23CS11E - WEBFRAMEWORK USING PYTHON

A

Micro Project

Report On

Event Management System

Submitted by

IMMANUEL A 2312020

In partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

NATIONAL ENGINEERING COLLEGE
(An Autonomous Institution affiliated to Anna University, Chennai)

K.R.NAGAR, KOVILPATTI - 628503

NOVEMBER- 2025

TABLE OF CONTENTS

ABSTRACT

Universities frequently struggle with manual event management, which relies on paper-based registrations and inefficient attendance tracking. This outdated process often results in poor communication, increased administrative overhead, and low student participation.

This project introduces "EventHive," a centralized web application designed to solve these challenges. Built with the **Flask** framework in Python, EventHive provides a comprehensive platform for managing all university events, streamlining the entire process for students, organizers, and administrators.

The system is designed with three distinct user roles. **Students** can easily browse upcoming events, register online, and receive a unique QR code for their registration. **Organizers** are empowered to create, edit, and manage their own events. A key feature is the integrated QR scanner, allowing organizers to take attendance digitally, eliminating paper-based methods. **Admins** have full oversight of the system, with access to user management, event analytics, and post-event feedback data.

Key technologies include Flask for the backend logic, **SQLAlchemy** for database management, and Python's qrcode library for the secure attendance system. By digitizing the event lifecycle, EventHive significantly reduces manual work and enhances student engagement across campus.

CHAPTER 1

INTRODUCTION

Universities often struggle with outdated, manual event management, relying on paper registrations and clipboard attendance. This inefficient system creates significant administrative work, leads to poor data quality, and results in a frustrating experience for students, which ultimately lowers participation in campus events.

EventHive directly solves this by providing a modern, centralized web application built with the Flask framework. It streamlines the entire event lifecycle by providing specific roles for Students (to register), Organizers (to create events), and Admins (to oversee data). By integrating key features like secure online registration, automated QR code attendance, and a built-in feedback system, EventHive replaces manual labor with an efficient, scalable solution designed to boost student engagement.

This integrated system moves beyond mere convenience to generate valuable, real-time data. The application's architecture, built on Flask Blueprints and an SQLAlchemy database, provides a secure and reliable experience. Role-based access, managed by Flask-Login, ensures students see a personalized dashboard with their QR codes, while organizers can access their specific event rosters and feedback

1.1 FLASK FEATURES

- Routing & Blueprints: Used Flask's routing to map URLs and Blueprints to organize the project into logical modules (like auth, events, and dashboards).
- Jinja2 Templating: Powered the frontend by dynamically rendering HTML, allowing pages to display user-specific data (like names and event lists) based on their role.
- Flask-Login: Managed the entire user authentication lifecycle, handling logins, logouts, sessions, and securing pages with the @login_required decorator.
- Flask-SQLAlchemy: Handled all database operations, defining the User, Event, and Feedback tables as Python models and running all queries.
- Flask-WTF: Secured the application by providing CSRF protection and data validation for all user-input forms (like login, registration, and feedback).

1.2 OBJECTIVE

The main objective of the EventHive project is to design, develop, and deploy a comprehensive, web-based platform that replaces traditional, manual event management processes within a university. The goal is to create a single, centralized, and user-friendly system for all stakeholders—students, event organizers, and administrators—to manage the entire lifecycle of an event, from creation and registration to attendance tracking and post-event analysis.

This project aims to significantly reduce administrative overhead, eliminate paper-based inefficiencies, improve data accuracy, and ultimately enhance overall student engagement and participation in campus activities.

Primary Objective: The primary objective of EventHive is to **streamline and automate the entire event registration and attendance tracking process** through a centralized, digital-first platform.

Specific Objectives:

- **To implement a secure, role-based authentication system** that provides distinct functionalities and access levels for three user roles: **Student**, **Organizer**, and **Admin**.
- **To provide Organizers and Admins with full CRUD** (Create, Read, Update, Delete) capabilities to easily create, schedule, and manage all event details.
- **To develop a user-friendly interface for Students** to browse all upcoming events, view detailed information, and register or unregister with a single click.
- **To design and implement an automated attendance system** by generating a unique QR code for each student registration and providing Organizers with a web-based scanner to verify attendance digitally.
- **To create distinct dashboards for each user role** that display relevant, at-a-glance information, such as registered events for students, created events for organizers, and system-wide analytics for admins.
- **To develop a post-event feedback collection system**, allowing students to submit ratings and comments for events they have attended, and for organizers to view this feedback.
- **To provide administrators with visual data analytics**, such as a chart for event registration counts, to monitor engagement and make informed decisions.

1.3 PROBLEM DEFINITION

In the contemporary university environment, student engagement is a key metric for success, and campus events are the primary driver of this engagement. However, the operational management of these events—ranging from academic seminars and technical workshops to cultural festivals—remains a significant institutional challenge.

The core problem is the widespread reliance on **outdated, manual, and fragmented processes**. Event creation is often advertised through disconnected channels like physical posters or siloed department emails. Registration typically involves paper-based sign-up sheets or basic email lists, which are inefficient to manage and prone to human error.

Furthermore, attendance tracking, one of the most critical metrics, is frequently conducted with paper lists and clipboards at the event entrance. This manual check-in process is slow, creates long queues, and provides no reliable data for post-event analysis.

Impact:

- Universal For Students: The process is inefficient and frustrating. It is difficult to discover new events, and the registration process is a high-friction barrier. This confusion and effort lead directly to low student participation and missed learning opportunities.

- For Event Organizers: The administrative overhead is immense. Staff and student volunteers must dedicate hours to manually transcribing names, sending reminders, and later attempting to reconcile registration lists with attendance sheets. This process is not only time-consuming but also highly susceptible to errors, such as lost data and inaccurate counts.

- For University Administration: There is no centralized data or "single source of truth." It is nearly impossible to measure student engagement, track which events are successful, or calculate the return on investment (ROI) for event expenditures. This lack of data hinders effective planning and resource allocation.

Solution Gap:

1. Centralize event discovery and promotion.
2. Provide instant, one-click online registration for students.
3. Automate attendance tracking using modern technology (like QR codes).
4. Aggregate data for analytics and feedback.

CHAPTER 2

IMPLEMENTATION

EventHive was developed using the Flask framework, with Blueprints organizing the application into auth, event, and dashboard components. The database is managed by **Flask-SQLAlchemy**, while **Flask-Login** and **Flask-WTF** handle user sessions and form security.

The core attendance feature uses Python's qrcode library to generate unique QR codes for students. Organizers then use a web-based scanner, built with the html5-qrcode JavaScript library, to capture the code. This data is sent to a secure Flask backend route for instant, real-time attendance validation. The entire application is built with a responsive **Bootstrap 5** frontend.

2.1PROJECT STRUCTURE

```
eventhive/
│
├── app.py              # Main Flask application (creates app, registers blueprints)
├── config.py           # Configuration settings (Secret Key, DB URI)
├── forms.py            # All WTForms classes (Register, Login, Event, Feedback)
├── requirements.txt    # All Python packages (Flask, SQLAlchemy, qrcode, etc.)
├── .flaskenv           # Environment variables (FLASK_APP, FLASK_ENV)
│
├── app.db              # Your SQLite database file (auto-generated)
├── migrations/         # Flask-Migrate folder (auto-generated)
│
├── models/
│   ├── __init__.py
│   └── models.py       # Database tables (User, Event, Feedback, registrations)
│
├── routes/
│   ├── __init__.py
│   ├── auth.py         # Login, register, logout routes
│   ├── dashboard.py    # Admin, organizer, and student dashboard routes
│   ├── events.py       # Event list, create, edit, delete, register routes
│   └── qr.py           # QR code scanner and verification routes
│
├── static/
│   ├── images/
│   │   └── eventhive.png   # Your website favicon/logo
│   └── qr_codes/       # Folder where QR codes are saved (auto-generated)
│
├── templates/
```

```
│   ├── base.html         # The main layout (navbar, footer)
│   ├── index.html        # Homepage
│   │
│   ├── login.html        # Login page
│   ├── register.html     # Registration page
│   │
│   ├── admin_dashboard.html
│   ├── organizer_dashboard.html
│   ├── student_dashboard.html
│   │
│   ├── events_list.html  # Page to browse all events
│   ├── create_event.html # Form to create a new event
│   ├── edit_event.html   # Form to edit an existing event
│   │
│   ├── scan.html         # The QR code scanner page for organizers
│   └── submit_feedback.html # Form for students to leave feedback
│
└── utils/
    ├── decorators.py     # The @role_required decorator
    └── qr_utils.py       # The generate_qr_code() helper function
```

2.1 IMPLEMENTATION STEPS

Step 1: Foundation & Authentication (Flask-Login, Flask-WTF)

- **Roles:** 3 user roles (Admin, Organizer, Student).
- **Secure Forms:** Login & Registration (with role selection).
- **Admin Creation:** Secure CLI command (flask create-admin).

