EVENT PLANNING

A

Mini Project Report

Submitted in partial fulfillment of the

Requirements for the award of the Degree of

BACHELOR OF ENGINEERING

IN

INFORMATION TECHNOLOGY

By

AKHIL THAKUR-1602-19-737-064

B.NIKHITA - 1602-19-737-084

M. TANMAYEE– 1602-19-737-119

Under Guidance of

Dr. Kezia Rani

Department of Information Technology

Vasavi College of Engineering (Autonomous)

(Affiliated to Osmania University)

Ibrahimbagh, Hyderabad - 500 031

2021-2022

Vasavi College of Engineering (Autonomous)

(Affiliated to Osmania University)

Hyderabad - 500 031

2021-2022

Department of Information Technology

DECLARATION BY THE CANDIDATE

We, Akhil Thakur, B.Nikhita, M.Tanmayee bearing hall ticket numbers, 1602-19-737-064, 1602-19-737-19-084 and 1602-19-737-119 respectively, hereby declare that the project report entitled EVENT PLANNING is submitted in fulfillment of the requirement for the award of the degree of Bachelor of Engineering in Information Technology.

This is a record of bonafide work carried out by us and the results embodied in this project report have not been submitted to any other university or institute for the award of any other degree or diploma.

AKHIL THAKUR 1602-19-737-064

B.NIKHITA 1602-19-737-084

M. TANMAYEE 1602-19-737-119

(Faculty In-Charge) (Head, Dept. Of IT)

ACKNOWLEDGEMENT

Our Mini Project would not have been successful without the help of several people. We are extremely thankful to our college, **Vasavi College of Engineering, Hyderabad** for providing the opportunity to implement our project, **“EVENT PLANNING”**.

We would like to express our gratitude to Dr. Kezia Rani, Mrs.Haseeba Yaseen, Mr.Mukesh Tripathi, Department of Information Technology for their esteemed guidance, moral support and invaluable advice provided by them for the success of the Mini Project.

Sincerely,

AKHIL THAKUR 1602-19-737-064

B.NIKHITA 1602-19-737-084

M. TANMAYEE 1602-19-737-119

TABLE OF CONTENTS

1. INTRODUCTION

1.1. ABSTRACT

Why Do You Need an Event Management System?

An Event Management System provides campus event planners a flexible, fully integrated solution to simplify the event management process and keep your customers, faculty and students happy, while maintaining important reports and data for making real estate and future planning decisions.

An event management system allows you to:

- Minimize administration efforts
- Eliminate missed communications
- Digitize how your events are run
- Comply with COVID-19 safety guidelines
- Save time planning future events
- Access detailed reports & analytics

1.2. FEATURES

* Log-in the system- Admin should login to the website to access the web app

* Category of events- Admin can categorize the events and display the list of events that he can provide to the customers.

* Event List- Adding new event to the list, selecting venue and the guest list.

* Updating and deleting the events from list

* Preparing Guest list- Adding, updating and removing a guest from the list.

This website is created in view of Admin, he can access the website to make and create a list as to what events are coming in near future to organize.

TECHNOLOGY

All computer software needs certain hardware components or other software resources to be present, in order for computers to be used efficiently. These prerequisites are known as System Requirements. Within this, we have two types – Software Requirements and Hardware Requirements.

2.1. SOFTWARE REQUIREMENTS

Software Requirements deal with defining the software resource requirements and prerequisites that need to be installed on a computer to provide optimal functioning of an application. These preconditions are generally not included in the software installation package and need to be installed separately.

Python:

Python benefitted from both new functionality and optimizations. Python is the language used to build the Django framework. It is a dynamic scripting language similar to Perl and Ruby. The principal author of Python is Guido van Rossum. Python supports dynamic typing and has a garbage collector for automatic memory management. Another important feature of Python is dynamic name solution which binds the names of functions and variables during execution.

• INTERPRETER:

Visual Studio Code- It features a lightning-fast source code editor, perfect for day-to-day use. With support for hundreds of languages, VS Code helps you be instantly productive with syntax highlighting, bracket-matching, autoindentation, box-selection, snippets, and more.

• PYTHON-DJANGO:

Django Framework: Django is a free and open-source web framework, written in Python, which follows the model-view-template architectural pattern. It is maintained by the Django Software Foundation, an independent organization established as a 501 5non-profit. The primary goal of Django is to make the development of complex, databased websites easier. Thus, Django emphasizes the reusability and pluggability of components to ensure rapid developments. Django consists of three major parts: model, view and template[4].

View:

A view function is a Python function that takes a Web request and returns a Web response. This response can be the HTML contents of a Web page, or a redirect, or a 404 error, or an XML document, or an image, anything that a web browser can display. Template: Being a web framework, Django needs a convenient way to generate HTML dynamically. The most common approach relies on templates. A template contains the static parts of the desired HTML output as well as some special syntax describing how dynamic content will be inserted.

