SYMPOSIUM EVENT MANAGEMENT SYSTEM

A PROJECT REPORT

Submitted by

LIPIKA S

(Reg. No.: 24MCR060)

KEERTHANA R

(Reg. No: 24MCR056)

in partial fulfilment of the requirements

for the award of the degree

of

MASTER OF COMPUTER APPLICATIONS

DEPARTMENT OF COMPUTER APPLICATIONS

KONGU ENGINEERING COLLEGE

(Autonomous)

PERUNDURAI, ERODE – 638 060

DECEMBER -2024

DEPARTMENT OF COMPUTER APPLICATIONS

KONGU ENGINEERING COLLEGE

(Autonomous)

PERUNDURAI, ERODE – 638 060

DECEMBER-2024

BONAFIDE CERTIFICATE

This is to certify that the project report entitled **"SYMPOSIUM EVENT MANAGEMENT SYSTEM"** is the bonafide record of project work done by **LIPIKA S (24MCR060)** and **KEERTHANA R (24MCR056)** in partial fulfilment of the requirements for the award of the Degree of Master of Computer Applications of Anna University, Chennai during the year 2024-2025.

SUPERVISOR	**HEAD OF THE DEPARTMENT**
Date:	**(Signature with seal)**

Submitted for the end semester viva-voce examination held on ________________

INTERNAL EXAMINER	**EXTERNAL EXAMINER**

DECLARATION

I affirm that the project report entitled **“SYMPOSIUM EVENT MANAGEMENT SYSTEM”** being submitted in partial fulfilment of the requirements for the award of Master of Computer Applications is the original work carried out by us. It has not formed the part of any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

DATE: **LIPIKA S**

(Reg. No: 24MCR060)

KEERTHANA R

(Reg. No: 24MCR056)

I certify that the declaration made by the above candidates is true to the best of my knowledge.

Date: **Name and Signature of the Supervisor**

ABSTRACT

The symposium event management system for a college is designed to cater to both admins and students in a college. The system allows the admin to log in and access an interface where they can add and manage different events, such as paper presentations, workshops, and other campus activities. The events are then stored in a database, ensuring that all information is easily accessible. Students, upon logging in, can view a list of these events. They can also use a search bar to find specific events based on their interests. Each event entry includes detailed information, and students can click a link that redirects them to the college's official website for more details.

The frontend of the system is built using modern technologies like React.js and Vite, providing a fast and responsive user experience. The backend is powered by Node.js, ensuring smooth data handling and server management. MongoDB is used as the database to store important data, such as admin and student login credentials and event details. The system ensures the security of user data and provides efficient management of events, making it easier for the admin to add, update, and remove events.

The use of this technology stack ensures that the platform is scalable, secure, and easy to maintain. The user-friendly interface allows students to stay informed about upcoming events and engage with the campus community. The admin interface, on the other hand, offers a seamless experience for event management. This system helps simplify event management tasks, providing an organized and efficient platform for both students and administrators.

ACKNOWLEDGEMENT

We respect and thank our **THIRU.A.K.ILANGO, B.Com., M.B.A., LLB.,** and our principal **Dr.V.BALUSAMY B.E(Hons)., M.Tech., Phd.** Kongu Engineerning College, Perundurai for providing us with the facilities offered.

We convey our gratitude and heartfelt thanks to our Head of the Department **Dr.A.TAMILARASI MSc., M.Phil., PhD** Department of Computer Applications, Kongu Engineering College, for her perfect guidance and support that made this work to be completed successfully.

We also like to express our gratitude and sincere thanks to our project coordinator **Dr.T.KAVITHA MCA., M.Phil., PhD** Assistant Professor (Sr.G), Department of Computer Applications, Kongu Engineering college who have motivated us in all aspects for completing the project in scheduled time.

We would like to express our gratitude and sincere thanks to our project guide **Dr.M.PYINGKODI MCA.,M.Phil,PhD** Associate Professor, Department of Computer Applications, Kongu Engineering College for giving her valuable guidance and suggestions which helped us in the successful completion of the project.

We owe a great deal of gratitude to our parents for helping overwhelm in all proceedings. We bow our heart with heartfelt thanks to all those who thought us their warm services to succeed and achieve our work.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVATIONS

UI	User Interface
HTML	Hyper Text Markup Language
DFD	Data Flow Diagram
DOM	Document Object Model
NPM	Node Package Manager
API	Application Programming Interface
XML	Extensible Markup Language
DB	Data Base

CHAPTER 1
INTRODUCTION

1.1 ABOUT THE PROJECT

The **Symposium Event Management System for a college** is a web-based platform designed to streamline the management of college events, built using modern technologies such as React.js, Vite, Node.js, and MongoDB. The system allows for two primary user roles Admin and Student**.**

Admins have the ability to securely log in and add, update, or delete events. These events can include activities like paper presentations, workshops, and seminars, making it easier to manage the events for the college community. After logging in, the admin has a user-friendly dashboard where they can input event details such as the event name, date, description, and location. All event data, along with admin login credentials, is securely stored in the MongoDB database using Mongoose for efficient data handling.

Students are able to register as new users, log in to the system, and view the events posted by the admin. Once logged in, students can access a list of upcoming events, which can be filtered or searched for by keywords using a dedicated search bar. This feature makes it easy for students to find specific events they are interested in attending. The system ensures that students can stay up-to-date with event information and never miss an opportunity. Additionally, students can click on a provided link to access the official college website for more details or resources.

The frontend is built using React.js and Vite react to provide a responsive, dynamic, and high performance user interface. React.js ensures that the application is fast and interactive, while Vite streamlines the development process for faster builds and live-reloading during development. On the backend, Node.js handles the server-side logic and facilitates communication with the database.

The MongoDB database stores essential data, such as user login details (both admin and student), as well as event-related information. Mongoose is used to model and interact with the MongoDB database in a structured manner.

The application's overall architecture ensures that the system is secure, scalable, and efficient. The use of modern technologies like React.js, Node.js, and MongoDB ensures that both the frontend and backend can handle large volumes of data and users, providing a smooth experience for all participants. This platform serves as an all-in-one solution for managing events within a college, allowing both admins and students to access, share, and interact with event information easily. By leveraging these technologies, the system supports enhanced user engagement, simplifies event management tasks, and provides a seamless and enjoyable experience for both students and administrators.

CHAPTER 2

SYSTEM ANALYSIS

2.1 EXISTING SYSTEM

An existing symposium event management system for a college is a platform where both admins and students can manage and access event information. Admins can log in to add and manage events like workshops, paper presentations, and cultural activities. They input details such as the event name, description, date. This information is stored in a central database. Students can register, log in, and view upcoming events. They can search for events by keywords and get more details. Some systems also provide a link to the college website for additional information. Using an automated system would make everything easier, faster, and more accurate.

2.1.1 DRAWBACKS OF EXISTING SYSTEM

- Students may only be able to view events and sign up, but they might not have the ability to interact with other students or participate in discussions related to events.
- Many systems do not send reminders or notifications to students about upcoming events, which could lead to low attendance or missed opportunities.
- Admins may need to manually update event details, which can be time-consuming and prone to errors.
- Some systems might not integrate with other college platforms like student portals or email systems, making it harder to share information.
- If not properly secured, user data like login information and event details can be vulnerable to unauthorized access or data breaches.

