Chapter 1
INTRODUCTION

1.1 Organization/ Client Profile:

Name: Hotel Surbhi Pure Veg

Address: The Presidency, Old Agra Rd, near Kalika Mandir, Gadkari Chowk, Renuka Nagar, Nashik, Maharashtra 422001

About Organization:

Hotel Surbhi Pure Veg, located in Nashik, is a fantastic place for folks who love vegetarian food. It's right in the middle of Nashik, surrounded by beautiful vineyards and a peaceful atmosphere. At Surbhi, we serve only vegetarian dishes made with the freshest ingredients. Whether you're craving Maharashtrian favorites or North Indian delights, we've got something for everyone. Our restaurant has a cozy and welcoming vibe, perfect for spending quality time with family and friends. Whether you're celebrating a special occasion or just want to enjoy a deliciousmeal, our friendly staff will make sure you have a great experience.

1.2 Abstract:

The restaurant management application, named "Hotel Surbhi," aims to streamline restaurant operations and enhance customer experience by providing a comprehensive solution for both restaurant staff and customers. Developed using the MERN(MongoDB, Express.js, React.js, Node.js) Stack, Hotel Surbhi offers intuitive interfaces for managing orders, inventory, and reservations, while also providing customers with a user-friendly platform for browsing menus, placing orders, and providing feedback

1.3 Existing System:

The records are maintained manually. Handling and updating these records manually increase the chances of mistakes. It takes a lot of time and needs many employees to accomplish the task. It even lacks security and disability to produce

various types of reports. It will also require a lot of space for storing the details and records of users and staff. For management in the current system all work is done on paper. The whole session is stored in register and at the end of the session the reports are generated. This is a very time consuming process.

- The papers can miss placed and documents can be loss this will cause extra work for the staffs to maintain the records.
- The system is not user friendly because the retrieval of data is very slow and data is stored manually.

1.4 Scope of System

Admin will register new staffs and users and remove their account when they leave the institute.users able to check their stored by staff for respective menu. Admin can change the menus as per the syllabus changes. Staff can maintain the user for the assigned menus. Everyone can see graphical representation of of users.

There are three modules Admin, users and Staff in this system.

Admin:-

- Admin can manage profile, menus, staff, tables.
- Admin can update the profile
- Admin need to register the users and staff before giving them login credentials for their use.
- Admin can assign menus, sessions of staff.
- Admin can check the staff performance.
- Admin can accept or reject the leave request of staff.

users:-

- users can login to system using login credentials given by admin.
- users can update their profile information and password.
- users can check their as filled by the user/staff.
- users can check the notifications.
- user can apply for request for leave.
- user can give feedback or response.

Staff:-

- Staff can login to system using login credentials given by admin.
- Staff can update their profile information and password.
- Staff can add/update the user of their respective menus.
- Staff can check the notifications.
- Staff can give feedback or response.

1.5 Operating Environment – Hardware & Software:

1.5.1 Client-Side System Specification:

1.5.1.1 Hardware

Item name	Specification
Laptop/Desktop	Minimum Intel Pentium IV or Above Minimum RAM: 512 MB or more Minimum Hard disk: 1 GB of free space

1.5.1.2 Software

Particular	Specification
Operating System	Minimum Windows 7 or above / Minimum Linux 6.1 or above/ Minimum Mac OS X 10.1 or above
Browser(s)	Google Chrome 5 or Higher Mozilla Firefox 3.6 or Higher Internet Explorer 9 or Higher

1.5.2 Server-Side System Specification:

Database	MongoDB(8.0)
Browser(s)	Google Chrome 5

1.5.3 Developer-Side System Specification:

1.5.3.1 Hardware

Item name	**Specification**
Laptop/Desktop	Intel(R) Core(TM) i5-10210U CPU @ 1.60GHz 2.11 GHz RAM: 8GB Hard disk: 512 GB SSD

1.5.3.2 Software

Particular	**Specification**
Operating System	Windows 10 and Above
Documentation	Microsoft Office 2007 or higher
Browser(s)	Google Chrome 5 or higher
Text Editor(s)	Visual Studio Code (1.77)
Database	MongoDB(8.0)

1.6 Detail Description of Technology Used:

- **HTML (Version: HTML5)**

  It stands for HyperText Markup Language which is used to add basic structure to web pages. HTML is used to develop the web pages. HTML provides the building block of the webpage. Other than that HTML is very useful in building and connecting multiple pages. HTML uses markup to annotate text, images, and other content. HTML tags are used to build the block of the web page.

- **CSS (Version: CSS3)**

  It stands for Cascading Style Sheets which is used to make websites more attractive web pages. It is used to design the page and to create animation as well. It is used for describing the presentation of a document written in mark-up languages such as HTML or XHTML. It is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript. CSS was developed by World Wide Web Consortium (W3C). It provides more flexibility and control to the HTML tags and content.

- **JavaScript (Version: ES022)**

  It is a scripting language. It was developed by Brendan Eich of Netscape. It is one of the most popular languages in 2022. It is used to make web pages more interactive and used to create web applications as well as games and mobile applications. JavaScript is used to program the behavior of the web page. All major web browsers have a dedicated JavaScript engine to execute the code on the user's device.

- **TailwondCSS(3.4)**

  Utility classes help you work within the constraints of a system instead of littering your stylesheets with arbitrary values. They make it easy to be consistent with color choices, spacing, typography, shadows, and everything else that makes up a well-engineered design system.

Tailwind CSS works by scanning all of your HTML files, JavaScript components, and any other templates for class names, generating the corresponding styles and then writing them to a static CSS file.
It's fast, flexible, and reliable — with zero-runtime.

- **NodeJS(9.11.2)**

  Node.js is composed of Google's V8 JavaScript engine, the libUV platform abstraction layer, and a core library that is written in JavaScript. Additionally, Node.js is based on the open web stack (HTML, CSS, and JS), and operates over the standard port 80.

  Node.js provides developers a comprehensive tool for working in the non-blocking, event-driven I/O paradigm. Ryan Dahl, the creator of Node.js was "inspired by applications like Gmail" and—in creating Node.js—aimed to create real-time websites with push capability.

  Node.js shines in real-time web applications employing push technology over WebSocket. After over 20 years of stateless-web based on the stateless request-response paradigm, we finally have web applications with real-time, two-way connections, where both the client and server can initiate communication, allowing them to exchange data more freely. This is in stark contrast to the typical web response paradigm, where the client always initiates communication.

- **MongoDB(8.0)**

  Most databases force you to use heavy wrappers, like ORMs (Object-Relational Mappers), to get data into Object form for use in programs. MongoDB's decision to store and represent data in a document format means that you can access it from any language, in data structures that are native to that language (e.g., dictionaries in Python, objects in JavaScript, Maps in Java, etc.).

  MongoDB is designed to make data easy to access, and rarely to require joins or transactions, but when you need to do complex querying, it's more than up to the task.

The MongoDB Query API allows you to query deep into documents, and even perform complex analytics pipelines with just a few lines of declarative code. MongoDB is designed from the ground up to be a distributed database.

Create clusters with real-time replication, and shard large or high-throughput collections across multiple clusters to sustain performance and scale horizontally.

Thanks to the document model used in MongoDB, information can be embedded inside a single document rather than relying on expensive join operations from traditional relational databases. This makes queries much faster, and returns all the necessary information in a single call to the database. If needed, you can perform a left outer join with the $lookup aggregation pipeline stage, which delivers similar performance to RDBMSs.

Chapter 2
PROPOSED SYSTEM

2.1 Proposed System:

In contrast to the current system, the proposed digital solution for Hotel Surbhi Pure Veg aims to offer administrators a comprehensive and efficient platform for managing hotel facilities.

The proposed system will offer the following key features :

- Retrieval of data will be fast.
- System gives graphical representation of of users.
- users can request for leave.
- users can give feedback or response about college.
- Admin can add users and staff in the system.
- Admin can accept or reject request for leave.
- Admin can update the course and menu details.
- users do not need to check the notice board every day and every one will stay updated.

2.2 Objectives Of System :

- To provide user and staff details to admin
- To increase the efficiency of record management.
- To decrease time required to access and deliver user records.
- To store the record in the system.
- To display the menu wise record.
- To accept or decline leave for request of users.
- To provide graphical representation of stored record.
- To generate the report as per admin request.
- To collect feedback from customer and staff.

2.3 Feasibility Study:

1. Technical Feasibility: -

This system assesses the current resources and technology, which are required to accomplish user requirements in the software within the allocated time and budget. For this, the software development team ascertains whether the current resources and technology can be upgraded or added in the software to accomplish specific user requirements.

2. Operational Feasibility: -

This system assesses the extent to which the required software performs a series of steps to solve business problems and user requirements. This feasibility is dependent on human resources and involves visualizing whether the software will operate after it is developed and be operative once it is installed.

3. Economic Feasibility: -

This system determines whether the required software is capable of generating financial gains for an organization. It involves the cost incurred on the software development team, estimated cost of hardware and software, cost of performing feasibility study, and so on. For this, it is essential to consider expenses made on purchases and activities required to carry out software development. In addition, it is necessary to consider the benefits that can be achieved by developing the software.

4. Market Feasibility: -

A market feasibility study for a user system would involve assessing the demand and viability of such a system. Analyze existing user systems in the market, their features, pricing, and market share. Consider trends in education technology and the increasing demand for efficient administrative processes. By conducting this study, you can determine the potential demand, market acceptance, and likelihood of success for your user system.

