EVENT MANAGEMENT SYSTEM

[21MCA4208]

Submitted By: -

CHARAN N [ENG22MCA034] VARUN J [ENG22MCA062] ARUN KUMAR A[ENG22MCA026]

In partial fulfilment for the award of the degree of

MASTER OF COMPUTER APPLICATIONS

DEPARTMENT OF COMPUTER APPLICATIONS

DAYANANDA SAGAR UNIVERSITYSCHOOL OF ENGINEERING

KUDLUGATE, HOSUR ROAD, BANGALORE -560068

OCTOBER 2023

ACKNOWLEDGEMENT

We wish to express my salutations to the beloved and highly esteemed institute **DAYANANDA SAGAR UNIVERSITY.** we thank **Dr. Uday Kumar Reedy**, Dean, School of Engineering, Dayananda Sagar University, Bangalore, for his valuable suggestions and expert advice. we would like to thank **Dr. Vasanthi Kumari P**, Chairman, Department of Computer Applications, Dayananda Sagar University for her valuable guidelines and constant support throughoutmy project.

We are grateful to our guide and would like to express special thanks to **Prof. Padmageetha**, Assistant Professor, Department of Computer Applications, DSU for her constant support, necessary advice, and valuable guidance that helped me in completing the project/report work successfully.

We would like to thank all the Teaching and non-teaching staff of the Departmentof Computer Applications, DSU for their help. We would like to thank **My parents** and **friends** who have always supported me in every path of the Reportwork.

We perceive this opportunity as a milestone in my career development. We will strive to use the gained skills and knowledge in the best possible way and We will continue to work on my improvement in order to achieve the desired career objectives. We hope to continue the cooperation with all of you in the future.

CHARAN N [ENG22MCA034] VARUN J [ENG22MCA062] ARUN KUMAR A[ENG22MCA026]

DAYANANDA SAGAR UNIVERSITY

Kudlu Gate, Hosur Road, Bangalore-560068

DEPARTMENT OF COMPUTER APPLICATIONS

Bonified Certificate

This is to certify that the project titled "**EVENT MANAGEMENT**" is a Bonified record of the work done by **CHARAN N [ENG22MCA034], VARUN J [ENG22MCA062]** and **ARUN KUMAR A[ENG22MCA026].**

In partial fulfilment of the requirements for the award of the degree of **Bachelor of Computer Applications** in **DAYANANDA SAGAR UNIVERSITY, BANGALORE,** during the year 2022-2023

Guide	**Chairman**
Prof. Padmageetha	**Dr. Vasanthi Kumari. P**
Assistant Professor	Professor &Chairperson
Dept. of Computer Applications,	Dept. of Computer Applications,
Dayananda Sagar University	Dayananda Sagar University

Project Viva Held on: - _____

Internal Examiner **External Examiner**

ABSTRACT

Our Event Management System is a web-based application powered by Django, designed to simplify event organization and planning. Users can effortlessly create, update, and manage events, with the option to categorize them by type. It includes a payment module for transaction management and provides real-time event updates and notifications. Role-based user access control ensures security. The system offers robust reporting and analytics for administrators and event organizers. With a mobile-responsive design, it can be accessed on various devices, making it a valuable tool for efficient event management.

Overall, this project of ours is being developed to help and manage the events efficiently in the best way possible and also reduce the human efforts in planning and organizing the events in a better way

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION OT THE PROJECT

The Event Management System is a sophisticated web application engineered using Django as the backend framework and supported by a robust database management system (DBMS). The frontend components, built with CSS and JavaScript, contribute to a visually appealing and responsive user interface. This comprehensive system simplifies the entire event lifecycle, allowing event creators to efficiently plan, update, and manage events.

The integration of DBMS ensures secure and efficient data storage and retrieval, while CSS and JavaScript contribute to a seamless user experience by providing an attractive and dynamic user interface. This Event Management System is a valuable tool that streamlines event coordination, making it user-friendly for both event organizers and participants. It combines the power of technology and user-centric design to facilitate event planning and management with ease and efficiency.

1.2 OBJECTIVES

User-Friendly Interface: Design a responsive and intuitive user interface using HTML, CSS, and JavaScript to ensure easy navigation and an enjoyable user experience.

Event Creation and Management: Implement a system that allows event organizers to create, manage, and edit event details, including event name, date, time, location, and description.

Registration and Ticketing: Develop a ticketing system that enables attendees to register for events, purchase tickets, and receive e-tickets for entry. This system should also support various payment methods.

User Authentication and Authorization: Implement user authentication and authorization mechanisms to ensure the security and privacy of user data. Different user roles, such as organizers and attendees, should have varying levels of access.

Database Management: Create a robust relational database using a DBMS like PostgreSQL or MySQL to store event-related data, user information, and transaction records.

Feedback and Ratings: Enable attendees to leave feedback and ratings for events they have attended, providing valuable insights for event organizers and helping others make informed decisions.

Mobile Responsiveness: Ensure that the system is mobile-responsive, making it accessible on a variety of devices, including smartphones and tablets.

Performance Optimization: Optimize the system for performance, ensuring fast loading times and efficient data retrieval, even during high traffic events.

Scalability: Design the system architecture in a way that allows for easy scalability to accommodate a growing number of users and events.

Security Measures: Implement security measures such as data encryption, secure authentication, and protection against common web vulnerabilities (e.g., SQL injection, XSS).

CHAPTER 2

LITERATURE REVIEW

Review of Event Management in Sports

Event management systems have become integral tools in the world of sports, streamlining the planning and execution of various events. The advent of digital technologies has revolutionized the way sports events are organized, making the process more efficient and engaging. Several key features have emerged as essential components of effective event management systems:

Event Creation, Digital platforms allow for the seamless creation of events. Organizers can input event details, including date, time, venue, and description, facilitating a structured approach to event planning.

Event Categorization, based on sports type, date, or location, enabling easy navigation and search functionalities. Categorization ensures that participants and attendees can quickly find events of their interest.

Event Updates, Real-time updates are crucial in the dynamic world of sports. Event management systems enable organizers to provide instant updates on schedule changes, participant lists, and other important announcements, ensuring all stakeholders are well-informed.

Event Notifications, Automated notifications via email, SMS, or in-app alerts keep participants and attendees informed about upcoming events, deadlines for registration, and any changes in event details. Notifications enhance communication and engagement among users.

User-Friendly Dashboard, A user-friendly interface is paramount for the success of any event management system. Intuitive dashboards provide a centralized hub where students, organizers, and attendees can access event information, register for events, and track their participation progress.

Previous Sports Events and Their Impact

Previous sports events serve as valuable case studies, highlighting the impact of effective event management on participant experience and community engagement. Examining these events sheds light on successful strategies and best practices:

Digital Ticketing and Access Control, Events like the Super Bowl have implemented digital ticketing systems, reduced the hassle of physical tickets and enabled secure access control. Digital ticketing ensures smooth entry for attendees and enhances security measures.

Fan Engagement through Mobile Apps, Major sports leagues, such as the NBA and NFL, have developed dedicated mobile apps for fans. These apps provide real-time updates, interactive features, and exclusive content, enhancing fan engagement before, during, and after events.

Social Media Integration, Events like the FIFA World Cup leverage social media platforms to connect with a global audience. Live streaming, fan discussions, and behind-the-scenes content create a sense of community among fans, amplifying the event's impact and reach.

Incorporating Virtual Reality (VR) and Augmented Reality (AR), Events like the Olympics have explored VR and AR technologies to offer immersive experiences to viewers. Virtual tours of stadiums, interactive maps, and 360-degree videos enhance the overall viewing experience, captivating audiences worldwide.

Community Outreach and Legacy Programs, Successful sports events often leave a lasting legacy within communities. Investments in sports facilities, youth development programs, and local infrastructure contribute to the overall well-being of the community, fostering a positive image of the event organizers.

CHAPTER 3

SYSTEM CONFIGURATION

3.1 **Hardware Specification**

Processor : Corei5 – 44604 Gen

RAM : 32GB

System type : 64BitOperatingSystem

Monitor : Dell24inch

Keyboard : Dual USB enter keyboard

Mouse : Dual optical two ball mouse USB

3.2 Software Specification

Platform : Microsoft Windows 10

Designing language : CSS 3.0, Java script

Backend : Django

3.3 Front End Coding Language

HTML 5.0

HTML stands for Hypertext Markup Language. It is the standard markup language for documents designed to be displayed in a web browser. HTML is heavily used for creating pages that are displayed on the World Wide Web. Every page contains a set of HTML tags including hyperlinks which are usedfor connecting to other pages. Every page that we witness, on the World Wide Web, is written using a version of HTML code. The basic building blocks of any HTML pages are HTML elements.

HTML is a tag-based language used for development of web pages. HTML tags are keywords (tag names) surrounded by angle brackets. The syntax is,

<tag name> content </tag name>

The following are the features of HTML,

It is easy to learn and easy to use.

It is platform-independent.

Images, videos and audio can be added to a webpage

Hypertext can be added to text.

It is a markup language.

The benefits and advantages of HTML are,

Implementation is easy.

It is used to create a website.

Helps in developing fundamentals about web programming.

HTML is used to build websites.

It is supported by all browsers.

It can be integrated with other languages like CSS, JavaScript and other programming languages.

Cascading Style Sheet (CSS) 3.0

Cascading Style Sheets, fondly referred to as CSS, is a simple design language intended to simplify the process of making web pages presentable. CSS handles the look and feel part of a web page. Using CSS, you can control the color of the text, the style of fonts, the spacing between paragraphs, how columns are sized and laid out, what background images or colors are used, layout designs, and variations in display for different devices and screen sizes as well as a variety of other effects. The advantages of the CSS are:

CSS saves time, the programmer writes the CSS code once and then reuse same sheet in multiple HTML pages. Defining a style for each HTML element and apply it to as many web pages, the programmer needs.

Pages load faster, CSS, allows the programmer to write the HTML tag attributes every time. Just write one CSS rule of a tag and apply it to all the occurrences of that tag. So, less code means faster download times.

Easy maintenance, to make a global change, simply change the style, and all elements in all the web pages will be updated automatically.

Superior styles to HTML, CSS has a much wider array of attributes than HTML, it provides the better look to the HTML page in comparison to HTML attributes.

Multiple Device Compatibility, style sheets allow the content to be

optimized for more than one type of device. By using the same HTML document, different versions of a website can be presented for handheld devices such as PDAs and cell phones or for printing.

Global web standards, the HTML attributes are being deprecated and it is being recommended to use CSS. It is a good idea to start using CSS in all the HTML pages and it makes compatible to future browsers.