• **BOOTSTRAP- WEB DESIGNING:** Bootstrap is a free and open-source front-end framework for designing websites and web applications. It contains HTML- and CSS-based design templates for typography, forms, buttons, navigation and other interface components, as well as optional JavaScript extensions.

2.2. HARDWARE REQUIREMENTS

Hardware requirements refer to the common set requirements defined by any operating system or software application and are usually the physical computer resources. In this, we look into the architecture, processing power, memory, secondary memory, display adapter and peripherals.

- **Processor:** Intel Core i5 and above
- **Memory:** 8 GB RAM

PROPOSED WORK

3.1. ABOUT THE PROJECT

We all celebrate the precious moments of our life such as birthdays, holidays, parties, weddings, etc. To organize the perfect party as per the occasion, we used to contact the event planner. They plan and organize the parties for us, they ensure their concepts are unique and match the scale and the nature of the party.The event planners are connected with the decorations and other sources needed for organizing the event. The event planning web page is Admin oriented, this gives a platform for the admin to maintain his records about the events that he is organizing.

3.2. RELATED WORK

The present projects already developed provide most of the basic functionality required for an event. It allows the user to select from a list of event types, etc. Our project is Admin oriented which is online, where he can use it like a diary to make a list of events.

3.3. USER CASE

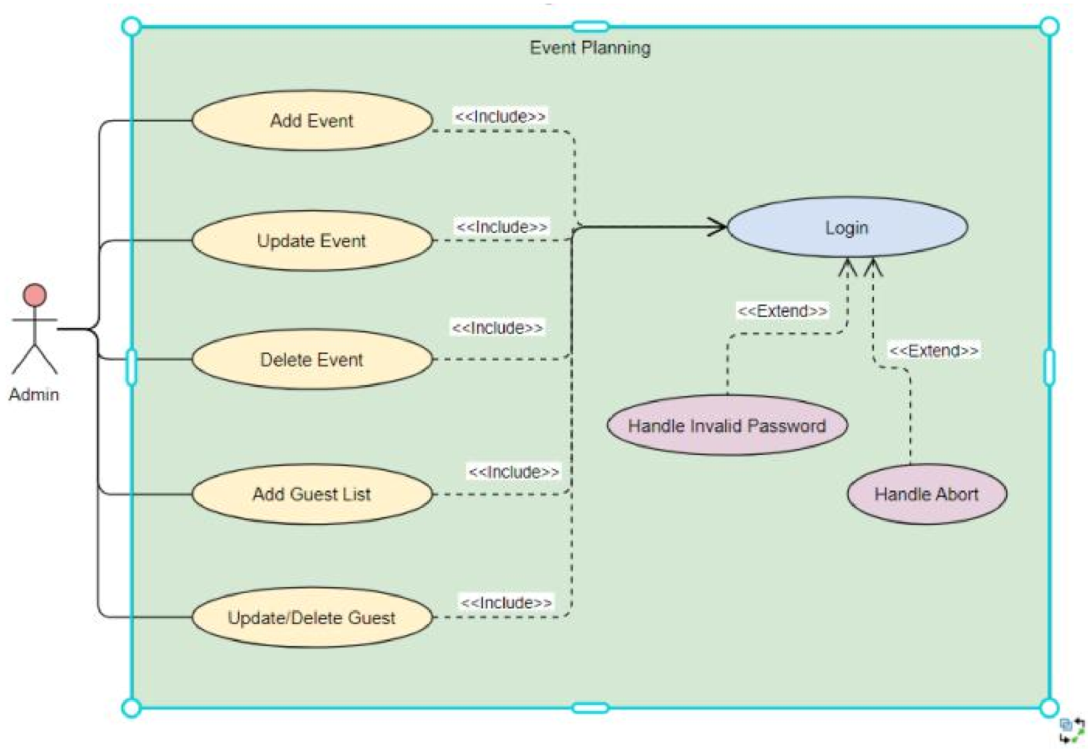
Event Category:

Here the admin provides a list of events that he wants to provide service for.

Event List:

Here the admin has a list of booked events that he is providing or has to provide the service in the future.

