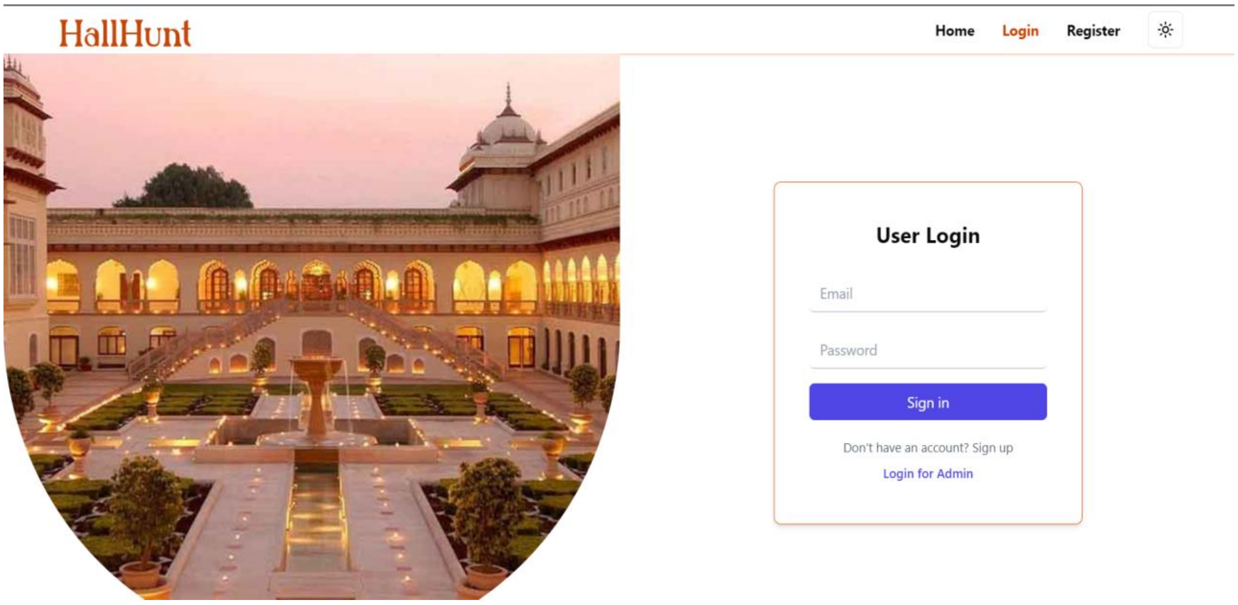


Fig 5.2 LOGIN PAGE

5.3 REGISTER PAGE

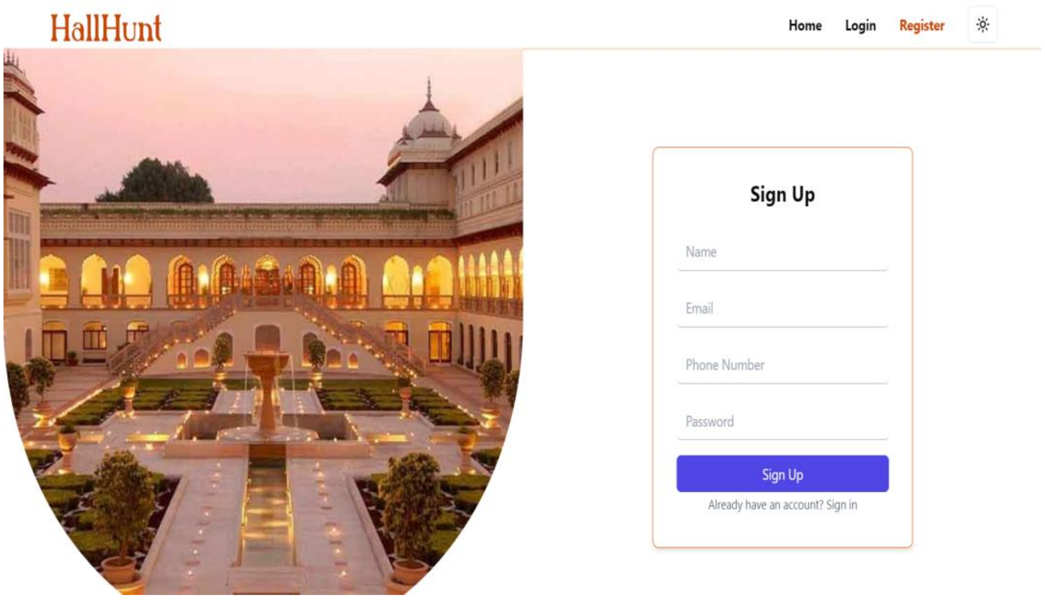
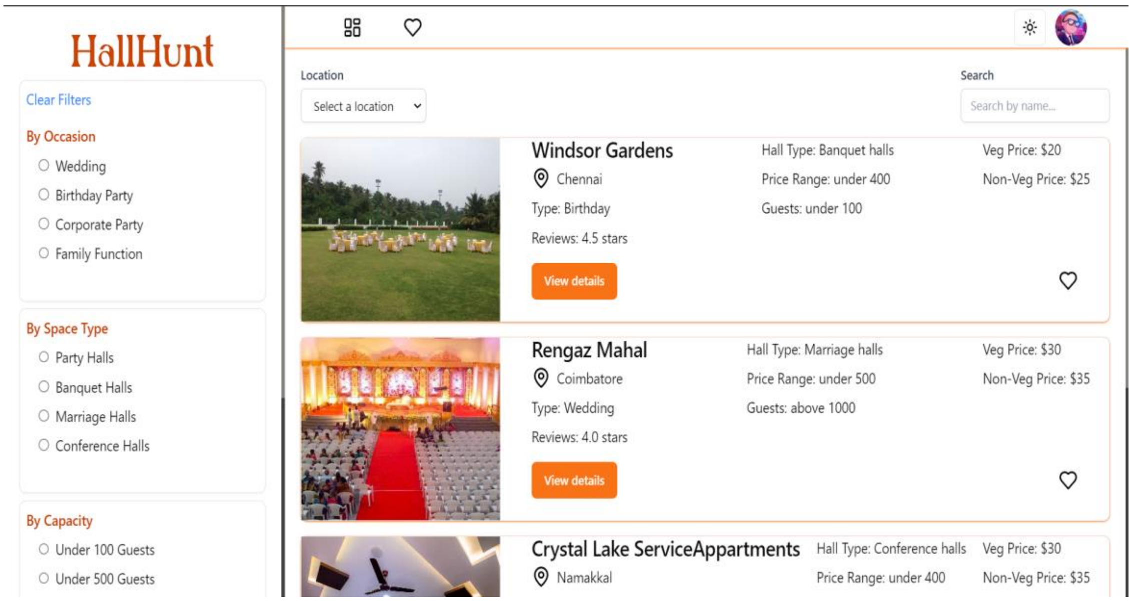