JavaScript

JavaScript is one of the most popular languages which includes numerous features when it comes to web development. It's amongst the top programming languages as per GitHub and we must know the features of JavaScript properly to understand what it is capable of. The following are the main features JavaScript are

Light weight, is a lightweight scripting language because it is made for data handling at the browser level only. Since it is not a general-purpose language so it has a limited set of libraries.

Dynamic typed, supports dynamic typing which means types of the variable are defined based on the stored value. For example, declare a variable x it stores either a string number type value.

Object based language, is an object-based scripting language that provides built-in objects like string, math, date and other types

Functional style uses a functional approach, even objects are created from the constructor functions and each constructor function represents.

Platform independent, is platform-independent portable; which simply means by writing the script once and run it anywhere and

anytime. In general, the JavaScript applications and it can be executed on any platform or any browser without affecting the output of the Script.

Prototype based uses prototypes instead of classes. In JavaScript, each constructor function is associated with a prototype object.

A class in Java which is like a blueprint and to create objects for the class in JavaScript, define object prototype which is used to create other objects.

Interpreted means the script written inside JavaScript is processed line by line and is not compiled before execution. The scripts are interpreted by JavaScript interpreter which is a built-in component of the Web browser.

Asynchronous can be used to do complex processing asynchronously which means that the whole page will not have to wait for the JavaScript processing, and once the script processing completes and easily modify the HTML code to show or hide data. JavaScript to send async HTTP requests to the server for server-side processing too.

3.4 Backend Language

Django

Django framework Django is an open-source web application frame work written in Python. The primary goal of Django is to make the development of complex, data-based websites easier. Thus, Django emphasizes the reusability and pluggability of components to ensure rapid developments. Django consists of four major parts: model, view and template, URLs,

Model is a single, definitive data source which contains the essential field and behavior of the data. Usually, one model is one

table in the database. Each attribute in the model represents a field of a table in the database.

View is short form of view file. It is a file containing Python function which takes web requests and returns web responses. A response can be HTML content or XML documents or a "404 error" and so on. The logic inside the view function can be arbitrary as long as it returns the desired response. To link the view function with a particular URL we need to use a structure called Rulon which maps URLs to view functions.

Django' s template is a simple text file which can generate a text-based format like HTML and XML. The template contains variables and tags. Variables will be replaced by the result when the template is evaluated. Tags control the logic of the template. We also can modify the variables by using filters. For example, a lowercase filter can convert the variable from uppercase into lowercase

URLs: While it is possible to process requests from every single URL via a single function, it is much more maintainable to write a separate view function to handle each resource. A URL mapper is used to redirect HTTP requests to the appropriate view based on the request URL. The URL mapper can also match particular patterns of strings or digits that appear in a URL and pass these to a view function as data

Python

Python is the language used to build the Django framework. It is a dynamic

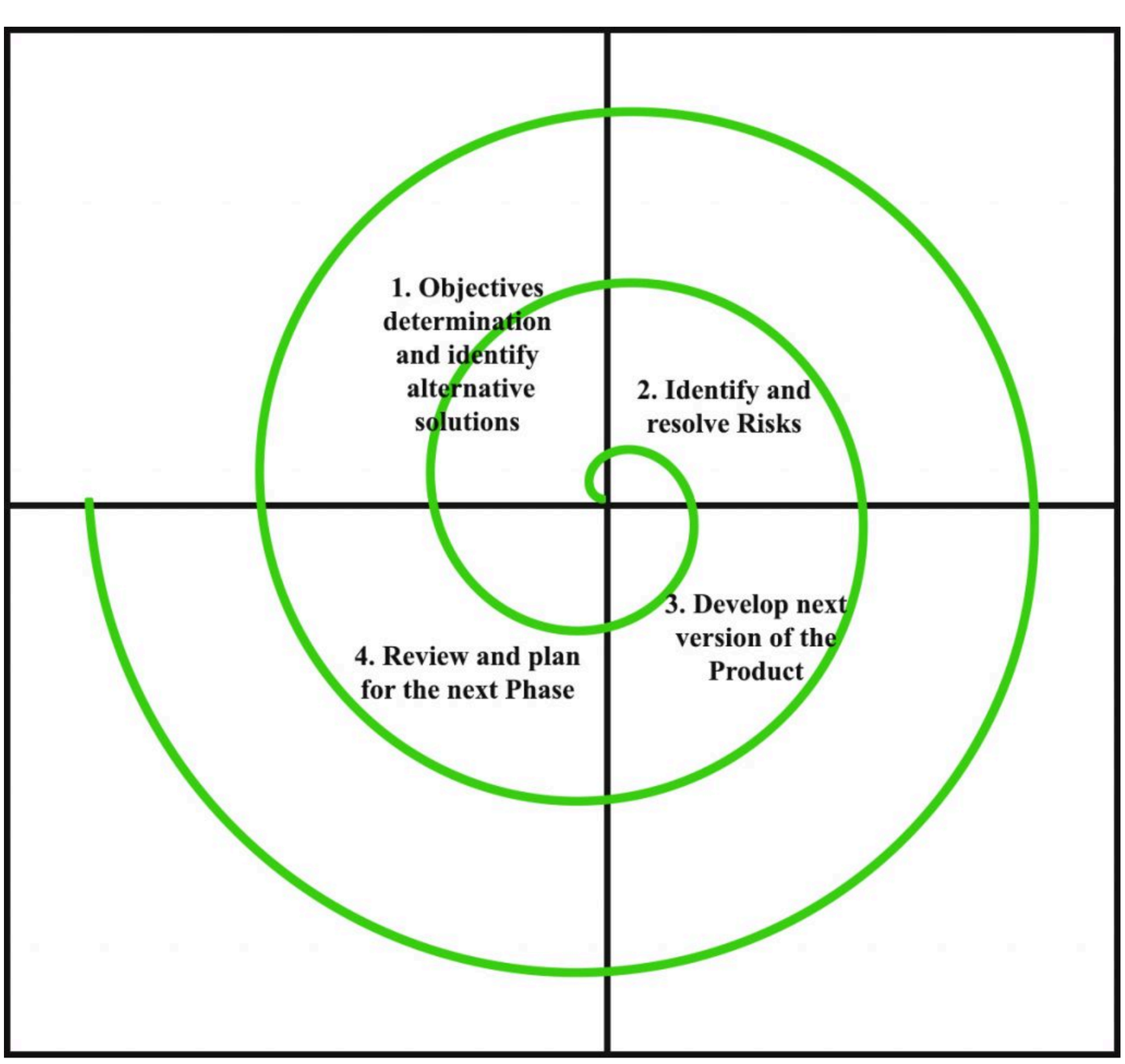
scripting language similar to Perl and Ruby. The principal author of Python is Guido van Rossum. Python supports dynamic typing and has a garbage collector for automatic memory management. Another important feature of Python is dynamic name solution which binds the names of functions and variables during execution.

CHAPTER 4

Solving Method

4.1 Solving Method Using Spiral Model

Spiral Model – Design



Spiral Model Approach

The Spiral Model is a risk-driven model, meaning that the focus is on managing risk through multiple iterations of the software development process. It consists of the following phases:

Planning: The first phase of the Spiral Model is the planning phase, where the scope of the project is determined and a plan is created for the next iteration of the spiral.

Risk Analysis: In the risk analysis phase, the risks associated with the project are identified and evaluated.

Engineering: In the engineering phase, the software is developed based on the requirements gathered in the previous iteration.

Evaluation: In the evaluation phase, the software is evaluated to determine if it meets the customer’ s requirements and if it is of high quality.

Planning: The next iteration of the spiral begins with a new planning phase, based on the results of the evaluation

Each phase of the Spiral Model is divided into four quadrants as shown in the above figure. The functions of these four quadrants are discussed below-

Requirements are gathered from the customers and the objectives are identified, elaborated, and analysed at the start of every phase. Then alternative solutions possible for the phase are proposed in this quadrant.

Identify and resolve Risks: During the second quadrant, all the possible solutions are evaluated to select the best possible solution.

Then the risks associated with that solution are identified and the risks are resolved using the best possible strategy. At the end of this quadrant, the Prototype is built for the best possible solution.

Develop the next version of the Product: During the third quadrant, the identified features are developed and verified through testing. At the end of the third quadrant, the next version of the software is available.

Review and plan for the next Phase: In the fourth quadrant, the Customers evaluate the so-far developed version of the software.

CHAPTER 5

DATA FLOW DIAGRAM

An important part of system analysis is the model of system flows and the system data. During the system analysis phase, a model of the system is developed which will be used during the system design phase. The aim of the design process is to produce the design specifications, which can be translated into machine understandable form during the coding stage.

Apart from achieving this principal objective, the design should lead to a program that is maintainable, extensible and understandable.

A dataflow diagram is graphical representation that depicts flow and transformations are applied as data move from input to output. It may be used to represent a system or software at any level of abstraction. It can be partitioned into levels that represent increasing information flow and functional details.

Data flow diagram is used to study the information flow and transformation through a computer-based systems. Raw data is fed through one end, the computer-based systems transform the data and the output come through the other end. The flow of datais represented using data flow diagrams.

