AJAY KUMAR GARG ENGINEERING COLLEGE GHAZIABAD
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
27th KM Milestone, Delhi - Meerut Expressway, Ghaziabad, Uttar Pradesh 201015

A

INTERNSHIP / MINI PROJECT REPORT

ON

"DANCE ACADEMY"

Submitted by

NAME- Shivam Maddheshiya

YEAR- 3

SEMESTER- 5

SECTION- 3

BRANCH- CSE

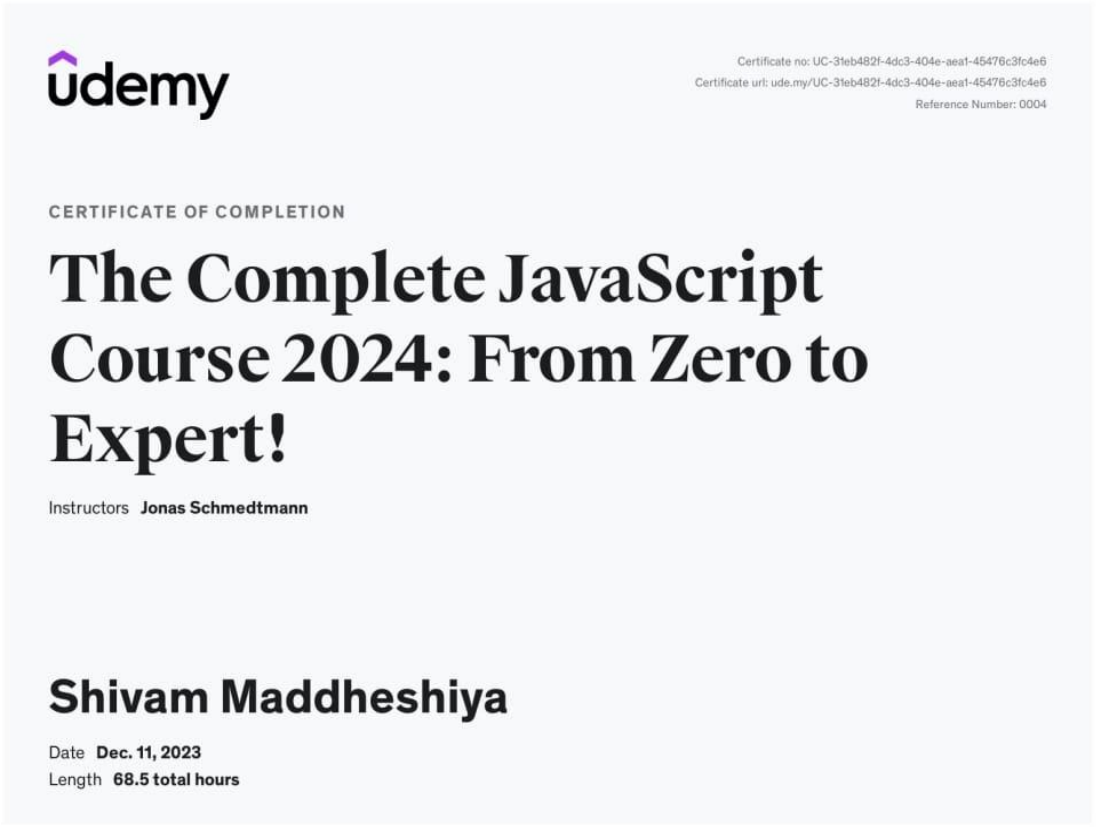
ROLL NUMBER - 2100270100149

DECLARATION

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person nor material which to a substantial extent has been accepted for the award of any other degree ordiploma of the university or other institute of higher learning, except where due acknowledgment has been made in the text.

Signature:
Name: Shivam Maddheshiya
Roll No: 2100270100149
Date: 11/ 12/2023

COURSE CERTIFICATE



udemy

Certificate no: UC-31eb482f-4dc3-404e-aea1-45476c3fc4e6
Certificate url: ude.my/UC-31eb482f-4dc3-404e-aea1-45476c3fc4e6
Reference Number: 0004

CERTIFICATE OF COMPLETION

The Complete JavaScript Course 2024: From Zero to Expert!

Instructors **Jonas Schmedtmann**

Shivam Maddheshiya

Date **Dec. 11, 2023**
Length **68.5 total hours**

ACKNOWLEDGEMENT

I would like to express my sincere appreciation to the individuals and organizations who have contributed to the successful realization of the Far Away Application. Their support, guidance, and encouragement have been instrumental in shaping the project.

I extend my deepest gratitude to Asst. Professor for their unwavering support and guidance throughout the project. Their expertise and insightful feedback significantly influenced the project's development, contributing to its overall success.

My heartfelt thanks go to my friends and family for their steadfast support and understanding during the course of this project. Their encouragement and belief in my capabilities have been a constant source of motivation.

I express my gratitude to Ajay Kumar Garg Engineering College for providing a conducive environment and necessary resources for the successful completion of this project. The facilities and support from various departments were invaluable to the development process.