Fig 5.3 USER REGISTER

5.4 USER DASHBOARD


HallHunt

Clear Filters

By Occasion
- Wedding
- Birthday Party
- Corporate Party
- Family Function

By Space Type
- Party Halls
- Banquet Halls
- Marriage Halls
- Conference Halls

By Capacity
- Under 100 Guests
- Under 500 Guests

Location: Select a location

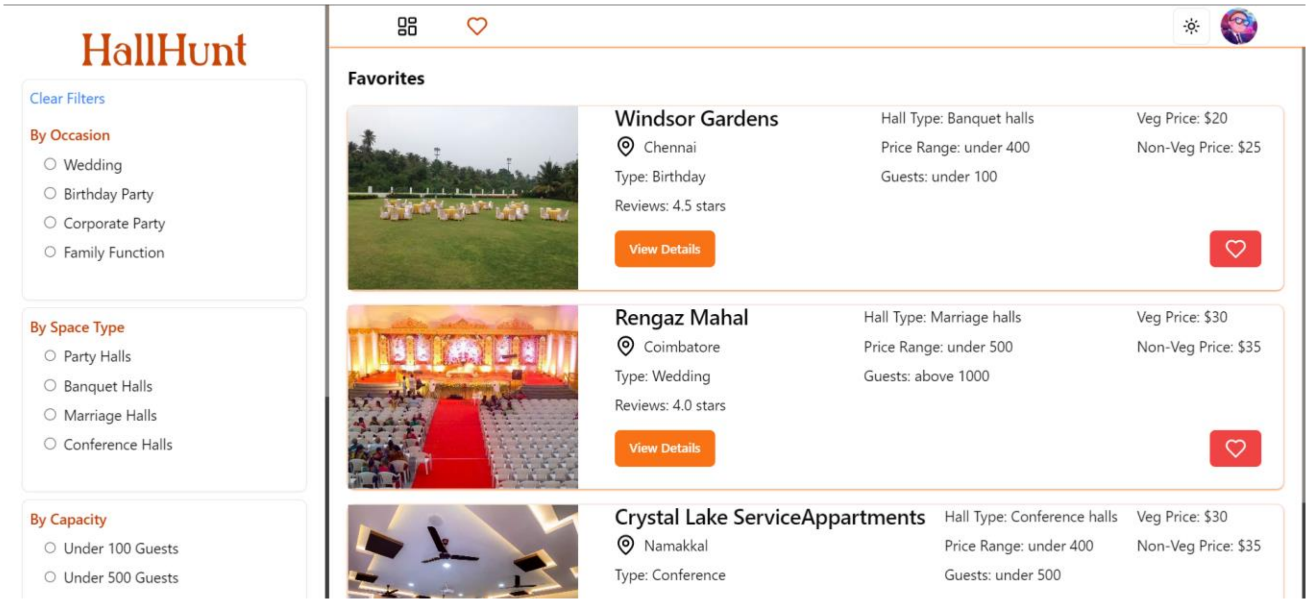
Search: Search by name...

Windsor Gardens
Chennai
Type: Birthday
Reviews: 4.5 stars
Hall Type: Banquet halls
Price Range: under 400
Guests: under 100
Veg Price: $20
Non-Veg Price: $25
View details

Rengaz Mahal
Coimbatore
Type: Wedding
Reviews: 4.0 stars
Hall Type: Marriage halls
Price Range: under 500
Guests: above 1000
Veg Price: $30
Non-Veg Price: $35
View details

Crystal Lake ServiceAppartments
Namakkal
Hall Type: Conference halls
Price Range: under 400
Veg Price: $30
Non-Veg Price: $35