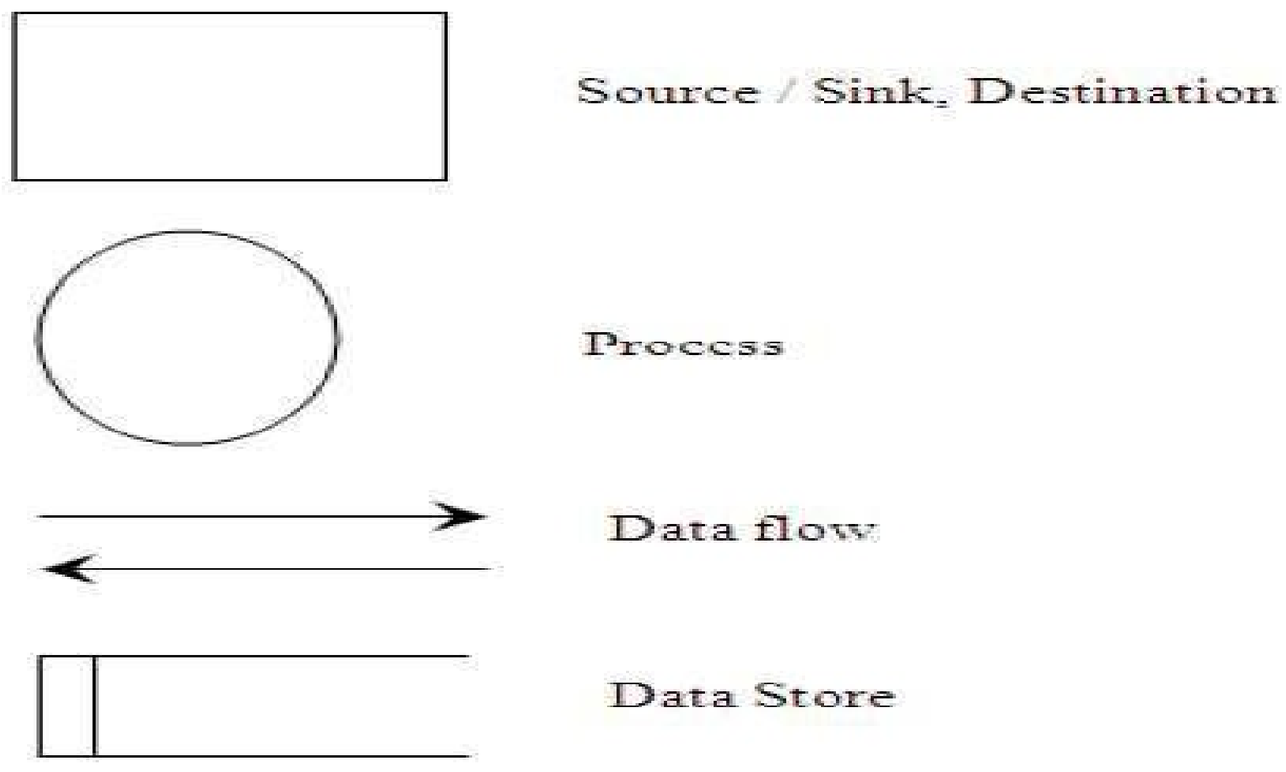
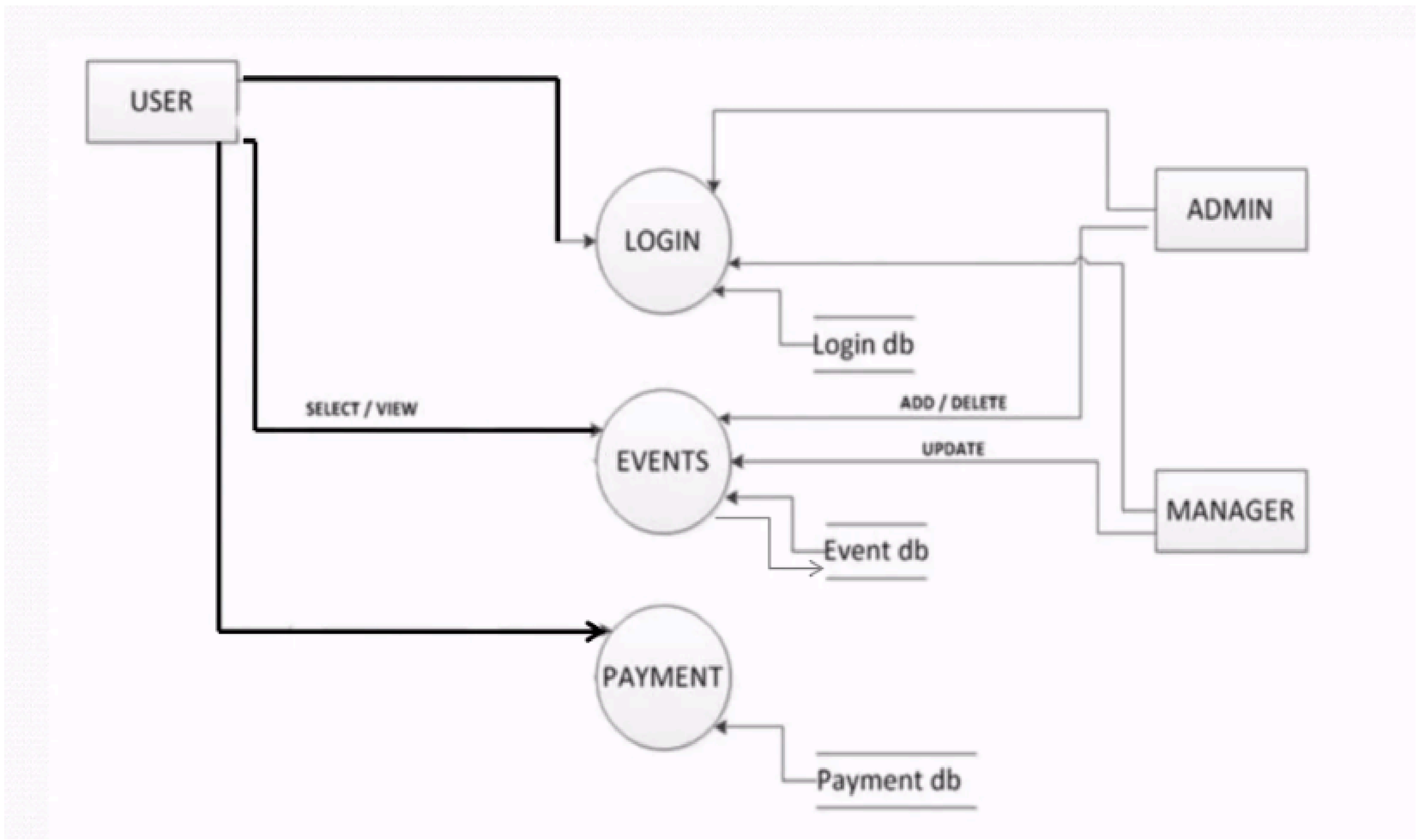


Fig No.5.1.1. Data Flow Diagram (DFD) Symbols

5.1 DATA FLOW DIAGRAM



5.2 Entity Relationship Diagram

Entity relationship set uses the two major abstractions to describe the data. The entity, defines about the distinct things in the organization. The relationships, defines about the meaningful interactions between the objects. An entity set is a collection of similar objects. A relationship set represents the relationship between the entity sets.