3.4. UI PROTOTYPE OR SCREENSHOTS



3.5. ARCHITECTURE AND TECHNOLOGY

Front end:

HTML

CSS

Java script

Python

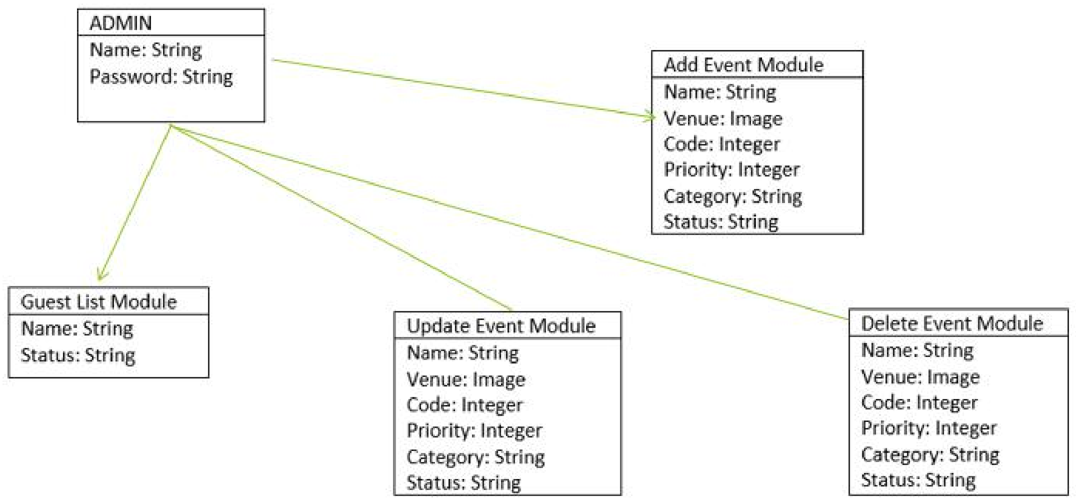
Back end:

Django

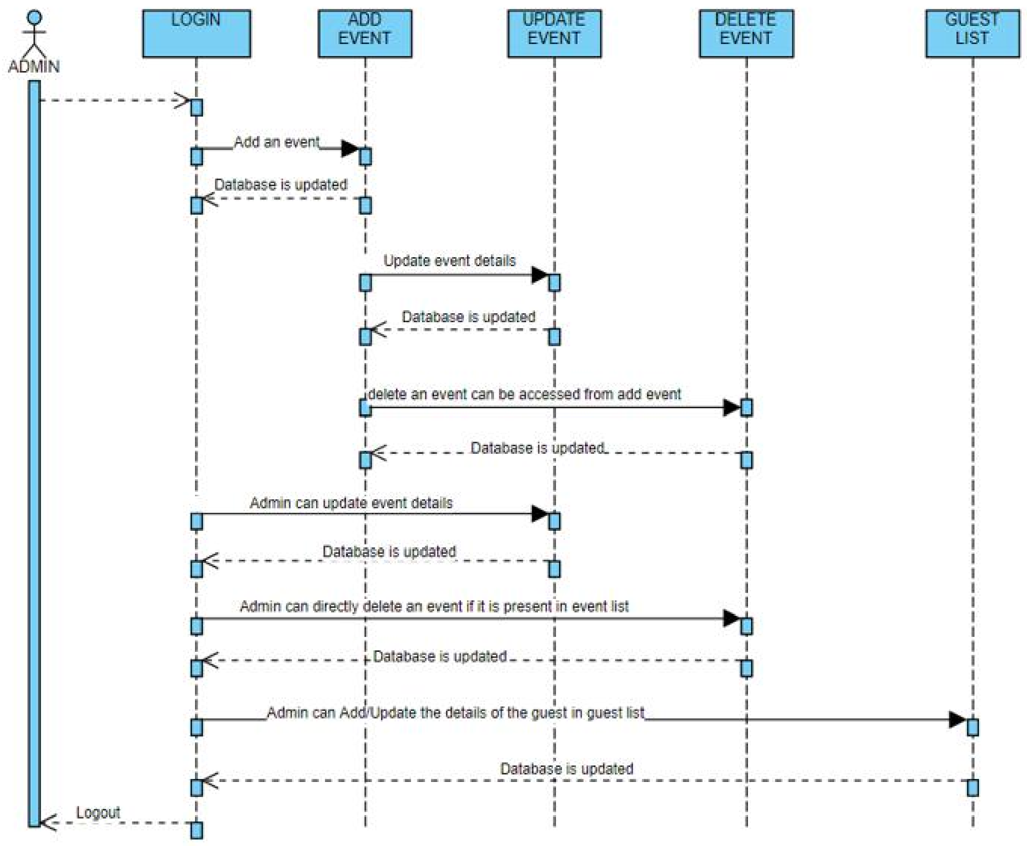
Sqlite

3.6. DESIGN

1.UML Static diagram - Class Diagram



2. UML Dynamic diagram - Sequence Diagram



3.7. IMPLEMENTATION

CODE for views.py : This module includes the basic functionalities of the whole project.

```
from django.views.generic import (

    ListView,

    CreateView,

    UpdateView,

    DetailView,

    DeleteView,

    View,

)

from django.urls import reverse_lazy

from django.shortcuts import render, redirect

from django.contrib.auth.decorators import login_required

from django.contrib.auth.mixins import LoginRequiredMixin

from functools import reduce

from .models import (

    EventCategory,

    Event,

    JobCategory,

    EventJobCategoryLinking,

    EventMember,

    EventUserWishList,

    UserCoin,

    EventImage,

    EventAgenda,
)
from .forms import EventForm, EventImageForm, EventAgendaForm,
EventCreateMultiForm



# Event category list view
class EventCategoryListView(LoginRequiredMixin, ListView):
    login_url = 'login'
    model = EventCategory
    template_name = 'events/event_category.html'
    context_object_name = 'event_category'



class EventCategoryCreateView(LoginRequiredMixin, CreateView):
    login_url = 'login'
    model = EventCategory
    fields = ['name', 'code', 'image', 'priority', 'status']
    template_name = 'events/create_event_category.html'

    def form_valid(self, form):
        form.instance.created_user = self.request.user
        form.instance.updated_user = self.request.user
        return super().form_valid(form)



class EventCategoryUpdateView(LoginRequiredMixin, UpdateView):
    login_url = 'login'
    model = EventCategory
    fields = ['name', 'code', 'image', 'priority', 'status']
    template_name = 'events/edit_event_category.html'



class EventCategoryDeleteView(LoginRequiredMixin, DeleteView):
    login_url = 'login'
    model =  EventCategory
    template_name = 'events/event_category_delete.html'
    success_url = reverse_lazy('event-category-list')


@login_required(login_url='login')
def create_event(request):
    event_form = EventForm()
    event_image_form = EventImageForm()
    event_agenda_form = EventAgendaForm()
    catg = EventCategory.objects.all()
    if request.method == 'POST':
        event_form = EventForm(request.POST)
        event_image_form = EventImageForm(request.POST, request.FILES)
        event_agenda_form = EventAgendaForm(request.POST)
        if event_form.is_valid() and event_image_form.is_valid() and
event_agenda_form.is_valid():
            ef = event_form.save()
            created_updated(Event, request)
            event_image_form.save(commit=False)
            event_image_form.event_form = ef
            event_image_form.save()

            event_agenda_form.save(commit=False)
            event_agenda_form.event_form = ef
            event_agenda_form.save()
            return redirect('event-list')
    context = {
        'form': event_form,
        'form_1': event_image_form,
        'form_2': event_agenda_form,
        'ctg': catg
    }
    return render(request, 'events/create.html', context)


class EventCreateView(LoginRequiredMixin, CreateView):
    login_url = 'login'
    model = Event
    fields = ['category', 'name', 'description', 'scheduled_status', 'venue', 'location',
'status']
    template_name = 'events/create_event.html'

    def form_valid(self, form):
        form.instance.created_user = self.request.user
        form.instance.updated_user = self.request.user
        return super().form_valid(form)
    """login_url = 'login'
```

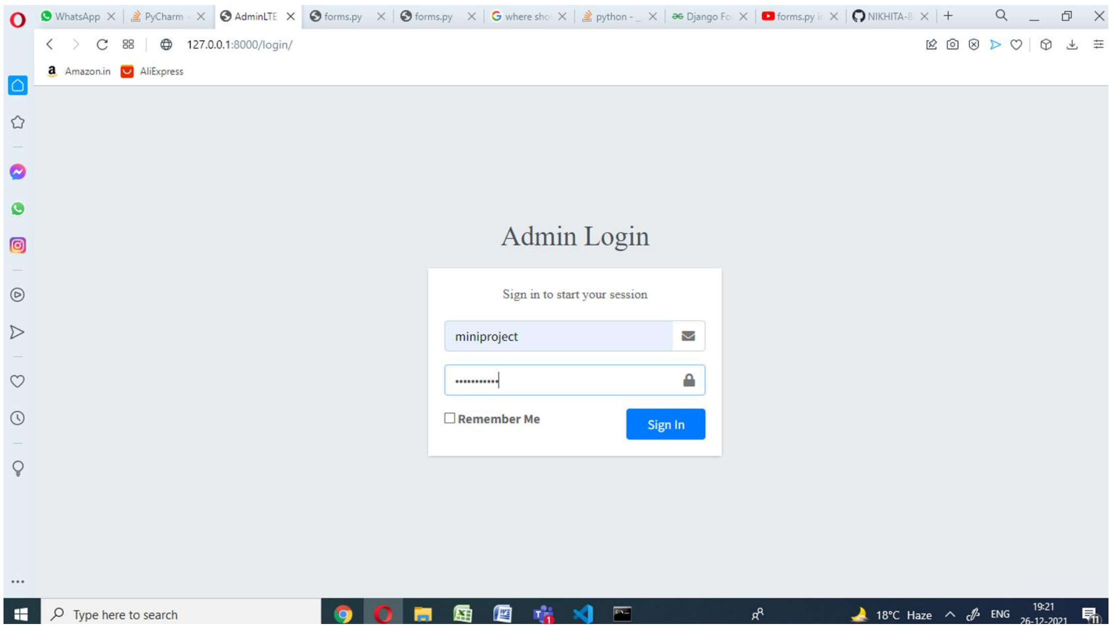
```
model = Event
fields = ['name', 'category' , 'status']
template_name = 'events/create_event.html'
success_url = reverse_lazy('event-list')
def form_valid(self, form):
    form.instance.created_user = self.request.user
    form.instance.updated_user = self.request.user
    return super().form_valid(form)
def form_valid(self, form):
    evt = form['event'].save()
    event_image = form['event_image'].save(commit=False)
    event_image.event = evt
    event_image.save()

    event_agenda = form['event_agenda'].save(commit=False)
    event_agenda.event = evt
    event_agenda.save()

    return super().form_valid(form)"""

def get_context_data(self, **kwargs):
    context = super().get_context_data(**kwargs)

    context['ctg'] = EventCategory.objects.all()
    return context
```

```
class EventListView(LoginRequiredMixin, ListView):
    login_url = 'login'
    model = Event
    template_name = 'events/event_list.html'
    context_object_name = 'events'


class EventUpdateView(LoginRequiredMixin, UpdateView):
    login_url = 'login'
    model = Event
    fields = ['category', 'name', 'uid', 'description', 'scheduled_status', 'venue',
'start_date', 'end_date', 'location', 'points', 'maximum_attende', 'status']
    template_name = 'events/edit_event.html'


class EventDetailView(LoginRequiredMixin, DetailView):
    login_url = 'login'
    model = Event
    template_name = 'events/event_detail.html'
    context_object_name = 'event'


class EventDeleteView(LoginRequiredMixin, DeleteView):
    login_url = 'login'
    model = Event
    template_name = 'events/delete_event.html'
    success_url = reverse_lazy('event-list')


class AddEventMemberCreateView(LoginRequiredMixin, CreateView):
    login_url = 'login'
    model = EventMember
    fields = ['event', 'user', 'attend_status', 'status']
    template_name = 'events/add_event_member.html'

    def form_valid(self, form):
        form.instance.created_user = self.request.user
        form.instance.updated_user = self.request.user
        return super().form_valid(form)


class JoinEventListView(LoginRequiredMixin, ListView):
    login_url = 'login'
    model = EventMember
    template_name = 'events/joinevent_list.html'
    context_object_name = 'eventmember'


class RemoveEventMemberDeleteView(LoginRequiredMixin, DeleteView):
    login_url = 'login'
    model = EventMember
    template_name = 'events/remove_event_member.html'
    success_url = reverse_lazy('join-event-list')


class UpdateEventStatusView(LoginRequiredMixin, UpdateView):
    login_url = 'login'
    model = Event
    fields = ['status']
    template_name = 'events/update_event_status.html'



class CompleteEventList(LoginRequiredMixin, ListView):
    login_url = 'login'
    model = Event
    template_name = 'events/complete_event_list.html'
    context_object_name = 'events'

    def get_queryset(self):
        return Event.objects.filter(status='completed')



class AbsenseUserList(LoginRequiredMixin, ListView):
    login_url = 'login'
    model = EventMember
    template_name = 'events/absense_user_list.html'
    context_object_name = 'absenseuser'

    def get_queryset(self):
        return EventMember.objects.filter(attend_status='absent')


class CompleteEventUserList(LoginRequiredMixin, ListView):
    login_url = 'login'
    model = EventMember
    template_name = 'events/complete_event_user_list.html'
    context_object_name = 'completeuser'

    def get_queryset(self):
        return EventMember.objects.filter(attend_status='completed')


@login_required(login_url='login')
def search_event_category(request):
    if request.method == 'POST':
        data = request.POST['search']
        event_category = EventCategory.objects.filter(name__icontains=data)
        context = {
            'event_category': event_category
        }
        return render(request, 'events/event_category.html', context)
    return render(request, 'events/event_category.html')

@login_required(login_url='login')
def search_event(request):
```

```
if request.method == 'POST':
    data = request.POST['search']
    events = Event.objects.filter(name__icontains=data)
    context = {
        'events': events
    }
    return render(request, 'events/event_list.html', context)
return render(request, 'events/event_list.html')
```

3.8. TESTING

Login page:



Admin Login

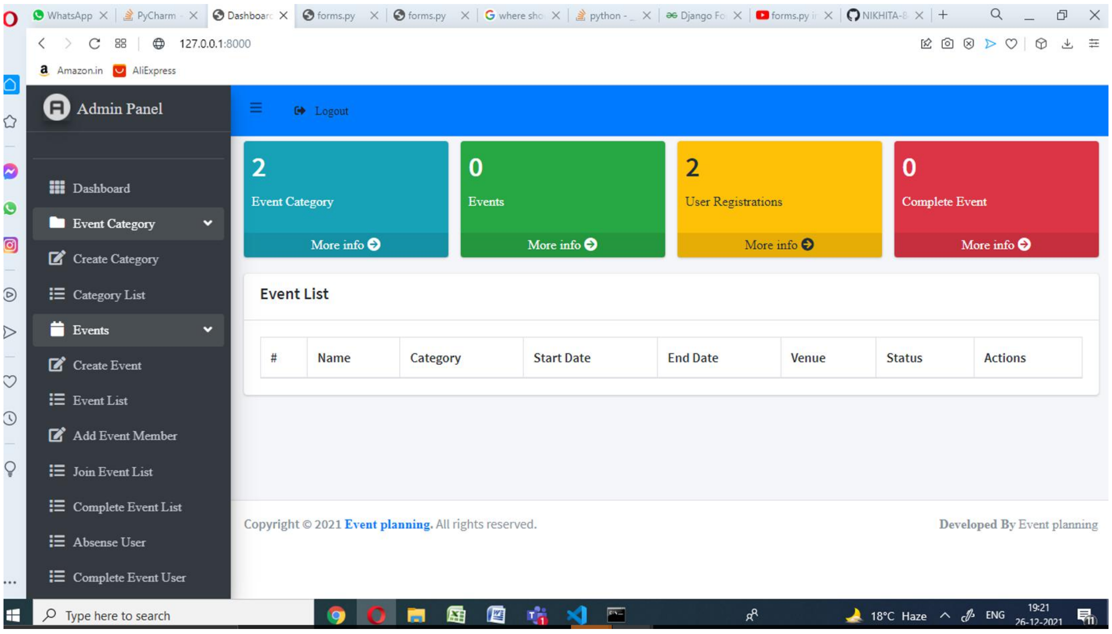
Sign in to start your session

miniproject

Remember Me

Sign In

Dashboard:



Admin Panel

Logout

- Dashboard
- Event Category
- Create Category
- Category List
- Events
- Create Event
- Event List
- Add Event Member
- Join Event List
- Complete Event List
- Absense User
- Complete Event User

2 Event Category More info

0 Events More info

2 User Registrations More info

0 Complete Event More info

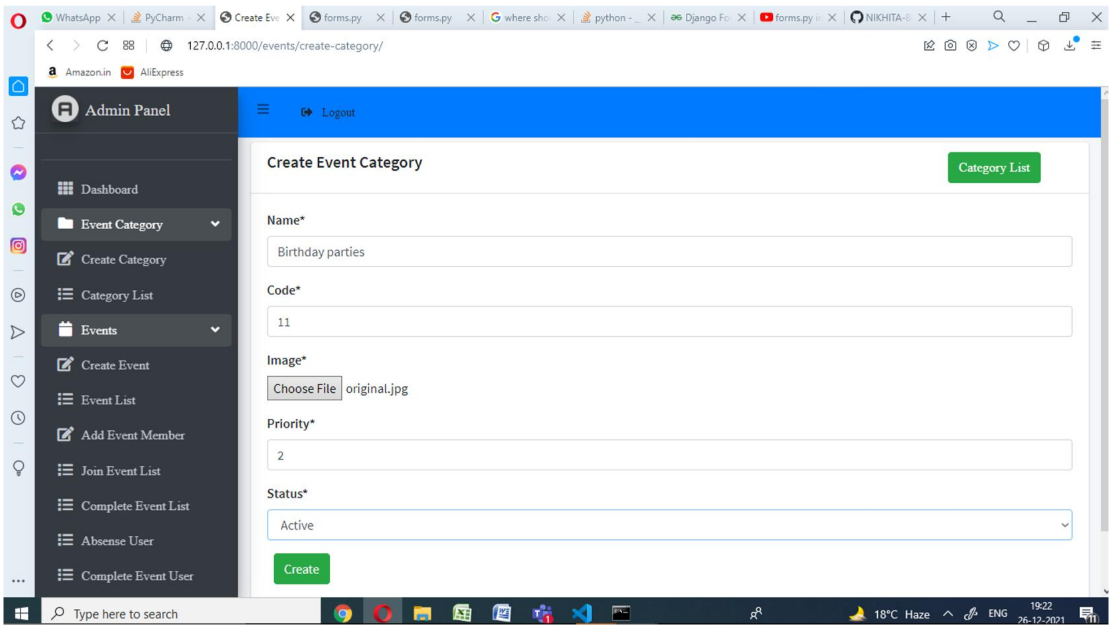
Event List

#	Name	Category	Start Date	End Date	Venue	Status	Actions

Copyright © 2021 Event planning. All rights reserved.

Developed By Event planning

Event Category:



Admin Panel

Logout

Dashboard

Event Category

Create Category

Category List

Events

Create Event

Event List

Add Event Member

Join Event List

Complete Event List

Absense User

Complete Event User

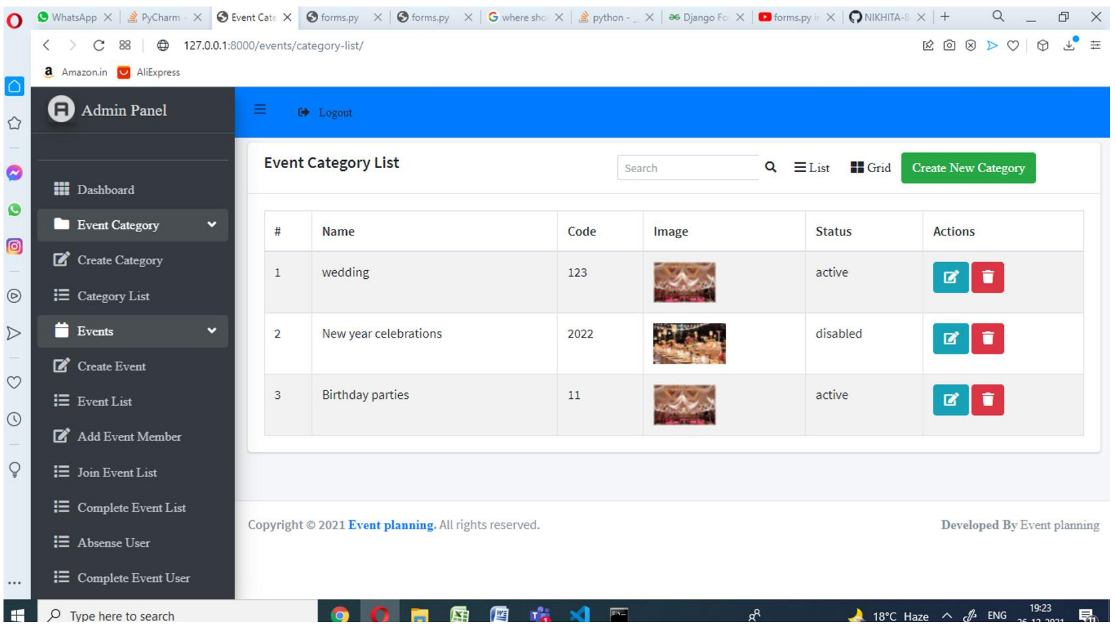
Create Event Category

Category List

Name*	Birthday parties
Code*	11
Image*	Choose File original.jpg
Priority*	2
Status*	Active

Create

Event Category List: List of services provided



Admin Panel

Logout

Dashboard

Event Category

Create Category

Category List

Events

Create Event

Event List

Add Event Member

Join Event List

Complete Event List

Absense User

Complete Event User

Event Category List

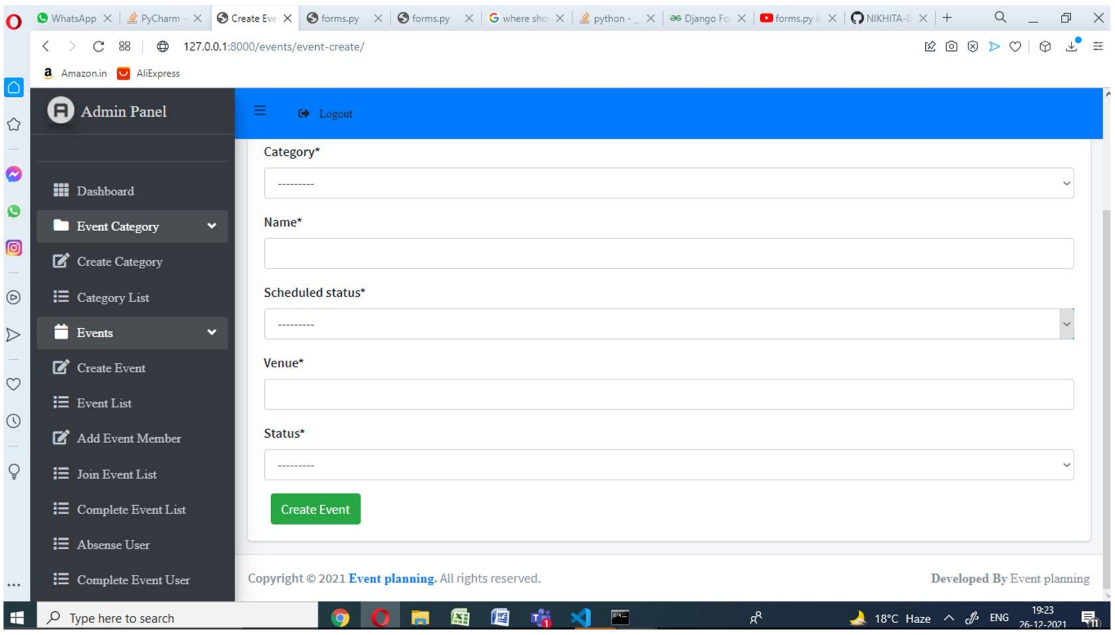
Search | List | Grid | Create New Category

#	Name	Code	Image	Status	Actions
1	wedding	123		active	
2	New year celebrations	2022		disabled	
3	Birthday parties	11		active	

Copyright © 2021 **Event planning.** All rights reserved.

Developed By Event planning

Create Event:



Category*

Name*

Scheduled status*

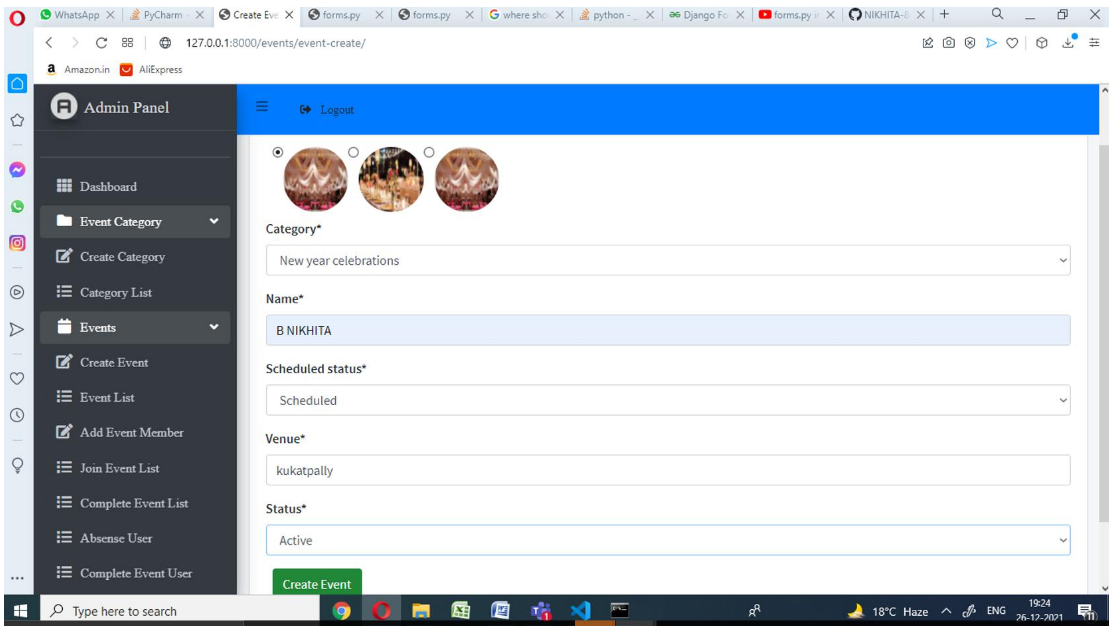
Venue*

Status*

Copyright © 2021 Event planning. All rights reserved.

Developed By Event planning

Event List: List of events that have to be organized in the future



Category*

New year celebrations

Name*

B NIKHITA

Scheduled status*

Scheduled

Venue*

kukatpally

Status*

Active

GIT-HUB Links

https://github.com/NIKHITA-84

https://github.com/tanmayee043/

RESULTS

We have successfully completed creating a website (Event Planning) using HTML, CSS, JavaScript with the help of Django wherein Admin can make a list of events that have to be implemented in the future and the events that are already in process or completed.

DISSCUSSION & FUTURE WORK

Present website is developed in view of Admin, this website can further be developed for the customers to access and select the event planner they want. Event planners can set their profile onto the website so that the customers can select planners as per their requirement. Online interactions can be held for the customers and event planners to interact.

REFERENCES

HTML Study material- https://www.w3schools.com/html/

CSS Study material- https://www.w3schools.com/css

https://www.tutorialspoint.com/css

Understanding DJango Framework- https://www.djangoproject.com/