Fig 5.4 USER DASHBOARD


HallHunt

Clear Filters

By Occasion
- Wedding
- Birthday Party
- Corporate Party
- Family Function

By Space Type
- Party Halls
- Banquet Halls
- Marriage Halls
- Conference Halls

By Capacity
- Under 100 Guests
- Under 500 Guests

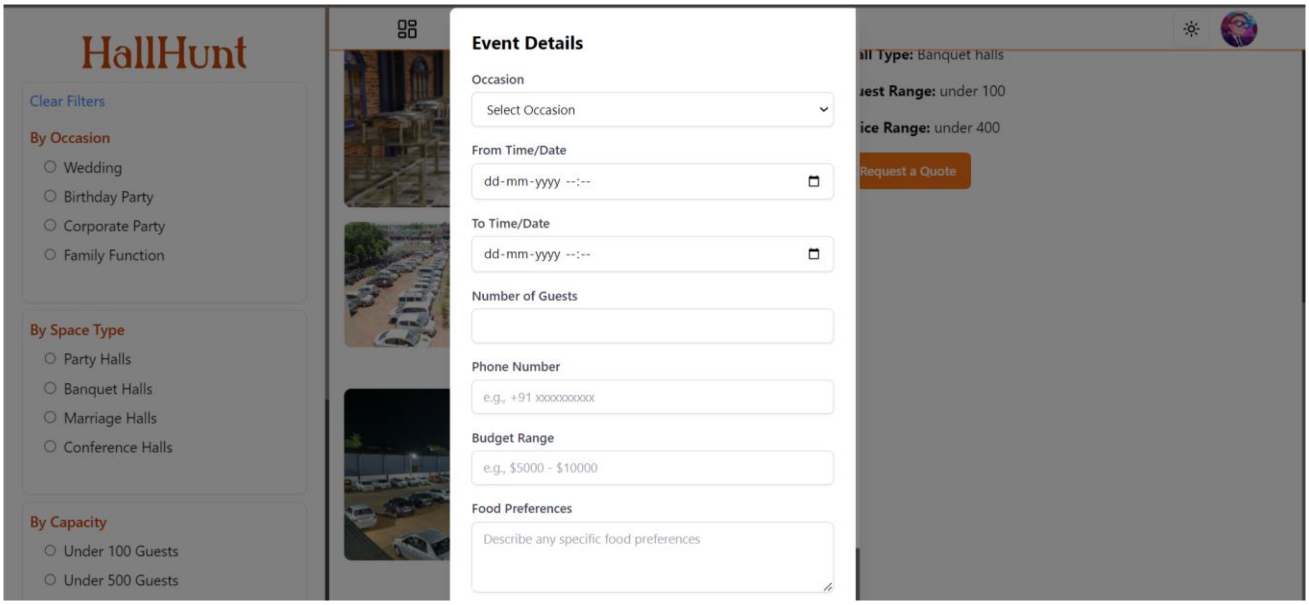
Favorites

Windsor Gardens
Chennai
Type: Birthday
Reviews: 4.5 stars
Hall Type: Banquet halls
Price Range: under 400
Guests: under 100
Veg Price: $20
Non-Veg Price: $25
View Details

Rengaz Mahal
Coimbatore
Type: Wedding
Reviews: 4.0 stars
Hall Type: Marriage halls
Price Range: under 500
Guests: above 1000
Veg Price: $30
Non-Veg Price: $35
View Details

Crystal Lake ServiceAppartments
Namakkal
Type: Conference
Hall Type: Conference halls
Price Range: under 400
Guests: under 500
Veg Price: $30
Non-Veg Price: $35

Fig 5.41 FAVORITES PAGE

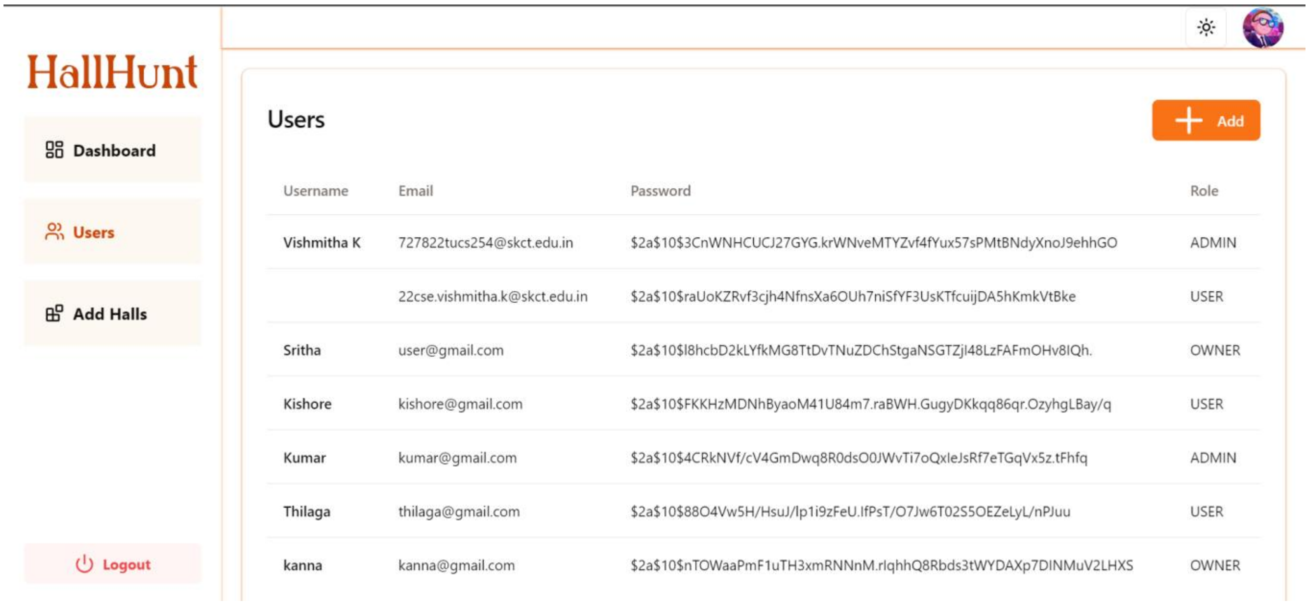


Event Details

Occasion	Select Occasion
From Time/Date	dd-mm-yyyy --:--
To Time/Date	dd-mm-yyyy --:--
Number of Guests	
Phone Number	e.g., +91 xxxxxxxxxx
Budget Range	e.g., $5000 - $10000
Food Preferences	Describe any specific food preferences

