



**DHARMSINH DESAI UNIVERSITY,
NADIAD**

**FACULTY OF TECHNOLOGY
DEPARTMENT OF COMPUTER
ENGINEERING**

BTech CE Semester-VI

Subject: (CE624) Web Service Development

Project Title: Hotel Booking System

Submitted by:

Name: Jay Kalpeshkumar Shah

Roll No: CE124

ID: 20CEUOF057

Guided By:

Prof. Ankit P. Vaishnav

DHARMSINH DESAI UNIVERSITY



Certificate

This is to certify the practical / term work carried out in the subject of
Web Service Development *and recorded in this journal is*
the bonafide work of Mr Jay Kalpeshkumar Shah *Roll No:*
CE124 *Identity No:* 20CEUOF057 *of BTech semester*
VI *in the branch of* Computer Engineering *during the*
academic year 2022-2023

Prof. Ankit P. Vaishnav
Dept. Computer Engineering
Faculty Of Technology
D.D.University Nadiad

Dr. C.K. Bhensdadia
HOD Of Computer Engg,
Faculty Of Technology
D.D.University Nadiad

Table of Content

Index	Content	Page No
1	Abstract	4
2	Introduction	5
3	Software Requirement Specification	6
4	Database Design	9
5	Implementation Detail	11
6	Testing	18
7	Screen-shots	19
8	Conclusion	23
9	Limitation and Future Extension	24
10	Bibliography	24

1. Abstract

Hotel booking system is an online platform that facilitates hotel reservations and provides a convenient way for travelers to book accommodations. It allows users to search for hotels based on their preferences, such as location, price, and amenities, and make bookings through a secure and user-friendly interface. The system also enables hotel owners to manage their inventory, set room rates, and monitor reservations in real-time. By automating the booking process, the hotel booking system improves efficiency, enhances customer experience, and increases revenue for hotels. This abstract provides a brief overview of the key features and benefits of a hotel booking system.

2. Introduction

The main aim of this project is to book your room as per user requirement . User need to first create profile for yourself for get facilitate with our hotel facility and have entertainment. User can easily book and get their room in advance for your tour.

Technology and Tools used

❖ Technology

- Asp.Net Core
- React
- CSS
- MUI
- Tailwind CSS
- Web API

❖ Tools

- Git
- Visual Studio 2022
- Visual Studio Code

3. Software Requirement Specification (SRS)

USERS OF THE SYSTEM:

1. User
2. Admin

FUNCTIONAL REQUIREMENTS:

1. Authentication

1.1 Sign up

Description : The users would have to sign up in order to create a new account and provide relevant details.

Input : Email id , Name , Mobile No. , Username , Password.

Output : The users would be directed to the welcome page and their account would be created. They can then access their account when required.

1.2 Login

Description : The user needs to login with their username / password to have access to their account.

Input : Mobile No / Password.

Output : The users will be directed to their blog feed and would be able to access their account.

2. Manage Room

2.1 Book Room:

Description: The users can book their room according to their requirements in our hotels.

Input: According to customer preference select room type and number of persons

Output: Acknowledge for booking the room

2.2 Read Booking details:

Description: Admin can read booking details

2.3. Cancel Room:

Description: The users can cancel their room if he/she has valid reason for cancellation

Input: Cancel room

Output: Acknowledge for canceling the room

2.4 Update Room:

Description: According to their comfort level they can update their choice of the room.

Input: Select your new preferred room type.

Output: Acknowledge for Updation of the room

3. Manage Customers

3.1 Update details:

Description: Customer can update their details for better information.

Input: Update the required details

Output: Acknowledge for Updation of the details

3.2 Read details:

Description : Admin can read all user details

3.3 Delete details:

Description: The admin can delete their details.

Input: Cancel room

Output: Acknowledge for canceling the profile

NON-FUNCTIONAL REQUIREMENTS:

N.1: The server hardware can be any computer capable of running both the web and database servers and handling the expected traffic.

N.2: System should be easily used by the users.