2.2 PROPOSED SYSTEM

The proposed symposium event management system for a college will allow students to register, log in, and view events such as paper presentations, workshops, and seminars, with a search bar for easy navigation. Admins can log in securely to add, update, or delete events, with all event and user data stored in a central database. The system will use React.js and Vite for the frontend, Node.js for the backend, and MongoDB for secure data storage. Students will also have access to links to the college website. Automated notifications and reminders will improve communication. The system will be mobile-optimized and feature strong security with data encryption and two-factor authentication.

2.2.1 ADVANTAGES OF PROPOSED SYSTEM

- Students can easily register, log in, view, and search events with a simple and responsive interface, improving engagement.
- Admins can quickly add, update, and manage events, reducing manual effort and improving accuracy.
- Automatic reminders and notifications ensure that students are always updated on upcoming events, leading to higher participation.
- The system is designed to work seamlessly on both desktop and mobile devices, ensuring accessibility anytime, anywhere.
- Strong security measures like data encryption and two-factor authentication protect sensitive user and event data from unauthorized access.
- By providing links to the college website and integrating with other college systems, the platform enhances communication and data sharing.

2.3 FEASABILITY STUDY

The feasibility study deals with all analysis that takes up in developing the project. Each structure has to be thought of in the developing of the project, as it has to serve the end user in friendly manner. A feasibility study is an essential phase in the project management process, as it helps determine whether a proposed system can be successfully developed, implemented, and maintained.

Three considerations involved in feasibility are

- Economic Feasibility
- Operational Feasibility
- Technical Feasibility

2.3.1 ECONOMIC FEASIBILITY

Economic feasible focuses on determining whether the proposed event management system because it uses cost-effective technologies like React.js, Node.js, and MongoDB, which are open-source and don't require expensive licenses. The initial development cost will be reasonable, and ongoing maintenance costs will be low since the system can be managed by a small IT team. The system is also scalable, meaning it can grow with the college without needing significant extra costs. Hosting the platform on cloud services allows the college to pay only for the resources used, making it flexible and affordable. Overall, the system will save time and effort, improve event participation, and provide long-term benefits at a reasonable cost.

2.3.2 OPERATIONAL FEASIBILITY

The operational feasibility of the proposed event management system is high because it is designed to be user-friendly and easy to use for both admins and students.

The system's interface will be simple and intuitive, so no extensive training will be required for users to get started. Admins can manage events with minimal effort, and students can easily register, view, and search for events. The system is scalable, meaning it can handle more users and events as the college grows. It will also be accessible on both desktop and mobile devices, ensuring that users can interact with the system from anywhere. With its automated notifications and simple management tools, the system will make event management more efficient, reducing administrative workload and improving overall operational effectiveness.

2.3.3 TECHNICAL FEASIBILITY

The "Symposium Event Management System for a college" is technically viable, employing a robust technology stack. Developed with React for the frontend and Node.js for server-side scripting, the application ensures a dynamic and responsive user interface. MongoDB serves as the database, providing efficient data management. Visual Studio Code facilitates seamless development, offering a suite of tools. The system incorporates business logic, secure data connections, and analysis through MongoDB. With its user-friendly interface, scalability, and data security measures, the "College Event Management" stands as a technologically sound solution for events.

CHAPTER 3

SYSTEM REQURIEMENTS

3.1 SOFTWARE REQURIEMENT SPECIFICATION

The Software Requirements Specification (SRS) defines the system's functionality required to meet the project's goals and user needs. These specifications help to design the system effectively and ensure it meets the customer's requirements while optimizing resources.

3.1.1 Hardware Requirements

The hardware requirements for the system are selected considering factors such as CPU speed, memory capacity, peripheral access speed, and storage requirements.

The following are the minimum hardware specifications for the event management system:

System: Processor Intel Core i3
RAM: 2 GB
Hard Disk: 512 GB
Keyboard: 108 Key Standard Keyboard Mouse:
Scroll Mouse

These hardware specifications will ensure smooth performance during both the development and usage phases of the application. The hardware will be capable of supporting Visual Studio Code for development, as well as handling MangoDB for data storage and user management.

3.1.2 Software Requirements

The software for the event management project is selected with considerations on ease of use, flexibility, and scalability. The system incorporates both front-end and back- end technologies, along with a database for data management.

Operating System	: Windows 11 and above
Frontend	: ReactJs
Backend	: Node.js
Database	: Mango DB
Tools	: visual studio code

This combination of technologies will ensure the system is user-friendly, scalable, and capable of handling event management tasks efficiently. Visual Studio Code will be used for front-end development, while Firebase will handle data storage and management for seamless event handling.

3.1.2.1 Front End

ReactJS is a declarative, efficient, and flexible JavaScript library for building reusable UI components. It is an open-source, component-based front end library which is responsible only for the view layer of the application. The primary goal of ReactJS is to create User Interfaces (UI) that accelerate apps. It makes use of virtual DOM (JavaScript object), which enhances the app's performance. Faster than the standard DOM is the JavaScript virtual DOM. ReactJS integrates with different frameworks and can be used on the client and server sides. It makes use of component and data patterns to make larger apps more readable and maintainable.

A ReactJS application is made up of multiple components, each component responsible for outputting a small, reusable piece of HTML code.

The components are the heart of all React applications. These Components can be nested with other components to allow complex applications to be built of simple building blocks. ReactJS uses virtual DOM based mechanism to fill data in HTML DOM. The virtual DOM works fast as it only changes individual DOM elements instead of reloading complete DOM every time. NodeJS and NPM are the platforms need to develop any ReactJS application. React-Bootstrap replaces the Bootstrap JavaScript. Each component has been built from scratch as a true React component, without unneeded dependencies like jQuery. The Bootstrap JavaScript is replaced by React-Bootstrap. Each component was created from the ground up as a genuine React component, free of other dependencies like jQuery. React-Bootstrap is one of the first React libraries, and because it has developed and matured alongside React, it is a great option for your UI foundation.

ReactJS primary goal is to create User Interfaces (UI) that increase the speed of apps. It makes use of virtual DOM (JavaScript object), which enhances the app's performance. Faster than the standard DOM is the JavaScript virtual DOM. ReactJS integrates with different frameworks and can be used on the client and server sides. It makes use of component and data patterns to make larger apps more readable and maintainable.

3.1.2.2 Features of React JS

- There is virtual DOM, which is basically a simplified version of the real DOM. Therefore, there is one object in React Virtual DOM for every object that is present in the real DOM. Although it is identical, it is unable to alter the document's design in any way. Virtual DOM manipulation is quick compared to DOM manipulation since no images are generated on the screen.
- One of React fundamental building blocks is the component. Or, to put it another way, every application you create with React will be built up of what are known as components. Using components makes creating user interfaces considerably simpler. You can see a user interface (UI) divided into numerous separate parts, or components, and work on them independently before merging them all into a parent component to create your final UI.

- React.js use JSX, which is faster compared to normal JavaScript and HTML. Virtual DOM is a less time taking procedure to update web pages content.