Fig 5.42 BOOKING FORM

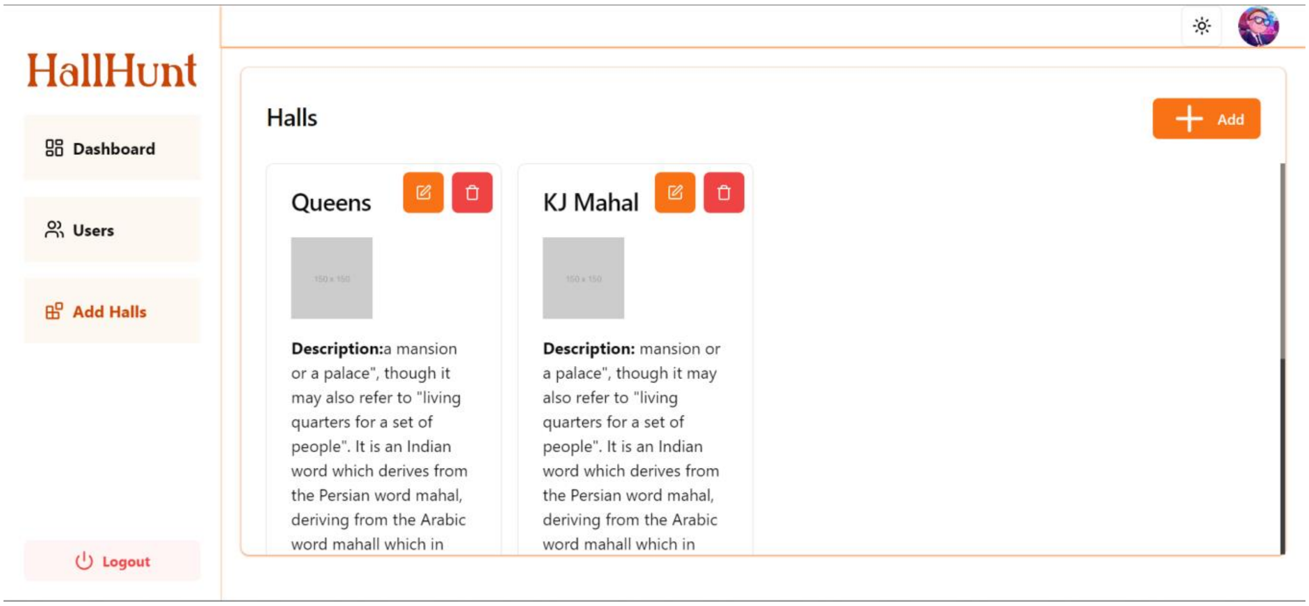
5.5 ADMIN DASHBOARD



Users

Username	Email	Password	Role
Vishmitha K	727822tucs254@skct.edu.in	$2a$10$3CnWNHCUCJ27GYG.krWNveMTYZvf4fYux57sPMtBNdyXnoJ9ehhGO	ADMIN
	22cse.vishmitha.k@skct.edu.in	$2a$10$raUoKZRvf3cjh4NfnsXa6OUh7niSfYF3UsKTfcuijDA5hKmkVtBke	USER
Sritha	user@gmail.com	$2a$10$l8hcbD2kLYfkMG8TtDvTNuZDChStgaNSGTZjl48LzFAFmOHv8IQh.	OWNER
Kishore	kishore@gmail.com	$2a$10$FKKHzMDNhByaoM41U84m7.raBWH.GugyDKkqq86qr.OzyhgLBay/q	USER
Kumar	kumar@gmail.com	$2a$10$4CRkNVf/cV4GmDwq8R0dsO0JWvTi7oQxleJsRf7eTGqVx5z.tFhfq	ADMIN
Thilaga	thilaga@gmail.com	$2a$10$88O4Vw5H/HsuJ/lp1i9zFeU.lfPsT/O7Jw6T02S5OEZeLyL/nPJuu	USER
kanna	kanna@gmail.com	$2a$10$nTOWaaPmF1uTH3xmRNNnM.rlqhhQ8Rbds3tWYDAXp7DINMuV2LHXS	OWNER

Fig 5.5 ADMIN DASHBOARD



Queens

Description:a mansion or a palace", though it may also refer to "living quarters for a set of people". It is an Indian word which derives from the Persian word mahal, deriving from the Arabic word mahall which in

KJ Mahal

Description: mansion or a palace", though it may also refer to "living quarters for a set of people". It is an Indian word which derives from the Persian word mahal, deriving from the Arabic word mahall which in

Fig 5.5 1 ADMINADDHALLS

5.6 CODING

CONFIG:

```
@Configuration
@RequiredArgsConstructor
public class ApplicationConfig {

    private final UserRepo userRepo;

    @Bean
    public UserDetailsService userDetailsService(){
        return  username -> userRepo.findByEmail(username)
                .orElseThrow(() -> new UsernameNotFoundException("User not Found"));
    }

    @Bean
    public AuthenticationProvider authenticationProvider(){
        DaoAuthenticationProvider authProvider = new DaoAuthenticationProvider();
        authProvider.setUserDetailsService(userDetailsService());
        authProvider.setPasswordEncoder(passwordEncoder());
        return authProvider;
    }

    @Bean
    public AuthenticationManager authenticationManager(AuthenticationConfiguration config)
throws Exception {
        return config.getAuthenticationManager();
    }

    @Bean
    public PasswordEncoder passwordEncoder() {
        return new BCryptPasswordEncoder();
    }
}

@Configuration
public class CorsConfig {

    @Bean
    public WebMvcConfigurer corsConfigurer() {
        return new WebMvcConfigurer() {
            @Override
            public void addCorsMappings(CorsRegistry registry) {
                registry.addMapping("/**")
                        .allowedOrigins("localhost:5173")
                        .allowedMethods("GET", "POST", "PUT", "DELETE", "OPTIONS")
                        .allowedHeaders("*")
                        .allowCredentials(true);
            }
        };
    }
}

@Component
@RequiredArgsConstructor
public class JwtAuthenticationFilter extends OncePerRequestFilter {
    private final JwtService jwtService;
    private final UserDetailsService userDetailsService;

    @Override
    protected void doFilterInternal(
            @NonNull HttpServletRequest request,
            @NonNull HttpServletResponse response,
            @NonNull FilterChain filterChain) throws ServletException, IOException {
        final String authHeader =request.getHeader("Authorization");
        final String jwtToken;
        final String userEmail;

        if(authHeader == null || !authHeader.startsWith("Bearer "))
        {
            filterChain.doFilter(request,response);
            return;
        }

        jwtToken = authHeader.substring(7);
        userEmail = jwtService.extractUserName(jwtToken);
        if(userEmail != null && SecurityContextHolder.getContext().getAuthentication() == null){
            UserDetails userDetails = this.userDetailsService.loadUserByUsername(userEmail);
            if(jwtService.isTokenValid(jwtToken, userDetails)){
                UsernamePasswordAuthenticationToken authToken = new
UsernamePasswordAuthenticationToken(
                        userDetails,
                        null,
                    userDetails.getAuthorities()
                );
                authToken.setDetails(
                    new WebAuthenticationDetailsSource().buildDetails(request)
                );
                SecurityContextHolder.getContext().setAuthentication(authToken);
            }
        }
        filterChain.doFilter(request,response); }}
@Service
public class JwtService {

    private static final String SECRET_KEY =
"EbeEsh7VhXpHMAkLz7Xb3TYm7a4KLMlYn0Kr1NJEhTIOeU9HJsv3t2bMa5OjoiaD";

    public String extractUserName(String token) {
        return extractClaim(token, Claims::getSubject);
    }

    public <T> T extractClaim(String token, Function<Claims, T> claimsResolver){
        final Claims claims  = extractAllClaims(token);
        return  claimsResolver.apply(claims);
    }

    public String generateToken(UserDetails userDetails){
        return generateToken(new HashMap<>(), userDetails);
    }

    public String generateToken(
            Map<String, Object> extraClaims,
            UserDetails userDetails
    ){
        return Jwts
```

```
            .builder()
            .claims(extraClaims).
            subject(userDetails.getUsername())
            .issuedAt(new Date(System.currentTimeMillis()))
            .expiration(new Date(System.currentTimeMillis() + 1000 * 60 * 24))
            .signWith(getSignInKey(), SignatureAlgorithm.HS256)
            .compact();

}

public Boolean isTokenValid(String token ,UserDetails userDetails)
{
    final  String username = extractUserName(token);
    return  (username.equals(userDetails.getUsername()) && !isTokenExpired(token));
}

private boolean isTokenExpired(String token) {
    return extractExpiration(token).before(new Date());
}

private Date extractExpiration(String token) {
    return extractClaim(token, Claims::getExpiration);
}

private Claims extractAllClaims(String token)
{
    return Jwts
            .parser()
            .setSigningKey(getSignInKey())
            .build()
            .parseClaimsJws(token)
            .getBody();
}
```

```
    private Key getSignInKey() {
        byte[] keyByte = Decoders.BASE64.decode(SECRET_KEY);
        return Keys.hmacShaKeyFor(keyByte);
    }
}

@Configuration
public class LogoutConfiguration {

    @Bean
    public CustomLogoutHandler logoutHandler(TokenRepo tokenRepo, JwtService jwtService)
{
        return new CustomLogoutHandler(tokenRepo, jwtService);
    }

    @Bean
    public LogoutSuccessHandler logoutSuccessHandler() {
        return new CustomLogoutSuccessHandler();
    }
}
```

