INTERDISCIPLINARY PROJECT REPORT
at
Sathyabama Institute of Science and Technology
(Deemed to be University)

Submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering Degree in Computer Science and Engineering

By

BANDEPALLI SURYA ANJANI KUMAR
(Reg. No – 40110156)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited with Grade "A" by NAAC | 12 B Status
by UGC | Approved by AICTE
JEPPIAR NAGAR, RAJIV GANDHISALAI,
CHENNAI – 600119

APRIL 2023

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited with Grade "A" by NAAC | 12B Status by UGC | Approved by AICTE

www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **BANDEPALLI SURYA ANJANI KUMAR (40110156)** who carried out the project entitled "**EVENTS@SATHYABAMA**" under my supervision from FEB 2023 to APR 2023.

Internal Guide

Dr. M. D. Anto Praveena, M.E., Ph.D.,

Head of the Department

Dr. L. Lakshmanan, M.E., Ph. D.,

Submitted for Viva-voce Examination held on _____

Internal Examiner **External Examiner**

DECLARATION

I, **BANDEPALLI SURYA ANJANI KUMAR (Reg. No – 40110156),** hereby declare that the project report entitled "**EVENTS@SATHYABAMA"** was done by me under the guidance of **Dr. M. D. Anto Praveena**, **M.E., Ph.D.,** is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering Degree in Computer Science and Engineering.

DATE:

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. S. Vigneshwari, M.E., Ph.D.,** and **Dr. L. Lakshmanan, M.E., Ph.D.,** Heads of the Department of Computer Science and Engineering for providing me 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. M. D. Anto Praveena, M.E., Ph.D.,** for her valuable guidance, suggestions, and constant encouragement that paved way for the successful completion of my project work.

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

ABSTRACT

The Event Management System is a web-based application that simplifies the process of managing events within a college campus. It can be used to advertise events to the whole college or a specific branch easily.

Every event created has to pass through three levels of acceptance, namely Head of Department, Dean and Vice Chancellor, before it is shown to students. The system has powerful smart authentication that distinguishes between a student, VC, Teacher or HOD. The application allows teachers to create and post events, and students to view and apply to events. The system provides the event organizer the option to accept or deny a student's request to participate in an event.

The data model includes tables for events, users, departments, and certificates. The events table stores details such as the name, description, date, time, location, and maximum capacity. The users table stores information about users, including their name, email, and password. The departments table stores information about the different departments within the college. The certificates table stores information about the certificates that are issued to students upon completion of an event.

The system architecture is based on the Model-View-Controller (MVC) pattern, with the backend implemented using Django and the frontend built with NextJS, React and Tailwind CSS. The application is deployed using Docker and is hosted on a cloud platform like AWS or GCP.

The Event Management System provides a user-friendly interface for teachers to create and post events and for students to view and apply to events. It also ensures that events are properly approved before they are visible to students and provides a streamlined process for issuing certificates to students after the completion of an event.

CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

ABBREVIATION	*EXPANSION*
API	Application Programming Interface
CDN	Content Delivery Network
CSS	Cascading Style Sheets
DB	Database
DB	Database
DBMS	Database Management Systems
DRF	Django Rest Framework
ES6	ECMAScript 6
HTML	Hyper Text Markup Language
HTTP	Hypertext Transfer Protocol
JS	JavaScript
NPM	Node Package Manager
REST	Representational State Transfer
RWD	Responsive Website Development

CHAPTER 1

INTRODUCTION

Organizing events can be a daunting task, especially in large institutions like colleges. From planning to execution, it requires meticulous attention to detail, coordination, and effective communication. To simplify this process, the Event Management System is a web-based application that provides an efficient and streamlined solution for managing events within a college campus.

1.1 Why Events and Event Management Systems Matter?

Events are a critical part of student life, providing opportunities for personal growth, social interaction, and professional development. From orientation week to graduation ceremonies, colleges host various events throughout the academic year. The complexity of event management in colleges highlights the need for efficient and effective event management systems. Such systems simplify the event management process, enabling event planners to manage events seamlessly, while providing students with a platform to view and apply to events quickly and easily.

1.2 Introduction to Web Development

Web developers often work for clients who are trying to get their product or service onto the web. The work is typically very project focused and involves collaborating with a team that helps to coordinate the client's needs into the end product. The client could be an individual, a tech company, an organization, or a government.
The work could involve

- Front-end (Client-Side Development)
- Back-end (Server-Side Development) [or]
- Full-stack web development.

1.3 Introduction to Front-end Web Development

The front-end is the stuff you see on the website in your browser, including the

presentation of content and user interface elements like the navigation bar. Front-end developers use HTML, CSS, JavaScript, and their relevant frameworks to ensure that content is presented effectively and that users have an excellent experience.

1.4 Introduction to Back-end Web Development

The back-end refers to the guts of the application, which live on the server. The back-end stores and serves program data to ensure that the front end has what it needs. This process can become very complicated when a website has millions of users. Back-end developers use programming languages like JavaScript, Java, Python, and Ruby to work with data.

1.5 What is an API?

API is the acronym for Application Programming Interface, which is a software intermediary that allows two applications to talk to each other. Each time you use an app like Facebook, send an instant message, or check the weather on your phone, you're using an API. Modern APIs adhere to standards (typically HTTP and REST), that are developer-friendly, easily accessible and understood broadly.

1.6 What is a Database?

A database is an organized collection of structured information, or data, typically stored electronically in a computer system. A database is usually controlled by a database management system (DBMS). Together, the data and the DBMS, along with the applications that are associated with them, are referred to as a database system, often shortened to just database.

1.7 What is RWD?

Responsive web design (RWD) is about creating web pages that look good on all devices!
A responsive web design will automatically adjust for different screen sizes and viewports. Some frameworks of CSS like Bootstrap are responsive by default.

1.8 What is Deployment?

Deployment is the process of making software available to be used on a system by users and other programs.

Chapter 3 of this documentation provides thorough descriptions of the aforementioned subjects and the chosen approach and methods that were used for this project.

CHAPTER 2

AIM AND SCOPE OF THE PRESENT INVESTIGATION

2.1 PRESENT INVESTIGATION:

With the rise of online platforms and the increasing demand for seamless event planning and execution, colleges are increasingly turning to event management systems to simplify the process. However, the use of event management systems in college environments raises questions about their effectiveness, security, and user-friendliness.

2.2 AIM:

The primary goals of this project were to create a web application with the following features:

1. **Simplifying the Process of Managing Events**

The system will provide a streamlined process for creating and posting events, with a user-friendly interface for event organizers. The system will enable teachers to easily create and post events for students to attend, with all the necessary details, such as name, description, date, time, location, and maximum capacity. Students will also be able to view and apply to events quickly and efficiently, with all the necessary details provided in a clear and concise format.

2. **Ensuring Proper Approval of Events**

To ensure that events are properly approved before they are visible to students, the system will implement a three-level approval process. Events will be approved by the Head of Department (HOD), Dean and Vice Chancellor (VC) before they are made visible to students. This process will ensure that events are approved by the relevant authorities and that they meet the necessary standards.

3. **Smart Authentication and User Management**

The system will implement smart authentication for different user roles,

distinguishing between students, teachers, HOD, and VC.

4. Applications Management

The system will also provide the event organizer with the ability to accept or deny student requests, ensuring that only eligible students can attend events.

5. Reporting and Certificate Management

The system should provide reports on application status, and certificate generation. The system should distribute certificates for the students who attend the events.

2.3 SCOPE:

The scope of the event management system project is vast, as it provides numerous benefits to colleges and their students. The primary focus of this system is to provide a comprehensive and user-friendly platform for managing events within a college campus.

