TECHNICAL DOCUMENTATION

IITG CSE DEPARTMENT EVENT MANAGEMENT

WEB PART…….

Submitted for Lab-4 of CS243 Jan-May ‘19

Group Number:4

Group Member:

1. ROHAN NIGAM
2. LUCKY
3. GEDDAM IKYA VENUS
4. PARTHA PRATIM MALAKAR
5. ROUNAK PARIHAR
6. RUTVIK GHUGHAL
7. AVNEET SINGH CHANNA
8. MAYANK CHANDRA
9. PRIYANSHU SINGH
10. VEMURI SAHITHYA
11. SAYALKUMAR SUBHASH HAJARE
12. SIDDHARTH AGARWAL
13. SEELAM PRADEEPA
14. THAHIR MAHMOOD POOVADA
15. RAJANALA HARSHAVARDHAN REDDY

WEB PART

1. MAYANK CHANDRA
2. LUCKY
3. RAJANALA HARSHAVARDHAN REDDY
4. PARTHA PRATIM MALAKAR
5. VEMURI SAHITHYA
6. SAYALKUMAR SUBHASH HAJARE
7. THAHIR MAHMOOD POOVADA
8.GEDDAM IKYA VENUS

INTRODUCTION

Purpose

This the technical documentation of the IITG CSE department event management (web part) as a part of the CS243 lab to demonstrate a working model of a Event management app.

Intended Users

Anyone intending to improve, evaluate or use the application or a part of it can use this document.

REQUIRMENTS

A) For using purpose only, the web application will run in any web browser.

B) For other purposes including development, evaluation or other uses,

PC should have python3.6 , virtual environment(pipenv) and Django,Rest Framework,markdown framework installed.

Internet connectivity is needed to access bootstrap and to make the app fully functional.

WEBAPP DESIGN

Aim

The aim of this application is to provide a simple but efficient Event management application to create and manage events.

The application uses Django as web development platform. The application uses sqlite databases which comes with django and the webpages are made with html, css and bootstrap .

TEST USERs

1. **Username: test**

   Password : test

2. **Username: admin**

   Password: admin

Steps for installations:-

1.Virtual Environment

Commands for installing pipenv

- pip3 install pipenv(requires python3)
- pipenv update
- pipenv shell

Commands for installing Django

- pip3 install Django

Steps for running web:-

Navigate to the folder containing the app and type the following commands

- python3 manage.py makemigrations
- python3 manage.py migrate
- python3 manage.py runserver

It will open the web application at localhost (127.0.0.0:8000)

