BUS TICKET RESERVATION SYSTEM

Project report submitted
in partial fulfillment of the requirement for award of the degree of

Bachelor of Technology
in
Computer Science & Engineering

By

MALLELA GNANESWAR (21UECM0502)
G LEELA PRASANNA KUMAR (21UECM0085)
K VISHNU VARDHAN (21UECS0297)

10211CS212 - WEB AND MOBILE APPLICATION DEVELOPMENT

SUMMER 2023-2024

Under the guidance of
Minu Inba Shanthini Watson Benjamin,M.E.,
ASSISTANT PROFESSOR

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
SCHOOL OF COMPUTING

VEL TECH RANGARAJAN DR. SAGUNTHALA R&D INSTITUTE OF SCIENCE AND TECHNOLOGY

(Deemed to be University Estd u/s 3 of UGC Act, 1956)
Accredited by NAAC with A++ Grade
CHENNAI 600 062, TAMILNADU, INDIA

October,2023

CERTIFICATE

It is certified that the work contained in the project report titled ”BUS TICKET RESERVATION SYSTEM” by ”MALLELA GNANESWAR (21UECM0502), G LEELA PRASANNA KUMAR (21UECM0085), K VISHNU VARDHAN (21UECS0297)” has been carried out under my supervision and that this work has not been submitted elsewhere for a degree.

Signature of Supervisor
Minu Inba Shanthini Watson Benjamin
Assistant Professor
Computer Science & Engineering
School of Computing
Vel Tech Rangarajan Dr.Sagunthala R&D
Institute of Science & Technology
October, 2023

Signature of Head of the Department
Computer Science & Engineering
School of Computing
Vel Tech Rangarajan Dr. Sagunthala R&D
Institute of Science & Technology
October, 2023

Signature of the Dean
Dr. V. Srinivasa Rao
Professor & Dean
Computer Science & Engineering
School of Computing
Vel Tech Rangarajan Dr. Sagunthala R&D
Institute of Science & Technology
October, 2023

DECLARATION

We declare that this written submission represents my ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

(MALLELA GNANESWAR)

Date: / /

(G LEELA PRASANNA KUMAR)

Date: / /

(K VISHNU VARDHAN)

Date: / /

6

APPROVAL SHEET

This project report entitled BUS TICKET RESERVATION SYSTEM by MALLELA GNANESWAR (21UECM0502), G LEELA PRASANNA KUMAR (21UECM0085), K VISHNU VARDHAN (21UECS0297) is approved for the degree of B.Tech in Computer Science & Engineering.

Examiners **Handling faculty**

Minu Inba Shanthini Watson Benjamin M.E.,

Date: / /

Place:

ABSTRACT

The Bus Ticket Reservation System is an innovative and user-friendly online platform designed to simplify the process of booking and managing bus tickets.This system is intended to cater to the needs of both passengers and bus operators,providing a seamless and efficient solution for bus ticket booking, seat selection, payment processing, and ticket management. The primary objective of the Bus Ticket Reservation System is to eliminate the hassle associated with traditional ticket booking methods. Passengers can access the system through a web-based or mobile application, allowing them to search for available bus routes, view schedules, choose seats, and make secure online payments.Additionally, the system provides real-time information on bus availability and pricing, ensuring transparency and convenience for passengers.

Keywords:
User Account,Security, Payment Methods, Transaction, Payment process.

LIST OF FIGURES

LIST OF FIGURES

LIST OF ACRONYMS AND ABBREVIATIONS

CSS	Cascading Stlye Sheet
HTML	Hyper Text Markup Language
IDE	Integrated Development Environment
JS	Java Script
PHP	Hypertext Preprocessor
SQL	Structured Query Language

TABLE OF CONTENTS

Chapter 1

INTRODUCTION

1.1 Introduction

A Bus Ticket Reservation System simplifies the process of booking bus tickets by providing passengers with user-friendly and convenient options. Passengers can easily access the system online through websites or mobile applications, allowing them to check bus schedules, select routes, and choose their preferred seats. This system offers flexibility, enabling passengers to plan their journeys with options for one-way, round-trip, or multi-stop travel. It also provides various payment methods, including credit/debit card payments and digital wallets, making transactions hassle-free. After booking, passengers receive a confirmation with essential details, and they can even cancel bookings if plans change, subject to the system's refund policy. For bus operators, the system offers management tools for inventory control, scheduling, and financial reporting, while passengers benefit from improved travel experiences with options for digital or physical tickets and convenient customer support. In summary, a Bus Ticket Reservation System enhances both the booking process and the overall travel experience.

1.2 Aim of the project

The Bus Ticket Reservation System leverages online technology to provide passengers with a convenient and efficient platform for booking bus tickets. This system not only eliminates the need for physical visits to booking counters but also offers real-time updates on route availability and seat selection, empowering passengers to make informed travel choices.

1.3 Project Domain

The project domain for the Bus Ticket Reservation System primarily falls within the realm of Transportation and Travel Management. Specifically, it focuses on improving the ticket booking and management process for bus transportation services.

- Multi-Modal Transportation Integration: Expand the system to include not only bus ticket reser-

vations but also integrate other modes of transportation, such as trains, flights, and taxis. This would create a comprehensive transportation booking platform.

- Geo-Location and Real-Time Tracking: Implement GPS and real-time tracking for buses, enabling passengers to track the exact location and estimated arrival time of their booked buses.
- Seat Selection and Virtual Bus Layouts: Allow passengers to choose their preferred seats by providing a virtual bus layout during the booking process, complete with seat availability status.
- Integration with Mobile Wallets: Enable users to make payments through popular mobile wallets like Apple Pay, Google Pay, or PayPal for a seamless and secure booking and payment experience.
- Route Planning and Optimization: Incorporate route planning algorithms to suggest the most efficient and cost-effective bus routes for passengers, considering factors like distance, traffic, and travel time.
- User Reviews and Ratings: Implement a system for passengers to le

1.4 Scope of the Project

The scope of the Bus Ticket Reservation System project encompasses the creation of a user-friendly online platform that revolutionizes the bus ticketing process. This system will include web and mobile applications for passengers to easily search for routes, select seats, make secure payments, and receive booking confirmations.

1.5 Methodology

Designing a methodology for a bus ticket booking system involves several stages and steps to ensure the development process is systematic and effective. Here's a step-by-step approach to creating a bus ticket booking system:

- **Project Planning and Feasibility Study:** Define the project scope, objectives, and constraints.Conduct a feasibility study to assess the viability of the project in terms of technical, economic, operational, and scheduling aspects.
- **Requirement Gathering:** Identify and document all the requirements for the bus ticket booking system. This should include functional and non-functional requirements. Involve stakeholders, including bus operators, customers, and administrators, to gather comprehensive requirements.
- **System Design:** Create a detailed system design that includes the system's architecture, components, and data structures. Decide on the technology stack, programming languages, and development tools. Plan the database schema, data flow, and user interfaces.
- **Database Design:** Develop a database schema to store information about buses, routes, schedules, ticket availability, user profiles, and transactions. Ensure data security, integrity, and backup mechanisms.
- **Development:** Write code for the various system components, including the front-end (user interface) and back-end (server-side) components. Implement the business logic for booking, payment processing, and ticket generation. Integrate APIs for payment gateways and any external services.
- **User Interface Design:** Create user-friendly interfaces for customers to search for buses, view schedules, and book tickets. Develop an admin interface for managing buses, schedules, and user accounts.
- **Testing:** Conduct unit testing, integration testing, and system testing to identify and rectify any bugs or issues. Implement security testing to protect user data and transactions.