The system's usability is a significant advantage, as it simplifies the event management process, making it easier for teachers to create and post events and for students to view and apply to events. The system ensures that events are properly approved before they are visible to students, providing a secure and streamlined process.

The system's existence is also significant, as it eliminates the need for manual event management processes, which can be tedious and time-consuming. The event management system provides an efficient solution to manage events in a college environment, reducing the workload for event organizers and administrators.

The system's advantages are numerous, including the ability to advertise events to the whole college or a specific branch easily, which makes it easier for students to stay up-to-date on events. Additionally, the system provides a powerful smart authentication system that distinguishes between student, VC, teacher, and HOD, providing a secure platform for all users.

The event management system's advantages also include the ability to create and update events easily, as well as the power to accept or deny a student's request to attend an event. Finally, the system provides a streamlined process for issuing certificates to students upon completion of an event, which is a significant advantage for students who want to showcase their skills and knowledge to future employers.

CHAPTER 3

METHODS AND ALGORITHMS USED

3.1.1 HARDWARE REQUIREMENTS TO DEVELOP THE PROJECT:

1. CPU: Intel i3 7th Gen or higher
2. RAM: 4 GB or higher
3. Storage: 10 GB available space recommended
4. Internet Connection: Required

3.1.2 SOFTWARE REQUIREMENTS TO DEVELOP THE PROJECT:

1. VS Code – (Recommended Code Editor)

Extensions that have to be installed:

- Prettier (Code Formatter)
- Auto Close Tag
- IntelliCode
- Tailwind CSS Intellisense

2. NodeJS – v18.14.2 or later

3. NPM – v9.5.0 or later

Dependencies that have to be installed:

- **material-ui v5.11.11**
- **react v18.2.0**
- **next v13.2.3**
- **git v2.37.0**

4. Tailwind CSS – v3.2.7 or later

5. Python – v3.10.5 or later

6. Django – v4.1.7 or later

7. PyJWT - v2.6.0 or later

3.1.3 TECHNOLOGIES USED:

3.1.3.1 FRONT-END

1. HTML:

The preferred markup language for Web pages is HTML. This project made use of HTML5, the most recent version of HTML.

2. CSS:

The language we employ to style an HTML document is CSS. The latest release of CSS, CSS3, was used to dictate how HTML components should be rendered in this project.

Tailwind CSS:

Tailwind CSS is a utility-first CSS framework that provides a set of pre-defined classes that can be used to style HTML elements quickly and easily as illustrated in Figure 3. 1. It will be used to style the frontend of the application.

```
<div className="flex flex-row w-full mt-3 justify-between
items-center">
     <div className="flex flex-row mx-4 items-center gap-
2">
          <p className="text-lg font-normal text-
black">{date}</p>
     </div>
</div>
```

Figure 3. 1 Tailwind CSS Illustration

3. JavaScript

The Web's primary programming language is JavaScript. The newest version, ES6, was applied to the project.

ReactJS:

React is a popular JavaScript library for building user interfaces. It will be used as the primary framework for building the frontend of the application. The project also

made advantage of it.

NextJS:

Next.js is a React framework that provides server-side rendering as illustrated in Figure 3. 2, automatic code splitting, and other useful features out of the box. It will be used to build the frontend of the application.

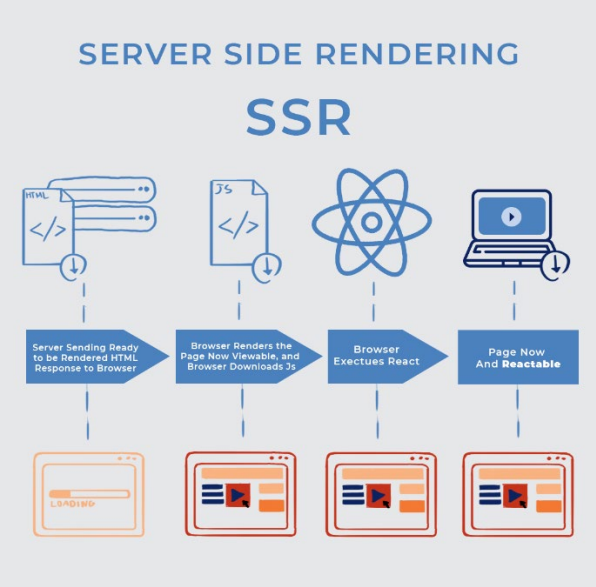

Figure 3. 2 Illustration on how SSR works

NPM

NPM is the package manager for JavaScript. It provides a command-line interface called npm.

In addition to the above, the application will also use various other libraries and tools such as webpack, babel, material-ui, axios, yup, formik, dayjs, typescript and others to facilitate the development process.

3.1.3.2 BACK-END

1. Django

Django is a high-level Python web framework that provides a robust set of tools for building web applications quickly and easily. It will be used as the primary framework

for building the backend of the application.

2. Django REST Framework:

Django Rest Framework is a powerful and flexible toolkit for building Web APIs. It will be used to build a RESTful API that can be used by the frontend to communicate with the backend.

3. MySQL

MySQL is a popular open-source relational database management system. It will be used as the database for the application.

DATA MODEL's USED:

Users table

The Users table as illustrated in Figure 3. 3 holds the information of all users who can log in to the system. Each user has a unique username and email address, and can be either a student or a teacher, as indicated by the is_student and is_teacher fields.

Field	Type	Description
id	int	Primary key for the user
username	varchar(255)	Unique username for the user
password	varchar(255)	Encrypted password for the user
email	varchar(255)	Unique email address for the user
first_name	varchar(255)	First name of the user
last_name	varchar(255)	Last name of the user
is_student	bool	Flag to indicate if the user is a student
is_teacher	bool	Flag to indicate if the user is a teacher

Figure 3. 3 User Table

Registrations Table

The Registrations table as illustrated in Figure 3. 4 holds the information of all registrations made by students for events. Each registration is made by a user and

for an event, which is indicated by the user_id and event_id fields, respectively. The is_attended field indicates whether the student attended the event or not.

Field	Type	Description
id	int	Primary key for the registration
user_id	int	Foreign key to the user who registered for the event
event_id	int	Foreign key to the event that the user registered for
is_attended	bool	Flag to indicate if the user attended the event

Figure 3. 4 Registration Table

Events table

The Events table as illustrated in Figure 3. 5 holds the information of all events that can be posted by teachers. Each event has a title, description, start time, end time, location, and a maximum number of attendees. The is_approved field indicates whether the event has been approved by the HOD, Dean, and Vice Chancellor. The is_completed field indicates whether the event has already taken place. Each event is created by a user, which is indicated by the created_by_id field.

Field	Type	Description
id	int	Primary key for the event
title	varchar(255)	Title of the event
description	text	Description of the event
start_time	datetime	Start time of the event
end_time	datetime	End time of the event
location	varchar(255)	Location of the event
max_attendees	int	Maximum number of attendees for the event
is_approved	bool	Flag to indicate if the event is approved
is_completed	bool	Flag to indicate if the event is completed
created_by_id	int	Foreign key to the user who created the event

Figure 3. 5 Event Table

3.1.3.3 DEPLOYMENT

Our deployment process for the Event Management System is designed to ensure that the application runs smoothly and reliably in a production environment. We use Docker containers to package the application and its dependencies, which makes it easy to deploy the same application on different platforms.

We will deploy the application on cloud platforms like AWS or GCP, which offer scalability and flexibility in managing resources. Before deployment, we perform extensive testing to ensure that the application works as expected and is ready for production.