Step 2: Database & Event Management (SQLAlchemy, Flask-Migrate)

- **4 Main Tables:** User, Event, Feedback, registrations (association).
- **Relationships:** User <-> Event (M:M via registrations), Event <-> Feedback (1:M).
- **Event CRUD:** Organizers/Admins can Create, Read, Update, & Delete events.

Step 3: Student Registration System (SQLAlchemy)

- **Browse:** Students can view all upcoming events in a list.
- **Register/Unregister:** Simple button-click logic to join or leave an event.
- **Dashboard:** Students see a list of *only* their registered events.

Step 4: QR Code Attendance System (qrcode + html5-qrcode)

- **Generation:** Auto-generate a unique QR code PNG upon student registration.
- **Verification:** Organizer-only scanner page (/qr/scan) uses device camera.
- **Validation:** Backend route validates the QR data and marks the student as attended.

Step 5: Dashboards & Analytics (Bootstrap 5 + Chart.js)

- **3 Dashboards:** Custom UIs for Admin, Organizer, and Student.
- **Analytics:** Admin dashboard shows a Chart.js bar chart of registrations per event.
- **Responsive Design:** Bootstrap 5 ensures a mobile-first UI for students' QR codes.

Step 6: Feedback & Security (Flask-WTF + Custom Decorators)

- **Feedback:** Students can submit feedback *only* for past events.
- **Security:** CSRF protection on all forms and role-based route protection (@role_required).
- **Event Logic:** Organizers view feedback; students see "Finished" vs "Upcoming" status.

app.py

```
# eventhive/app.py

from flask import Flask
from config import Config
from models.models import db
from flask_migrate import Migrate
from flask_login import LoginManager

# Create and configure the app
app = Flask(__name__)
app.config.from_object(Config)

# Initialize database and migration engine
db.init_app(app)
migrate = Migrate(app, db)

# Initialize Flask-Login
login_manager = LoginManager()
login_manager.init_app(app)
login_manager.login_view = 'auth.login' # Redirect to login page if user is not authenticated

# This function is required by Flask-Login to load a user
from models.models import User
@login_manager.user_loader
def load_user(user_id):
    return User.query.get(int(user_id))

# Import and register blueprints
from routes.events import events_bp
from routes.auth import auth_bp
from routes.dashboard import dashboard_bp
# from routes.dashboard import dashboard_bp # We'll create these later
from routes.qr import qr_bp
```

```
    app.register_blueprint(events_bp)
    app.register_blueprint(auth_bp, url_prefix='/auth')
    app.register_blueprint(dashboard_bp)
    # app.register_blueprint(dashboard_bp, url_prefix='/dashboard')
    app.register_blueprint(qr_bp, url_prefix='/qr')

    @app.cli.command("create-admin")
    def create_admin():
        """Creates a new admin user."""
        import getpass
        username = input("Enter admin username: ")
        email = input("Enter admin email: ")
        password = getpass.getpass("Enter admin password: ")

        # Check if user already exists
        if User.query.filter_by(email=email).first() or User.query.filter_by(username=username).first():
            print("Error: A user with that email or username already exists.")
            return

        admin = User(username=username, email=email, role='Admin')
        admin.set_password(password)
        db.session.add(admin)
        db.session.commit()
        print(f'Admin user {username} created successfully!')
    if __name__ == '__main__':
        app.run(debug=True)
```

config.py

```
 # eventhive/config.py

import os
from dotenv import load_dotenv

# Find the absolute path of the root directory
basedir = os.path.abspath(os.path.dirname(__file__))
load_dotenv(os.path.join(basedir, '.flaskenv'))

class Config:
    """Base configuration settings."""
    SECRET_KEY = os.environ.get('SECRET_KEY') or 'you-will-never-guess'

    # Database configuration
    SQLALCHEMY_DATABASE_URI = os.environ.get('DATABASE_URL') or \
        'sqlite:///' + os.path.join(basedir, 'app.db')
    SQLALCHEMY_TRACK_MODIFICATIONS = False
```

forms.py

```
# eventhive/forms.py

from flask_wtf import FlaskForm
from wtforms import StringField, PasswordField, SubmitField, SelectField, BooleanField,
TextAreaField, RadioField
from wtforms.fields import DateTimeLocalField   # ✅ Modern DateTimeLocalField for browser
compatibility
from wtforms.validators import DataRequired, Email, EqualTo, ValidationError
from models.models import User

# --------------------------- Registration Form --------------------------- #
class RegistrationForm(FlaskForm):
    """
    Form for users to create a new account.
    """
    username = StringField('Username', validators=[DataRequired()])
    email = StringField('Email', validators=[DataRequired(), Email()])
    password = PasswordField('Password', validators=[DataRequired()])
    confirm_password = PasswordField('Confirm Password', validators=[DataRequired(),
EqualTo('password')])

    # Role dropdown
    role = SelectField('Register as',
                       choices=[('Student', 'Student'), ('Organizer', 'Organizer')],
                       validators=[DataRequired()])

    submit = SubmitField('Sign Up')

    # Validate unique username
    def validate_username(self, username):
        user = User.query.filter_by(username=username.data).first()
        if user:
            raise ValidationError('That username is already taken. Please choose a different one.')

    # Validate unique email
    def validate_email(self, email):
        user = User.query.filter_by(email=email.data).first()
        if user:
            raise ValidationError('That email is already registered. Please choose a different one.')

# --------------------------- Login Form --------------------------- #
class LoginForm(FlaskForm):
    """
    Form for users to login.
    """
    email = StringField('Email', validators=[DataRequired(), Email()])
```

```
    password = PasswordField('Password', validators=[DataRequired()])
    remember_me = BooleanField('Remember Me')
    submit = SubmitField('Login')

# -------------------------- Event Form -------------------------- #
class EventForm(FlaskForm):
    """
    Form for users to create a new event.
    """
    title = StringField('Event Title', validators=[DataRequired()])
    description = TextAreaField('Description', validators=[DataRequired()])

    # ✅ Updated field: DateTimeLocalField for modern HTML5 datetime input
    event_date = DateTimeLocalField('Event Date and Time',
                                    format='%Y-%m-%dT%H:%M',
                                    validators=[DataRequired()])

    location = StringField('Location', validators=[DataRequired()])
    submit = SubmitField('Create Event')

# -------------------------- Feedback Form -------------------------- #
class FeedbackForm(FlaskForm):
    """
    Form for students to submit event feedback.
    """
    rating = RadioField('Rating',
                        choices=[('5', 'Excellent'),
                                 ('4', 'Good'),
                                 ('3', 'Average'),
                                 ('2', 'Poor'),
                                 ('1', 'Terrible')],
                        validators=[DataRequired()])
    comment = TextAreaField('Comment (Optional)', render_kw={'rows': 5})
    submit = SubmitField('Submit Feedback')
```