Steps for creating a Super User:-

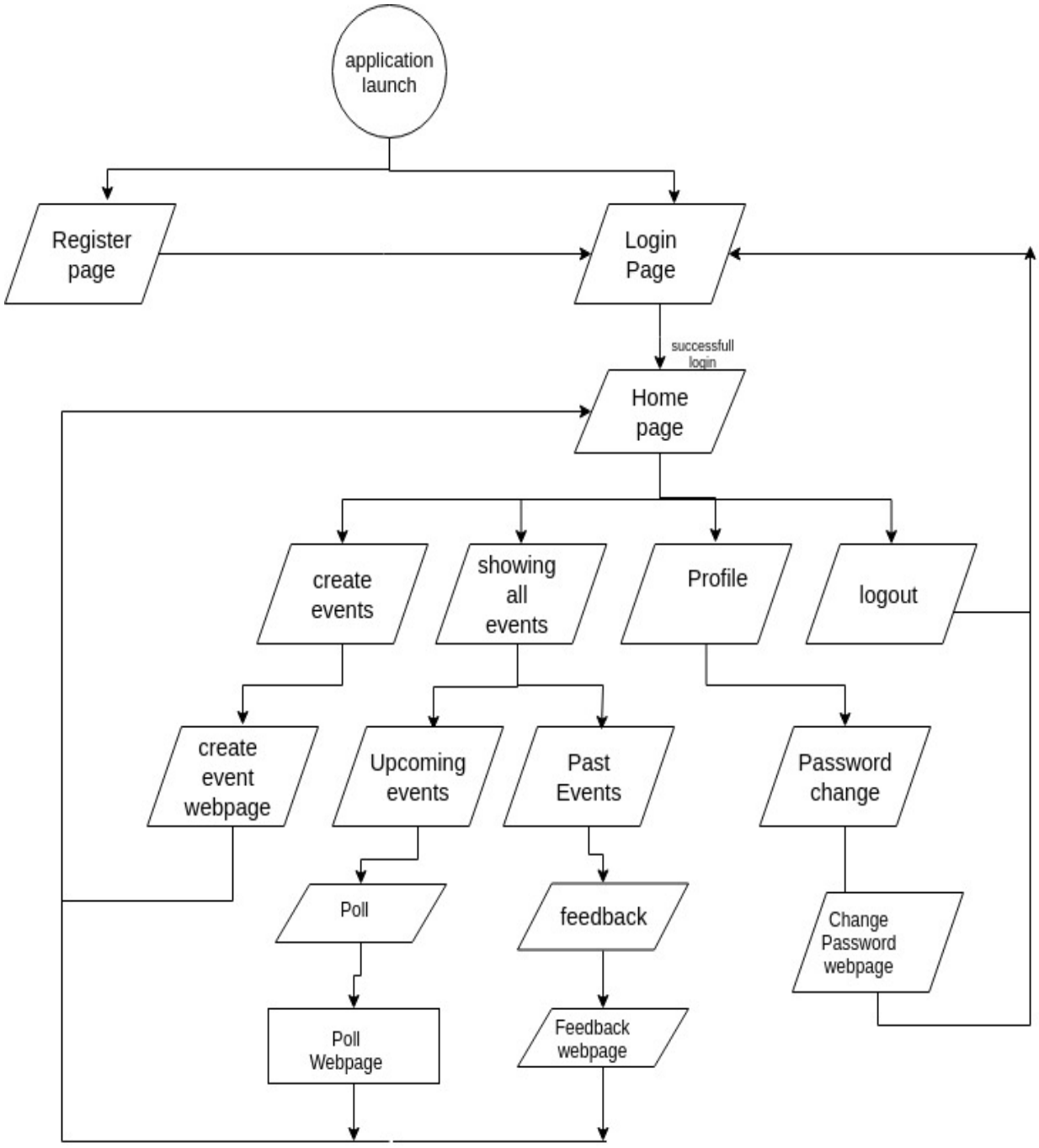
Navigate to the folder containing the app and type the following commands

- python3 manage.py createsuperuser
    - The following fields will open asking for username,email, password (and confirmation of password).

Steps for installing frameworks:-

- pip3 install djangorestframework
- pip3 install markdown

FLOW CHART DIAGRAM:



Component Description

Django is used as web development platform for this application. We named the app csea-events which have the following contents ————

csea-events:

- **csea-events:**
  - **__init.py__**
  - **Settings.py**
  - **urls.py**
  - **wsgi.py**
- **events:**
  - **__init.py__**
  - **admin.py**
  - **app.py**
  - **forms.py**
  - **models.py**
  - **tests.py**
  - **views.py**
- **templates:**
  - **base.html**
  - **change_password.html**
  - **create_event.html**
  - **done.html**
  - **error.html**
  - **event_info.html**
  - **home.html**
  - **login.html**
  - **poll_view.html**
  - **profile.html**

