M.Kumarasamy
College of Engineering
NAAC Accredited Autonomous Institution
Approved by AICTE & Affiliated to Anna University
ISO 9001:2015 Certified Institution
Thalavapalayam, Karur - 639 113, TAMILNADU.

A Project Report

on

COLLEGE EVENT MANAGEMENT SYSTEM

Submitted in partial fulfillment of requirements for the award of the course

of

CGB1221-DATABASE MANAGEMENT SYSTEMS

Under the guidance of

Ms.A.Nithyasri

Assistant Professor / AI

Submitted By

SARVANA PRIYAN .S T	**(927623BAD100)**
SARDHEESH. M	**(927623BAD101)**

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

M.KUMARASAMY COLLEGE OF ENGINEERING
(Autonomous)

KARUR – 639 113

MAY 2025

M.Kumarasamy
College of Engineering
NAAC Accredited Autonomous Institution
Approved by AICTE & Affiliated to Anna University
ISO 9001:2015 Certified Institution
Thalavapalayam, Karur - 639 113, TAMILNADU.

M. KUMARASAMY COLLEGE OF ENGINEERING

(Autonomous Institution affiliated to Anna University, Chennai)

KARUR – 639 113

BONAFIDE CERTIFICATE

This is to certify that this project report on **“COLLEGE EVENT MANAGEMENT SYSTEM”** is the bonafide work of **SARAVANA PRIYAN S T (927623BAD100),SARDHEESH M(927623BAD101)** who carried out the project work during the academic year 2024 - 2025 under my supervision.

Signature

Ms. A. NITHYASRI, M.E.,

SUPERVISOR,

Department of Artificial Intelligence,

M. Kumarasamy College of Engineering,

Thalavapalayam, Karur - 639 113.

Signature

Dr. A. SELVI, M.Tech., Ph.D.,

HEAD OF THE DEPARTMENT,

Department of Artificial Intelligence,

M. Kumarasamy College of Engineering,

Thalavapalayam, Karur - 639 113.

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

VISION OF THE INSTITUTION

To emerge as a leader among the top institutions in the field of technical education

MISSION OF THE INSTITUTION

- Produce smart technocrats with empirical knowledge who can surmount the global challenges
- Create a diverse, fully-engaged, learner-centric campus environment to provide quality education to the students
- Maintain mutually beneficial partnerships with our alumni, industry, and Professional associations

VISION OF THE DEPARTMENT

To excel in education, innovation, and research in Artificial Intelligence and Data Science to fulfil industrial demands and societal expectations.

MISSION OF THE DEPARTMENT

M1: To educate future engineers with solid fundamentals, continually improving teaching methods using modern tools.

M2: To collaborate with industry and offer top-notch facilities in a conducive learning environment.

M3: To foster skilled engineers and ethical innovation in AI and Data Science for global recognition and impactful research.

M4: To tackle the societal challenge of producing capable professionals by instilling employability skills and human values.

PROGRAM EDUCATIONAL OBJECTIVES (PEOs)

Graduates will be able to:

- **PEO 1:** Compete on a global scale for a professional career in Artificial Intelligence and Data Science.
- **PEO 2:** Provide industry-specific solutions for the society with effective communication and ethics.
- **PEO 3:** Hone their professional skills through research and lifelong learning initiatives.

PROGRAM OUTCOMES (POs)

Engineering students will be able to:

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

10. Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, give and receive clear instructions.

10. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one’s own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
11. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

PROGRAM SPECIFIC OUTCOMES (PSOs)

- **PSO1:** Capable of finding the important factors in large datasets, simplify the data, and improve predictive model accuracy.
- **PSO2:** Capable of analyzing and providing a solution to a given real-world problem by designing an effective program.

ABSTRACT

The College Event Management System (CEMS) is a web-based application developed to streamline the organization, management, and participation of events within educational institutions. It allows users to register, log in, explore events, and register for main, sub, or standalone events. Participants can track their registration status and receive real-time updates, enhancing the overall experience. Admins can manage users, oversee event activities, track resource usage, and generate detailed analytics. Organisers can create events, allocate resources, handle deallocations, and access reports. The system uses HTML, CSS, and JavaScript for the frontend, Python with Flask for backend operations, and MySQL for storing data. It supports UPI payment Key features include document verification for organisers, resource conflict detection, email notifications, and exportable reports in PDF and Excel formats. DBMS principles such as ER modeling, normalization, and transaction handling are applied to ensure data accuracy and consistency. Modules like User Roles, Event Management, Resource Allocation, Payment, and Reporting work together seamlessly to deliver complete functionality. The architecture is modular, scalable, and responsive across devices. CEMS ensures security, efficiency, and a smooth user experience while enabling institutions to manage events digitally and professionally.

KEYWORDS:

Event Registration, Real-time Notifications, Report Generation, Event Analytics, payment gateway

ABSTRACT WITH POs AND PSOs MAPPING

ABSTRACT	COs MAPPED	POs MAPPED	PSOs MAPPED

The College Event Management System (CEMS) is a web-based platform that streamlines event organization, resource allocation, and participation in educational institutions. Built using Flask, MySQL, and modern web technologies, it supports role-based access, real-time updates, and secure payments . With features like analytics, and responsive design, CEMS ensures efficient and professional event management. It also provides exportable reports in PDF/Excel formats and enables organisers to manage main, sub, and standalone events seamlessly. The modular and scalable architecture ensures smooth performance across various devices. Notifications, email alerts, and a user-friendly dashboard further enhance the experience for admins, organisers, and participants.	CO1
	CO2
	CO3
	CO4
		PO1(3)
		PO2(3)
		PO3(3)
		PO5(2)
		PO10(3)
		PO11(1)
			PSO1(2)
			PSO2(1)

Note: 1- Low, 2-Medium, 3- High

SUPERVISOR **HEAD OF THE DEPARTMENT**

TABLE OF CONTENTS

CHAPTER 1

CHAPTER 1

INTRODUCTION

1.1 PROBLEM STATEMENT

Design and develop a College Event Management System (CEMS) that streamlines the planning, organization, and management of various events within an educational institution. The system should enable users to register and log in with role-based access as participants, organizers, or administrators. It must support creating, updating, and managing main, sub, or standalone events, handling participant registrations, allocating and deallocating event resources, and generating detailed reports in PDF or Excel formats. The platform should integrate secure payment processing, provide real-time updates and notifications, and include features such as document verification and conflict detection. The aim is to ensure seamless coordination, efficient resource usage, and a user-friendly experience for both students and faculty involved in college events.

1.2 INTRODUCTION

The College Event Management System (CEMS) is a comprehensive web-based application designed to digitize and simplify the organization and management of events within educational institutions. It allows users to register, log in, explore events, and register for main, sub, or standalone events. Participants can view event details, receive real-time updates, and track their registration status. Organisers can create events, allocate resources, manage deallocations, and generate detailed reports. Admins oversee user activity, monitor platform-wide events, and manage resources and analytics through a centralized dashboard. The system integrates UPI for secure payment processing, supporting UPI, cards, and net banking. Built using HTML, CSS, JavaScript for the frontend and Python with Flask and MySQL for the backend, it ensures a responsive, modular, and scalable environment. The application applies core DBMS concepts such as ER modeling, normalization, SQL queries, and transaction handling to ensure data integrity and consistency. Features like document verification, exportable reports, and notification alerts enhance user experience. Overall, CEMS provides an efficient and professional digital platform for managing institutional events.

1.3 DATABASE MANAGEMENT SYSTEMS

A Database Management System (DBMS) in the context of the College Event Management System (CEMS) is a critical component that enables efficient storage, management, and manipulation of event-related data. It supports functionalities such as user registration, event creation, resource allocation, and payment tracking while ensuring data integrity, security, and consistency. It also handles key processes like transaction management, access control, and data backup to maintain reliable and secure operations. In CEMS, a relational DBMS like MySQL is typically used to manage complex relationships between users, events, resources, and registrations.

1.4 PYTHON FLASK

Python with Flask is a powerful and lightweight open-source web framework used for developing dynamic and data-driven web applications. Flask is a micro-framework, meaning it provides essential tools to build web applications without the complexity of larger frameworks. It allows developers to write Python code that handles server-side logic and interacts seamlessly with frontend templates. Flask supports integration with databases like MySQL, making it ideal for building scalable and interactive systems. It includes features like routing, form handling, session management, and authentication. Well-known applications built using Flask include Pinterest’s early versions, Netflix internal tools, and the LinkedIn data insights platform.

1.5 OBJECTIVE

The main objective of the College Event Management System (CEMS) is to develop a user-friendly platform for organizing, managing, and participating in college events efficiently. It provides seamless event browsing, secure registration, and real-time updates for participants. enables administrators and organizers to create events, allocate resources, and monitor registrations effectively. It ensures secure authentication and role-based access for admins, organizers, and participants. With integrated payment gateway support, the system handles secure transactions. Emphasizing data integrity and minimizing redundancy, the project enhances user experience with an interactive interface and timely notifications. It simplifies event coordination and improves efficiency for both college students.

CHAPTER 2

PROJECT METHODOLOGY

2.1 PROPOSED WORK

Admin Workflow:

1. **Login/Sign Up:** Login with secure credentials. Dashboard Overview of platform metrics, user activity, and event stats.
2. **Manage users:** Approve organizer accounts, manage roles and permissions.
3. **Monitor events:** View all events, track registration and resource allocation
4. **Generate reports:** Access analytics on participation, revenue, and resource allocation.
5. **System Oversight::** Handle verification, resolve conflicts, and maintain data integrity.

Organizer Workflow:

1. **Login/Signup:** Register/Login and submit documents for verification. Admin Approval for organizer access.
2. **Allocate Resources**: Assign and manage equipment (e.g., projectors, chairs).
3. **Monitor Registrations:** Track participant count and manage event logistics.
4. **View Reports**: Analyze event success, feedback, and resource usage.
5. **Handle Payments** : Track participant payments and transactions.

Participant Workflow:

1. **Sign Up/Login :** Login to the platform. Browse Events by categories, tags, or dates.
2. **Register for Events**: Fill forms and make payments.
3. **Notifications**: Get real-time updates on registration, venue, or time changes.
4. **Track Registrations**: View registered events and statuses.
5. **Participate in Events**: Attend and engage based on registration confirmation.