auth.py

```
from flask import Blueprint, render_template, flash, redirect, url_for, request
from flask_login import login_user, logout_user, current_user
from forms import LoginForm, RegistrationForm
from models.models import db, User

# Create a Blueprint
auth_bp = Blueprint('auth', __name__)

# -------------------------- LOGIN -------------------------- #
```

```
@auth_bp.route('/login', methods=['GET', 'POST'])
def login():
    """Handles user login."""
    # If user is already logged in, redirect to homepage
    if current_user.is_authenticated:
        return redirect(url_for('events.index'))

    form = LoginForm()
    if form.validate_on_submit():
        # Find the user by email
        user = User.query.filter_by(email=form.email.data).first()

        # Verify credentials
        if user and user.check_password(form.password.data):
            login_user(user, remember=form.remember_me.data)
            flash('Logged in successfully!', 'success')

            # Redirect to the intended page or homepage
            next_page = request.args.get('next')
            return redirect(next_page) if next_page else redirect(url_for('events.index'))
        else:
            flash('Login Unsuccessful. Please check email and password.', 'danger')

    return render_template('login.html', title='Login', form=form)

# ---------------------------- LOGOUT ---------------------------- #
@auth_bp.route('/logout')
def logout():
    """Handles user logout."""
    logout_user()
    return redirect(url_for('events.index'))

@auth_bp.route('/register', methods=['GET', 'POST'])
def register():
    """Handles user registration."""
    if current_user.is_authenticated:
        return redirect(url_for('events.index'))

    form = RegistrationForm()
    if form.validate_on_submit():
        # ✅ Updated: include the role from the dropdown
        user = User(
            username=form.username.data,
            email=form.email.data,
            role=form.role.data  # Save selected role (Student/Organizer)
        )

        user.set_password(form.password.data)
```

```
        db.session.add(user)
        db.session.commit()

        flash('🎉 Registration successful! You can now log in.', 'success')
        return redirect(url_for('auth.login'))

    return render_template('register.html', title='Register', form=form)
```

dashboard.py

```
# eventhive/routes/dashboard.py

from flask import Blueprint, render_template
from flask_login import login_required, current_user
from utils.decorators import role_required
from models.models import User, Event, db, registrations, Feedback
from sqlalchemy import func
from datetime import datetime

# ---------------------------- Blueprint ---------------------------- #
dashboard_bp = Blueprint('dashboard', __name__)

# ---------------------------- ADMIN DASHBOARD ---------------------------- #
@dashboard_bp.route('/admin_dashboard')
@login_required
@role_required('Admin')
def admin_dashboard():
    # Keep existing queries
    users = User.query.all()
    events = Event.query.order_by(Event.date_posted.desc()).all()

    # --- THIS IS THE NEW, MORE EXPLICIT QUERY ---
    event_analytics = db.session.query(
        Event.title,
        func.count(registrations.c.user_id) # Count user IDs from the registration table
    ).select_from(Event).join(
        registrations,
        registrations.c.event_id == Event.id, # Explicitly state the join
        isouter=True # This makes it a LEFT JOIN (to include events with 0)
    ).group_by(Event.id, Event.title).all()

    print(f"DEBUG: Chart data is: {event_analytics}")

    # Prepare data for Chart.js
    chart_labels = [event[0] for event in event_analytics]
    chart_data = [event[1] for event in event_analytics]
    # ----------------------------------
```

```
    return render_template('admin_dashboard.html',
                           title='Admin Dashboard',
                           users=users,
                           events=events,
                           chart_labels=chart_labels,
                           chart_data=chart_data)

# -------------------------- ORGANIZER DASHBOARD -------------------------- #
@dashboard_bp.route('/organizer_dashboard')
@login_required
@role_required('Organizer')
def organizer_dashboard():
    """Organizer dashboard displaying events created by the current organizer."""
    events = (
        Event.query.filter_by(organizer_id=current_user.id)
        .order_by(Event.event_date.desc())
        .all()
    )

    # Prepare attendee + feedback data for each event
    event_data = []
    for event in events:
        # Registered attendees
        attendees = (
            db.session.query(User, registrations.c.attended)
            .join(registrations, User.id == registrations.c.user_id)
            .filter(registrations.c.event_id == event.id)
            .all()
        )

        # Feedback for this event
        feedbacks = Feedback.query.filter_by(event_id=event.id).all()

        # Combine all event data
        event_data.append({
            'event': event,
            'attendees': attendees,
            'feedbacks': feedbacks
        })

    return render_template(
        'organizer_dashboard.html',
        title='Organizer Dashboard',
        event_data=event_data
    )

# -------------------------- STUDENT DASHBOARD -------------------------- #
```

```
@dashboard_bp.route('/student_dashboard')
@login_required
@role_required('Student')
def student_dashboard():
    """Student dashboard showing registered events."""
    registered_events = current_user.registered_events.order_by(Event.event_date.asc()).all()

    return render_template(
        'student_dashboard.html',
        title='My Dashboard',
        events=registered_events,
        datetime=datetime  # Pass datetime to template for comparisons
    )
```

Events.py

```
from flask import Blueprint, render_template, redirect, url_for, flash, current_app, abort
from flask_login import login_required, current_user
from models.models import db, Event,Feedback
from forms import EventForm,FeedbackForm
from datetime import datetime
from utils.decorators import role_required
from utils.qr_utils import generate_qr_code
from utils.decorators import role_required
# Create a Blueprint
events_bp = Blueprint('events', __name__)

# ------------------------- HOMEPAGE -------------------------
@events_bp.route('/')
@events_bp.route('/index')
def index():
    """Renders the homepage with a few upcoming events."""
    events = Event.query.order_by(Event.event_date.asc()).limit(3).all()
    return render_template('index.html', title='Welcome', events=events)

# ------------------------- EVENT LIST -------------------------
@events_bp.route('/events')
def events_list():
    """Renders the full list of events."""
    all_events = Event.query.order_by(Event.event_date.asc()).all()
    return render_template('events_list.html', title='Upcoming Events', events=all_events)

# ------------------------- CREATE EVENT -------------------------
@events_bp.route('/create_event', methods=['GET', 'POST'])
@login_required
@role_required('Admin', 'Organizer')
def create_event():
```

```
    """Handles event creation."""
    form = EventForm()
    if form.validate_on_submit():
        event = Event(
            title=form.title.data,
            description=form.description.data,
            event_date=form.event_date.data,
            location=form.location.data,
            organizer_id=current_user.id
        )
        db.session.add(event)
        db.session.commit()
        flash('Your event has been created!', 'success')
        return redirect(url_for('events.events_list'))

    return render_template('create_event.html', title='Create Event', form=form)

# ------------------------- EDIT EVENT -------------------------
@events_bp.route('/edit_event/<int:event_id>', methods=['GET', 'POST'])
@login_required
def edit_event(event_id):
    """Handles editing an existing event."""
    event = Event.query.get_or_404(event_id)

    # Security check: Only organizer or admin can edit
    if event.organizer_id != current_user.id and current_user.role != 'Admin':
        flash('You do not have permission to edit this event.', 'danger')
        abort(403)

    form = EventForm(obj=event)  # Pre-fill with existing event data

    if form.validate_on_submit():
        # Update event details
        event.title = form.title.data
        event.description = form.description.data
        event.event_date = form.event_date.data
        event.location = form.location.data

        db.session.commit()
        flash('Your event has been updated successfully!', 'success')

        # Redirect to appropriate dashboard
        if current_user.role == 'Admin':
            return redirect(url_for('dashboard.admin_dashboard'))
        else:
            return redirect(url_for('dashboard.organizer_dashboard'))

    return render_template('edit_event.html', title='Edit Event', form=form, event=event)
```

```
# ------------------------- DELETE EVENT -------------------------
@events_bp.route('/delete_event/<int:event_id>', methods=['POST'])
@login_required
def delete_event(event_id):
    """Handles deleting an event."""
    event = Event.query.get_or_404(event_id)

    # Security check: Only organizer or admin can delete
    if event.organizer_id != current_user.id and current_user.role != 'Admin':
        flash('You do not have permission to delete this event.', 'danger')
        abort(403)

    # Remove all registrations (attendees) to avoid foreign key conflicts
    event.attendees = []
    db.session.commit()

    # Now delete the event
    db.session.delete(event)
    db.session.commit()

    flash('The event has been deleted successfully.', 'success')

    # Redirect to dashboard
    if current_user.role == 'Admin':
        return redirect(url_for('dashboard.admin_dashboard'))
    else:
        return redirect(url_for('dashboard.organizer_dashboard'))

# ------------------------- REGISTER FOR EVENT -------------------------
@events_bp.route('/register/<int:event_id>', methods=['POST'])
@login_required
@role_required('Student')
def register(event_id):
    """Handles event registration for a student."""
    event = Event.query.get_or_404(event_id)

    if current_user.is_registered(event):
        flash('You are already registered for this event.', 'info')
    else:
        current_user.registered_events.append(event)
        db.session.commit()

        # Generate a QR code for event registration
        qr_data = f"user_id:{current_user.id},event_id:{event.id},event_title:{event.title}"
        generate_qr_code(qr_data, current_user.id, event.id, current_app)

        flash('You have successfully registered for the event!', 'success')
```

```
    return redirect(url_for('events.events_list'))

# ------------------------ UNREGISTER FROM EVENT ------------------------
@events_bp.route('/unregister/<int:event_id>', methods=['POST'])
@login_required
@role_required('Student')
def unregister(event_id):
    """Handles event unregistration for a student."""
    event = Event.query.get_or_404(event_id)

    if not current_user.is_registered(event):
        flash('You are not registered for this event.', 'info')
    else:
        current_user.registered_events.remove(event)
        db.session.commit()
        flash('You have successfully unregistered from the event.', 'success')

    return redirect(url_for('events.events_list'))
@events_bp.route('/feedback/<int:event_id>', methods=['GET', 'POST'])
@login_required
@role_required('Student')
def submit_feedback(event_id):
    """Handles feedback submission for an event."""
    event = Event.query.get_or_404(event_id)

    # Security checks:
    # 1. Has the event already passed?
    if event.event_date > datetime.now():
        flash('You can only leave feedback for events that have finished.', 'info')
        return redirect(url_for('dashboard.student_dashboard'))

    # 2. Was the student registered for this event?
    if not current_user.is_registered(event):
        flash('You must be registered for an event to leave feedback.', 'danger')
        return redirect(url_for('dashboard.student_dashboard'))

    # 3. Has the student already left feedback?
    existing_feedback = Feedback.query.filter_by(user_id=current_user.id, event_id=event.id).first()
    if existing_feedback:
        flash('You have already submitted feedback for this event.', 'info')
        return redirect(url_for('dashboard.student_dashboard'))

    form = FeedbackForm()
    if form.validate_on_submit():
        feedback = Feedback(
            rating=int(form.rating.data),
            comment=form.comment.data,
```

```
            user_id=current_user.id,
            event_id=event.id
        )
        db.session.add(feedback)
        db.session.commit()
        flash('Thank you for your feedback!', 'success')
        return redirect(url_for('dashboard.student_dashboard'))

    return render_template('submit_feedback.html', title='Submit Feedback', form=form, event=event)
```