3.1.4 SOURCE CODE:

Back-end data models:

```
class Event(models.Model):
    STATUS_CHOICES = (
        (1, 'Pending'),
        (2, 'Approved'),
        (3, 'Rejected'),
        (4, 'Completed'),
        (5, 'Cancel'),
        (6, 'Certified'),
        (7, 'Ongoing'),
    )
    owner = models.ForeignKey(User, on_delete=models.CASCADE)
    status = models.PositiveIntegerField(choices=STATUS_CHOICES, default=1)
    organizer = models.ManyToManyField(User, related_name='event_organizer',
blank=True)
    participant = models.ManyToManyField(User, related_name='event_participant',
blank=True)

    image = models.ImageField(upload_to='poster/')
    title = models.CharField(max_length=250)

short_description = models.CharField(max_length=100)
    long_description = models.TstField(null=True, blank=True)
    club = models.CharField(max_length=CLUB_LENGTH)
    venue = models.CharField(blank=True, null=True, max_length=100)

    start_date = models.DateField(blank=True, null=True)
    end_date = models.DateField(blank=True, null=True)
```

```
    date = models.TextField(blank=True, null=True)
    time = models.TextField(blank=True, null=True)

    branch = models.ManyToManyField(Branch, blank=True, null=True)

    messages = models.JSONField(blank=True, null=True)  # [{'message':'', from:'',
datetime:'', status:'Rejected}]
    hod_verified = models.BooleanField(default=False)
    dean_verified = models.BooleanField(default=False)
    vc_verified = models.BooleanField(default=False)

    def __str__(self):
        return self.title

class UserManager(BaseUserManager):
    def create_user(self, college_id, email=None, password=None, role=0):
        if not college_id:
            raise ValueError('Users must have College id')
        if not email:
            raise ValueError('User must have a email id')
        if role is None:
            raise ValueError('Role is Required')
        user = self.model(
            college_id=college_id,
            email=self.normalize_email(email),
            role=role
        )
        user.set_password(password)
        user.save(using=self._db)
        return user

    def create_staffuser(self, college_id, email, password, role):
        user = self.create_user(
            college_id=college_id,
            email=email,
            password=password,
            role=role
        )
        user.is_staff = True
        user.save(using=self._db)
        return user

    def create_superuser(self, college_id, email, password, role):
        user = self.create_user(
            college_id=college_id,
            email=email,
            password=password,
            role=role
        )
```

```
        user.is_staff = True
        user.is_superuser = True
        user.save(using=self._db)
        return user

class User(AbstractBaseUser, PermissionsMixin):
    ROLE_CHOICE = (
        (0, 'Student'),
        (1, 'Teacher'),
        (2, 'HOD'),
        (3, 'Dean'),
        (4, 'Vice-Chancellor'),
    )
    college_id = models.CharField(
        verbose_name='Register/Emp Number',
        max_length=10,
        unique=True,
    )
    first_name = models.CharField("First Name", max_length=150, blank=True)
    last_name = models.CharField("Last Name", max_length=150, blank=True)
    email = models.EmailField("Email Address", blank=True)
    date_joined = models.DateTimeField(_("date joined"), default=timezone.now)

    is_active = models.BooleanField(default=True)
    is_staff = models.BooleanField(default=False)

    role = models.PositiveIntegerField(choices=ROLE_CHOICE)

    branch = models.ForeignKey(Branch, on_delete=models.CASCADE,
blank=True, null=True)

    joining_year = models.CharField(max_length=4, blank=True, null=True)
    leaving_year = models.CharField(max_length=4, blank=True, null=True)

    objects = UserManager()

    EMAIL_FIELD = "email"
    USERNAME_FIELD = 'college_id'
    REQUIRED_FIELDS = ['email', 'role'] # Email & Password are required by
default.

    class Meta(AbstractUser.Meta):
        swappable = "AUTH_USER_MODEL"

    def __str__(self):
        return f"{self.first_name} {self.last_name} ({self.email})"

    @property
    def full_name(self):
```

```
        return f"{self.first_name} {self.last_name}".strip()
```

```
import axios from 'axios';
import {Constructor} from 'type-fest';

const instance = axios.create({
    baseURL: (() => {
        if (typeof window !== 'undefined') {
            return (
                window.location.protocol + '//' + window.location.hostname + ':8000/api/'
            );
        }
        return '';
    })(),
    // withCredentials: true,
    headers: {
        'Content-Type': 'application/json',
        'Authorization':
            'Bearer ' +
            (() => {
                if (
                    typeof window !== 'undefined' &&
                    window.localStorage.hasOwnProperty('access')
                ) {
                    return window.localStorage.access;
                }
                return '';
            })(),
    },
});

const hierarchy = {
    0: 'Student',
    1: 'Teacher',
    2: 'HOD',
    3: 'Dean',
    4: 'Vice-Chancellor',
};

class AxiosInstance {
    get_config(headers?: any) {
        return {
            headers: {
                ...headers,
                Authorization: (() => {
                    if (
                        typeof window !== 'undefined' &&
                        window.localStorage.hasOwnProperty('access')
                    ) {
```

```
                return 'Bearer ' + window.localStorage.access;
            }
            return '';
        })(),
    },
};
}

async post(pathname: string, data: {[key: string]: string}, headers: {}) {
    await this.update_token();
    const config = this.get_config(headers);
    return await instance.post(pathname, data, config);
}
async put(pathname: string, data: {[key: string]: string}, headers: {}) {
    await this.update_token();
    const config = this.get_config(headers);
    return await instance.put(pathname, data, config);
}
async patch(pathname: string, data: {[key: string]: string}, headers: {}) {
    await this.update_token();
    const config = this.get_config(headers);
    return await instance.patch(pathname, data, config);
}

async login(username: string, password: string) {
    if (typeof window === 'undefined') {
        return false;
    }
    const form = new FormData();
    form.append('college_id', username);
    form.append('password', password);
    const request = await instance.post(get_url('login'), form);
    for (let key in request.data) {
        window.localStorage.setItem(key, request.data[key]);
    }
    return request;
}

async get(pathname: string, data?: {[key: string]: string}, headers?: any) {
    await this.update_token();
    const config = this.get_config(headers);
    let data_str = '?';
    if (data !== null) {
        for (let key in data) {
            data_str += `${key}=${data[key]}&`;
        }
    }
    return await instance.get(pathname + data_str, config);
}
```

```
    async update_token() {
        let status = false;
        if (!this.__check_expiry()) {
            status = await this.__refresh_token();
            if (!status) {
                window.location.href = '/';
            }
        }
        return status;
    }

    async __refresh_token() {
        if (typeof window === 'undefined') return false;

        const token = window.localStorage.getItem('refresh');
        const expiryTime = token !== null ? parseJwt(token).exp : null;
        let currTime = parseInt(String(new Date().getTime() / 1000));
        if (expiryTime === null || token === null || parseInt(expiryTime) < currTime) {
            return false;
        }
        const form_data = new FormData();
        form_data.append('refresh', token);
        const request = await instance.post(get_url('token_refresh'), form_data);
        if (request.status !== 200) {
            return false;
        }
        const access = request.data.access;
        if (access === null) {
            return false;
        }
        window.localStorage.setItem('access', access);
        return true;
    }

    __check_expiry() {
        if (typeof window === 'undefined') return false;
        const token = window.localStorage.getItem('access');
        if (token !== null) {
            const expiryTime = parseJwt(token).exp;
            return (
                expiryTime !== null &&
                parseInt(expiryTime) >= parseInt(String(new Date().getTime() / 1000))
            );
        }
        return false;
    }
}
```

```
function parseJwt(token: string) {
    if (token === null || token.trim() === '') {
        return null;
    }
    let base64Url = token.split('.')[1];
    let base64 = base64Url.replace(/-/g, '+').replace(/_/g, '/');
    let jsonPayload = decodeURIComponent(
        window
            .atob(base64)
            .split('')
            .map(function (c) {
                return '%' + ('00' + c.charCodeAt(0).toString(16)).slice(-2);
            })
            .join('')
    );
    return JSON.parse(jsonPayload);
}

function get_url(key: string, params?: Array<string>) {
    if (params) {
        return url[key](...params);
    }
    return url[key]();
}

const is_logged_in = () => {
    if (typeof window === 'undefined') {
        return false;
    }
    const access = API.jwt(window.localStorage.getItem('access') || '');
    if (access === null) {
        return false;
    }
    const refresh = API.jwt(window.localStorage.getItem('refresh') || '');
    if (refresh === null) {
        return false;
    }
    return parseInt(refresh.exp) >= parseInt(String(new Date().getTime() / 1000));
};

const API: {[key: string]: any} = {
    Axios: AxiosInstance,
    jwt: parseJwt,
    get_url: get_url,
    is_logged_in: is_logged_in,
};

const url: {[key: string]: Function} = {
    'login': () => {
```

```
        return 'user/token/';
    },
    'profile_detail': (id: Number) => {
        return `user/detail/${id}/`;
    },
    'token_refresh': () => {
        return `user/token/refresh/`;
    },
    'event:completed_list': () => {
        return 'event/completed/list/';
    },
    'event:ongoing_list': () => {
        return 'event/ongoing/list/';
    },
    'event:upcoming_list': () => {
        return 'event/upcoming/list/';
    },
    'event:detail': (id: Number) => {
        return `event/detail/${id}/`;
    },
    'event:club_branch': () => {
        return 'event/club/branch/';
    },
    'event:create': () => {
        return 'event/create/';
    },
    'event:update': (id: Number) => {
        return `event/update/${id}/`;
    },
};
export default API;
```