HOME MODEL:

```
package com.security.template.model;

import java.util.List;

import com.fasterxml.jackson.annotation.JsonBackReference;
import com.fasterxml.jackson.annotation.JsonIdentityInfo;
import com.fasterxml.jackson.annotation.JsonIgnore;

import com.fasterxml.jackson.annotation.ObjectIdGenerators;
import jakarta.persistence.*;
```

```
@Entity

public class Hall {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String hallname;
    private String location;
    private String description;
    private String type;
    private String halltype;

    private String organiser;
    private String contact;

    @OneToMany(mappedBy = "halls")
    @JsonIgnore
//    @JsonBackReference
    private List<Booking> bookings;

    @Embedded
    private HallDetails hallDetails;

//    @OneToMany(mappedBy = "halls")
//    @JsonIgnore
//    private List<HallImages> hallImages;



    @ManyToOne
    @JoinColumn(name = "user_id")
    @JsonBackReference
    private User user;
    public Hall() {}

    public Hall(Long id, String hallname, String location, String description, String type, String
halltype, String organiser, String contact, List<Booking> bookings, HallDetails hallDetails, User
user) {
        this.id = id;
        this.hallname = hallname;
        this.location = location;
        this.description = description;
        this.type = type;
        this.halltype = halltype;
        this.organiser = organiser;
        this.contact = contact;
        this.bookings = bookings;
        this.hallDetails = hallDetails;
        this.user = user;
    }

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public String getHallname() {
        return hallname;
    }

    public void setHallname(String hallname) {
        this.hallname =hallname;
```

```
}


public String getLocation() {
   return location;
}

public void setLocation(String location) {
   this.location = location;
}

public String getDescription() {
   return description;
}

public void setDescription(String description) {
   this.description = description;
}

public String gettype() {
   return type;
}

public void settype(String type) {
   this.type = type;
}

public String getHalltype() {
   return halltype;
}

public void setHalltype(String halltype) {
   this.halltype = halltype;
```

```
}



public String getOrganiser() {
    return organiser;
}

public void setOrganiser(String organiser) {
    this.organiser = organiser;
}

public String getContact() {
    return contact;
}

public void setContact(String contact) {
    this.contact = contact;
}

public List<Booking> getBookings() {
    return bookings;
}

public void setBookings(List<Booking> bookings) {
    this.bookings = bookings;
}

public HallDetails getHallDetails() {
    return hallDetails;
}

public void setHallDetails(HallDetails hallDetails) {
        this.hallDetails = hallDetails;
    }

    public User getUser() {
        return user;
    }

    public void setUser(User user) {
        this.user = user;
    }
}
```