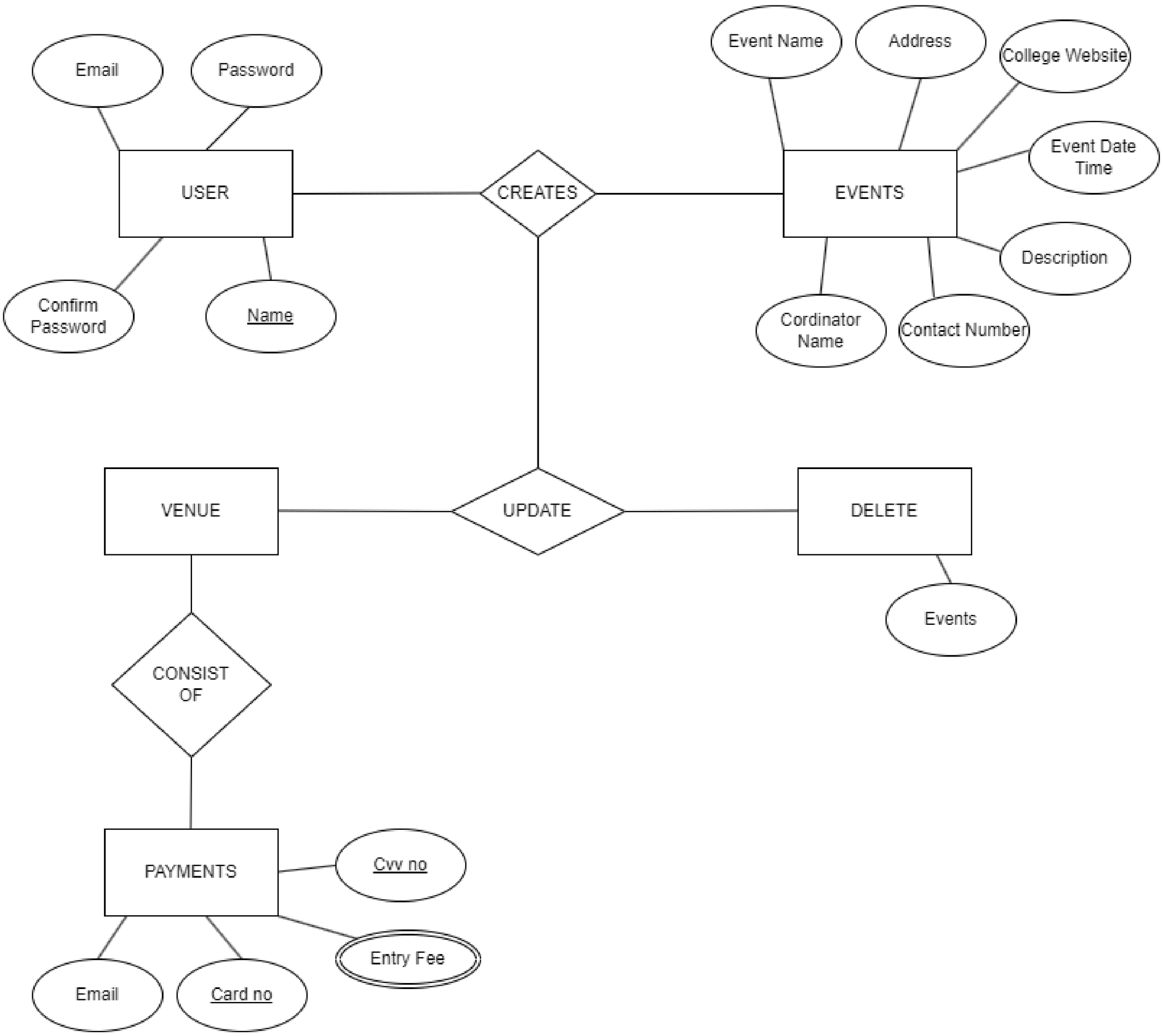


Fig No 5 .2.1 Entity Relationship Diagram (ERD) Symbols

APPENDICES

SOURCE CODE

Views

```
from Django. Shortcuts import render, redirect
from django.contrib.auth.models import User
from django.contrib.auth import authenticate, login, logout
from django.contrib.auth.decorators import login_required
from django.shortcuts import render, redirect
from django.contrib import messages
from django.template.loader import render_to_string
from .forms import EventForm
from datetime import datetime
from .models import Event
import random
from django.shortcuts import render, get_object_or_404, redirect
from django.utils import timezone
import random
from django.db import transaction  # Import the transaction module

from django.core.mail import send_mail
from django.conf import settings
from django.core.mail import send_mail
from .forms import PaymentForm
import logging
from .models import Payment

# Create your views here.
```

```
@login_required(login_url='login')
def HomePage(request):
    username = request.user.username
    # Retrieve upcoming events from your database or data source


    return render(request, 'home.html', {'username': username,})



def SignupPage(request):
    if request.method == 'POST':
        uname = request.POST.get('username')
        email = request.POST.get('email')
        pass1 = request.POST.get('password1')
        pass2 = request.POST.get('password2')

        if pass1 != pass2:
            error_message = "Your password and confirm password do not match."
            return render(request, 'signup.html', {'error_message': error_message})
        else:
            if User.objects.filter(username=uname).exists():
                error_message = "Username already exists. Please choose a different username."
                return render(request, 'signup.html', {'error_message': error_message})
            my_user = User.objects.create_user(uname, email, pass1)
            my_user.save()
            success_message = "User registered successfully! You can now log in."
            return render(request, 'login.html', {'success_message': success_message})


    return render(request, 'signup.html')


def LoginPage(request):
    if request.method == 'POST':
        username = request.POST.get('username')
        pass1 = request.POST.get('pass')
        user = authenticate(request, username=username, password=pass1)
```

```
        if user is not None:
            login(request, user)
            return redirect('home')
        else:
            error_message = "Username or Password is incorrect!!!"
            return render(request, 'login.html', {'error_message': error_message})
    return render(request, 'login.html')
# ... other imports ...
# ...

def forgot_password(request):
    if request.method == 'POST':
        email = request.POST.get('email')
        try:
            user = User.objects.get(email=email)
            otp = str(random.randint(100000, 999999))

            # Send OTP to user's email
            subject = 'Password Reset OTP'
            message = f'Your OTP is: {otp}'
            from_email = settings.DEFAULT_FROM_EMAIL
            recipient_list = [email]
            send_mail(subject, message, from_email, recipient_list)

            # Store email and OTP in session
            request.session['reset_email'] = email
            request.session['otp'] = otp

            return render(request, 'password/verify_otp.html')
        except User.DoesNotExist:
            error_message = "No user with this email address exists."
            return render(request, 'password/forgot_password.html', {'error_message': error_message})

    return render(request, 'password/forgot_password.html')

# verify_otp view
def verify_otp(request):
    if request.method == 'POST':
        entered_otp = request.POST.get('otp')
        email = request.session.get('reset_email')
```

```
        otp_from_session = request.session.get('otp')  # Retrieve the OTP from session
        user = User.objects.get(email=email)

        if entered_otp == otp_from_session:  # Compare entered OTP with OTP from session
            return redirect('reset_password', email=email)
        else:
            error_message = "Invalid OTP. Please try again."
            return render(request, 'password/verify_otp.html', {'error_message': error_message})
    return render(request, 'password/verify_otp.html')

# reset_password view
def reset_password(request, email):
    if request.method == 'POST':
        new_password = request.POST.get('new_password')
        confirm_password = request.POST.get('confirm_password')

        if new_password == confirm_password:
            user = User.objects.get(email=email)
            user.set_password(new_password)
            user.save()
            del request.session['reset_email']  # Remove stored email from session
            del request.session['otp']  # Remove stored OTP from session
            return redirect('login')
        else:
            error_message = "Passwords do not match."
            return render(request, 'password/reset_password.html', {'error_message': error_message})

    return render(request, 'password/reset_password.html')

def LogoutPage(request):
    logout(request)
    return redirect('login')

def new_events(request):
    return render(request, 'NewEvents.html')



def create_events(request):
    if request.method == 'POST':
```

```
        form = EventForm(request.POST, request.FILES)
        if form.is_valid():
            event = form.save(commit=False)
            event.username = request.user
            event.event_datetime = timezone.now()  # Set the event datetime with timezone
information
            event.save()
            messages.success(request, "Event created successfully.")
            return redirect('success_page')
        else:
            messages.error(request, "Form validation failed. Please check the form fields.")
    else:
        form = EventForm()

    return render(request, 'create.html', {'form': form})
def success_page(request):
    return render(request, 'success/suucesscreate.html')



def update_event(request, event_id=None):
    if event_id:
        # Handle updating a specific event
        try:
            event = Event.objects.get(pk=event_id)

            if request.method == 'POST':
                form = EventForm(request.POST, instance=event)
                if form.is_valid():
                    form.save()
                    messages.success(request, 'Event details have been successfully updated.')
                    return redirect('success_page')  # Redirect to the success page

        events = Event.objects.filter(username=request.user)
        event_form = EventForm(instance=event)

        return render(request, 'update.html', {'events': events, 'event_form': event_form})
    except Event.DoesNotExist:
        # Handle the case where the event with the given ID does not exist
        messages.error(request, 'Event does not exist.')
```

```
            return redirect('update_event')
        else:
            # Handle displaying all updated events
            events = Event.objects.filter(username=request.user)
            event_form = EventForm()

            return render(request, 'update.html', {'events': events, 'event_form': event_form})


def event_view(request):
    # Get the event with the specified event_id
    events = Event.objects.all()
    return render(request, 'event_view.html', {'events': events})



from django.shortcuts import render, get_object_or_404, redirect
from django.contrib import messages
from .models import Event
from .forms import EventForm
from django.contrib.messages import get_messages

def edit_event(request, event_id):
    event = get_object_or_404(Event, id=event_id)
    form = EventForm(request.POST or None, instance=event)

    if request.method == 'POST':
        try:
            if form.is_valid():
                form.save()

                # Check if changes were made
                if form.has_changed():
                    messages.success(request, 'Event details have been successfully updated.')

                    # Delay the redirection by a few seconds (adjust the delay as needed)
                    return render(request, 'editupdate.html', {'form': form, 'event': event, 'delay_redirect':
True})
                else:
                    messages.info(request, 'No changes were made to the event details.')
```

```
            return redirect('update_event')
        except Exception as e:
            # Handle exceptions here
            messages.error(request, f'An error occurred: {str(e)}')

    return render(request, 'editupdate.html', {'form': form, 'event': event})


def delete_event(request):
    if request.method == 'POST':
        event_id = request.POST.get('event_id')  # Get the event_id from the form
        try:
            event = Event.objects.get(id=event_id)
            if event.username == request.user:  # Check if the event belongs to the logged-in user
                event.delete()
                return redirect('update_event')  # Redirect back to the update_event page
            else:
                # If the event doesn't belong to the user, show an error or redirect to an appropriate page
                pass
        except Event.DoesNotExist:
            pass
    return redirect('update_event')  # Redirect back to the update_event page

def about_us(request):
    return render(request, 'about_us.html')

#registration and payment views here
def registration_details(request,event_id):
    event = get_object_or_404(Event, pk=event_id)
    context = {'event': event}
    return render(request, 'registration.html', context)

from django.shortcuts import render, get_object_or_404
from .models import Event
from django.core.mail import send_mail
from django.conf import settings
import logging

def payment(request, event_id):
```

```
event = get_object_or_404(Event, pk=event_id)
email_sent = 0  # Initialize email_sent with a default value

if request.method == 'POST':
    # Handle the payment processing logic here (you can simulate a successful payment)
    user_email = request.POST.get('user_email')
    entry_fee = event.entry_fee

    # Simulate a successful payment

    # Send a confirmation email
    subject = 'Payment Confirmation'
    message = 'Thank you for your payment. Your payment was successful.'
    from_email = settings.EMAIL_HOST_USER
    recipient_list = [user_email]

    try:
        # Attempt to send the email
        email_sent = send_mail(subject, message, from_email, recipient_list)

        if email_sent == 1:
            # Render the payment success template with the user's email and event details
            payment = Payment.objects.create(
                email=user_email,
                payment_date=timezone.now(),  # Use the current date and time as the payment date
                eventname=event.event_name,
                amount=entry_fee  # Use the entry fee as the payment amount
            )
            payment.save()

            # Render the payment success template with the user's email and event details
            context = {'user_email': user_email, 'event': event}
            return render(request, 'payment/payment_success.html', context)
        else:
            # Log the error
            logging.error(f'Error sending confirmation email to {user_email}')

            # Pass the error message to the template
            error_message = 'There was an issue sending the confirmation email. Please try again.'
```

```
            return render(request, 'payment/payment.html', {'event': event, 'error_message':
error_message})
        except Exception as e:
            logging.error(f'Exception occurred while sending email: {str(e)}')

    # Handle GET requests or invalid submissions
    return render(request, 'payment/payment.html', {'event': event, 'email_sent': email_sent})



def payment_success(request,event_id):
    event = get_object_or_404(Event, pk=event_id)
    context = {'event': event}
    # You can also render a success page or redirect to a specific URL
    return render(request, 'payment/payment_success.html',{'context': context} )

@login_required
def user_registered_events(request, event_id=None):
    # Get the user
    user = request.user

    # Fetch the events registered by the currently logged-in user
    registered_events = Event.objects.filter(payment__email=user.email)

    if event_id is not None:
        # If 'event_id' is provided, get the specific event for the given event_id
        event = get_object_or_404(registered_events, pk=event_id)
        context = {'event': event}
    else:
        # If 'event_id' is not provided, you can provide a list of all registered events
        context = {'registered_events': registered_events}

    # Render the template with the appropriate context
    return render(request, 'registered_events.html', context)
```

Models

```
from django.db import models
from django.contrib.auth.models import User

class Event(models.Model):
```

```
username = models.ForeignKey(User, on_delete=models.CASCADE)
event_name = models.CharField(max_length=100)
college_name = models.CharField(max_length=100)
address = models.CharField(max_length=200)
college_website = models.CharField(max_length=200)
event_datetime = models.DateTimeField(blank=True, null=True)
event_description = models.TextField()
event_brochure = models.FileField(upload_to='event_brochures/', blank=True, null=True)
coordinator_name = models.CharField(max_length=100)
coordinator_contact = models.CharField(max_length=20)
entry_fee = models.DecimalField(max_digits=10, decimal_places=2, default=0.00)
    def _str_(self):
        return self.event_name

class Payment(models.Model):
    email = models.EmailField()
    payment_date = models.DateTimeField(auto_now_add=True)
    eventname = models.CharField(max_length=100)  # Added event_name field
    amount = models.DecimalField(max_digits=10, decimal_places=2, default=0.00)  # Added
amount field

    def _str_(self):
        return self.email
```

