# EVENTCHRONICLE

**SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS TO AWARD THE DEGREE OF**

**BACHELOR OF COMPUTER SCIENCE (ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING)**

**DESIGNED AND DEVELOPED BY**

Sai Abhinav Goud  Patamata(21241A66J3)

B.Srujan Tej (21241A66E4)

J.Kruthik Reddy (21241A66F6)

T.Karthikeya Manikanta (21241A66J8)

G. Naga Shiva Sai(21241A66F1)

**UNDER THE GUIDANCE OF:**

Ms. M. Shamila

(Asst. professor)

**DEPARTMENT OF**

**COMPUTER SCIENCE (ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING)**

**GOKARAJU RANGARAJU INSTITUTE OF ENGINEERING AND TECHNOLOGY** (Bachupally, Hyderabad,500090)

Signature of Faculty                                        Signature of HOD

**INDEX**

## Problem Statement

In today's dynamic educational environment, college students frequently miss valuable opportunities to engage in various events, both technical and non-technical, due to a pervasive lack of awareness. This lack of awareness often stems from the students either forgetting about upcoming events or struggling to keep track of the numerous activities happening on campus. Additionally, students may not be fully aware of the diverse range of clubs and organizations within their college. This situation poses challenges not only for students but also for faculty members who find it difficult to monitor students' participation in various projects and events.

## Goal of the Project

To address this multifaceted problem, we propose the development of a comprehensive solution called "EventChronicle." This innovative project aims to bridge the gap in communication and awareness within the college community by offering a centralized platform where students can easily access information about upcoming events, discover the diverse range of clubs and organizations available, and efficiently manage their participation. EventChronicle will also provide faculty members with a convenient tool to track and verify student involvement in various events and projects, promoting a more engaged and informed campus community.

EventChronicle strives to enhance the overall college experience by fostering better communication, awareness, and participation among students, clubs, and faculty. With this project, we aim to create a cohesive and well-informed college environment where no student misses out on valuable opportunities, and faculty can better support and assess student engagement in various academic and extracurricular activities.

## INTRODUCTION:

Model-View-Template (MVT) in Django:

**Model**

- Represents data and database structure.
- Defines how data is stored and managed.
- Includes Python classes for database tables.
- Handles data operations like queries and updates.

**View**

- Processes user requests.
- Retrieves data from the Model.
- Defines logic for HTTP request handling.
- Maps URLs to specific views.

**Template**

- Focuses on the presentation layer.
- Contains HTML files with placeholders.
- Dynamically generates web pages.
- Separates design from content.

MVT in Django separates data (Model), request handling (View), and web page presentation (Template) for organized and maintainable web development.

**User authentication in Django:**

The Django authentication system handles both authentication and authorization. Briefly, authentication verifies a user is who they claim to be, and authorization determines what an authenticated user is allowed to do. Here the term authentication is used to refer to both tasks.

The auth system consists of:

- Users

- Permissions: Binary (yes/no) flags designating whether a user may perform a certain task.

**Authentication support** is bundled as a Django contrib module in django.contrib.auth. By default, the required configuration is already included in the settings.py generated by django-admin startproject, these consist of two items listed in your INSTALLED_APPS setting:

- 'django.contrib.auth' contains the core of the authentication framework, and its default models.
- 'django.contrib.contenttypes' is the Django content type system, which allows permissions to be associated with models you create.

The Google Calendar API is a robust and versatile RESTful API designed to facilitate seamless integration with Google Calendar functionalities. Developers can interact with the API through explicit HTTP requests or by utilizing the Google Client Libraries. This API exposes a comprehensive set of features that mirror those available within the Google Calendar web interface:

**Purpose:** The primary purpose of the Google Calendar API is to empower developers to seamlessly incorporate Google Calendar features into their applications, enhancing their functionality and utility.

**Authentication:** To ensure the security of calendar data, developers utilize OAuth 2.0, a robust and industry-standard authentication protocol, to access calendar information securely.

**API Endpoints:** The API provides a wide range of endpoints, enabling developers to create, read, update, and delete events, manage calendars, and perform various other operations to manipulate calendar data as needed.