qr.py

```
# eventhive/routes/qr.py

from flask import Blueprint, render_template, request, jsonify
from flask_login import login_required, current_user
from models.models import db, User, Event,registrations
from utils.decorators import role_required
import json

qr_bp = Blueprint('qr', __name__)

@qr_bp.route('/scan')
@login_required
@role_required('Organizer')
def scan():
    """Renders the QR code scanner page for organizers."""
    return render_template('scan.html', title='Scan QR Code')

@qr_bp.route('/verify_attendance', methods=['POST'])
@login_required
@role_required('Organizer')
def verify_attendance():
    """Verifies the scanned QR code data against the database."""
    try:
        data = request.get_json()
        qr_data_str = data.get('qr_data')

        # The QR data is a string like: "user_id:1,event_id:2,event_title:Workshop"
        # We need to parse it to get the user_id and event_id
        parts = qr_data_str.split(',')
        user_id = int(parts[0].split(':')[1])
        event_id = int(parts[1].split(':')[1])

        user = User.query.get(user_id)
        event = Event.query.get(event_id)
```

```
        if not user or not event:
            return jsonify({'success': False, 'message': 'Invalid QR Code: User or Event not found.'})

        # Check if the user is actually registered for this event
        if not user.is_registered(event):
            return jsonify({'success': False, 'message': f'{user.username} is not registered for {event.title}.'})

        # --- (Future Step): Mark attendance in the database ---
        # For now, we just confirm the registration is valid.
        stmt = db.update(registrations).where(
            registrations.c.user_id == user_id,
            registrations.c.event_id == event_id
        ).values(attended=True)

        # Execute the statement and commit
        db.session.execute(stmt)
        db.session.commit()
        return jsonify({
            'success': True,
            'message': f'Success! Attendance confirmed for {user.username} at {event.title}.'
        })

    except Exception as e:
        return jsonify({'success': False, 'message': f'An error occurred: {str(e)}'})
```

base.html

```
<!doctype html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">

    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" rel="stylesheet">

    <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>

    <title>{{ title }} - EventHive</title>

    <link rel="icon" href="{{ url_for('static', filename='images/eventhive.png') }}">

</head>
<body class="bg-light">

    <nav class="navbar navbar-expand-lg navbar-dark bg-dark shadow-sm">
        <div class="container">
            <a class="navbar-brand fw-bold" href="{{ url_for('events.index') }}">EventHive 🏠</a>
```

```
        <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-
target="#navbarNav"
                aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">
            <span class="navbar-toggler-icon"></span>
        </button>

        <div class="collapse navbar-collapse" id="navbarNav">
            <ul class="navbar-nav ms-auto">
                <li class="nav-item">
                    <a class="nav-link" href="{{ url_for('events.index') }}">Home</a>
                </li>
                <li class="nav-item">
                    <a class="nav-link" href="{{ url_for('events.events_list') }}">Events</a>
                </li>

                {% if current_user.is_authenticated %}
                    {% if current_user.role == 'Admin' %}
                        <li class="nav-item">
                            <a class="nav-link" href="{{ url_for('dashboard.admin_dashboard') }}">Admin
Panel</a>
                        </li>

                    {% elif current_user.role == 'Organizer' %}
                        <li class="nav-item">
                            <a class="nav-link" href="{{ url_for('dashboard.organizer_dashboard')
}}">Organizer Panel</a>
                        </li>
                        <li class="nav-item">
                            <a class="nav-link" href="{{ url_for('events.create_event') }}">Create Event</a>
                        </li>

                    {% else %}
                        <li class="nav-item">
                            <a class="nav-link" href="{{ url_for('dashboard.student_dashboard') }}">My
Dashboard</a>
                        </li>
                    {% endif %}

                    <li class="nav-item">
                        <a class="nav-link text-danger" href="{{ url_for('auth.logout') }}">Logout</a>
                    </li>

                {% else %}
                    <li class="nav-item">
                        <a class="nav-link" href="{{ url_for('auth.login') }}">Login</a>
                    </li>
                    <li class="nav-item">
```

```
                    <a class="nav-link btn btn-primary text-white ms-lg-2" href="{{ url_for('auth.register')
}}">Register</a>
                </li>
            {% endif %}
        </ul>
    </div>
  </div>
</nav>

<main class="container mt-4">
  {% with messages = get_flashed_messages(with_categories=true) %}
    {% if messages %}
      {% for category, message in messages %}
        <div class="alert alert-{{ category }} alert-dismissible fade show" role="alert">
          {{ message }}
          <button type="button" class="btn-close" data-bs-dismiss="alert" aria-
label="Close"></button>
        </div>
      {% endfor %}
    {% endif %}
  {% endwith %}

  {% block content %}{% endblock %}
</main>

<footer class="bg-dark text-white text-center py-3 mt-5">
  &copy; 2025 EventHive. All Rights Reserved.
</footer>

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js"></script>

{% block scripts %}{% endblock %}

</body>
</html>
```