5. Legal Feasibility: -

A legal feasibility study for a user system would assess the legal and regulatory aspects associated with its implementation. By conducting a legal feasibility study, you can identify and address the legal requirements and potential risks associated with implementing a user system. This helps ensure compliance with relevant laws, protect user data, and mitigate legal liabilities.

2.4 User Requirements:

- The system should give login option to user and staff after admin registers the user and staff.
- The system should be capable of managing the profile for both user and staff.
- The system allows staff to add the of users of their menu.
- The system shows the of user in a graphical manner
- The system displays the names of users who are defaulters.
- The system admin can add or remove the staff and users from the system.

Chapter 3

ANALYSIS AND DESIGN

3.1 System Requirements (Functional and Non-Functional Requirements):

Functional Requirements:

User Registration and Login: Allow users to create accounts and log in to the system securely.

Update Profile: The system should allow to make changes or update profile according to their choice.

Update Course: Admin should be able to update and manage course.

Update menu: Admin should be able to update and manage menu.

Update Session: Admin should be able to add and manage session.

Add Staff: Admin should be able to add and manage new staff.

Notify User: user should get notified about their order.

View Reporting: user should be able to view their Reporting.

Logout: user should be able log out.

Non-Functional Requirements:

User Interface: The system should have a user-friendly and intuitive interface that is easy to navigate.

Performance: The system should provide fast response time. It should be scalable to accommodate increased traffic during peak periods.

Security: The system should ensure the security of sensitive data. It should implement measures to protect against unauthorized access.

Usability and User Experience: The system should have an intuitive and user-friendly interface that is easy to navigate. It should provide clear instructions, error messages, and feedback to users.

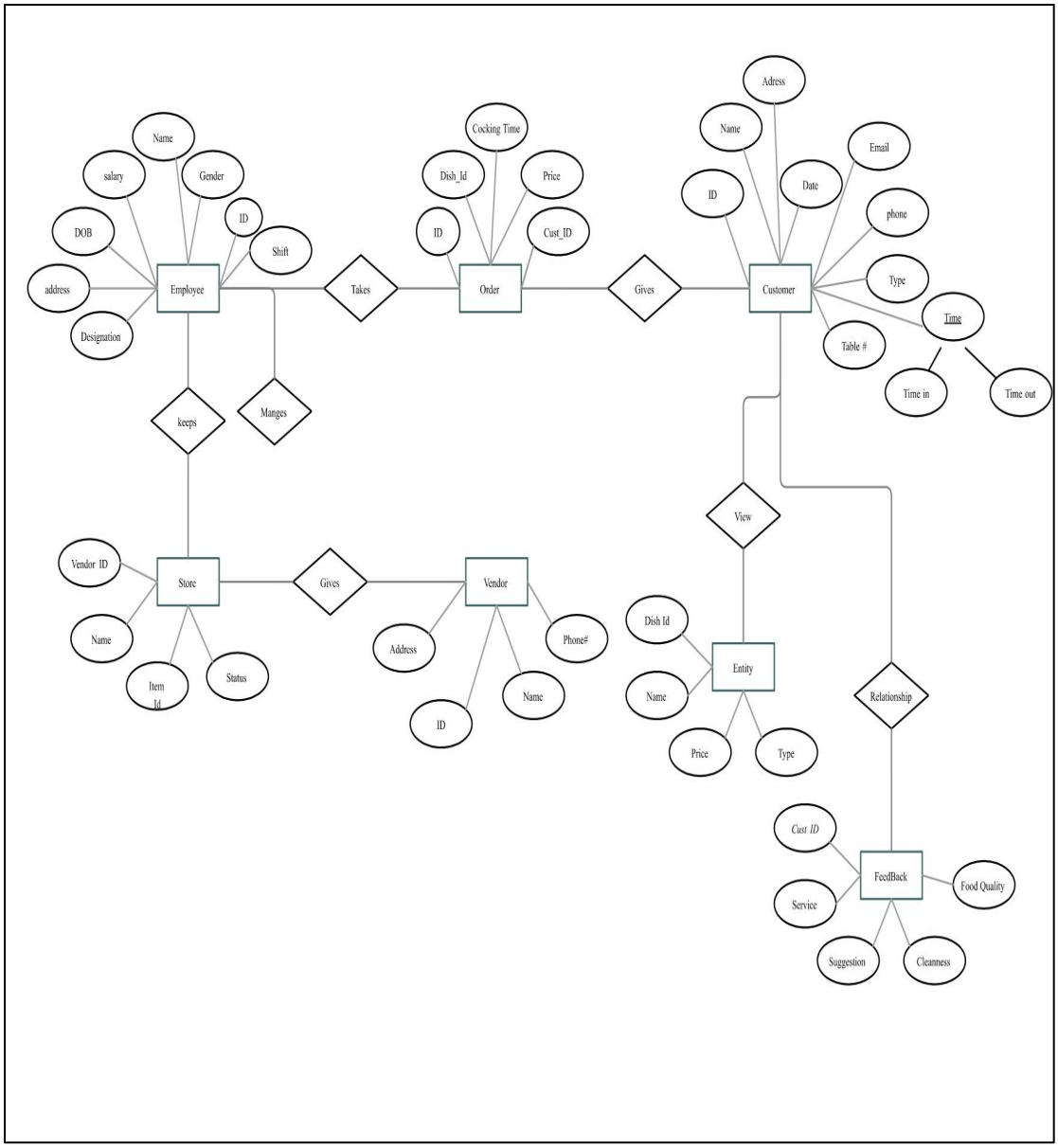
Reliability: The system should be available and operational 24/7, with minimal downtime for maintenance or upgrades.

Scalability: The should be designed to handle increasing volumes of data.

Compatibility: The system should be compatible with different web browsers, devices, and operating systems to provide a seamless experience for users.

Maintenance and Support: Consider ease of system maintenance, provide regular updates, and offer technical support to address any issues or queries.

3.2 Entity Relationship Diagram



3.3 Table Structure

Table 1:

Table Name	User
Primary Key	u_id
Foreign key	-
Table Description	It is used to store user authorization details.

Sr No.	**Field Name**	**Data Type**	**Constraints**	**Description**
1	u_id	int(5)	Primary key	Primary id of the authorize user table to store the user id.
2	u_username	varchar(25)	Not null	To store the username of the user.
3	u_fname	varchar(20)	Not null	To store the first name of the user.
4	u_lname	varchar(20)	Not null	To store the last name of the user.
5	u_gender	varchar(10)	Not null	To store the gender of user
6	u_address	varchar(50)	Not null	To store the address of user
7	u_email	varchar(20)	Not null	To store the email of the user.
8	u_password	varchar(50)	Not null	To store the password.

Table 2:

Table Name	Reporting
Primary Key	r_id
Foreign key	menu_id, u_id, staff_id
Table Description	It is used to store details.

Sr No.	**Field Name**	**Data Type**	**Constraints**	**Description**
1	r_id	int(10)	Primary key	Primary id of the table to store the id.
2	r_date	date	Not null	To store the of given date.
3	r_menu	varchar(10)	Not null	To store the menu for filling Reporting
4	r_created	datetime	Not null	To store the filled datetime.
5	menu_id	int(10)	Foreign key	Foreign key of the table to store the menu details.
6	u_id	int(10)	Foreign key	Foreign key of the table to store the user details.
7	staff_id	int(10)	Foreign key	Foreign key of the table to store the staff details

Table 3:

Table Name	Reporting_report			
Primary Key	r_id			
Foreign key	menu_id, u_id, staff_id			
Table Description	It is used to store reports of users			
Sr No.	**Field Name**	**Data Type**	**Constraints**	**Description**
1	r_id	int(10)	Primary key	Primary key of the report to store the report id.
2	r_date	int	Not null	To store the report of given date.
3	r_created	varchar(200)	Not null	To store the report created datetime.
4	menu_id	int(10)	Foreign key	Foreign key of the table to store the menu details.
5	u_id	int(10)	Foreign key	Foreign key of the table to store the user details.
6	staff_id	int(10)	Foreign key	Foreign key of the table to store the staff details

Table 4:

Table Name	menu_details			
Primary Key	menu_id			
Foreign key	u_id, staff_id			
Table Description	It is used to store exam results.			
Sr No.	**Field Name**	**Data Type**	**Constraints**	**Description**
1	menu_id	int(10)	Primary key	Primary key of the menu details table to store the menu id
2	menu_name	varchar(20)	Not null	To store the menu name.
3	menu_created	varchar(25)	Not null	To store the menu created datetime.
4	u_id	int(10)	Foreign Key	Foreign Key of the menu table to store the user id.
5	staff_id	int(10)	Foreign Key	Foreign Key of the menu table to store the staffid.

Table 5:

Table Name	User_details			
Primary Key	u_id			
Foreign key	u_id			
Table Description	It is used to store user details.			
Sr No.	**Field Name**	**Data Type**	**Constraints**	**Description**
1	u_id	int(15)	Primary key	Primary key of the user details table to store the user id.
2	s_profilepic	int(5)	Not null	To store the profile photo of the User
3	s_address	int(10)	Not null	To store the user address
4	s_mobile	varchar(15)	Not null	To store the users phone number.
5	u_id	int(10)	Foreign key	Foreign key of the user details table and used to store the user id.

Table 6:

Table Name	user_details
Primary Key	staff_id
Foreign key	u_id
Table Description	It is used to store staff details.