3.1.2.3 Back End

Node JS

Node.js is an open-source, cross-platform runtime environment for JavaScript code execution outside of a browser that is based on Chrome's V8 JavaScript engine. You must keep in mind that NodeJS is not a programming language or a framework. It offers a cross-platform runtime environment for event-driven, non-blocking (asynchronous) I/O and extremely scalable server-side JavaScript applications. The majority of folks are perplexed and realise it's a programming language or framework. Building back-end services like APIs, web applications, and mobile applications frequently uses Node.js. Large corporations like ,Uber, Netflix, Walmart, etc. use it in their manufacturing.

3.1.2.4 Features of Node.js

- Node.js is built on Google Chrome's V8 JavaScript Engine, so its library is very fast in code execution.
- Node.js follows a single threaded model with event looping.
- Node.js is highly scalable because event mechanism helps the server to respond in a non-blocking way.
- The processing time required to upload audio and video files is reduced overall thanks to Node.js Applications using Node.js never buffer any data.

These programmes merely output the data in sections.

- Node.js has an open source community which has produced many excellent modules to add additional capabilities to Node.js applications.
- Node.js is released under the MIT license.

3.1.2.5 MONGO DB

MongoDB serves as a dynamic and versatile database solution for my project, offering a robust platform to handle unstructured or semi-structured data efficiently. Departing from the constraints of traditional relational databases, MongoDB adopts a document-oriented data model. In this paradigm, data is stored in BSON format, a binary representation of JSON, enabling flexibility in representing complex relationships within a single document. This approach proves invaluable for managing the evolving and diverse data structures inherent in modern applications. Being an open-source and cross-platform database system, MongoDB a lignsseamlessly with my project's requirements, providing flexibility in development and deployment across various operating systems. Its scalability is a key asset, allowing my application to grow from small-scale projects to large-scale enterprise applications effortlessly. MongoDB's expressive query language empowers developers to craft intricate queries for data retrieval, filtering, and sorting with ease.

- The database excels in atomic operations at the document level, ensuring data consistency. The aggregation framework facilitates sophisticated data transformations, and the support for indexing optimizes query performance.
- MongoDB's horizontal scalability, achieved through sharding, accommodates expanding datasets and high-traffic scenarios. Security features, including authentication, access control, encryption, and auditing, enhance the protection of sensitive data, aligning with project requirements for robust data privacy and compliance. The inclusion of change streams enables real-time monitoring of database updates, a crucial feature for building responsive and reactive applications.

3.1.2.6 Features of MongoDB

- MongoDB supports a dynamic and flexible schema, allowing for diverse data structures within a collection.
- MongoDB offers a powerful query language for easy filtering, sorting, and projection of data.

- MongoDB's extensive documentation serves as a valuable resource for developers, offering guidance and examples.
- Supports geospatial indexing for efficient storage and retrieval of location-based data.
- Provides an aggregation framework for advanced data transformation and manipulation.

CHAPTER 4

SYSTEM DESIGN

4.1 MODULE DESCRIPTION

Admin Module

Student Module

Register

Login

4.1.1 Register

This is the most basic and the vital module where the new users get their chance to create their account in this web application. Proper validations will be provided to keep only authenticated users i.e. those users who will provide correct information. All the data supplied by the user will be stored in database and it will be used for further validation and authentication. During registration, user has to give login and password of their choice. Login names and passwords will be stored in the database so that the user can directly login without registering again and again.

4.1.2 Login

The login module for the Event Management for college serves as the gateway for both students and administrators, each with distinct roles and functionalities.i.e. user id and password and then only he can be logged on. The user id and password given by the user are checked from the data stored in the database if not the page is forwarded to register module.

4.1.3 Admin Module

The Admin Module is like the control center for Event management for college. It's a place where easily see all the upcoming events and important details. These details of each event's (Read), make changes to existing events (Update), add new events (Insert), and even remove events if needed (Delete). This module makes it super easy for event management for college to stay organized.

4.1.4 Student Module

The Student Module is where students can see or view the upcoming events and colleges and which department it is, and the students can search their events in the search bar it will list down the upcoming events in the student home page, and there will be a link the students can clicking by the link they can view the college websites.

4.2 DATAFLOW DIAGRAM

The Data Flow Diagram provides information about the inputs and outputs of each entity and process itself.

LEVEL 0

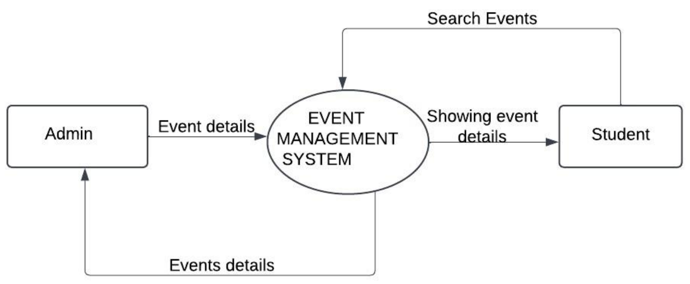


Figure 4.3 Dataflow Diagram Level 0

In Figure 4.3 Admin and Students can Login to the application and make use of their respective process in the system. In this Admin can able to accept or deny there request events.

LEVEL 1

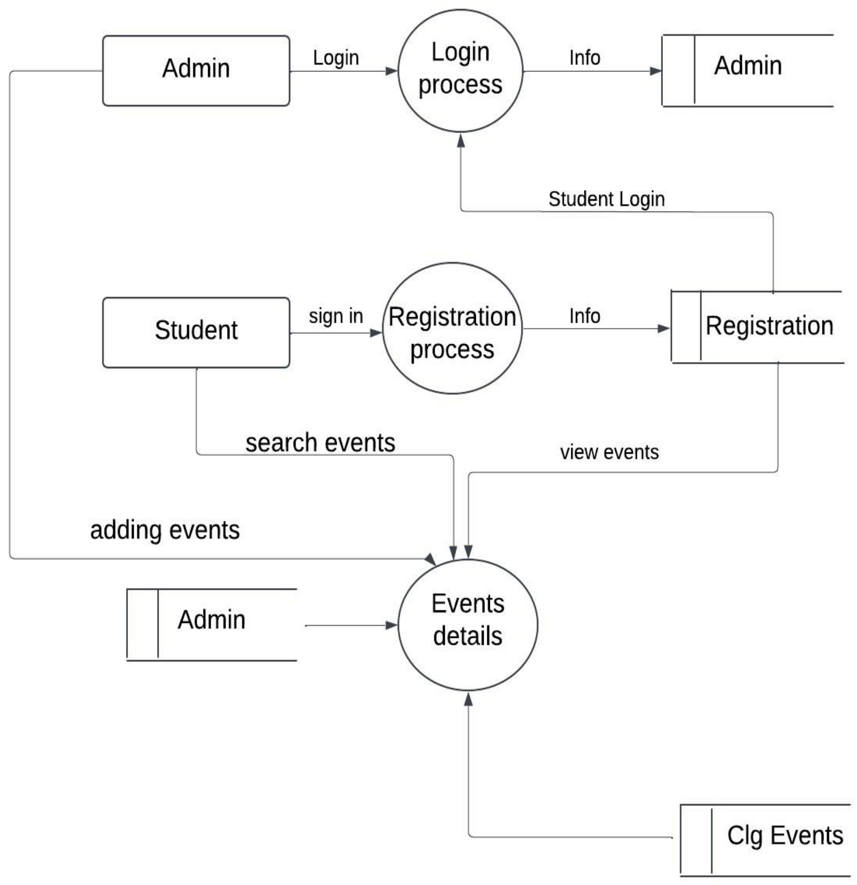


Figure 4.4 Dataflow Diagram Level 1

In Figure 4.4 Student should register the application and after register student login fill the details. After login the students can view the upcoming events and can search the events. The events details are stored in database and admin can add the events and delete the events.