admindashboard.html

```
{% extends "base.html" %}

{% block content %}
<h1 class="mb-4">Admin Dashboard</h1>

<div class="row">
  <div class="col-md-6 col-lg-3 mb-4">
    <div class="card text-white bg-primary h-100">
      <div class="card-body">
```

```
            <h5 class="card-title">Total Users</h5>
            <p class="card-text fs-2">{{ users|length }}</p>
          </div>
        </div>
      </div>
      <div class="col-md-6 col-lg-3 mb-4">
        <div class="card text-white bg-success h-100">
          <div class="card-body">
            <h5 class="card-title">Total Events</h5>
            <p class="card-text fs-2">{{ events|length }}</p>
          </div>
        </div>
      </div>
    </div>

    <h2 class="mt-4">Event Registrations</h2>
    <div class="card shadow-sm mb-4">
      <div class="card-body">
        <canvas id="registrationChart"></canvas>
      </div>
    </div>

    <h2 class="mt-4">Manage Users</h2>
    <div class="card shadow-sm">
      <div class="card-body">
        <div class="table-responsive">
          <table class="table table-striped table-hover">
            <thead>
              <tr>
                <th>ID</th>
                <th>Username</th>
                <th>Email</th>
                <th>Role</th>
                <th>Actions</th>
              </tr>
            </thead>
            <tbody>
              {% for user in users %}
              <tr>
                <td>{{ user.id }}</td>
                <td>{{ user.username }}</td>
                <td>{{ user.email }}</td>
                <td><span class="badge bg-secondary">{{ user.role }}</span></td>
                <td>
                  <a href="#" class="btn btn-sm btn-info">View</a>
                </td>
              </tr>
              {% endfor %}
```

```
            </tbody>
        </table>
    </div>
</div>
</div>

<h2 class="mt-5">Manage Events</h2>
<div class="card shadow-sm">
    <div class="card-body">
        <div class="table-responsive">
            <table class="table table-striped table-hover">
                <thead>
                    <tr>
                        <th>ID</th>
                        <th>Title</th>
                        <th>Location</th>
                        <th>Event Date</th>
                        <th>Actions</th>
                    </tr>
                </thead>
                <tbody>
                    {% for event in events %}
                    <tr>
                        <td>{{ event.id }}</td>
                        <td>{{ event.title }}</td>
                        <td>{{ event.location }}</td>
                        <td>{{ event.event_date.strftime('%Y-%m-%d %H:%M') }}</td>
                        <td>
                            <a href="{{ url_for('events.edit_event', event_id=event.id) }}" class="btn btn-sm btn-
info">Edit</a>
                            <form action="{{ url_for('events.delete_event', event_id=event.id) }}"
method="POST" class="d-inline" onsubmit="return confirm('Are you sure you want to delete this
event?');">
                                <button type="submit" class="btn btn-sm btn-danger">Delete</button>
                            </form>
                        </td>
                    </tr>
                    {% endfor %}
                </tbody>
            </table>
        </div>
    </div>
</div>

{% endblock %}

{% block scripts %}
<script>
```

```
    // Get the data from Flask (using the 'tojson' filter for safety)
    const labels = {{ chart_labels|tojson }};
    const data = {{ chart_data|tojson }};

    // Get the canvas element
    const ctx = document.getElementById('registrationChart').getContext('2d');

    // Create the chart
    const registrationChart = new Chart(ctx, {
        type: 'bar', // We want a bar chart
        data: {
            labels: labels,
            datasets: [{
                label: '# of Registrations',
                data: data,
                backgroundColor: 'rgba(0, 123, 255, 0.5)',
                borderColor: 'rgba(0, 123, 255, 1)',
                borderWidth: 1
            }]
        },
        options: {
            scales: {
                y: {
                    beginAtZero: true,
                    ticks: {
                    stepSize: 1
                }
                }
            },
            responsive: true,
            maintainAspectRatio: true
        }
    });
</script>
{% endblock %}
```

create_event.html

```
{% extends "base.html" %}

{% block content %}
<div class="row justify-content-center">
    <div class="col-md-8">
        <div class="card shadow-sm">
            <div class="card-body p-4">
                <h2 class="card-title text-center mb-4">Create a New Event</h2>
                <form method="POST" action="" novalidate>
```

```
                    {{ form.hidden_tag() }}
                    <div class="mb-3">
                        {{ form.title.label(class="form-label") }}
                        {{ form.title(class="form-control") }}
                    </div>
                    <div class="mb-3">
                        {{ form.description.label(class="form-label") }}
                        {{ form.description(class="form-control", rows="5") }}
                    </div>
                    <div class="mb-3">
                        {{ form.event_date.label(class="form-label") }}
                        {{ form.event_date(class="form-control", placeholder="YYYY-MM-DD
HH:MM:SS") }}
                    </div>
                    <div class="mb-3">
                        {{ form.location.label(class="form-label") }}
                        {{ form.location(class="form-control") }}
                    </div>
                    <div class="d-grid mt-4">
                        {{ form.submit(class="btn btn-primary btn-lg") }}
                    </div>
                </form>
            </div>
        </div>
    </div>
</div>
{% endblock %}
```

Edit_event.html

```
{% extends "base.html" %}

{% block content %}
<div class="row justify-content-center">
    <div class="col-md-8">
        <div class="card shadow-sm">
            <div class="card-body p-4">
                <h2 class="card-title text-center mb-4">Edit Your Event</h2>

                <form method="POST" action="" novalidate>
                    {{ form.hidden_tag() }}
                    <div class="mb-3">
                        {{ form.title.label(class="form-label") }}
                        {{ form.title(class="form-control") }}
                    </div>
                    <div class="mb-3">
```

```
                    {{ form.description.label(class="form-label") }}
                    {{ form.description(class="form-control", rows="5") }}
                </div>
                <div class="mb-3">
                    {{ form.event_date.label(class="form-label") }}
                    {{ form.event_date(class="form-control", placeholder="YYYY-MM-DD
HH:MM:SS") }}
                </div>
                <div class="mb-3">
                    {{ form.location.label(class="form-label") }}
                    {{ form.location(class="form-control") }}
                </div>
                <div class="d-grid mt-4">
                    {{ form.submit(class="btn btn-primary btn-lg", value="Save Changes") }}
                </div>
            </form>
        </div>
    </div>
</div>
{% endblock %}
```

event_list.html

```
{% extends "base.html" %}

{% block content %}
    <div class="d-flex justify-content-between align-items-center mb-4">
        <h1 class="h2">Upcoming Events</h1>
        {% if current_user.is_authenticated and (current_user.role == 'Organizer' or current_user.role ==
'Admin') %}
            <a href="{{ url_for('events.create_event') }}" class="btn btn-primary">Create New
Event</a>
        {% endif %}
    </div>

    {% if events %}
        <div class="row g-4">
            {% for event in events %}
                <div class="col-md-6 col-lg-4">
                    <div class="card h-100 shadow-sm d-flex flex-column">
                        <div class="card-body">
                            <h5 class="card-title">{{ event.title }}</h5>
                            <h6 class="card-subtitle mb-2 text-muted">{{ event.location }}</h6>
                            <p class="card-text">{{ event.description[:100] }}...</p>
                        </div>
                        <div class="card-footer bg-transparent border-top-0">
                            <small class="text-muted">Date: {{ event.event_date.strftime('%B %d,
```

```
%Y at %I:%M %p') }}</small>

                                {% if current_user.is_authenticated and current_user.role == 'Student' %}
                                    {% if current_user.is_registered(event) %}
                                        <form action="{{ url_for('events.unregister', event_id=event.id) }}"
method="POST" class="d-inline float-end">
                                            <button type="submit" class="btn btn-sm btn-
warning">Unregister</button>
                                        </form>
                                    {% else %}
                                        <form action="{{ url_for('events.register', event_id=event.id) }}"
method="POST" class="d-inline float-end">
                                            <button type="submit" class="btn btn-sm btn-
success">Register</button>
                                        </form>
                                    {% endif %}
                                {% endif %}
                            </div>
                        </div>
                    </div>
                {% endfor %}
            </div>
        {% else %}
            {% endif %}
{% endblock %}
```

index.html

```
{% extends "base.html" %}
{% block content %}
<div class="p-5 mb-4 bg-white rounded-3 shadow-sm text-center">
    <div class="container-fluid py-5">
        <h1 class="display-5 fw-bold">Welcome to EventHive</h1>
        <p class="fs-4">The central hub for all university events. Discover, register, and engage with
campus life.</p>
        <a href="{{ url_for('events.events_list') }}" class="btn btn-primary btn-lg mt-3">Browse
Upcoming Events</a>
    </div>
</div>
{% endblock %}
```

login.html

```
{% extends "base.html" %}

{% block content %}
<div class="row justify-content-center">
    <div class="col-md-6 col-lg-5">
```

```
        <div class="card shadow-sm">
            <div class="card-body p-4">
                <h2 class="card-title text-center mb-4">Login</h2>

                <form method="POST" action="" novalidate>
                    {{ form.hidden_tag() }} <div class="mb-3">
                        {{ form.email.label(class="form-label") }}
                        {{ form.email(class="form-control", placeholder="you@example.com") }}
                        {% for error in form.email.errors %}
                            <span class="text-danger small">{{ error }}</span>
                        {% endfor %}
                    </div>

                    <div class="mb-3">
                        {{ form.password.label(class="form-label") }}
                        {{ form.password(class="form-control", placeholder="Enter your password") }}
                        {% for error in form.password.errors %}
                            <span class="text-danger small">{{ error }}</span>
                        {% endfor %}
                    </div>

                    <div class="mb-3 form-check">
                        {{ form.remember_me(class="form-check-input") }}
                        {{ form.remember_me.label(class="form-check-label") }}
                    </div>

                    <div class="d-grid mt-4">
                        {{ form.submit(class="btn btn-primary btn-lg") }}
                    </div>
                </form>

                <div class="text-center mt-3">
                    <small>Need an account? <a href="{{ url_for('auth.register') }}">Sign
Up</a></small>
                </div>
            </div>
        </div>
    </div>
</div>
{% endblock %}
```

orgainser_dashboard.html

```
{% extends "base.html" %}

{% block content %}
```

```
<div class="d-flex justify-content-between align-items-center mb-4">
   <h1 class="h2">Organizer Dashboard</h1>
   <a href="{{ url_for('qr.scan') }}" class="btn btn-success">Open Attendance Scanner</a>
</div>

<h2 class="h4">My Created Events</h2>

{% if event_data %}
   <div class="accordion" id="eventsAccordion">
      {% for data in event_data %}
      <div class="accordion-item mb-3 shadow-sm">
         <h2 class="accordion-header" id="heading-{{ data.event.id }}">
            <button class="accordion-button collapsed" type="button" data-bs-toggle="collapse" data-bs-
target="#collapse-{{ data.event.id }}" aria-expanded="false" aria-controls="collapse-{{ data.event.id
}}">
               <strong>{{ data.event.title }}</strong>  ({{ data.attendees|length }} Registered)
            </button>
         </h2>
         <div id="collapse-{{ data.event.id }}" class="accordion-collapse collapse" aria-
labelledby="heading-{{ data.event.id }}" data-bs-parent="#eventsAccordion">
            <div class="accordion-body">
               <h5>Registered Students</h5>
               {% if data.attendees %}
                  <ul class="list-group mb-3">
                  {% for user, attended in data.attendees %}
                     <li class="list-group-item d-flex justify-content-between align-items-center">
                        {{ user.username }} ({{ user.email }})
                        {% if attended %}
                           <span class="badge bg-success rounded-pill">Attended</span>
                        {% else %}
                           <span class="badge bg-secondary rounded-pill">Registered</span>
                        {% endif %}
                     </li>
                  {% endfor %}
                  </ul>
               {% else %}
                  <p>No students have registered for this event yet.</p>
               {% endif %}

               <hr>
               <h5>Manage Event</h5>
               <div class="d-flex gap-2 mt-2">
                  <a href="{{ url_for('events.edit_event', event_id=data.event.id) }}" class="btn btn-sm
btn-outline-primary">
                     <i class="bi bi-pencil-square"></i> Edit Event
                  </a>
                  <form action="{{ url_for('events.delete_event', event_id=data.event.id) }}"
method="POST" class="d-inline" onsubmit="return confirm('Are you sure you want to delete this
```

```
event? This action cannot be undone.');">
                        <button type="submit" class="btn btn-sm btn-outline-danger">
                            <i class="bi bi-trash"></i> Delete Event
                        </button>
                    </form>
                </div>

                <!-- 📝 Event Feedback Section -->
                <hr>
                <h5>Event Feedback ({{ data.feedbacks|length }})</h5>
                {% if data.feedbacks %}
                    {% for fb in data.feedbacks %}
                        <div class="card mb-2">
                            <div class="card-body">
                                <strong>Rating: {{ fb.rating }}/5</strong>
                                <p class="card-text mb-0">{{ fb.comment }}</p>
                                <small class="text-muted">by {{ fb.author.username }}</small>
                            </div>
                        </div>
                    {% endfor %}
                {% else %}
                    <p>No feedback has been submitted for this event yet.</p>
                {% endif %}
                <!-- End Feedback Section -->

            </div>
        </div>
    </div>
    {% endfor %}
  </div>
{% else %}
  <div class="text-center p-5 bg-white rounded shadow-sm">
    <h3>You haven't created any events yet.</h3>
    <a href="{{ url_for('events.create_event') }}" class="btn btn-primary mt-3">Create Your First
Event</a>
  </div>
{% endif %}

{% endblock %}
```

register.html

```
{% extends "base.html" %}

{% block content %}
<div class="row justify-content-center">
  <div class="col-md-6 col-lg-5">
    <div class="card shadow-sm">
```

```
        <div class="card-body p-4">
            <h2 class="card-title text-center mb-4">Create Account</h2>

            <form method="POST" action="" novalidate>
                {{ form.hidden_tag() }}

                <div class="mb-3">
                    {{ form.username.label(class="form-label") }}
                    {{ form.username(class="form-control", placeholder="Enter your username") }}
                    {% for error in form.username.errors %}
                        <span class="text-danger small">{{ error }}</span>
                    {% endfor %}
                </div>

                <div class="mb-3">
                    {{ form.email.label(class="form-label") }}
                    {{ form.email(class="form-control", placeholder="you@example.com") }}
                    {% for error in form.email.errors %}
                        <span class="text-danger small">{{ error }}</span>
                    {% endfor %}
                </div>

                <div class="mb-3">
                    {{ form.password.label(class="form-label") }}
                    {{ form.password(class="form-control", placeholder="Enter your password") }}
                    {% for error in form.password.errors %}
                        <span class="text-danger small">{{ error }}</span>
                    {% endfor %}
                </div>

                <div class="mb-3">
                    {{ form.confirm_password.label(class="form-label") }}
                    {{ form.confirm_password(class="form-control", placeholder="Confirm your password")
}}
                    {% for error in form.confirm_password.errors %}
                        <span class="text-danger small">{{ error }}</span>
                    {% endfor %}
                </div>

                <!-- ✅ New Role Dropdown Field -->
                <div class="mb-3">
                    {{ form.role.label(class="form-label") }}
                    {{ form.role(class="form-select") }}
                    {% for error in form.role.errors %}
                        <span class="text-danger small">{{ error }}</span>
                    {% endfor %}
                </div>
```

```
                <div class="d-grid mt-4">
                    {{ form.submit(class="btn btn-primary btn-lg") }}
                </div>
            </form>

            <div class="text-center mt-3">
                <small>Already have an account? <a href="{{ url_for('auth.login') }}">Log
In</a></small>
            </div>

        </div>
    </div>
</div>
</div>
{% endblock %}
```

scan.html

```
{% extends "base.html" %}

{% block content %}
<div class="row justify-content-center">
    <div class="col-md-8 col-lg-6">
        <div class="card shadow-sm">
            <div class="card-body text-center">
                <h2 class="card-title mb-4">Scan Attendance QR Code</h2>

                <div id="qr-reader" style="width:100%;"></div>

                <div id="qr-reader-results" class="mt-3"></div>
            </div>
        </div>
    </div>
</div>

<script src="https://unpkg.com/html5-qrcode" type="text/javascript"></script>
<script>
    function docReady(fn) {
        // see if DOM is already available
        if (document.readyState === "complete" || document.readyState === "interactive") {
            // call on next available tick
            setTimeout(fn, 1);
        } else {
            document.addEventListener("DOMContentLoaded", fn);
        }
    }

    docReady(function () {
        const resultContainer = document.getElementById('qr-reader-results');
```

```
        let lastResult, countResults = 0;

        // This function is called when a QR code is successfully scanned
        async function onScanSuccess(decodedText, decodedResult) {
            if (decodedText !== lastResult) {
                lastResult = decodedText;
                resultContainer.innerHTML = `<div class="alert alert-info">Scanning...</div>`;

                // --- Send the scanned data to our Flask backend ---
                try {
                    const response = await fetch("{{ url_for('qr.verify_attendance') }}", {
                        method: 'POST',
                        headers: {
                            'Content-Type': 'application/json',
                        },
                        body: JSON.stringify({ qr_data: decodedText }),
                    });

                    const result = await response.json();

                    if (result.success) {
                        resultContainer.innerHTML = `<div class="alert alert-
success">${result.message}</div>`;
                    } else {
                        resultContainer.innerHTML = `<div class="alert alert-
danger">${result.message}</div>`;
                    }
                } catch (error) {
                    console.error('Error:', error);
                    resultContainer.innerHTML = `<div class="alert alert-danger">Could not verify
attendance.</div>`;
                }
            }
        }

        // Initialize the QR Code scanner
        var html5QrcodeScanner = new Html5QrcodeScanner(
            "qr-reader", { fps: 10, qrbox: 250 });
        html5QrcodeScanner.render(onScanSuccess);
    });
</script>
{% endblock %}
```

student_dashboard.html

```
{% extends "base.html" %}
{# Import datetime to compare event dates #}
{% set now = [datetime.now()] %}
```

```
{% block content %}
<h1 class="mb-4">My Registered Events</h1>
{% if events %}
  <div class="row g-4">
    {% for event in events %}
      <div class="col-md-6">
        <div class="card shadow-sm h-100">
          <div class="row g-0 h-100">
            <!-- Left side: Event Details -->
            <div class="col-md-8 d-flex flex-column">
              <div class="card-body">
                <h5 class="card-title">{{ event.title }}</h5>
                <p class="card-text"><strong>Location:</strong> {{ event.location }}</p>
                <p class="card-text">
                  <small class="text-muted">
                    {{ event.event_date.strftime('%A, %B %d, %Y at %I:%M %p') }}
                  </small>
                </p>
              </div>
              <div class="card-footer bg-transparent border-top-0 mt-auto">
                {% if event.event_date < now[0] %}
                  <span class="badge bg-secondary">Event Finished</span>
                  <a href="{{ url_for('events.submit_feedback', event_id=event.id) }}"
                    class="btn btn-sm btn-outline-primary float-end">
                     Leave Feedback
                  </a>
                {% else %}
                  <span class="badge bg-success">Upcoming</span>
                {% endif %}
              </div>
            </div>
            <div class="col-md-4 d-flex align-items-center justify-content-center p-3">
              {% set qr_filename = 'qr_codes/event' + event.id|string + '_user' +
current_user.id|string + '.png' %}
                <img src="{{ url_for('static', filename=qr_filename) }}"
                    alt="Event QR Code"
                    class="img-fluid rounded shadow-sm border">
            </div>
          </div>
        </div>
      </div>
    {% endfor %}
  </div>
{% else %}
  <div class="text-center p-5 bg-white rounded shadow-sm">
    <h3>You haven't registered for any events yet.</h3>
    <p class="text-muted">Explore upcoming events and join to participate!</p>
```

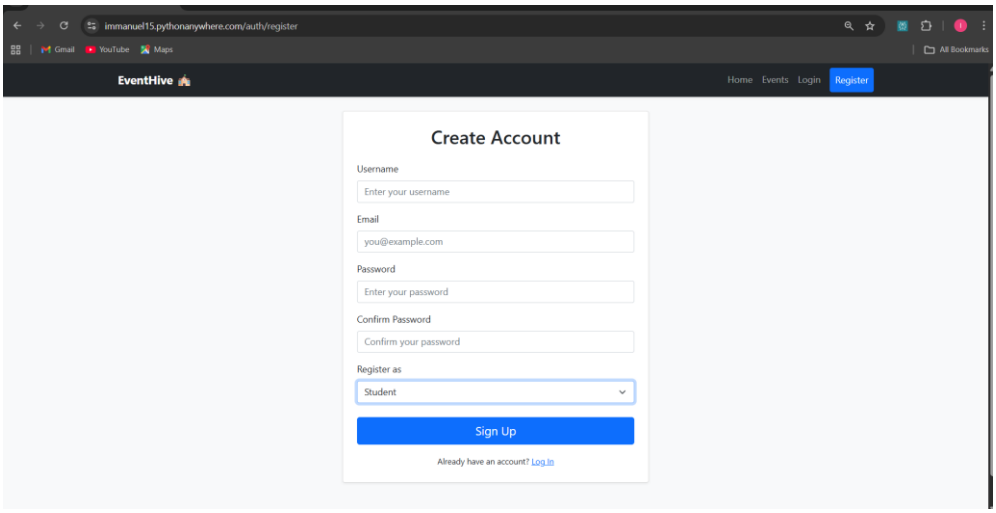
```
        <a href="{{ url_for('events.events_list') }}" class="btn btn-primary mt-3">Browse Events</a>
    </div>
{% endif %}
{% endblock %}
```

submit_feedback.html

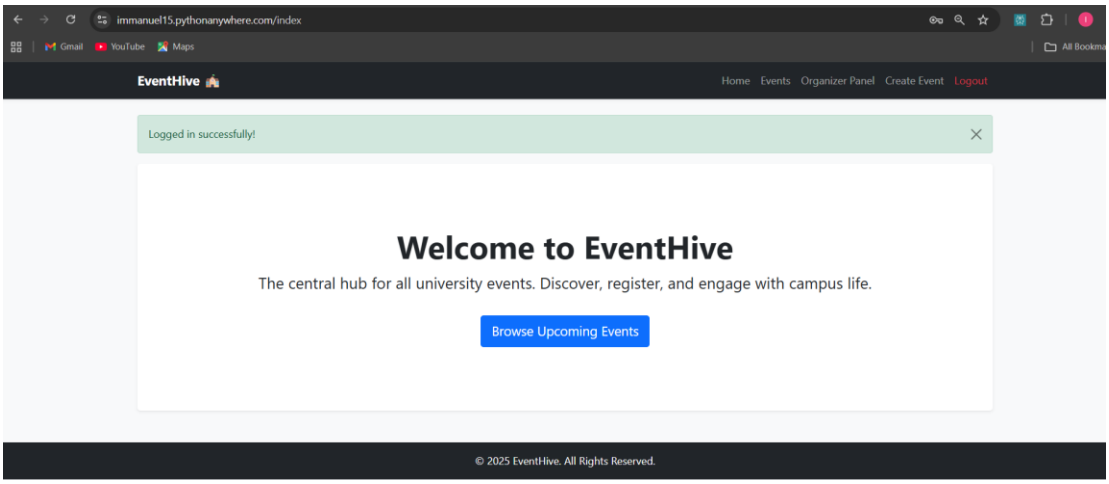
```
{% extends "base.html" %}
{% block content %}
<div class="row justify-content-center">
    <div class="col-md-8">
        <div class="card shadow-sm">
            <div class="card-body p-4">
                <h2 class="card-title text-center mb-1">Submit Feedback</h2>
                <p class="text-center text-muted mb-4">For: <strong>{{ event.title }}</strong></p>
                <form method="POST" action="" novate>
                    {{ form.hidden_tag() }}
                    <div class="mb-3">
                        <label class="form-label">{{ form.rating.label }}</label>
                        {% for subfield in form.rating %}
                            <div class="form-check">
                                {{ subfield(class="form-check-input") }}
                                {{ subfield.label(class="form-check-label") }}
                            </div>
                        {% endfor %}
                    </div>
                    <div class="mb-3">
                        {{ form.comment.label(class="form-label") }}
                        {{ form.comment(class="form-control") }}
                    </div>
                    <div class="d-grid mt-4">
                        {{ form.submit(class="btn btn-primary btn-lg") }}
                    </div>
                </form>
            </div>
        </div>
    </div>
</div>
{% endblock %}
```

CHAPTER -3

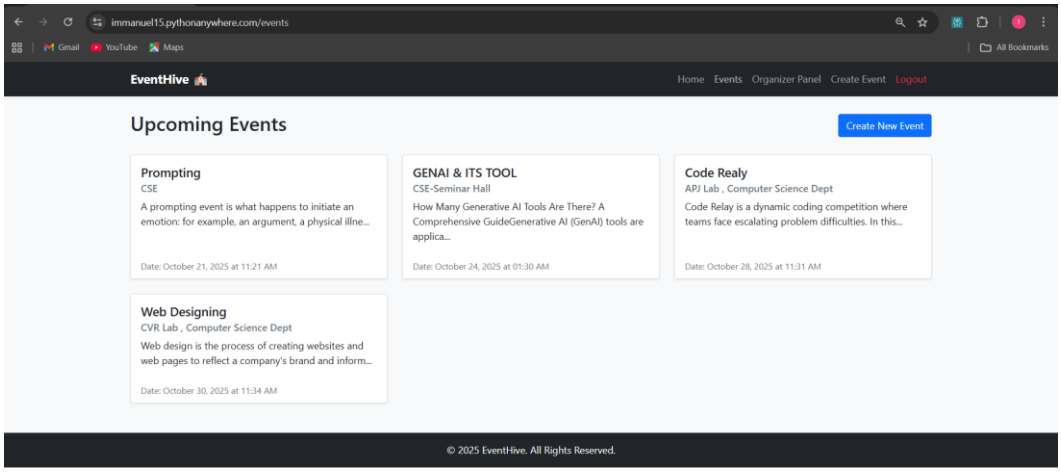
IMPLEMENTATION RESULTS



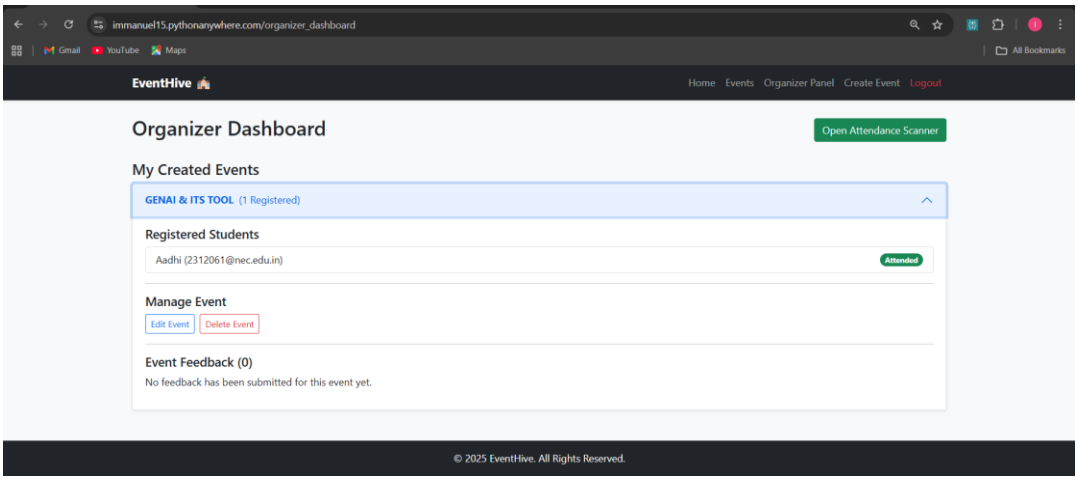
Secure form captures 5 fields (Username, Email, Password, Role). A dropdown allows users to self-select as a "Student" or "Organizer," and 100% server-side validation prevents duplicate accounts or invalid data.



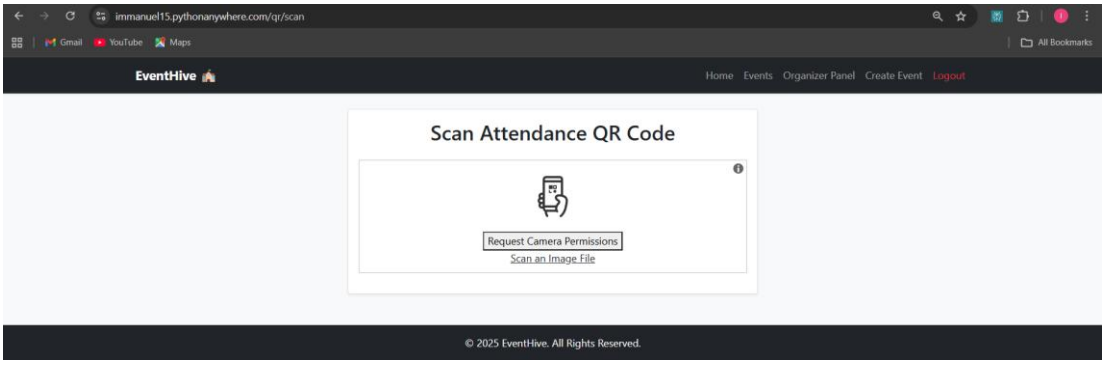
The home page is the main landing area, featuring a welcome message that explains the platform's purpose to new users. Its main goal is to guide visitors to the event list page using the "Browse Upcoming Events" button



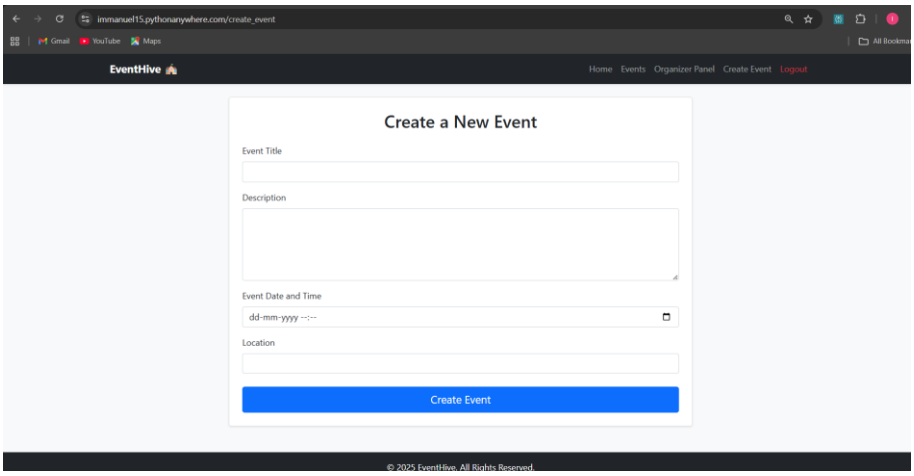
This page displays all available events in a card-based layout, allowing users to see details like the event title, location, and date. It's the main registration hub where students can click "Register" or "Unregister," and Organizers/Admins can access the "Create New Event" button.



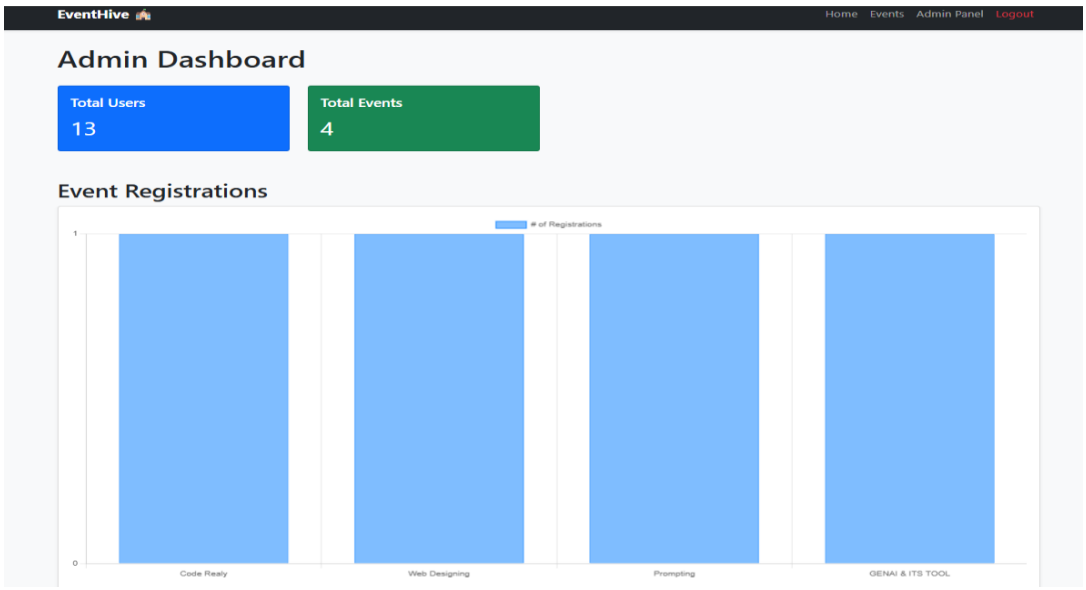
The Organizer Dashboard is the central hub for event creators, showing a list of *only* the events they've personally made. From there, they can expand each event to see the full list of registered students, check their attendance status, and read any submitted feedback.



The attendance scanner is a page for Organizers that uses the device's camera to read a student's unique QR code. When a valid code is scanned, it instantly sends a request to the backend to mark that student as "Attended" in the database.



This secure form allows Organizers and Admins to enter new event details, which are then validated, saved to the database, and listed publicly.



The Admin Dashboard is the high-level control center, providing analytics like "Total Users" and a chart of event registrations. It also allows the administrator to manage and see all users and all events on the entire platform.

CHAPTER – 4

CONCLUSION

This project successfully addresses the critical inefficiencies of manual university event management by delivering EventHive, a fully functional, centralized web application. Built with Flask, the system provides a complete, end-to-end solution, from event creation and online student registration to modern, QR-code-based attendance tracking and post-event feedback collection. By implementing distinct roles for Admins, Organizers, and Students, along with a data-driven analytics dashboard, EventHive replaces error-prone paper processes with an automated, secure, and user-friendly platform. The project successfully meets all its objectives, proving that this digital solution can significantly reduce administrative overhead and enhance student engagement across the campus.