2.2 ARCHITECTURE

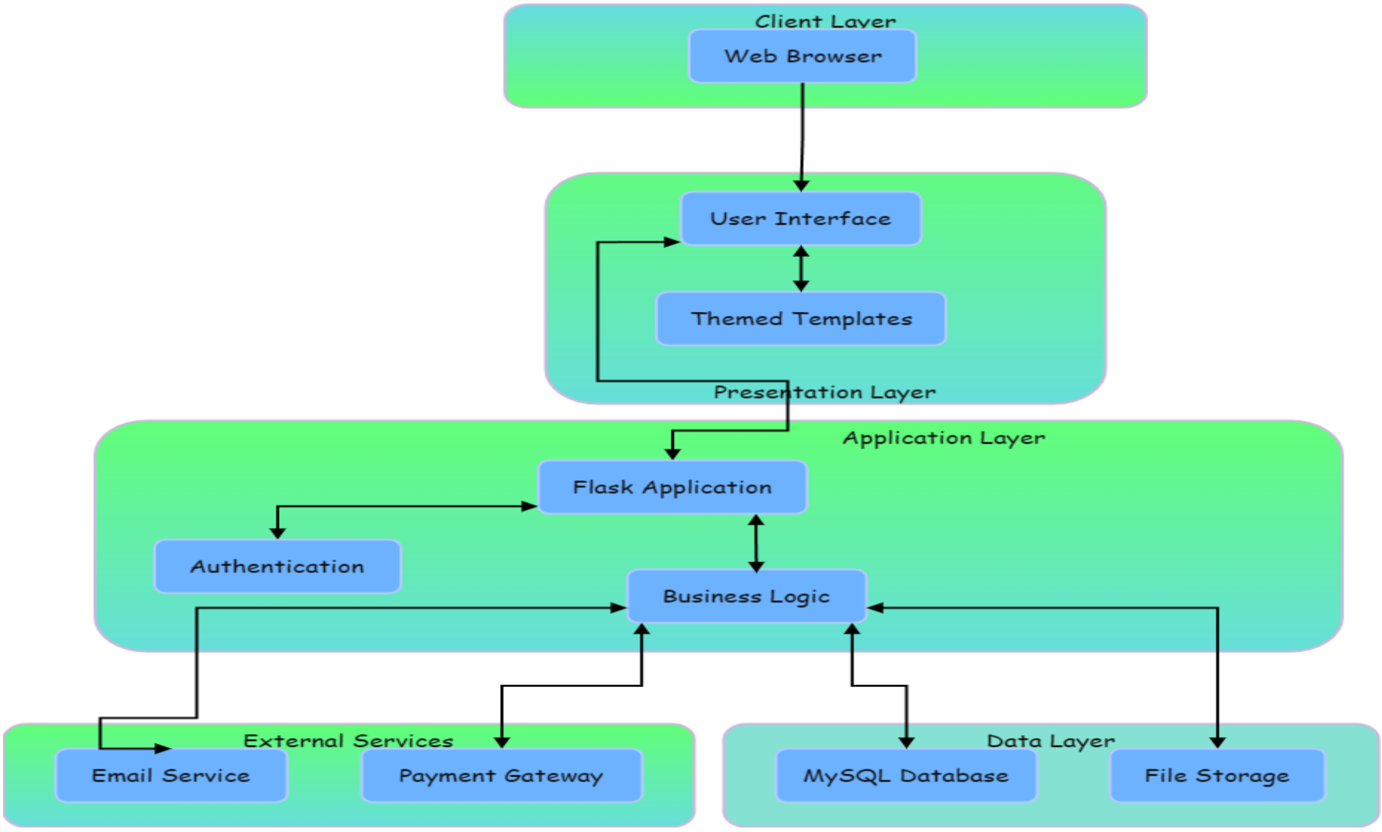


Figure 2.2 Architecture

2.3 E-R DIAGRAM:

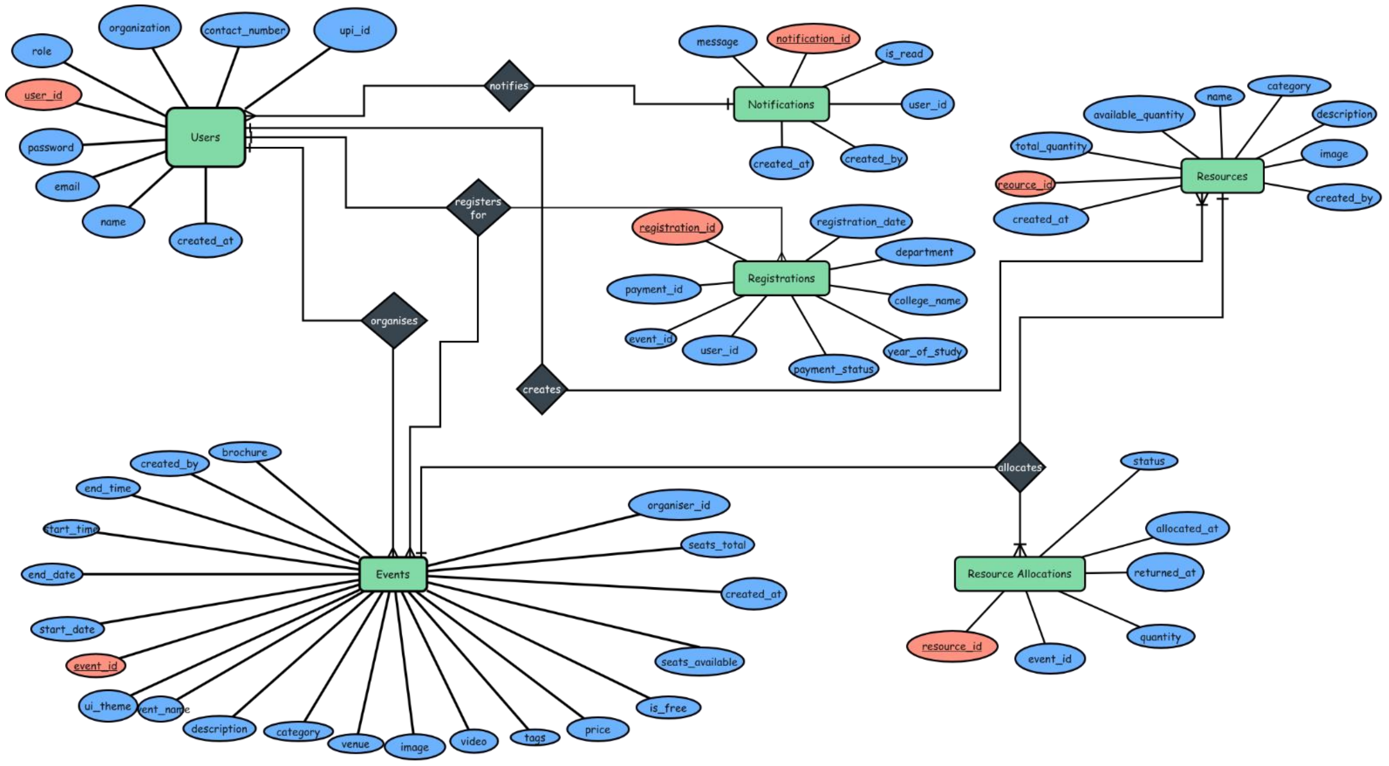


Figure 2.3 ER diagram

2.4 SCHEMA DIAGRAM

User table:

id	name	Password	Role	Email	contact	upi_id
1	saravana	********	organiser	saravanapriyanst@gmail.co	1234567890	saravana@okici
2	sardheesh	********	participant	sardheeshmuthusamy@gmail.com	1234567890	-
3	admin	********	admin	admin@example.com	1234567890	-

Events Table:

event _id	name	start_date	End_date	Start_time	End_time	Organiser_id	Venue
1	orlia	25.05.2025	25.05.2025	9:00 AM	5:00 PM	1	MKCE

Notification Table:

id	message	created_at	created_by	User_id	is_read	Is_important
1	Your otp is 234123	24.5.2025	Organizer	1	1	1

Resource Table:

id	name	category	description	total_quantity	Avilable_quantity	created_by
1	Chair	essential	Uses for seating of the participant	12	20	sarvana

Resource allocation Table:

id	resource _id	event_id	quantity	allocated_at	returned_at	status
1	1	1	3	25.05.2025	26.05.2025	avilable

Registration Table:

id	event_id	User_id	registration Date	Payment ID	Payement_status	Payment Method	Verification _notes	Verification Date	Verified_by	amount
1	1	2	26.05.2025	1	paid	upi	verified	26.05.2025	sarvana	100

CHAPTER 3
SOFTWARE REQUIREMENTS

3.1 Front End

In the College Event Management System (CEMS), HTML (Hypertext Markup Language) is used to structure each webpage, including layouts for dashboards, event listings, registration forms, and resource management interfaces. HTML defines the core elements such as buttons, tables, input fields, and cards that users interact with across roles. CSS (Cascading Style Sheets) styles these elements to create a visually appealing and consistent interface, managing colors, fonts, spacing, and responsive layouts through Flexbox, Grid, and media queries. Bootstrap, a widely-used CSS framework, is also employed to ensure mobile responsiveness and UI consistency. JavaScript enhances interactivity and user engagement by handling tasks like form validation, dynamic event filtering, interactive charts, modals, and real-time updates without page reloads. Together, HTML, CSS, JavaScript, and supporting frameworks provide a responsive, intuitive, and user-focused front-end experience for admins, organizers, and participants in the CEMS platform.

3.1 Back End

In the backend of the College Event Management System (CEMS), Python with the Flask framework is used as the main server-side technology to handle all business logic and facilitate communication between the front end and the database. Flask processes user actions such as login, registration, event creation, resource allocation, and payment handling. Python scripts manage dynamic content rendering, database operations, and API responses based on user roles. The MySQL database, managed through MySQL Workbench, stores all key data including user accounts, event details, registrations, resources, payments, and system notifications. Each front-end interaction triggers a Flask route that executes SQL queries using ORM or raw queries to update or retrieve data. MySQL Workbench serves as the development and management tool for database design, testing, and monitoring. This backend setup ensures efficient data processing, secure authentication, and real-time system updates, providing a reliable foundation for the entire CEMS platform.

CHAPTER 4

MODULE DESCRIPTION

4.1 Event Management Module

The event management module enables organizers to create, update, delete, and view events. Each event includes data like name, type, date, venue, capacity, pricing, description, media, and brochures. The interface allows fast editing and event categorization for easy browsing. Events can be main, sub, or standalone. This module keeps the platform dynamic, organized, and updated with current activities.

4.2 Registration &payment Module

The registration and payment module enables participants to register for events through a streamlined interface. It supports secure payments via direct UPI transactions exclusively. Users receive real-time confirmation upon successful registration and payment completion. Organizers and admins can track registration statuses, payment history, and attendance efficiently. The module ensures a smooth experience with proper transaction handling, accurate data storage, and immediate user notifications throughout the registration process. The UPI-based payment system provides a familiar and accessible payment method for all participants while maintaining transaction security.

4.3 Resource Allocation Module

Organizers allocate venues, equipment, and materials to events using this module. It tracks availability and resource conflicts. Admins can view current allocations and pending returns. Alerts notify users of low resource availability. It ensures proper planning and supports smooth execution of events by managing logistics effectively.

4.4 Feedback Module

The Feedback Module enables participants to provide comments and ratings after attending events, helping organizers evaluate performance and identify areas for improvement. Feedback can be submitted through a user-friendly interface, with options for rating specific aspects like organization, content, and speaker quality. The collected feedback is stored securely in the database and can be viewed by admins and organizers through detailed reports. This module promotes continuous improvement, enhances event quality, and ensures participant satisfaction in future events.

4.5 Notification Module

The Notification Module delivers real-time alerts to users about event updates, registration confirmations, schedule changes, and reminders. Notifications are sent via email or in-app messages. Organizers can schedule or trigger messages based on event status. This keeps users informed and improves communication across the system. The module enhances engagement and reduces missed updates.