Front-end code illustration:

```
"use client";
import HomeCard from "./card";
import { useState } from "react";
import API from "../API";
import Page from "./pagination";
import TextField from "@mui/material/TextField";
const axios = new API.Axios();
import InputAdornment from "@mui/material/InputAdornment";
import Image from "next/image";
import useEffect from "../useEffect";
```

```
import ApiLoader from "../apiLoader";
import handleError from "../handleError";

function LoadingCard() {
  return (
    <div className="flex flex-col border border-gray-400 rounded-md bg-white transition-all duration-300 w-80 justify-center items-center p-2">
      <div className="w-72 truncate h-8 animate-pulse bg-gray-300 rounded-xl mb-2 mt-2"></div>
      <div className="w-72 truncate h-4 animate-pulse bg-gray-200 rounded-xl mb-4"></div>
      <div className="flex items-center h-96 w-11/12 border border-gray-300 p-2 animate-pulse">
        {" "}
        <svg
          className="text-gray-300"
          xmlns="http://www.w3.org/2000/svg"
          aria-hidden="true"
          fill="currentColor"
          viewBox="0 0 640 512"
        >
          <path d="M480 80C480 35.82560 456.1L0 456.1z" />
        </svg>
      </div>
      <div className="w-72 truncate h-6 animate-pulse bg-gray-200 rounded-xl mb-2 mt-4"></div>
    </div>
  );
}

export default function Main(props: { url: string; heading: string }) {
  const [data, setData] = useState([
    {
```

```
    pk: "",
    title: "",
    club: "",
    image: "",
    short_description: "",
    date: "",
  },
]);
const [pageNo, setPageNo] = useState(1);
const [totalPage, setTotalPage] = useState(1);
const [search, setSearch] = useState("");
const [Loader, setLoader] = useState(0);

const [showSearch, setShowSearch] = useState(false);

useEffect(() => {
  const width = window.innerWidth;
  if (width >= 1200) {
    setNumCards(4);
  } else if (width >= 600) {
    setNumCards(2);
  } else {
    setNumCards(1);
  }
}, []);

const [numCards, setNumCards] = useState(4);
const cards = Array(numCards).fill(0);

useEffect(
  () => {
    const query = window.setTimeout(async () => {
      try {
```

```
      const response = await axios.get(API.get_url(props.url), {
        page: pageNo,
        q: search,
      });
      if (response.status === 200) {
        if (!response.data.hasOwnProperty("results")) {
          setPageNo(1);
          return;
        }
        setTotalPage(response.data.total_pages);
        setData(response.data.results);
        setShowSearch(true);
        if (response.data.count === 0) {
          setLoader(404);
        } else {
          setLoader(200);
        }
      } else {
        setLoader(response.status);
      }
    } catch (err) {
      handleError(err, setLoader);
      setPageNo(1);
    }
  }, 500);

  return () => {
    window.clearInterval(query);
  };
 },
 [pageNo, search],
 setLoader
);
```

```
return (
  <div className="flex flex-col w-full h-full items-center gap-3">
    {showSearch ? (
      <>
        <h1 className="text-2xl text-center mt-3 underline animateFadeIn">
          {props.heading}
        </h1>
        <div className="p-3 w-11/12 md:w-1/2 rounded-xl animateFadeIn">
          <TextField
            autoComplete="off"
            onChange={(e) => setSearch(e.target.value)}
            label="Search for events by name, club, branch, or description."
            value={search}
            placeholder="Start typing..."
            size="medium"
            className="w-full"
            InputProps={{
              startAdornment: (
                <InputAdornment position="start">
                  <svg
                    xmlns="http://www.w3.org/2000/svg"
                    fill="none"
                    viewBox="0 0 24 24"
                    strokeWidth={1.5}
                    stroke="currentColor"
                    className="w-6 h-6"
                  >
                    <path
                      strokeLinecap="round"
                      strokeLinejoin="round"
                      d="M21 21l-5.197-5.197m0 0A7.5d0 0010.607 10.607z"
                    />
```

```
                </svg>
              </InputAdornment>
            ),
          }}
          variant="standard"
        />
      </div>
    </>
  ) : (
    <>
      <div className="mt-3 w-48 rounded-xl">
        <div className="h-10 animate-pulse bg-gray-300 rounded-xl w-full"></div>
      </div>
      <div className="p-3 pt-0 w-11/12 md:w-1/2 rounded-xl">
        <div  className="h-12  animate-pulse  bg-gray-300  rounded-xl  w-full  mb-
4"></div>
      </div>
    </>
  )}
  {data.length != 0 && (Loader !== 200 || data[0].pk === "") ? (
    <div className="flex flex-col justify-center items-center w-full min-h-[65vh]">
      <div className="flex flex-wrap justify-center items-center gap-3">
        {cards.map((_, index) => (
          <LoadingCard key={index} />
        ))}
      </div>
    </div>
  ) : (
    <div className="flex justify-center flex-col items-center gap-4">
      <div className="flex flex-row flex-wrap justify-center gap-3">
        {data.length !== 0 && data[0].pk != "" ? (
          data.map((card) => (
            <HomeCard
```

```
              key={card.pk}
              title={card.title}
              subheader={card.club}
              imageUrl={card.image}
              description={card.short_description}
              date={card.date}
              learnMoreLink={"/details/" + card.pk}
            />
          ))
        ) : (
          <div className="flex flex-col justify-center items-center w-full h-[61vh] animateFadeIn">
            <Image
              src="/eventsNotFound.avif"
              width={500}
              height={500}
              priority
              alt=""
            ></Image>
            <p className="text-2xl font-light text-[#60adf5] mt-4">
              No events found!!
            </p>
          </div>
        )}
      </div>
      <Page totalPage={totalPage} pageNo={pageNo} setPageNo={setPageNo} />
    </div>
  )}
 </div>
 );
}
```

CHAPTER 4

RESULTS AND PERFORMANCE ANALYSIS

4.1 CLIENT REQUIREMENTS:

Minimum Hardware Requirements:

1. CPU: Intel Pentium 4 or later
2. RAM: 2 GB or higher
3. Storage: 2 GB available space recommended
4. Internet Connection: Required

Software Requirements:

5. Latest version of any Browser (Chrome Recommended)

Web Testing, or website testing is checking your web application or website for potential bugs before it is made live and is accessible to general public. Web Testing checks for functionality, usability, security, compatibility, performance of the web application or website.