**Data Format:** The Google Calendar API employs JSON (JavaScript Object Notation) as the standard format for data exchange between applications and Google Calendar, ensuring compatibility and ease of data manipulation.

**Use Cases:** This API is frequently employed in the development of applications related to calendar integrations, scheduling applications, and reminder systems, among others, where calendar functionality plays a critical role.

**Libraries/SDKs:** To facilitate development across different programming languages and platforms, Google offers a rich selection of client libraries and Software Development Kits (SDKs), streamlining the integration process and reducing development time.

**Access Control:** Developers have the ability to define permissions within their applications, allowing users to grant access to their calendar data. This ensures data privacy and user control.

**Security:** Adhering to the highest industry security standards, the Google Calendar API employs HTTPS encryption, safeguarding data transmission between applications and the Google Calendar servers.

**Rate Limiting:** To maintain the stability and performance of the API, it enforces rate limits on usage, preventing excessive requests and ensuring fair access for all developers.

**Documentation:** Google provides extensive and well-documented resources, including guides, reference materials, and code samples, to support developers in effectively utilizing the Google Calendar API, making the integration process more accessible and efficient.

## Advantages:

1. **Convenient Event Registration:** EventChronicle facilitates seamless event registration through the use of Google Forms, a widely adopted and user-friendly tool. This streamlines the registration process for students, saving them time and effort, while also ensuring that all necessary information is efficiently collected.
2. **Flexible Event Management:** EventChronicle offers flexibility to tailor the project to your college's unique requirements. You have the freedom to customize features, add or remove functionalities as needed, and even reorganize the project structure to suit your preferences.
3. **Scalable Event Management:** With its scalability, EventChronicle can efficiently handle a large volume of events. This scalability is particularly valuable for colleges that frequently host numerous events or seek to expand their event offerings in the future.

4. **Secure Event Management:** EventChronicle prioritizes data security, implementing industry-standard security practices to safeguard user data. This commitment to security ensures that students can trust their personal information is protected.
5. **Open Source Event Management:** Being an open-source project, EventChronicle is accessible to a broader community of developers. This openness allows for collaboration and contributions from the Django community, providing valuable support options and opportunities for customization.
6. **Centralized Calendar of Events:** EventChronicle offers a centralized platform where students can effortlessly access information about all campus events, including event details. This centralized calendar assists students in making informed decisions about which events to attend.
7. **Dashboard for Faculty Members:** Faculty members benefit from a dedicated dashboard within EventChronicle. This tool enables them to monitor event registrations for events they organize, ensuring that sufficient seating is available to accommodate interested students.
8. **Tool for Event Managers:** Event managers can efficiently manage events within EventChronicle, with features for adding, editing, and deleting events. This functionality is invaluable for event managers who need to adjust schedules or cancel events as required.
9. **Automated Event Registration:** EventChronicle simplifies event registration with one-click registration options, reducing the time and effort required for students to sign up. This automation also minimizes registration errors, improving the overall user experience.

## Limitations:

1. **Monolithic Framework:** EventChronicle relies on a monolithic framework, which means that changes made in one part of the project may have unintended consequences in other areas. This may pose challenges when attempting to customize the project to meet specific college needs.
2. **Limited Flexibility Compared to Other Platforms:** In comparison to some other event management platforms, EventChronicle may offer less flexibility in terms of feature customization and project organization. This limitation could impact the adaptability of the platform.

3. **Development Time and Effort:** Implementing EventChronicle may demand a significant amount of development time and effort, particularly for individuals not familiar with Django or similar technologies.
4. **Suitability for Specific Colleges:** EventChronicle's suitability may vary based on the specific needs and requirements of individual colleges. Not all institutions may find the platform aligns perfectly with their unique circumstances.
5. **User-Friendliness:** While EventChronicle offers robust event management capabilities, it may not be as intuitive or user-friendly as some alternative event management platforms. This could potentially lead to challenges for both students and event managers in navigating and utilizing the system effectively.