CHAPTER 5

IMPLEMENTATION

5.1 SOURCE CODE

APP.PY

```
from flask import Flask, render_template, jsonify, request, redirect, url_for, flash
from flask_login import LoginManager, current_user, login_user, logout_user, login_required
from flask_wtf.csrf import CSRFProtect
import os
import pymysql
from datetime import datetime, timedelta
import json
from models.models import db, User, Event, Registration
from modules.auth import auth_bp
from modules.event import event_bp
from modules.admin import admin_bp
from modules.organiser import organiser_bp
from modules.participant import participant_bp
from modules.payment import payment_bp
from modules.api_organiser import api_organiser_bp
from modules.api_profile import api_profile_bp
from modules.feedback import feedback_bp
app = Flask(name)
app.secret_key = 'cems_secret_key_for_development'
app.config['SQLALCHEMY_DATABASE_URI'] = 'mysql+pymysql://root:1234@localhost/cems_db'
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
login_manager = LoginManager()
login_manager.init_app(app)
login_manager.login_view = 'auth.login'
@login_manager.user_loader
def load_user(user_id):
return User.query.get(int(user_id))

@app.template_filter('fromjson')
def from_json(value):
if value is None:
return {}
try:
return json.loads(value)
except (ValueError, TypeError):
return value
app.register_blueprint(auth_bp)
app.register_blueprint(event_bp)
app.register_blueprint(admin_bp)
app.register_blueprint(organiser_bp)
app.register_blueprint(participant_bp)
app.register_blueprint(payment_bp)
app.register_blueprint(api_organiser_bp)
app.register_blueprint(api_profile_bp)
app.register_blueprint(feedback_bp)
for dir_path in ['static/uploads/images', 'static/uploads/videos', 'static/uploads/brochures']:
if not os.path.exists(dir_path):
os.makedirs(dir_path)
@app.route('/')
def home():
try:
```

```
featured_events = Event.query.order_by(Event.start_date.desc()).limit(6).all()
return render_template('index.html', featured_events=featured_events)
except Exception as e:
return f"Error loading home page: {e}", 500
@app.route('/events')
def events_list():
try:
page = request.args.get('page', 1, type=int)
per_page = 9
category = request.args.get('category', '')
date_filter = request.args.get('date', '')
status = request.args.get('status', '')
search = request.args.get('search', '')
payment_filter = request.args.get('payment', '')
query = Event.query
if category:
query = query.filter_by(category=category)
if payment_filter:
if payment_filter == 'free':
query = query.filter_by(is_free=True)
elif payment_filter == 'paid':
query = query.filter_by(is_free=False)
if date_filter:
today = datetime.now().date()
if date_filter == 'today':
query = query.filter(db.func.date(Event.start_date) == today)
elif date_filter == 'tomorrow':
tomorrow = today + timedelta(days=1)
```

```
query = query.filter(db.func.date(Event.start_date) == tomorrow)
elif date_filter == 'week':
week_end = today + timedelta(days=7)
query = query.filter(db.func.date(Event.start_date) >= today, db.func.date(Event.start_date) &lt;= week_end)
elif date_filter == 'month':
month_end = today + timedelta(days=30)
query = query.filter(db.func.date(Event.start_date) >= today, db.func.date(Event.start_date) &lt;= month_end)
if status:
if status == 'open':
query = query.filter(Event.seats_available > 0)
elif status == 'closed':
query = query.filter(Event.seats_available &lt;= 0)
if search:
search_term = f"%{search}%"
query = query.filter(db.or_(
Event.name.like(search_term),
Event.description.like(search_term),
Event.venue.like(search_term)
))
query = query.order_by(Event.start_date.asc())
events_pagination=query.paginate(page=page, per_page=per_page, error_out=False)
events = events_pagination.items
pagination = {
'page': page,
'pages': events_pagination.pages,
'total': events_pagination.total,
'prev_url': url_for('events_list', page=page-1, category=category, date=date_filter,
status=status, search=search) if events_pagination.has_prev else None,
'next_url': url_for('events_list', page=page+1, category=category, date=date_filter,
status=status, search=search) if events_pagination.has_next else None,
'iter_pages': events_pagination.iter_pages,
'get_url': lambda p: url_for('events_list', page=p, category=category, date=date_filter,
status=status, search=search)
}
current_date = datetime.now().date()1
return render_template('events.html', events=events, pagination=pagination,
current_date=current_date)
except Exception as e:
return f"Error loading events page: {e}", 500
@app.route('/events/int:event_id/register')
@login_required
def register_event(event_id):
if current_user.role != 'participant':
flash('Only participants can register for events', 'error')
return redirect(url_for('event.event_detail', event_id=event_id))
try:
event = Event.query.get_or_404(event_id)
if not hasattr(event, 'is_free') or event.is_free is None:
event.is_free = True
if not hasattr(event, 'price') or event.price is None:
event.price = 0
if event.seats_available &lt;= 0:
flash('This event is sold out', 'error')
return redirect(url_for('event.event_detail', event_id=event_id))
existing_registration = Registration.query.filter_by(
event_id=event_id,
user_id=current_user.id
).first()
if existing_registration:
flash('You are already registered for this event', 'info')
return redirect(url_for('participant.dashboard')
new_registration = Registration(
event_id=event_id,
user_id=current_user.id,
payment_status='pending' if not event.is_free and event.price > 0 else 'completed'
)
event.seats_available -= 1
db.session.add(new_registration)
db.session.commit(
if event.price > 0:
return redirect(url_for('payment.process', registration_id=new_registration.id))
from utils.email_service import send_registration_confirmation
send_registration_confirmation(current_user, event, 'completed')
flash('Successfully registered for the event!', 'success')
return redirect(url_for('participant.dashboard'))
except Exception as e:
db.session.rollback()
flash(f'An error occurred during registration: {str(e)}', 'error')
return redirect(url_for('event.event_detail', event_id=event_id))

@app.route('/preview_registration_theme/&lt;theme>')
def preview_registration_theme(theme):
try:
```

```
available_themes = ['creative', 'elegant', 'minimalist', 'retro', 'tech']
if theme not in available_themes:
return f"Theme '{theme}' is not available.", 404
event_name = request.args.get('event_name', 'Sample Event')
event_description = request.args.get('event_description', 'This is a preview of how the registration form will look with this theme.')
is_preview = request.args.get('is_preview', 'false').lower() == 'true'
event_data = {
'name': event_name,
'start_date': datetime.now().strftime('%Y-%m-%d'),
'start_time': '10:00 AM',
'venue': 'Sample Venue, Location',
'description': event_description,
'organiser': 'Event Organiser',
'price': 500,
'is_free': False
}
context = {
'event': event_data,
'is_preview': is_preview,
'theme_name': theme,
'theme_assets_path': '/static/theme_assets/',
'static_url': '/static'
}
theme_template = f'registration_themes/{theme}.html'
return render_template(theme_template, **context)
except Exception as e:
import traceback
```

```
traceback.print_exc()

return f"Error loading theme: {str(e)}", 500

@app.errorhandler(404)

def not_found(e):

return "Page not found.", 404

@app.errorhandler(500)

def server_error(e):

return f"Internal server error: {e}", 500

with app.app_context():

db.create_all()

if name == 'main':

app.run(debug=True)
```

EMAIL SERVICE.PY

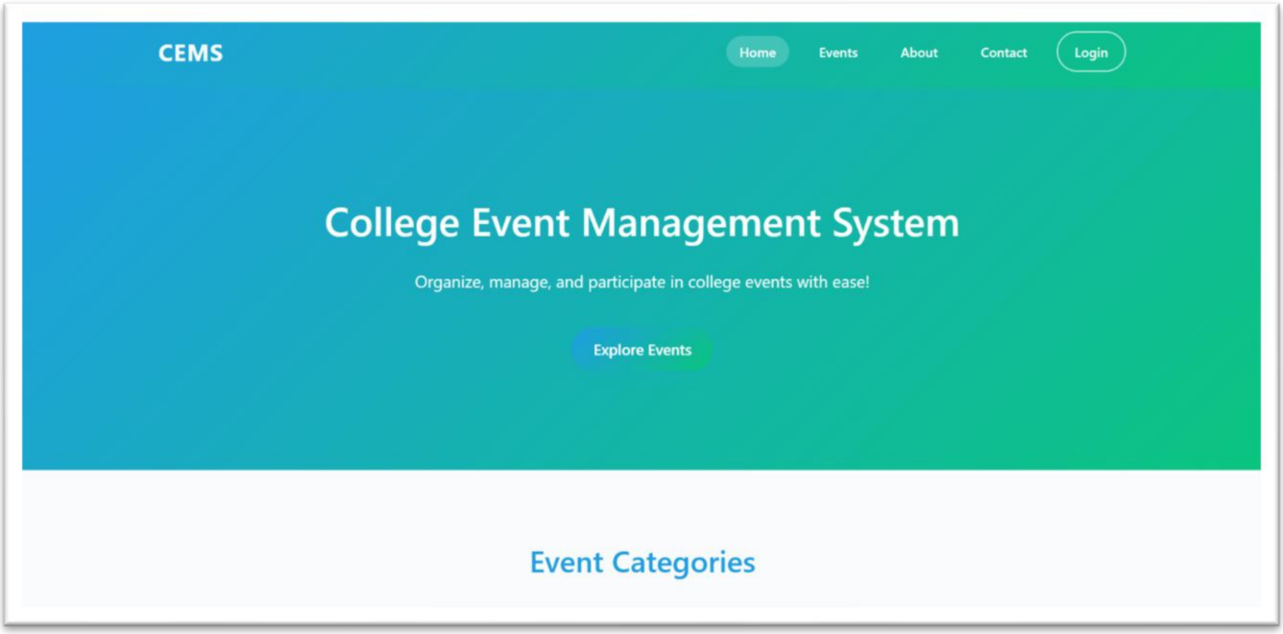
```
import os
import smtplib
import random
import string
from email.mime.text import MIMEText
from email.mime.multipart import MIMEMultipart
from flask import current_app, session
EMAIL_HOST = "smtp.gmail.com"
EMAIL_PORT = 587
EMAIL_USER = "saravanapriyan99940@gmail.com"
EMAIL_PASSWORD = "yzyd rymg hlmd qzqf"
def send_email(to_email, subject, body):
try:
msg = MIMEMultipart()
msg['From'] = EMAIL_USER
msg['To'] = to_email
msg['Subject'] = subject
msg.attach(MIMEText(body, 'html'))
server = smtplib.SMTP(EMAIL_HOST, EMAIL_PORT)
server.starttls()
server.login(EMAIL_USER, EMAIL_PASSWORD)
server.send_message(msg)
server.quit()
print(f"Email sent successfully to {to_email}")
return True
except Exception as e:
print(f"Email error: {str(e)}")
return False
def generate_otp(length=6):
return ''.join(random.choices(string.digits, k=length))
def send_otp_email(email, otp, purpose="verification"):
try:
email_body = f"""
&lt;html>
&lt;head>
&lt;style>
body {{ font-family: Arial, sans-serif; line-height: 1.6; color: #333; }}
.container {{ max-width: 600px; margin: 0 auto; padding: 20px; }}
.header {{ background-color: #3e64ff; color: white; padding: 10px 20px; border-radius: 5px 5px 0 0;
}}
.content {{ padding: 20px; border: 1px solid #ddd; border-top: none; border-radius: 0 0 5px 5px; }}
.otp-box {{ font-size: 24px; letter-spacing: 5px; text-align: center;
background-color: #f5f5f5; padding: 15px; border-radius: 5px; margin: 20px 0; }}
.warning {{ color: #dc3545; font-size: 12px; margin-top: 15px; }}
.footer {{ margin-top: 20px; font-size: 12px; color: #777; }}
&lt;/style>
&lt;/head>
&lt;body>
&lt;div class="container">
&lt;div class="header">
&lt;h2>Your Verification Code&lt;/h2>
&lt;/div>
&lt;div class="content">
&lt;p>Hello,&lt;/p>
&lt;p>You requested a verification code for {purpose}. Please use the following OTP to complete
your process:&lt;/p>

&lt;div class="otp-box">
&lt;strong>{otp}&lt;/strong>
&lt;/div>
```

```
&lt;p>This OTP is valid for 10 minutes.&lt;/p>
&lt;p class="warning">If you didn't request this OTP, please ignore this email or contact support if you believe this is suspicious activity.&lt;/p>
&lt;p>Regards,&lt;br>College Event Management System Team&lt;/p>
&lt;/div>
&lt;div class="footer">
&lt;p>This email was sent to {email}. If you prefer not to receive these emails, please contact support.&lt;/p>
&lt;/div>
&lt;/div>
&lt;/body>
&lt;/html>
"""
return send_email(
email,
"Your OTP Verification Code for CEMS",
email_body
)
except Exception as e:
print(f"OTP email error: {str(e)}")
return False
```