Testing was done in 4 phases:

1. Functionality Testing

It is the process that includes several testing parameters like user interface, APIs, database testing, security testing, client and server testing and basic website functionalities. Functional testing is very convenient and it allows users to perform both manual and automated testing. It is performed to test the functionalities of each feature on the website.

2. Usability Testing

It has now become a vital part of any web-based project. It can be carried out

like a small focus group similar to the target audience of the web application.

3. Interface Testing

   Three areas to be tested here are – Application, Web and Database Server.

4. Database Testing

Database is one critical component of your web application and stress must be laid to test it thoroughly.

Testing activities will include-

- Test if any errors are shown while executing queries
- Data Integrity is maintained while creating, updating or deleting data in database.

4.2 TESTING CONDITIONS:

Configuration: Ryzen 7 5800H 8 CPU cores, Nvidia GTX 1650, 16 GB RAM
Server-Side and Client-Side were run on the local machine.

4.3 ANALYSIS AND IMPROVEMENTS

1. Server-Side Rendering (SSR) - To improve the initial page load time, we implemented Server-Side Rendering which generates HTML on the server and sends it to the client, reducing the time needed to render the page.

2. Optimized queries - To minimize database access and reduce page loading time, queries were optimized to at most 1 query per user click. This helps reduce the number of queries needed to be executed and saves time.

3. Optimized images - To improve the loading speed of images, we optimized them to load faster. This includes compressing images and serving them in the appropriate size and format for the device.

4. Pruned unnecessary packages - We removed unnecessary packages from our

codebase to reduce its size and simplify its dependencies. This helps to optimize the build time, and makes the application faster and more efficient.

5. Optimized production builds - To meet the standards of DevOps providers, we optimized our production builds, enabling faster and more efficient deployment.

6. Limited package usage - We limited the usage of packages to only those that were necessary for the project, which helped to reduce the size of the codebase and simplify its dependencies.

7. Employed CDNs - We employed Content Delivery Networks (CDNs) wherever possible to serve static assets such as images, videos, and JavaScript files from a network of servers located around the world. This helped to improve the loading speed of the application for users across different regions.

CHAPTER 5

CONCLUSION

- The Event Management System for a college streamlines the process of managing events in a college.
- Teachers can create events and submit them for approval by the HOD, Dean, and Vice Chancellor.
- Once approved, students can view the events and apply to attend them through their accounts.
- The system generates certificates for students who have successfully attended the events.
- The system increases efficiency and streamlines event management, allowing teachers to focus on creating great events.
- Students can easily view and apply to events they are interested in attending, making the process of participating in extracurricular activities much easier.
- The Event Management System is a valuable tool for any college looking to manage their events more efficiently.
- With its user-friendly interface and robust features, it can help colleges streamline their event management process and provide a better experience for both teachers and students.

REFERENCES:

1. Django documentation.
   https://docs.djangoproject.com/en/3.2/
2. Django Rest Framework documentation.
   https://www.django-rest-framework.org/
3. Next.js documentation.
   https://nextjs.org/docs/getting-started
4. Tailwind CSS documentation.
   https://tailwindcss.com/docs
5. React documentation.
   https://reactjs.org/docs/getting-started.html
6. MySQL documentation.
   https://dev.mysql.com/doc/
7. "Event Management System: A Comprehensive Guide" by John Smith, published in Event Planning Journal, vol. 20, no. 2, pp. 30-45, 2020.
8. "Designing Distributed Systems: Patterns and Paradigms for Scalable, Reliable Services" by Brendan Burns, published by O'Reilly Media, 2018.
9. "Clean Architecture: A Craftsman's Guide to Software Structure and Design" by Robert C. Martin, published by Prentice Hall, 2017.

APPENDIX

SNAPSHOTS OF THE WEB–APP

SIGN-IN PAGE:

This page will allow users to log in or register on the website.

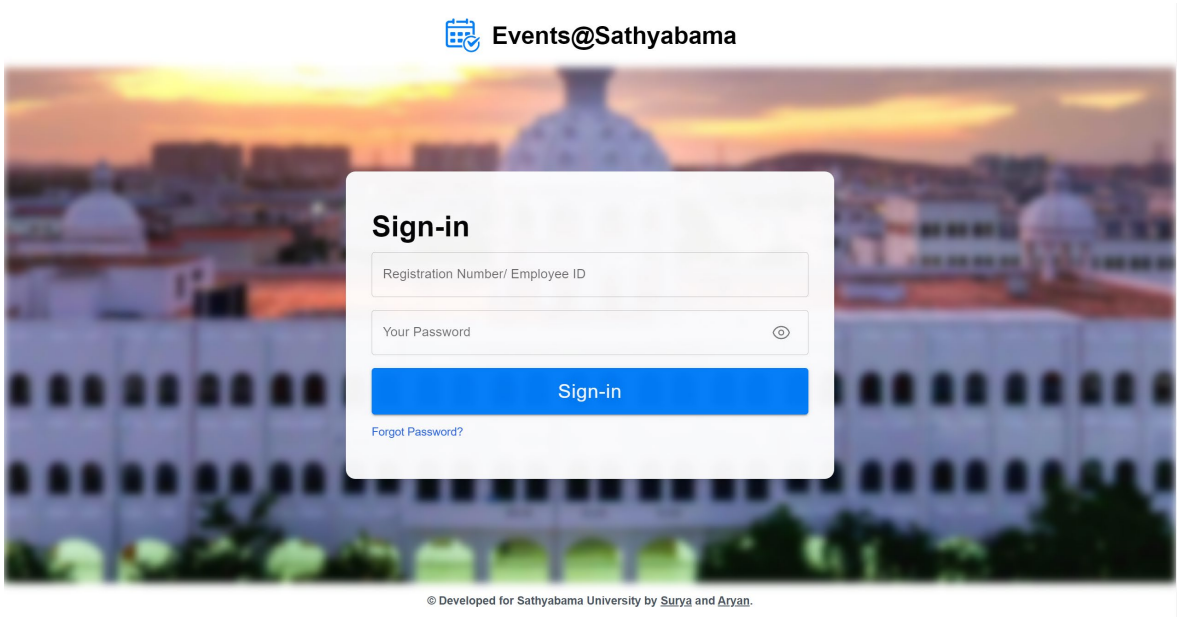

Figure 5.1 Sign-in Page

FORM-VALIDATION WITH YUP:

Schema-based, efficient, and customizable form validation library.