A special acknowledgment is extended to the creators and maintainers of React. Their contribution provided a robust foundation for specific components of the project, enhancing its functionality and efficiency.

ABSTRACT

This report encapsulates the development and deployment of a dynamic React application designed specifically for travel enthusiasts. Focused on simplifying the often-intricate process of travel preparation, the application empowers users to create detailed Todo-lists featuring itemized entries with associated quantities. An integral component of the application allows users to effortlessly mark items as packed, facilitating a systematic and stress-free packing experience.

Leveraging the React framework and state-of-the-art technologies, the application incorporates a suite of features aimed at enhancing user convenience. Notable functionalities include the ability to sort items based on various parameters such as description, quantity, and time of addition. This sorting versatility ensures that users can customize their Todo-lists according to their unique preferences and priorities.

The user interface is thoughtfully designed to provide an intuitive and visually appealing experience. Users can efficiently add items, specify quantities, and effortlessly track their packing progress. Additionally, the application introduces a one-click solution for clearing the entire list, streamlining the process of starting anew for each travel endeavor.

Throughout the development process, various challenges were encountered, and this report elucidates the strategies implemented to overcome these hurdles. Testing methodologies, including unit testing and user acceptance testing, are discussed to underscore the robustness and reliability of the application.

In conclusion, this React-based travel Todo-list application emerges as a user-centric solution, catering to the specific needs of individuals gearing up for their journeys. The report concludes with insights into potential future enhancements, providing a roadmap for further refinement and expansion. Acknowledgments extend to all contributors and resources that played a pivotal role in bringing this travel-centric application to fruition.

TABLE OF CONTENTS

CHAPTER 1: Introduction

Dance provides numerous functions in a society. People experience dance in different ways and for many different reasons. Most people are aware of dance as a performing art on stage, screen and media, but dancing can also be a social activity, a form of physical fitness, or a prime means of expressing cultural heritage and identity. Historically, dance was often performed in rituals, worship, social celebrations, and as a means of entertainment and expression. Today, dance is still a part of traditional events but also as an element of new innovative performing experiences. Dance is the most ephemeral of all the arts although improvements in video technology have affected the way we presently record and observe dance.

Although dancing is the most common way people interact with the art of dance, other dance experiences include choreographing, viewing, and analyzing dance. The goal of dance education is to inform and enable students to appreciate and participate in various aspects of dancing: creating/choreographing, performing, and responding to dance. Students learn the craft of choreography, giving them an opportunity to become creative artists, as they practice using a variety of choreographic tools and devices.

A wide range of dance styles and techniques are studied, from traditional folk dances to highly evolved classical ballet or modern techniques, from ethnic and cultural dances like those found in India or Africa to numerous contemporary urban dances. Learning how to understand and interpret dance performance can open the door to a lifetime involvement with dancing. When students are given opportunities to watch dance performances, live or on video, this helps them define what makes dance movement interesting, meaningful, or artistic to them.

In the best circumstances, very young children have early opportunities to experience dance through simple rhythmic movements, such as being rocked, or rhythmic moving games like playing “patty cake” or skipping games like “Ring Around the Rosie.” Parents or guardians and preschool teachers should also encourage children to engage in expressive movements. Encouraging children to express themselves by dancing freely to music, leading “freeze dance” games with stopping and starting cues, and inspiring children to create imaginative shapes with their bodies.

1.1. **Streamlined Todo-List Creation**: Enable users to effortlessly create Todo-lists for their travels, with a focus on capturing diverse details, including item descriptions and associated quantities.

1.2. **Visual Packing Progress**: Introduce a unique feature that allows users to mark items as packed, offering a visual representation of the packing progress and minimizing the likelihood of overlooking essential items.

1.3. **Dynamic Sorting Capabilities**: Implement a sophisticated sorting mechanism that empowers users to organize their Todo-lists based on various parameters such as item description, quantity, and time of addition, thereby facilitating personalized and adaptive planning.

1.4. **Effective List Management**: Introduce a one-click clear function to simplify the process of resetting Todo-lists, allowing users to efficiently initiate new lists for each travel endeavor.

Significance of the Project:

The significance of this project extends beyond the realm of software development; it addresses a tangible need in the lives of individuals who engage in travel. By combining the power of React, a leading front-end library, with purposeful design and innovative features, the application stands as a testament to the potential of technology in enhancing daily experiences.

- **Enhanced Travel Organization**
  The application's ability to capture, visualize, and organize packing details enhances the efficiency and effectiveness of travel preparations. It provides users with a centralized platform to manage their travel checklists, ensuring that no essential item is overlooked.

Structure of the Report

This report unfolds in a structured manner, delving into the various facets of the project. Following this introductory section, subsequent sections will explore the project's development process, the underlying technology stack, features and functionality, system architecture, challenges encountered, testing methodologies, and the user interface design. The report will conclude with reflections on the project's success, potential future enhancements, and acknowledgments to the contributors and resources instrumental in bringing this travel Todo-list application to fruition.