## Code :

```
1.  @login_required

def admindashboard(request):

    return render(request,'admindashboard.html')

#add event

def addevent(request):

    if request.method=="POST":

        title= request.POST.get('title')

        location= request.POST.get('location')

        Description= request.POST.get('Description')

        stimedate= request.POST.get('sdate')+'T'+request.POST.get('stime')+':00+05:30'

        etimedate= request.POST.get('edate')+'T'+request.POST.get('etime')+':00+05:30'

        addtocalendar(request,title,location,Description,stimedate,etimedate)

        print(stimedate+" "+etimedate)
    return render(request,'addevent.html')

#calendar

SCOPES = ['https://www.googleapis.com/auth/calendar']

def addtocalendar(request,title,location,Description,stime,etime):

    """Shows basic usage of the Google Calendar API.

    Prints the start and name of the next 10 events on the user's calendar.

    """

    creds = None

    # The file token.json stores the user's access and refresh tokens, and is

    # created automatically when the authorization flow completes for the first

    # time.

    if os.path.exists('token.json'):

        creds = Credentials.from_authorized_user_file('token.json', SCOPES)

    # If there are no (valid) credentials available, let the user log in.

    if not creds or not creds.valid:

        if creds and creds.expired and creds.refresh_token:

            creds.refresh(Request())

        else:

            flow = InstalledAppFlow.from_client_secrets_file(

                'adminsignup\credentials.json', SCOPES)

            creds = flow.run_local_server(port=0)

        # Save the credentials for the next run

        with open('token.json', 'w') as token:

            token.write(creds.to_json())

    try:

        service = build('calendar', 'v3', credentials=creds)

        event = {
```

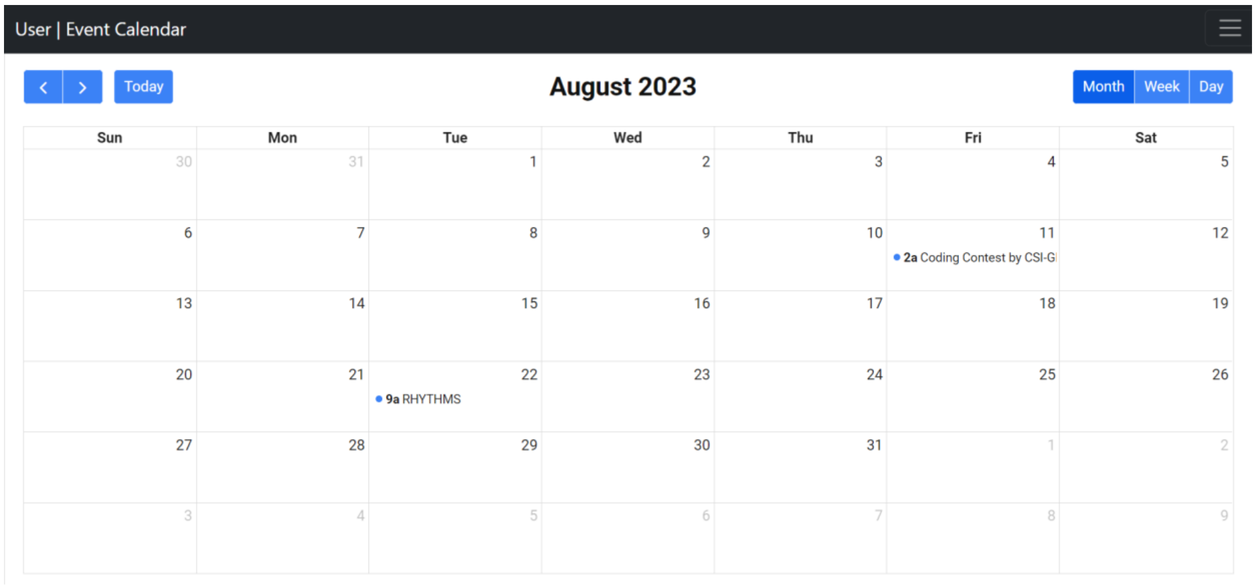
```
    'summary': title,

    'location': location,

    'description': Description,

    'start': {

      'dateTime': stime,

      'timeZone': 'Asia/Kolkata',

    },

    'end': {

      'dateTime': etime,

      # '2023-07-15T17:23:00-07:00'

      'timeZone': 'Asia/Kolkata',

    },

    'recurrence': [

      'RRULE:FREQ=DAILY;COUNT=1'

    ],

    'attendees': [

      {'email': 'lpage@example.com'},

      {'email': 'sbrin@example.com'},

    ],

    'reminders': {

      'useDefault': False,

      'overrides': [

        {'method': 'email', 'minutes': 24 * 60},

        {'method': 'popup', 'minutes': 10},

      ],

    },

  }
```