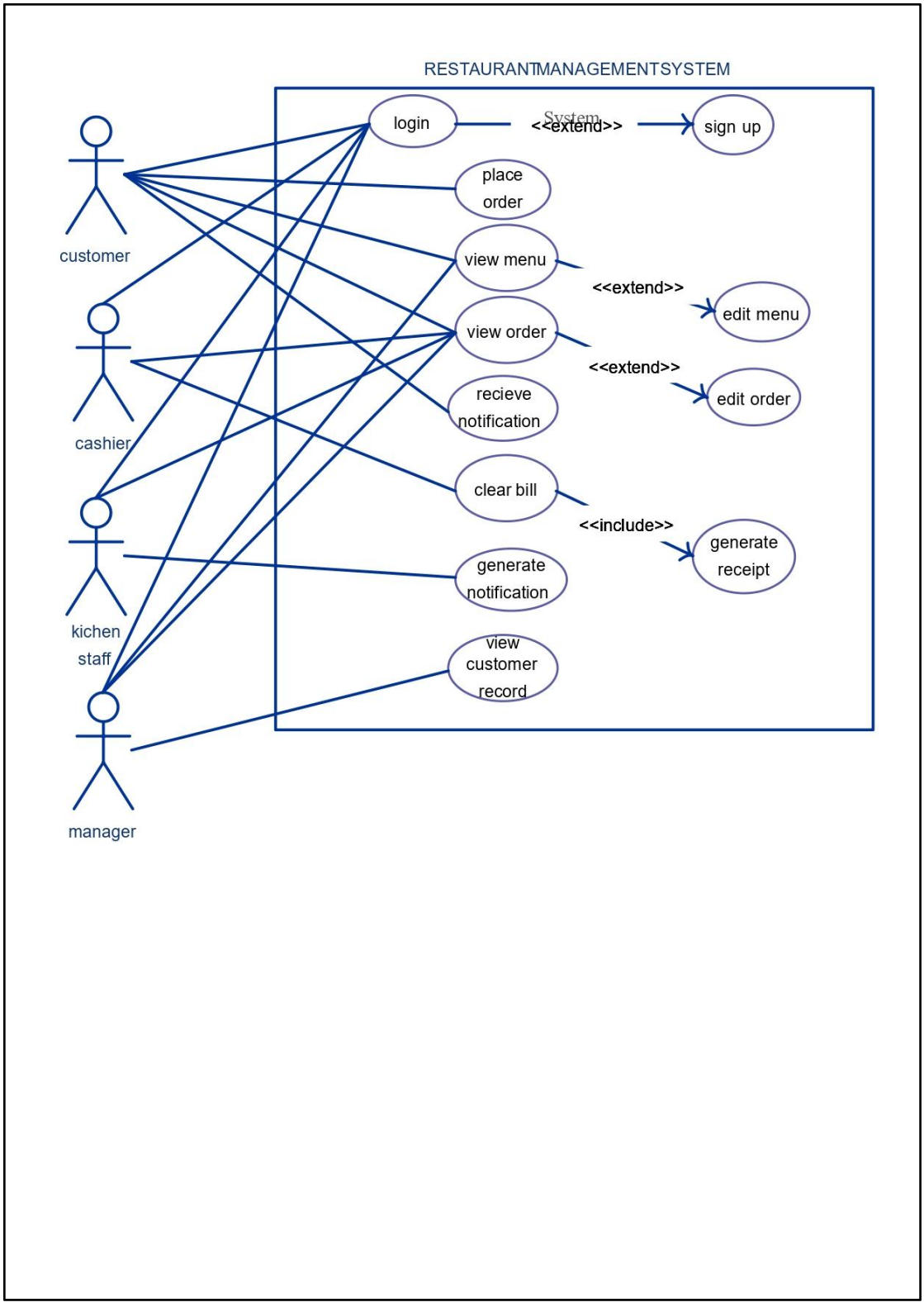
Sr No.	**Field Name**	**Data Type**	**Constraints**	**Description**
1	staff_id	int(15)	Primary key	Primary key of the staffdetails table to store the staffid.
2	t_name	varchar(50)	Not null	To store the staffname
3	t_address	varchar(50)	Not null	To store the users address
4	t_mobile	int(13)	Not null	To store the users phone number.
6	u_id	int	Not null	Foreign key of the user details table and used to store the user id.

3.4 Data Dictionary

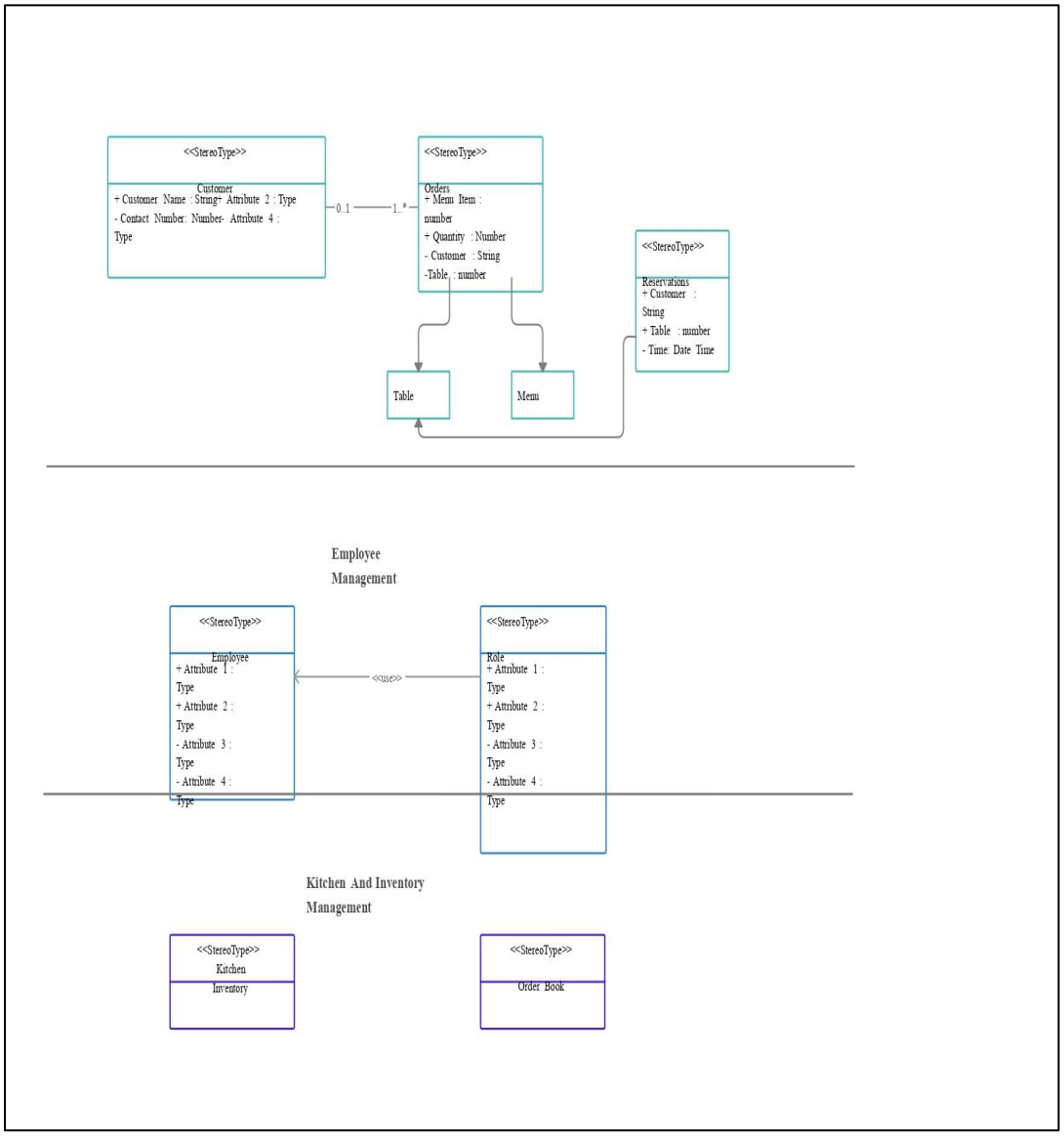
Field Name	Data Type	Constraints	Description
r_created	datetime	Not null	To store the filled datetime.
r_date	date	Not null	To store the of given date.
r_id	int(10)	Primary key	Primary id of the table to store the id.
r_menu	varchar(10)	Not null	To store the menu for filling Reporting
r_created	varchar(200)	Not null	To store the report created datetime.
r_date	int	Not null	To store the report of given date.
r_id	int(10)	Primary key	Primary key of the report to store the report id.
s_address	int(10)	Not null	To store the user address
u_id	int(15)	Primary key	Primary key of the user details table to store the user id.
u_id	int(10)	Foreign Key	Foreign Key of the menu table to store the user id.
u_id	int(10)	Foreign key	Foreign key of the table to store the user details.
u_id	int(10)	Foreign key	Foreign key of the table to store the user details.
s_mobile	varchar(15)	Not null	To store the users phone number.
s_profilepic	int(5)	Not null	To store the profile photo of the User
menu_created	varchar(25)	Not null	To store the menu created datetime.

menu_id	int(10)	Primary key	Primary key of the menu details table to store the menu id
menu_id	int(10)	Foreign key	Foreign key of the table to store the menu details.
menu_id	int(10)	Foreign key	Foreign key of the table to store the menu details.
menu_name	varchar(20)	Not null	To store the menu name.
t_address	varchar(50)	Not null	To store the users address
staff_id	int(15)	Primary key	Primary key of the staffdetails table to store the staffid.
staff_id	int(10)	Foreign Key	Foreign Key of the menu table to store the staffid.
staff_id	int(10)	Foreign key	Foreign key of the table to store the staffdetails
staff_id	int(10)	Foreign key	Foreign key of the table to store the staffdetails
t_mobile	int(13)	Not null	To store the users phone number.
t_name	varchar(50)	Not null	To store the staffname
u_address	varchar(50)	Not null	To store the address of user
u_email	varchar(20)	Not null	To store the email of the user.
u_fname	varchar(20)	Not null	To store the first name of the user.
u_gender	varchar(10)	Not null	To store the gender of user
u_id	int	Not null	Foreign key of the user details table and used to store the user id.
u_id	int(10)	Foreign key	Foreign key of the user details table and used to store the user id.
u_id	int(5)	Primary key	Primary id of the authorize user table to store the user id.
u_lname	varchar(20)	Not null	To store the last name of the user.
u_password	varchar(50)	Not null	To store the password.
u_username	varchar(25)	Not null	To store the username of the user.