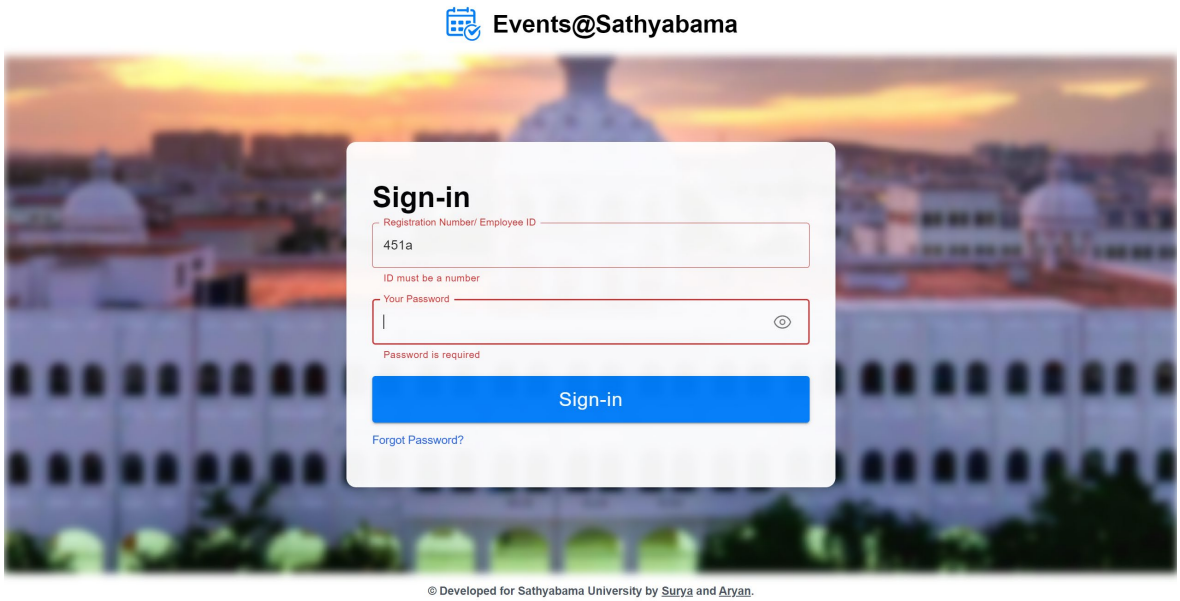

Figure 5. 2 Form-Validation with Yup

CREATE EVENT PAGE:

This page will allow teachers and admins to create new event cards. It should include all the fields that are required to create an event card.

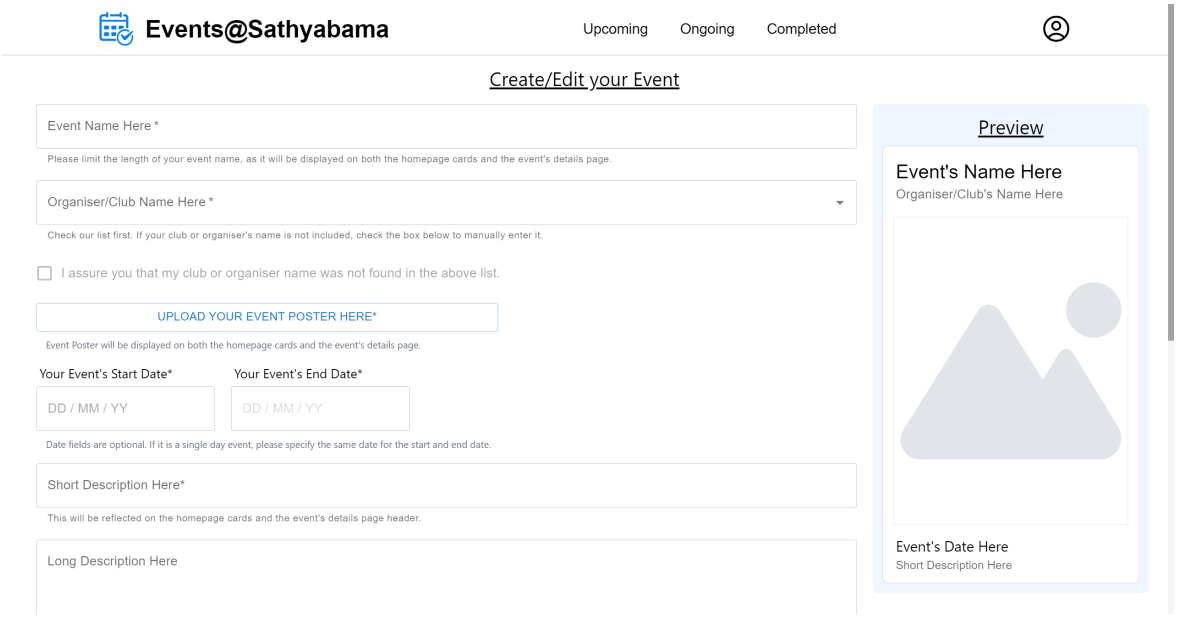

Figure 5. 3 Create-Event Page

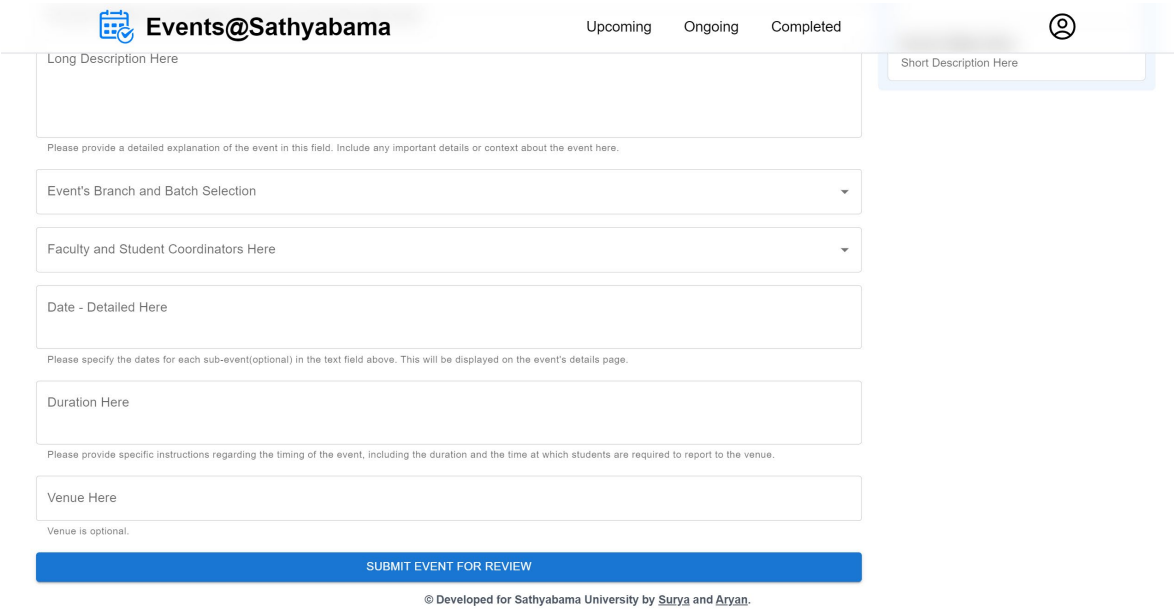

Figure 5. 4 Create-Event Page (Ctd.)

EVENTS-DISPLAY PAGE:

The page will display available events in three sections: Upcoming, Ongoing, and Completed. Upcoming events are for future events, Ongoing events are for current events, and Completed events are for past events.