Login Page

```
{% load static %}
<!DOCTYPE html>
<html lang="en">
<head>
  <!-- Design by foolishdeveloper.com -->

    <title>Login Page</title>
    {% if error_message %}
    <script>
        alert("{{ error_message }}");
    </script>
    {% endif %}
    <link rel="preconnect" href="https://fonts.gstatic.com">
    <link rel="stylesheet"
```

```
href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.15.4/css/all.min.css">
    <link
href="https://fonts.googleapis.com/css2?family=Poppins:wght@300;500;600&display=swap"
rel="stylesheet">
    <!--Stylesheet-->
    <style media="screen">
      *,
*:before,
*:after{
    padding: 0;
    margin: 0;
    box-sizing: border-box;
}
body{
    background-color: #080710;
}
.background{
    width: 430px;
    height: 520px;
    position: absolute;
    transform: translate(-50%,-50%);
    left: 50%;
    top: 50%;
}
.background .shape{
    height: 200px;
    width: 200px;
    position: absolute;
    border-radius: 50%;
}
.shape:first-child{
    background: linear-gradient(
        #1845ad,
        #23a2f6
    );
    left: -80px;
    top: -80px;
}
.shape:last-child{
    background: linear-gradient(
```

```
        to right,
        #ff512f,
        #f09819
    );
    right: -30px;
    bottom: -80px;
}
form{
    height: 520px;
    width: 400px;
    background-color: rgba(255,255,255,0.13);
    position: absolute;
    transform: translate(-50%,-50%);
    top: 50%;
    left: 50%;
    border-radius: 10px;
    backdrop-filter: blur(10px);
    border: 2px solid rgba(255,255,255,0.1);
    box-shadow: 0 0 40px rgba(8,7,16,0.6);
    padding: 50px 35px;
}
form *{
    font-family: 'Poppins',sans-serif;
    color: #ffffff;
    letter-spacing: 0.5px;
    outline: none;
    border: none;
}
form h3{
    font-size: 32px;
    font-weight: 500;
    line-height: 42px;
    text-align: center;
}

label{
    display: block;
    margin-top: 30px;
    font-size: 16px;
    font-weight: 500;
```

```
}
input{
    display: block;
    height: 50px;
    width: 100%;
    background-color: rgba(255,255,255,0.07);
    border-radius: 3px;
    padding: 0 10px;
    margin-top: 8px;
    font-size: 14px;
    font-weight: 300;
}
::placeholder{
    color: #e5e5e5;
}
button{
    margin-top: 50px;
    width: 100%;
    background-color: #ffffff;
    color: #080710;
    padding: 15px 0;
    font-size: 18px;
    font-weight: 600;
    border-radius: 5px;
    cursor: pointer;
}
.social{
  margin-top: 30px;
  display: flex;
}
.social div{
  background: red;
  width: 150px;
  border-radius: 3px;
  padding: 5px 10px 10px 5px;
  background-color: rgba(255,255,255,0.27);
  color: #eaf0fb;
  text-align: center;
}
.social div:hover{
```

```
  background-color: rgba(255,255,255,0.47);
}
.social .fb{
  margin-left: 25px;
}
.social i{
  margin-right: 4px;
}

    </style>
</head>
<body>
    <div class="background">
        <div class="shape"></div>
        <div class="shape"></div>
    </div>
    <form method="post">
        {% if success_message %}
<div class="alert alert-success" role="alert">
  {{ success_message }}
</div>
{% endif %}

        <h3>Login Here</h3>
        {% csrf_token %}
        <!-- Inside login.html -->

        <label for="username">Username</label>
        <input type="text" placeholder="Enter Username" id="username"  name="username"
Required>

        <label for="password">Password</label>
        <input type="password" placeholder="Password" id="password" name="pass" Required><br>
        <a href="{% url 'forgot_password' %}">forget password</a>
        <button type="submit">Log In</button>
        <a href="{% url 'signup' %}" >Create a account</a>

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

HOME PAGE

```
<!-- Custom CSS styles -->
<style>
  .container {
    background-color: #f8f9fa;
    padding: 20px;
    border-radius: 10px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.2);
    text-align: center; /* Center-align the container content */
  }

.form-group {
  margin-bottom: 20px;
  text-align: left; /* Left-align form elements within the container */
}

label {
  font-weight: bold;
}

.btn-success {
  background-color: #28a745;
  color: #fff;
  border: none;
  border-radius: 5px;
  font-weight: bold;
}

  .btn-success:hover {
    background-color: #218838;
  }

  /* Center-align the form button */
  .btn-success.btn-block {
    margin: 0 auto;
  }
</style>
<div class="container">
  <div class="col-md-8 offset-md-2">
```

```
<h2 class="text-center my-3">Update Event</h2>
<form method="POST" class="text-left" id="updateEventForm">
  {% csrf_token %}
  <!-- College Name Field -->
  <div class="form-group">
    <label for="collegeName">College Name:</label>
    {{ form.college_name }}
  </div>

  <!-- Event Name Field -->
  <div class="form-group">
    <label for="eventName">Event Name:</label>
    {{ form.event_name }}
  </div>

  <!-- Address Field -->
  <div class="form-group">
    <label for="address">Address:</label>
    {{ form.address }}
  </div>

  <!-- College Website Field -->
  <div class="form-group">
    <label for="collegeWebsite">College Website:</label>
    {{ form.college_website }}
  </div>

  <!-- Event Datetime Field -->
  <div class="form-group">
    <label for="eventDatetime">Event Date and Time:</label>
    {{ form.event_datetime }}
  </div>

  <!-- Event Brochure Field -->
  <div class="form-group">
    <label for="eventBrochure">Event Brochure:</label>
    {{ form.event_brochure }}
  </div>

  <!-- Coordinator Name Field -->
```

```
<div class="form-group">
  <label for="coordinatorName">Coordinator Name:</label>
  {{ form.coordinator_name }}
</div>

<!-- Coordinator Contact Field -->
<div class="form-group">
  <label for="coordinatorContact">Coordinator Contact:</label>
  {{ form.coordinator_contact }}
</div>

<!-- Entry Fee Field -->
<div class="form-group">
  <label for="entryFee">Entry Fee:</label>
  {{ form.entry_fee }}
</div>

<!-- Event Description Field -->
<div class="form-group">
  <label for="eventDescription">Event Description:</label>
  {{ form.event_description }}
</div>

<button type="submit" class="btn btn-success btn-block">Save Changes</button>
</form>
{% if delay_redirect %}
<!-- JavaScript code to trigger the redirection after displaying the success message -->
<script>
  setTimeout(function() {
    alert("Event details have been successfully updated.");
    window.location.href = "{% url 'update_event' %}";
  }, 1000);  // Redirect after a 3-second delay (adjust as needed)
</script>
{% endif %}
</div>
</div>

{% endblock %}

{% extends 'home.html' %}
```

{% load static %}

{% block title %}Payment{% endblock %}
{% block video %}{% endblock video %}

{% block content %}

PAYMENT

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

<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Payment Form</title>
<link href="paymentform.css" rel="stylesheet">
<link
href="https://stackpath.bootstrapcdn.com/font-awesome/4.7.0/css/font-awesome.min.css"
rel="stylesheet"

integrity="sha384-wvfXpqpZZVQGK6TAh5PVlGOfQNHSoD2xbE+QkPxCAFlNEevoEH3Sl0sibVcOQ
VnN" crossorigin="anonymous">
<style>
*{
margin: 0;
padding: 0;
box-sizing: border-box;
}
body{
background-color: #f5f5f5;
font-family: Arial, Helvetica, sans-serif;
}
.wrapper{
background-color: #fff;
width: 500px;
padding: 25px;
margin: 25px auto 0;
box-shadow: 0px 0px 20px rgba(0,0,0,0.5);
}

```
.wrapper h2{
  background-color: #fcfcfc;
  color: #7ed321;
  font-size: 24px;
  padding: 10px;
  margin-bottom: 20px;
  text-align: center;
  border: 1px dotted #333;
}
h4{
  padding-bottom: 5px;
  color: #7ed321;
}
.input-group{
  margin-bottom: 8px;
  width: 100%;
  position: relative;
  display: flex;
  flex-direction: row;
  padding: 5px 0;
}
.input-box{
  width: 100%;
  margin-right: 12px;
  position: relative;
}
.input-box:last-child{
  margin-right: 0;
}
.name{
  padding: 14px 10px 14px 50px;
  width: 100%;
  background-color: #fcfcfc;
  border: 1px solid #00000033;
  outline: none;
  letter-spacing: 1px;
  transition: 0.3s;
  border-radius: 3px;
  color: #333;
}
```

```
.name:focus, .dob:focus{
  -webkit-box-shadow:0 0 2px 1px #7ed32180;
  -moz-box-shadow:0 0 2px 1px #7ed32180;
  box-shadow: 0 0 2px 1px #7ed32180;
  border: 1px solid #7ed321;
}
.input-box .icon{
  width: 48px;
  display: flex;
  justify-content: center;
  align-items: center;
  position: absolute;
  top: 0px;
  left: 0px;
  bottom: 0px;
  color: #333;
  background-color: #f1f1f1;
  border-radius: 2px 0 0 2px;
  transition: 0.3s;
  font-size: 20px;
  pointer-events: none;
  border: 1px solid #00000033;
  border-right: none;
}
.name:focus + .icon{
  background-color: #7ed321;
  color: #fff;
  border-right: 1px solid #7ed321;
  border: none;
  transition: 1s;
}
.dob{
  width: 30%;
  padding: 14px;
  text-align: center;
  background-color: #fcfcfc;
  transition: 0.3s;
  outline: none;
  border: 1px solid #c0bfbf;
  border-radius: 3px;
```

```
}
.radio{
  display: none;
}
.input-box label{
  width: 50%;
  padding: 13px;
  background-color: #fcfcfc;
  display: inline-block;
  float: left;
  text-align: center;
  border: 1px solid #c0bfbf;
}
.input-box label:first-of-type{
  border-top-left-radius: 3px;
  border-bottom-left-radius: 3px;
  border-right: none;
}
.input-box label:last-of-type{
  border-top-right-radius: 3px;
  border-bottom-right-radius: 3px;
}
.radio:checked + label{
  background-color: #7ed321;
  color: #fff;
  transition: 0.5s;
}
.input-box select{
  display: inline-block;
  width: 50%;
  padding: 12px;
  background-color: #fcfcfc;
  float: left;
  text-align: center;
  font-size: 16px;
  outline: none;
  border: 1px solid #c0bfbf;
  cursor: pointer;
  transition: all 0.2s ease;
}
```

```
.input-box select:focus{
  background-color: #7ed321;
  color: #fff;
  text-align: center;
}
button{
  width: 100%;
  background: transparent;
  border: none;
  background: #7ed321;
  color: #fff;
  padding: 15px;
  border-radius: 4px;
  font-size: 16px;
  transition: all 0.35s ease;
 }
 button:hover{
  cursor: pointer;
  background: #5eb105;
 }
   </style>
</head>

<body>


   <div class="wrapper">
      <h2>Event Payment</h2>
   <form method="POST">
      {% csrf_token %}
      <h4>Account</h4>
      <div class="input-group">
         <div class="input-box">
            <input type="text" placeholder="Full Name" required class="name">
            <i class="fa fa-user icon"></i>
         </div>
         <div class="input-box">
            <input type="text" placeholder="Nick Name" required class="name">
            <i class="fa fa-user icon"></i>
         </div>
```