LANDING PAGE:

```
const carouselItems = [
  {
    id: 1,
    imgSrc: 'https://cdn.venuelook.com/images/new-home-images/wedding.png',
    text:'Wedding',
    alt: 'slide1',
  },
  {
    id: 2,
    imgSrc: 'https://cdn.venuelook.com/images/new-home-images/cake.png',
    text:'Birthday',
    alt: 'slide2',
  },
  {
    id: 3,
    imgSrc: 'https://cdn.venuelook.com/images/new-home-images/ring.png',
    text:'Engagement',
    alt: 'slide3',
  },
  {
    id: 4,
    imgSrc: 'https://cdn.venuelook.com/images/new-home-images/pool.png',
    text:'Pool party',
    alt: 'slide4',
  },
  {
    id: 5,
    imgSrc: 'https://cdn.venuelook.com/images/new-home-images/cocktail.png',
    text:'Cocktail Party',
    alt: 'slide5',
  },
  {
    id: 6,
    imgSrc: 'https://cdn.venuelook.com/images/new-home-images/ofc-party.png',
    text:'Corporate Party',
    alt: 'slide6',
  },
  {
    id: 7,
    imgSrc: 'https://cdn.venuelook.com/images/new-home-images/banquet.png',
    text:'Banquet Halls',
    alt: 'slide7',
  },
  {
    id: 8,
    imgSrc: 'https://cdn.venuelook.com/images/new-home-images/kitty.png',
    text:'Kitty Party',
    alt: 'slide8',
  },

];


const carouselItemsvenue = [
  {
    id: 1,
    img:
'https://cdn.venuelook.com/uploads/albums/album_145/thumb/photo_145_168139203169
2552_fix_480.jpg',
    head:'Birthday celebration',
    tail:'Selvam Mahaal at Chennai',
    alt: 'slide1',
  },
  {
    id: 2,
    img:
'https://cdn.venuelook.com/uploads/albums/album_148/thumb/photo_148_168533943359
4665_fix_480.jpg',
    head:'Team Outing',
    tail:'At Rishikesh',
    alt: 'slide2',
  },
  {
    id: 3,
    img:
'https://www.conferencerental.com/media/zoo/images/CorporateMeetings_9e625fab6858
4fa202a53e8b49ebb2b7.JPG',
    head:'Corporate meeting',
    tail:'VK Residency at Coimbatore',
    alt: 'slide3',
  },

];


const reviews = [
  {
    name: 'User 1',
    date: 'July 20, 2024',
    rev: 'We booked our wedding hall through HALLHUNT, and the experience was
fantastic! The venue was exactly as described, and the staff were incredibly helpful. The
booking process was smooth, and we had no issues on our big day. Highly
recommended!',
    rating: 5,
    avatar: 'https://cdn.pixabay.com/photo/2015/10/05/22/37/blank-profile-picture-
973460_640.png',
  },
  {
    name: 'User 2',
    date: 'July 18, 2024',
    rev: 'The party hall we booked was fantastic, with great amenities and a friendly staff.
The only downside was the slightly higher price compared to other venues. Nonetheless,
it was worth it for the quality and convenience.',
    rating: 4,
    avatar: 'https://cdn.pixabay.com/photo/2015/10/05/22/37/blank-profile-picture-
973460_640.png',
  },
  {
    name: 'User 3',
    date: 'July 18, 2024',
    rev: 'We booked a conference hall through HALLHUNT for our annual company
meeting. The venue was top-notch, equipped with all the necessary facilities. The
booking process was straightforward, and the event went off without a hitch. Excellent
service!',
    rating: 4,
    avatar: 'https://cdn.pixabay.com/photo/2015/10/05/22/37/blank-profile-picture-
973460_640.png',
  },
  {
    name: 'User 4',
    date: 'July 18, 2024',
    rev: 'The function hall we chose was ideal for our event, with ample space and good
facilities. The only improvement could be better signage to find the venue more easily.
Other than that, we were very satisfied with our booking.',
    rating: 4,
    avatar: 'https://cdn.pixabay.com/photo/2015/10/05/22/37/blank-profile-picture-
973460_640.png',
  },

];
const Home = () => {

  const [searchResults, setSearchResults] = useState(null);

  const handleSearch = (searchParams) => {

    console.log('Search parameters:', searchParams);

    setSearchResults([]);
  };

  const [query, setQuery] = useState('');
  const [filteredHalls, setFilteredHalls] = useState([]);
  const [searched, setSearched] = useState(false);

  const handleInputChange = (e) => {
    const value = e.target.value;
    setQuery(value);
    filterHalls(value);
    setSearched(value.trim() !== '');
  };

  const filterHalls = (query) => {
```

```
  const results = halls.filter(hall =>
    hall.name.toLowerCase().includes(query.toLowerCase()) ||
    hall.location.toLowerCase().includes(query.toLowerCase())
  );
  setFilteredHalls(results);
};

const [currentIndex, setCurrentIndex] = useState(0);

useEffect(() => {
  const interval = setInterval(() => {
    setCurrentIndex((prevIndex) => (prevIndex + 1) % reviews.length);
  }, 5000);
  return () => clearInterval(interval);
}, []);

const review = reviews[currentIndex];



const [currentIndi, setCurrentIndi] = useState(0);



useEffect(() => {
  const intervals = setInterval(() => {
    setCurrentIndi((prevIndex) => (prevIndex + 2) % reviews.length);
  }, 4000);
  return () => clearInterval(intervals);
}, []);

const reviee = reviews[currentIndi];
return (
  <>
```

```
      <div className="relative w-full h-[70vh] flex items-center justify-center">
  <img
                                src="https://www.elizabethanne-weddings.com/wp-
content/uploads/2023/02/louisgabriel-367-scaled.jpg"
    alt="Banner Image"
    className="w-full h-full object-cover bg-opacity-5 rounded-ee-full"
  />
  <div className="absolute container mx-auto flex flex-col items-center justify-center
text-center px-4">
    <h1 className="text-2xl md:text-4xl font-bold mb-4 md:mb-6 text-white">
      Find & Book the Best Venue
      <span className="block">For Every Special Event</span>
    </h1>
    <p className="text-sm md:text-lg mb-6 md:mb-8 text-white">
      Explore a wide variety of venues that suit your special occasion. From
      <span className="block">weddings to corporate events, we have the perfect place for
you!</span>
    </p>
    <NavLink to="/login">
      <button className="px-4 py-2 md:px-6 md:py-3 bg-orange-700 text-white rounded-
lg shadow hover:bg-primary">
        Get Started
      </button>
    </NavLink>
  </div>
</div>




    <div className='w-full flex justify-center items-start p-8'>
        <Carousel
```

```
          opts={{
            align: "start",
          }}
          className="w-[80%] h-auto"
        >
          <CarouselContent>
            {carouselItems.map((item) => (
              <CarouselItem key={item.id} className="basis-1/5">
                <div className="p-1">
                  <Card>
                    <CardContent className="flex aspect-square items-center justify-center p-
6">
                      <img src={item.imgSrc} alt={item.alt} className="w-full h-full object-
cover" />
                    </CardContent>
                    {/* <div className='w-full'>
                          <p className="mt-2 text-lg font-normal text-center pb-5  cursor-
pointer">{item.text}</p>
                      </div> */}
                          <p className="mt-2 text-lg font-normal text-center pb-5  cursor-
pointer">{item.text}</p>
                  </Card>
                </div>


              </CarouselItem>
            ))}
          </CarouselContent>
          <CarouselPrevious />
          <CarouselNext />
        </Carousel>
      </div>
      <hr className='border-solid p-5'/>
    <div className='w-full flex justify-center items-center text-3xl font-semibold'>
```

```
      <h6>How it works ?</h6>
    </div>

    <div className='w-full p-8'>
  <div className="flex justify-center space-x-4">

    <div className="max-w-xs rounded overflow-hidden shadow-lg bg-neutral-500 bg-
opacity-5">
      <div className='w-full flex items-center justify-center'>
        <img className="flex aspect-square" src="https://cdn.venuelook.com/images/new-
home-images/search.png" alt="Card image"/>
      </div>
      <div className="px-6 py-4">
        <div className="font-bold text-xl mb-2 text-center">Browse Venues</div>
        <p className="text-gray-700 text-base text-center">
          Check out the best suited Venues, compare photos, special offers and function
packages.
        </p>
      </div>
    </div>



    <div className="max-w-xs rounded overflow-hidden shadow-lg bg-neutral-500 bg-
opacity-5">
      <div className='w-full flex items-center justify-center'>
        <img className="flex aspect-square" src="https://cdn.venuelook.com/images/new-
home-images/rupee.png" alt="Card image"/>
      </div>
      <div className="px-6 py-4">
        <div className="font-bold text-xl mb-2 text-center">Request Quotes</div>
        <p className="text-gray-700 text-base text-center">
        Get custom quotes of your short-listed Venues at the click of GET FREE QUOTES
button.
```

```
        </p>
       </div>
      </div>



     <div className="max-w-xs rounded overflow-hidden shadow-lg bg-neutral-500 bg-
opacity-5">
       <div className='w-full flex items-center justify-center'>
         <img className="flex aspect-square" src="https://cdn.venuelook.com/images/new-
home-images/calender.png" alt="Card image"/>
       </div>
       <div className="px-6 py-4">
        <div className="font-bold text-xl mb-2 text-center">Book a Venue</div>
        <p className="text-gray-700 text-base text-center">
        Select and Book the perfect venue in no time at all. Time is money, save both.
        </p>
       </div>
      </div>
    </div>
</div>

<div >
      <h6 className='w-full flex justify-center items-center text-3xl font-semibold p-4'>Get
Inspired</h6>
        <p className='w-full flex justify-center items-center'>Discover some real events
organized by us.</p>
    </div>

    <div className='h-full w-full flex justify-center items-center pb-14'>
        <Carousel
          opts={{
            align: "start",
          }}
          className="w-[70%] h-auto"
        >
          <CarouselContent>
            {carouselItemsvenue.map((thing) => (
              <CarouselItem key={thing.id} className="basis-1/3">
                <div className="p-1">
                  <Card>
                    <CardContent className="flex aspect-square items-center justify-center p-
6">
                    <img src={thing.img} alt={thing.alt} className="w-full h-full object-cover
rounded" />
                   </CardContent>
                    <p className="mt-2 text-xl font-semibold text-center">{thing.head}</p>
                   <p className="mt-1 text-sm font-normal text-center pb-5">{thing.tail}</p>
                  </Card>
                </div>
              </CarouselItem>
            ))}
          </CarouselContent>
          <CarouselPrevious />
          <CarouselNext />
        </Carousel>
      </div>
      <div className="flex flex-col items-center  pb-20">
      <h1 className="text-2xl font-bold mb-8 text-gray-900 dark:text-gray-100">Customer
Reviews</h1>
      <div className='flex flex-row items-center gap-8'>
         <div className="max-w-sm rounded overflow-hidden shadow-lg p-6 bg-white
dark:bg-gray-800">
        <div className="flex items-center mb-4">
          <img className="w-10 h-10 rounded-full mr-4" src={review.avatar} alt="Avatar"
/>
          <div className="text-sm">
```

```
                    <p className="text-gray-900 dark:text-gray-100 leading-
none">{review.name}</p>
      <p className="text-gray-600 dark:text-gray-400">{review.date}</p>
     </div>
    </div>
    <p className="text-gray-700 dark:text-gray-300 text-base">{review.rev}</p>
    <div className="mt-4">
     {Array(review.rating).fill().map((_, i) => (
              <svg key={i} className="w-5 h-5 text-yellow-500 inline-block"
fill="currentColor" viewBox="0 0 20 20">
         <path d="M10 15l-5.09 2.67L6.18 11 2 7.24l6.91-.64L10 2l1.09 4.6L18 7.24
13.82 11l1.27 6.67z"/>
      </svg>
     ))}
    </div>
   </div>
     <div className="max-w-sm rounded overflow-hidden shadow-lg p-6 bg-white
dark:bg-gray-800">
    <div className="flex items-center mb-4">
      <img className="w-10 h-10 rounded-full mr-4" src={reviee.avatar} alt="Avatar"
/>
     <div className="text-sm">
      <p className="text-gray-900 dark:text-gray-100 leading-none">{
       <path d="M10 15l-5.09 2.67L6.18 11 2 7.24l6.91-.64L10 2l1.09 4.6L18 7.24
13.82 11l1.27 6.67z"/>
      </svg>
     ))}
    </div>
    </div>
   </div>
  </div>


  </>
  )
}

export default Home
```