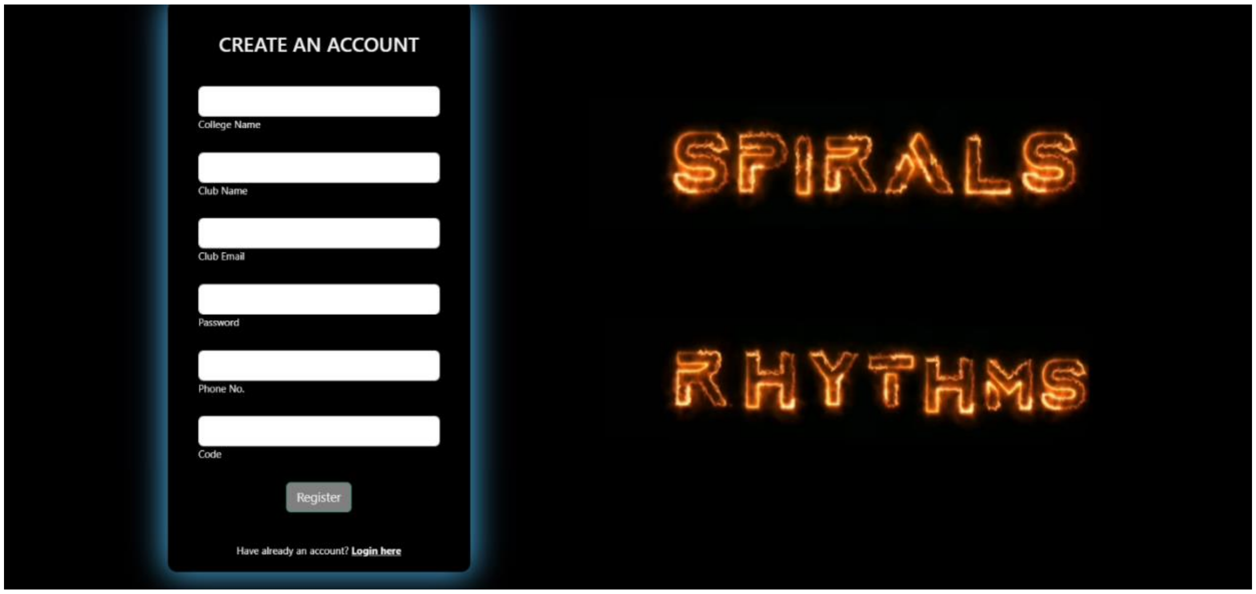
```
        event = service.events().insert(calendarId='primary', body=event).execute()

        messages.success(request, "Event created")

        print ('Event created: %s' % (event.get('htmlLink')))

    except HttpError as error:

        print('An error occurred: %s' % error)

        messages.success(request, "Error")
```