```
</div>
<div class="input-group">
   <div class="input-box">
      <input type="email" placeholder="Email Adress" required class="name">
      <i class="fa fa-envelope icon"></i>
   </div>
</div>
<div class="input-group">
   <div class="input-box">
      <h4> Date of Birth</h4>
      <input type="text" placeholder="DD" class="dob">
      <input type="text" placeholder="MM" class="dob">
      <input type="text" placeholder="YYYY" class="dob">
   </div>
   <div class="input-box">
      <h4> Gender</h4>
      <input type="radio" id="b1" name="gendar" checked class="radio">
      <label for="b1">Male</label>
      <input type="radio" id="b2" name="gendar" class="radio">
      <label for="b2">Female</label>
   </div>
</div>
<div class="input-group">
   <div class="input-box">
      <h4>Payment Details</h4>
      <input type="radio" name="pay" id="bc1" checked class="radio">
      <label for="bc1"><span><i class="fa fa-cc-visa"></i>Credit Card</span></label>
      <input type="radio" name="pay" id="bc2" class="radio">
      <label for="bc2"><span><i class="fa fa-cc-paypal"></i> Paypal</span></label>
   </div>
</div>
<div class="input-group">
   <div class="input-box">
      <input type="tel" placeholder="Card Number" required class="name">
      <i class="fa fa-credit-card icon"></i>
   </div>
</div>
<div class="input-group">
   <div class="input-box">
      <input type="tel" placeholder="Card CVV" required class="name">
```