CHAPTER 6

SNAPSHOT

6.1 Home Page

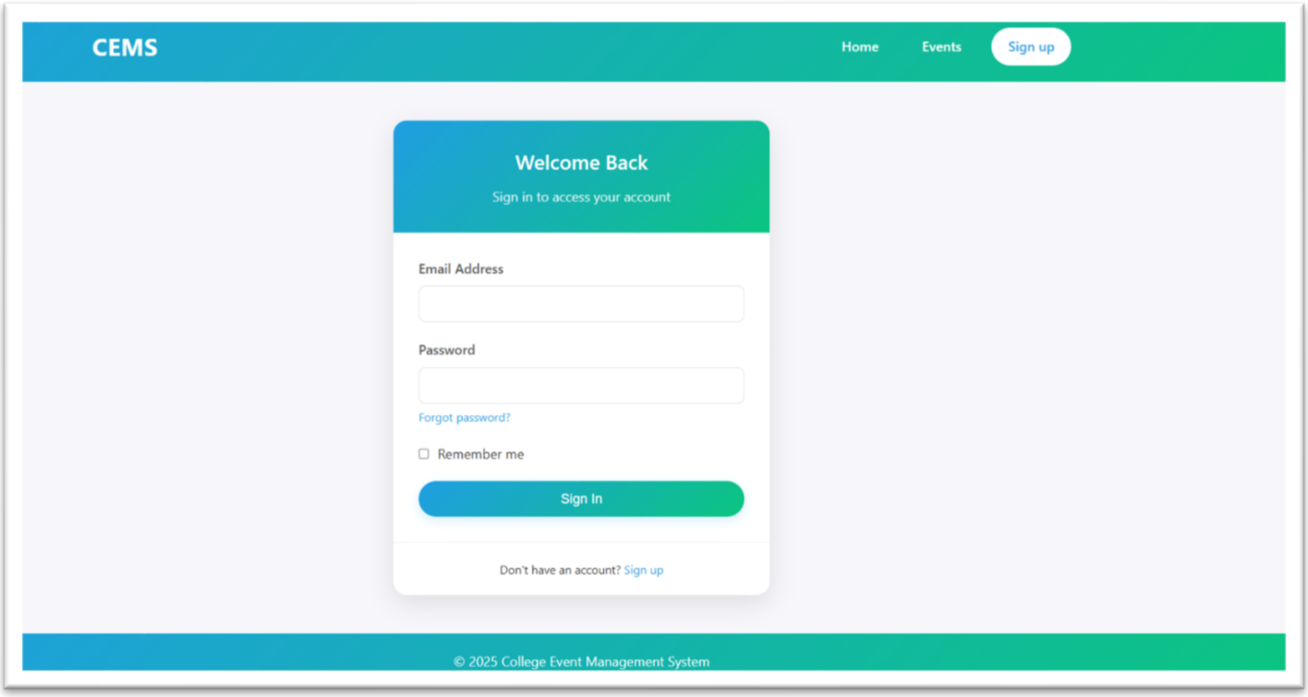


College Event Management System

Organize, manage, and participate in college events with ease!

Event Categories

6.2 Login/Signup Page



Welcome Back

Sign in to access your account

Email Address

Password

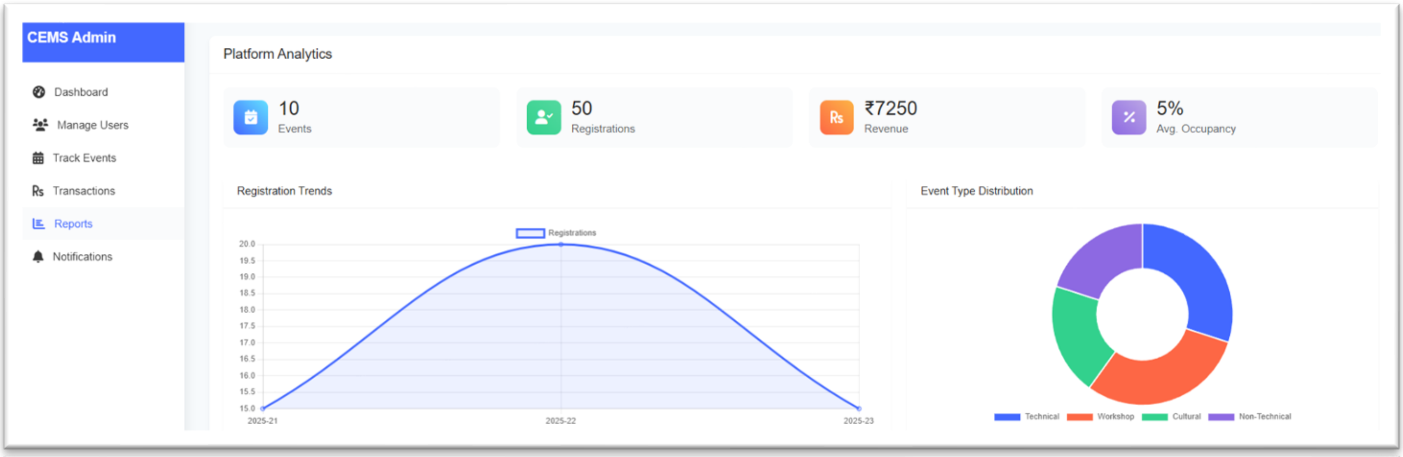
Forgot password?