register.html

urls.py

```
urlpatterns = [
    path('admin/', admin.site.urls),
    path('', views.loginPage, name='loginPage'),
    path('home/',views.home_page, name='home_page'),
    path('register/', views.registerPage, name='register'),
    path('api/', include('events.urls')),  #forwards any request with api/ to the api urlpatterns
    path('create/', views.create_event, name='create_event'),
    path('logout/', views.logout_user, name='logout'),
    path('event/<uuid:event_id>/',views.poll_view,name='polling'),
    path('event/', views.poll_view, name='poll'),
    path('change_passswd/',views.change_password,name='change_passwd'),
    path('profile/',views.profile_view,name='profile'),
    path('app-login/', views.api_resp, name='app_login'),
    path('past-events/',views.past,name='past'),
    path('my-events/',views.my_events,name='my_events'),
    path('event/edit/<uuid:id>',views.event_edit,name='event_edit'),
    path('feedback/<uuid:id>',views.feedback_view,name='feedback'),
    path('acceptor/',views.api_reg),
    path('poll/<uuid:event_id>/',views.poll_count_view,name='poll_count'),
    path('poll/<uuid:event_id>/vote',views.poll_vote,name='poll_count_vote'),
    path('poll/<uuid:event_id>/modify',views.poll_modify,name='poll_count_modify'),
    path('api-change-pw/',views.api_change_pw),


]
```

- **path=' '**
  - **function called = views.login_page**
  - **work : The login page is opened**
  - **alias = loginPage**
- **path='home/'**
  - **function called = views.home_page**
  - **work : The homepage is opened**
  - **alias = home_page**
- **path='register/'**
  - **function called=views.registerPage**
  - **work : Opens the registration page**
  - **alias = register**
- **path='create/'**
  - **function called=views.create_event**

- work : Open the form for creating
- alias='create_event'

- path='logout/'
  - function called=views.logout_user
  - work : It opens the Logout page
  - alias='logout'
- path='event/<uuid:event_id>/'
  - function called=views.poll_view
    - parameters passed : event_id
  - work : It opens the form showing details of the event of the particular event_id
  - alias='polling'
- path='change_passswd/'
  - function called=views.change_password
  - work : It open the interface for changing the password
  - alias='change_passwd'
- path='profile/'
  - function called=views.profile_view
  - work : It displays the profile of the user
  - alias='profile'
- path='app-login/'
  - function called=views.api_resp
  - work : It is used for logging in to the app.
  - alias='app_login'

- **path=’poll/<uuid:event_id>/’**
  - **function called=views.poll_count_view**
    - **parameters passed : event_id**
  - **work : A form opens with the current result of poll.**
  - **alias=‘poll_count’**

- **path=’poll/<uuid:event_id>/vote’**
  - **function called=views.poll_vote**
    - **parameters passed: event_id**
  - **work : It is the form used for taking input of the choice of the user.**
  - **alias=’poll_count_vote’**

- **path=’poll/<uuid:event_id>/modify’**
  - **function called=views.poll_modify**
    - **parameters passed: event_id**
  - **work : It is the form used for modifying vote of the user.**
  - **alias=’poll_count_modify’**

- **path='past-events/'**
  - **function called=views.past**
  - **work : It opens a form showing the past events.**
  - **alias='past'**

- **path='my-events/'**

- function called=views.my_events
- work : It opens a form showing the events of user.
- alias='my_events'

- path='event/edit/<uuid:id>'
  - function called = views.event_edit
    - parameters passed : event_id
  - work : It opens form for editing the event info.
  - alias='event_edit'

- path='feedback/<uuid:id>'
  - function called=views.feedback_view
    - parameters passed = id
  - work: It opens form for submitting the feedback
  - alias='feedback'

- path='acceptor/'
  - function called=views.api_reg
  - work : It allows the user to register through app.
- path='api/'
  - function called=events.urls
  - work : Access point for rest_api frameworks.

Events:

It is a Django app created inside csea-events app.

admins.py

This file is used to register or save user data imported as models class from models.py in Django database.

'EventFeedback', 'Event', 'Btech', 'Mtech' , 'phd', 'AppFeedback' , 'profile' registered from here.

models.py

This file contains some classes with each having some fields. Each class used as a database .The fields of each class acts as database field and contains user information according to its name or properties .

forms.py

This file contains all types of forms used in our app. Forms are made as class containing the form fields ,their properties and others necessary functions.

views.py

This files contains the functions that directly deals with the webpages or the forms used by webpages. Linking webpages ,saving user data to database are done in the functions of views.py .URLs from urls.py links webpages using functions from views.py .