```
            <i class="fa fa-user icon"></i>
          </div>
          <div class="input-box">
            <input type="date" class="form-control" id="eventDateTime" name="event_datetime"
required>
          </div>
          <div>
            <div class="input-group">
              <div class="input-box">
                <i class="fa fa-inr" aria-hidden="true"></i>
                <input type="number" required class="name" value="{{ form.entry_fee.value }}">
              </div>
            </div>
          </div>
```

CHAPTER 6

TEST CASES

The aim of the system testing process was to determine all defects in our project. The programwas subjected to a set of test inputs and various observations that were made and based on these observations it will be decided whether the program behaves as expected or not.

Verified Test Cases

User Registration:

Verify that a new user can successfully register with valid information.

Check that the system displays an error message when a user attempts to register with an already registered email address.

Verify that the system enforces password requirements (e.g., minimum length, special characters).

Check that the system sends a confirmation email after successful registration.

User Authentication:

Verify that a registered user can successfully log in with correct credentials.

Check that the system displays an error message when a user enters incorrect login credentials.

Verify that the "Forgot Password" functionality allows users to reset their passwords through email verification.

Event Creation and Management:

Verify that an event organizer can successfully create a new event with valid information.

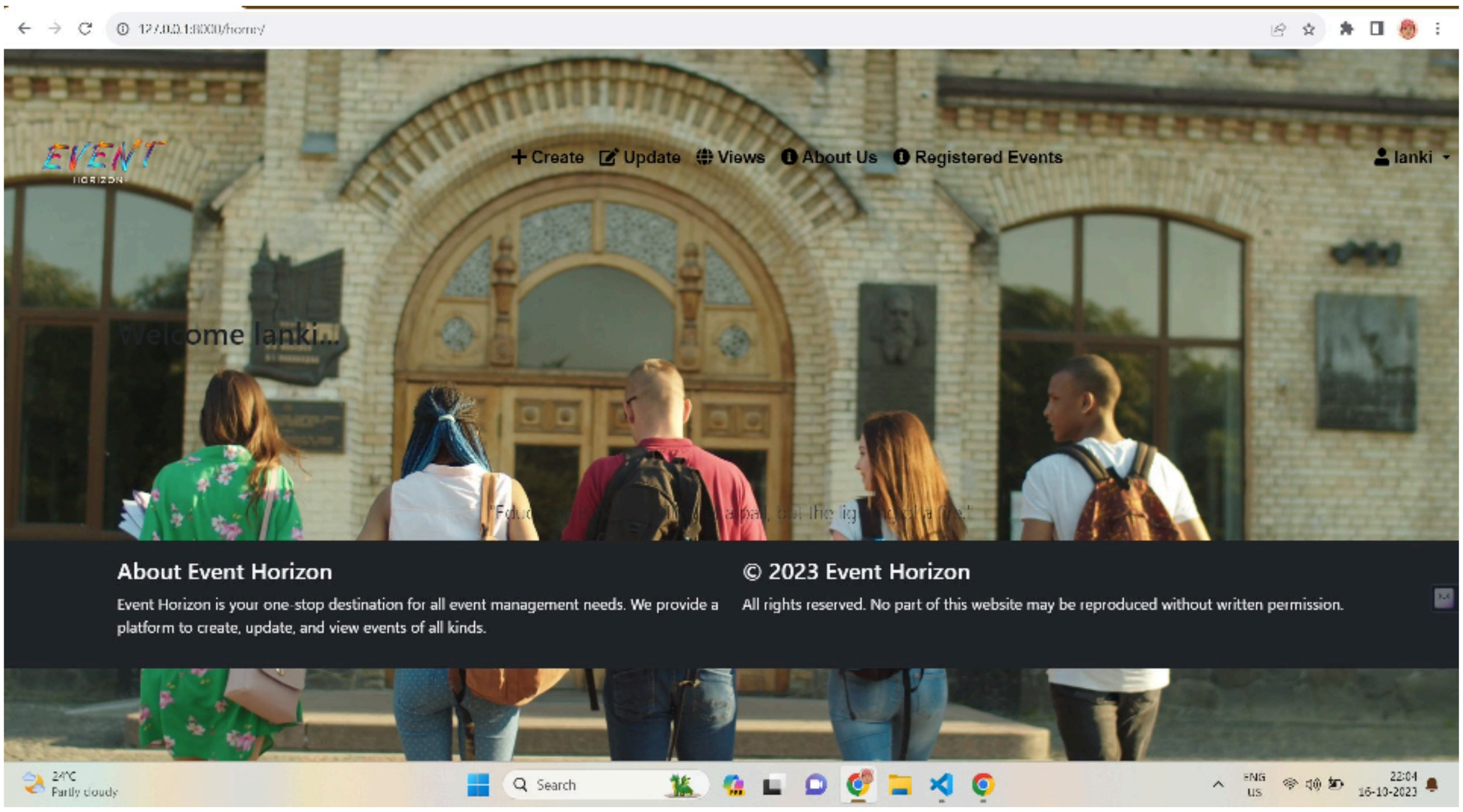
Check that the system displays an error message when an event organizer attempts to create an event with missing or invalid data.

Verify that an organizer can edit event details and save changes.

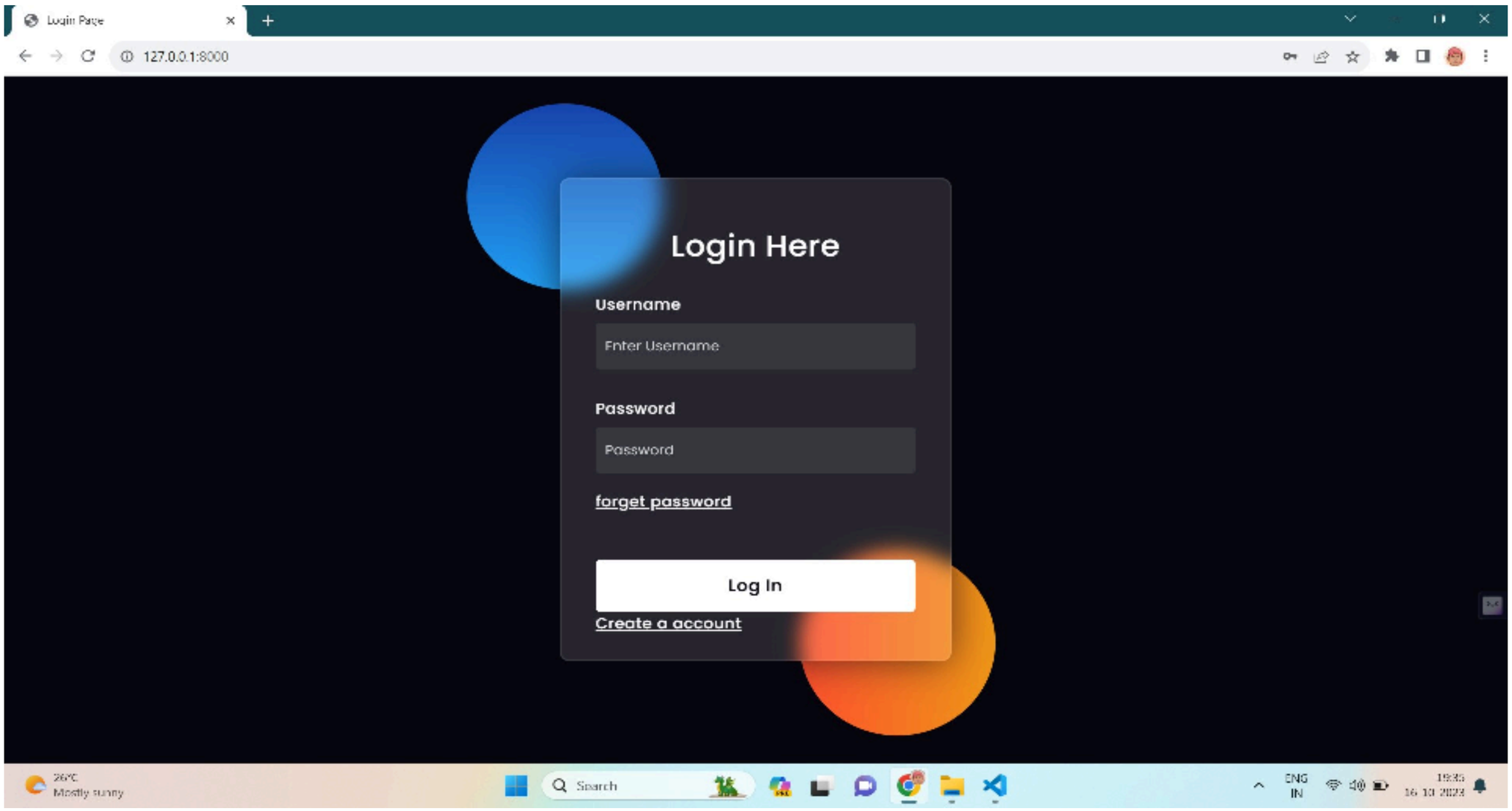
Check that an organizer can delete an event, and the event is no longer visible.

MODULES

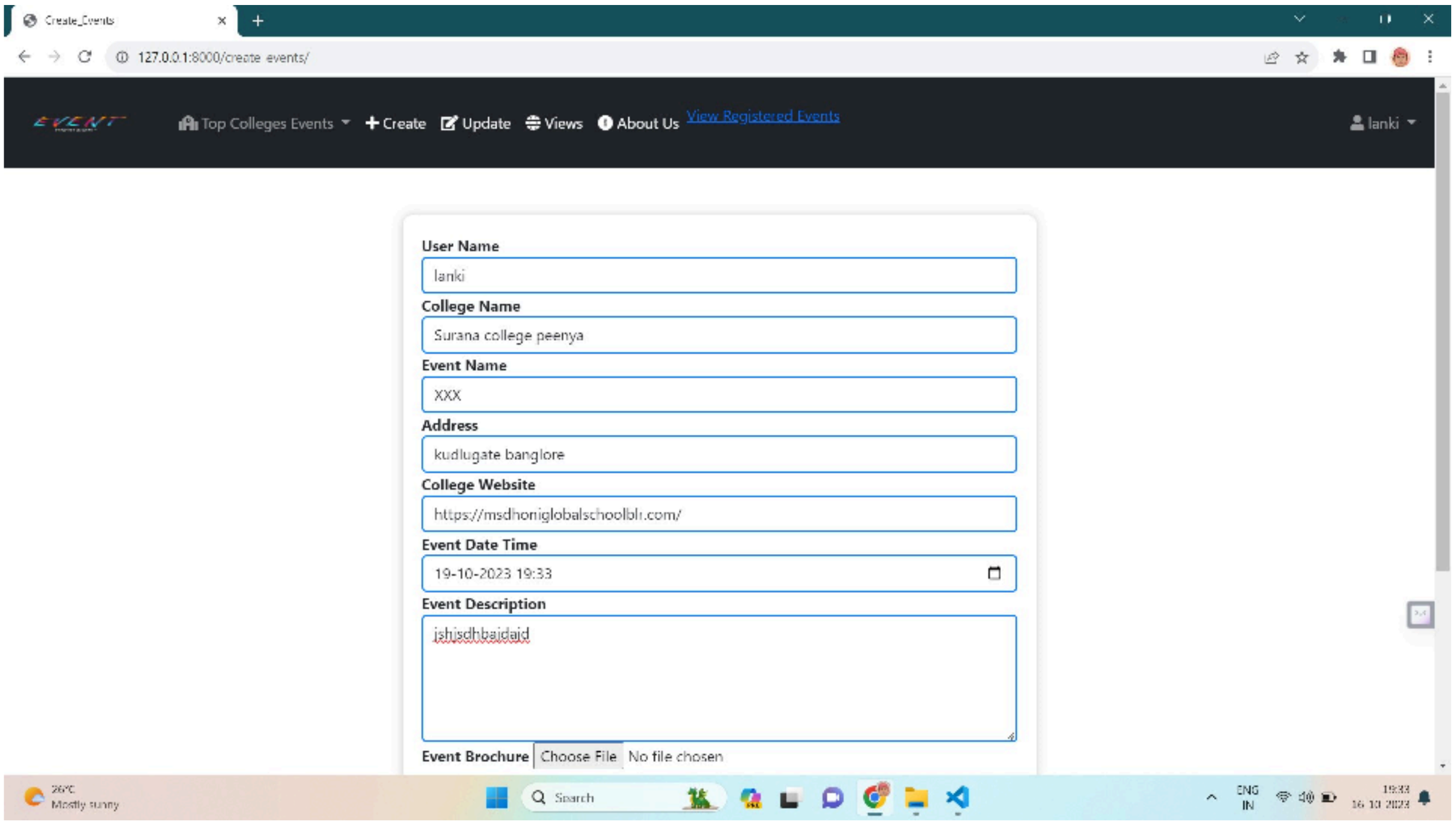
HOME PAGE



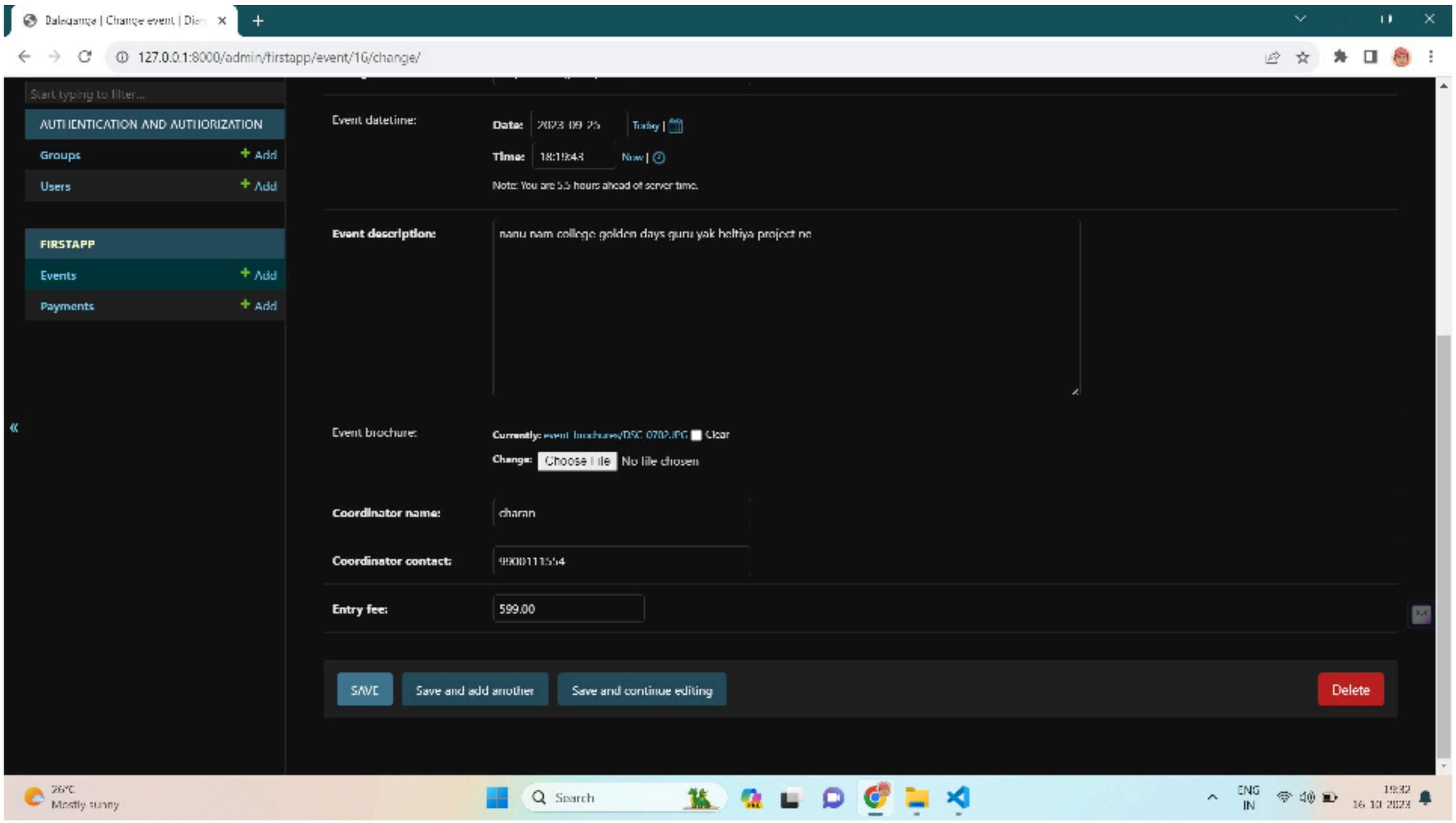
LOGIN FORM


Login Page
127.0.0.1:8000
Login Here
Username
Enter Username
Password
Password
forget password
Log In
Create a account
Search

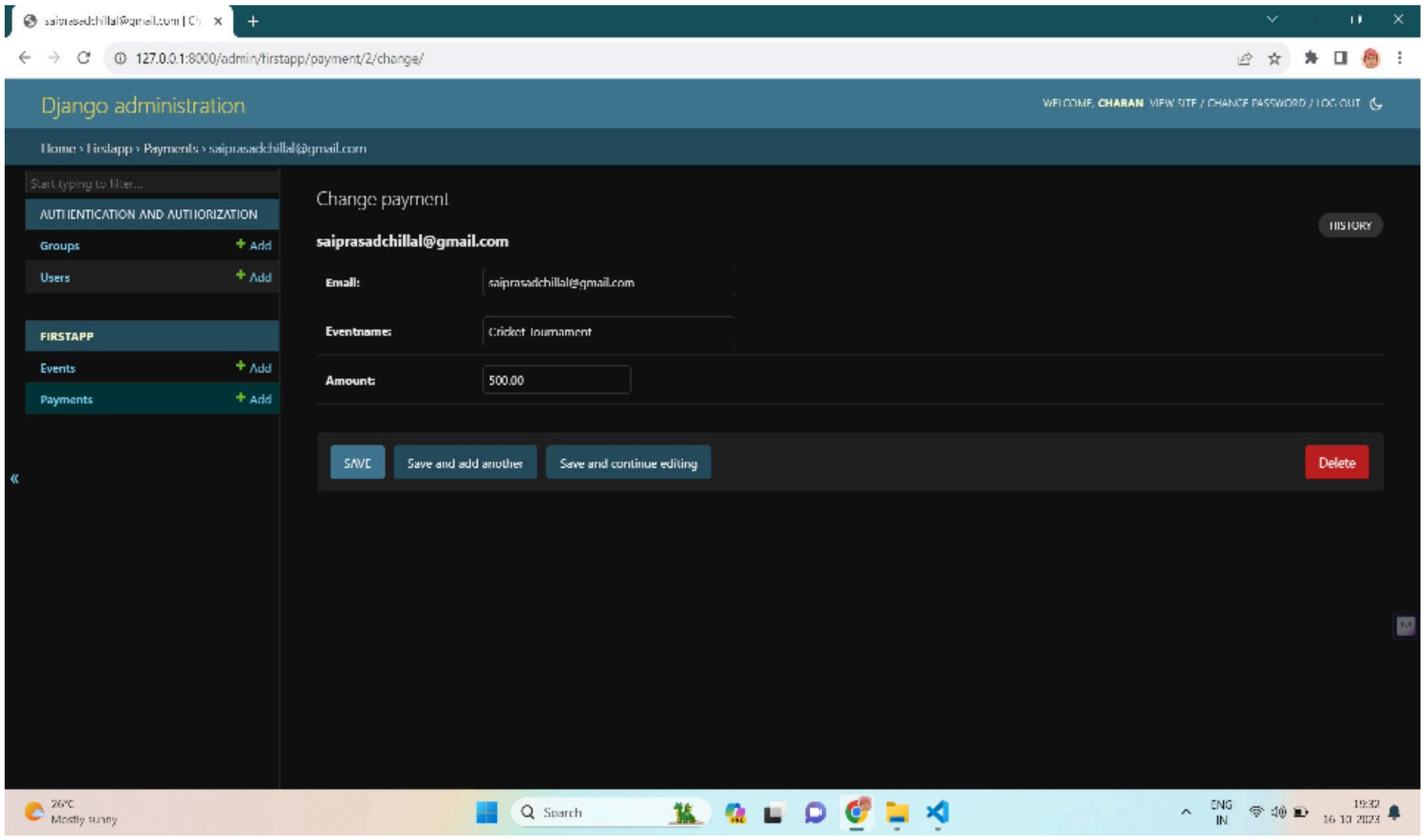
ADMIN EVENT CREATION FORM



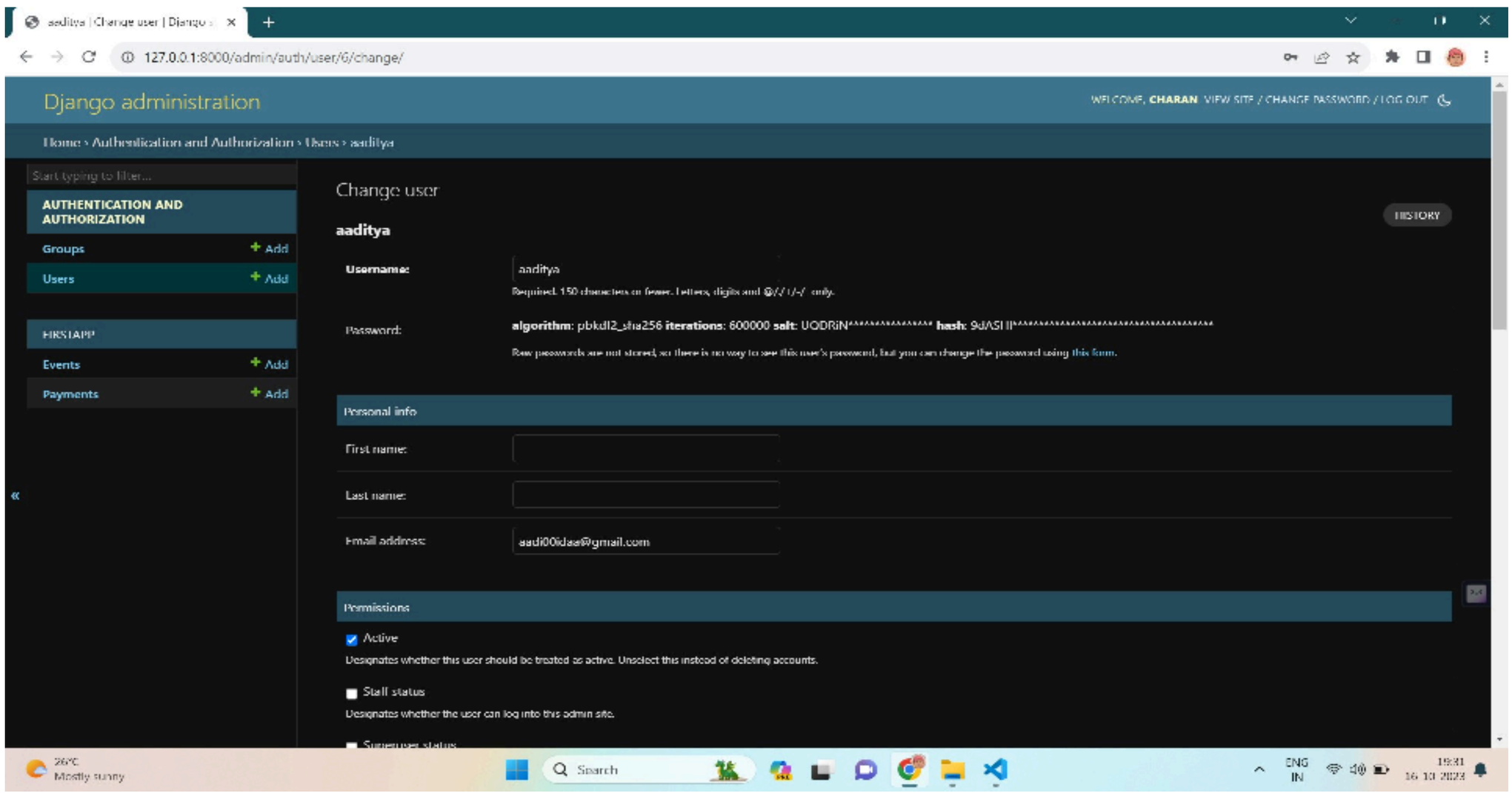
EVENT DESCRIPTION PAGE