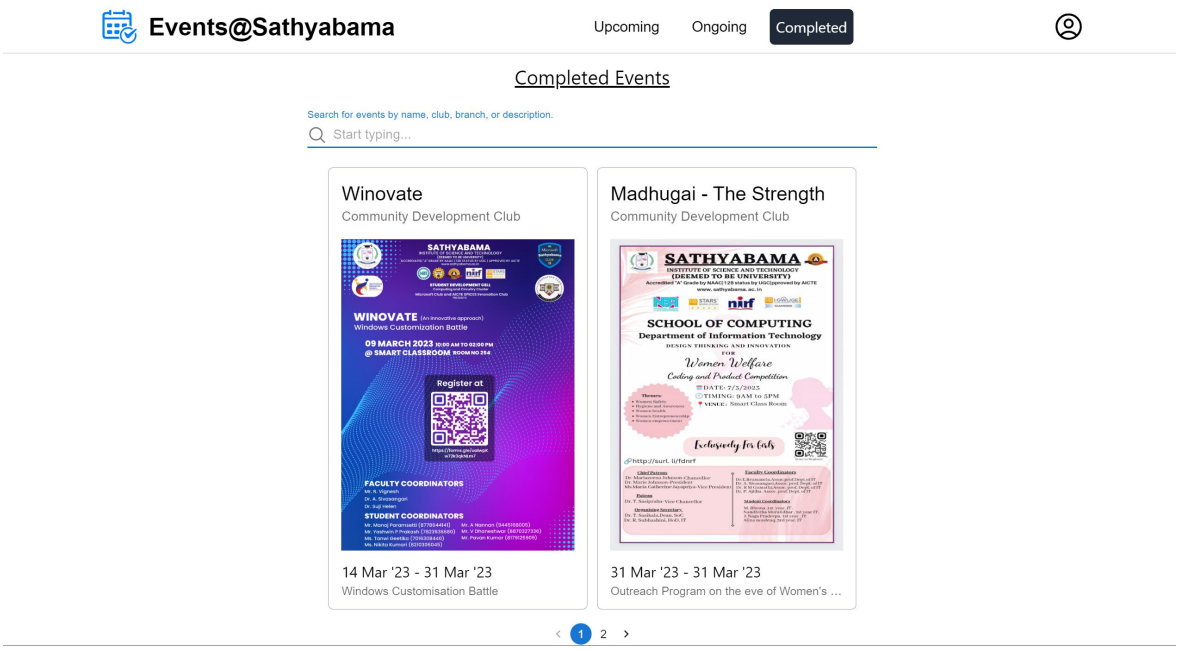

Figure 5. 5 Events-Display Page

EVENT DETAILS PAGE:

This page will display the details of a particular event. Application for the event can be done here.

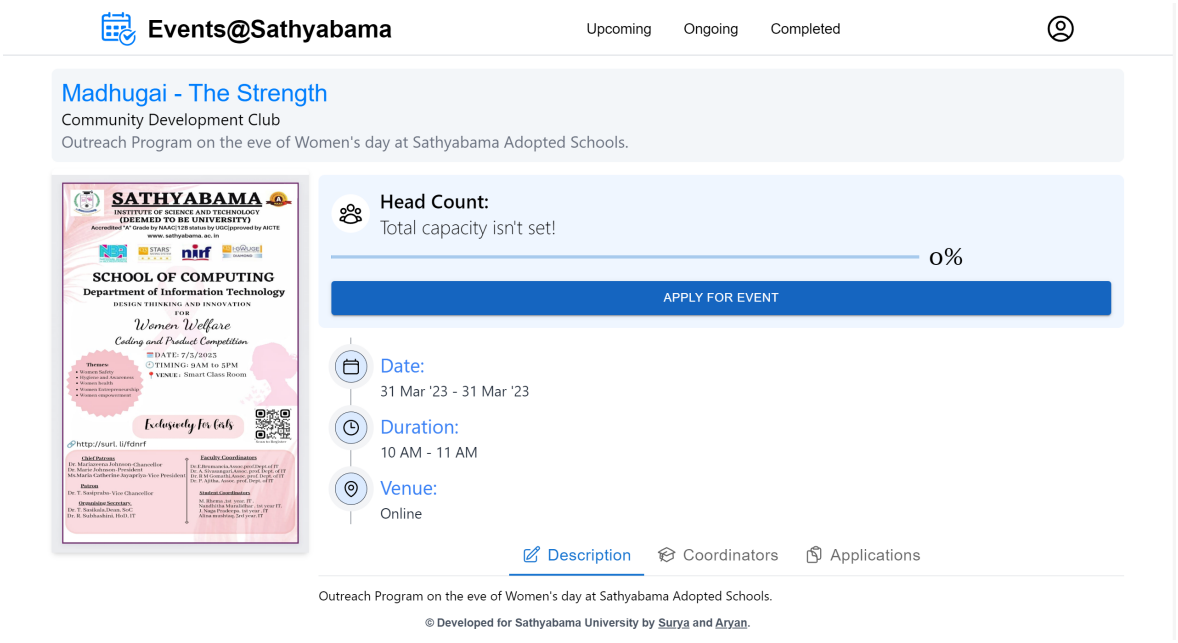

Figure 5. 6 Event-Details Page

EVENT COORDINATOR'S VIEW:

This page will contain information about the coordinators of a particular event.

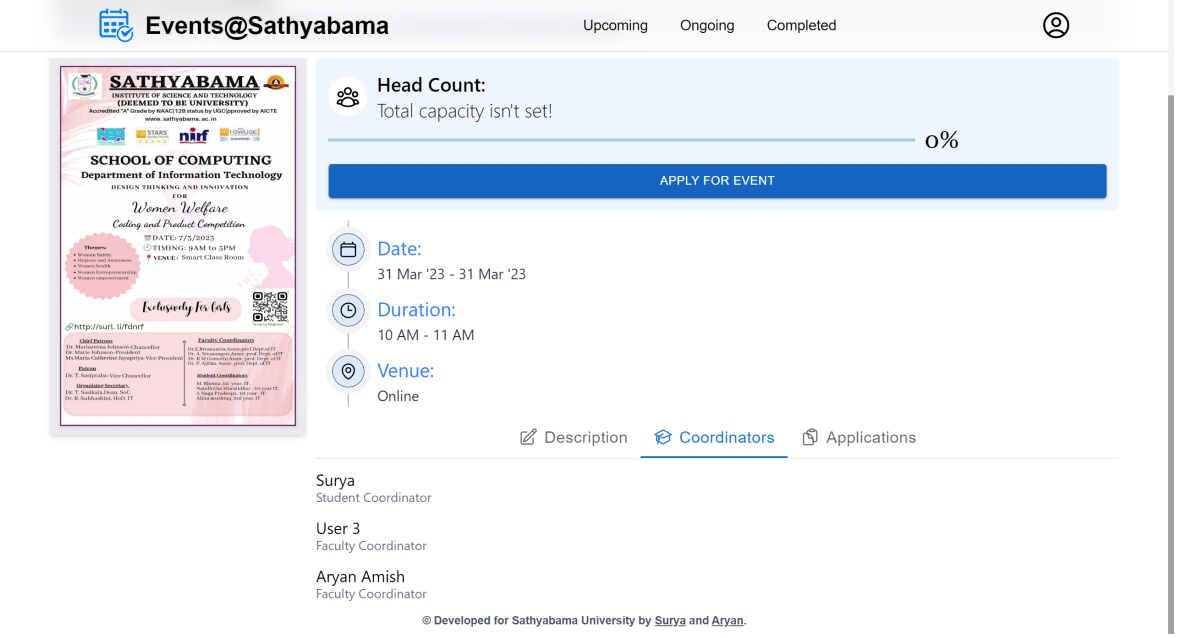

Figure 5. 7 Event Coordinator's View

EVENT'S PENDING APPLICATIONS VIEW:

Teacher's view of the pending student applications are displayed here.