Chapter 2

REQUIREMENT SPECIFICATION

2.1 User characteristics

- **Casual Travelers:** These users book bus tickets occasionally for leisure trips or special occasions.
- **Frequent Commuters:** Daily or regular travelers who rely on buses for their daily commute.
- **Small and Local Operators:** Smaller companies managing local or regional routes.
- **Intercity and Long-Distance Operators:** Companies offering long-distance travel services.
- **System Administrators:** Responsible for maintaining and managing the technical aspects of the reservation system.
- **Support Staff:** Handling customer inquiries, assisting with bookings, and resolving issues.
- **Payment Gateway Providers:** Companies that offer secure payment processing services integrated into the system.
- **Transportation Regulatory Bodies:** Governing bodies responsible for overseeing and regulating bus transportation services, fares, and safety.

2.2 Dependencies

In the context of a Bus Ticket Booking System, dependencies refer to the various factors, components, and entities that rely on or influence the functioning and success of the system.

- **Web Development Technologies:** The website's functionality and design rely on web development technologies such as HTML, CSS, JavaScript, and other programming languages.
- **Database Management Systems:** The website stores and retrieves data from a database management system, such as MySQL or MongoDB.

- **Payment Gateways:** The website relies on payment gateways to securely process online payments made by customers.
- **Third-party APIs:** The website may integrate with third-party APIs such as Google pay , phonepe, credit card, amazon pay, upi for making payments.

Infrastructure, financial, and user dependencies further shape the system's effectiveness, highlighting the intricate network of factors and entities that contribute to its functionality and overall viability. Managing these dependencies is crucial for the system's success and its ability to provide a seamless and user-friendly booking experience for all stakeholders involved..

2.3 Hardware specification

The hardware specifications for an bus ticket reservation system website can vary depending on the scale of the website,the number of users, and the complexity of the website's functionality. However, some common hardware specifications that can be considered when building an online payment booking system website are:

- **Server:** A server is required to host the website, store the website's data.
- **Processor:**Intel Core i3 or equivalent AMD processors are suitable for basic web development tasks involving HTML, CSS, and small-scale JavaScript. Mid-Range Processor:
- **Ram:** A minimum 4GB RAM is recommended.
- **Storage:** It is recommended to use solid-state drives (SSDs) as they are faster and more reliable.
- **Backup and Recovery Systems:** to prevent data loss in case of hardware failures or other emergencies

2.4 Software specification

• Web server software: Apache or Node.js.
• Back-end programming language: PHP,Java Script
• Database management system: MySQL
• Front-end technologies: HTML, CSS, JavaScript, jQuery, Bootstrap.
• Integrated development environment (IDE): Visual Studio Code

Chapter 3

WEBSITE DESIGN

3.1 Sitemap

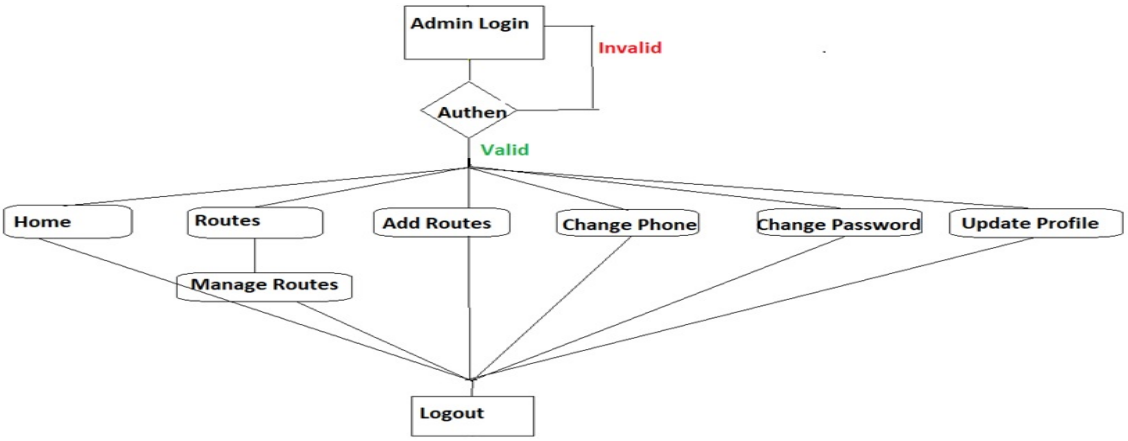


Figure 3.1: **Architecture Diagram**

Figure 3.1 Illustrates the architecture of the overall website. Overall, a sitemap is an essential tool for any website that wants to provide a good user experience and optimize its content for search engines. It helps users understand the organization and structure of a website, navigate to the pages they need, and discover new content that may be of interest to them. At the same time, it helps search engines crawl and index the site more efficiently, ensuring that all the pages are properly ranked and displayed in search results.

3.2 Design Phase

3.2.1 Data Flow Diagram

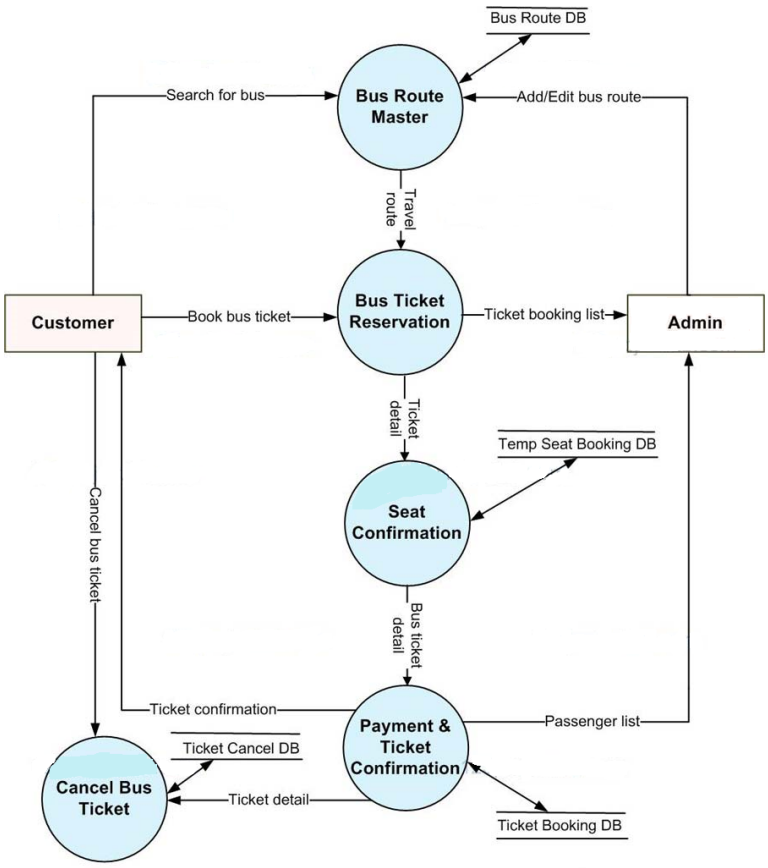


Figure 3.2: **Data Flow Diagram**

- Process 1: Represents the process of user registration, login, and authentication.
- Process 2: Represents the process of searching for bus routes, selecting seats, and making a reservation.
- Data Flow 1: Represents the flow of user registration and login data.
- Data Flow 2: Represents the flow of search criteria (e.g., destination, date, time) from the user to the system.
- Data Flow 3: Represents the flow of available bus routes, schedules, and seat information from the Bus Operator Database to the reservation system.
- Data Flow 4: Represents the flow of reservation details (e.g., passenger information, selected seats) from the user to the system.
- Data Flow 5: Represents the flow of payment information from the user to the system.
- Data Flow 6: Represents the flow of booking confirmation and ticket information from the system to the user.