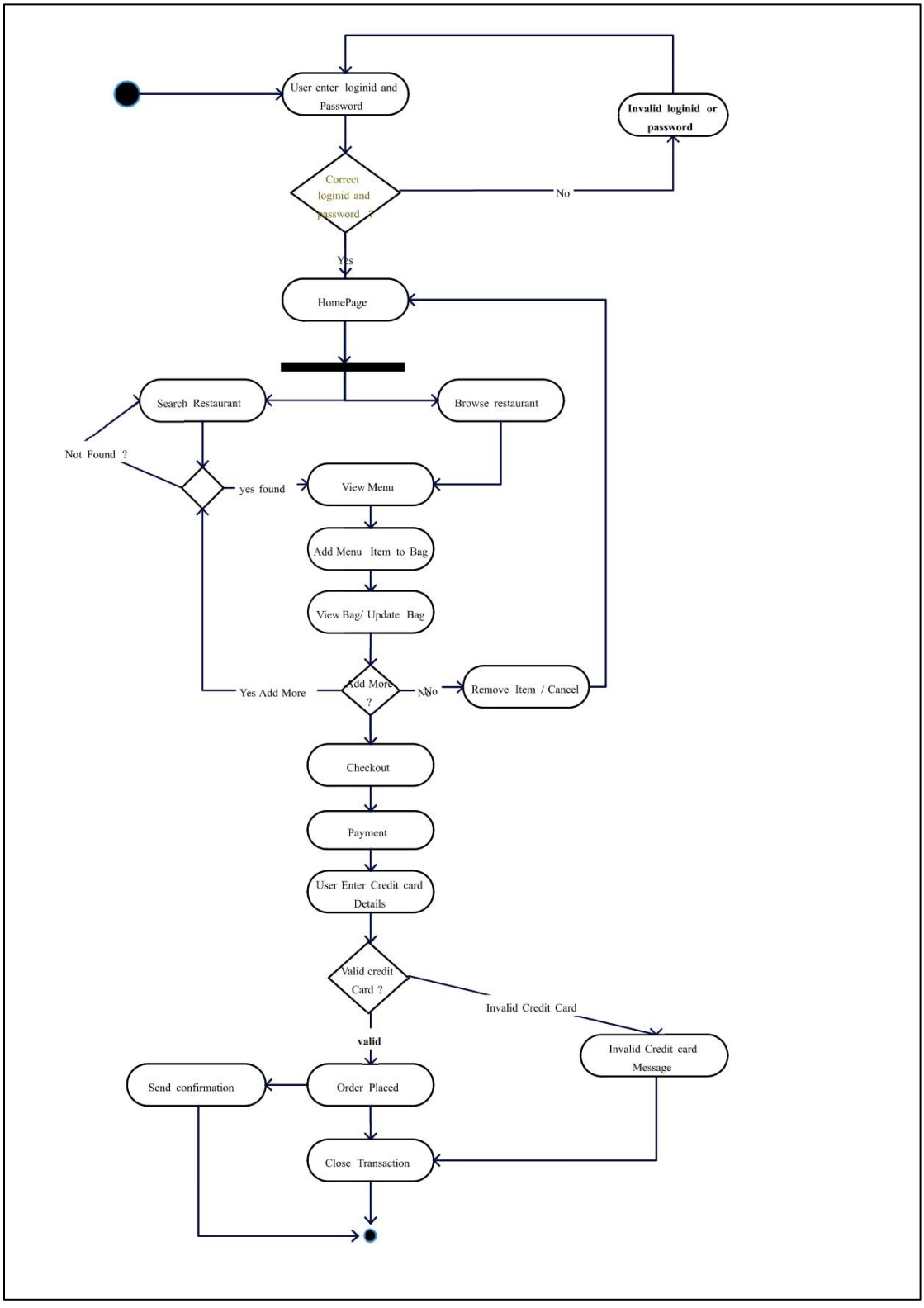
3.5 Use Case Diagram:



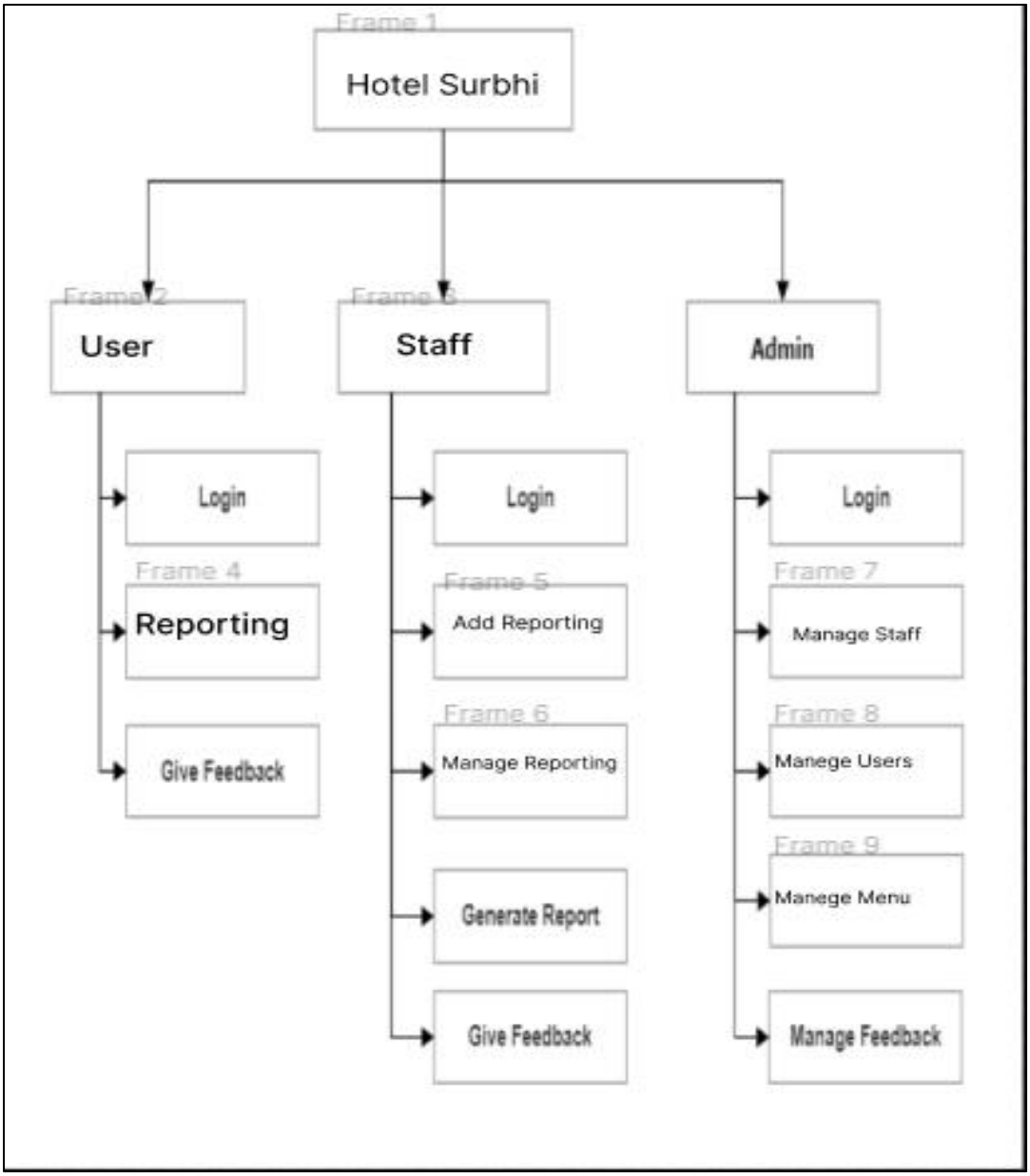
3.6 Class Diagram:



3.7 Activity Diagram:

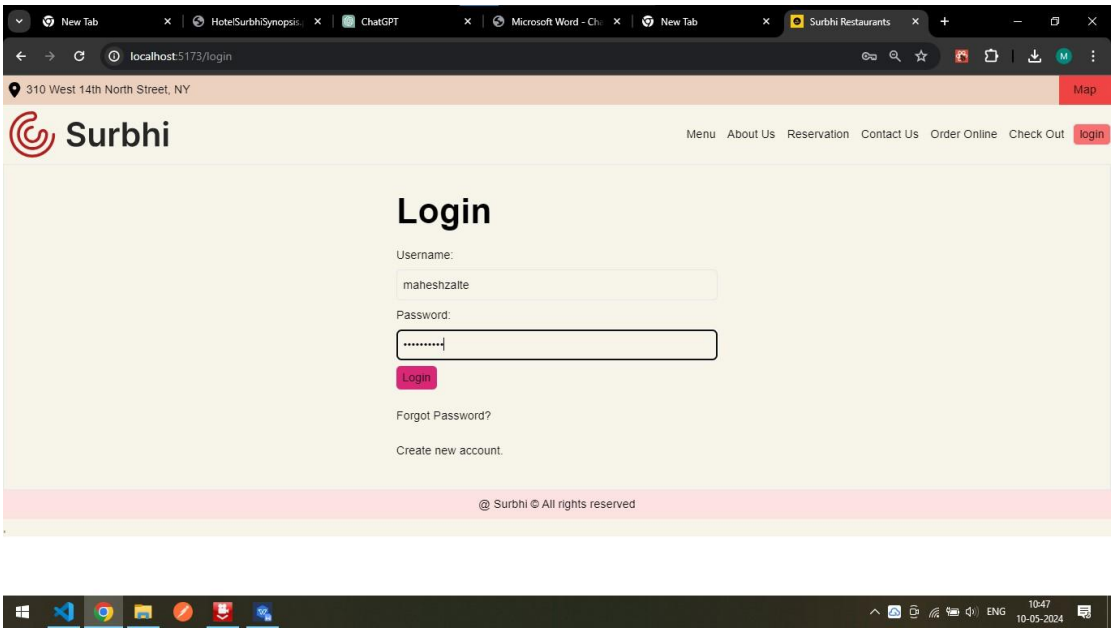


3.8 Module Hierarchy Diagram:

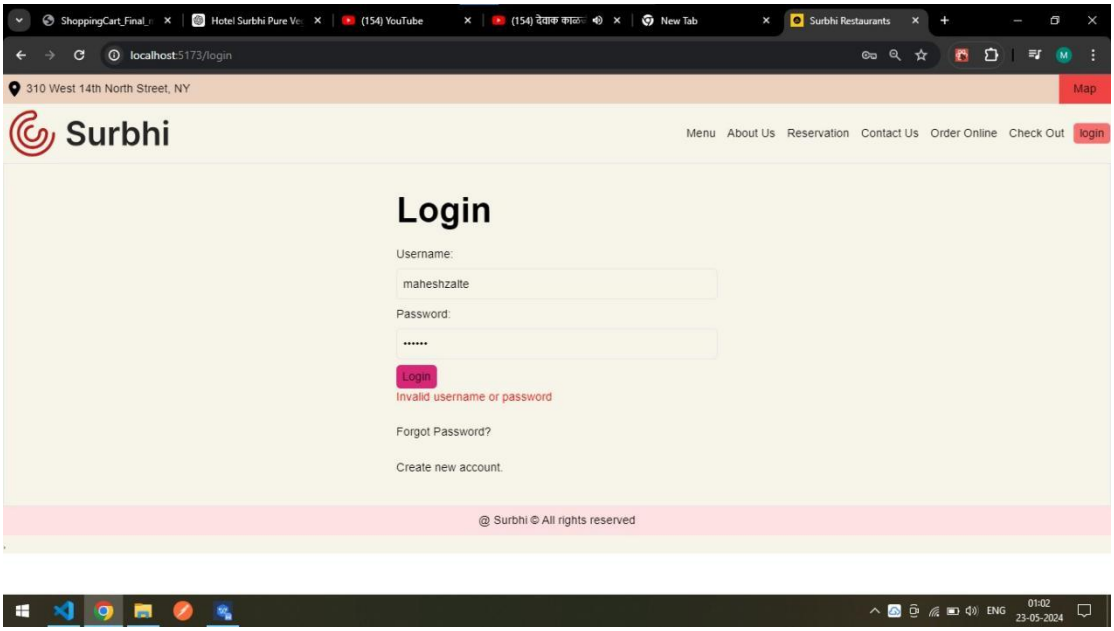


3.9 Sample Input and Output Screens:

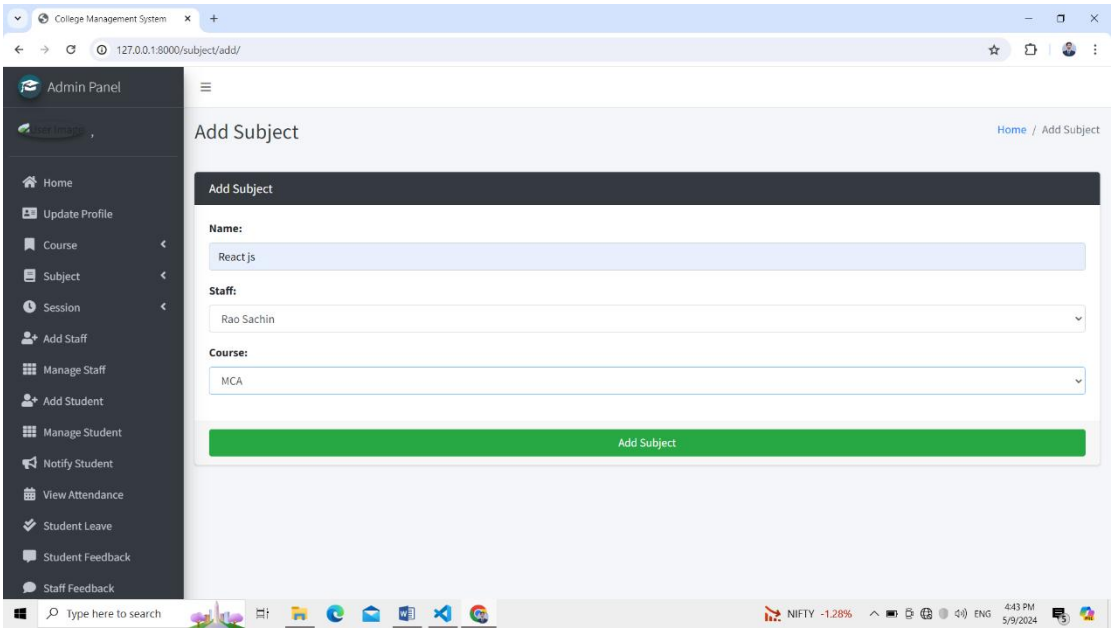
Login Page:- A login page serves secure gateway that allows users to access a system by providing their credentials.



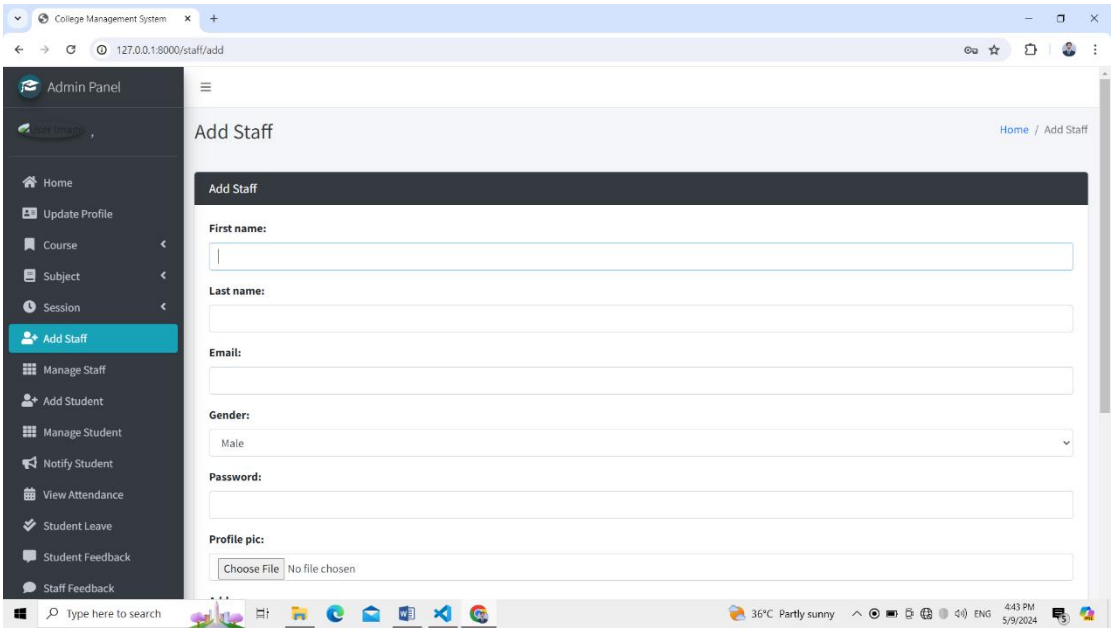
Login Page Validation:- To check whether user entered username in correct format.



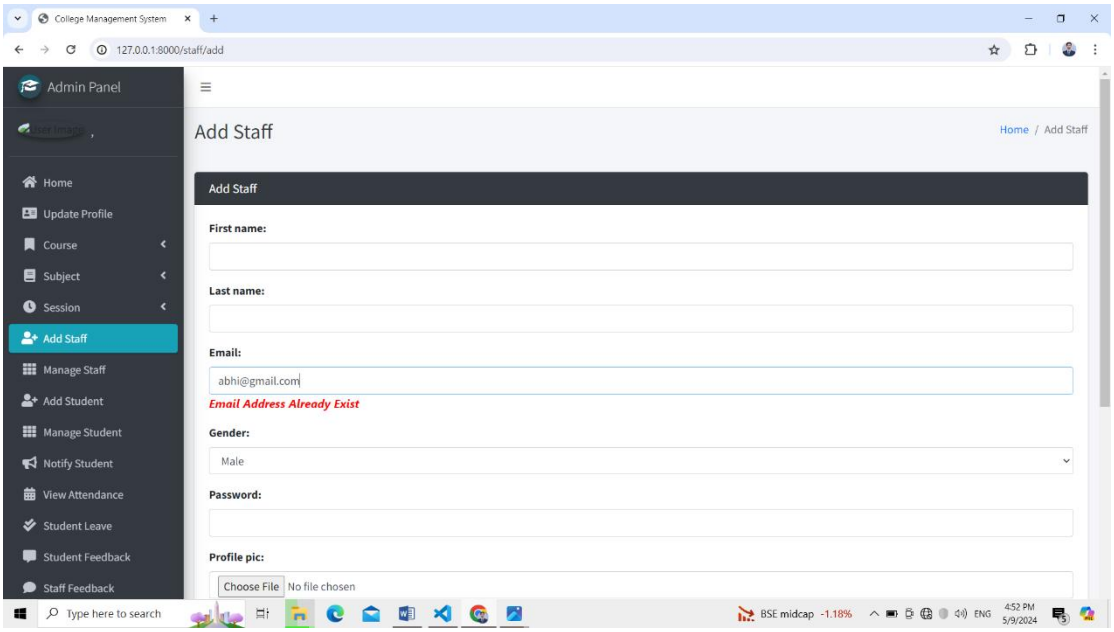
Adding menu and assigning staff:



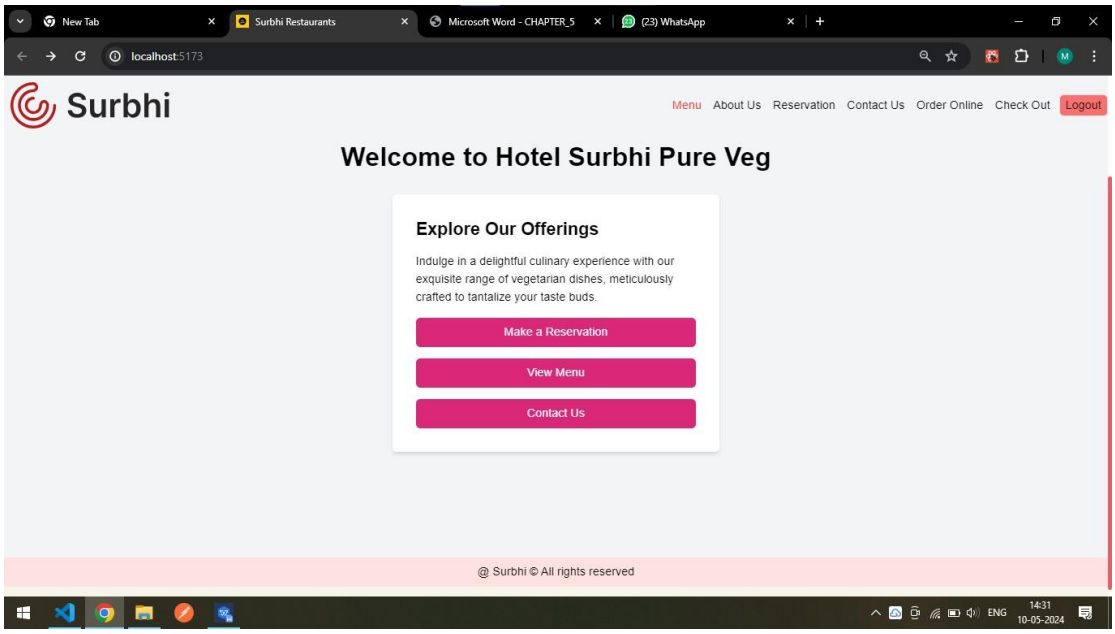
Adding user & Staff:- Adding users and staff to system



Checking Email:- System doesn't allow duplicate entry in email.



User Dashboard: - An user dashboard is a user interface that provides user with organized overview of essential information and checking his ratio.



Chapter 4
CODING

Login Page (Login.jsx)

```
iimport React, { useState } from 'react';
import { Link, useNavigate } from 'react-router-dom';
import { useAuth } from '../Store/auth';

const Login = () => {
  const [username, setUsername] = useState('');
  const [password, setPassword] = useState('');
  const [error, setError] = useState('');

  const navigate = useNavigate();
  const { StoreToken } = useAuth()

  const handleLogin = async (e) => {
    e.preventDefault();

    try {
      const response = await fetch(`http://localhost:3000/login`, {
        method: 'POST',
        headers: {
          'Content-Type': 'application/json',
        },
        body: JSON.stringify({ username, password }),
      });

      if (response.ok) {
        const data = await response.json();

        // Set Token in to local Storage.
        StoreToken(data.token)
        console.log('Login successful:', data.token);

        // Here you can handle successful login, e.g., redirect to dashboard
        navigate('/')
      }
      else setError('Invalid username or password');


    } catch (error) {
      console.error('Login error:', error.message);
      setError('Invalid username or password');
    }
  };
```

```
 const handleForgotPassword = () => {
   console.log('Forgot password clicked');
   // Here you can implement the logic for forgot password functionality
 };

 return (
   <div className='flex flex-col border items-center justify-center bg-transparent py-
10'>
     <div className='grid gap-6 sm:w-[440px]'>
       <h2 className=' text-5xl font-bold bg-transparent'>Login</h2>
       <form onSubmit={handleLogin} className='bg-transparent'>
         <div className='flex flex-col bg-transparent'>
           <label htmlFor="username" className='bg-transparent'>Username:</label>
           <input
             className='p-2 border my-2 rounded-md bg-transparent'
             type="text"
             id="username"
             placeholder='Username'
             value={username}
             onChange={(e) => setUsername(e.target.value)}
             required
           />
         </div>
         <div className='flex flex-col bg-transparent'>
           <label htmlFor="password" className='bg-transparent'>Password:</label>
           <input
             className='p-2 border my-2 rounded-md bg-transparent'
             type="password"
             id="password"
             placeholder='Password'
             value={password}
             onChange={(e) => setPassword(e.target.value)}
             required
           />
         </div>
         <div className='bg-transparent'>
           <button type="submit" className=' bg-pink-600 px-2 py-1 rounded-
md'>Login</button>
         </div>
         {error && <div style={{ color: 'red' }}>{error}</div>}
       </form>
       <div className='bg-transparent'>
         <button onClick={handleForgotPassword}>Forgot Password?</button>
       </div>
       <div className='bg-transparent'>
         <Link to='/register'>Create new account.</Link>
       </div>
```

```
    </div>
   </div>
  );
};

export default Login;
```

Reservation (Reservation.jsx)

```
import React, { useState } from 'react';

 const Reservation = () => {
 const [name, setName] = useState('');
 const [email, setEmail] = useState('');
 const [phone, setPhone] = useState('');
 const [date, setDate] = useState('');
 const [time, setTime] = useState('');
 const [error, setError] = useState('');

 const handleSubmit = (e) => {
  e.preventDefault();
  // Validation logic can be added here
  if (!name || !email || !phone || !date || !time) {
   setError('Please fill out all fields');
   return;
  }
  // Reservation submission logic goes here
  console.log('Reservation submitted:', { name, email, phone, date, time });
  // Clear form fields after submission
  setName('');
  setEmail('');
  setPhone('');
  setDate('');
  setTime('');
  setError('');
 };

 return (
  <div className="flex items-center justify-center h-screen">
   <div className="bg-gray-100 p-8 rounded-md shadow-md w-full max-w-md">
    <h2 className="text-2xl font-bold mb-4">Make a Reservation</h2>
    <form onSubmit={handleSubmit}>
     <div className="mb-4">
      <label htmlFor="name" className="block mb-2">Name:</label>
      <input
       type="text"
       id="name"
       value={name}
       onChange={(e) => setName(e.target.value)}
```

```
    className="w-full p-2 border rounded-md"
    required
   />
  </div>
  <div className="mb-4">
   <label htmlFor="email" className="block mb-2">Email:</label>
   <input
    type="email"
    id="email"
    value={email}
    onChange={(e) => setEmail(e.target.value)}
    className="w-full p-2 border rounded-md"
    required
   />
  </div>
  <div className="mb-4">
   <label htmlFor="phone" className="block mb-2">Phone:</label>
   <input
    type="tel"
    id="phone"
    value={phone}
    onChange={(e) => setPhone(e.target.value)}
    className="w-full p-2 border rounded-md"
    required
   />
  </div>
  <div className="mb-4">
   <label htmlFor="date" className="block mb-2">Date:</label>
   <input
    type="date"
    id="date"
    value={date}
    onChange={(e) => setDate(e.target.value)}
    className="w-full p-2 border rounded-md"
    required
   />
  </div>
  <div className="mb-4">
   <label htmlFor="time" className="block mb-2">Time:</label>
   <input
    type="time"
    id="time"
    value={time}
    onChange={(e) => setTime(e.target.value)}
    className="w-full p-2 border rounded-md"
    required
   />
  </div>
```

```
        {error && <div className="text-red-500 mb-4">{error}</div>}
        <button type="submit" className="bg-pink-600 text-white py-2 px-4 rounded-
md">Submit Reservation</button>
      </form>
    </div>
  </div>
 );
};

export default Reservation;
```

Chapter 5
TESTING

5.1 Testing Strategy:

TESTING:

After finishing the development of any computer-based system the next complicated time-consuming work is system testing. During the time of testing, only the development company can know, how far the user requirements have been met, and so on. Following are some of the testing methods applied to this effective project:

SPECIFICATION TESTING:

It defines system requirements for login, profile management, menu management, session management, staff management, notification, viewing, user leave, and feedback functionality. It evaluates system performance against these predefined specifications to ensure compliance.