4.5 CLASSBASE DAIGARM

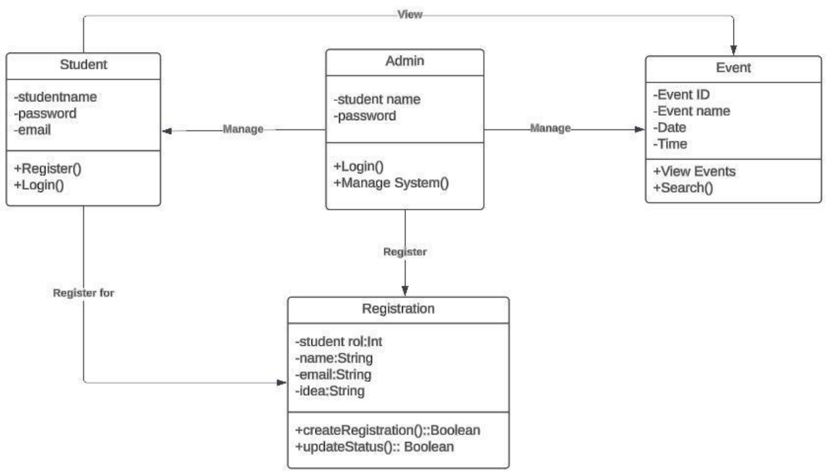


Figure 4.5 ClassBase Diagram

In Figure 4.5 shows the class diagram where student, admin, registration are the class name and each one have fields respectively. It also contains the functions or methods that are going to perform.

4.6 DATABASE DESIGN

Database design is the organization of data according to a database model. The designer determines what data must be stored and how the data elements interrelate. With this information. Database management system manages the data accordingly. The term database design can be used to describe many different part of the design of an overall database system. Principally, and most correctly, it can be thought of as the logical design of the base data structure used to store the data.

In an object database the entities and relationships map directly to object classes and named relationships. However, the term database design could also be used to apply to the overall process of designing, the base data structure, but also the forms and queries used as part of the overall database application within the database management system.

4.10 INPUT DESIGN

Input design is the process of converting user-originated inputs to a computer understandable format. Input design is one of the most expensive phase of the operation of computerized system and is often the major problem of a system. A large number of problems with a system can usually be tracked back to fault input design and method.

Every moment of input design should be analyzed and designed with almost care. The decisions made during the input design are the project gives the low time consumption to make sensitive application made simple. Thus, the developed system is well within the budget. This was achieved because most of the technologies used are freely available. In the project, the forms are designed with easy-to-use option. The coding is being done such that proper validation are made to get the prefect input. No error inputs are accepted.

Admin Login

This is a admin login form, the admin can login with their registered name and password to access the application. After completing the registration, the admin should give their correct register details else the user cannot able to login.

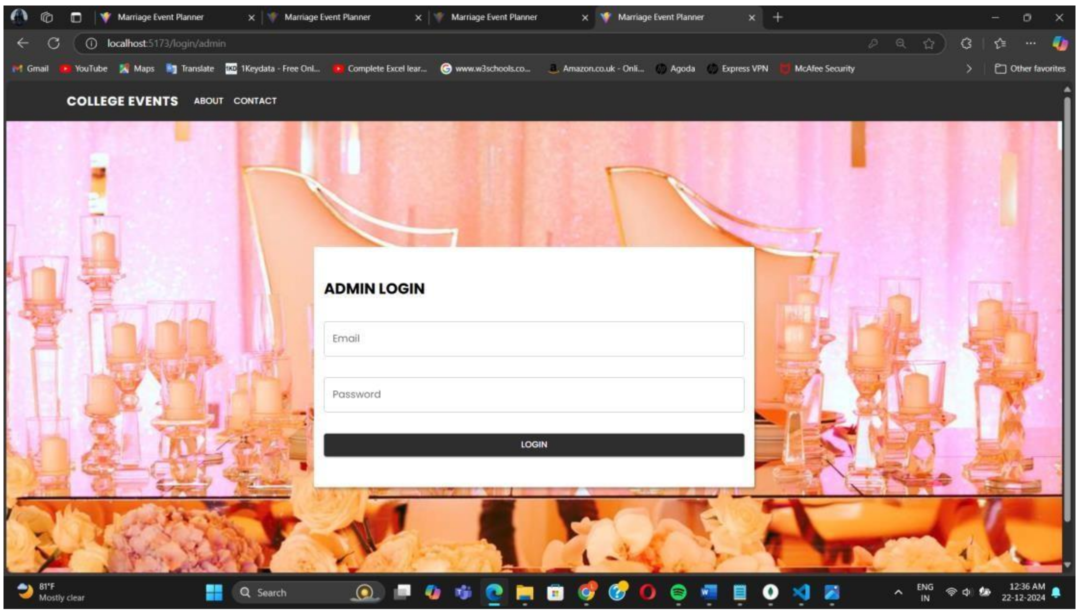
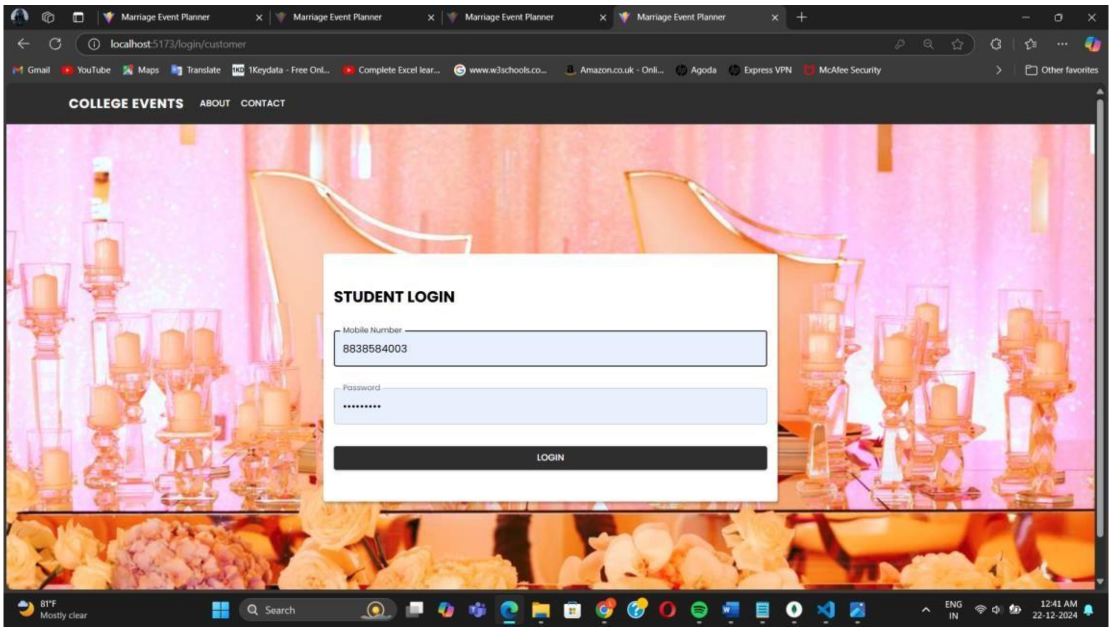


Figure 4.11 Admin Login page

The Figure 4.11 Admin login form shows the Login form has Email and password. The username should be in proper email format. If the fields are not filled, form will not be submitted. Password must contain at least 8 digit or characters. If the form is submitted successfully the page redirected towards Events page.