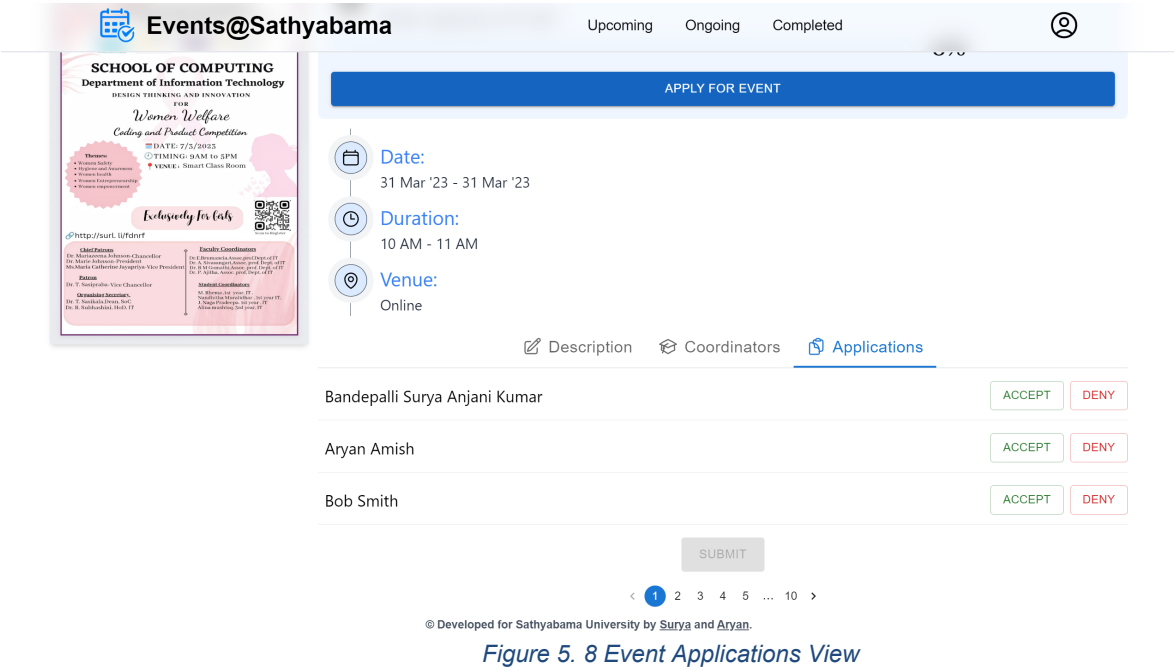

Figure 5. 8 Event Applications View

STUDENT’S PROFILE VIEW:

The page will showcase a user's profile, containing personal information along with a list of events they have participated in.

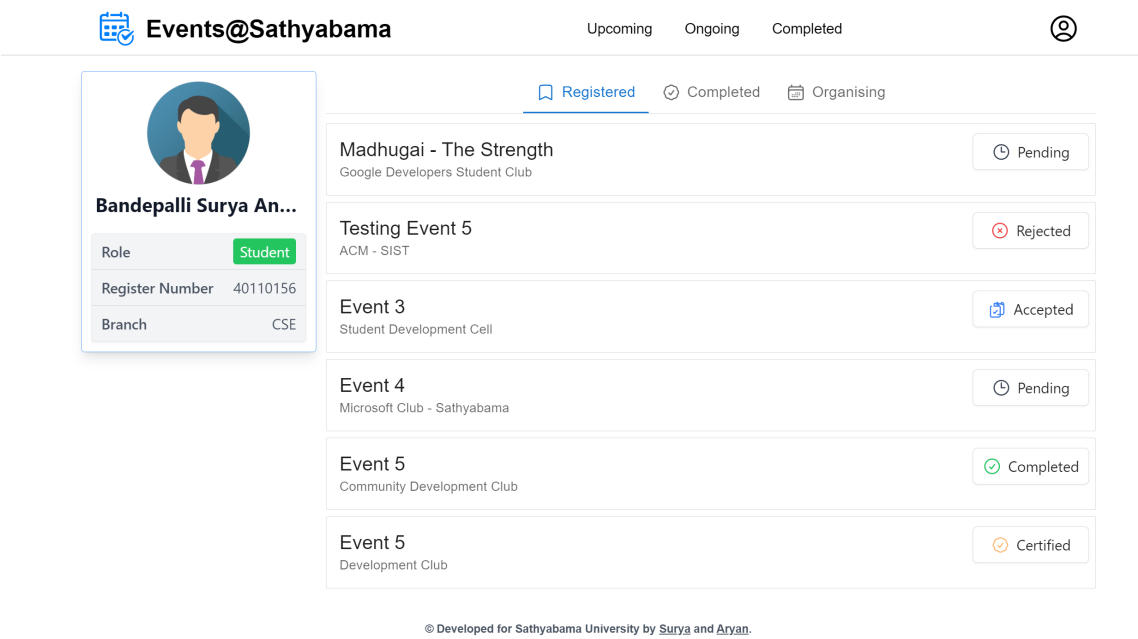

Figure 5. 9 Student Profile View

ORGANISER’S PROFILE VIEW:

This page will provide access to event organizer tools.

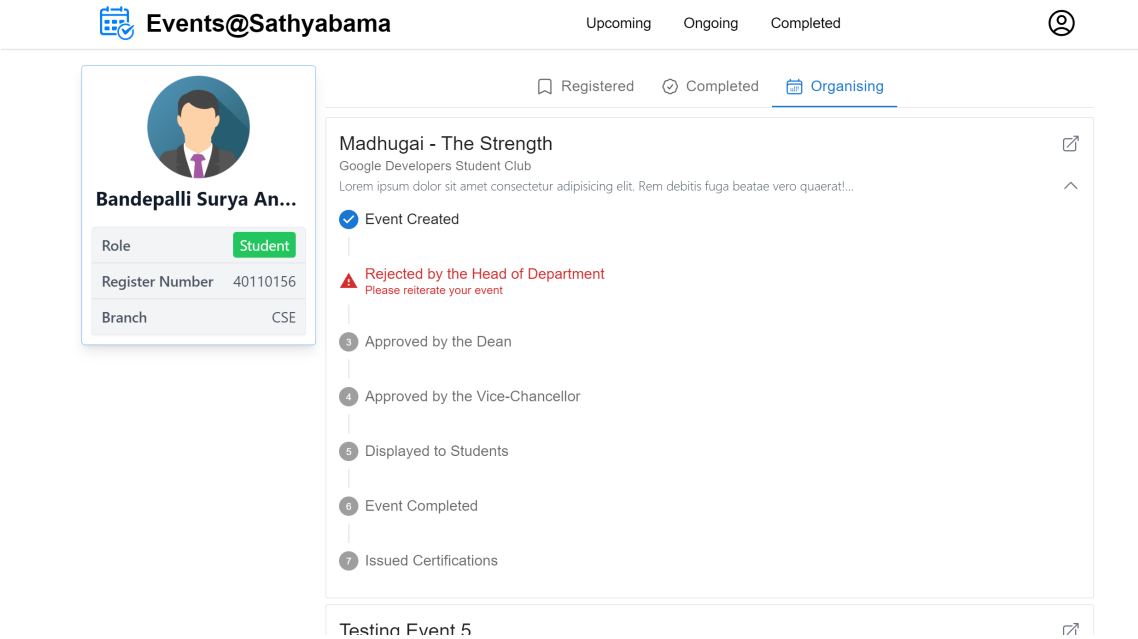

Figure 5. 10 Organiser's Profile View