MODULE-LEVEL TESTING:

It test each module individually to identify and rectify errors without affecting other modules. Test login functionality, profile management, menu management, session management, staff management, notification system, viewing, leave management, and feedback system separately.

UNIT TESTING:

It verifies each unit of source code for functionality and usability. Test individual functions within modules to ensure they perform as expected. Test boundary cases and data integrity within local data structures. Unit testing focuses on verifying the effort on the smallest unit of the software module.

INTEGRATION TESTING:

It combines unit-tested modules to form larger aggregates. Test interfaces between modules to uncover integration errors. Validate data flow between modules. In the Integration testing (Modules) the data can be tested across an interface. Conducting tests to uncover errors associated with interring while integration testing is a technique for constructing a program structure.

VALIDATION TESTING:

It validates each module's functionality against user requirements. It performs black-box testing to verify system behaviour. Ensure the system functions as expected in various scenarios. In this, the majority of the validation is done during the data entry operation where there is a maximum possibility of entering wrong data other validation will be performed in all processes where correct details and data should be entered to get the required results.

RECOVERY TESTING:

It test the system's ability to recover from hardware failures and crashes. Verify that recovery mechanisms function as intended and provide confirmation of recovery. It detects where the failure of the software is occurred and provides many ways to recover the failure module and it sends the confirmation about the recovery. In the process of recovery testing do not be confused with reliability testing because it tries to discover the point at which failure occurs.

SECURITY TESTING:

Verify authentication, integrity, availability, confidentiality, non-repudiation, and authorization mechanisms. Test for vulnerabilities offensive penetrations. Ensure data protection mechanisms are in place and effective. It is a procedure to verify that the system maintains and protects data functionality as intended.

PERFORMANCE TESTING:

The usefulness of computer applications, networks, and devices can be decided by performance testing. In the context of an integrated testing system, it checks the performance of the software at runtime. Performance testing can be also used for

calculating the characteristics like reliability, scalability, and interoperability. It is generally used together with stress testing.

BLACKBOX TESTING:

It focuses on the functional requirement of the software. It enables to derive of input states that will completely implement every one useful requirement for a program. Black box testing is used for finding errors in the categories shown below: It detects the wrong or misplaced functions. Poorly developed interfaces. It is also used for finding errors in data structures or external database access and performance errors.

5.2 Unit Test Plan

Unit Test Plan for Management System:

Test Plan Overview:

The purpose of this test plan is to ensure the functionality and accuracy of the user System. The focus will be on testing individual units or components of the system, including login, profile management, menu management, session management, staff management, notification, viewing, user leave, and user feedback functionalities. The tests will cover both positive and negative scenarios to ensure the system handles various situations appropriately.

Reservation:

Test Case 1: Verify successful reservation with valid details.

Expected Outcome: Reservation confirmation message is displayed.

Test Case 2: Verify error message display for invalid reservation details (e.g., invalid date or time).

Expected Outcome: Appropriate error message is displayed.

Test Case 3: Verify error message display for blank reservation fields.

Expected Outcome: Error message prompts to fill all required fields.

Test Case 4: Verify reservation system handles duplicate reservations gracefully.

Expected Outcome: Error message or prompt to update the existing reservation.

Update Profile:

Test Case 1: Verify successful update of user profile information.

Expected Outcome: Profile updated confirmation message is displayed.

Test Case 2: Verify successful change of password.

Expected Outcome: Password change confirmation message is displayed.

Test Case 3: Verify error message display for incomplete profile update.

Expected Outcome: Error message prompts to fill all required fields.

Test Case 4: Verify error message display for incorrect current password during password change.

Expected Outcome: Error message indicating incorrect current password.

Add Menu Item:

Test Case 1: Verify successful addition of a new menu item.

Expected Outcome: New menu item is listed and available for customers.

Test Case 2: Verify error message display for adding a duplicate menu item.

Expected Outcome: Error message indicating item already exists.

Test Case 3: Verify error message display for incomplete menu item details during addition.

Expected Outcome: Error message prompts to fill all required fields.

Manage Menu Item:

Test Case 1: Verify successful update of menu item details.

Expected Outcome: Menu item details are updated and reflected in the menu.

Test Case 2: Verify successful deletion of a menu item.

Expected Outcome: Menu item is removed from the menu list.

Add Staff:

Test Case 1: Verify successful addition of a new staff member.

Expected Outcome: New staff member is listed and available for scheduling.

Test Case 2: Verify error message display for adding a staff member with incomplete details.

Expected Outcome: Error message prompts to fill all required fields.

Test Case 3: Verify error message display for adding a duplicate staff member.

Expected Outcome: Error message indicating staff member already exists.

Manage Staff:

Test Case 1: Verify successful update of staff details.

Expected Outcome: Staff details are updated and reflected in the system.

Test Case 2: Verify successful deletion of a staff member.

Expected Outcome: Staff member is removed from the staff list.

Notify Customers:

Test Case 1: Verify successful notification sent to all customers.

Expected Outcome: Notification is sent and confirmation message is displayed.

Test Case 2: Verify successful notification sent to individual customers.

Expected Outcome: Notification is sent and confirmation message is displayed.

View Reservations:

Test Case 1: Verify successful viewing of reservation records for a particular date.

Expected Outcome: Reservation records are displayed.

Test Case 2: Verify appropriate error message display for viewing reservations when no data is available.

Expected Outcome: Error message indicating no reservations found.

Customer Feedback:

Test Case 1: Verify successful submission of feedback by a customer.

Expected Outcome: Feedback submission confirmation message is displayed.

Test Case 2: Verify successful viewing of feedback by staff.

Expected Outcome: Feedback is displayed in the staff portal.

Test Case 3: Verify error message display for submitting feedback with empty content.

Expected Outcome: Error message prompts to fill feedback content.

Online Ordering:

Test Case 1: Verify successful submission of an online order.

Expected Outcome: Order confirmation message is displayed.

Test Case 2: Verify error message display for incomplete order details.

Expected Outcome: Error message prompts to fill all required fields.

Test Case 3: Verify error message display for ordering out-of-stock items.

Expected Outcome: Error message indicating item is out of stock.

Payment Processing:

Test Case 1: Verify successful payment processing for an order.

Expected Outcome: Payment confirmation message is displayed.

Test Case 2: Verify error message display for payment failure due to invalid card details.

Expected Outcome: Error message indicating invalid payment details.

Test Case 3: Verify error message display for payment failure due to insufficient funds.

Expected Outcome: Error message indicating payment failure.

Leave Request (for staff):

Test Case 1: Verify successful submission of leave request by staff.

Expected Outcome: Leave request submission confirmation message is displayed.

Test Case 2: Verify successful approval or rejection of leave request by administrator.

Expected Outcome: Leave request status is updated and notification is sent.

Test Case 3: Verify error message display for submitting leave request with incomplete details.

Expected Outcome: Error message prompts to fill all required fields.

Test Execution:

Each test case should be executed individually and independently. Test data should be prepared specifically for each test case to cover different scenarios. Any test environment setup or prerequisites should be clearly defined and met before executing the tests.

Test Reporting:

Each test case should be reported with a clear description of the test scenario, expected results, and actual results. Any deviations or issues encountered during testing should be documented along with steps to reproduce the problem, if applicable:

5.3 Acceptance Plan:

Purpose:

The purpose of this acceptance plan is to outline the criteria and steps for validating the functionality of the user System. The system should meet the requirements and expectations of the stakeholders, including accurate tracking, efficient user management, notification delivery, and feedback handling.

Acceptance Criteria:

The following acceptance criteria will be used to determine the successful acceptance of the user acceptance system:

- All specified features and functionality are implemented and working as intended.
- The system accurately tracks user for various sessions and menus.
- User accounts can be created, managed, and assigned appropriate roles(admin,staff, User).
- Notification delivery is reliable an timely, both for individual users and group notifications.

- user feedback submissions are handled correctly and accessible to staff for review.
- The system provides accurate reports on Reporting, leave requests, and feedback submissions.

Test Environment:

The acceptance testing will be conducted in an environment that replicates the production setup of the user System. This includes the necessary hardware, software, and network configurations.

Test Approach:

The acceptance testing will be performed using a combination of manual testing and automated testing tools. The test cases will cover different scenarios to ensure comprehensive coverage of the system's functionality.

Test Scenarios and Test Cases:

The acceptance test scenarios and test cases are as follows:

Login:

Test Scenario: Succesful login with valid credentials.

Test Case: Verify that users can log in with correct username and password.

Tracking:

Test Scenario: Accurate tracking of user Reporting.

Test Case: Verify that records are updated correctly for each session and menu.

User Management:

Test Scenario: Creation and management of user accounts.

Test Case: Verify that admins can create, update, and delete user accounts.

Notification Delivery:

Test Scenario: Reliable delivery of notifications.

Test Case: Verify that notifications are delivered promptly to the intended recipients.

Feedback Handling:

Test Scenario: Submission and review of user feedback.

Test Case: Verify that users can submit feedback and staff can review and respond to it.

Test Execution and Reporting:

The acceptance tests will be executed according to the defined test scenarios and test cases. The results will be documented, and any deviations from the expected results will be reported. A test report will be generated, including the test execution status and any issues or defects encountered during testing.