Customer login



STUDENT LOGIN

Mobile Number: 8838584003

Password: •••••••••

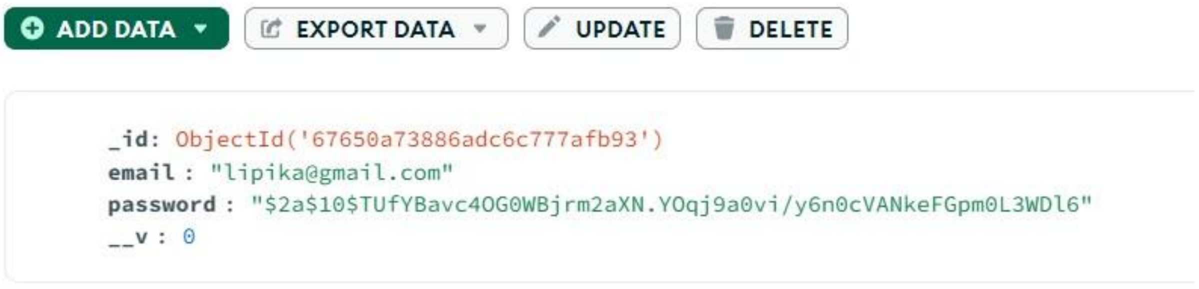
Figure 4.12 Student login page

The Figure 4.12 Student login page shows the Students has to register the form. After register Students were login. It will show the upcoming event in the next page.

4.13 OUTPUT DESIGN

Output design generally refers to the results and information that are generated by the system for many end-users; it should be understandable with the enhanced format. Computer output is most important direct source of information to the user. Output design deals with form design. Efficient output design should improve the interfacing with user. The term output applies to any information produced by an information system in terms of data displayed.

Previewing the output reports by the user is extremely important because the user Is the ultimate judge of the quality of the output and in turn, the success of the system. When designing output, system analysis accomplishes more things like, to determine what applications, website or documents are blocked or allowed. The output is designed in such way that is attractive, convenient and informative.



```
_id: ObjectId('67650a73886adc6c777afb93')
email : "lipika@gmail.com"
password : "$2a$10$TUfYBavc4OG0WBjrm2aXN.YOqj9a0vi/y6n0cVANkeFGpm0L3WDl6"
__v : 0
```

Figure 4.13 Storing in MongoDB

The Figure 4.13 shows the information of admin detail. Data are stored as documents with values mapped to fields. Collections, which are containers for the documents that may use to arrange data and create queries, store these documents.

CHAPTER 5

SYSTEM TESTING

5.1 TESTING

Software testing serves as the final assessment of specifications, designs, and coding and is a crucial component of software quality assurance. The system is tested throughout the testing phase utilizing diverse test data. The preparation of test data is essential to the system testing process. The system under study is tested after the test data preparation. Once the source code is complete, relevant data structures should be documented. The finished project must go through testing and validation, when errors are explicitly targeted and attempted to be found.

The project developer is always in charge of testing each of the program's separate units, or modules. Developers frequently also perform integration testing, which is the testing phase that results in the creation of the complete program structure.

This project has undergone the following testing procedures to ensure its correctness.

- Unit testing
- Integration Testing
- Validation Testing

5.1.1 Unit Testing

A testing strategy known as unit and integration testing has been used to check that the system behaves as expected. The testing strategy was based on the functionality and the requirements of the system. In unit checking out, have to check the applications making up the device. For this reason, Unit checking out once in a while referred to as a program checking out.

The software program in a device are the modules and workouts which might be assembled and included to carry out a selected function, Unit checking out the first at the modules independently of 1 another, to find mistakes. This enables, to stumble on mistakes in coding and common sense which might be contained with the module alone. The checking out became completed at some stage in programming level itself.

Test Case 1

Module	: Customer Login
Input	: Phone number and Password
Event	: Button click
Output	: Logged in successfully

Test 1

Module	: Customer Login
Phn number	: 9842777560
Password	: abcdef
Output	: Logged in successfully
Event	: Button click
Analysis	: Phone number and Password has been verified

5.1.2 Integration Testing

Integration testing is done to test itself if the individual modules work together as one single unit. In integration testing, the individual modules that are to be integrated are available for testing. Thus, the manual test data that used to test the interfaces replaced by that which in generated automatically from the various modules. It can be used for testing how the module would actually interact with the proposed system. The modules are integrated and tested to reveal the problem interfaces.

Test Case 1

Module : Admin Login

Input : Username and Password

Output : Redirect to home page

Test 1

Module : Admin Login

Username : abc@gmail.com

Password : 123456

Output : Redirected to approval page

Analysis : Username and Password has been verified

5.1.3 VALIDATION TESTING

Validation testing for a college event typically involves ensuring that the website or application functions correctly across various devices and browsers that the searching the events process works smoothly and that students can easily find and view the events.

This includes Email validation, Password validation, Admin validation, and Students validation. When these fields are filled correctly with the given constraints then only it navigates to other pages or modules.

Test case 1

Module : Students Registration

Input : Password

Output : Password must be at least 8 numbers or characters

Test 1

Module : Students Registration

Password : 12345678

Output : Registered successfully

Analysis : The given password is 6 numbers. So, it is validated.

CHAPTER 6

IMPLEMENTATION

6.1 CODE DESCRIPTION

Code description can be used to summarize code or to explain the programmer's intent. Good comments do not repeat the code or explain it. Comments are sometimes processed in various ways to generate documentation external to the source code itself by document generator or used for integration with systems and other kinds of external programming tools. have chosen react js as front end and node js as back end.

6.2 STANDARDIZATION OF THE CODING

Coding standards define a programming style. A coding standard does not usually concern itself with wrong or right in a more abstract sense. It is simply a set of rules and guidelines for the formatting of source code. The other common type of coding standard is the one used in or between development teams. Professional code performs a job in such a way that is easy to maintain and debug. All the coding standards are followed while creating this project. Coding standards become easier, the earlier you start. It is better to do a neat job than cleaning up after all is done. Every coder will have a unique pattern than he adheres to. Such a style might include the conventions he uses to name variables and functions and how he comments his work. When the said pattern and style is standardized, It pays off the effort well in the long.

6.3 ERROR HANDLING

Exception handling is a process or method used for handling the abnormal statements in the code and executing them. It also enables to handle the flow control of the code/program. For handling the code, various handlers are used that process the exception and execute the code. Mainly if-else block is used to handle errors using condition checking, if-else catch errors as a conditional statement. In many cases there are many corner cases which must be checking during an execution but "if-else" can only handle the defined conditions. In if-else, conditions are manually generated based on the task.