Templates and .html files:

Templates is a directory created inside csea-events app.The html files inside templates are used to create the webpages.

base.html:

This html file is not used for creating any webpage directly. It is mainly for designing purpose. It also contains css and bootstrap. This filed is included in all other .html files as `{%extends 'base.html'%}`

Description of designer components

1.Login

Functions Used:-

Views.loginPage(request):

This function is called in the first or main webpage of our app. This function use LoginForm defined in form.py and shows a login form. After pressing submit button in login form ,if form 'is_valid' and 'if user is not None' means user exists in database, this function login the user by 'login(request, user)' and redirected to homepage. If form is not valid or if user don't exists in database the submit button redirected to login page again.

Django form used:

LoginForm(forms.form):

This is Django form used for taking input from the user for login purpose. It has two fields 'email' and 'password'.

3.User HomePage

Functions Used:-

home_page(request):

This function is called after a successful user login. It shows all the events in the database, allows the user to **'change password'**, allows the user to **'create event'**. This also has a logout button which logout the user and redirect to the login page.

Each events in home page has **'read more'**, **'polling'** and **'feedback'**

Change password:

This links the home page to another page 'change_password.html'

Function Used:

Views.change_password(request):

This function is called after clicking change_password . **If** ‘request.method=='POST'’ and then if the form ‘is_valid’ means if the old password match the original password and new password and confirm new password matches, the user is logged out and redirected to login page.

Django form used:

It uses the PasswordChangeForm from django.contrib.auth.forms.

Create Event:

This links the home page to another page ‘create_event.html’.

Function used:

Views.create_event:

This function is called after clicking create event in user home page. It links to ‘create_event.html’. After filling up all required fields and pressing submit button,this function save the fields in database. This function also save the user name in the field ‘**requestor**’.

Django model used:

It contains the information that an event have in its different fields in database---

Name=This field stores Event name.(filled by user or event creator)

event_id=This field stores event id(auto generated)

Fee=This field stores Event fees.(filled by user or event creator)

Capcity=This field stores event capacity.(filled by user or event creator)

Target audience=This field stores number of target audience.(filled by user or event creator)

Date= This field stores the date of the event.(filled by user or event creator)

Time= This field stores the time of the event.(filled by user or event creator)

summary= This field stores the summery of the event.(filled by user or event creator)

organisors= This field stores the organisors name of the event.(filled by user or event creator)

tags= This field stores the tag of the event.(filled by user or event creator)

contat_info= This field stores the contact information of the organisors.(filled by user or event creator)

venue= This field stores the date of the event.(filled by user or event creator)

requestor= This field stores the user of the event.(filled from views.create_event function using ‘request.user’)

invitees_btech= This field stores the btech invitees of IITG to the event.(filled by user or event creator)

invitees_mtech= This field stores the Mtech invitees of IITG to the event.(filled by user or event creator)

invitees_phd= This field stores the phd invitees of IITG to the event.(filled by user or event creator)

comment_for_admin= This field stores the comment to admin by requestor of the event.(filled by user or event creator)

approval=This filled shows the event is approved or not.(filled by admin)

faq_question1=This field stores FAQ question1.(filled by user or event creator)

faq_answer1=This field contain answer of faq question1.(filled by user or event creator)

faq_question2= This field stores FAQ question2.(filled by user or event creator)

faq_answer2= This field contain answer of faq question2.(filled by user or event creator)

faq_question3= This field stores FAQ question3.(filled by user or event creator)

faq_answer3= This field contain answer of faq question3.(filled by user or event creator)

faq_question4= This field stores FAQ question4.(filled by user or event creator)

faq_answer4= This field contain answer of faq question4.(filled by user or event creator)

faq_question5= This field stores FAQ question5.(filled by user or event creator)

faq_answer5= This field contain answer of faq question5.(filled by user or event creator)