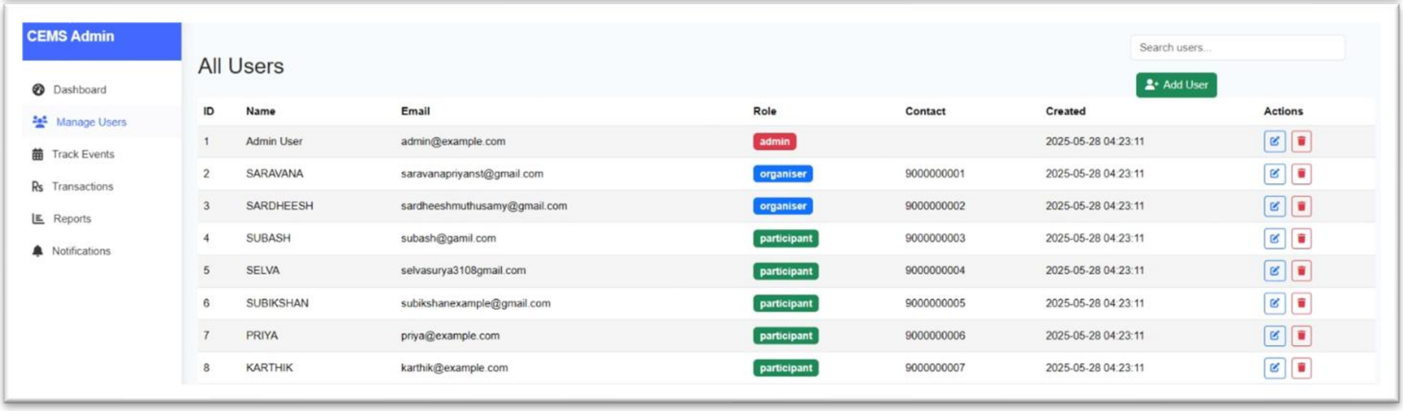
☐ Remember me

Don't have an account? Sign up

© 2025 College Event Management System

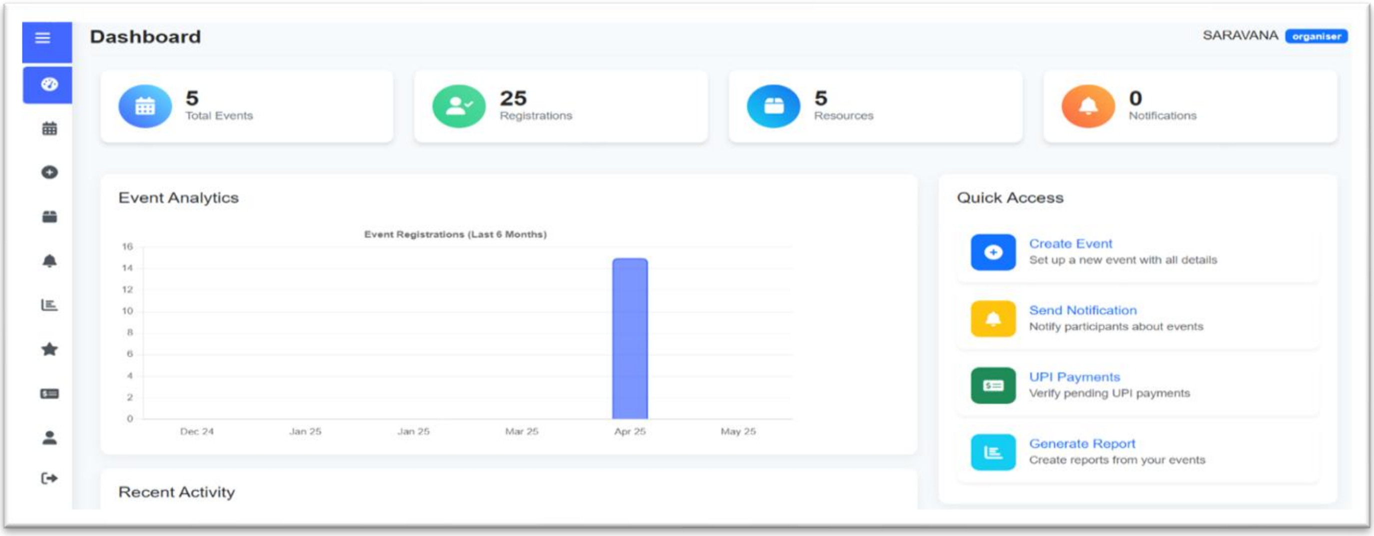
6.3 Admin Page

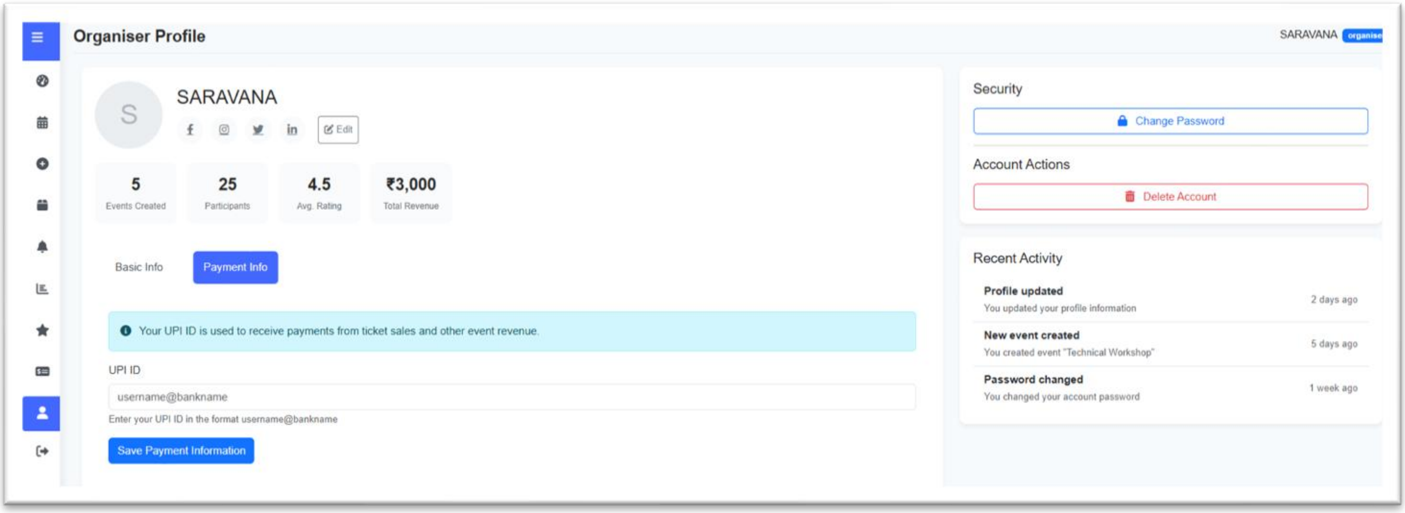




ID	Name	Email	Role	Contact	Created	Actions
1	Admin User	admin@example.com	admin		2025-05-28 04:23:11	
2	SARAVANA	saravanapriyanst@gmail.com	organiser	9000000001	2025-05-28 04:23:11	
3	SARDHEESH	sardheeshmuthusamy@gmail.com	organiser	9000000002	2025-05-28 04:23:11	
4	SUBASH	subash@gamil.com	participant	9000000003	2025-05-28 04:23:11	
5	SELVA	selvasurya3108gmail.com	participant	9000000004	2025-05-28 04:23:11	
6	SUBIKSHAN	subikshanexample@gmail.com	participant	9000000005	2025-05-28 04:23:11	
7	PRIYA	priya@example.com	participant	9000000006	2025-05-28 04:23:11	
8	KARTHIK	karthik@example.com	participant	9000000007	2025-05-28 04:23:11	

6.4Organizer Page





Organiser Profile

SARAVANA organiser

SARAVANA

Edit

5	25	4.5	₹3,000
Events Created	Participants	Avg. Rating	Total Revenue

Basic Info | Payment Info

Your UPI ID is used to receive payments from ticket sales and other event revenue.

UPI ID: username@bankname

Enter your UPI ID in the format username@bankname

Save Payment Information

Security

Change Password

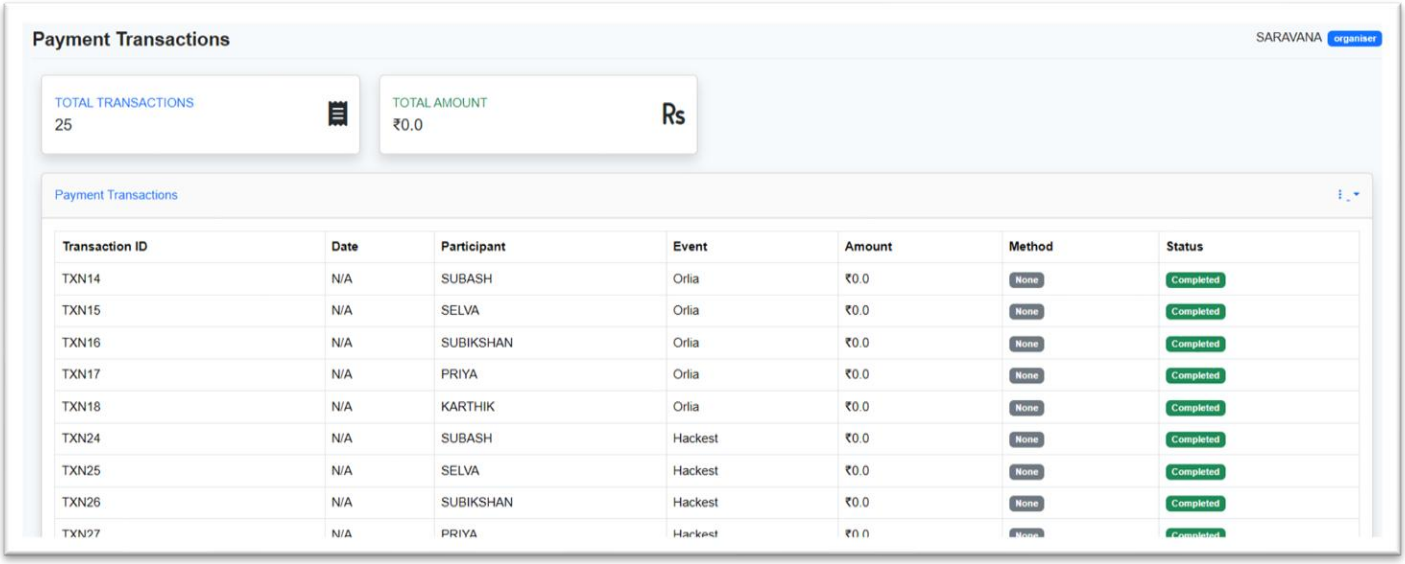
Account Actions

Delete Account

Recent Activity

Profile updated You updated your profile information	2 days ago
New event created You created event "Technical Workshop"	5 days ago
Password changed You changed your account password	1 week ago

Payment Transaction Page



Payment Transactions

SARAVANA organiser

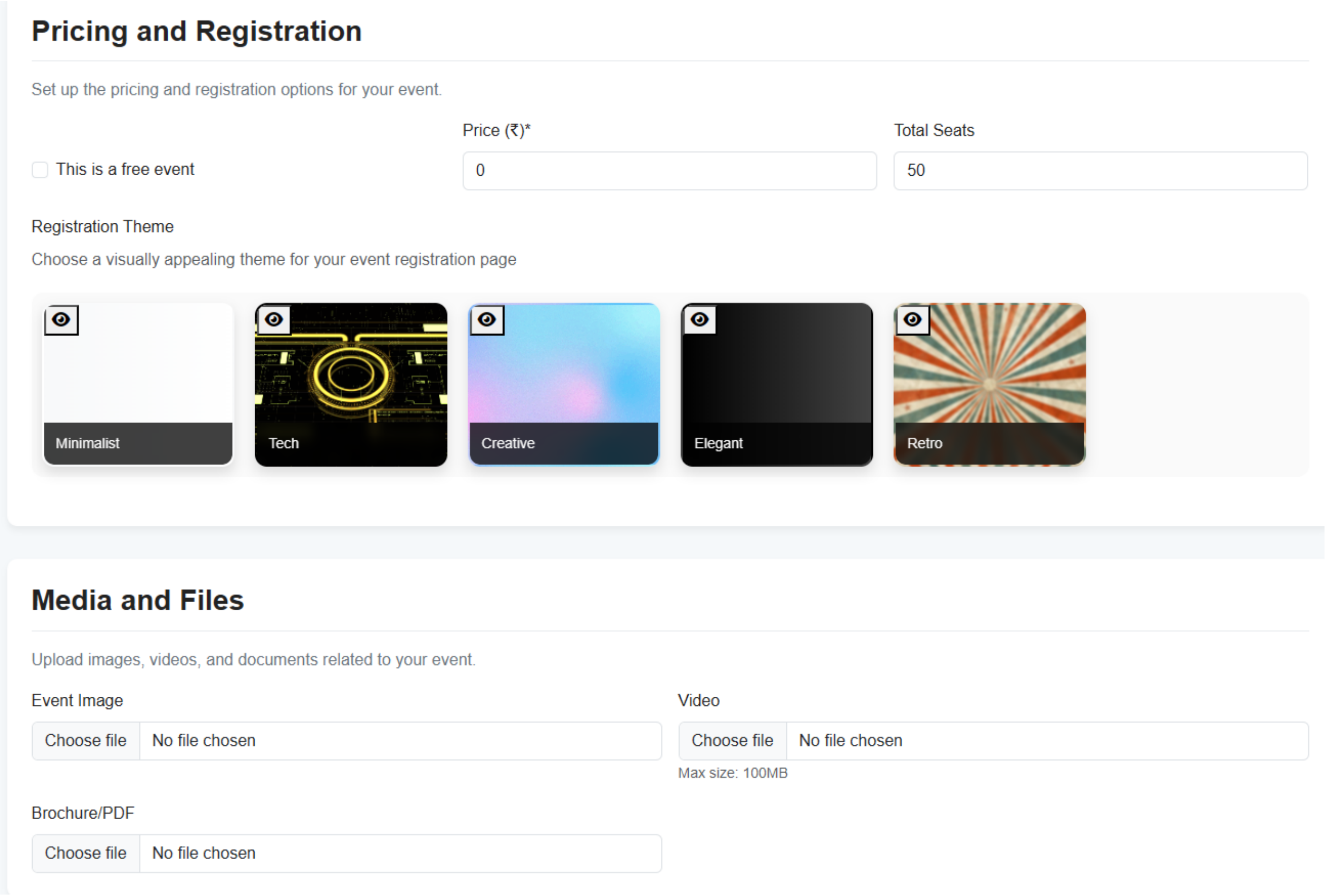
TOTAL TRANSACTIONS: 25

TOTAL AMOUNT: ₹0.0

Payment Transactions

Transaction ID	Date	Participant	Event	Amount	Method	Status
TXN14	N/A	SUBASH	Orlia	₹0.0	None	Completed
TXN15	N/A	SELVA	Orlia	₹0.0	None	Completed
TXN16	N/A	SUBIKSHAN	Orlia	₹0.0	None	Completed
TXN17	N/A	PRIYA	Orlia	₹0.0	None	Completed
TXN18	N/A	KARTHIK	Orlia	₹0.0	None	Completed
TXN24	N/A	SUBASH	Hackest	₹0.0	None	Completed
TXN25	N/A	SELVA	Hackest	₹0.0	None	Completed
TXN26	N/A	SUBIKSHAN	Hackest	₹0.0	None	Completed
TXN27	N/A	PRIYA	Hackest	₹0.0	None	Completed

Event Creation Page



Pricing and Registration

Set up the pricing and registration options for your event.

□ This is a free event

Price (₹)*: 0

Total Seats: 50

Registration Theme

Choose a visually appealing theme for your event registration page

Media and Files

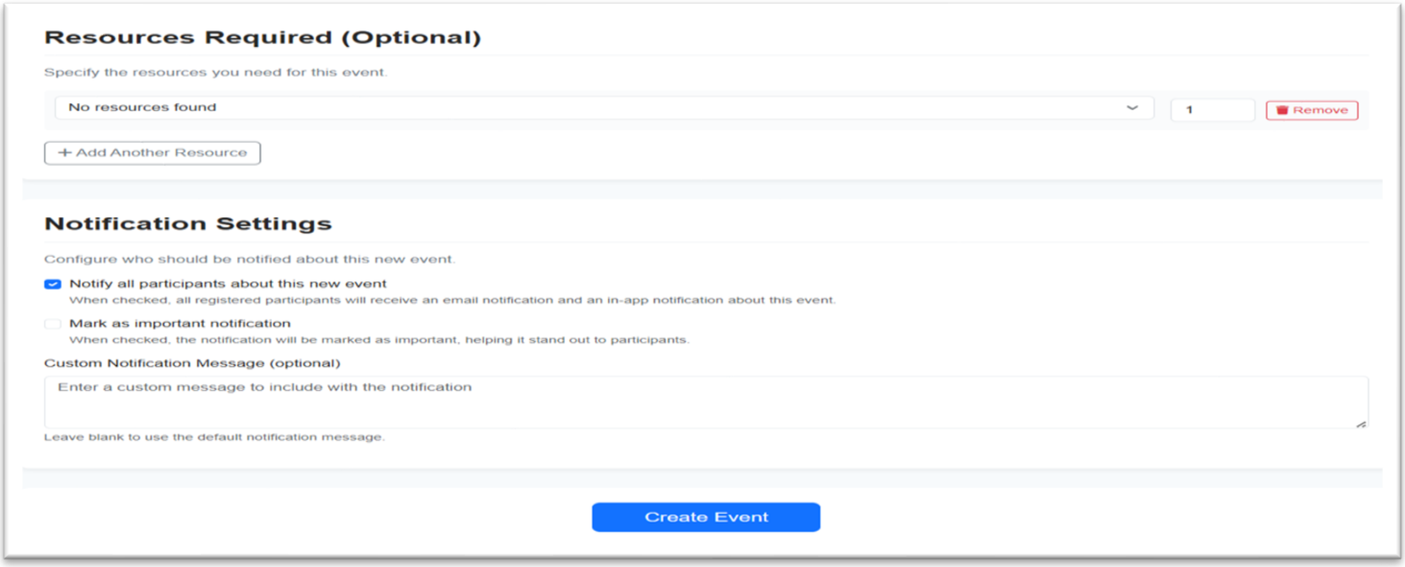
Upload images, videos, and documents related to your event.