N.3: The system should be available at all times, meaning the user can access it using a web-browser.

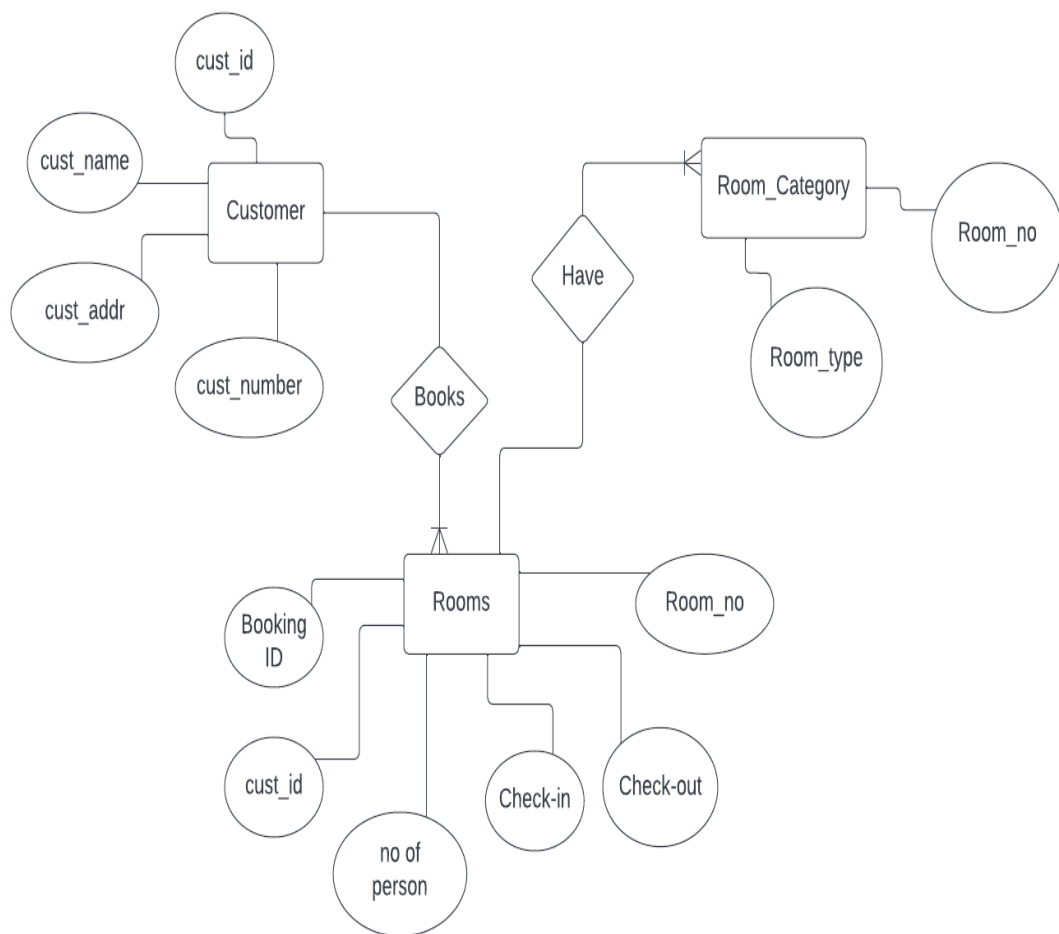
N.4: Secure access to confidential information of the users.

N.5: The system will be supported by Windows.

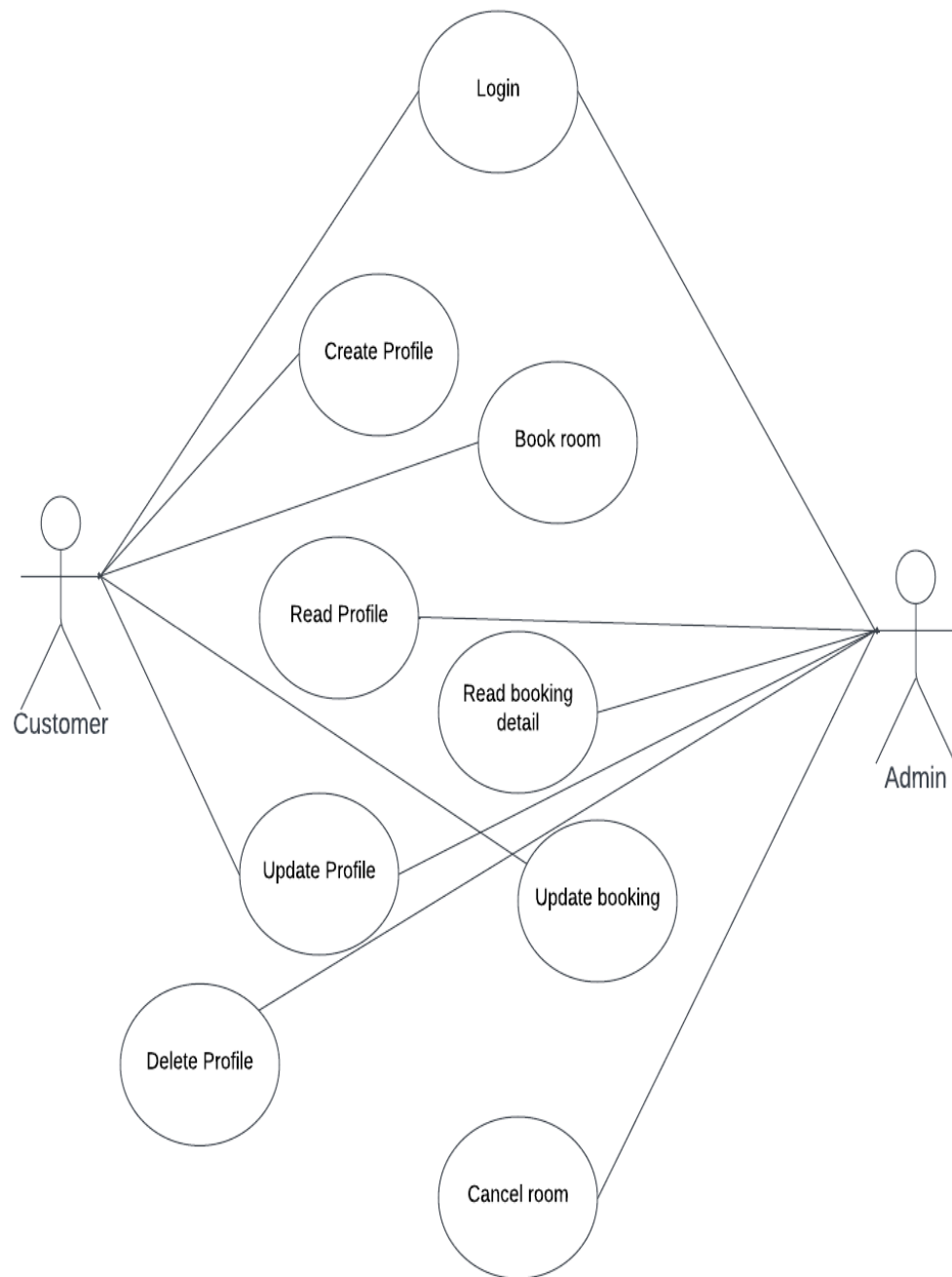
N.6: A database management system that is available free of cost in the public domain should be used.