127.0.0.1:8000/admin/firstapp/event/16/change/
AUTHENTICATION AND AUTHORIZATION
Groups + Add
Users + Add
FIRSTAPP
Events + Add
Payments + Add
Event datetime:
Date:
Today |
Time:
Now |
Note: You are 5.5 hours ahead of server time.
Event description:
nanu nam college golden days guru yak heltiya project ne
Event brochure:
Currently: event_brochures/DSC_0707.JPG
Clear
Change:
Choose File
No file chosen
Coordinator name:
charan
Coordinator contact:
Entry fee:
599.00
SAVE
Save and add another
Save and continue editing
Delete
Search
ENG
IN

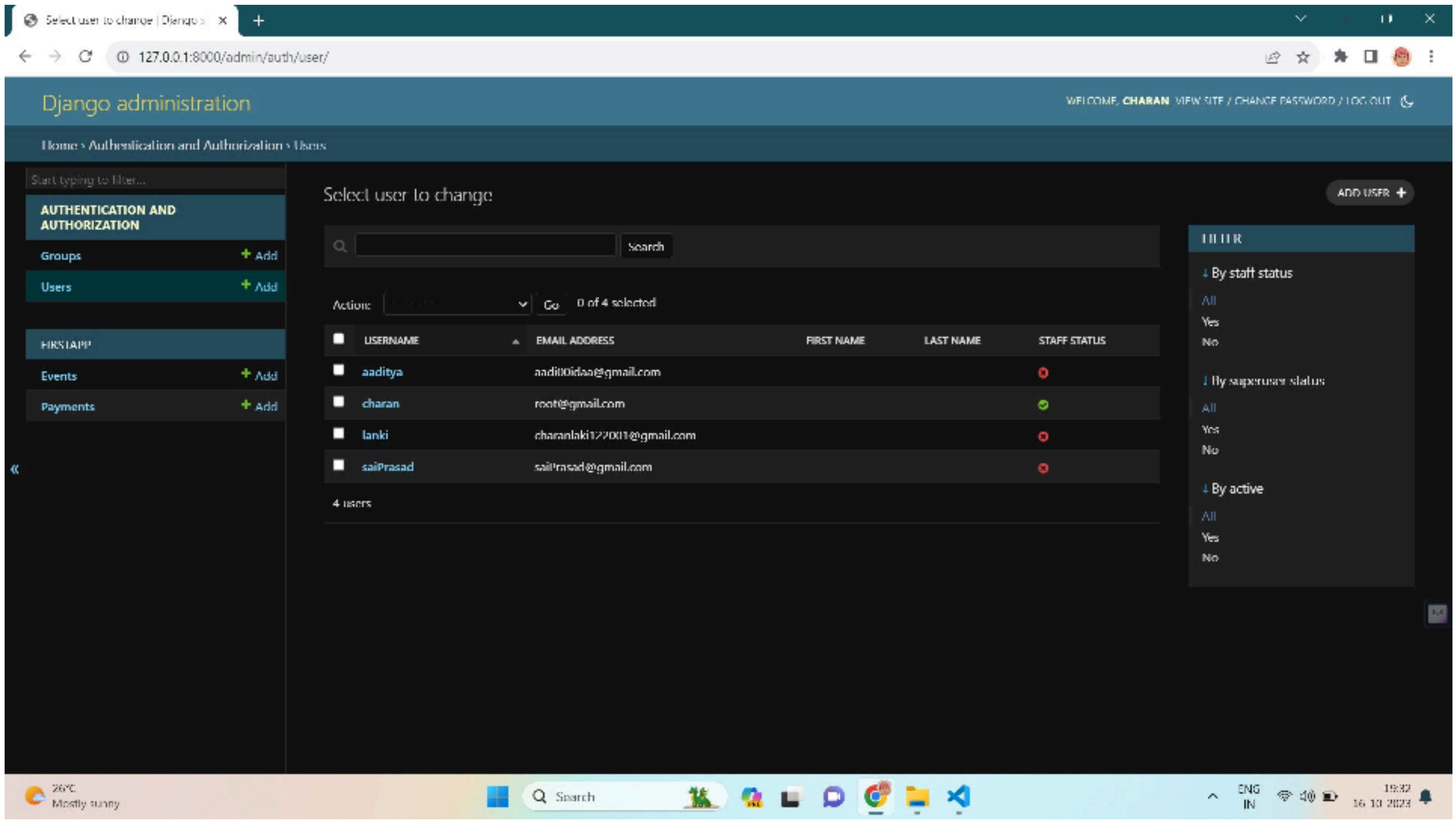
PAYMENT INTERFACE



USER INTERFACE



STORED USER DATABASE



CONCLUSION AND FUTURE SCOPE

In conclusion, the development of the Event Management System using Django, DBMS, and front-end technologies such as CSS and JavaScript has successfully met its primary aim of simplifying event planning, organization, and management. The system provides a user-friendly interface for both event organizers and attendees, offering features like event creation, registration, ticketing, notifications, and feedback. It incorporates security measures, compliance with data protection regulations, and performance optimization to ensure a reliable and secure user experience.

Future Enhancements:

Social Media Integration: Implement social media integration to allow users to sign in with their social media accounts, share events, and gather user data for better targeting.

Geolocation Services: Integrate geolocation services to provide users with event recommendations based on their location and preferences.

Live Streaming and Virtual Events: Enable event organizers to host virtual events with live streaming capabilities, expanding the system's scope to accommodate hybrid and online events.

Multi-Language Support: Offer multi-language support to attract a diverse

user base and support events in different regions.

Offline Mode: Develop an offline mode for users to access event information and tickets when they have limited or no internet connectivity.

Event Calendar Integration: Integrate with popular calendar applications like Google Calendar and Apple Calendar to enable users to add events directly to their schedules.

REFERENCES

https://www.udemy.com/share/107tY03@WbhJ-Kiu9jHgCKr4ED-AzNRjfGkcmFgPSX2O66kWT-SprrmnDldO_cmmwWtRGv0wCg==/

https://www.djangoproject.com/

https://stackoverflow.com/

https://www.w3schools.com/django/