Event Image: Choose file | No file chosen

Video: Choose file | No file chosen

Max size: 100MB

Brochure/PDF: Choose file | No file chosen



Resources Required (Optional)

Specify the resources you need for this event.

No resources found | 1 | Remove

Notification Settings

Configure who should be notified about this new event.

☑ Notify all participants about this new event

When checked, all registered participants will receive an email notification and an in-app notification about this event.

□ Mark as important notification

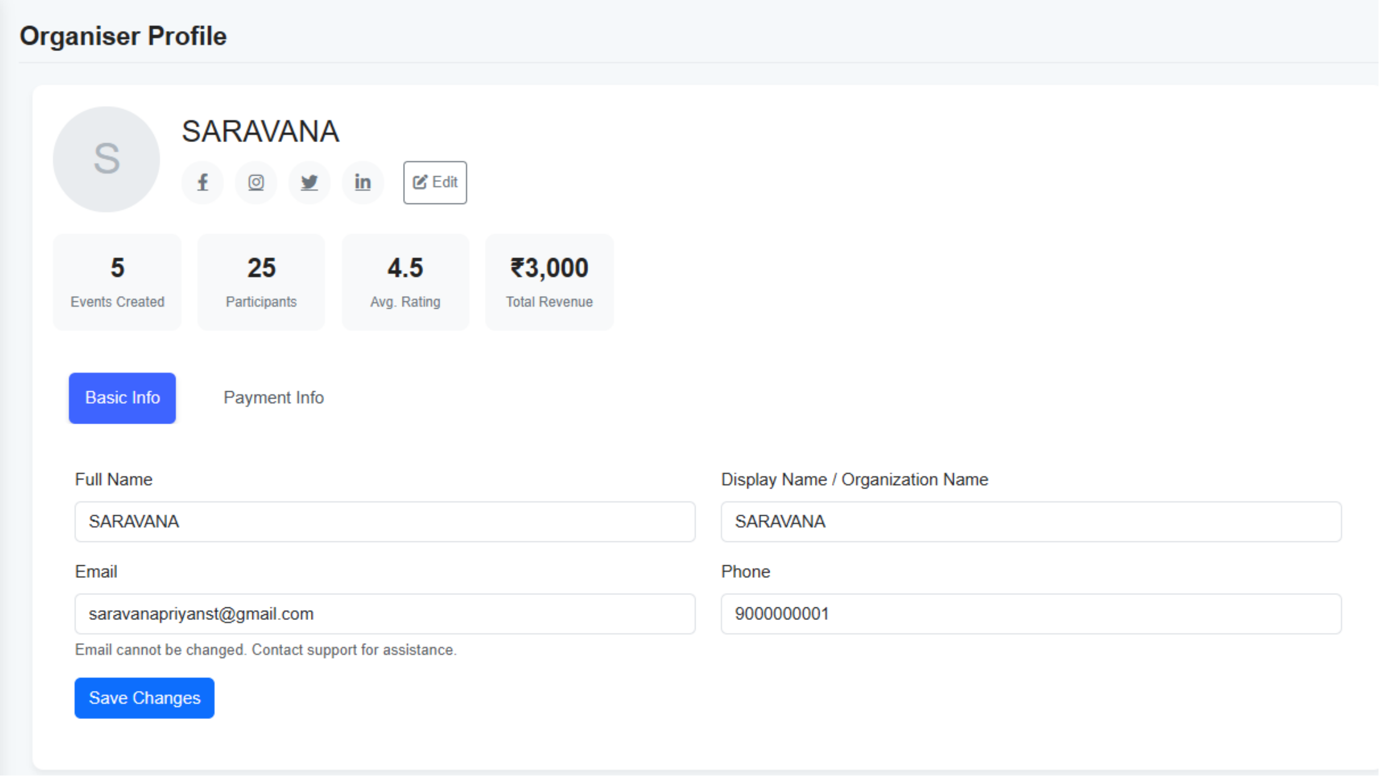
When checked, the notification will be marked as important, helping it stand out to participants.

Custom Notification Message (optional)

Enter a custom message to include with the notification

Leave blank to use the default notification message.

Organiser Profile Page



Organiser Profile

SARAVANA

Edit

5	25	4.5	₹3,000
Events Created	Participants	Avg. Rating	Total Revenue

Basic Info | Payment Info

Full Name: SARAVANA

Display Name / Organization Name: SARAVANA

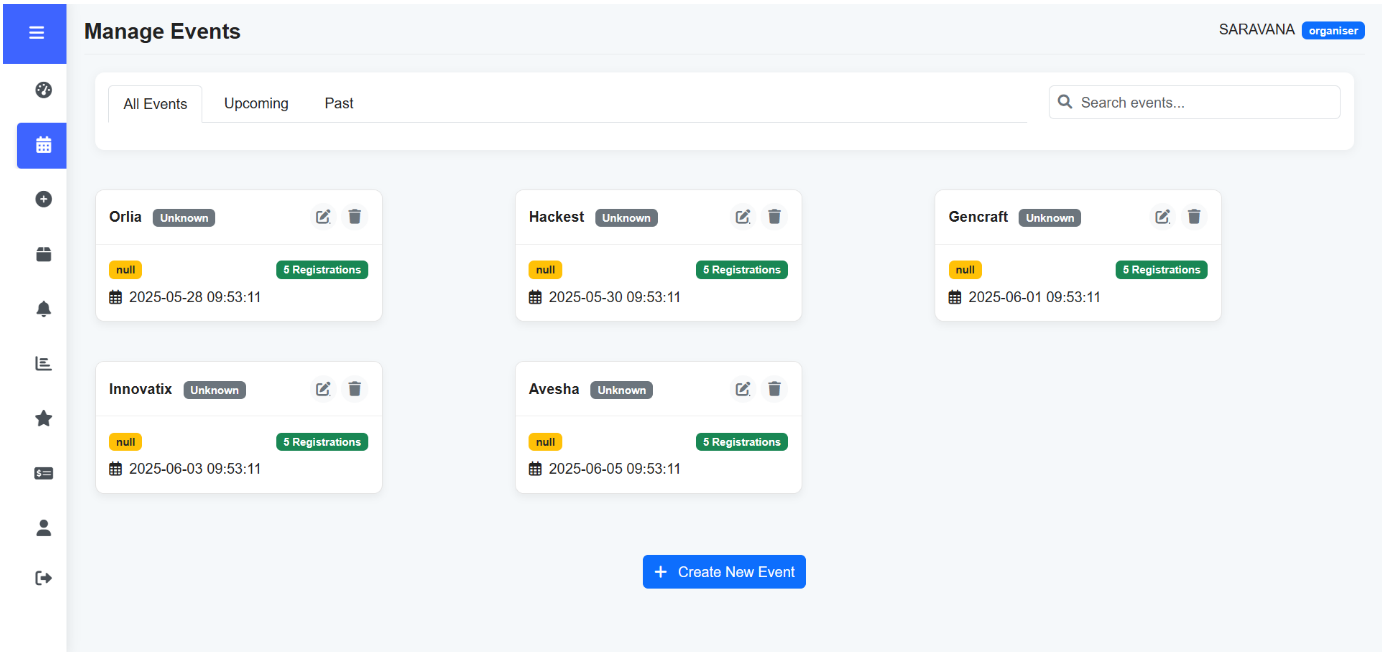
Email: saravanapriyanst@gmail.com

Phone: 9000000001

Email cannot be changed. Contact support for assistance.

Save Changes

Event Management Page



Manage Events

SARAVANA organiser

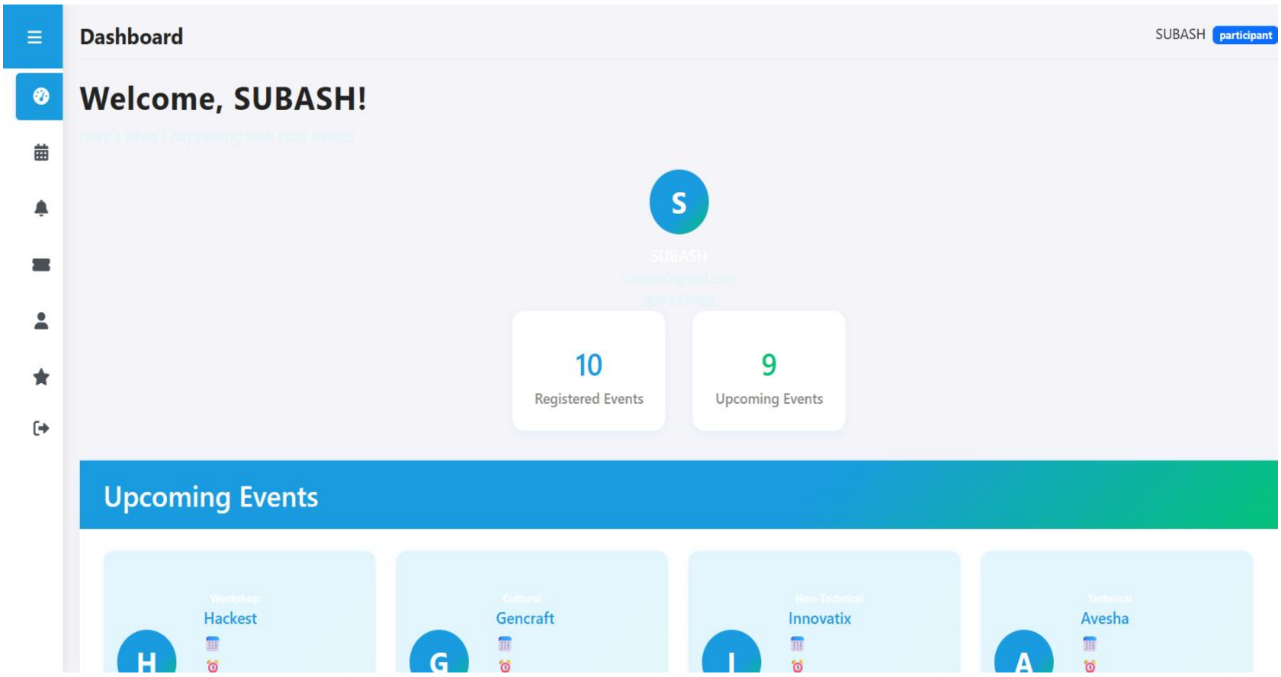
All Events | Upcoming | Past

Search events...