USER DASHBOARD:

```
const UserDashboard = () => {

  const[invoices,setInvoices]=useState([]);

    return (
        <div>
        <div className="flex flex-row p-4 gap-4">
            <Card className='w-1/4 border border-primary'>
                <CardHeader className="flex flex-row items-center justify-between space-y-0
pb-2">
                    <CardTitle className="text-sm font-medium">
                        Total Halls booked
                    </CardTitle>
                    <Users className="h-6 w-6 text-primary" />
                </CardHeader>
                <CardContent>
                    <div className="text-2xl font-bold">3</div>
                </CardContent>
            <Card className='shadow-sm shadow-primary'>
        <CardHeader className='w-full flex flex-row justify-between items-center'>
          <CardTitle>Halls Booked</CardTitle>
        </CardHeader>
        <CardContent>
          <Table>
            <TableHeader>
              <TableRow>
            <TableHead>Hall Name</TableHead>
            <TableHead>Occasion</TableHead>
            <TableHead >Status</TableHead>

           </TableRow>
          </TableHeader>
          <TableBody>
           {invoices.map((invoice) => (
            <TableRow key={invoice.invoice}>

             <TableCell>{invoice.hall}</TableCell>
             <TableCell>{invoice.date}</TableCell>
             <TableCell >{invoice.status}</TableCell>

            </TableRow>
           ))}
          </TableBody>
         </Table>
        </CardContent>
       </Card>

        </div>
  )
}

export default UserDashboard
```