3.3 Front End and Back End Design

3.3.1 Home Page

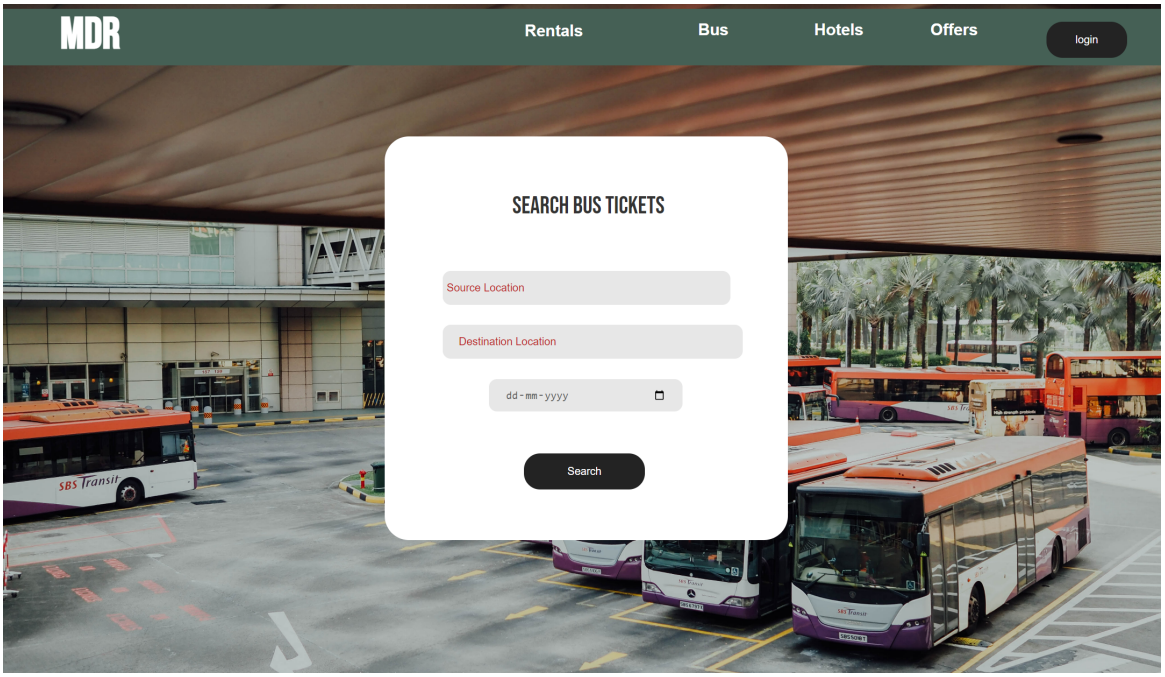


Figure 3.3: **Home Page**

Figure 3.3 Illustrates a Home Page is our complete outlook of the homepage for our website. our website has a source location and destination location at which date and search button,background image it looks like depot and there will be a login button for who are register and offers and hotel,etc..

3.3.2 Signup and Login page

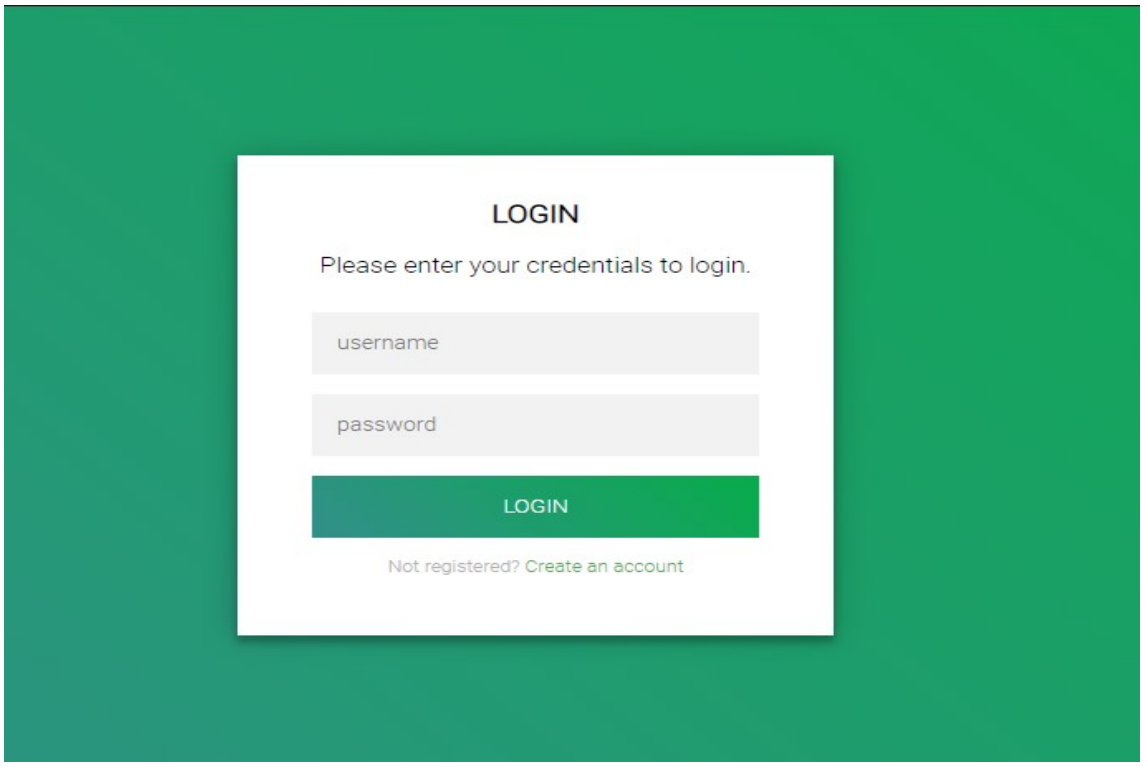


Figure 3.4: **Login Page**

Figure 3.4 The bus ticket booking website's signup page is the gateway for new users to join the community of travelers. It boasts a welcoming and user-friendly design, making it easy for individuals to create their accounts and start their travel journey. The signup form is comprehensive, requesting essential details such as name, email address, contact number, and a secure password. To enhance security and user trust, the page may include a password strength indicator. Once users fill out the required information and click the "Sign Up" button, they receive a verification email to confirm their account. The signup process ensures that travelers have access to personalized booking options, special promotions, and a seamless ticket reservation experience.

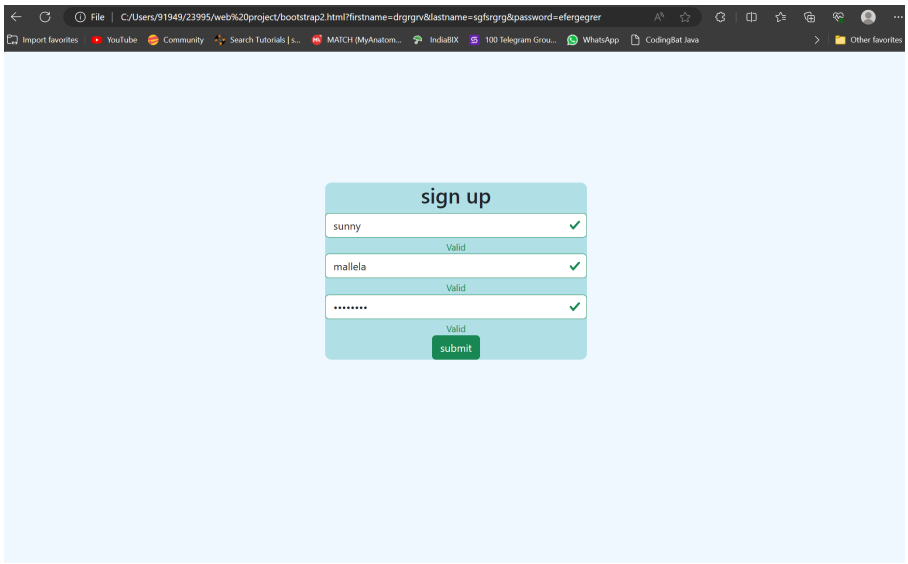


Figure 3.5: **Sign Up Page**

Figure 3.5 The bus ticket booking website's login page is the portal to a world of convenient and hassle-free travel planning. It features a clean and intuitive design, inviting users to access their accounts and embark on their journey. The page prominently displays two input fields: "Username" and "Password," ensuring a secure and straightforward login process. Beneath the login form, there's a "Forgot Password" link to assist users in case they've forgotten their credentials. A "Sign Up" button allows new users to easily navigate to the registration page if they don't yet have an account. The login page exudes a sense of trust and reliability, assuring passengers that their travel plans are in safe hands. Upon successful login, travelers gain access to a wide range of bus routes, schedules, and booking options, turning their travel dreams into reality.

3.3.3 Form Validation

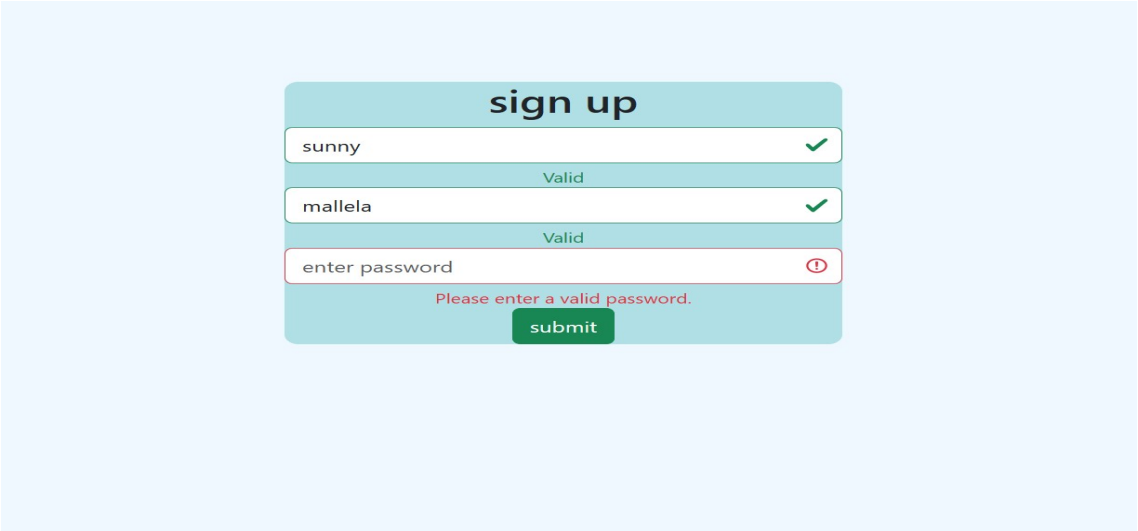


Figure 3.6: **Form Validation**

Figure 3.6: Form validation in a signup process is a crucial step to ensure that user-provided information is accurate and complete before creating an account. It typically involves checking that all required fields are filled out, verifying the format and validity of data such as email addresses, and confirming that passwords meet security criteria, like length and complexity. Effective form validation helps prevent errors, enhance security, and improve the overall user experience by guiding users to provide correct information and ensuring the integrity of data stored in the system.

3.3.4 Parse the webpage using Jquery and DOM

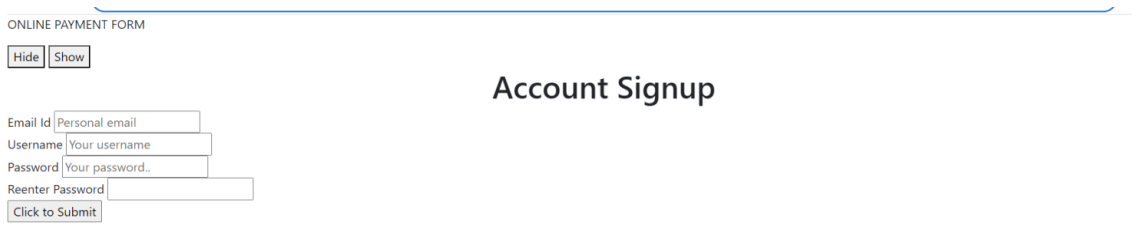


Figure 3.7: **Website with Jquery and DOM**

- Initialization and Setup: Start by including the jQuery library in your HTML document using a script tag. Create a separate JavaScript file for your code.
- Select Elements: Utilize jQuery selectors to pinpoint specific HTML elements on the webpage. These selectors can be element names, classes, IDs, or attribute values
- Access and Manipulate Elements: Once you've selected elements, you can access their properties, attributes, and content using jQuery methods.

3.3.5 Creation of Webserver using Node Js

Figure 3.8: **webserver using Node Js**

Figure 3.8 depicts Creating a web server with Node.js empowers you to build powerful web applications and APIs. The process outlined above provides a foundational structure, but the specifics will depend on your project's requirements and complexity.The figure 3.8 It is a JavaScript runtime that allows you to run JavaScript on the server-side, making it a versatile platform for building web applications.It contains creating a file ,read and write on it .

3.3.6 Design of Three Tier application using Node js and MySQL

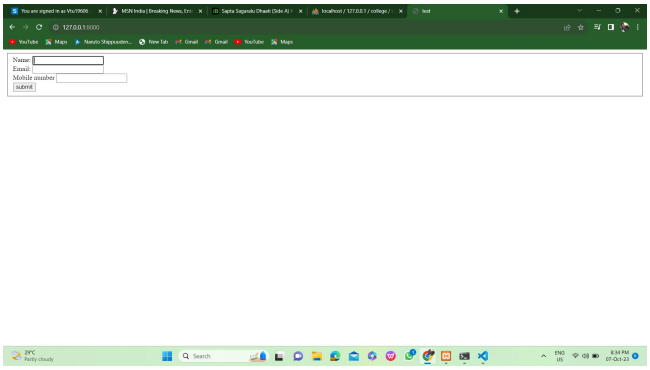


Figure 3.9: **Three Tier Appilication Using Node js and MySQL**

Figure 3.9 Depicts creating a reactive form for user registration using Angular. A Three-Tier Application is a software architecture pattern where an application is divided into three interconnected components or tiers: the Presentation Tier, the Application Logic Tier, and the Data Storage Tier. Each tier has a specific role and responsibility within the application.This form allows users to input their information, by displaying the submitted data in the browser console upon submission, developers can efficiently debug and troubleshoot the registration process.