Event	Category	Status	Registrations	Date
Orlia	Unknown	null	5 Registrations	2025-05-28 09:53:11
Hackest	Unknown	null	5 Registrations	2025-05-30 09:53:11
Gencraft	Unknown	null	5 Registrations	2025-06-01 09:53:11
Innovatix	Unknown	null	5 Registrations	2025-06-03 09:53:11
Avesha	Unknown	null	5 Registrations	2025-06-05 09:53:11

+ Create New Event

6.4 Participant Page



Dashboard

SUBASH participant

Welcome, SUBASH!

Here's what's happening with your events

S

SUBASH

subash@gmail.com

9000000000

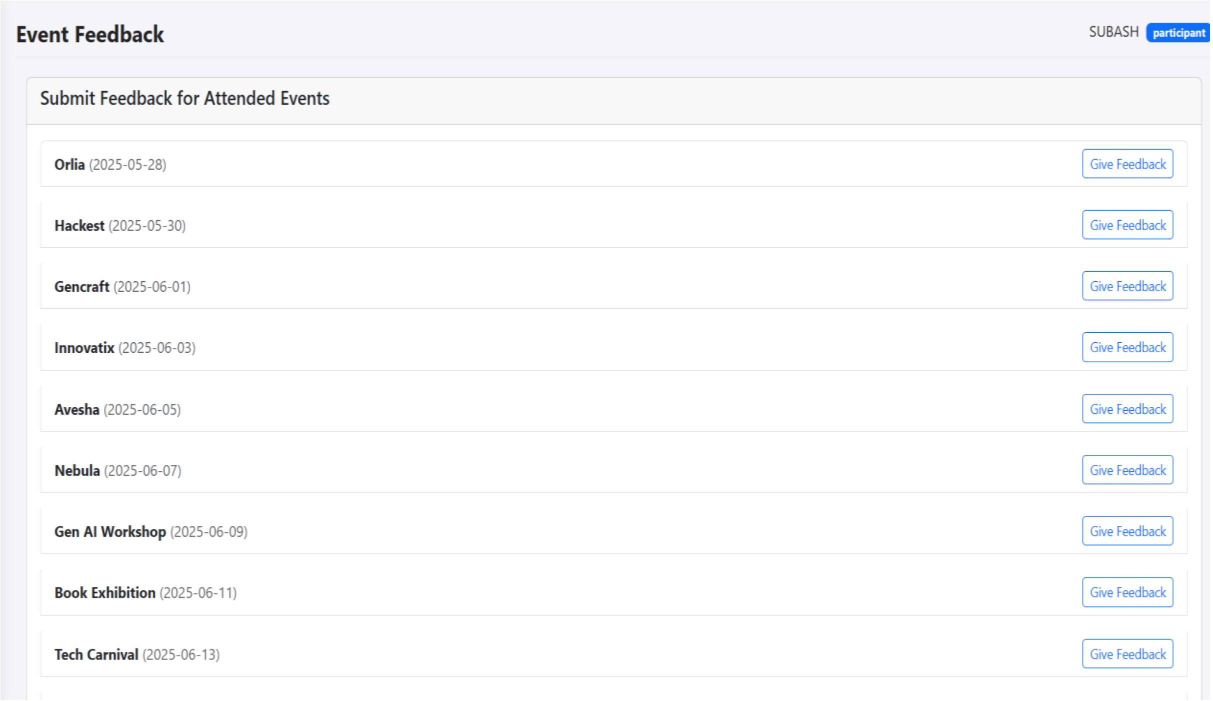
10 Registered Events

9 Upcoming Events

Upcoming Events

Workshop Hackest	Cultural Gencraft	Non-Technical Innovatix	Technical Avesha

Event Feedback page



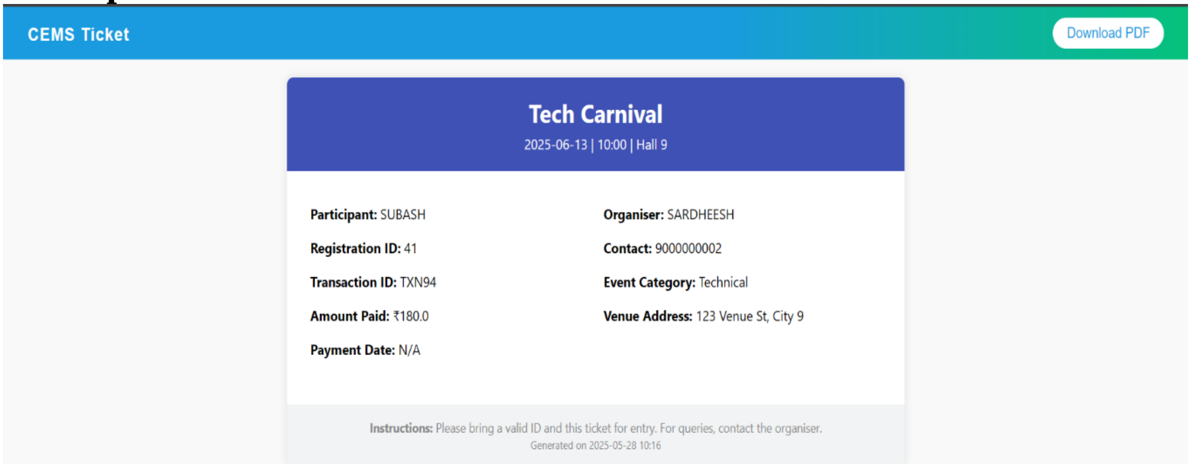
Event Feedback

SUBASH participant

Submit Feedback for Attended Events

Orlia (2025-05-28)	Give Feedback
Hackest (2025-05-30)	Give Feedback
Gencraft (2025-06-01)	Give Feedback
Innovatix (2025-06-03)	Give Feedback
Avesha (2025-06-05)	Give Feedback
Nebula (2025-06-07)	Give Feedback
Gen AI Workshop (2025-06-09)	Give Feedback
Book Exhibition (2025-06-11)	Give Feedback
Tech Carnival (2025-06-13)	Give Feedback

Participant Ticket



CEMS Ticket

Tech Carnival

2025-06-13 | 10:00 | Hall 9

Participant: SUBASH

Registration ID: 41

Transaction ID: TXN94

Amount Paid: ₹180.0

Payment Date: N/A

Organiser: SARDHEESH

Contact: 9000000002

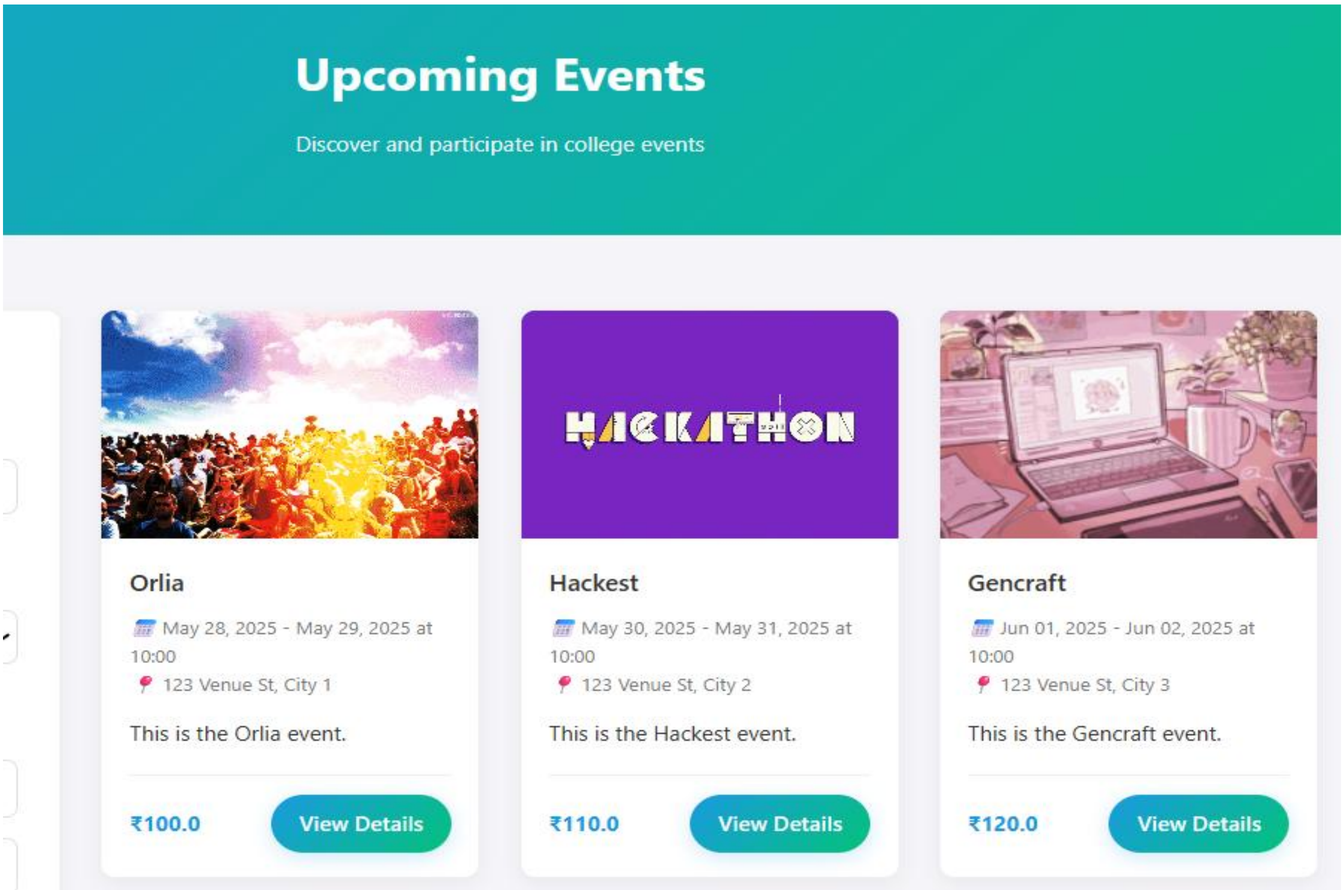
Event Category: Technical

Venue Address: 123 Venue St, City 9

Instructions: Please bring a valid ID and this ticket for entry. For queries, contact the organiser.

Generated on 2025-05-28 10:16

6.5Event Page



Upcoming Events

Discover and participate in college events

Orlia

May 28, 2025 - May 29, 2025 at 10:00

123 Venue St, City 1

This is the Orlia event.

₹100.0

Hackest

May 30, 2025 - May 31, 2025 at 10:00

123 Venue St, City 2

This is the Hackest event.

₹110.0

Gencraft

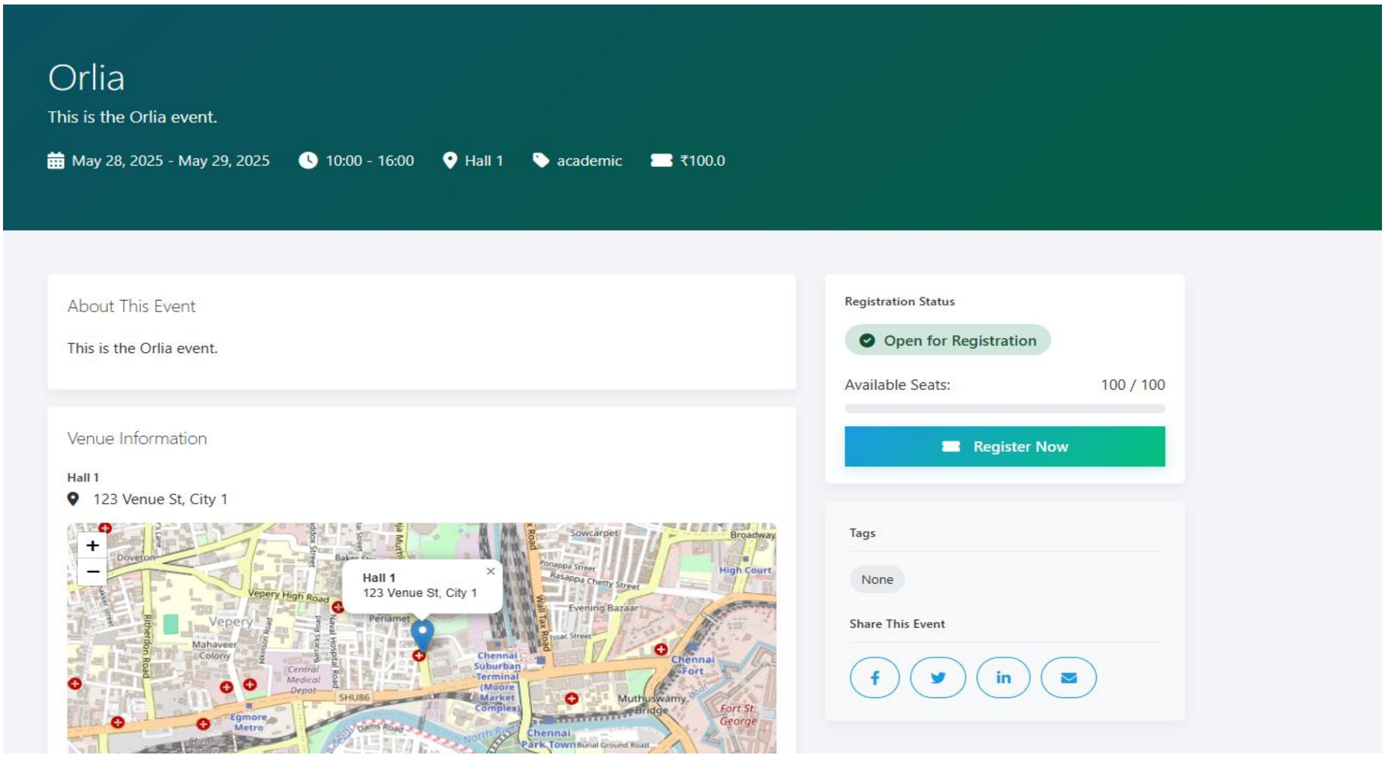
Jun 01, 2025 - Jun 02, 2025 at 10:00

123 Venue St, City 3

This is the Gencraft event.

₹120.0

Event details page



Orlia

This is the Orlia event.

May 28, 2025 - May 29, 2025   10:00 - 16:00   Hall 1   academic   ₹100.0

About This Event

This is the Orlia event.

Venue Information

Hall 1

123 Venue St, City 1

Registration Status

Open for Registration

Available Seats: 100 / 100

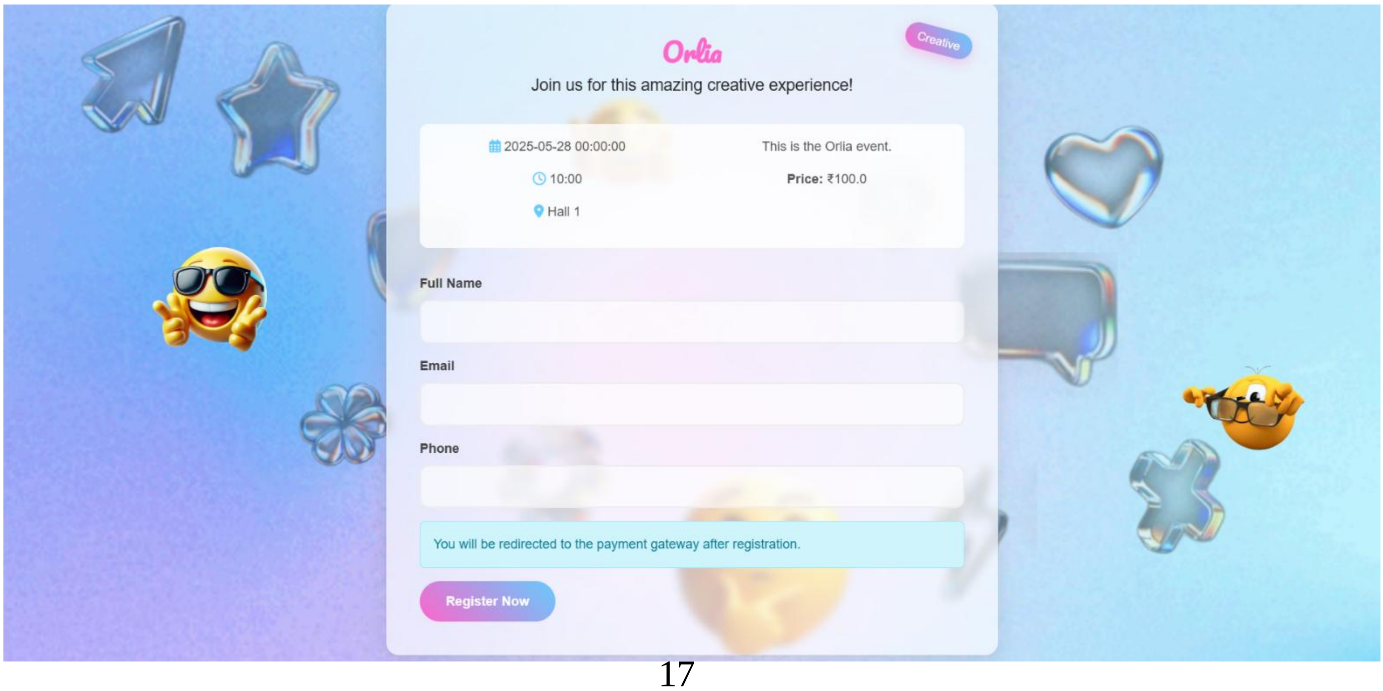
Register Now

Tags

None

Share This Event

Event registration page



Orlia

Creative

Join us for this amazing creative experience!

2025-05-28 00:00:00

10:00

Hall 1

This is the Orlia event.

Price: ₹100.0

Full Name

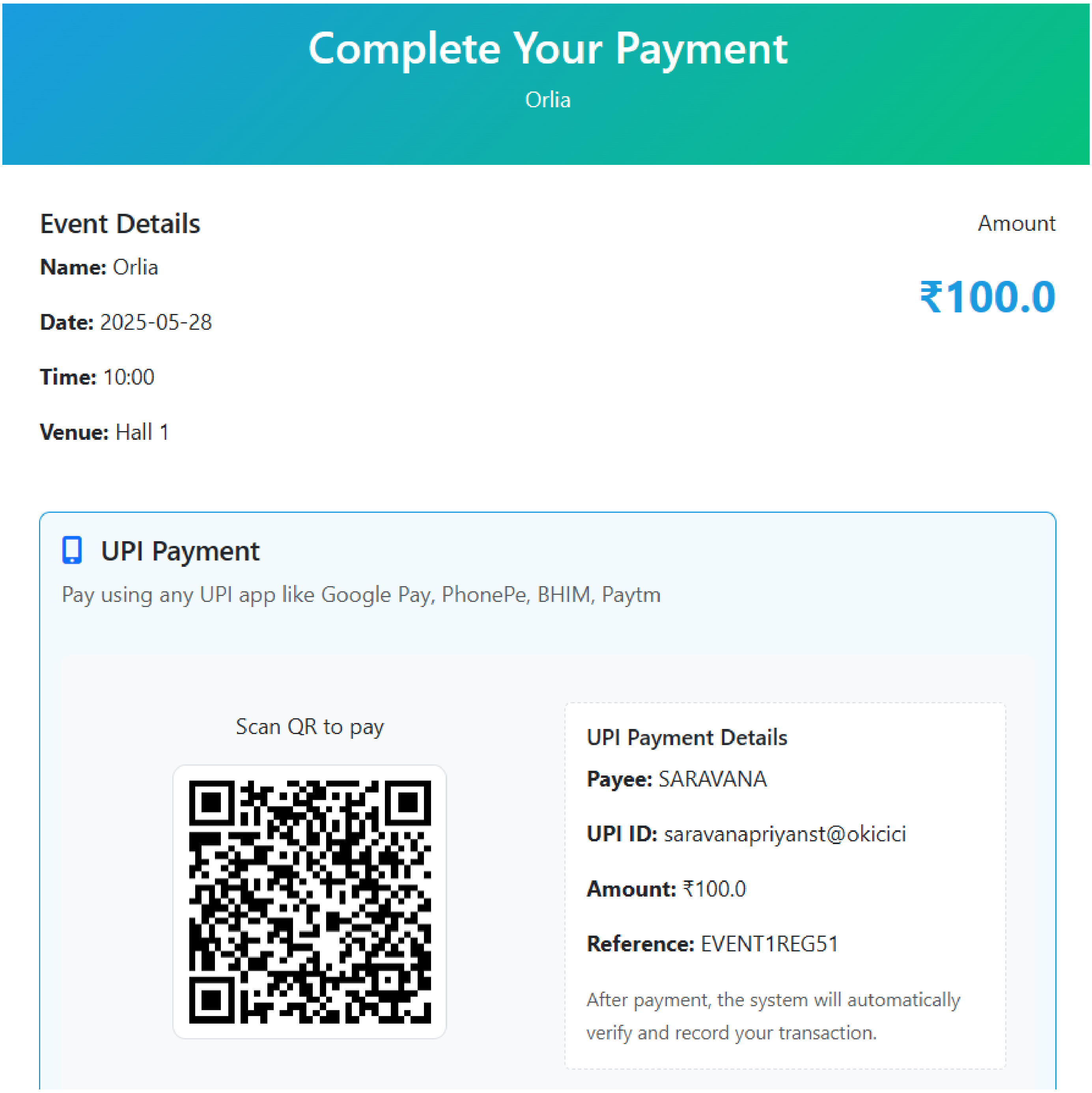
Email

Phone

You will be redirected to the payment gateway after registration.

Register Now

Event Payment page



Complete Your Payment

Orlia

Event Details

Name: Orlia

Date: 2025-05-28

Time: 10:00

Venue: Hall 1

Amount

₹100.0

UPI Payment

Pay using any UPI app like Google Pay, PhonePe, BHIM, Paytm

Scan QR to pay

UPI Payment Details

Payee: SARAVANA

UPI ID: saravanapriyanst@okicici

Amount: ₹100.0

Reference: EVENT1REG51

After payment, the system will automatically verify and record your transaction.

CONCLUSION

The College Event Management System (CEMS) is a fully functional web application developed to simplify the planning, organization, and execution of college events. It offers a secure and user-friendly platform for participants to explore events, register, and receive real-time updates. Admins and organizers can manage events, registrations, payments, and resource allocations through dedicated dashboards. The system applies DBMS principles such as ER modeling, normalization, and SQL queries to maintain data consistency and optimize performance. Role-based access control and encrypted authentication safeguard the platform from unauthorized use. Real-time notifications and analytics improve coordination and decision-making. The UPI payment system ensures secure and flexible transactions for paid events. Built with a modular architecture, CEMS supports scalability and easy maintenance. Each component is designed to enhance user engagement, operational efficiency, and overall system functionality. This project effectively demonstrates how database-driven systems can support complex, real-time applications in the education sector.

REFERENCES

1. https://pubs.aip.org/aip/acp/article/2742/1/020056/3263618/Web-application-of-college-event-management
2. https://ijarsct.co.in/Paper17045.pdf