ADMIN DASHBOARD

```
import React, { useState } from 'react';
import { useNavigate } from 'react-router-dom';

import RequestSummary from './RequestSidePanel';
```

```
function RequestManagement() {

    const navigate = useNavigate();

    const [requests, setRequests] = useState([
        {
            id: 1,
            client: 'John Doe',
                amenities: ['Projector', 'Whiteboard', 'Wi-Fi']
            }
        },
  {
            id: 3,
            client: 'Bob Johnson',
            date: '2024-08-25',
            event: 'Birthday Party',
            status: 'pending',
            hallDetails: {
                name: 'Party Hall',
                capacity: 100,
                price: '$1000',
            date: '2024-08-20',
            event: 'Corporate Event',
            status: 'pending',

'Sound system']
            }
        },
    ]);

    const [expandedId, setExpandedId] = useState(null);
    const [showSummary, setShowSummary] = useState(false);
  const handleAccept = (id) => {
    setRequests(requests.map(req =>
      req.id === id ? { ...req, status: 'accepted' } : req
    ));
  };

  const handleDecline = (id) => {
    setRequests(requests.map(req =>
      req.id === id ? { ...req, status: 'declined' } : req
    ));
  };
    setShowSummary(false);
  };



              onClick={handleRequestSummary}
              className="bg-blue-500 hover:bg-blue-600 text-white font-bold py-2 px-
4 rounded transition-colors duration-200 mb-2 sm:mb-0"
            >
              Request Summary
            </button>
            <button
              onClick={handleLogout}
              className="bg-red-500 hover:bg-red-600 text-white font-bold py-2 px-4
rounded transition-colors duration-200"
            >
              Logout
            </button>
          </div>

              </thead>
              <tbody className="bg-white divide-y divide-indigo-100">
                {requests.map((request) => (
                <React.Fragment key={request.id}>
                  <tr className="hover:bg-indigo-50 transition-colors duration-
200">
                    <td className="px-4 py-3 whitespace-nowrap text-sm font-
medium text-gray-800">{request.client}</td>
                    <td className="px-4 py-3 whitespace-nowrap text-sm text-
gray-600">{request.date}</td>
                            Accept
                          </button>
                          <button onClick={() => handleDecline(request.id)}
className="bg-red-500 hover:bg-red-600 text-white font-bold py-1 px-2 rounded mr-1
transition-colors duration-200">
                            Decline
                          </button>
                        </>
                      )}
                      <button onClick={() => toggleExpand(request.id)}
className="bg-blue-500 hover:bg-blue-600 text-white font-bold py-1 px-2 rounded
transition-colors duration-200">
                        {expandedId === request.id ? 'Hide Details' : 'Show
Details'}
                      </button>
                    </td>
                  </tr>
                  {expandedId === request.id && (
                    <tr>
                      <td colSpan="5" className="px-4 py-3">
                        <div className="bg-indigo-50 p-4 rounded-lg shadow-
inner">
                          <h4 className="font-bold text-lg mb-3 text-indigo-
800">Hall Details</h4>
                          <div className="grid grid-cols-1 sm:grid-cols-2 gap-
4">
```

```
                                                  <p><span className="font-semibold text-indigo-
700">Name:</span> {request.hallDetails.name}</p>
                                                  <p><span className="font-semibold text-indigo-
700">Capacity:</span> {request.hallDetails.capacity}</p>
                                                  <p><span className="font-semibold text-indigo-
700">Price:</span> {request.hallDetails.price}</p>
                                                  <p><span className="font-semibold text-indigo-
700">Amenities:</span> {request.hallDetails.amenities.join(', ')}</p>
                                               </div>
                                            </div>
                                         </td>
                                      </tr>
                                   )}
                                </React.Fragment>
                             ))}
                          </tbody>
                       </table>
                    </div>
                 </div>
              </div>
   );
} export default RequestManagement;
```

CHAPTER 6

CONCLUSION

This chapter tells about the conclusion that anyone can drive from the project andthe learning we learnt by taking over this project.

6.1 CONCLUSION

In conclusion, the Hall Management System is a comprehensive and advanced solution designed to revolutionize the management and booking of event spaces. This project integrates a sophisticated tech stack comprising React.js for a dynamic and responsive user interface, Tailwind CSS for modern and customizable styling, Spring Boot for a robust backend framework, REST APIs for efficient communication, and MySQL for secure and reliable data management. The system empowers users to explore a wide range of halls based on key criteria such as location, size, rating, and amenities, while also offering features to bookmark favorites and make reservations with ease. By streamlining the booking process and enhancing operational efficiency, the Hall Management System not only simplifies the task of finding and securing the ideal venue but also provides a user-friendly experience that meets both personal and professional event planning needs. Overall, this project exemplifies the effective integration of modern technologies to address the challenges of event space management and elevate the user experience.

6.2 FUTURE SCOPE :

Enhanced Security Measures:

Continuing to advance security protocols to protect customer data and ensure secure transactions. This includes implementing encryption, multi-factor authentication, and regular security audits to prevent unauthorized access and breaches.

Analytics and Reporting:

Creating robust analytics tools to offer insights into sales trends, customer preferences, and product performance. This data can help administrators and vendors make informed decisions, optimize inventory, and tailor marketing strategies to improve business outcomes.

Adaptive Pricing Models:

Implementing adaptive pricing strategies that adjust product prices based on factors such as demand, inventory levels, and competitor pricing. This ensures competitive pricing and maximizes profitability.

Collaborations with Product Suppliers:

Partnering with suppliers and brands to expand the product range and offer exclusive deals or promotions through the platform. These collaborations can enhance the product selection and provide additional value to customers.

Personalized Marketing:

Utilizing customer data to create targeted marketing campaigns and promotions. Personalized offers and recommendations based on shopping history and preferences can increase customer engagement and drive sales

REFERENCES:

- https://www.venuelook.com/delhi?area=Delhi
- https://github.com/aravindvnair99/Hall-Management-System
- https://github.com/MaX-NeO/MaX-Games-FrontEnd-AWS
- https://www.justdial.com/
- https://www.jesvenues.com/venues
- https://www.hallmaster.co.uk/