3.3.7 Design of Reactive form for User Registration using Angular

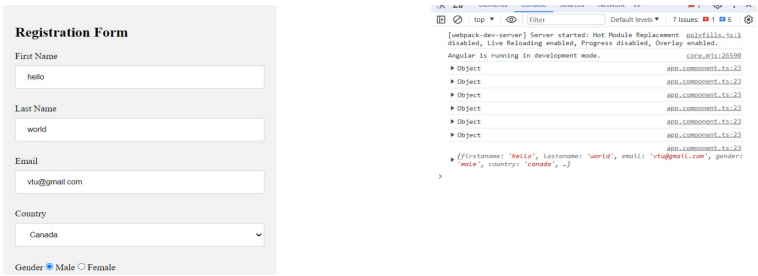


Figure 3.10: **User Registration Using Angular**

Figure 3.10 Designing a reactive form for user registration using Angular involves creating a dynamic and interactive form that allows users to input their information and register for a service or application. Reactive forms in Angular are built using the Reactive Forms module, which provides a more programmatic and flexible way to manage form data and validation. Here's a theoretical overview of how to design a reactive form for user registration in Angular: Set Up Your Angular Project:

- Make sure you have Node.js and Angular CLI installed. Create a new Angular project using the Angular CLI: ng new my-registration-app.
- Import the ReactiveFormsModule: In your Angular application, you need to import the ReactiveFormsModule from @angular/forms in your app.module.ts file to enable the use of reactive forms.

3.3.8 Develop Web Application to Implement Routing and Navigation in Angular

Figure 3.11: **Routing and Navigation in Angular**

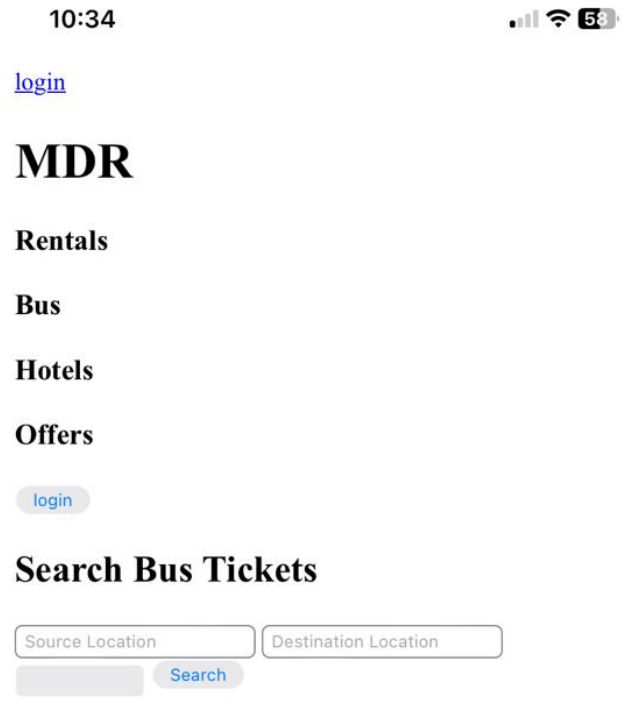
Figure 3.11 Angular's routing system also supports route parameters, enabling dynamic content generation. Additionally, you can implement route guards to protect routes and perform authentication or authorization checks.

3.3.9 Creation of Microservices

3.3.10 Creation of Microservices

Figure 3.11 Depicts development of a micro service to the website bus ticket reservation system through nodejs using sceneca toolkit.

3.3.11 Web Application to Mobile Application



3.3.12 Mobile Application

Figure 3.12 Onverting a web application into a mobile application involves adapting the web-based content and functionality for use on mobile devices, such as smartphones and tablets.

Chapter 4

TESTING

4.1 Testing

Testing is a critical phase in the development of an Bus ticket booking system to ensure its functionality, reliability, and security.

Functional Testing:

•User Registration and Login: Verify that users can create accounts, log in, and recover passwords if needed.

• Search and Booking: Test the system's ability to search for seats, display seats availability, and al- low users to book journey for specific stays.

• Seat Selection: Ensure that customers can select seats based on their preferences when booking ticket accommodations.

• Payment Processing: Test the payment gateway for various payment methods, including credit/debit cards, mobile wallets, and net banking when booking hotel accommodations.

• Booking Confirmation: Verify that users receive booking confirmations with relevant details when they make bus reservations.

•Booking Cancellation and Refunds: Test the cancellation process and refund mechanisms for canceled ticket reservations.

Performance Testing:

• Load Testing: Assess how the system performs under various levels of user traffic to ensure it can handle peak usage without slowdowns or crashes when users are making bus booking reservations.

• Scalability: Verify that the system can scale up to accommodate increasing user loads by adding server resources as needed.

• Response Time: To ensure a seamless user experience, it's crucial to actively monitor and optimize response times for search, booking, and payment processing, employing efficient strategies and tech- nologies to enhance overall system performance.

•Security Testing: Authentication and Authorization: Ensure that user authentication and authorization mechanisms are robust and secure.

• Payment Security: Test the payment gateway for vulnerabilities and compliance with industry standards such as PCI DSS.

4.1.1 Test Result

The above mentioned test cases all are passed successfully when they are tested against the online payment bus booking website created. All the test cases produced positive results. All the pages and modules in the website are functioning properly.

Testcase ID	Test Case Description	Input	Expected Result
TC-01	Valid Login	Username:sunnymallela Password:sunny@414	User is Successfully logged in and redirected to the homepage.
TC-02	Invalid Username	username:244664 password:akshi12	User sees an error message indicating that the username is incorrect.
TC-03	Invalid password	username:leela@07 password:12398	user sees an error message indicating that the password is incorrect.
TC-04	Empty fields	username:(empty) password:(empty)	user sees an error message indicating that the both fields are required.

Figure 4.1: **Test Results for Login Page**

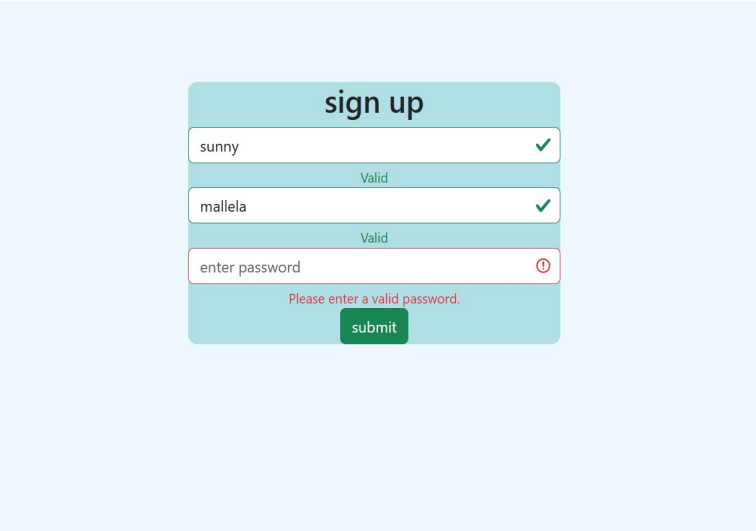


Figure 4.2: **Invalid Details**

4.1.2 Test Bugs

Based on the tests conducted, the potential bugs found are :

- Blank spaces are allowed in the username and password fields. Numbers and special characters are also allowed in the username field.

- The website takes some excessive amount of time to process a request, causing slow performance for the user.

- The website is vulnerable to SQL injection attacks, allowing malicious users to access the databaseand extract sensitive information. These are some of the bugs identified while testing the website. So it is important to test the website completely with all the possible test cases and ensure that it is functioning properly.