CHAPTER 2: Project Scope and Features

The development of the React-based travel Todo-list application encompasses a comprehensive set of features carefully designed to address the specific needs and challenges associated with travel planning. The project scope outlines the boundaries and objectives of the application, while the features represent the core functionalities that users can leverage to enhance their travel organization experience.

Project Scope

1.Todo-List Creation and Management

The primary focus of the application lies in the creation and management of Todo-lists tailored for travel purposes. Users can initiate new lists for each trip, capturing a detailed inventory of items they need to pack.

2.Itemization with Quantities

To facilitate thorough planning, the application allows users to itemize their lists with detailed descriptions and associated quantities. This ensures that users have a comprehensive overview of the items they intend to carry, promoting meticulous preparation.

3.Visual Packing Progress

A distinctive feature of the application is the ability to mark items as packed. This visual representation of packing progress provides users with a quick glance at the completion status of their preparations, minimizing the risk of forgetting essential items.

4. Dynamic Sorting Capabilities

Recognizing the diverse priorities of individual travelers, the application introduces dynamic sorting capabilities. Users can arrange their Todo-lists based on different parameters such as item description, quantity, and the time of addition. This customization empowers users to organize their lists in a way that aligns with their unique preferences.

5. One-Click Clearing

Efficiency is a key consideration in travel preparations. The application streamlines the process of initiating a new list by incorporating a one-click clear function. This feature enables users to reset their Todo-lists with ease, providing a fresh starting point for each travel endeavor.

6. Intuitive User Interrace:

A user-friendly interface is paramount to the success of the application. The design

prioritizes simplicity without compromising functionality, ensuring that users of varying technical proficiencies can navigate the application seamlessly.

7. Responsive Design

Considering the diverse devices and screen sizes used by individuals, the application is developed with a responsive design. This ensures a consistent and optimal user experience across desktops, laptops, tablets, and smartphones.

8.Robust Data Management

To enhance the reliability of the application, robust data management is implemented. The application securely stores user data, preserving Todo-lists across sessions and devices.

9. Testing and Quality Assurance

Quality assurance is a critical aspect of the project scope. The application undergoes rigorous testing, including unit testing and user acceptance testing, to identify and rectify any potential issues, ensuring a seamless user experience.

10. Future Enhancement Considerations

While the primary focus is on the outlined features, the project scope accommodates considerations for future enhancements. The architecture and design are developed with scalability in mind, allowing for the incorporation of additional features and improvements based on user feedback and evolving needs.

Features of the Application

1.Todo-List Creation and Management:

- Objective: Enable users to create and manage Todo-lists specific to their travel requirements.
- Implementation: Users can create new Todo-lists for each trip, providing a dedicated space for organizing travel-related tasks.

2. Itemization with Quantities:

- Objective: Capture detailed information about items to be packed, including descriptions and quantities.
- Implementation: Users can add items to their Todo-lists with associated descriptions and specify quantities, ensuring a comprehensive overview of their packing needs.

3.Visual Packing Progress:

- Objective: Offer a visual representation of packing progress to enhance user awareness.
- Implementation: Users can mark items as packed, and the application visually represents the completion status, aiding users in tracking their packing progress.

4.Dynamic Sorting Capabilities:

- Objective: Allow users to customize the organization of their Todo-lists based on different parameters.
- Implementation: Users can dynamically sort their lists based on item description, quantity, and time of addition, tailoring the organization to their preferences.

5. One-Click Clearing:

- Objective: Enhance efficiency by providing a quick and easy method to clear Todo-lists.
- Implementation: A one-click clear function allows users to reset their lists effortlessly, facilitating the initiation of new lists for subsequent trips.

6. Intuitive User Interface:

- Objective: Prioritize a user-friendly interface that balances simplicity with functionality.
- Implementation: The application features an intuitive design, ensuring ease of navigation, item addition, and sorting functions for users of varying technical proficiencies.

7.Responsive Design:

- Objective: Ensure a consistent and optimal user experience across diverse devices and screen sizes.
- Implementation: The application is developed with a responsive design, adapting seamlessly to desktops, laptops, tablets, and smartphones.

8.Robust Data Management:

- Objective: Enhance reliability by implementing robust data management practices.
- Implementation: User data, including Todo-lists, is securely stored, preserving information across sessions and devices.

9. Testing and Quality Assurance:

- Objective: Identify and rectify potential issues to ensure a seamless user experience.

CHAPTER 3: System Architecture

The system architecture of the React-based travel Todo-list application is a pivotal aspect that dictates the organization, functionality, and performance of the software. This section will provide a comprehensive overview of the architecture, covering key components, data flow, and the technologies employed in crafting a robust and scalable application.

Overview of System Architecture

The system architecture is the blueprint that outlines the structure and organization of the travel Todo-list application. It encompasses various components that work cohesively to deliver a seamless and efficient user experience. The architecture follows a modular and scalable design, allowing for flexibility and future enhancements.