4. Database Design

- ER Diagram



- **Use-case Diagram**



5. Implementation Detail

I. Modules created and brief description of each module

- User/Customer Model

It contains user's personal information:

- Username
- Password
- Name
- Email
- Mobile
- Address
- City
- State

```
namespace Project_Hotel.Model
{
    6 references
    public class User
    {
        4 references
        public int Id { get; set; }
        2 references
        public string? Username { get; set; }
        1 reference
        public string? Password { get; set; }
        //public List<Booking>? Book { get; set; }
        0 references
        public string? Name { get; set; }

        0 references
        public string? Email { get; set; }

        0 references
        public string? Mobile { get; set; }

        0 references
        public string? Address { get; set; }

        0 references
        public string? City { get; set; }

        0 references
        public string? State { get; set; }
    }
}
```

- Booking Model

It contains Booking Details Required:

- Username
- Number of Persons
- Check-In Time
- Check-Out Time
- Room_type(Normal,Delux,SuperDelux)

```
using System.ComponentModel.DataAnnotations;

namespace Project_Hotel.Model
{
    6 references
    public class Booking
    {
        [Key]
        3 references
        public int BookId { get; set; }
        0 references
        public string? UserName { get; set; }
        //public int Id { get; set; }
        //public User? customer { get; set; }
        0 references
        public int noOfPersons { get; set; }
        0 references
        public DateTime? Checkin { get; set; }

        0 references
        public DateTime? Checkout { get; set; }

        0 references
        public string? Room_type { get; set; }
    }
}
```

II. Function Prototype which implements major functionality

- User authentication

```
[HttpPost("Login")]
0 references
public async Task<ActionResult> Login(UserLoginDTO userDTO)
{
    var res = await _context.Users.FirstOrDefaultAsync(u => u.UserName.ToLower() == userDTO.Username.ToLower());
    Console.WriteLine(res);
    if (res == null)
    {
        return BadRequest($"Incorrect username or password!");
    }
    return Ok(new { token = res.Id, username=res.UserName, res.Password, status = 200 });
}
```

We will collect user input data from the registration form and determine whether or not the user's email address already exists. If the user does not exist, we will add that user to the database and redirect the user to the login page.

- CRUD operation
1)Customer Details:

```
// GET: api/Users
[HttpGet]
0 references
public async Task<ActionResult<IEnumerable<User>>> GetUsers()
{
    if (_context.Users == null)
    {
        return NotFound();
    }
    return await _context.Users.ToListAsync();
}
```

```

// GET: api/Users/5
[HttpGet("{id}")]
0 references
public async Task<ActionResult<User>> GetUser(int id)
{
    if (_context.Users == null)
    {
        return NotFound();
    }

    var user = await _context.Users.FindAsync(id);

    if (user == null)
    {
        return NotFound();
    }

    return user;
}

```

```

// To protect from overposting attacks, see https://go.microsoft.com/fwlink/?linkid=2123754
[HttpPut("{id}")]
0 references
public async Task<IActionResult> PutUser(int id, User user)
{
    if (id != user.Id)
    {
        return BadRequest();
    }

    _context.Entry(user).State = EntityState.Modified;

    try
    {
        await _context.SaveChangesAsync();
    }
    catch (DbUpdateConcurrencyException)
    {
        if (!UserExists(id))
        {
            return NotFound();
        }
        else
        {
            throw;
        }
    }

    return NoContent();
}

```

```

// POST: api/Users
// To protect from overposting attacks, see https://go.microsoft.com/fwlink/?linkid=2123754
[HttpPost]
0 references
public async Task<ActionResult<User>> PostUser(User user)
{
    if (_context.Users == null)
    {
        return Problem("Entity set 'HotelDbContext.Users' is null.");
    }
    _context.Users.Add(user);
    await _context.SaveChangesAsync();

    return CreatedAtAction("GetUser", new { id = user.Id }, user);
}

```

```

// DELETE: api/Users/5
[HttpDelete("{id}")]
0 references
public async Task<IAActionResult> DeleteUser(int id)
{
    if (_context.Users == null)
    {
        return NotFound();
    }
    var user = await _context.Users.FindAsync(id);
    if (user == null)
    {
        return NotFound();
    }

    _context.Users.Remove(user);
    await _context.SaveChangesAsync();

    return NoContent();
}

1 reference
private bool UserExists(int id)
{
    return (_context.Users?.Any(e => e.Id == id)).GetValueOrDefault();
}