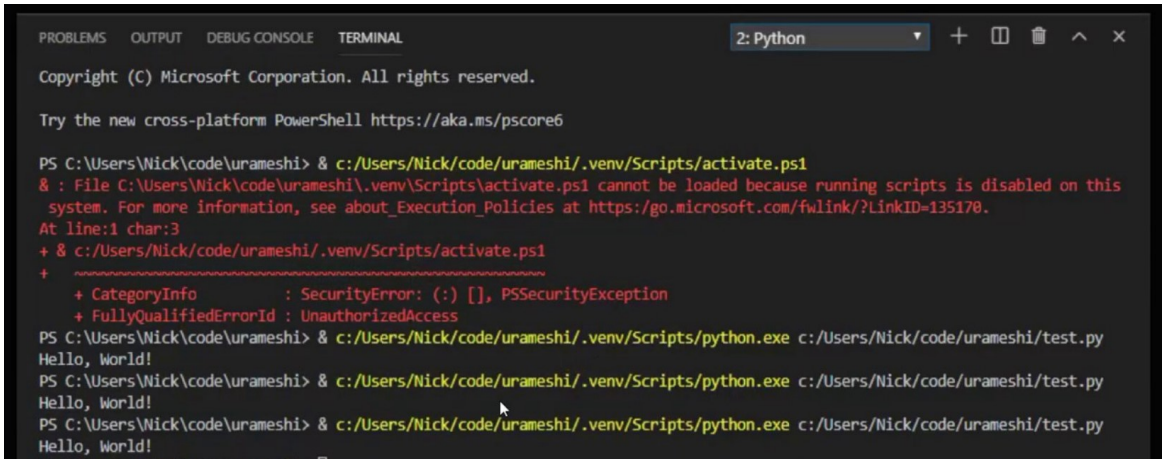


Figure 4.3: Test bugs

Chapter 5

WEBSITE LAUNCH

Figure 5.1: **Test Home Image**

Chapter 6

RESULTS AND DISCUSSIONS

6.1 Website performance

The website loads quickly and smoothly, with all pages and features easily accessible without delays or glitches.Website speed is increased by optimizing images, minimizing HTTP requests, and using content delivery networks (CDNs) to reduce server response times. The website is responsive and mobile-friendly, with a clear and easy-to-use interface that adapts to different screen sizes.Customer experience is enhanced by offering engaging features such as ratings, reviews, and recommendations.Prioritizing performance considerations from the outset of development can significantly enhance the system's effectiveness and user satisfaction.

6.2 Security

The Bus Ticket Reservation System can safeguard sensitive information and maintain the trust of both passengers and bus operators. The identity of website is authenticated the encyrption techniques are used to convert the data into the code to prevent unauthorized access or viewing of the data.Data backup refers to the process of creating copies of important data and storing them in a secure location, to prevent data loss in the event of a security breach or technical issue.Firewalls and Vulnerability scanners for monitoring network traffic and scanning security vulnerabilities.

6.3 Responsiveness and mobile-friendliness

Optimization of website for different screen sizes can be achieved by using a responsive design that adapts to different screen sizes or by creating a separate mobile version of the website.Text size is legible and easily readable on a small screen and a font size that is easy to read without having to zoom in or scroll horizontally.Using compressed images, image sprites, and responsive images to ensure that they load quickly and look great on any device.

Chapter 7

CONCLUSION AND FUTURE ENHANCEMENTS

7.1 Conclusion

The development and operation of a Bus Ticket Reservation System represent a significant step forward in modernizing and streamlining the bus transportation industry.It addresses the diverse needs of passengers, bus operators, and system administrators, offering secure and user-friendly online booking and management functionalities. an online payment website must prioritize performance, security, and mobile-friendliness to provide a seamless and enjoyable experience for users. The website should load quickly, be easy to navigate, and offer secure payment options to ensure that users have a positive experience. Additionally,By focusing on these key areas, an online payment website can build trust with its customers, increase sales, and stay ahead of its competition in the ever-growing digital payment industry.

7.2 Future Enhancements

In the ever-evolving landscape of bus transportation and online booking systems, the potential for future enhancements in a Bus Ticket Reservation System is both exciting and promising. As technology continues to advance, there are numerous avenues for improvement and expansion. From seamless mobile wallet integration and predictive analytics to AI-driven chatbots and multi-language support, the focus remains on enhancing passenger convenience and optimizing bus operator operations. These planned enhancements aim not only to streamline the booking process but also to provide passengers with a more personalized and efficient travel experience

Chapter 8

SOURCE CODE

HTML Code - Home page

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>BUS TICKET RESERVTION SYSTEM</title>
    <link rel="stylesheet" href="index.css">

</head>
<body>
    <header>
        <h1>ticket booking</h1>
        <nav>
            <a href="bootstarp.html">Contact Us</a>
            <a href="my1.html">Login</a>
                <a href="my3.html">sign up</a>

        </nav>
    </header>

    <main>
        <section class="hero">
            <div class="hero-content">
                <h2>welcome to bus reservation</h2>
                <p>select your destination</p>
                <div class="-buttons">
                    <h3>Payment</h3>

                    <div class="booking">
                        <a href="https://www.google.com/url?sa=t&source=web&rct=j&opi=89978449&url=
                            https://www.phonepe.com/&ved=2ahUKEwi94cyeh9yAAxWKZmwGHeVRCRMQFnoECAUQAQ
                            &usg=AOvVaw3f8k5FEApWYRYqzuU62ZOV">phonepe</a>
                        <p>here is phonepe we can help all kinds of payments</p>
      <p> easy payment without any technical issue.</p>
                    </div>

                    <div class="payment">
                        <a href="https://paytm.com/credit-card-bill-payment">creditcard</a>
                        <p>here is credit card payment we can help all payment with this. </p>
```

```
        <p> easy payment without any technical issue.</p>
                    </div>

                    <div class="payment">
                        <a href="https://www.bankbazaar.com/login.html">UPI</a>
                        <p>here is upi payment,it consists of all app.</p>
                    </div>
                    <div class="payment">
                        <a href="https://www.amazon.in/amazonpay/home/">Amazon pay</a>
                        <p>here is Amazon pay payment you can make shopping .</p>
        <p> you'r welcome sir.</p>
                    </div>
                </div>
            </div>
        </section>
    </main>
</body>
</html>
```

CSS Code - Home page

```
    body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
    background-color: blue;
}

header {
    display: flex;
    justify-content: space-between;
    align-items: center;
    background-color: greenyellow;
    color: black;
    padding: 15px;
}

header h1 {
    margin: 0;
}

nav a {
    color: white;
    text-decoration: none;
    padding: 8px 12px;
    margin-left: 10px;
    border-radius: 5px;
    background-color: #333;
}
```

```
29
30 nav a:hover {
31     background-color: #555;
32 }
33
34 main {
35     padding: 20px;
36 }
37
38 section {
39     margin-bottom: 30px;
40 }
41
42 .hero {
43     position: relative;
44     text-align: center;
45 }
46
47
48
49 .hero-content h2 {
50     font-size: 36px;
51     color: #333;
52     margin-bottom: 10px;
53 }
54
55 .hero-content p {
56     color: #777;
57     margin-bottom: 20px;
58 }
59
60 .payment-buttons {
61     display: flex;
62     flex-direction: column;
63     align-items: flex-start;
64 }
65
66 .payment-buttons a {
67     display: inline-block;
68     padding: 10px 20px;
69     margin: 5px 0;
70     border-radius: 5px;
71     background-color: #4CAF50;
72     color: white;
73     text-decoration: none;
74 }
75
76 .payment-buttons a:hover {
77     background-color: #45a049;
78 }
```

```
.payment {
    text-align: left;
    margin-bottom: 30px;
}

.payment p {
    color: #555;
    margin-top: 10px ;
}
```

HTML Code - Login Page