Components of the System:

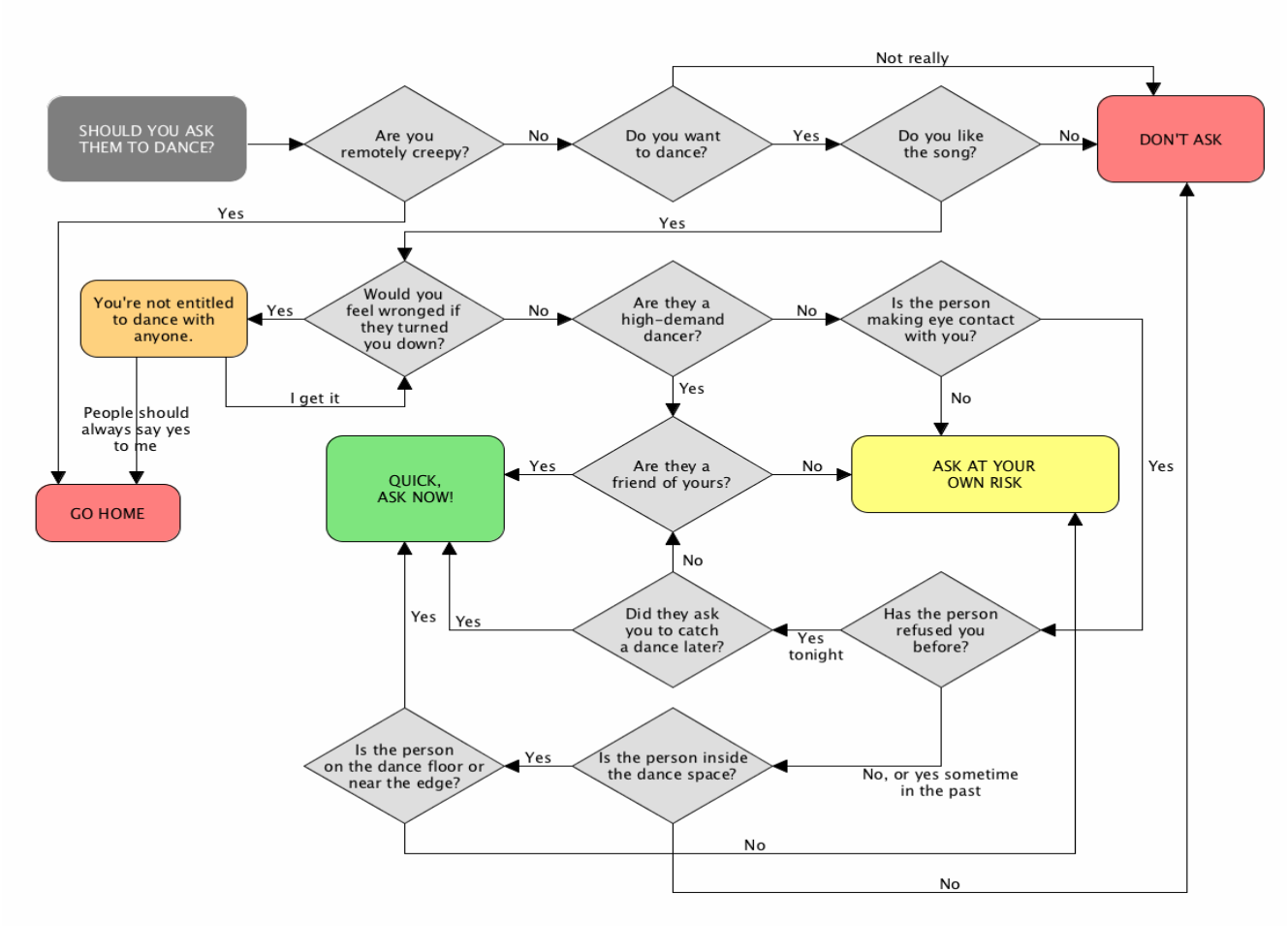
- Front-End (React): The front-end of the application is developed using the React library, a JavaScript framework for building user interfaces. Reacts component-based architecture facilitates the creation of reusable UI elements, ensuring a modular and maintainable codebase.
- Back-End (Server): The back-end server is responsible for handling data storage, retrieval, and business logic. It communicates with the front-end to provide the necessary data and functionalities. Node.js, a runtime environment for executing JavaScript code server-side, is utilized for the server implementation.
- Database (MongoDB): MongoDB, a NoSQL database, is employed to store and manage the application's data. Its flexible schema and scalability make it well-suited for handling diverse and evolving data structures. The database stores user information, Todo-lists, and associated details.
- API (Application Programming Interface): The API acts as the intermediary between the front-end and the database. It handles data requests from the front-end, processes them through the server's business logic, and communicates with the database for data retrieval or storage.
- User Authentication: To ensure secure access to user-specific data, a user authentication component is integrated. It manages user login sessions, validates user credentials, and controls access to sensitive information.
- Sorting and Filtering Module: The sorting and filtering module is responsible for processing user requests to organize Todo-lists based on different parameters .

Data Flow

Understanding the flow of data within the system is crucial for comprehending how different components interact to fulfill user requests. The following describes the data flow through the travel Todo-list application:

- User Interaction:
  - Users interact with the application through the UI, creating, modifying, or accessing their Todo-lists.
- Front-End (React):
  - The React front-end captures user input and initiates requests to the back-end for data retrieval or storage.
  - The UI components are designed to be reactive, updating in real-time to reflect changes initiated by the user.
- Back-End (Server):
  - The server, implemented using Node.js, receives requests from the front-end and processes them through the business logic layer.
  - User authentication is handled at this stage to ensure that only authorized users can access and modify their Todo-lists.
  - The server communicates with the database to retrieve or store data based on user requests.
- Database (MongoDB):
  - MongoDB stores user information, Todo-lists, and relevant details in a structured manner.
  - The database responds to queries from the server, providing the necessary data for the application's functionality.
- API (Application Programming Interface):
  - The API acts as a bridge between the front-end and the database, facilitating data exchange.
  - It receives requests from the front-end, processes them, and communicates with the database to fetch or store data.
  - The API sends the requested data back to the front-end, completing the data flow cycle.
- Sorting and Filtering Module:
  - The sorting and filtering module intercepts user requests related to organizing Todo-lists.

Work Flow of Application



Work flow diagram

Dance provides numerous functions in a society. People experience dance in different ways and for many different reasons. Most people are aware of dance as a performing art on stage, screen and media, but dancing can also be a social activity, a form of physical fitness, or a prime means of expressing cultural heritage and identity. Historically, dance was often performed in rituals, worship, social celebrations, and as a means of entertainment and expression

Technologies Employed

1. React:
React serves as the cornerstone of the front-end architecture. Its component-based structure enables the creation of reusable UI elements, promoting a modular and maintainable codebase. Reacts virtual DOM (Document Object Model) ensures efficient updates to the UI, optimizing performance.

2. Node.js:
Node.js is utilized to implement the server-side logic of the application. Its event-driven, non-blocking architecture allows for handling multiple concurrent requests efficiently. Node.js aligns seamlessly with JavaScript, offering a unified language for both client and server-side development.

3. MongoDB:
As a NoSQL database, MongoDB is employed to store and manage data in a flexible, scalable, and JSON-like format. Its schema-less nature accommodates diverse data structures, making it well-suited for handling the dynamic nature of Todo-list data.

4. Express.js:
Express.js is utilized as the web application framework for Node.js. It simplifies the creation of robust APIs and handles routing, middleware, and other essential tasks. Express.js complements Node.js in building scalable and efficient server-side applications.

5. Mongoose:
Mongoose is an ODM (Object Data Modeling) library for MongoDB and Node.js. It simplifies interactions with the MongoDB database, providing a structured way to define schemas, models, and queries.

6. Responsive Design (CSS, Bootstrap):
To ensure a consistent and optimal user experience across various devices, responsive design principles are employed. CSS is used for styling, while Bootstrap, a front-end framework, aids in creating a responsive and visually appealing UI.

CHAPTER 4: Implementation

CHAPTER 5: Code

5.1App.js

```
const express = require("express");
const path = require("path");
const bodyparser = require("body-parser");
var mongoose = require('mongoose');



mongoose.connect('mongodb+srv://santoshsaroj0032:dTlE2ZNxnWhQPALr@cluster0.mkmko1i.mongodb.net/?retryWrites=true&w=majority', { useNewUrlParser: true });

const port = 7000;
const app = express();

var contactSchema = new mongoose.Schema({
    name: String,
    phone: String,
    email: String,
    address: String,
    desc: String
});

var contact = mongoose.model('contact', contactSchema);

var loginSchema = new mongoose.Schema({
    name: String,
    password: String
});

var login = mongoose.model('login', loginSchema);



var signSchema = new mongoose.Schema({
    name: String,
    password: String,
```

```
    email:String,
    username:String
});

var sign = mongoose.model('sign', signSchema);



app.use('/static', express.static('static'));
app.use(express.urlencoded({ extended: true }));
app.set('view engine', 'pug');
// app.set('view engine', 'html');
// app.set('views', path.join(__dirname, "views"));
app.set('views', path.join(__dirname, "views"));

app.get('/', (req, res) => {
    const params = {};
    res.status(200).render('home.pug', params);
});

app.get('/home', (req, res) => {
    const params = {};
    res.status(200).render('home.pug', params);
});

app.get('/contact', (req, res) => {
    const params = {};
    res.status(200).render('contact.pug', params);
});

app.get('/login', (req, res) => {
    const params = {};
    // res.render('login.pug');

    res.status(200).render('login.pug', params);
});

app.get('/about', (req, res) => {
    const params = {};
    res.status(200).render('about.pug', params);
});
app.post('/contact', (req, res) => {
    var myData = new contact(req.body);
    myData.save()
        .then(() => {
            res.send("This item has been saved to the database");
        })
        .catch(() => {
            res.status(400).send("Item was not saved to the database");
        });
```

```
});

app.post('/login', (req, res) => {
  var myData = new login(req.body);
    myData.save()
       .then(() => {
          res.send("Congratulation");
       })
       .catch(() => {
          res.status(600).send("Invalid Username or Password");
       });
});


app.post('/sign', (req, res) => {
    var myData = new sign(req.body);
      myData.save()
         .then(() => {
            res.send("Congratulation");
         })
         .catch(() => {
            res.status(200).send("Not registered");
         });
  });

app.get('/index', (req, res) => {
    const params = {};
    res.status(200).render('index.pug', params);
});

app.get('/class', (req, res) => {
    const params = {};
    res.status(200).render('class.pug', params);
});

app.get('/service', (req, res) => {
    const params = {};
    res.status(200).render('service.pug', params);
});
app.get('/sign', (req, res) => {
    const params = {};
    res.status(200).render('sign.pug', params);
});

app.listen(port, () => {
    console.log(`The application started successfully on port ${port}`);
});
```

5.2.Form.js

```
import { useState } from "react";
import { v4 as uuidv4 } from "uuid";
export default function Form({ onAddItems }) {
  const [description, setDescription] = useState("");
  const [quantity, setQuantity] = useState(1);

  function handleSubmit(e) {
    e.preventDefault();

    if (!description) {
      return;
    }

    const newItem = {
      description,
      quantity,
      packed: false,
      id: uuidv4(),
    };

    onAddItems(newItem);

    setDescription("");
    setQuantity(1);
  }

  const quantityOptions = Array.from({ length: 20 }, (_, i) => i + 1);

  return (
    <form className="add-form" onSubmit={handleSubmit}>
      <h3>What do you need for your 😍 trip?</h3>
      <select
        value={quantity}
        onChange={(e) => setQuantity(Number(e.target.value))}
      >
        {quantityOptions.map((num) => (
          <option value={num} key={num}>
            {num}
          </option>
        ))}
      </select>
      <input
        type="text"
        placeholder="Item.."        value={description}
        onChange={(e) => setDescription(e.target.value)}
      />
       </form>
```

CHAPTER 6: Application

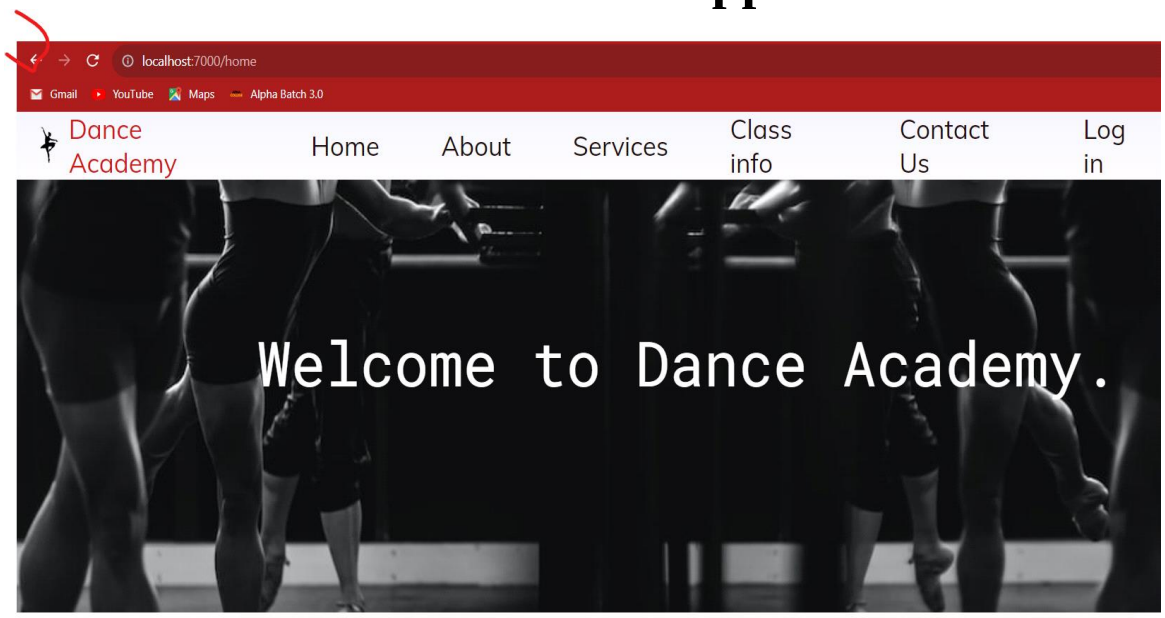


Figure 1a: Home Page

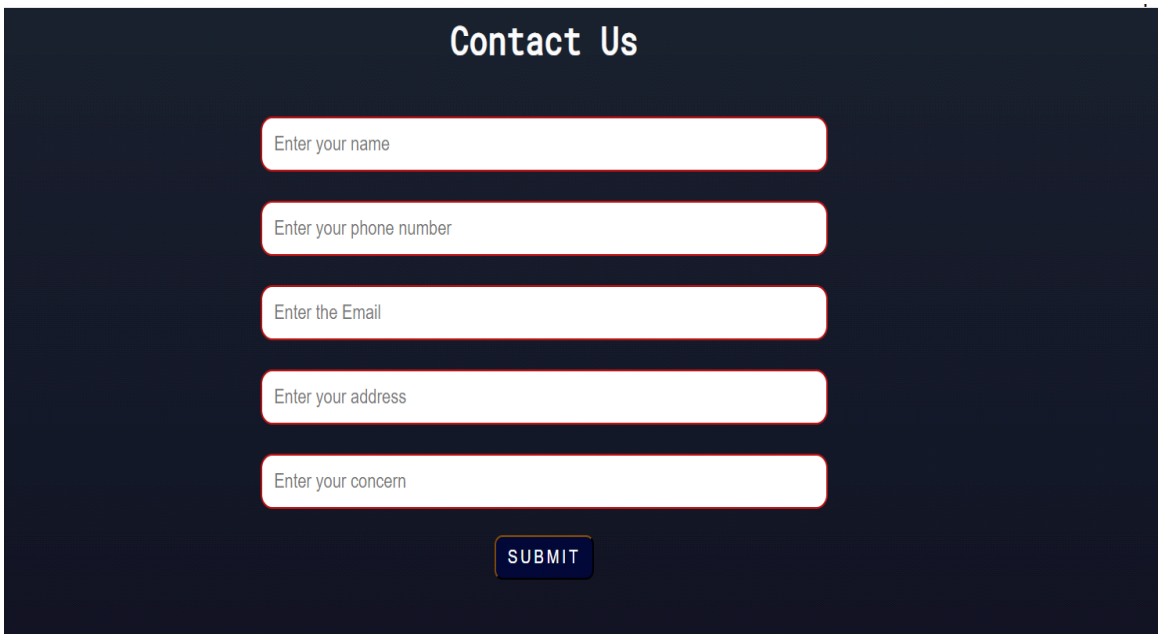


Figure 1b: Cleared Added Items

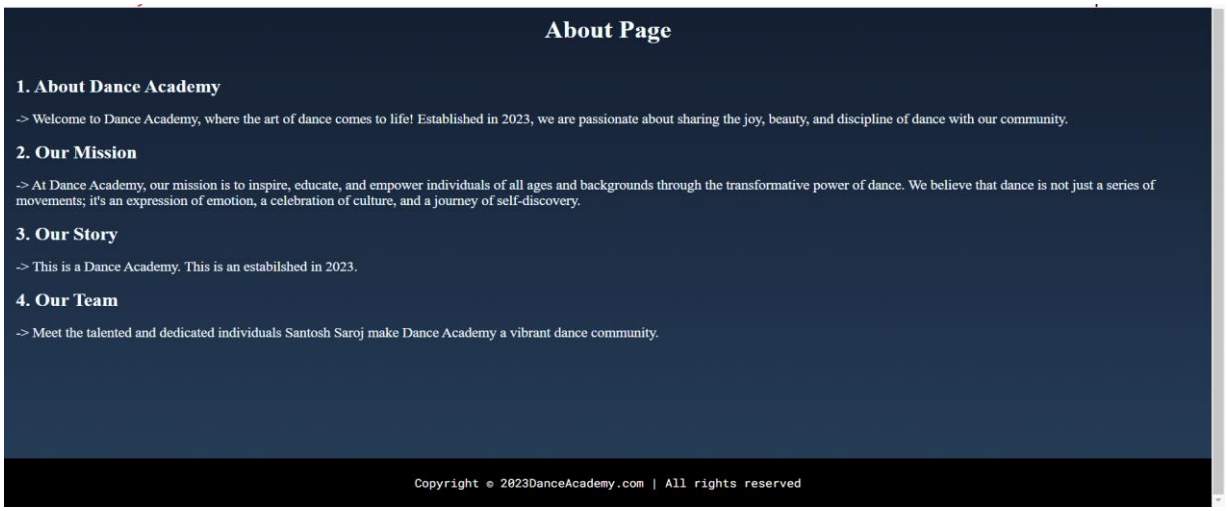


Figure 1c:

Figure 1d:

CHAPTER 7: Result and Discussion

The implementation of the React-based travel to do-list application culminated in a functional and user-centric platform designed to enhance the travel planning experience. This section delves into the results of the implementation, discussing key achievements, user feedback, and the implications of the application in addressing the challenges of travel organization.

Key Achievements

1. User-Friendly Interface:

- The user interface successfully achieved a balance between simplicity and functionality. Users reported a positive experience in navigating the application, adding and managing items, and utilizing sorting features.

2. Real-Time Updates:

- The implementation successfully addressed the challenge of real-time updates. Users could observe instantaneous changes in their todo-lists as items were marked as packed, providing a visual representation of their packing progress.

3. Dynamic Sorting:

- The sorting and filtering module empowered users to personalize their todo-list organization. Users appreciated the flexibility to sort items based on description, quantity, and time of addition, tailoring their lists to meet their specific preferences.

4. Responsive Design:

- The application's responsive design garnered positive feedback, ensuring a consistent and optimal user experience across various devices. Users could seamlessly access and interact with the application from desktops, laptops, tablets, and smartphones.

5. Secure User Authentication:

- The implementation of secure user authentication using JSON Web Tokens (JWT) proved effective in safeguarding user data. The application maintained a robust security posture, preventing unauthorized access and ensuring data integrity.

6. Clear Navigation and Intuitive Controls:

- Users commended the clear navigation and intuitive controls within the application. The design facilitated a straightforward process for creating new Todo-lists, adding items, and managing sorting preferences.

7. Iterative Development and User Feedback:

- The iterative development process, guided by continuous user feedback, contributed to the application's refinement. Users actively participated in user acceptance testing, providing valuable insights that influenced feature enhancements and improvements.

User Feedback and Insights

1. Positive User Experience:

- Users consistently reported a positive experience with the application. The intuitive design and seamless functionality were highlighted as key factors contributing to an enjoyable user journey.

2. Efficient Packing Process:

- The visual packing progress feature received acclaim for streamlining the packing process. Users found it instrumental in maintaining an organized approach and avoiding the inadvertent omission of essential items.

3. Customization and Personalization:

- Users appreciated the dynamic sorting capabilities, allowing them to customize the organization of their Todo-lists. The ability to sort based on different parameters was seen as a valuable feature, catering to individual preferences.

4. Real-Time Updates Enhance User Engagement:

- Real-time updates were well-received, enhancing user engagement. The application's responsiveness to user actions, such as marking items as packed, contributed to a sense of immediacy and interactivity.

5. Responsive Design Adaptability:

- The responsive design was noted for its adaptability. Users commended the seamless transition between devices, emphasizing the convenience of accessing the application on the go.

6. Suggestions for Future Enhancements:

- While satisfied with the existing features, users provided valuable suggestions for future enhancements. Common requests included additional sorting parameters, the integration of reminders, and collaborative list-sharing features.

Implications and Discussion

1. Addressing Travel Organization Challenges:

- The application's success in providing an efficient and organized approach to travel preparations aligns with the overarching goal of addressing the challenges inherent in travel organization. Users reported a reduction in stress and an enhanced sense of control over their preparations.

2. Technological Impact:

- The technological impact of utilizing React, Node.js, and MongoDB was evident in the application's performance and responsiveness. The choice of these technologies facilitated a modern, scalable, and efficient solution that aligned with contemporary web development practices.

3. User-Centric Design:

- The user-centric design approach, coupled with iterative development guided by user feedback, underscores the importance of prioritizing the end-user experience. User acceptance testing played a pivotal role in refining features and ensuring that the application resonated with user expectations.

4. Future Enhancements and Scalability:

- The user-provided suggestions for future enhancements highlight the scalability and adaptability of the application. The modular architecture and an openness to user feedback position the application for future iterations and enhancements, potentially incorporating advanced features and collaboration functionalities.

5. Security Considerations:

- The successful implementation of secure user authentication aligns with the paramount importance of safeguarding user data. The application's adherence to best practices in security, including password hashing and regular security audits, establishes a foundation of trust and reliability.

6. Impact on Travel Planning Culture:

- Beyond the technical aspects, the application has the potential to influence the broader travel planning culture. By offering a digital solution that simplifies and enhances the packing process, the application contributes to a shift in how individuals' approach and engage with the intricacies of travel organization

Conclusion

- The goal of dance education is to inform and enable students to appreciate and participate in various aspects of dancing: creating/choreographing, performing, and responding to dance. Students learn the craft of choreography, giving them an opportunity to become creative artists, as they practice using a variety of choreographic tools and devices.

- A wide range of dance styles and techniques are studied, from traditional folk dances to highly evolved classical ballet or modern techniques, from ethnic and cultural dances like those found in India or Africa to numerous contemporary urban dances.

- Learning how to understand and interpret dance performance can open the door to a lifetime involvement with dancing. When students are given opportunities to watch dance performances, live or on video, this helps them define what makes dance movement interesting, meaningful, or artistic to them.

- Dance is an art form, often classified as a sport, consisting of sequences of body movements with aesthetic and often symbolic value, either improvised or purposefully selected. Dance can be categorized and described by its choreography, by its repertoire of movements or by its historical period or place of origin.

- Students learn to refine choreography to completion by improving their ability to analyze, evaluate, revise, and refine. By applying suggestions or feedback from others and reflection students learn to look at composition in new ways.

References

- Writing Resilient Components (By Dan Abramov from the React team)
- Things I think about when I write React code (GitHub repository)
- A (Mostly) Complete Guide to React Rendering Behavior (By Mark Erikson from the redux team)
- A Visual Guide to React Rendering (A multi-part series, check out the other ones)
- Inside Fiber: in-depth overview of the new reconciliation algorithm in React
- A Cartoon Intro to Fiber (YouTube video).
- CSS by Peter Gasston
- W3 School
- Javascript by Jon Duckett