## GitHub Link:

https://github.com/SaiAbhinavGoud/Event-Calendar

## Result




CREATE AN ACCOUNT

College Name

Club Name

Club Email

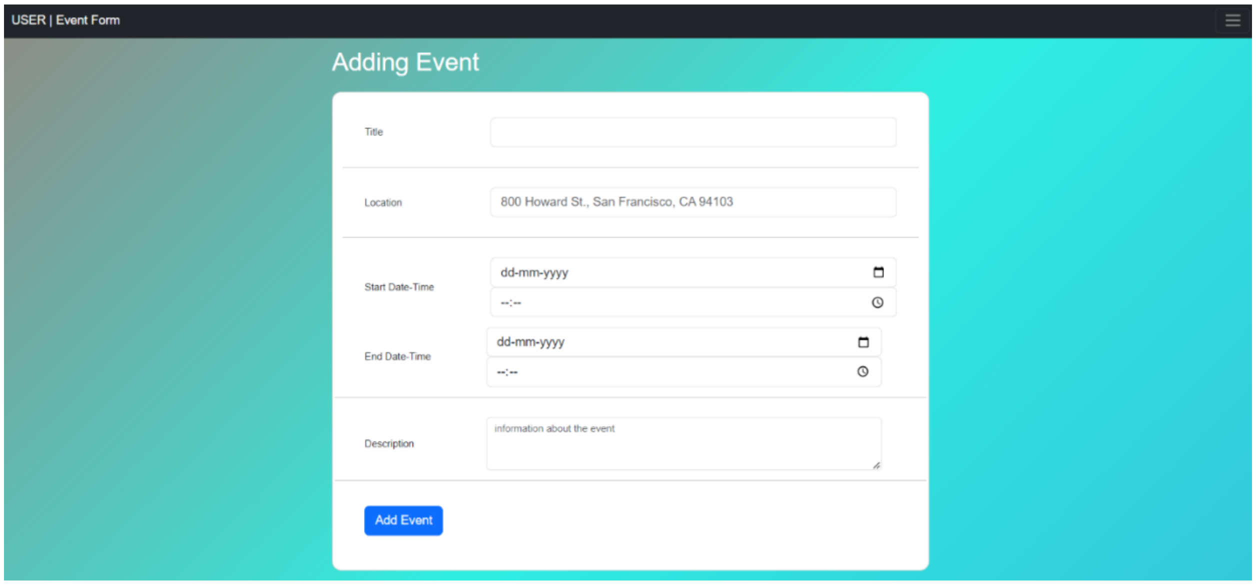
Password

Phone No.

Code

Register

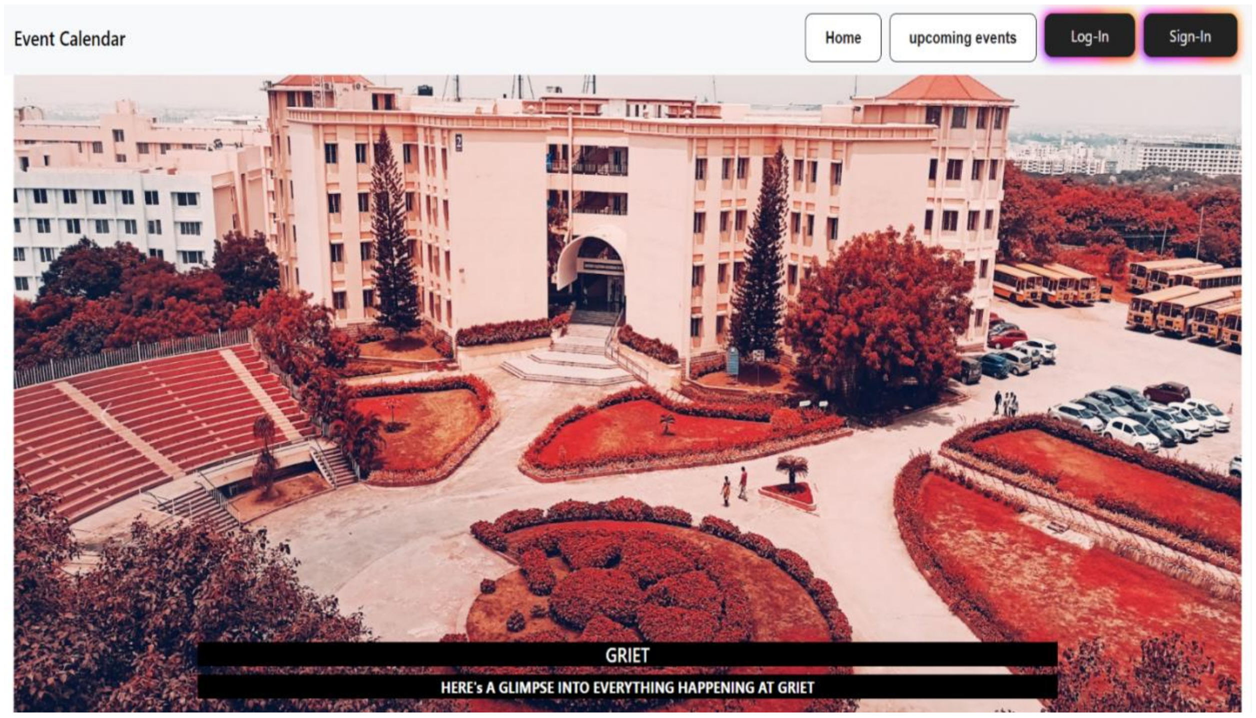
Have already an account? **Login here**


USER | Event Form

Adding Event

| | |
|---|---|
| Title | |
| Location | 800 Howard St., San Francisco, CA 94103 |
| Start Date-Time | dd-mm-yyyy<br>--:-- |
| End Date-Time | dd-mm-yyyy<br>--:-- |
| Description | information about the event |

Add Event



## CASE STUDY:

### Case Study 1:

Event Chronicle is a college event calendar web app designed to connect students, faculty, and staff with upcoming events and activities happening on campus. The app aims to centralize event information, streamline event management, and promote community engagement within the college campus. This case study highlights the development and success of Event Chronicle.

### Challenges:

1. Information Fragmentation: Before Event Chronicle, event information was scattered across various platforms, including bulletin boards, email newsletters, and social media, making it challenging for students to find and keep track of events.

2. Event Management: The college lacked an efficient system to manage event submissions, approvals, and scheduling, resulting in potential scheduling conflicts and administrative inefficiencies.

**Solution:**

1. Centralized Event Calendar: Event Chronicle provides a user-friendly, centralized event calendar that aggregates event details from different sources, including student organizations, departments, and clubs. Users can access it via a website.

2. User-Friendly Event Submission: Event organizers can easily submit event details through the app, which are then reviewed and approved by college administrators. A user-friendly interface simplifies the process, reducing the workload on administrative staff.

3. Personalized User Profiles: Event Chronicle offers personalized user profiles where students can select their interests and receive event recommendations tailored to their preferences. This enhances user engagement and attendance.

4. Integration with College Systems: The app integrates with the college's authentication system, ensuring that only students, faculty, and staff can access event information and features. It also integrates with college email systems for event notifications.

**Results:**

1. Increased Event Attendance: Event Chronicle has significantly increased event attendance rates as it offers an easily accessible and organized event calendar. Students are more informed about campus activities.
2. Streamlined Event Management: College administrators have reported reduced workload and fewer scheduling conflicts due to the streamlined event submission and approval process.
3. Enhanced Engagement: Users report feeling more engaged with campus life, thanks to personalized event recommendations and the ability to connect with others who share their interests.
4. Positive Feedback: Event Chronicle has received positive feedback from students, faculty, and staff for its ease of use and its contribution to campus community-building.

**Case Study 2:**

" Event Chronicle " - The College Event Calendar Web App Project Overview: Event Chronicle is a college event calendar web app designed to provide comprehensive event details to college students. It aims to bridge the communication gap between student organizations, academic departments, and the student body by offering a user-friendly platform for event discovery. This case study explores the development and success of Event Chronicle.

**Challenges:**

1. Event Visibility: College events were often under-publicized, leading to low attendance and a lack of engagement among students.
2. Coordination: Coordinating events across various student organizations and departments was challenging, leading to scheduling conflicts and event duplication.

**Solution:**

1. Comprehensive Event Listings: Event Chronicle aggregates event details from student organizations and academic departments, providing a one-stop solution for students to discover upcoming events.

2. Event Promotion: The app offers event promotion tools, including social media sharing and email notifications, to ensure events receive the attention they deserve.

3. Event Analytics: Event Chronicle provides event organizers with analytics tools to track attendance, engagement, and feedback, allowing for data-driven event planning.

4. Event Collaboration: The app facilitates collaboration between different college entities by allowing them to co-host events and share resources.

**Results:**

1. Data-Driven Decisions: Event organizers use the app's analytics tools to make data-driven decisions about event planning and promotion, leading to more successful events.
2. College Branding: The app's customization options allow colleges to integrate their branding and messaging, creating a seamless and visually appealing user experience.

## Conclusion:

In conclusion, Event Chronicle demonstrates the potential of a college event calendar web app to enhance campus life by improving event visibility, coordination, and engagement among students and other stakeholders. These case studies showcase successful solutions to common challenges faced by colleges in managing and promoting events.