```
    <!DOCTYPE html>
<html lang="en">

<head>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.1/dist/css/bootstrap.min.css" rel="
        stylesheet" integrity="sha384-4bw+/aepP/YC94hEpVNVgiZdgIC5+VKNBQNGCHeKRQN+
        PtmoHDEXuppvnDJzQIu9" crossorigin="anonymous">
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.1/dist/js/bootstrap.bundle.min.js"
        integrity="sha384-HwwvtgBNo3bZJJLYd8oVXjrBZt8cqVSpeBNS5n7C8IVInixGAoxmnlMuBnhbgrkm"
        crossorigin="anonymous"></script>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Login =BUS BOOKING</title>
    <link rel="stylesheet" href="you.css">
</head>


    <header>
        <h1>ONLINE PAYMENT</h1>
        <nav>
            <a href="contact.html">Contact Us</a>
            <a href="login.html">Login</a>
            <a href="signup.html">Signup</a>
        </nav>
    </header>

    <main>
        <h2>Login</h2>
        <form action="login.php" method="post" class="was-validated">
            <label for="username">Username:</label>
            <input type="text" id="username" name="username" required pattern="[A-Za-z]{1,8}"
                placeholder="Please enter the username" title="Username should be an alphabet" class
                ="form-control">
            <div class="invalid-feedback"></div>
            <div class="valid-feedback">valid username</div>

```

```
            </div>
            <br>
            <label for="password">Password:</label>
            <input type="password" id="password" name="password" required placeholder="Please enter
                the password" class="form-control">
            <div class="invalid-feedback"></div>
            <div class="valid-feedback">valid password</div>
            </div>
            </br>
            <input type="submit" value="Login" class="btn btn-primary">
        </form>
    </main>
</body>

</html>
```

HTML Code - Signup Page Parse the webpage using Jquery and DOM

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.1/dist/css/bootstrap.min.css" rel="
        stylesheet" integrity="sha384-4bw+/aepP/YC94hEpVNVgiZdgIC5+VKNBQNGCHeKRQN+
        PtmoHDEXuppvnDJzQIu9" crossorigin="anonymous">
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.1/dist/js/bootstrap.bundle.min.js"
        integrity="sha384-HwwvtgBNo3bZJJLYd8oVXjrBZt8cqVSpeBNS5n7C8IVInixGAoxmnlMuBnhbgrkm"
        crossorigin="anonymous"></script>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.4/jquery.min.js"></script>
    <title>Payment Sign Up</title>

    <script>
       $(document).ready(function(){
  $("button").click(function(){
    $("p").hide("slow", function(){
      alert("The paragraph is now hidden");
    });
  });
});
        function validateForm() {
            const emailInput = document.getElementById('email');
            const usernameInput = document.getElementById('username');
            const passwordInput = document.getElementById('password');
            const reenterPasswordInput = document.getElementById('reenter-password');

            const email = emailInput.value;
            const username = usernameInput.value;
            const password = passwordInput.value;
```

```
            const reenterPassword = reenterPasswordInput.value;

            const emailPattern = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
            const passwordPattern = /^(?=.\d)(?=.[a-z])(?=.*[A-Z]).{8,}$/;

            if (!emailPattern.test(email)) {
                alert('Please enter a valid email address.');
                return false;
            }

            if (username.trim() === '') {
                alert('Please enter a username.');
                return false;
            }

            if (!passwordPattern.test(password)) {
                alert('Enter a valid password.');
                return false;
            }

            if (password !== reenterPassword) {
                alert('Passwords do not match.');
                return false;
            }

            alert('Form submitted successfully!');
            return true;
        }
    </script>
    <style>
        #panel, #flip {
          padding: 5px;
          text-align: center;
          background-color: #e5eecc;
          border: solid 1px #c3c3c3;
        }

        #panel {
          padding: 50px;
          display: none;
        }
        </style>
</head>
<body>
    <button>Hide</button>

    <p>This is a paragraph with little content.</p>

    <form action="login.html" onsubmit="return validateForm();">
        <form action="login.html" onsubmit="return onSubmitForm();">
```

```
            <h1 align="center">Account Signup</h1>

            <label for="email">Email Id</label>
            <input type="email" id="email" name="email" placeholder="Personal email"><br>

            <label for="username">Username</label>
            <input type="text" id="username" name="username" placeholder="Your username"><br>

            <label for="password">Password</label>
            <input type="password" id="password" name="password" placeholder="Your password.."><br>

            <label for="reenter-password">Reenter Password</label>
            <input type="password" id="reenter-password" name="reenter password"><br>

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

    </form>
</body>
</html>
```

Creation of Webserver Using Node Js

```
    var http = require('http');
var fs = require('fs');
http.createServer(function (req, res) {
     fs.readFile('java.html',function(err, data) {
        res.writeHead(200, {'Content-Type': 'text/html'});
        res.write(data);
        return res.end();
     });
}).listen(8080);
console.log('Server running at http://127.0.0.1:8080/');
```

Design of Three tier application using Node js and MySQL

```
<html>
<head>
      <title> test </title>
     </head>
      <body>
            <form action="/" method="POST">
                  <fieldset>
                        <label for="name">Name: </label>
                        <input type="text" id="name" name="name" autofocus />
                        <br/>
                        <label for="email">Email: </label>
                        <input type="email" id="email" name="email"   />
                        <br/>
                        <label for="username">Mobile number </label>
                        <input type="number" id="username" name="mno" />
                        <br/>
                        <input type="submit" value="submit" />
                  </fieldset>
            </form>
      </body>
</html>
var mysql = require('mysql');
var con = mysql.createConnection({
  host  : "localhost",
  user  : "root",
  password  : "",
   database  :"college"
});

module.exports=con;
```

Design of Reactive form for User Registration using Angular

```
    App.component.html
    <div class ="form">
  <h2 id="registration"> Registration Form</h2>
  <form [formGroup]="reactiveForm" (ngSubmit)="onSubmit()">
  <label for="fname"> First Name </label>
  <input type="text" id="fname" placeholder="Enter your first name" formControlName="firstaname">
  <br>

  <br>
  <label for="lname"> Last Name </label>
  <input type="text" id="lname" placeholder="Enter your last name" formControlName="lastaname">
  <br>

  <br>
```

```
<label for="email"> Email </label>
<input type="email" id="email" placeholder="Enter your last email" formControlName="email">
<br>

<br>
<label for="country">Country</label>
<select id="country" formControlName="country">
<option value="australia">Australia</option>
<option value="canada">Canada</option>
<option value="usa" selected>USA</option>
</select>
<br>
<br>
<label for="gender">Gender</label>
<input type="radio" id="male" value="male" formControlName="gender">
<label for="male">Male</label>
<input type="radio" id="female" value="female" formControlName="gender">
<label for="male">Female</label>
<br>
<br>
<label>Hobbies</label>
<div class="form-inline">
<label><input type="checkbox" value="sports" checked formControlName="hobbies">Sports</label>
<label><input type="checkbox" value="movies" formControlName="hobbies" >Movies</label>
<label><input type="checkbox" value="music" formControlName="hobbies" >Music</label>
</div><br>
<input type="submit" value="Submit" id="btn">
</form>
</div>
```