An error is a serious problem than an application doesn't usually get pass without incident. Errors cause an application to crash, and ideally send an error message offering some suggestion to resolve the problem and return to a normal operating state, There's no way to deal with errors "live" or in production the only solution is to detect them via error monitoring and bug tracking and dispatch a developer or two to sort out the code.

Exceptions, on the other hand are exceptional conditions an application should reasonably be accepted to handle. Programming language allow developers to include try...catch statements to handle exceptions and apply a sequence of logic to deal with the situation instead of crashing. And when an application encounters an exception that there's no workaround for, that's called an unhandled exception, which is the same thing as an error.

CHAPTER 7

CONCLUSION AND FUTURE ENHANCEMENTS

7.1 CONCLUSION

In conclusion, the proposed college event management system is an advanced and efficient platform aimed at improving the process of event management for both admins and students. It allows admins to securely log in, add, update, and delete events such as paper presentations, workshops, and seminars, ensuring all event details are well-organized and stored securely in a database. Students can register, log in, view upcoming events, and use a search bar to easily find specific events. The system also provides a link to the college website, giving users direct access to additional resources and official information.

This system is built using modern technologies such as React.js and Vite for a fast and responsive frontend, Node.js for a robust backend, and MongoDB for secure and scalable data storage. Its responsive design ensures a seamless user experience across both mobile and desktop devices, allowing access anytime, anywhere. Students can benefit from automated features like notifications and reminders, keeping them informed about upcoming events and improving overall participation.

For admins, the system simplifies event management through a streamlined interface, reducing manual effort and administrative workload. The platform's use of open-source technologies makes it cost-effective, requiring minimal investment for development and maintenance. It is also designed to be scalable, ensuring it can handle increasing numbers of users and events as the college grows.

7.2 FUTURE ENHANCEMENT

In the future, the event management system can be enhanced with additional features to make it even more useful and efficient. Features like a feedback system can be added so students can share their opinions about events, helping admins improve future events. Integration with social media platforms can allow students to share event details and updates easily. A live chat feature could enable real-time interaction between students and event organizers for quick communication.

The system can also support advanced analytics to help admins track event participation and success rates. Adding QR code-based check-ins for events can simplify attendance tracking. Furthermore, the system can be integrated with payment gateways for events that require registration fees. With time, implementing AI-based recommendations can suggest events to students based on their interests or past participation. These enhancements will make the system more versatile, engaging, and effective for both admins and students.

APPENDIX 1

SAMPLE CODING

Events

```
const { decode } = require("jsonwebtoken"); const
Event = require("../models/Event.model");
exports.createEvent = async (req, res) => {
  try        {
const {
     title,      date,      cust: { c_id: cust
},      opts,      link, // Add the link
parameter here
   } = req.body;
   // Create the event with the link field    const event = new
Event({ title, date, cust, opts, link });    await event.save();
res.status(201).json({ message: "Event created successfully" });
  } catch (error) {    res.status(500).json({ error:
error.message });
  } }; exports.getAllEvents = async (req,
res) => { try {   const decodedData =
decode(req.signedCookies.token);
```

```
    // Fetch events based on the customerId decoded from the token
const events = await Event.find(       decodedData?.customerId ? {
cust: decodedData.customerId } : {}
    ).populate({       path: "cust",       select: "-_id",       options: {
projection: { c_id: "$_id", mobileNo: "$mobileNo" } },
    });
    res.status(200).json(events); // `link` will be included here by default
  } catch (error) {     res.status(500).json({ error:
error.message });
  } }; exports.getById = async (req,
res) => {
  try {     const event = await
Event.findById(req.params.id).populate({       path: "cust",       select:
"-_id",       options: { projection: { c_id: "$_id", mobileNo:
"$mobileNo" } },
    });     if (!event) {       return res.status(404).json({ error:
"Event not found" }); } res.status(200).json({ data: event
}); // `link` is part of the event data
  } catch (error) {     console.log(error);
res.status(500).json({ error: error.message });
  } }; exports.deleteEvent = async (req,
res) => {
  try {     const { id } = req.query;     await
Event.findByIdAndDelete(id);     res.status(200).json({ message:
"Event deleted successfully" });
  } catch (error) {     res.status(500).json({ error:
error.message });
  }
};
```

App.jsx

```
import { Button, Stack, ThemeProvider } from "@mui/material"; import {
QueryClient, QueryClientProvider } from "@tanstack/react-query"; import
axios from "axios"; import { useEffect } from "react"; import {
  Navigate,
  Outlet,
  Route,
  BrowserRouter as Router,
  Routes,
} from "react-router-dom"; import "./App.css"; import cover from
"./assets/images/11582499_21034617.jpg"; import About from
"../src/contexts/components/About.jsx"; import AddEventForm from
"../src/contexts/components/AddEventForm.jsx"; import AdminLoginForm from
"../src/contexts/components/AdminLoginForm.jsx"; import AdminRegistrationForm from
"../src/contexts/components/AdminRegistration.jsx"; import AppHeader from
```

```
"../src/contexts/components/AppHeader.jsx"; import Contact from
"../src/contexts/components/Contact.jsx"; import CustomerLoginForm from
"../src/contexts/components/CustomerLoginForm.jsx"; import CustomerRegistrationForm
from
"../src/contexts/components/CustomerRegistration.jsx"; import EventList
from "../src/contexts/components/EventList.jsx"; import Home from
"../src/contexts/components/Home.jsx"; import AuthCtxProvider, {
useAuthCtx } from "./contexts/AuthContext"; import theme from
"./theme.jsx"; import SearchBar from
"./contexts/components/Search.jsx"; const queryClient = new
QueryClient(); function App() {   return (
  <Router>
    <ThemeProvider theme={theme}>
      <QueryClientProvider client={queryClient}>
        <AuthCtxProvider>
          <AppHeader />
          <AppRoutes />
        </AuthCtxProvider>
      </QueryClientProvider>
    </ThemeProvider>
  </Router>
 );
```

```
} const AppRoutes = () => {   const { isLoading, setAuth,
setIsLoading, auth } = useAuthCtx();   useEffect(() => {     axios
    .get("/api/auth", { withCredentials: true })
    .then((data)                =>                {
setAuth(data.data);
      // navigate("/events");
    })
    .catch((err) => {   //
    navigate("/login");
    console.log(err);
    })
    .finally(() => setIsLoading(false));
  }, []);
  if (isLoading) return <>Loading</>;
  return       (
<>
    <Routes>       <Route          path="/"          index          element={
auth && !isLoading ? <Navigate to="/events" replace /> : <Home />
        }
      />
      <Route path="/login" element={<NonAuthRoute />}>
```

```
        <Route
index
element={
            <>
<Stack
sx={{
                alignItems: "center",
                justifyContent: "center",
                flexDirection: "column",
                gap: "1rem",
                width: "100%",   minHeight: "100vh",    background:
              "url(" + cover + ") center center no-repeat",
                backgroundSize: "cover",
              }}
            >
              <Button
variant="contained"                    href="./admin"
sx={{                    width: "15rem",
background: "rgba(255, 255, 255, .5)",
padding: "1rem",
                }}
              >
                Admin Login
              </Button>
<Button
variant="contained"
href="./customer"
               sx={{   width: "15rem",   background:
              "rgba(255, 255, 255, .5)",   padding:
              "1rem",
               }}
              >
               Student Login
              </Button>
            </Stack>
          </>
        }
      />
      <Route path="admin" element={<AdminLoginForm />} />
      <Route path="customer" element={<CustomerLoginForm />} />
    </Route>
    <Route path="/register" element={<NonAuthRoute />}>
      <Route
index
element={
```

```
            <>              <Stack
sx={{                 alignItems:
"center",                  justifyContent:
"center",
                flexDirection: "column",
                gap: "1rem", width:
                "100%", minHeight:
                "100vh", background:
                "url(" + cover + ")
                center center no-repeat",
                backgroundSize:
                "cover",
              }}
            >
              <Button
variant="contained"                  href="./admin"
sx={{                   width: "15rem",
background: "rgba(255, 255, 255, .5)",
padding: "1rem",
                }}
              >
                Admin Registration
              </Button>
<Button
variant="contained"
href="./customer"
sx={{                    width:
"15rem",
                background:    "rgba(255,    255,    255,    .5)",
              padding: "1rem",
               }}
              >
               Student Registration
              </Button>
            </Stack>
          </>
        }
      />
      <Route path="admin" element={<AdminRegistrationForm />} />
      <Route path="customer" element={<CustomerRegistrationForm />} />
    </Route>
    <Route path="/events" element={<ProtectedRoute />}>
      <Route index element={<EventList />} />
      <Route path="new" element={<AddEventForm />} />
      <Route path=":id/edit" element={<AddEventForm />} />
```

```
        </Route>

        <Route path="/customer" element={<ProtectedRoute />}>

          {/* <Route index element={<EventList />} /> */}

          <Route path="new" element={<CustomerRegistrationForm />} />

        </Route>

        <Route path="/about" element={<About />} />

        <Route path="/contact" element={<Contact />} />

        <Route path="/search" element={<SearchBar />} />

      </Routes>

    </>

  );

}; const ProtectedRoute = () => {   const { auth, isLoading

} = useAuthCtx();   if (!auth && !isLoading) return

<Navigate to="/login" />;   return <Outlet />;

};

const  NonAuthRoute  =  ()  =>  {     const  {

auth, isLoading } = useAuthCtx();


  if (auth && !isLoading) return <Navigate to="/events" replace />;   return

<Outlet />;

};   export   default

App;
```

