

# **MULTI-FACILITY FUNCTION COORDINATOR**

## **USING REACT JS**

Submitted in partial fulfillment of the  
requirements for the award of  
Bachelor of Technology in Information Technology

By

**Surya S ( Reg.No - 39120123)**



**DEPARTMENT OF INFORMATION TECHNOLOGY**  
**SCHOOL OF COMPUTING**

# **SATHYABAMA**

**INSTITUTE OF SCIENCE AND TECHNOLOGY**  
**(DEEMED TO BE UNIVERSITY)**

Accredited with Grade "A" by NAAC | 12B Status by UGC | Approved by AICTE  
**JEPPIAAR NAGAR, RAJIV GANDHISALAI,**  
**CHENNAI - 600119**

**MAY - 2023**



# **SATHYABAMA**

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with —All grade by NAAC

Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai – 600 119

[www.sathyabama.ac.in](http://www.sathyabama.ac.in)

## **DEPARTMENT OF INFORMATION TECHNOLOGY**

### **BONAFIDE CERTIFICATE**

This is to certify that this Project Report is the bonafide work of **Surya S (39120123)** who carried out the Project Phase-2 entitled "**MULTI-FACILITY FUNCTION COORDINATOR USING REACT JS**" under my supervision from January 2023 to May 2023.

Internal Guide

**Dr. S. GOWRI., M.E., Ph.D.**

Head of the Department

**Dr. R. SUBHASHINI, M.E., Ph.D.**

Submitted for Viva voce Examination held on 11.05.2023

Internal Examiner

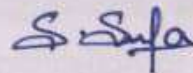
External Examiner

## DECLARATION

I Surya S(Reg. No- 39120123) here by declare that the Project Phase-2 Report entitled "MULTI-FACITLITY FUNCTION COORDINATOR USING REACT JS" done by me under the guidance of Dr. S.Gowri, M.E., Ph.D. is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in **Information Technology**.

DATE : 11.05.23

PLACE : Chennai



SIGNATURE OF THE CANDIDATE

## ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **the Board of Management of SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T.Sasikala M.E., Ph.D, Dean**, School of Computing, **Dr. R.Subhashini M.E., Ph.D**, Head of the Department of Information Technology for providing me with the necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr.S.Gowri M.E.,Ph.D**, for her valuable guidance, suggestions and constant encouragement that paved the way for the successful completion of my phase-2 project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Information Technology** who were helpful in many ways for the completion of the project.

## **ABSTRACT**

React and Bootstrap are two popular tools used for front-end web development. React is a JavaScript library that allows developers to build user interfaces based on UI components. It efficiently updates and renders the necessary components when data changes, making it a popular choice for building dynamic and responsive web applications. Bootstrap, on the other hand, is a CSS framework that provides pre-designed HTML, CSS, and JavaScript templates for various interface components, such as typography, forms, buttons, and navigation. This application helps users to book function hall, party hall, events, catering services, decoration services, online hotel accommodation and beauticians. In now a days, If we want to book a function hall, catering service, hotel accommodation, invitations etc..., for that we want go each and every office to book that hall or anything. For that process It took so much time and wandering for us. Our application rectifies this all problems. Now, users don't have to stand in a queue for hours and wait for their turn. After Booking the application admin panel members contact the user confirm the order. After confirming the event payment updated by the admin , payment gateway integrated in this application. User can also able to edit their own invitation cards. Thereby, Our application becomes a complete different types of event booking and management in single platform and reduces users effort and time. In this busy and hectic schedule, time management plays a vital role.

# TABLE OF CONTENTS

| Chapter<br>No. | TITLE                                                                                           | Page<br>No. |
|----------------|-------------------------------------------------------------------------------------------------|-------------|
|                | <b>ABSTRACT</b>                                                                                 | V           |
|                | <b>LIST OF FIGURES</b>                                                                          | Vii         |
| 1              | <b>INTRODUCTION</b>                                                                             | 8           |
| 2              | <b>LITERATURE SURVEY</b>                                                                        | 11          |
|                | 2.1 Inferences from Literature Survey                                                           | 11          |
|                | 2.2 Open problems in Existing System                                                            | 14          |
| 3              | <b>REQUIREMENTS ANALYSIS</b>                                                                    | 15          |
|                | 3.1 Software Requirements Specification Document                                                | 15          |
| 4              | <b>DESCRIPTION OF PROPOSED SYSTEM</b>                                                           | 18          |
|                | 4.1 Selected Methodology or process model                                                       | 20          |
|                | 4.2 Architecture / Overall Design of Proposed System                                            | 21          |
|                | 4.3 Description of Software for Implementation and<br>Testing plan of the Proposed Model/System | 22          |
| 5              | <b>IMPLEMENTATION</b>                                                                           | 23          |
|                | 5.1 Deployment Setup                                                                            | 23          |
|                | 5.2 Algorithm used                                                                              | 24          |
|                | 5.3 Testing                                                                                     | 25          |
| 6              | <b>RESULTS AND DISCUSSION</b>                                                                   | 27          |
| 7              | <b>CONCLUSION</b>                                                                               | 31          |
|                | 7.1 Conclusion                                                                                  | 31          |
|                | 7.2 Future work                                                                                 | 31          |
|                | <b>REFERENCES</b>                                                                               | 33          |
|                | <b>APPENDIX</b>                                                                                 | 35          |
|                | <b>A. SOURCE CODE</b>                                                                           | 35          |
|                | <b>B. SCREENSHOTS</b>                                                                           | 49          |
|                | <b>C. RESEARCH PAPER</b>                                                                        | 54          |

## LIST OF FIGURES

| <b>FIGURE NO.</b> | <b>FIGURE NAME</b>               | <b>PAGE NO.</b> |
|-------------------|----------------------------------|-----------------|
| 4.1               | System Architecture              | 21              |
| 4.2               | Flow Diagram                     | 21              |
| 6.1               | Event Page form                  | 27              |
| 6.2               | Profile Page                     | 27              |
| 6.3               | Engagement event<br>booking form | 28              |
| 6.4               | Payment Page                     | 28              |
| 6.5               | Admin Dashboard                  | 29              |
| 6.6               | Admin Payment update<br>page     | 29              |

# CHAPTER 1

## INTRODUCTION

The Internet and technologies were very different from what they are now just ten years ago. Technology was still "the future" and "an increasing market." At the same time, the Internet is something you can access from your computer and is genuinely understood by a small number of geeks, making today more present than ever. Today, however, we constantly carry "the Web" in our pockets thanks to mobile technology. Users no longer expect a website page to load slowly; they now demand it instantly. They will probably switch to another website if the page doesn't load within a couple of seconds. Performance of a company website is viewed as a barometer of the calibre of the company as a whole. Users are no longer need to wait hours in a line before getting their turn. People found it simple to view all the online verticals at once thanks to the event management system's integration of technology improvements. Since single page applications (SPAs) can use repeating layouts and function as "content on demand" apps, they are more processing-efficient than standard MPA sites, can be built for less money, and require less time from developers. The REST API allows you to connect applications to each other like pieces in a puzzle. The REST APIs are based on HTTP and imitate web communication styles, making them very advantageous to use in the MERN stack. With the REST APIs, developers have the possibility, for example, of developing new front-ends and developing new applications that can be linked to the same system very easily. Node.js is very fast simply because it is an asynchronous language. Its performance is impressive compared to the average of other languages. A quick back-end means that users of your application will have access to their data much faster, and that is very advantageous indeed. This is especially the case when considering that today, the average acceptable time for a user to wait for a web page to load is less than 1 second. Added to that, this increased speed will decrease the costs of operating the server. The MERN stack uses the famous React.js, which is fantastic for front end development. We can't help but mention a few big names that use React.js for its unrivaled performance: Facebook, Dropbox, Airbnb, Altassian, Tesla, etc. React.js is based on the single-page application (SPA) which involves having a web application accessed from a single web page. This avoids loading a new page with each action, and makes for a

greatly streamlined user experience. Facebook uses the single page application, which can be clearly felt in the comfort and effortlessness of using the social network.

- The speed of design and development of websites and web applications,
- Reducing server costs,
- The performance of greatly optimized web applications and software,
- The ease of transposing a web application to a mobile application or software, thanks in particular to React Native,
- The luxury of designing a website using a single HTML document,
- The development of a computer application using a single language, JavaScript

A vital benefit of the MERN stack is the server-side framework Express JS, which runs on a Node JS server. Express JS is a speedy, unbiased, and light Node JS web framework. It has modules that enable matching an incoming URL with a server function that can deal with HTTP requests and responses. To connect to Express JS functions that enhance your application, you can utilize XML HTTP requests or React JS front-end POSTs. The functions use Node.js drivers for MongoDB to read and update data in your MongoDB database. One of its benefits is the MERN stack's ability to store data for content, user profiles, uploads, events, and comments. The ideal situation would be for a database to work seamlessly with Node, Express, and React. React JS generates the JSON files on the front end, which are then sent to the Express JS server for processing and instant archival in MongoDB for recovery. You can get a lot out of the MERN stack by creating a web application that depends on the system's functional requirements. If you want a technology stack that prevents common coding errors by its very design, then the MEAN stack is a better choice. As Angular uses Typescript, so prevents common coding errors at the coding stage itself. However, React lags behind here. MongoDB, which is the base of the MERN stack, is designed to store JSON data natively. Everything in it, including CLI and query language, is built using JSON and JS. The NoSQL database management system works well with NodeJS and thus, allows manipulating, representing, and storing JSON data at every tier of the application. It comes in a variant called MongoDB Atlas that further eases database management by offering an auto-scaling MongoDB cluster on any cloud provider and with just a few clicks. Express is a server-side framework that wraps HTTP requests and responses and makes mapping URLs to server-side functions easy. This perfectly complements the

ReactJS framework, a front-end JS framework for developing interactive UIs in HTML while communicating with the server. As the two technologies work with JSON, data flows seamlessly, making it possible to develop fast and debug easily. To make sense of the entire system, you need to understand only one language, i.e., JavaScript and the JSON document structure. The expectations from today's applications have become increasingly more complex, from being available 24/7 around the world to complex scenarios with an efficient mix of rich user experiences built on the client using JavaScript and communicating efficiently through seamless web APIs.

## **CHAPTER 2**

### **LITERATURE SURVEY**

#### **2.1 INFERENCES FROM LITREATURE SURVEY**

Literature survey is a crucial step in the software development process. It involves researching and gathering information about the requirements, needs, and challenges related to developing a software tool. Before starting the development process, it is important to determine the time factor, resource requirement, manpower, economy, and company strength to ensure that the project is feasible. Once these factors are considered, the next step is to determine the operating system and programming language that would be used for developing the tool. The programmers need external support from senior programmers, books, and websites to develop the tool efficiently. Literature survey also helps in identifying the software specifications required for the system. This includes determining the operating system and necessary software needed to proceed with the development process. Overall, literature survey plays a crucial role in ensuring that the software development process is smooth and efficient. It helps in identifying the requirements, challenges, and best practices for developing the software tool.

##### **2.1.1 Enhancing MongoDB with Purpose-Based Access Control :**

MongoDB is a popular NoSQL database that is widely used in modern web applications. Enhancing MongoDB with purpose-based access control can provide a more fine-grained access control mechanism that can be tailored to the specific needs of the application. The use of purpose-based access control in MongoDB can help prevent data breaches and unauthorized access to sensitive data. It can also provide more flexibility in managing data access, allowing administrators to set policies that restrict access to data based on the purpose for which it is being accessed. Implementing purpose-based access control in MongoDB requires careful planning and design. It involves identifying the different types of data that need to be protected and determining the access policies that should be put in place for each type of data. Overall, enhancing MongoDB with purpose-based access control can provide a more robust and flexible access control mechanism that can help protect sensitive data in modern web applications.

### 2.1.2 Webapp Service for Booking Handyman Using Mongodb, Express JS, React JS, Node JS

A webapp service for booking handyman can provide a platform for customers to easily book handyman services online. MongoDB, Express JS, React JS, Node JS are all popular technologies used for building modern web applications. MongoDB is a NoSQL database that can provide flexibility in storing and retrieving data. Express JS and Node JS are used for building the backend server of the web application, while React JS is used for building the frontend user interface. The use of these technologies can provide a scalable and flexible platform for booking handyman services online. The database can store customer and handyman information, service requests, and other relevant data. The backend server can handle user authentication, service requests, and communication between the frontend and backend. The frontend user interface can provide a user-friendly way for customers to browse handyman services, select the desired service, and book the service. Implementing a webapp service for booking handyman using MongoDB, Express JS, React JS, Node JS requires careful planning and design. It involves identifying the requirements of the web application, designing the database schema, implementing the backend server and frontend user interface, and testing and deploying the application. Overall, a webapp service for booking handyman using MongoDB, Express JS, React JS, Node JS can provide a modern and convenient way for customers to book handyman services online. The use of these technologies can provide a scalable and flexible platform that can be tailored to the specific needs of the business.

### 2.1.3 A Frictionless and Secure User Authentication

A frictionless and secure user authentication is essential for ensuring that only authorized users can access sensitive data or perform specific actions. It involves designing an authentication process that is easy and convenient for the user while preventing unauthorized access through the use of strong password policies, multi-factor authentication, and encryption. Implementing a frictionless and secure user authentication requires careful planning and design, and businesses must identify the specific requirements of the application, select appropriate authentication methods, and test and monitor the system to ensure its security and usability.

#### 2.1.4 An Intelligent Book Sorting Management System

An Intelligent Book Sorting Management System is a computerized system designed to automatically sort books based on various parameters, such as author, genre, or publication date. By using technologies such as machine learning, computer vision, and barcode scanning, the system can quickly and accurately sort large numbers of books. The system can also provide real-time information on the location and status of each book, allowing for efficient management of a library or bookstore. Implementing an Intelligent Book Sorting Management System requires careful planning and design, and businesses must consider factors such as the size of the book collection, the available technology, and the specific needs of the users. Overall, an Intelligent Book Sorting Management System can improve the efficiency and accuracy of book sorting and management, providing a more convenient and user-friendly experience for both librarians and customers.

#### 2.1.5 Proposal of expert system for hotel booking system :

The proposal of an expert system for a hotel booking system is an approach to providing an efficient and user-friendly hotel booking experience. An expert system is a computer program that uses artificial intelligence to simulate the decision-making ability of a human expert. By analyzing data such as customer preferences, room availability, and pricing, the expert system can provide personalized recommendations and suggest optimal room selections for individual customers. The system can also handle booking cancellations and rescheduling, as well as handle payment processing. Implementing an expert system for a hotel booking system requires careful planning and design, and businesses must consider factors such as the size of the hotel, the available technology, and the specific needs of the customers. Overall, an expert system for a hotel booking system can improve the efficiency and accuracy of hotel bookings, providing a more personalized and user-friendly experience for customers.

### 2.2 OPEN PROBLEMS IN EXISTING SYSTEM

1. Limited accessibility: Traditional booking systems usually require customers to physically visit or call the booking office during business hours, which limits accessibility and convenience for customers.
2. Manual processing: Manual processing of bookings can lead to errors and inconsistencies in the system, resulting in incorrect bookings and unhappy customers.

3. Limited payment options: Traditional booking systems may only accept payment in person or over the phone, limiting payment options for customers and potentially losing business.
4. Time-consuming: The traditional booking process may be time-consuming, involving paperwork and manual entry of data, which can lead to delays and frustration for both customers and booking staff.
5. Poor customer experience: With limited accessibility, manual processing, limited payment options, and time-consuming procedures, the old booking system can result in a poor customer experience, leading to customer dissatisfaction and loss of business.

## **CHAPTER 3**

### **REQUIREMENT ANALYSIS**

#### **3.1 SOFTWARE REQUIREMENTS SPECIFICATION DOCUMENT**

Windows 10 or above NPM or Yarn

Workspace – Visual Studio Code

Frontend - Bootstrap, CSS, HTML, Font Awesome, JavaScript.

Backend - Node JS, MongoDB, Express

##### **3.1.1 Windows 10 or above**

Windows 10 is a Microsoft operating system for personal computers, tablets, embedded devices and internet of things devices. Microsoft released Windows 10 in July 2015 as a follow-up to Windows 8. The company has said it will update Windows10 in perpetuity rather than release a new, full-fledged operating system as a successor.

##### **3.1.2 Node JS**

As an asynchronous event-driven JavaScript runtime, Node.js is designed to build scalable network applications. In the following "hello world" example, many connections can be handled concurrently. Upon each connection, the callback is fired, but if there is no work to be done, Node.js will sleep. This is in contrast to today's more common concurrency model, in which OS threads are employed. Thread-based networking is relatively inefficient and very difficult to use. Furthermore, users of Node.js are free from worries of dead-locking the process, since there are no locks. Almost no function in Node.js directly performs I/O, so the process never blocks except when the I/O is performed using synchronous methods of Node.js standard library. Because nothing blocks, scalable systems are very reasonable to developed in Node.js. If some of this language is unfamiliar, there is a full article on Blocking vs. Non- Blocking. Node.js is similar in design to, and influenced by, systems like Ruby's Event Machine and Python's Twisted. Node.js takes the event model a bit further. It presents an event loop as a runtime construct instead of as a library. In other systems, there is always a blocking call to start the event-loop. Typically, behavior is defined through call backs at the beginning of a script, and at the end a server is started through a blocking call like Event Machine::run() In 18 Node.js, there is no such start-the-event-loop call. Node.js simply enters the event loop after executing the input script. Node.js exits the event loop when there are no

more call backs to perform. This behaviour is like browser JavaScript — the event loop is hidden from the user. HTTP is a first-class citizen in Node.js, designed with streaming and low latency in mind. This makes Node.js well suited for the foundation of a web library or framework. Node.js being designed without threads doesn't mean you can't take advantage of multiple cores in your environment. Child processes can be spawned by using our `child_process.fork()` API, and are designed to be easy to communicate with. Built upon that same interface is the cluster module, which allows you to share sockets between processes to enable load balancing over your cores.

### 3.1.3 NPM

NPM is short for node package manager, an online directory that contains the various already registered open-source packages. NPM modules consume the various functions as a third-party package when installed into an app using the NPM command `npm install` .

### 3.1.4 Database (mongodb)

MongoDB is a source-available cross-platform document-oriented database program. Classified as a NoSQL database program, MongoDB uses JSON-like documents with optional schemas. MongoDB is developed by MongoDB Inc. and licensed under the Server Side Public License (SSPL) which is deemed non-free by several distributions.

### 3.1.5 Express JS

Express.js, or simply Express, is a back end web application framework for building RESTful APIs with Node.js, released as free and open-source software under the MIT License. It is designed for building web applications and APIs. It has been called the de facto standard server framework for Node.js.

### 3.1.6 Components for the application

- [1] Admin Login component
- [2] User Login component
- [3] Terms and Conditions component
- [4] Event registration component
- [5] Payment component
- [6] Profile page component
- [7] Service Gallery component
- [8] Forgot password component
- [9] To-do component.

### 3.1.7 Backend

1. The backend is the server-side of the website. It stores and arranges data, and also makes sure everything on the client-side of the website works fine.
2. It is the part of the website that you cannot see and interact with.
3. It is the portion of software that does not come in direct contact with the users.  
Backend Frameworks - right software backend frameworks are the foundation that makes applications/ websites faster.
4. They are considered an essential aspect of web development projects since they provide much-needed interactivity for web applications.
5. The back end of a website comprises things like servers, databases, operating systems, APIs, and more.
6. All of which come together to ensure that the user is served with the correct information as quickly as possible.

## CHAPTER 4

### DESCRIPTION OF PROPOSED SYSTEM

The project on Multi-facility function coordinator in the place where clients come to book their function program in one place and pay for that. So, there is need to calculate how many services are booked and generate the bill for the client.

if there is a requirement to create a new system to alleviate the constraints of the current system. it is important to consider the requirements of both the clients and the vendors. The new system should be designed to meet the needs of both parties while addressing the bottlenecks that are currently present in the existing system.

It has following qualities :

- Lower processing costs.
- Fewer errors.
- Automated posting.
- Flexibility.
- Ability to meet user requirement Reduction in manpower.
- Improved reporting.
- Faster response time.
- Improves resources use.

A proposed system should have the same qualities as the existing system, while also aiming to improve processing speed and ensure long-term use without errors or maintenance. The main objective of the project should be to provide a convenient solution for billing patterns and create an easy-to-use environment for users and customers.

*Automation* – In simpler terms, the application replaces the manual system with an automated one, which improves its efficiency and speed. As a result, the system is able to process tasks much quicker and with a shorter response time.

*Accuracy* – To put it differently, the application offers users a prompt and precise response, providing them with accurate information about bill calculations, customer details, and other relevant system information whenever needed.

*User Friendly* – In simpler terms, the Super Store Sales Management System comes with a user-friendly interface that enables users to work on it easily. The application combines accuracy with an attractive interface, thereby enhancing the interaction of the present manual system. This makes the system more intuitive, faster, and user-

friendly.

*Maintain Cost* – The application is a standalone system that does not require any maintenance cost, resulting in reduced costs.

*HttpOnly Cookie* is a tag that is added to a browser cookie that forbids data access by client side scripts. It offers a fence that stops anyone other than the server from accessing the specialised cookie. The possibility of client-side scripts accessing a protected cookie is reduced by using the HttpOnly tag when creating cookies, making these cookies more safe. Say a browser notices a cookie with the HttpOnly flag in it. The browser will return an empty string if client-side code tries to read the cookie. This lessens the likelihood that malicious code—typically cross-site scripting (XSS)—will convey the information to an attacker's website.

*JSON Web Tokens (JWT)* is an open standard that facilitates the secure transfer of data between parties as a JSON object. It is a condensed and self-contained method that can be pronounced as "jot". Although all JWTs are tokens, not all tokens are JWTs since JWT is a specific standard. A JWT can be delivered through a URL, a POST parameter, or an HTTP header, and is quickly transferred due to its small size. By providing all the necessary data about an entity, a JWT eliminates the need for multiple database queries. Moreover, unlike other tokens, a JWT does not require the token recipient to verify it by calling a server.

*More condensed* - To put it differently, a JWT is smaller in size compared to a SAML token when encoded because JSON has a more concise syntax than XML. This makes JWT a better choice for passing data in HTML and HTTP environments.

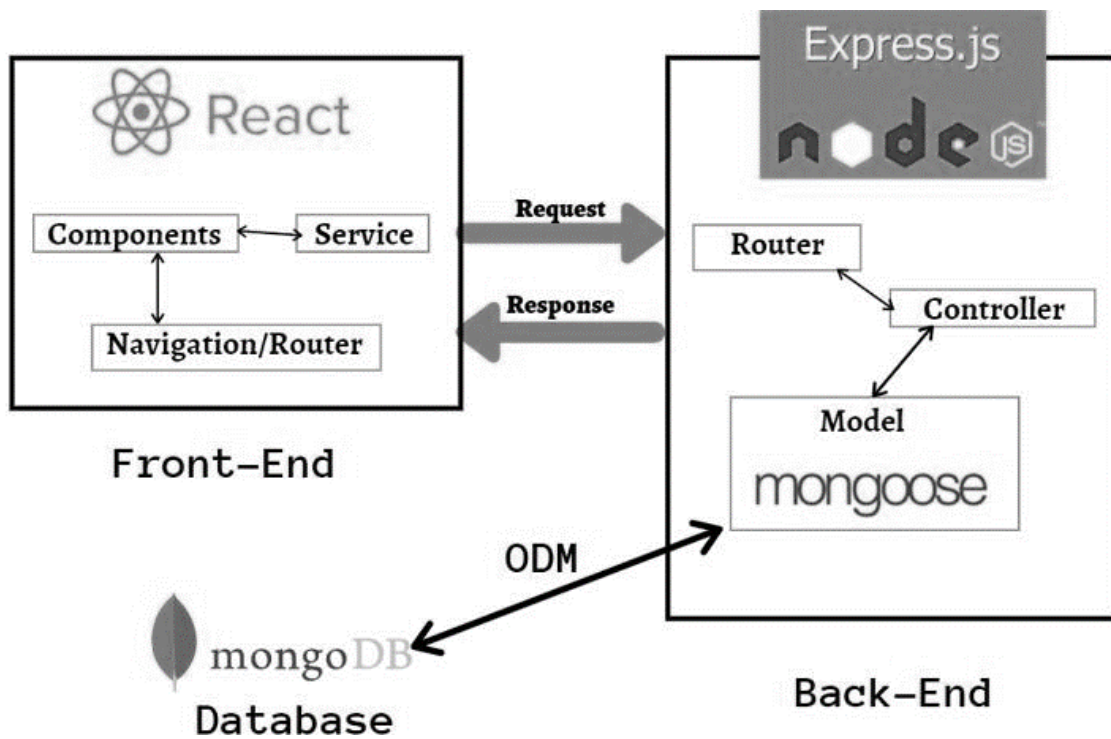
*More Secure* : JWT signing algorithms are used to ensure the integrity and authenticity of the data contained within a JWT. The two main types of JWT signing algorithms are symmetric and asymmetric.

Symmetric algorithms, such as HMAC, use a shared secret key to both sign and verify the JWT. The same secret key is used by both the issuer and the verifier to ensure that the JWT has not been tampered with during transmission. The HMAC algorithm is fast, efficient, and easy to implement, making it a popular choice for many applications.

*Bcrypt* is hashing hash passwords with the aid of the Bcrypt library. It makes use of a Blowfish-based password-hashing algorithm. It can be utilised in cypher suites and encryption products since the Blowfish cypher, a symmetric block cypher, has the finest encryption rate in the business

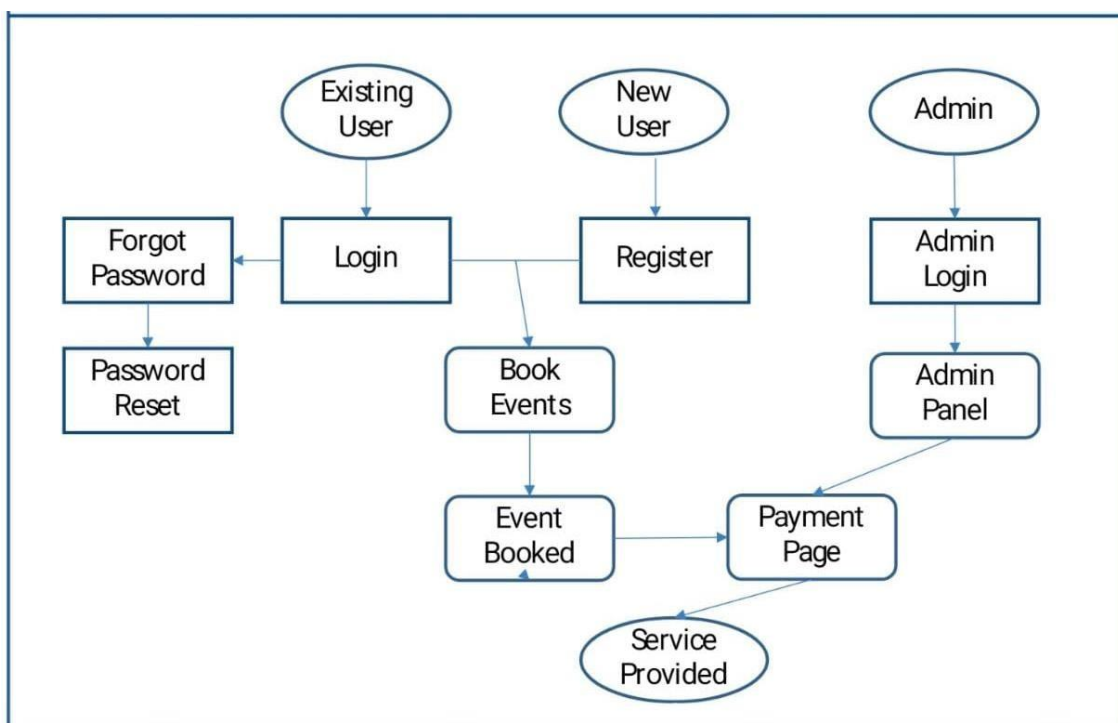
#### 4.1 SELECTED METHODOLOGY OR PROCESS MODEL

- The first step is to create the components that are to be used in web page once after installing node JS, mongoDB and npm or yarn.
- “npx create-react-app app-name” is the command for creating a react app from any command line.
- The installed project is opened by typing “npm start” in the command prompt
- opened from the application folder. For all the coding purposes between the components, code are written from scratch i.e. Visual Studio Code.
- After creating components like the user login, register, terms and conditions, Booking forms, payment page, vendor form, forgot password and new password, write code for the respective HTML , bootstrap node JS , CSS and javascript.
- Once after writing code for the respective component, import all routes for those components in the main app.js component.
- The main task is to connect the application to backend in order to collect the users information and event information and store that in db
- After all that we need to integrate the payment page from the payment api
- We need to provide the chatbot for the 24/7 support by doing the chatbot by coverflow ChatBot
- It is done by creating a cluster followed by database.



**Fig 4.1: System Architecture.**

## 4.2 ARCHITECTURE / OVERALL DESIGN OF PROPOSED SYSTEM



**Fig 4.2: Flow Diagram**

#### 4.3 DESCRIPTION OF SOFTWARE FOR IMPLEMENTATION AND TESTING PLAN OF THE PROPOSED MODEL/SYSTEM

- Design simple views for each state in your application, and React will efficiently update and render just the right components when your data changes.
- React components implement a `render()` method that takes input data and returns what to display. This example uses an XML-like syntax called JSX.
- React allows you to interface with other libraries and frameworks.
- The react JS framework works by first reading the Hypertext Mark-up Language (HTML) page, which has additional custom HTML, CSS and Bootstrap attributes embedded into it.
- React interprets those attributes as directives to bind input or output parts of the page to a model that is represented by standard JavaScript variables.
- The values of those JavaScript variables can be manually set within the code, or retrieved from static or dynamic JSON resources.
- React JS is a toolset for building the framework most suited to your application development.
- It is fully extensible and works well with other libraries.
- Every feature can be modified or replaced to suit your unique development workflow and feature needs.

## CHAPTER 5

### IMPLEMENTATION DETAILS

#### 5.1 DEPLOYMENT SETUP

package.json files in server and client Folder manage packages for the server and client respectively. In the root folder we need another package.jsonfile to manage the deployment, similar to how we handled it with Heroku

We are assuming that your express server is serving client's production build from the client/build folder.

"main"key will have the path to your server's entry point file:

```
"main": "./server/index.js",
```

And start script will execute command to go into the server folder (from the root), install the packages and start the main server's file:

```
"start": "cd ./server && node index.js"
```

From Render.com's dashboard click "New +" button and select "Web Service".

Connect to the GitHub repository you want to use by linking to your GitHub account and searching for the repo's name. Once connected provide a name for this project, region for the server to be in, choose which branch you want to use and specify the root folder which should be our ./server if server is going to serve the build of your client.

The build command could look this:

```
npm i && cd ../client && npm i && npm run build
```

meaning that from the root folder (/server) in our case we will install all the packages for the server, then go to the client folder, install packages and create a production build.

For the start command it can be

```
node index.js
```

to start our server.

#### 5.2 ALGORITHM

By dictionary definition, hashing refers to "chopping something into small pieces" to make it look like a "confused mess". That definition closely applies to what hashing represents in computing. In cryptography, a hash function is a mathematical algorithm that maps data of any size to a bit string of a fixed size. We can refer to the function input as message or simply as input. The fixed-size string function output is known as the hash or the message digest. As stated by OWASP, hash functions used in cryptography have the following key properties:

- It's easy and practical to compute the hash, but "difficult or impossible to re-generate the original input if only the hash value is known."
- It's difficult to create an initial input that would match a specific desired output.

Thus, in contrast to encryption, hashing is a one-way mechanism. The data that is hashed cannot be practically "unhashed". Commonly used hashing algorithms include Message Digest (MDx) algorithms, such as MD5, and Secure Hash Algorithms (SHA), such as SHA-1 and the SHA-2 family that includes the widely used SHA-256 algorithm. Later on, we are going to learn about the strength of these algorithms and how some of them have been deprecated due to rapid computational advancements or have fallen out of use due to security vulnerabilities. In bitcoin, integrity and block-chaining use the SHA-256 algorithm as the underlying cryptographic hash function. Let's look at a hashing example using SHA-256 and Python. There are many hashing algorithms that people have used to hash passwords. Two of the most common hashing algorithms you may have come across are MD5 and the SHA-\* family of algorithms (SHA-1, SHA-2, SHA-3), but there are several reasons not to use these. For starters, they are extremely fast. In most areas of computing, you want fast algorithms, but for password hashing, the opposite is true. A slow hashing function makes brute-forcing passwords an expensive, time-consuming process thereby protecting organizations from such attacks. Cryptographic hashing algorithms like MD5 and the SHA family of algorithms make it easier for attackers to brute force passwords as attackers can use a lot of compute resources (CPUs, GPUs, etc.) to make a lot of password "guesses" in a short period of time. Another problem with these algorithms is that they don't natively use a salt, which is a useful tool in protecting hashed passwords. Unless coerced, users often take the path of least resistance by using passwords that are easily guessed. An attacker can hash lots of common passwords then search a database of stolen password digests to find a match. A salt can fix that problem. Salts are random, unique strings that are either appended or prepended to the user's password before hashing. Because the salt changes the input, it produces a unique hash that won't match with any stolen digests. Last.fm and Lifeboat both used unsalted MD5 hashes for their password storage, which made it extremely easy for attackers to find all of their users' plain text passwords.

## 5.3 TESTING

### 5.3.1 System Testing

Quality assurance is an important step in software engineering. This overlaps with all the phases of development right from the requirement analysis. This quality requirement of the software system must be clearly extracted during the requirement analysis and all the subsequent phases should be made biased to that, the final testing will become trivial and less expensive. There are number of quality parameters like correctness, accuracy, reliability, robustness, efficiency, effectiveness, reusability, maintainability etc.. The state of requirement of each of these parameters will vary depending upon the name and domain of the application. The testing should be done at the end of all development steps. Even though the final testing and verification are inevitable for better life and functionality of the software. The different software testing approaches and methods like white box testing and black box testing. The major phases in testing are design of test plan, setting up test case and test candidate and test procedure, testing and correction. This is a cycle process and the software will circulate through all the steps till it attends the required quality. The testing is carried in the following steps.

### 5.3.2 Unit testing

Unit testing focuses verification effort on the smallest unit of software design the module. Using the details design description as a guide, important control paths are tested to uncover errors within boundary of the boundary of the module. The relative complexity of tests and the errors detected as a result is limited by the constrained scope established for unit testing. Unit testing is normally considered an adjacent to coding steps. After source level code has been developed, reviewed, and verified for correct syntax, unit test case design begins. A review of design information provides guidance for establishing test cases that are likely to uncover error in each case of the 27 categories discussed above. Each test case should be coupled with a set of expected results

### 5.3.3 Integration testing

Integration testing is systematic technique for constructing the program structure while at the same time conducting test to uncover error associated with interfacing .The objective is to take unit tested modules and build a program structure that has been dictated by design. There is often a tendency to attempt no incremental integration; that is to construct the program using a “big bang “approach. The entire

modules are combined in advance. The entire program is tested as whole and chaoses usually result! A set of error is encountered. Correction is difficult because the isolation of cause is complicated by the vast expanse of entire program. Once errors are corrected, new ones appear and process continues in a seemingly endless loop.

## CHAPTER 6

### RESULTS AND DISCUSSION

**PERSONAL DETAILS**

User Name

Bride Name

Groom Name

Date

From

To

City

No of Guests

**Fig 6.1: Event Page form**

**PostWedding Event - [ 2023-04-27 ]**  
Order ID - [ d5f42 ]

|                      |                          |
|----------------------|--------------------------|
| Name of the customer | ram                      |
| Event Name           | PostWedding Event        |
| Order Id             | d5f42                    |
| Date of event        | 2023-04-27               |
| Date of Booking      | 2023-04-16T16:00:06.481Z |
| No of Guests         | 220                      |

| Event          | Promise Date | Confirmed                              | Reason               | Status                                 |
|----------------|--------------|----------------------------------------|----------------------|----------------------------------------|
| muh Dikhai     | dd-mm-yyyy   | <input type="checkbox"/> NOT CONFIRMED | <input type="text"/> | <input type="button" value="PENDING"/> |
| subaarambhatra | dd-mm-yyyy   | <input type="checkbox"/> NOT CONFIRMED | <input type="text"/> | <input type="button" value="PENDING"/> |
| Honeymoon      | dd-mm-yyyy   | <input type="checkbox"/> NOT CONFIRMED | <input type="text"/> | <input type="button" value="PENDING"/> |

**Fig 6.2: Profile page (Event details)**

## Engagement Event Booking

**Shows :**                      Sangeet Choreography ☐                      Music ☐                      Dance ☐

---

**Decorations :**                      Regular Decoration ☐                      Theme Decoration ☐

**Catering :**                      ☐ Veg                      ☐ Jain

**Food Type**

Select..

**Other Services :**                      Invitation ☐                      Pooja Pandit Ji ☐                      Venue ☐                      Photography ☐

Beauty ☐                      Mehendi ☐                      Hosting ☐

---

**Estimate Budget :**                      Minimum                      Maximum

---

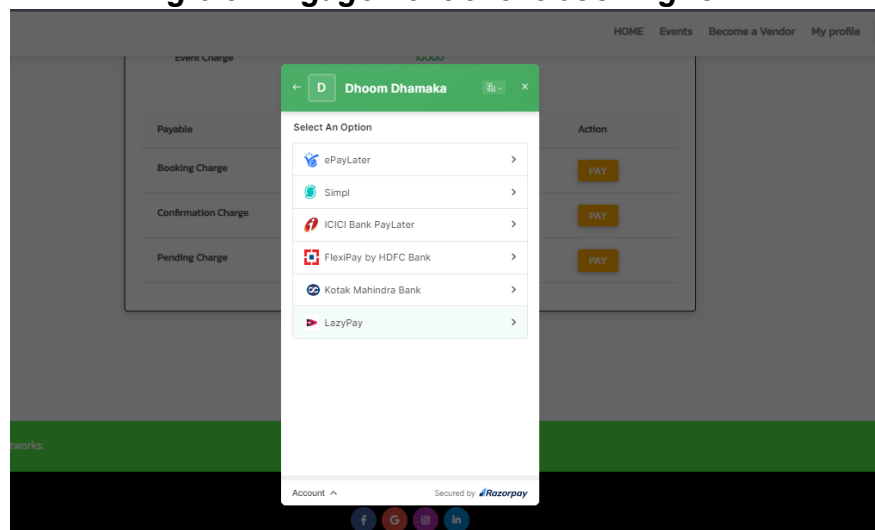
**Special Service :**  
Any additional service you expect from us that may be mentioned here:

**Note 1 :**  
**Once the form is submitted now, Our customer representative will contact you within 12-24 hours to discuss in more detail.**

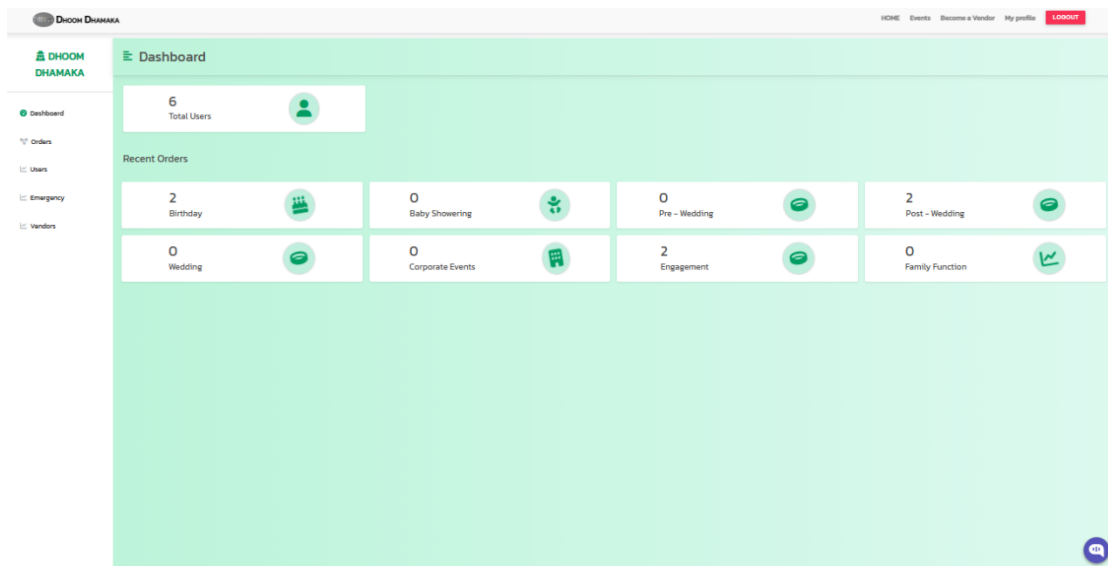
**Note 2 :**  
**[Terms and Conditions apply](#)**

**SUBMIT**

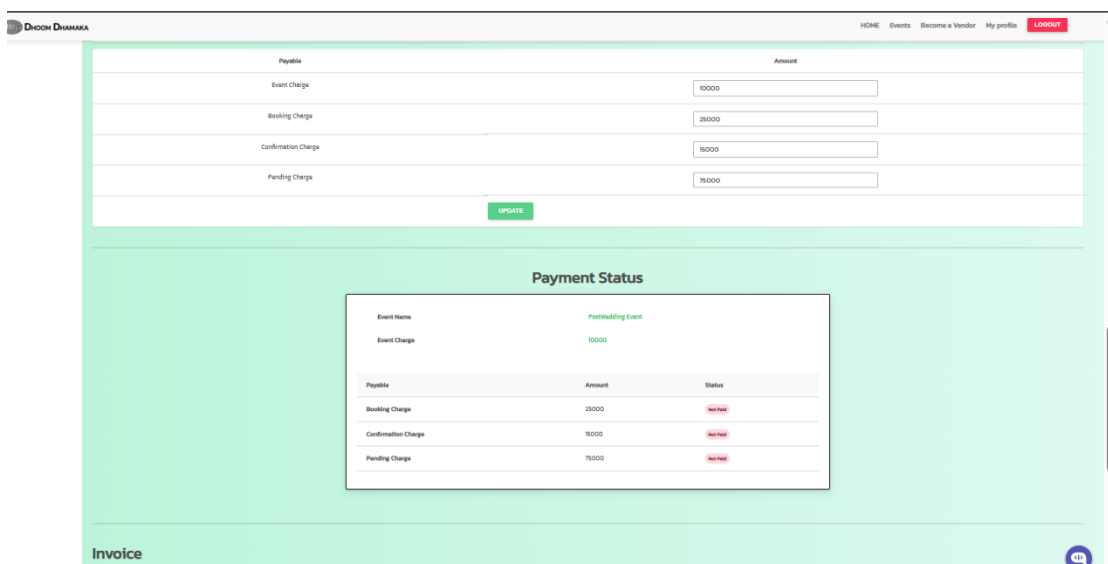
**Fig 6.3: Engagement event booking form**



**Fig 6.4: Payment Page**



**Fig 6.5: Admin Dashboard**



**Fig 6.6: Admin Payment update page**

The Multi-Facility Function Coordinator using React JS is an innovative web application that simplifies the event booking process. The application follows a step-by-step process that guides the user through the entire booking process. The user is required to enter their event details, and once the process is completed, a pop-up appears on their screen. This allows the user to be redirected to the home page where they can view the overview of their selected event in the event tab under the profile page. This feature helps users to keep track of their events and make any necessary changes before the event. Once the event is selected and the booking process is complete, the confirmation process occurs on the client side within 7-10

hours. This feature ensures that the user has ample time to make any necessary changes before the event. The application also allows for easy payment processing, which is updated by the admin. By using this web application, users can easily book all kinds of events with ease and comfort. The simplified booking process reduces the time it takes to book an event, making it convenient for users to book events from the comfort of their homes. Moreover, the web application's intuitive design and user-friendly interface allow users to navigate the site with ease. The application's interface is designed to cater to users with varying levels of technical expertise, ensuring that anyone can use the application without difficulty. The application's developers also ensure that the website is frequently updated with the latest features to enhance the user experience. Additionally, the web application is secure, ensuring that users' personal information and payment details are kept safe. Overall, the Multi-Facility Function Coordinator using React JS is an excellent web application that simplifies the event booking process and offers a seamless user experience.

## **CHAPTER 7**

### **CONCLUSION**

#### **7.1 CONCLUSION**

The article proposes the Multi-facility function coordinator using React JS, which aims to provide an easy-to-use platform for customers to book event organizers. The proposed system uses a user behavior-based authentication process, where user profiling is used to create a login method for user authentication. This authentication process is more secure and helps to protect user data from unauthorized access. Additionally, the proposed system allows easy management of user orders by the admin system, which can be sorted based on date. This helps the admin to manage their system efficiently and keep track of all the orders. One of the main features of the proposed system is that it brings all the event booking services to one place, making it easier for the customer to book services without having to switch between different websites or platforms. The system also provides a pop-up notification on the user's screen after the details have been entered, and the user is then redirected to the home page. The selected event's overview is displayed in a card under the event tab on the profile page, which allows users to review their bookings and track their orders. Additionally, the proposed system has a confirmation process that occurs on the client-side within 7-10 hours, which provides transparency to the user and helps to avoid any conflicts.

Another advantage of the proposed system is that it reduces the time required for booking each service. With the system's easy-to-use interface and streamlined booking process, users can quickly find the event services they need and book them efficiently. The proposed system's admin system also allows for easy deletion of orders, which helps to keep the system clean and organized. Overall, the proposed Multi-facility function coordinator using React JS provides an excellent solution for users who are looking for an efficient and secure platform to book event services..

#### **7.2 FUTURE WORK**

1. Adding a real-time booking confirmation feature: Instead of waiting for 7-10 hours for the confirmation process to occur, the web application can be enhanced to provide real-time confirmation to the users. This can be achieved by integrating the web application with a messaging platform that will notify users of the booking status as soon as the confirmation process is completed.

2. Implementing a feedback system: It is essential to gather feedback from users to improve the quality of the web application. Therefore, a feedback system can be implemented to allow users to rate their experience, provide suggestions for improvement, and report any issues they encountered while using the web application. This will help the developers to make necessary improvements and address user concerns.
3. Real-time availability updates: Currently, users may not be aware of the availability of facilities or amenities in real-time. By integrating with a real-time availability system, users can see the current availability of facilities and amenities while making their booking. This can reduce the chances of double bookings and ensure that users can book events with confidence.

## REFERENCES

- [1] ER. L. Sebesta, "An Introduction to React: The JavaScript Library for Building User Interfaces," IEEE Software, vol. 35, no. 3, pp. 101-103, May-Jun. 2018.
- [2] S. K. Madan and R. K. Madan, "An Overview of React Library and Its Application in Web Development," International Journal of Advanced Computer Science and Applications, vol. 10, no. 7, pp. 427-432, Jul. 2019.
- [3] A. Chen, "React Native: A Cross-Platform Framework for Mobile App Development," IEEE Internet Computing, vol. 23, no. 6, pp. 92-97, Nov-Dec. 2019.
- [4] S. J. Zaka and S. A. Memon, "React: A Comparative Study of Frontend Web Frameworks," International Journal of Computer Applications, vol. 179, no. 34, pp. 27-31, Feb. 2021.
- [5] N. S. M. Z. Shah and F. A. Memon, "ReactJS: A Comprehensive Survey on Its Architecture and Features," International Journal of Scientific & Engineering Research, vol. 11, no. 10, pp. 2058-2062, Oct. 2020.
- [6] N. V. Kumar and D. Raj, "A Study on React and Its Components," International Journal of Advanced Research in Computer Science, vol. 11, no. 5, pp. 20-24, Sep.-Oct. 2020.
- [7] R. Jindal and S. Jindal, "A Comparative Study of Angular and React Frameworks for Frontend Web Development," International Journal of Advanced Research in Computer Science and Software Engineering, vol. 10, no. 4, pp. 561-568, Apr. 2020.
- [8] H. D. R. Kusuma and W. Wijaya, "React Framework and Its Implementation in Web Application Development," Journal of Physics: Conference Series, vol. 1655, no. 1, pp. 012030, Mar. 2020.
- [9] T. H. Khan and M. Hussain, "Performance Evaluation of React and AngularJS Frameworks for Web Development," International Journal of Computer Science and Network Security, vol. 20, no. 1, pp. 167-174, Jan. 2020.
- [10] S. B. Tamboli and S. S. Chavan, "A Review on React and Its Applications in Web Development," International Journal of Innovative Technology and Exploring Engineering, vol. 9, no. 2, pp. 1752-1756, Nov. 2019.
- [11] A. B. Smith, "Design and Implementation of an Online Booking System for Hotels," Journal of Hospitality and Tourism Technology, vol. 11, no. 2, pp. 158-169, May 2020.

- [12] P. Liang and K. Li, "Design and Implementation of an Online Reservation System for Car Rental Companies," *Journal of Information Science and Engineering*, vol. 35, no. 5, pp. 1149-1166, Sep. 2019.
- [13] A. R. Khaliq and A. Asghar, "Implementation of an Online Bus Ticket Reservation System," *Journal of Computer Science and Technology*, vol. 19, no. 4, pp. 723-732, Jul. 2019.
- [14] R. T. Chen and M. C. Chen, "Design and Implementation of an Online Flight Booking System," *International Journal of Electronics and Electrical Engineering*, vol. 7, no. 3, pp. 163-170, May 2019.
- [15] L. J. Zhang and Y. J. Li, "Development of an Online Hotel Booking System," *International Journal of Digital Content Technology and its Applications*, vol. 13, no. 1, pp. 201-207, Jan. 2019.
- [16] R. K. Sah and P. Pandey, "Design and Implementation of an Online Movie Ticket Booking System," *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 9, no. 4, pp. 31-34, Apr. 2019.
- [17] S. R. Thapa and S. S. Shrestha, "Design and Implementation of an Online Restaurant Reservation System," *International Journal of Engineering Research and Applications*, vol. 8, no. 5, pp. 75-80, May 2018.
- [18] K. M. Kaur and P. Singh, "Design and Implementation of an Online Train Ticket Reservation System," *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, vol. 4, no. 1, pp. 82-86, Jan.-Feb. 2018.
- [19] S. Goyal and S. Rani, "Design and Implementation of an Online Ticket Booking System for Amusement Parks," *International Journal of Computer Science and Mobile Computing*, vol. 6, no. 3, pp. 86-94, Mar. 2017.
- [20] M. N. M. Yusof and N. M. Jusoh, "Design and Development of an Online Booking System for a Sport Complex," *International Journal of Advances in Science, Engineering and Technology*, vol. 5, no. 2, pp. 40-44, Feb. 2017.

## APPENDIX

### A. SOURCE CODE

#### [1] User Login component

```
import React, { useState } from "react";
import axios from "axios";
import { useDispatch } from "react-redux";
import { Link, NavLink, useNavigate } from "react-router-dom";
import { set, useForm } from "react-hook-form";
import * as yup from "yup";
import { yupResolver } from "@hookform/resolvers/yup";
import { authActions } from "../../store";
const schema = yup.object().shape({
  email: yup.string().required("Email must be required"),
  password: yup.string().required("password must be required"),
});
export default function Login() {
  const [errorLogin, setErrorLogin] = useState(null);
  const {
    register,
    handleSubmit,
    formState: { errors },
  } = useForm({ resolver: yupResolver(schema) });
  const dispatch = useDispatch();
  const navigate = useNavigate();
  axios.defaults.withCredentials = true;
  function handleSubmit2(data) {
    axios
      .post("api/login", { data })
      .then((res) => {
        const lctTok = res.data.jsonToken;
        console.log(res);
        localStorage.setItem("bandhanUserToken", lctTok);
        dispatch(authActions.login());
      });
  }
}
```

```

        navigate("/events");
    })
    .catch((err) => {
        if (err.response.status === 400) {
            if (err.response.data) {
                setErrorLogin("Invalid Email or Password");
            } else {
                setErrorLogin(err.response.data)
            }
        } else {
            setErrorLogin(err.response.data)
        }
        // navigate("/login");
        // if(err.sta)
    });
}
return (
    <div className="App">
        <section
            class="vh-100 bg-image"
            style={{
                backgroundImage:
                    "url('https://mdbcdn.b-cdn.net/img/Photos/new-templates/search-
box/img4.webp')",
            }}
        >
            <div class="mask d-flex align-items-center h-100 gradient-custom-3">
                <div class="container h-100">
                    <div class="row d-flex justify-content-center align-items-center h-100">
                        <div class="col-12 col-md-9 col-lg-7 col-xl-6">
                            <div class="card" style={{ borderRadius: "15px" }}>
                                <div class="card-body p-5">
                                    <h2 class="text-uppercase text-center mb-5 fw-bold">
                                        Login To Account

```

</h2>

```
<form
  onSubmit={handleSubmit((data) => {
    handleSubmit2(data);
  })}
>
  {errorLogin !== null && <div class="alert alert-danger mb-5 fw-bold
text-center" role="alert">
    {errorLogin}
  </div>}
  <div class="form-outline border-bottom mb-4">
    <input
      {...register("email")}
      name="email"
      type="email"
      id="form3Example3cg"
      class="form-control form-control-lg"
    />
    <label
      class="form-label fw-bold text-muted"
      for="form3Example3cg"
    >
      Email
    </label>
  </div>
  {errors.email && (
    <div class="alert alert-danger mt-2" role="alert">
      {errors.email?.message}
    </div>
  )}

  <div class="form-outline border-bottom mb-4">
    <input
```

```

    {...register("password")}
    name="password"
    type="password"
    id="form3Example4cg"
    class="form-control form-control-lg"
  />
  <label
    class="form-label fw-bold text-muted"
    for="form3Example4cg"
  >
    Password
  </label>
</div>
{errors.password && (
  <div class="alert alert-danger mt-2" role="alert">
    {errors.password?.message}
  </div>
)}
<div class="d-flex justify-content-center mt-5">
  <button
    type="submit"
    class="btn btn-success btn-block btn-lg gradient-custom-4 fw-bold
text-body"
  >
    Login <i class="fa-solid fa-unlock ps-2"></i>
  </button>
</div>
<p class="text-center text-muted mt-5 mb-0">
  Not have an account ?
  <Link class="fw-bold text-body" to={"/register"}>
    <u> Register here</u>
  </Link>
</p>
<p class="text-center text-muted mt-2 mb-0">

```



```

    dispatch(authActions.logout());
    navigate("/login");
    console.log("Succesfully logged out");
  }
  if (!!localStorage.getItem("bandhanUserToken")) {
    axios
      .get("api/login/getLoginStatus")
      .then((res) => {
        if (res.status === 200) {
          console.log("soop...");
          setDisplayEve(true);
        } else {
          doLogout();
        }
      })
      .catch((err) => {
        doLogout();
      });
  }
}, []);
return (
  <div>
    <Babyshowercarousel />
    <Babyshowerprotofolio />
    <BabyShowerForm />
  </div>
);
}

```

### [3] Payment component

```
import axios from "axios";
import React, { useEffect, useState } from "react";
import { authActions } from "../store";
import { useDispatch } from "react-redux";
import { useNavigate } from "react-router-dom";
import "../payment.css";
import PaymentDetail from "../paymentDetail";

function Payment() {
  const dispatch = useDispatch();
  const navigate = useNavigate();
  const [datas, setdata] = useState([]);
  const [currentDetail, setCurrentDetail] = useState();
  const [display, setDisplay] = useState(false);

  useEffect(() => {
    function doLogout() {
      localStorage.clear("bandhanUserToken");
      dispatch(authActions.logout());
      navigate("/login");
      console.log("Succesfully logged out");
    }
    if (!localStorage.getItem("bandhanUserToken")) {
      axios
        .get("api/login/getLoginStatus")
        .then((res) => {
          if (res.status === 200) {
            console.log("soop...");
          } else {
            doLogout();
          }
        })
    }
  })
}
```

```

        .catch((err) => {
            doLogout();
        });
    }
    axios.get("api/payment/getPaymentLists").then((res) => {
        if (res.status === 200) {
            setdata(res.data);
        }
    });
}, []);
async function detailsCall121(item)
return (
    <div className="mb-10 mt-5" style={{minHeight: "80vh"}}>
        <div className="row justify-content-center mx-5">
            {datas.map((item, i) => {
                return (
                    <div className="col-md-4" key={i}>
                        <div className="card rounded shadow-4-strong">
                            <div className="card-body text-center">
                                <h3 className="mb-4 fw-bold">{item.eventName}</h3>
                                <button
                                    type="button"
                                    className="btn btn-secondary"
                                    onClick={() => detailsCall121(item)}
                                >
                                    Details
                                </button>
                            </div>
                        </div>
                    </div>
                );
            })}
        </div>
    </div>
);
}}
</div>

```

```

    {display && <PaymentDetail data={currentDetail} />}
  </div>
);
{
  /* <div className="row justify-content-center">
    <div className="col-md-8 mt-4 mb-7">
      <h3 className=" fw-bold text-center mb-4">Invoices</h3>
      <table class="table align-middle mb-0 bg-white">
        <thead class="bg-light">
          <tr className=" text-center">
            <th className=" fw-bolder text-capitalize">Title</th>
            <th className=" fw-bolder text-capitalize">Amount</th>
          </tr>
        </thead>
        <tbody>
          <tr>
            <td className=" text-center">
              <p class="fw-normal mb-1">Designer</p>
              <p class="text-muted mb-0">UI/UX</p>
            </td>
            <td className=" text-center">₹10,0000</td>
          </tr>
        </tbody>
      </table>
    </div>
  </div>
  <div className="row justify-content-center">
    <div className=" col-md-4">
      <div class="card text-center rounded shadow-5-strong mb-7">
        <div class="card-body border-5">
          <h5 class="card-title mb-4 mt-3 fw-bold">Payment</h5>

          <div className="row justify-content-center">
            <div className="col-md-8">

```

```

    <div class="form-floating mb-5">
      <input
        type="number"
        id="form12"
        class="form-control mb-4"
        placeholder="Enter Amount"
        value={orderAmount}
        onChange={(e) => setamount(e.target.value)}
      />
      <label for="floatingInput">Enter amount ₹</label>
    </div>
  </div>
</div>
<button
  href="#"
  type="button"
  class="btn btn-primary btn-block btn-lg"
  disabled={loading}
  onClick={loadRazorpay}
>
  Proceed to payment <i class="fas fa-long-arrow-alt-right"></i>
</button>
</div>
</div>
</div>
</div> *
}}
export default Payment;

```

#### [4] Backend Server module

```

require("dotenv").config();
const express = require("express");
const cors = require("cors");
const { urlencoded } = require("express");

```

```
const app = express();
const fileupload = require("express-fileupload");
const cookieParser = require("cookie-parser");
app.use(cookieParser());
app.use(cors());
app.use(fileupload());
app.use(express.static("files"));
app.use(express.json());
app.use(urlencoded({ extended: true }));
require("./startups/db")();
require("./startups/config")();
require("./startups/routes")(app);
```

```
// const port = process.env.PORT || 3001;
const port = 3001;
app.listen(port, () => {
  console.log(`listening port on ${port}`);
});
```

#### [5] Routes Module :

```
const register = require("../routes/register");
const login = require("../routes/login");
const profile = require("../routes/profile");
const wedding = require("../routes/wedding");
const familyFunction = require("../routes/familyFunction");
const corporateEvent = require("../routes/corporateEvent");
const birthday = require("../routes/birthday");
const engagement = require("../routes/engagement");
const prewedding = require("../routes/prewedding");
const postwedding = require("../routes/postwedding");
const babyshower = require("../routes/babyshower");
const eventsinfos = require("../routes/eventsinfos");
const home = require("../routes/home");
const todo = require("../routes/todo");
```

```

const newPassword = require("../routes/newPassword");
const passwordReset = require("../routes/passwordReset");
const adminuserlist = require("../routes/adminuserlist");
const becomeVendor = require("../routes/becomeVendor")
const emergencyform = require("../routes/emergencyform")
const setUsers = require('../routes/setUser')
const imgupload = require('../routes/imgupload');
const logout = require("../controllers/user-controller");
const birthdayorder = require("../routes/adminorderslist/birthdayorder")
const vendoradminlist = require("../routes/vendoradminlist");
const auth = require("../middleware/auth");
const adminauth = require("../routes/adminauth")
const PaymentRoute = require("../routes/paymentRoute")
const Invoices = require("../routes/invoices")
module.exports = function (app) {
  app.use("/api/register", register);
  app.use("/api/login", login);
  app.use("/api/logout", login);
  app.use("/api/passwordReset", passwordReset);
  app.use("/api/newPassword", newPassword);
  app.use("/api/becomeVendor", becomeVendor)
  app.use("/api/profile", profile);
  app.use("/api/home", home);
  app.use("/api/setUser", setUsers)
  app.use("/api/todo",todo )
  app.use("/api/birthday", birthday )
  app.use("/api/engagement", engagement )
  app.use("/api/prewedding", prewedding)
  app.use("/api/postwedding", postwedding)
  app.use("/api/babyshower", babyshower)
  app.use("/api/wedding", wedding)
  app.use("/api/familyFunction", familyFunction)
  app.use("/api/corporateEvent", corporateEvent)
  app.use("/api/eventInfo",auth,eventsinfos )

```

```

app.use("/api/isadmincheck",auth, adminauth)
app.use("/api/adminuserlist",auth, adminuserlist)
app.use("/api/emergencyform",emergencyform)
app.use("/api/vendoradminlist",vendoradminlist)
app.use("/api/payment", PaymentRoute)
app.use("/api/invoice",Invoices)
};

```

#### [6] Middleware auth.js :

```

const config = require("config");
const jwt = require("jsonwebtoken");
const dotenv = require("dotenv");
const { User } = require("../model/user");

function auth(req, res, next) {
  const token = req.cookies.bandhanUserToken;
  const jwtPrivateKey = process.env.jwtPrivateKey;
  if (!token) res.status(400).send("Accessss Denied. No Token Provided");

  try {
    const decoded = jwt.verify(token, jwtPrivateKey);
    req.user = decoded;
    req.id = decoded._id;
    req.userName = decoded.username;
    next();
  } catch (ex) {
    res.clearCookie("bandhanUserToken");
    res.status(400).send("Invalid Token");
  }
}

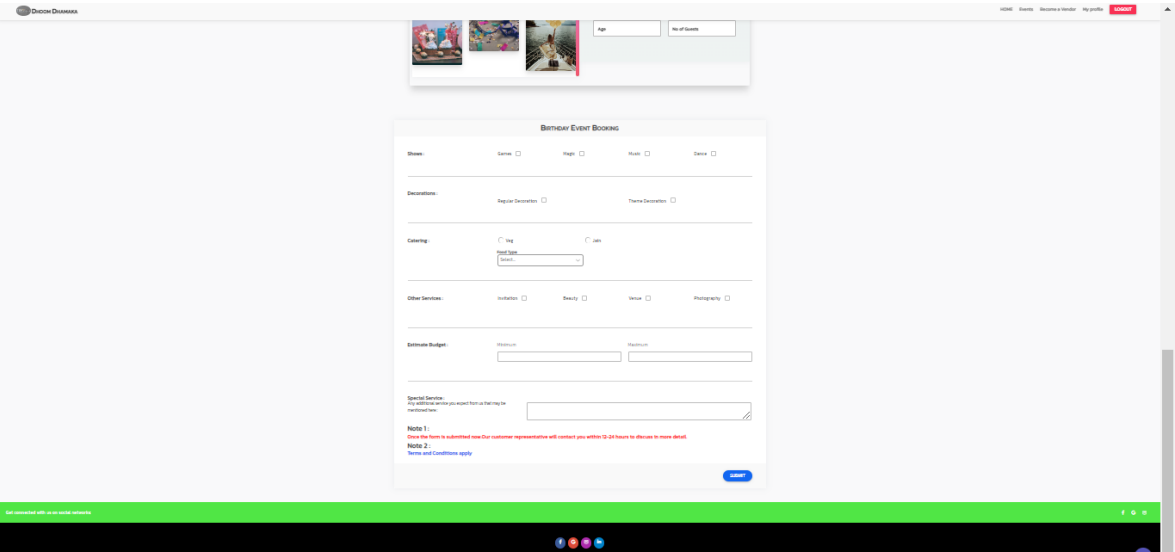
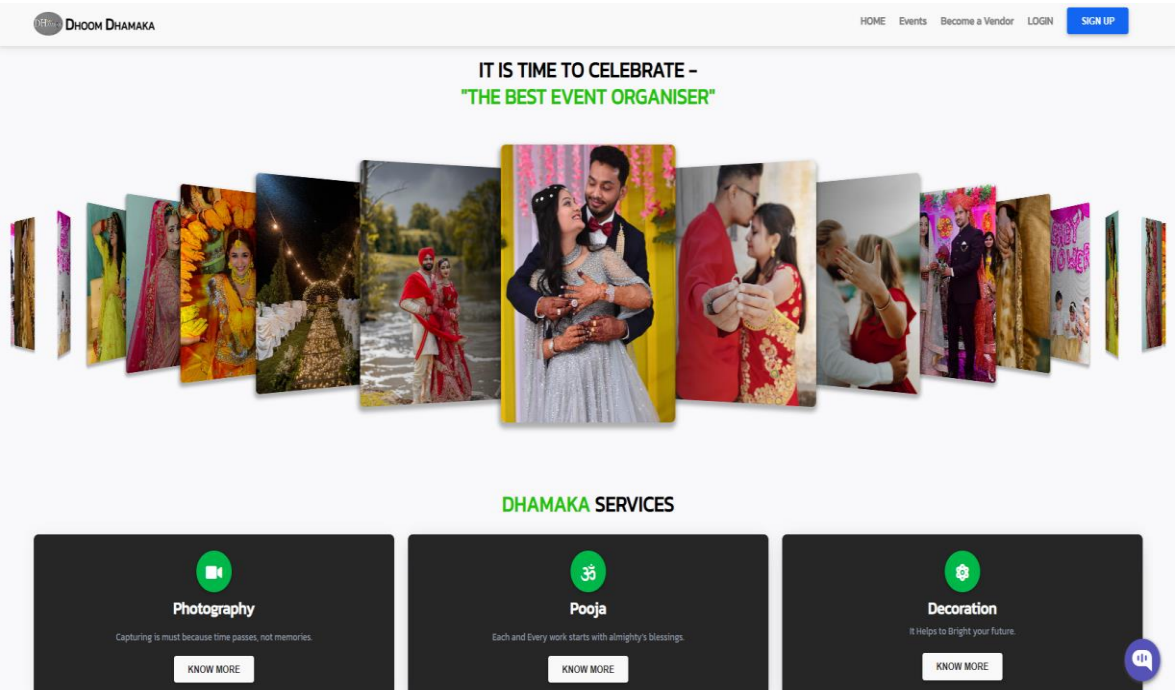
const getUser = async (req, res, next) => {
  console.log("dddd");
  const userId = req.id;
  console.log(userId);
}

```

```
let user;
user = await User.findById(userId, "-password");
if (!user) {
  console.log("User not found");
  return res.status(404).json({ message: "User Not Found" });
} else {
  console.log("User found");
  return res.status(200).json({ user });
}
next();
};

module.exports = auth;
```

# SCREENSHOTS





[HOME](#) [Events](#) [Become a Vendor](#) [LOGIN](#) [SIGN UP](#)

## LOGIN TO ACCOUNT

Email

Password

LOGIN

Not have an account ? [Register here](#)

Forgot Password ? [Reset here](#)



[HOME](#) [Events](#) [Become a Vendor](#) [LOGIN](#) [SIGN UP](#)

## CREATE AN ACCOUNT

Username

Email

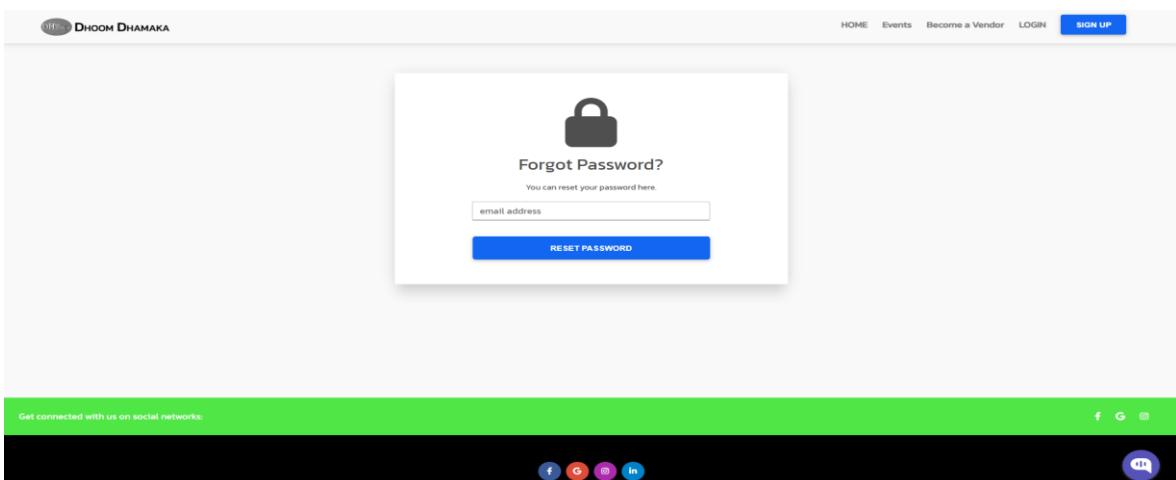
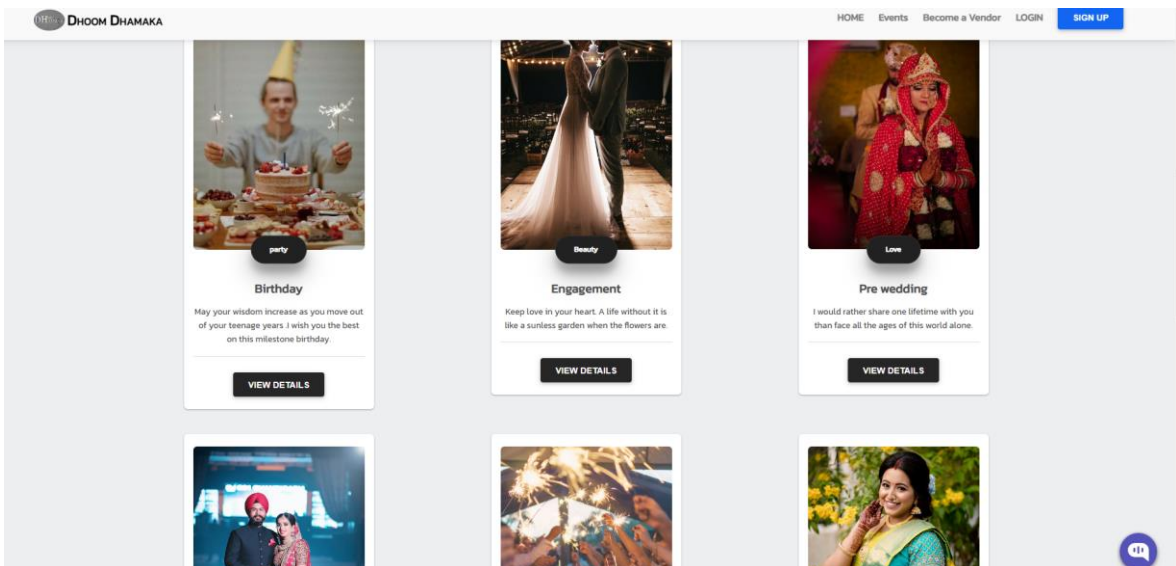
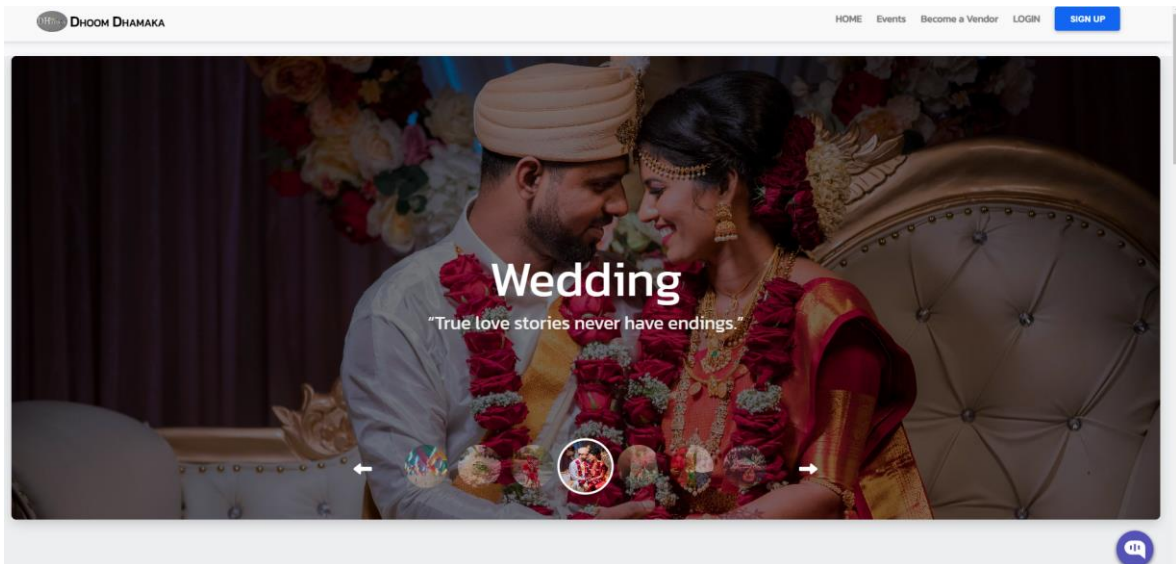
Number

Password

Confirm password

REGISTER

Have already an account ? [Login here](#)



## PRIVACY POLICY

Last updated: 20 September 2022

This Privacy Policy describes Our policies and procedures on the collection, use and disclosure of Your information when You use the Service and tells You about Your privacy rights and how the law protects You. We use Your Personal data to provide and improve the Service. By using the Service, You agree to the collection and use of information in accordance with this Privacy Policy.

### Interpretation and Definitions

#### Interpretation

The words of which the initial letter is capitalized have meanings defined under the following conditions. The following definitions shall have the same meaning regardless of whether they appear in singular or in plural.

#### Definitions

For the purposes of this Privacy Policy:

- **Account** means a unique account created for You to access our Service or parts of our Service.
- **Company** (referred to as either 'the Company', 'We', 'Us' or 'Our' in this Agreement) refers to Dhoom Dhamaka.
- **Cookies** are small files that are placed on Your computer, mobile device or any other device by a website, containing the details of Your browsing history on that website among its many uses.
- **Country** refers to: Tamil Nadu, India
- **Device** means any device that can access the Service such as a computer, a cellphone or a digital tablet.
- **Personal Data** is any information that relates to an identified or identifiable individual.
- **Service** refers to the Website.
- **Service Provider** means any natural or legal person who processes the data on behalf of the Company. It refers to third-party companies or individuals employed by the Company to facilitate the Service, to provide the Service on behalf of the Company, to perform services related to the Service or to assist the Company in analyzing how the Service is used.
- **Third-party Social Media Service** refers to any website or any social network website through which a User can log in or create an account to use the Service.
- **Usage Data** refers to data collected automatically, either generated by the use of the Service or from the Service infrastructure itself (for example, the duration of a page visit).
- **Website** refers to Dhoom Dhamaka, accessible from [dhoomdhamaka.com](http://dhoomdhamaka.com)
- **You** means the individual accessing or using the Service, or the company, or other legal entity on behalf of which such individual is accessing or using the Service, as applicable.

### Collecting and Using Your Personal Data

#### Types of Data Collected

##### Personal Data

While using Our Service, We may ask You to provide Us with certain personally identifiable information that can be used to contact or identify You. Personally identifiable information may include, but is not limited to:



## Terms & Conditions

This Event Management Agreement regulates transactions between Dhoom Dhamaka and its customer. This agreement is applicable for all events organized by Dhoom Dhamaka Team. Rules and agreements vary from one event to another.

### 1)Booking Policy:

#### A)General booking :

It is a type of booking where the customer pays the total cost of the event only for the services chosen by him/her. Simply put, there is no extra charge for booking an event.

General booking details:

Birthday/Engagement/Family Function/Baby Shower/ Corporate Event - 10 Days or Prior Pre Wedding or Post Wedding - 12 Days or Prior Wedding - 15 Days or Prior Destination Wedding - 20 Days or prior

#### B)Premium Booking :

It is a type of booking where the customer pays a premium fee for the event. As against, premium booking is a special service where the entire event can be booked in a shorter duration within the premium policy booking time frame given by Dhoom Dhamaka. In general terms, there is a special fee for booking any event under this booking. Once, the normal date has passed, the event is automatically booked under Premium Booking. Premium booking details: Birthday/Engagement/Family Function/Baby Shower/ Corporate Event - 5 Days or Prior Pre Wedding or Post Wedding - 5 Days or Prior Wedding - 7 Days or Prior Destination Wedding - 10 Days or prior

### 2)Payment Policy:

#### A)Payment Structure:

The total payment is divided into three instalments. 1st Instalment - 25% of the total payment to be paid on the booking date. 2nd Instalment - 25% to be paid thereafter on the vendor confirmation date. Third Payment - The balance 50% should be paid on the day before the event or on the same day of the event. Once all the payments are received, the delivery of the service to the customer starts with a bang.

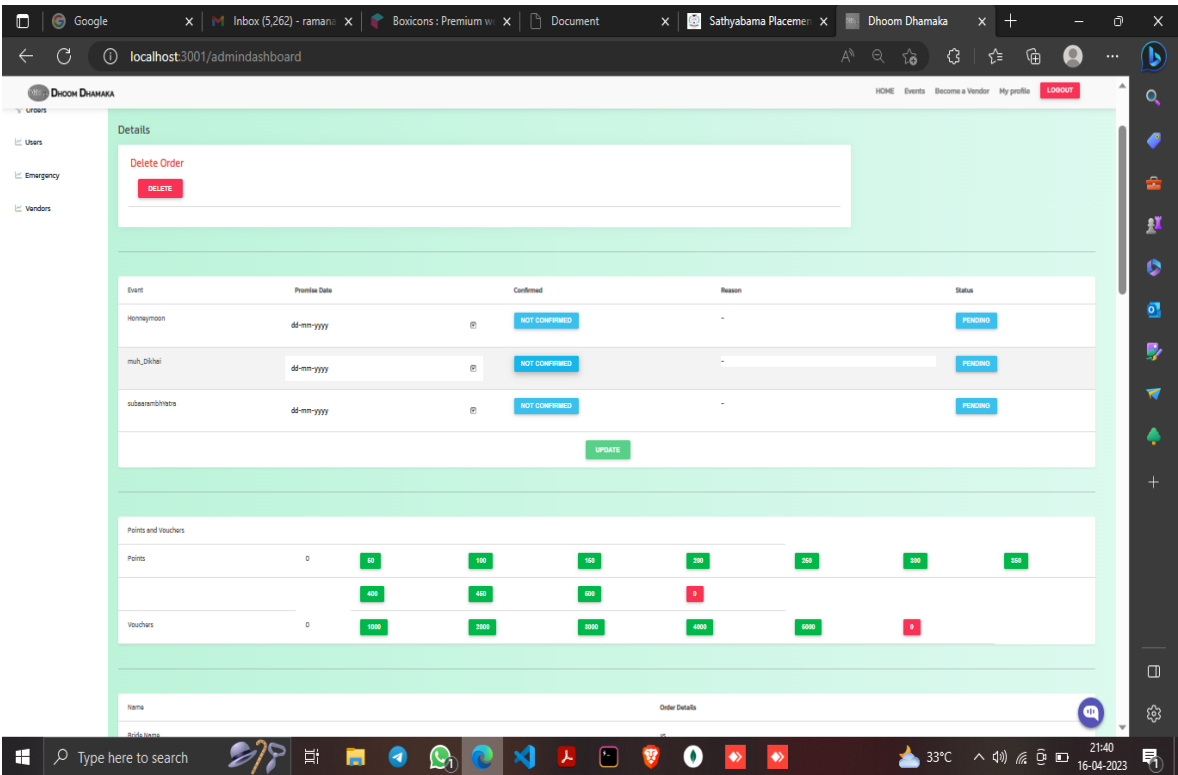
#### B)Payment Mode:

Payment of 1st and 2nd instalment is prepaid only. The payment of third instalment is both prepaid or cash. But cash payment is available on customer's request.

#### C)Penalty Charge due to delay in Payment:

If the customer delays the first payment, no booking confirmation will be given to the customer. If the customer delays the second payment, no vendor details will be shared with the customer. If the customer delays the third payment, delivery of service will be suspended till the third payment is made. If the due date of the first and second payment is passed, the penalty fee will be added. If the due date and time of the third payment passes, no service will be provided to the customer and no refund will be made for the previous payment. The third payment must be made six hours before the start of the event. Customer can make third payment before maximum 3 hours but customer will have to update this information to Dhoom Dhamaka team and will be fined for delay.





# Multi-Facility function coordinator using React JS

Surya S  
UG Scholar  
Department of IT  
Sathyabama Institute of Science and  
Technology  
Chennai, India  
[vasanthasurya484@gmail.com](mailto:vasanthasurya484@gmail.com)

Gowri S  
Associate Professor  
Department of IT  
Sathyabama Institute of Science and  
Technology  
Chennai, India  
[gowri.it@sathyabama.ac.in](mailto:gowri.it@sathyabama.ac.in)

**Abstract**— React and Bootstrap are two popular tools used for front-end web development. React is a JavaScript library that allows developers to build user interfaces based on UI components. It efficiently updates and renders the necessary components when data changes, making it a popular choice for building dynamic and responsive web applications. Bootstrap, on the other hand, is a CSS framework that provides pre-designed HTML, CSS, and JavaScript templates for various interface components, such as typography, forms, buttons, and navigation. This application helps users to book function hall, party hall, events, catering services, decoration services, online hotel accommodation and beauticians. In now a days, If we want to book a function hall, catering service, hotel accommodation, invitations etc..., for that we want go each and every office to book that hall or anything. For that process It took so much time and wandering for us. Our application rectifies this all problems. Now, users don't have to stand in a queue for hours and wait for their turn. After Booking the application admin panel members contact the user confirm the order. After confirming the event payment updated by the admin , payment gateway integrated in this application. User can also able to edit their own invitation cards. Thereby, Our application becomes a complete different types of event booking and management in single platform and reduces users effort and time. In this busy and hectic schedule, time management plays a vital role.

**Keywords**— *React JS , nodemailer , bcrypt , mongoose , razorpay, react redux and reacthook-form.*

## I. INTRODUCTION

The Internet and technologies were very different from what they are now just ten years ago. Technology was still "the future" and "an increasing market." At the same time, the Internet is something you can access from your computer and is genuinely understood by a small number of geeks, making today more present than ever. Today, however, we constantly carry "the Web" in our pockets thanks to mobile technology. Users no longer expect a website page to load slowly; they now demand it instantly. They will probably switch to another website if the page doesn't load within a couple of seconds. Performance of a company website is viewed as a barometer of the calibre of the company as a whole. Users are no longer need to wait

hours in a line before getting their turn. People found it simple to view all the online verticals at once thanks to the event management system's integration of technology improvements. Since single page applications (SPAs) can use repeating layouts and function as "content on demand" apps, they are more processing-efficient than standard MPA sites, can be built for less money, and require less time from developers.

## II. PROPOSED SYSTEM

The project on Multi-facility function coordinator in the place where clients come to book their function program in one place and pay for that. So, there is need to calculate how many services are booked and generate the bill for the client.

if there is a requirement to create a new system to alleviate the constraints of the current system. it is important to consider the requirements of both the clients and the vendors. The new system should be designed to meet the needs of both parties while addressing the bottlenecks that are currently present in the existing system.

It has following qualities:

- Lower processing costs.
- Fewer errors.
- Automated posting.
- Flexibility.
- Ability to meet user requirement Reduction in manpower.
- Improved reporting.
- Faster response time.
- Improves resources use

A proposed system should have the same qualities as the existing system, while also aiming to improve processing speed and ensure long-term use without errors or maintenance. The main objective of the project should be to

provide a convenient solution for billing patterns and create an easy-to-use environment for users and customers..

**Automation** – In simpler terms, the application replaces the manual system with an automated one, which improves its efficiency and speed. As a result, the system is able to process tasks much quicker and with a shorter response time.

**Accuracy** – To put it differently, the application offers users a prompt and precise response, providing them with accurate information about bill calculations, customer details, and other relevant system information whenever needed.

**User Friendly** – In simpler terms, the Super Store Sales Management System comes with a user-friendly interface that enables users to work on it easily. The application combines accuracy with an attractive interface, thereby enhancing the interaction of the present manual system. This makes the system more intuitive, faster, and user-friendly.

**Maintain Cost** – The application is a standalone system that does not require any maintenance cost, resulting in reduced costs.

**HttpOnly Cookie** is a tag that is added to a browser cookie that forbids data access by client side scripts. It offers a fence that stops anyone other than the server from accessing the specialised cookie. The possibility of client-side scripts accessing a protected cookie is reduced by using the HttpOnly tag when creating cookies, making these cookies more safe. Say a browser notices a cookie with the HttpOnly flag in it. The browser will return an empty string if client-side code tries to read the cookie. This lessens the likelihood that malicious code— typically cross-site scripting (XSS)—will convey the information to an attacker's website.

JSON Web Tokens (JWT) is an open standard that facilitates the secure transfer of data between parties as a JSON object. It is a condensed and self-contained method that can be pronounced as "jot". Although all JWTs are tokens, not all tokens are JWTs since JWT is a specific standard. A JWT can be delivered through a URL, a POST parameter, or an HTTP header, and is quickly transferred due to its small size. By providing all the necessary data about an entity, a JWT eliminates the need for multiple database queries. Moreover, unlike other tokens, a JWT does not require the token recipient to verify it by calling a server.

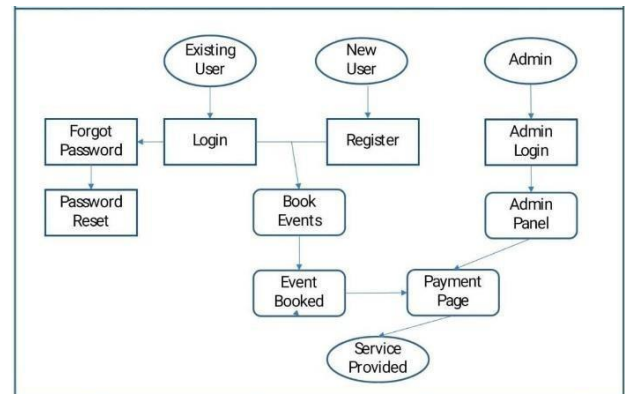
**More condensed** - To put it differently, a JWT is smaller in size compared to a SAML token when encoded because JSON has a more concise syntax than XML. This makes JWT a better choice for passing data in HTML and HTTP environments.

**More Secure** : JWT signing algorithms are used to ensure the integrity and authenticity of the data contained within a JWT. The two main types of JWT signing algorithms are symmetric and asymmetric.

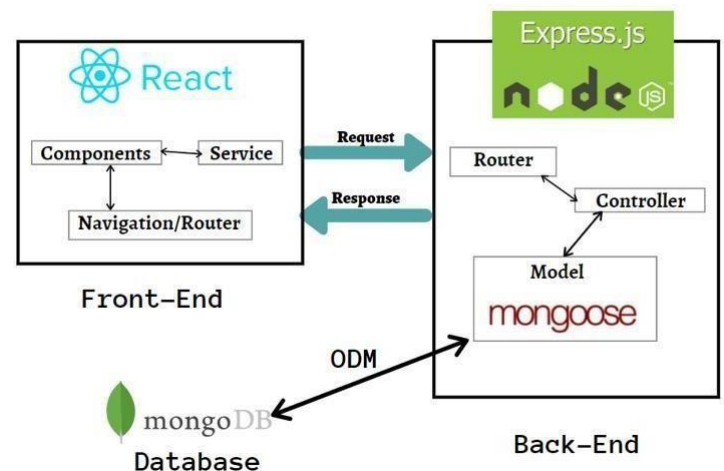
Symmetric algorithms, such as HMAC, use a shared secret key to both sign and verify the JWT. The same secret key is used by both the issuer and the verifier to ensure that the JWT has not been tampered with during transmission. The HMAC algorithm is fast, efficient, and easy to implement, making it a popular choice for many applications.

**Bcrypt** is hashing hash passwords with the aid of the Bcrypt library. It makes use of a Blowfish-based password-

hashing algorithm. It can be utilised in cypher suites and encryption products since the Blowfish cypher, a symmetric block cypher, has the finest encryption rate in the business. Compared to other hashing algorithms like MD5, SHA1, SHA2, and SHA3, Bcrypt provides a number of advantages. They can all quickly conduct data hashing on a big volume of data. Let's say an attacker has a powerful machine that can quickly test 700– 900 million passwords. In a matter of seconds, your password, which includes both alphanumeric and special character values, will be broken

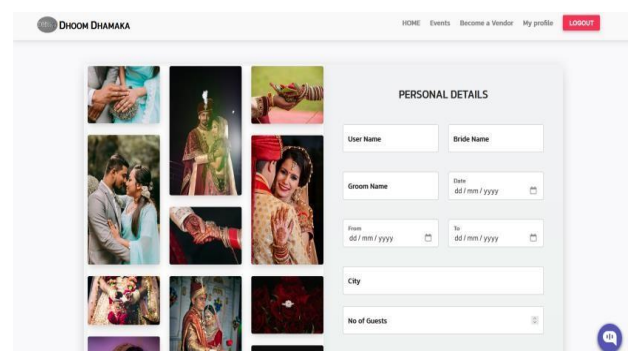


System Workflow.



System Architecture.

### III. RESULT AND DISCUSSION

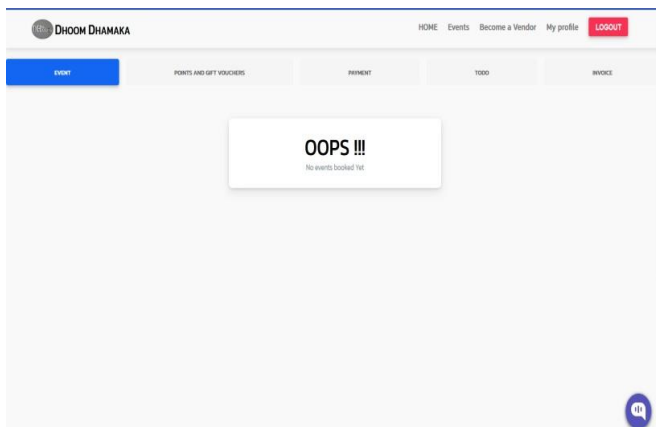


## IV. CONCLUSION

The article proposed Multi-facility function coordinator using React JS. The main aim of the proposed expert system is to gather all the service in one place to book an event organizer for our customers/clients. The authentication process uses user behavior information based on profiling to create a login method for user authentication. In this Multi-facility function coordinator using React JS admin system has the easy access to mange their user users and orders based on the date by this they can easily mange their system after completing the order the admin easy access to delete their orders. By this Multi-facility function coordinator using React JS we can reduce the time for booking each service.This makes your work lot easier.

## V. REFERENCES

- [1] ER. L. Sebesta, "An Introduction to React: The JavaScript Library for Building User Interfaces," IEEE Software, vol. 35, no. 3, pp. 101-103, May-Jun. 2018.
- [2] S. K. Madan and R. K. Madan, "An Overview of React Library and Its Application in Web Development," International Journal of Advanced Computer Science and Applications, vol. 10, no. 7, pp. 427-432, Jul. 2019.
- [3] A. Chen, "React Native: A Cross-Platform Framework for Mobile App Development," IEEE Internet Computing, vol. 23, no. 6, pp. 92-97, Nov-Dec. 2019.
- [4] S. J. Zaka and S. A. Memon, "React: A Comparative Study of Frontend Web Frameworks," International Journal of Computer Applications, vol. 179, no. 34, pp. 27-31, Feb. 2021.
- [5] N. S. M. Z. Shah and F. A. Memon, "ReactJS: A Comprehensive Survey on Its Architecture and Features," International Journal of Scientific & Engineering Research, vol. 11, no. 10, pp. 2058-2062, Oct. 2020.
- [6] N. V. Kumar and D. Raj, "A Study on React and Its Components," International Journal of Advanced Research in Computer Science, vol. 11, no. 5, pp. 20-24, Sep.-Oct. 2020.
- [7] R. Jindal and S. Jindal, "A Comparative Study of Angular and React Frameworks for Frontend Web Development," International Journal of Advanced Research in Computer Science and Software Engineering, vol. 10, no. 4, pp. 561-568, Apr. 2020.
- [8] H. D. R. Kusuma and W. Wijaya, "React Framework and Its Implementation in Web Application Development," Journal of Physics: Conference Series, vol. 1655, no. 1, pp. 012030, Mar. 2020.
- [9] T. H. Khan and M. Hussain, "Performance Evaluation of React and AngularJS Frameworks for Web Development," International Journal of Computer Science and Network Security, vol. 20, no. 1, pp. 167-174, Jan. 2020.
- [10] S. B. Tamboli and S. S. Chavan, "A Review on React and Its Applications in Web Development," International Journal of Innovative Technology and Exploring Engineering, vol. 9, no. 2, pp. 1752-1756, Nov. 2019.



From the the above attached images shows that what steps to be followed by the user to book an event based on their needs. Once the process for entering the detail is completed the pop up will appear on the user screen and the user should be redirected to the home page. once the process is done the overview of the selected event is displayed in card in event tab under profile page. Then the confirmation process will occurs on the client side within 7 - 10 hours. once the confirmation is done the payment details is updated by the admin. By using this web application it makes user to feels comfortable to book all kind of events. This process reduces the time for booking an event from home by just opening this web application.

- [11] A. B. Smith, "Design and Implementation of an Online Booking System for Hotels," *Journal of Hospitality and Tourism Technology*, vol. 11, no. 2, pp. 158-169, May 2020.
- [12] P. Liang and K. Li, "Design and Implementation of an Online Reservation System for Car Rental Companies," *Journal of Information Science and Engineering*, vol. 35, no. 5, pp. 1149-1166, Sep. 2019.
- [13] A. R. Khaliq and A. Asghar, "Implementation of an Online Bus Ticket Reservation System," *Journal of Computer Science and Technology*, vol. 19, no. 4, pp. 723-732, Jul. 2019.
- [14] R. T. Chen and M. C. Chen, "Design and Implementation of an Online Flight Booking System," *International Journal of Electronics and Electrical Engineering*, vol. 7, no. 3, pp. 163-170, May 2019.
- [15] L. J. Zhang and Y. J. Li, "Development of an Online Hotel Booking System," *International Journal of Digital Content Technology and its Applications*, vol. 13, no. 1, pp. 201-207, Jan. 2019.
- [16] R. K. Sah and P. Pandey, "Design and Implementation of an Online Movie Ticket Booking System," *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 9, no. 4, pp. 31-34, Apr. 2019.
- [17] S. R. Thapa and S. S. Shrestha, "Design and Implementation of an Online Restaurant Reservation System," *International Journal of Engineering Research and Applications*, vol. 8, no. 5, pp. 75-80, May 2018.
- [18] K. M. Kaur and P. Singh, "Design and Implementation of an Online Train Ticket Reservation System," *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, vol. 4, no. 1, pp. 82-86, Jan.-Feb. 2018.
- [19] S. Goyal and S. Rani, "Design and Implementation of an Online Ticket Booking System for Amusement Parks," *International Journal of Computer Science and Mobile Computing*, vol. 6, no. 3, pp. 86-94, Mar. 2017.
- [20] M. N. M. Yusof and N. M. Jusoh, "Design and Development of an Online Booking System for a Sport Complex," *International Journal of Advances in Science, Engineering and Technology*, vol. 5, no. 2, pp. 40-44, Feb. 2017.