```

2)Booking Details:

```

// GET: api/Bookings
[HttpGet]
0 references
public async Task<ActionResult<IEnumerable<Booking>>> GetBookings()
{
    if (_context.Bookings == null)
    {
        return NotFound();
    }
    return await _context.Bookings.ToListAsync();
}

// GET: api/Bookings/5
[HttpGet("{id}")]
0 references
public async Task<ActionResult<Booking>> GetBooking(int id)
{
    if (_context.Bookings == null)
    {
        return NotFound();
    }
    var booking = await _context.Bookings.FindAsync(id);

    if (booking == null)
    {
        return NotFound();
    }

    return booking;
}

```

```

// PUT: api/Bookings/5
// To protect from overposting attacks, see https://go.microsoft.com/fwlink/?linkid=2123754
[HttpPut("{id}")]
0 references
public async Task<IActionResult> PutBooking(int id, Booking booking)
{
    if (id != booking.BookId)
    {
        return BadRequest();
    }

    _context.Entry(booking).State = EntityState.Modified;

    try
    {
        await _context.SaveChangesAsync();
    }
    catch (DbUpdateConcurrencyException)
    {
        if (!BookingExists(id))
        {
            return NotFound();
        }
        else
        {
            throw;
        }
    }

    return NoContent();
}

```



```

// POST: api/Bookings
// To protect from overposting attacks, see https://go.microsoft.com/fwlink/?linkid=2123754
[HttpPost]
0 references
public async Task<ActionResult<Booking>> PostBooking(Booking booking)
{
    if (_context.Bookings == null)
    {
        return Problem("Entity set 'HotelDBContext.Bookings' is null.");
    }
    _context.Bookings.Add(booking);
    await _context.SaveChangesAsync();

    return CreatedAtAction("GetBooking", new { id = booking.BookId }, booking);
}

```

```

// DELETE: api/Bookings/5
[HttpDelete("{id}")]
0 references
public async Task<ActionResult> DeleteBooking(int id)
{
    if (_context.Bookings == null)
    {
        return NotFound();
    }
    var booking = await _context.Bookings.FindAsync(id);
    if (booking == null)
    {
        return NotFound();
    }

    _context.Bookings.Remove(booking);
    await _context.SaveChangesAsync();

    return NoContent();
}

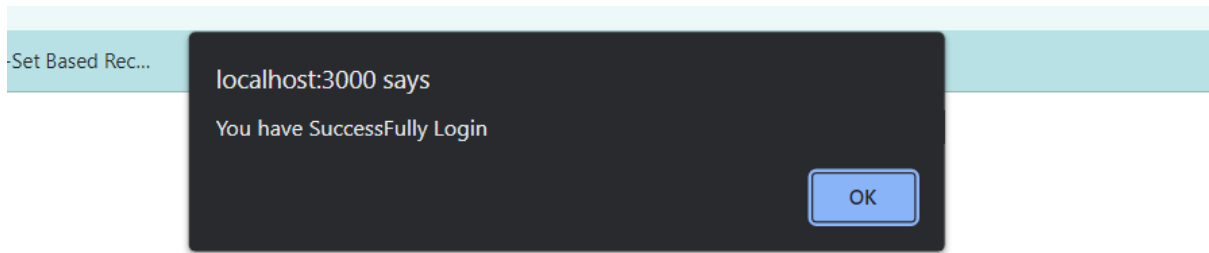
1 reference
private bool BookingExists(int id)
{
    return (_context.Bookings?.Any(e => e.BookId == id)).GetValueOrDefault();
}

```

6. Testing

Sr No	Test Scenario	Expected Results	Actual Result	Status
1.	Login in with the correct credentials	User should be able to log in to Home page.	User is logged in and redirected to home screen	success
2.	Login with incorrect credentials	User should not be able to logged in Alert for enter correct credentials	System gives a message to user and redirected to same page. Alert for enter correct credentials	success
3.	Register with correct details	User should be able to register in our system.	User is successfully register in our system	success
4.	Register with incorrect details	User should not be able to Register in our system	System gives a message to user and redirected to same page	success

7. Screenshots



Sign in

UserName *
keyur12

Password *
.....