Django form used:

EventCreatorForm(ModelForm):

This is Django form used for taking input from the user for event creation purpose.It has fields --- 'name','fee','capacity','target_audience','date','time','venue','tags','invitees_btech','invitees_mtech','invitees_phd','organisors','contact_info','summary','comment_for_admin','faq_question_1','faq_answer_1','faq_question_2' and 'faq_answer_2'

The homepage also shows all the events. **Each events ‘Read more’, ‘poll’, and ‘feedback’ option.**

Read more

Function Used:-

Views.poll_view: This function is called after clicking ‘Read more’ in user home page. It redirects to ‘event_info.html’ and show detailed information of the event.This function send ‘event_name’, ‘event_date’,

‘event_time’, ‘event_summary’, ‘contact_info’ , ‘event_fee’and ‘faq’ to the ‘event_info.html’ as context.

Poll

Functions Used:-

1. views.poll_count_view:

This function is called whenever the 'Poll' option is clicked.It checks whether a object of 'Poll' Model is created for the 'Event' for not. If the object is already created then the count of "People coming","People not coming" and "People not sure" are passed to "poll_view.html" template, If not then a new 'Poll' object is created with the same initial values of zero(0).

2. views.poll_vote:

This function is called when the 'Vote Here' option is clicked in "poll_view.html". This functions checks whether the user has already voted for the event or not . If the user has already voted then user is not allowed to vote but he/she can 'Modify' his/her vote.If the user has not voted already then a form opens asking for the choice of whether the user (wants to/does not want to/is not sure) come to the event.

3. views.poll_count_modify:

This function is called whenever 'Modify' option on "poll_view.html" is clicked. This opens the "poll_vote.html" form and asks the choice again from the user. The form redirects back to the "poll_view.html" form. The user can modify his/her choice as many times he/she wants.

Django Forms used:-

forms.PollCreatorForm:

This is django form used for taking input from the user for the choice in the poll.This contains only one field 'f_value' which is a dropdown allowing only a single selection. This also has function .save() which takes care of the counts of the 'Poll' object. It also decrements count when the user is modifying its choice.

Django Models Used:-

1 Poll:

This contains the information about the poll.It has the following fields-

event_id: This is the id of the event for which the poll is created(Primary Key).

response_not_coming: This stores the number of users who wants to come to the event.

response_coming : This stores the number of user who does not want to come to the event.

response_not_sure : This stores the number of user who are not sure about whether they want to come or not to the particular event.

user_id : This stores all the user who have voted in the poll.(List seperated by comma ',')

2 Vote:

This contains the information about the choices of the user related to a particular poll. It has the following fields-

vote_id: This contains the id of the poll for which the users have voted for.

user_id: The username of the user who has voted.

user_vote:

The choice of the user for the particular poll(referenced by vote_id). It contains three types of values.

[1:response_not_coming]

[2:response_coming]

[3:response_not_sure]

feedback

Functions Used:-

views.feedback_view :

This function is called whenever 'Feedback' option is clicked . If the request is of type GET then the form 'feedback.html' diplays the rating ,feedback etc. If the request is of type POST then form 'feedback.html' takes the input of feedback and rating in the approriate field and changes the database accordingly.

Form used:-

Feedback Form:

This Django form is used for taking user feedback and rating for an event. It has field ‘content ’ and ‘rating’.