Sign-off:

The acceptance plan will be considered complete when all defined test scenarios and test cases have been executed, and the system meets the acceptance criteria. The stakeholders will review the test results and provide their approval and sign-off for the user System.

Chapter 6

LIMITATIONS OF PROPOSED SYSTEM

While the proposed user system offers numerous benefits and functionality, it may also have some limitations that should be considered

Technical Limitations:

The effectiveness of the system relies on the stability and performance of the underlying technologies and infrastructure. Technical issues such as server downtime, network failures, or software bugs may impact the system's availability and performance.

Dependency on Internet Connectivity:

Since the system is online-based, it requires a stable internet connection for users to access and utilize its features. In areas with limited or unreliable internet connectivity, users may face difficulties in accessing the system, leading to potential disruptions in tracking and management.

Learning Curve and User Adoption:

Introducing a new system may require users, including administrative staff, users, and users, to adapt to a new interface and workflow. This learning curve may initially slow down operations and face resistance from users who are not familiar with the system.

Data Security and Privacy Concerns:

Handling sensitive user data, including records and personal information, requires robust security measures to protect against unauthorized access and data breaches.

Implementing strong encryption, access controls, and compliance with data protection regulations are essential to mitigate these concerns.

Integration Challenges:

Integrating the user system with existing school management systems or databases may present challenges in terms of data synchronization, compatibility, and technical dependencies. Ensuring seamless integration with other systems is crucial for maintaining data consistency and workflow efficiency.

Data Accuracy and Integrity:

Maintaining accurate and up-to-date records is essential for monitoring user progress and compliance. However, inaccuracies in data entry, system errors, or inconsistencies in recording may affect the reliability and integrity of the data, leading to potential challenges in analysis and reporting.

Chapter 7

Proposed Enhancements

Based on the proposed user system, here are some potential enhancements that could further improve its functionality and user experience:

Mobile Application:

Develop a mobile application alongside the web-based system to provide users, users, and staff with a convenient way to access Reporting-related information, receive notifications, and manage their accounts on their smart phones or tablets.

Gamification and Reward System:

Implement a gamification and rewards system to encourage user engagement and regular Reporting. Offer rewards, badges, or recognition for users with good records or active participation in class activities.

Customization Options:

Provide customization options for tracking, such as allowing users to define specific criteria for marking and customizing reports based on institutional requirements.

Advanced Reporting and Analytics:

Enhance the reporting capabilities by including advanced analytics and insights into trends, user performance, and class participation. Provide actionable insights to users and administrators for identifying patterns and addressing issues effectively.

Chapter 8

CONCLUSION

In conclusion, the proposed user System aims to streamline tracking and management processes in Surbhi Hotel. It offers benefits such as accurate recording, improved communication between users and staff, enhanced reporting capabilities, and data-driven insights into trends. However, it is essential to consider the limitations of the system, such as technical challenges, user adoption, data security concerns, and integration complexities. To enhance the system further, various suggestions can be implemented, including.

Developing a mobile application alongside the web-based system to provide users, users, and staff with convenient access to Reporting-related information, notifications, and account management features on their smartphones or tablets.

Improving search functionality to allow users and administrators to easily find specific records, users, or sessions based on various criteria such as date, menu, user ID, or class section, and including filters to refine search results and facilitate data analysis.

Integrating the user system with existing learning management systems or academic platforms to synchronize user data, class schedules, and course enrollments, ensuring seamless data exchange and improving overall system efficiency.

Implement a gamification and rewards system to encourage user engagement and regular Reporting. Offer rewards, badges, or recognition for users with good records or active participation in class activities.

Chapter 9

Bibliography

References:

/1/ Miva 2011. The History of Ecommerce: How Did It All Begin? Accessed 15.07.2017.
http://www.miva.com/blog/the-history-of-ecommerce-how-did-it-all-begin/

/2/Arline, K. 2015. What is E-commerce?
Accessed 12.07.2017.
http://www.businessnewsdaily.com/4872-what-is-e-commerce.htm

/3/Tutorialspoint 2017. Java Tutorial. Accessed 17.07.2017.
https://www.tutorialspoint.com/java/index.html

/4/ Stackoverflow 2017. What is the difference between JDK and JRE? Accessed
17.07.2017
https://stackoverflow.com/questions/1906445/

/5/ Evans, I. 2012. Differences between Java EE and Java SE - Your First Cup: An
Introduction to the Java EE Platform. Accessed 17.07.2017.
http://docs.oracle.com/javaee/6/firstcup/doc/gkhoy.html

/6/ Tutorialspoint 2017. HTML Tutorial. Accessed 17.07.2017
https://www.tutorialspoint.com/html/

/7/ W3Schools 2017. CSS Tutorial. Accessed 17.07.2017
https://www.w3schools.com/css/

8/ Oracle. What is a Servlet – The Java EE 5 Tutorial. Accessed 18.07.2017.
http://docs.oracle.com/javaee/5/tutorial/doc/bnafe.html

/9/ Oracle. JavaServer Pages Technology. Accessed 19.07.2017.
http://www.oracle.com/technetwork/java/javaee/jsp/index.html

/10/ Tutorialspoint 2017. Eclipse Tutorial. Accessed 21.07.2017
https://www.tutorialspoint.com/eclipse/

/11/ The Apache Software Foundation 2017. Apache Tomcat. Accessed 22.07.2017.
https://tomcat.apache.org/

Video References:

• https://www.researchgate.net/publication/
• www.youtube.com/watch?v=pKMqJAc6wYw
• www.youtube.com/watch?v=kh4W7z2FoXs&t=971

Websites:

• Google:
www.google.com
• Javacodegeeks:
https://www.javacodegeeks.com/
• Stack Overflow:
https://stackoverflow.com/
• GitHub:
https://github.com/
• Tutorials Point:
https://www.tutorialspoint.com/index.html

Chapter 10

User Manual

User Manual: Surbhi Pure Veg Restaurant

Table of Contents

1. Introduction

Welcome to the Surbhi Pure Veg Restaurant user manual. This guide will help you navigate and use the online ordering system efficiently.

2. System Requirements

A device with internet access (PC, tablet, smartphone).

A modern web browser (Chrome, Firefox, Safari, Edge).

Active internet connection.

3. Installation Guide

No installation is required. The system is web-based and can be accessed via the internet.

4. Configuration

Ensure your browser is up-to-date and cookies are enabled for optimal performance.

5. Navigation Guide

Home Page

- Access the restaurant's main features and latest offers.
- View daily specials and new menu items.

Menu Page

- Browse through different categories of vegetarian dishes.
- View item details, including ingredients and prices.

Admin Dashboard

- Access restricted to administrators.
- Manage menu items, inventory, and orders.

Cart Page

- Review selected items before checkout.
- Update item quantities or remove items.

Profile Page

- View and update personal information.
- Check order history and track current orders.

6. User Operations

User Registration

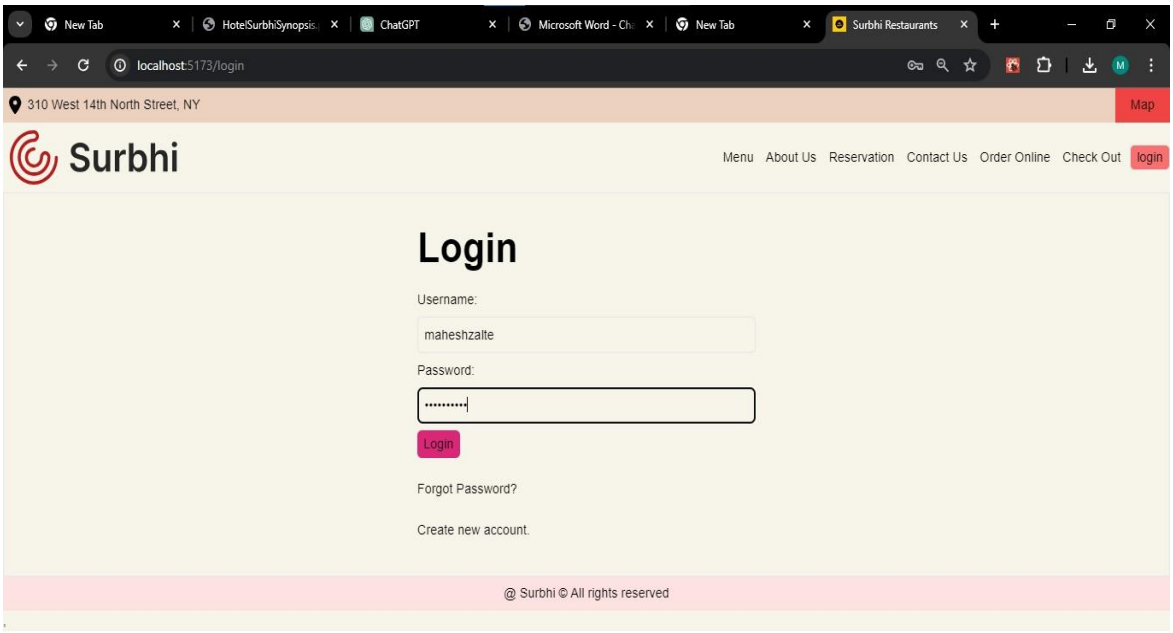
- Click on the "Register" button on the home page.
- Fill in the required details and submit the form.

- Verify your email to complete the registration process.

User Login

- Click on the "Login" button.

Enter your username and password to access your account.



Browsing Menu

- Navigate to the "Menu" page.
- Browse items by category or use the search bar to find specific dishes.