☐ Remember me

SIGN IN

[Don't have an account? Sign Up](#)

Easiest way to book your stay

- ✓ Clean and Hygenic Room
- ✓ Just come and enjoy the stay
- ✓ Easy to Book

Enter your email address

Get Notify



HOTEL INSIGHTS




HOTEL FACILITIES

Customers Profile

Id	UserName	Name	Email	Mobile Number	City	State	Address	
4	keyur12	KeyurJivani	keyur123@gmail.com	9943432352	Morbi	Gujarat	Borali	DELETE
6	Dhruvi12	Dhruvi	dhruvi@gmail.com	9032455221	Rajkot	Gujarat	rajkot	DELETE
8	Rahul12	Rahul Shah	rahul@gmail.com	7523234232	Vadodara	Gujarat	Ratri Bazaar	DELETE
10	Jash12	Jash Shah	jash123@gmail.com	609530459	Modasa	Gujarat	Harni	DELETE

te

Electronic S...  Skill-Set Based Rec...

localhost:3000 says
Thank you for visiting our site Your booking is Confirmed Successfully

[OK](#)

Book your Room

Number of Persons

5

Arrival

20-03-2023

Departure

15-03-2023

Room Type

Delux

[BOOK NOW](#)

USEFUL LINKS

About Us

is, we respond 1-2 business days.

Booking Details

Bookid	UserName	Number of Persons	Checkin	CheckOut	Room_type	
2	keyur12	4	2023-03-13T00:00:00	2023-05-05T00:00:00	Delux	DELETE
4	keyur12	2	2023-03-20T00:00:00	2023-03-30T00:00:00	Normal	DELETE
6	keyur12	6	2023-03-21T00:00:00	2023-03-17T00:00:00	Delux	DELETE
7	keyur12	10	2023-03-22T00:00:00	2023-03-15T00:00:00	Normal	DELETE
8	Rahul12	6	2023-03-14T00:00:00	2023-03-16T00:00:00	Super Delux	DELETE
9	Jash12	6	2023-03-13T00:00:00	2023-03-22T00:00:00	Delux	DELETE
10	<u>keyur12</u>	<u>5</u>	2023-03-20T00:00:00	2023-03-15T00:00:00	<u>Delux</u>	DELETE

onic S... Skill-Set Based Rec...

localhost:3000 says
Your Profile is Update SuccessFully

OK

Update Your Profile

Name

KeyurJivani

Email

keyur123@gmail.com

Mobile number

9943432352

City

Vadodara

State

Gujarat

Address

Borali

UPDATE PROFILE

The screenshot displays a web application interface. At the top, a dark notification box from 'localhost:3000' states 'Your Booking Details is Update SuccessFully' with an 'OK' button. Below this, the main heading is 'Update Your Booking Details'. The form contains the following fields:

- Number of Persons:** A text input field containing the value '5'.
- Arrival:** A date selection field showing '05-03-2023' with a calendar icon.
- Departure:** A date selection field showing '17-03-2023' with a calendar icon.
- Room Type:** A dropdown menu currently displaying 'Super Delux'.

A blue button at the bottom of the form is labeled 'UPDATE YOUR BOOKING DETAILS'.

8. Conclusion

Hotel Booking System is completely free, easy to use, good user interface, and great user experience booking site which is created through ASP.NET Core Web API Technology for a Hotel.

The functionalities which are implemented in this project is:

- User Registration
- User Login
- CRUD operation on Profile
- CRUD operation on Booking
- User logout

9. Limitations and Future Extension

Limitation:

- Current project is limited to only CRUD operation.
- This site is limited for single Hotel Booking System.

Future Extension:

- Converting this Hotel booking system to Hotel Management System.
- Login with Google API

10. Bibliography

- Stack Overflow
- <https://learn.microsoft.com/en-us/aspnet/core/tutorials/first-web-api?view=aspnetcore-7.0&tabs=visual-studio>