APPENDIX 2

SCREENSHOTS

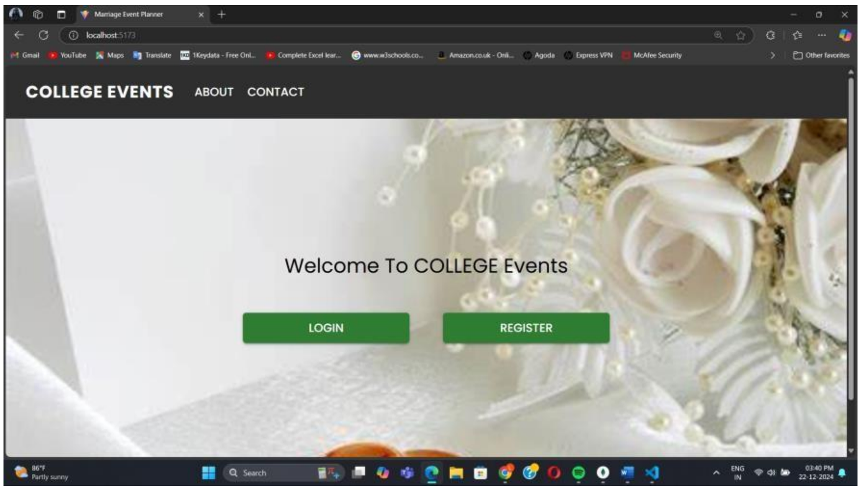


Figure A 2.1 Home Page

Figure A 2.1 Home Page shows the homepage of a College Event Management System. It features a welcoming banner with options for users to Login or Register to access the platform. The navigation bar at the top provides links to About and Contact sections for additional information.

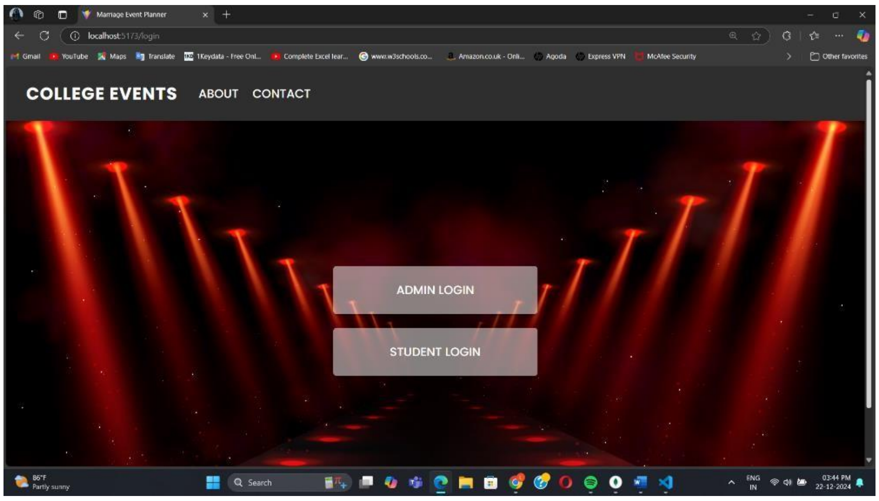
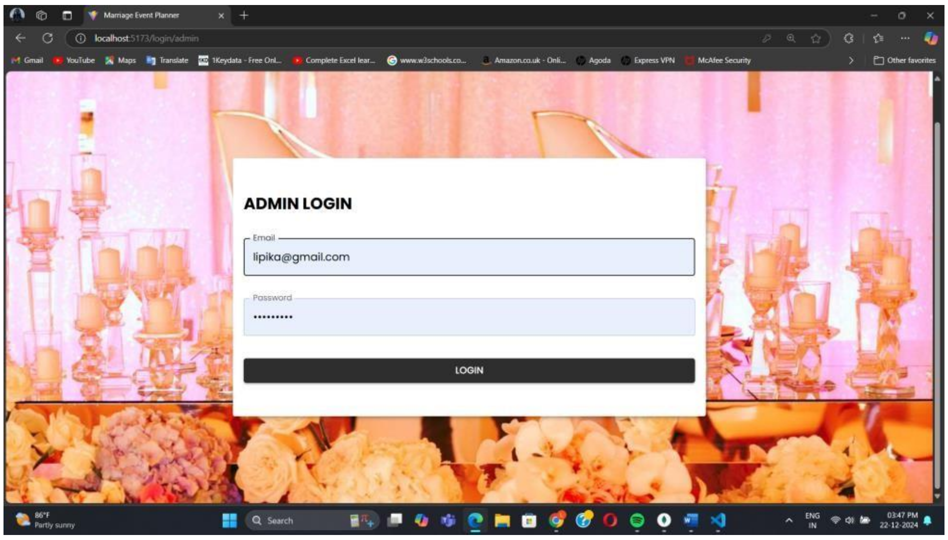


Figure A 2.2 LOGIN

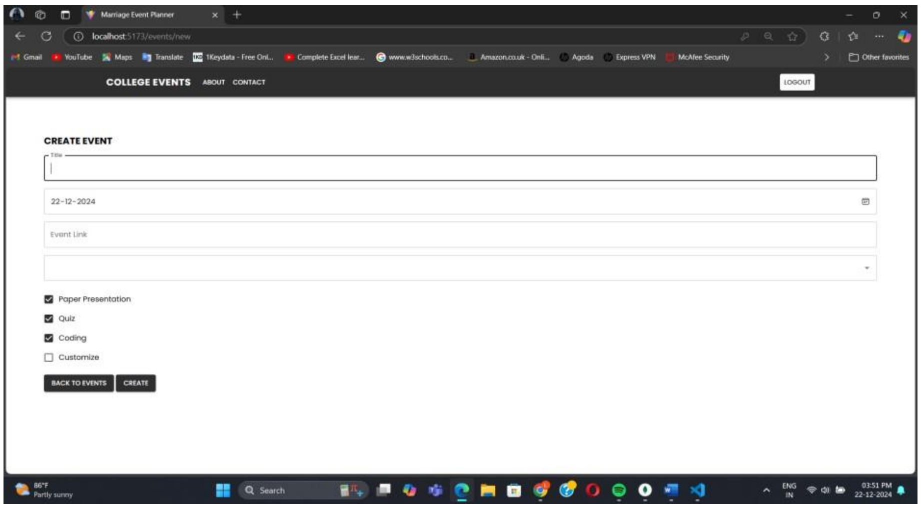
This page is the Login Interface of the College Event Management System. It allows users to choose between Admin Login and Student Login, providing separate access points for different roles. The visually appealing background enhances the user experience, aligning with the event theme.



Email: lipika@gmail.com
Password: •••••••••

Figure A 2.3 ADMIN LOGIN FORM

Figure A 2.3 admin login form shows the Admin Login Page where admins can securely enter their email and password. Once logged in, they can manage and update event details.

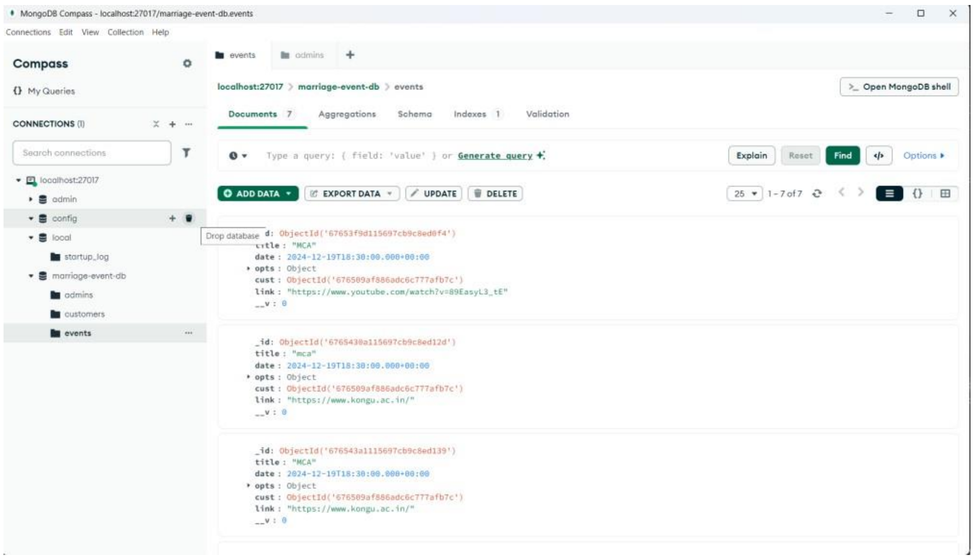


22-12-2024

- ✓ Paper Presentation
- ✓ Quiz
- ✓ Coding
- ☐ Customize

Figure A 2.4 ADMIN PAGE (MANAGE PAGE)

Figure A 2.4 admin page (manage page) shows the admin can add the events in this page by giving the add button.



```
  d: ObjectId('67653f9d115697cb9c8ed0f4')
  title : "MCA"
  date : 2024-12-19T18:30:00.000+00:00
▸ opts : Object
  cust : ObjectId('676509af886adc6c777afb7c')
  link : "https://www.youtube.com/watch?v=89EasyL3_tE"
  __v : 0

  _id: ObjectId('6765430a115697cb9c8ed12d')
  title : "mca"
  date : 2024-12-19T18:30:00.000+00:00
▸ opts : Object
  cust : ObjectId('676509af886adc6c777afb7c')
  link : "https://www.kongu.ac.in/"
  __v : 0

  _id: ObjectId('676543a1115697cb9c8ed139')
  title : "MCA"
  date : 2024-12-19T18:30:00.000+00:00
▸ opts : Object
  cust : ObjectId('676509af886adc6c777afb7c')
  link : "https://www.kongu.ac.in/"
  __v : 0
```

Figure A 2.5 MANGO DB

Figure A 2.5 mango db shows the events , students login data’s, and admin login data’s will be stored in the database.

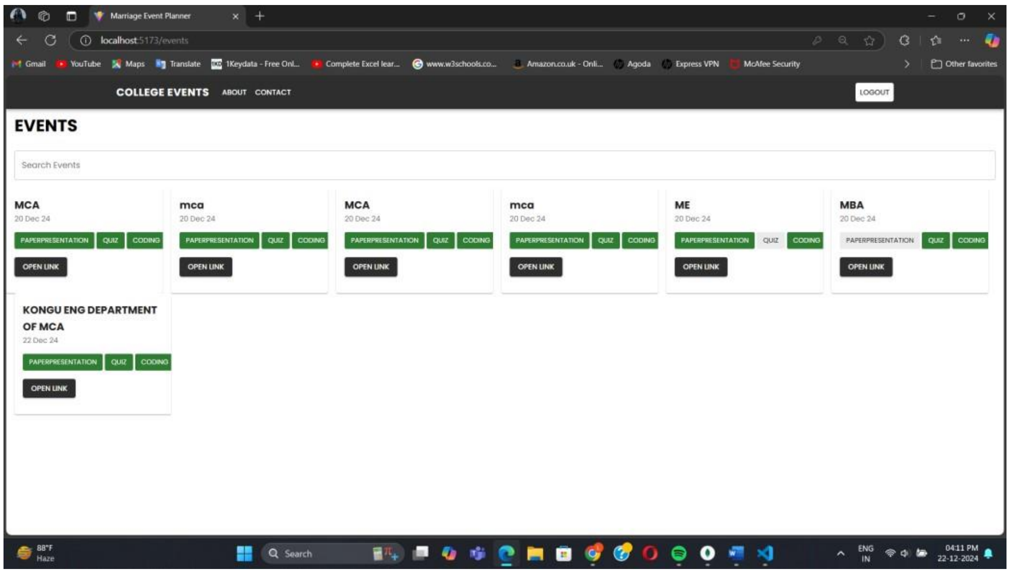


Figure A 2.6 STUDENT PAGE

Figure A 2.6 student page shows the students can view the upcoming events in this page and they can search the events in the search bar and there is the link , the students can visit the college website.

REFERENCES

[1] https://www.w3schools.com/django/

[2] https://www.geeksforgeeks.org/django-tutorial/

[3] Basics of Event Management ,by D. Ramkumar | 21 May 2024

[4] Basics of Event Management Perfect Paperback – 21 May 2024,by D. Ramkumar (Author)