Develop web application to implement routing and navigation in Angular

```
Login.component.html

<html>
<body>
<p>Login</p>
<table>
      <tr>
            <td>
                  <label>Username:<input type="text">
                   </label>
            </td>
    </tr>
<tr>
            <td>
                  <label>Password:<input type="password">
                   </label>
            </td>
    </tr>
<tr>
<td>
<button (click)="updateName()">Login</button>
</td>
</tr>
</table>
</body>
</html>

home.component.html
<p>Give some input to the input box</p>
<p>Name:<input type="text" [(ngModel)]="firstName"</p>
<p>You wrote:<b>{{firstName)}</b></p>

<div ng-bind="firstName"></div>
<button (click)="firstName()">Update Name</button>

app-routing.module.ts
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { LoginComponent } from './login/login.component';
import { HomeComponent } from './home/home.component';

const routes: Routes = [
    {path:'login',component:LoginComponent},          //need to add
    {path:'home',component:HomeComponent}
];

@NgModule({
    imports: [RouterModule.forRoot(routes)],
```

```
    exports: [RouterModule]
})
export class AppRoutingModule { }
export const routingcomponents=[LoginComponent,HomeComponent]

app.module.ts
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';

import { AppRoutingModule,routingcomponents } from './app-routing.module';//include dat alone
import { AppComponent } from './app.component';
import { FormsModule } from '@angular/forms';

import { HomeComponent } from './home/home.component';  //delete those two lines
import { LoginComponent } from './login/login.component';

@NgModule({
    declarations: [
        AppComponent,
        HomeComponent,  //delete those two lines
        LoginComponent
     routingcomponents     //include dat alone

    ],
    imports: [
        BrowserModule,
        AppRoutingModule, FormsModule

    ],
    providers: [],
    bootstrap: [AppComponent]
})
export class AppModule { }

app.component.html
<div>
   <h1>
        Angular Routing Application</h1>
        <nav>
            <a routerLink="/login" routerLinkActive="active">Login</a>
            <a routerLink="/home" routerLinkActive="active">Home</a>
        </nav>
</div>
<router-outlet></router-outlet>

login.component.ts
import { Component,OnInit } from '@angular/core';
import{ Router } from '@angular/router'; //add this line

@Component({
```

```
    selector: 'app-login',
    templateUrl: './login.component.html',
    styleUrls: ['./login.component.css']
})
export class LoginComponent implements OnInit{  //add this line
    constructor(private router:Router){}
        ngOnInit():void{}
        updateName(){
            this.router.navigate(['home']);
        }
    }

  home.component.ts
import { Component,OnInit } from '@angular/core';
import{Router} from '@angular/router';

@Component({
    selector: 'app-home',
    templateUrl: './home.component.html',
    styleUrls: ['./home.component.css']
})
export class HomeComponent implements OnInit{
    firstName: any;
    constructor(private router:Router){}
        ngOnInit(){

        }
        updateName(){
            console.log(this.firstName)
            this.router.navigate(['login']);
        }
    }

  Home.component.html
<p>Give some input to the input box</p>
<p>Name:<input type="text" [(ngModel)]="firstName"></p>
<p>You wrote:<b>{{firstName}}</b></p>

<div ng-bind="firstName"></div>
<button (click)="updateName()">Update Name</button>
```

Creation of Microservices

```
    var seneca = require('seneca')()

seneca.add({role: 'math', cmd: 'sum'}, function (msg, respond) {
    var sum = msg.left + msg.right
    respond(null, {answer: sum})
})
```

```
7
8  seneca.add({role: 'math', cmd: 'product'}, function (msg, respond) {
9      var product = msg.left * msg.right
10     respond(null, { answer: product })
11 })
12
13 seneca.act({role: 'math', cmd: 'sum', left: 1, right: 2}, console.log)
14         .act({role: 'math', cmd: 'product', left: 3, right: 4}, console.log)
```

Chapter 9

SCREENSHOTS

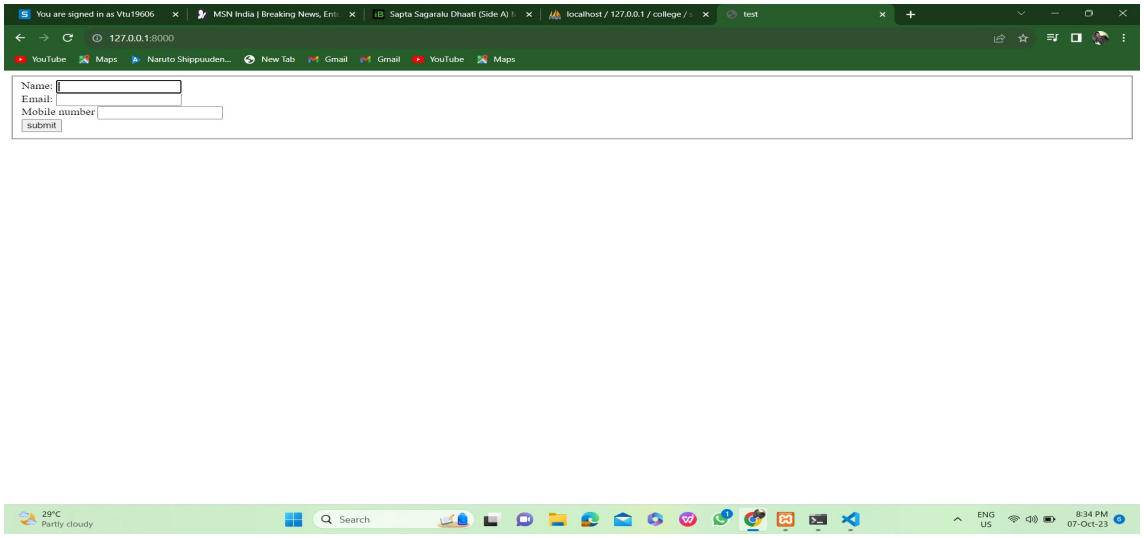


Figure 9.1: **Signup Page**

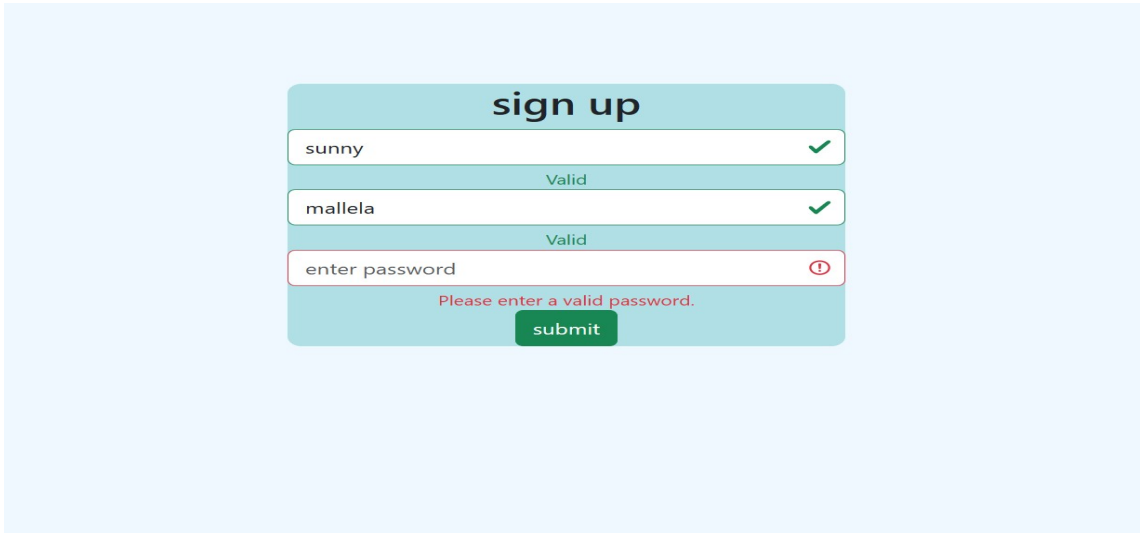


Figure 9.2: **Webpage using Bootstrap**

Figure 9.3: **Creation of webserver**

Chapter 10

REFERENCES