Event Feedback:

This contains the feedbacks,rating given by the users.It has the following fields-

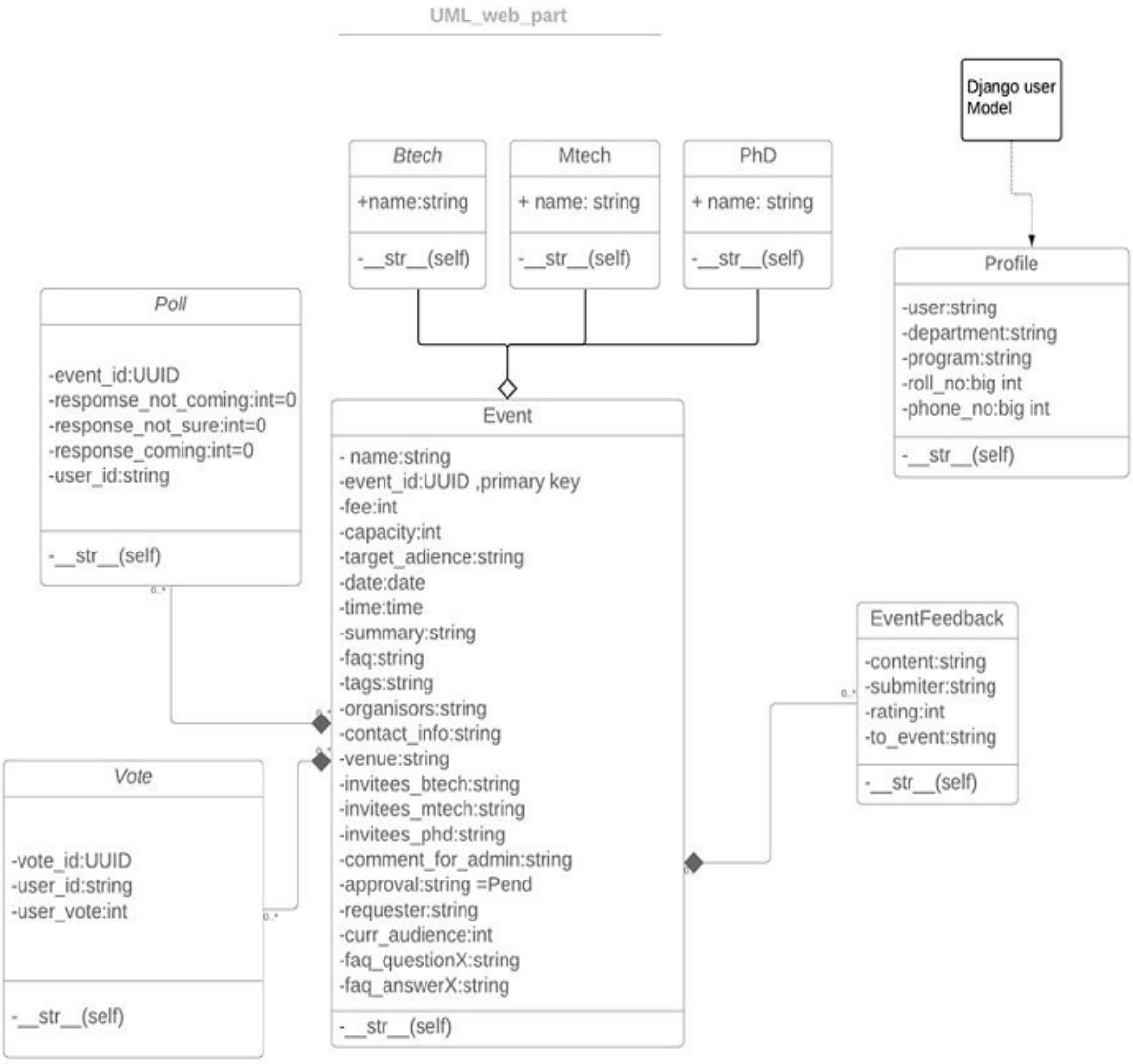
content: This contains the actual feeback given by the users.(filled by user)

submiter: This contains the username of the user who has given the feedback. (filled from views.feedback_view function using ‘request.user’)

rating : The actual rating given by the user.(filled by user)

to_event : The event_id that the feedback corresponds to. (filled from views.feedback_view function)

UML Class Diagram:



UML Use Class Diagram: