## 1. Abstract

Alumni Portal is a web portal where alumni of particular university can register themselves and build a social and professional network with students and facility of their alma-mater.

## 2. Objective

The aim of this project is to build a system that will be able to manage alumni data of the university and provide easy access for the same. Where new students will be initially given a student login ID by that they can access their accounts, alumni can do the same.

The portal will automatically list all college students as alumni on their graduation. Users will be also prompted to update their social network details such as Facebook, LinkedIn and Twitter handles. Users of the portal can also choose to automatically share new updates in work status from their LinkedIn profile, they will also be able to share their Facebook and Twitter updates.

The portal can also track user location as given by the user. Once the system notices that more than 10 alumni are available in the same city it can notify all of them about the possibility of a meet.

Alumni will also be able to provide public posts on the portal about possible job opportunities, new technologies & business possibilities or other university related news. Since it is unlikely that alumni will check the portal frequently so the portal will be able collate all public posts and create a newsletter that can be emailed to all alumni.

The portal will also have some privacy features where users will be able to determine what information they want to share and also to whom they want to share it with. For example : users can choose to share their Facebook profile name and mobile number with alumni who graduated in the same year as them.

They portal will also have a chat feature which will enable alumni to chat without revealing their mobile number or personal e–mail IDs.

## 3. Modules

### 3.1 Admin

The admin will be responsible for creating new login IDs for incoming students, admin will also have to browse the site to ensure no objectionable content is posted. The admin will also be notified about any complaints from users.

### 3.2 Alumni

An alumnus of the college will be able to access other alumni information and also will be able to view all their contact information (unless it is made private). An alumnus can be able to post any information they deem relevant on the site.

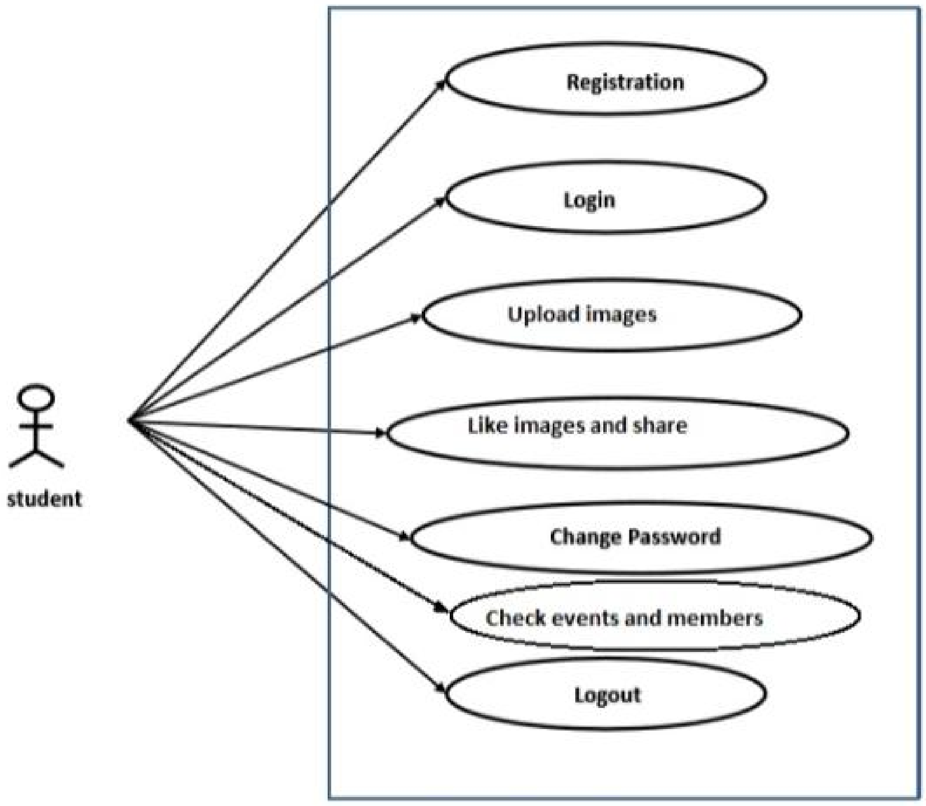


Fig 3.1 Use Case Diagram

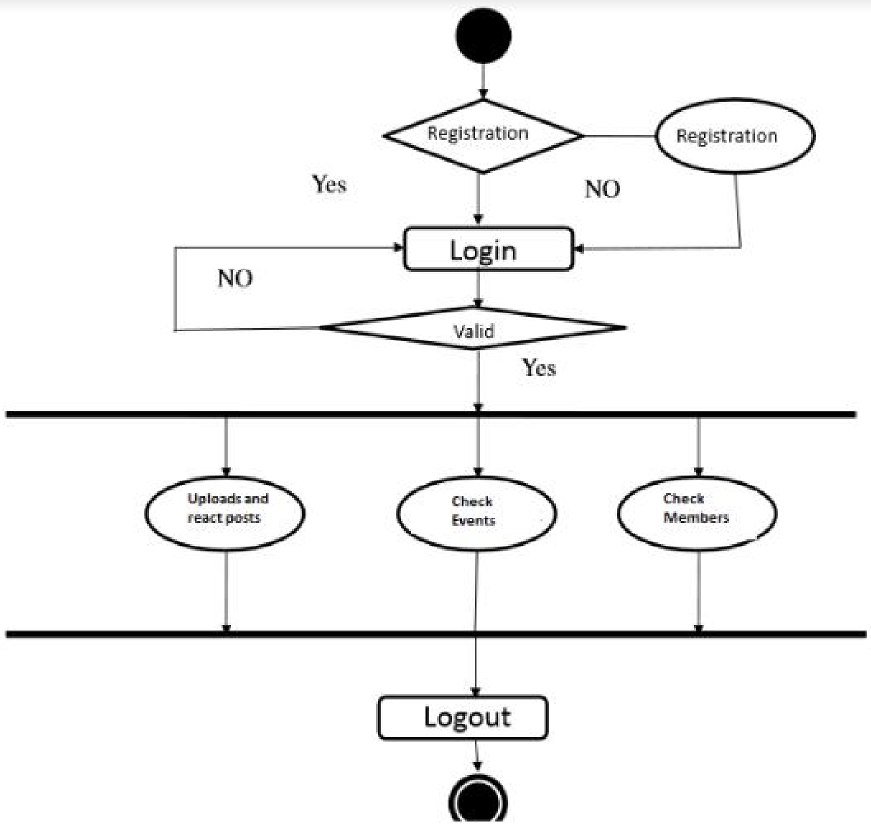


Fig 3.2 Activity Diagram

## 4. Software Requirements

1. Operating system (e.g. Windows, Linux),
2. PyCharm IDE (optional)

## 5. Technology Used

1. Python 2.7.x
2. Django 1.x
3. Other web technologies such as : HTML5, CSS3, Bootstrap, AJAX etc.

## 6. Hardware Requirements

1. Hard Disk – 20 GB
2. RAM – 1 GB
3. Processor – Dual Core or Above

## 7. Work on Project

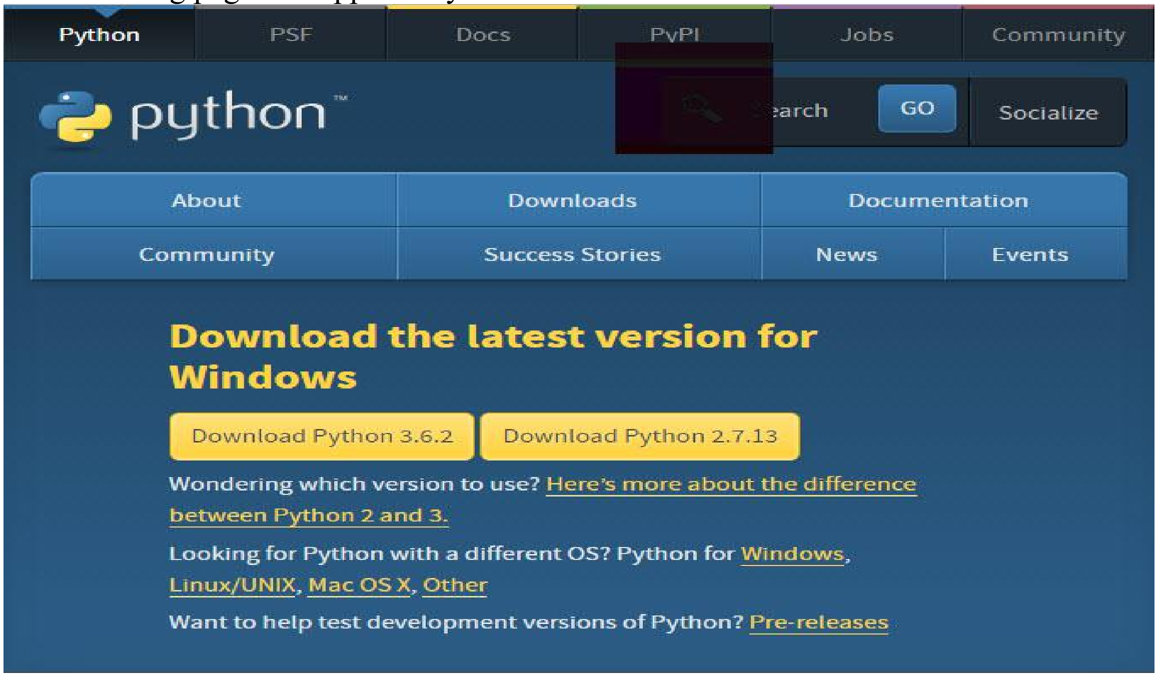
## 7.1 Installing Python 2.7.x

The Python download requires about 30 Mb of disk space; keep it on your machine, in case you need to re-install Python. When installed, Python requires about an additional 90 Mb of disk space.

**Downloading**

1. Open Link : [https://www.python.org/downloads/](https://www.python.org/downloads/)

The following page will appear in your browser.



Download the latest version for Windows

Wondering which version to use? Here's more about the difference between Python 2 and 3.

Looking for Python with a different OS? Python for Windows, Linux/UNIX, Mac OS X, Other

Want to help test development versions of Python? Pre-releases

2. Click the Download Python 2.7.x button.

The file named python-2.7.x.exe should start downloading into your standard download folder. This file is about 30 Mb so it might take a while to download fully if you are on a slow internet connection (it took me about 10 seconds over a cable modem).

The file should appear as

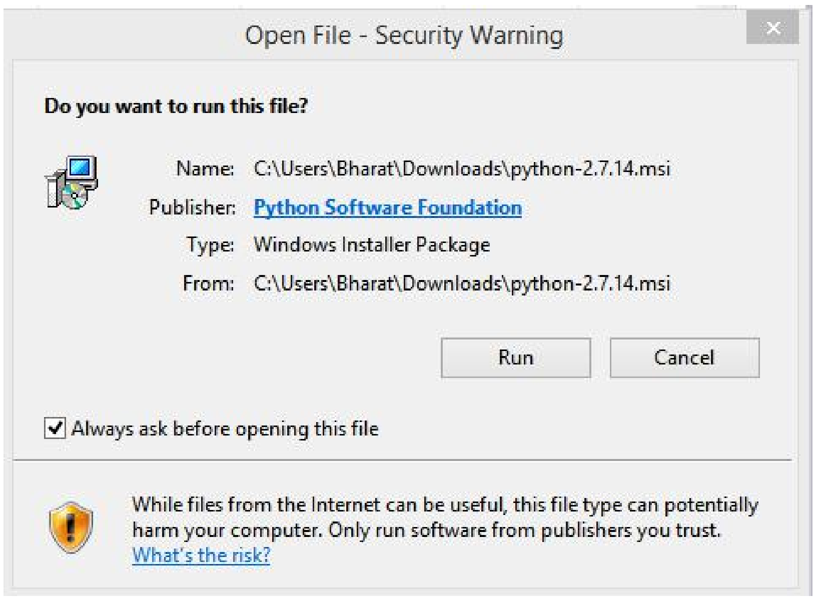
3. Move this file to a more permanent location, so that you can install Python (and reinstall it easily later, if necessary).

4. Feel free to explore this webpage further; if you want to just continue the installation, you can terminate the tab browsing this webpage.

5. Start the Installing instructions directly below.

**Installing**

1. Double-click the icon labeling the file python-2.7.x.exe.
An **Open File - Security Warning** pop-up window will appear.



Open File - Security Warning

**Do you want to run this file?**

Name: C:\Users\Bharat\Downloads\python-2.7.14.msi
Publisher: Python Software Foundation
Type: Windows Installer Package
From: C:\Users\Bharat\Downloads\python-2.7.14.msi

☑ Always ask before opening this file

While files from the Internet can be useful, this file type can potentially harm your computer. Only run software from publishers you trust. What's the risk?

2. Click Run.
A Python 2.7.x (32-bit) Setup pop-up window will appear.

Ensure that the Install launcher for all users (recommended) and the Add Python 2.7 to PATH checkboxes at the bottom are checked.

If the Python Installer finds an earlier version of Python installed on your computer, the Install Now message will instead appear as Upgrade Now (and the checkboxes will not appear).

3. Highlight the Install Now (or Upgrade Now) message, and then click it.
A User Account Conrol pop-up window will appear, posing the question Do you want the allow the following program to make changes to this computer?

4. Click the Yes button.
A new Python 2.7.x (32-bit) Setup pop-up window will appear with a Setup Progress message and a progress bar.

During installation, it will show the various components it is installing and move the progress bar towards completion. Soon, a new Python 2.7.x (32-bit) Setup pop-up window will appear with a Setup was successfully message.

5. Click the Close button.

Python should now be installed.

**Verifying**
1. To try to verify installation, run command in CMD

```
    $ python --version
```

It will show the python with its version.

## 7.2 Creating Virtual Environment

**What is Virtualenv?**

A Virtual Environment, put simply, is an isolated working copy of Python which allows you to work on a specific project without worry of affecting other projects.

It enables multiple side-by-side installations of Python, one for each project.

It doesn't actually install separate copies of Python, but it does provide a clever way to keep different project environments isolated.

**Verify if Virtualenv is installed**

There is a chance that virtualenv is already installed on your system.

Run the following command in your terminal

```
virtualenv --version
```

If you see a version number (in my case 1.6.1), it's already installed.

**Install Virtualenv**

There are a number of ways to install virtualenv on your system.

```
$ sudo pip install virtualenv
```

**Setup and Use Virtualenv**

Once you have virtualenv installed, just fire up a shell and create your own environment.

First create a directory for your new shiny isolated environment

```
mkdir ~/venv
```

To create a folder for your new app that includes a clean copy of Python,

simply run:

```
virtualenv ~/venv
```

(add --no-site-packages if you want to isolate your environment from the main site packages directory)

To begin working with your project, you have to cd into your directory (project)
and activate the virtual environment.

```
cd ~/venv/bin
```

Lastly, activate your environment:

```
source activate
```

Notice how the prompt of your shell changed to show the active environment.

That is how you can see that you're in your new environment.

Any packages you install now using pip or easy_install get installed into venv/lib/python2.7/site-packages.

To exit your virtualenv just type “deactivate”.

**What did Virtualenv do?**

Packages installed here will not affect the global Python installation. Virtualenv does not create every file needed to get a whole new python environment. It uses links to global environment files instead in order to save disk space end
speed up your virtualenv.

Therefore, there must already have an active python environment installed on your system.

## 7.3 Installing Django 1.x

Now that we have Python and are running a virtual environment, installing Django is super easy, just type the command:

```
$ pip install django==1.11.2
```

This will instruct `pip` to install Django into your virtual environment. Your command output should look like this:

```
(venv) C:\Users\Bharat\Documents\Project> pip install django==1.11.2

Collecting django==1.11.2
Using cached Django-1.11.2-py2.py3-none-any.whl
Collecting pytz (from django==1.11)
Using cached pytz-2017.2-py2.py3-none-any.whl
Installing collected packages: pytz, django
Successfully installed django-1.11.2 pytz-2017.2
```

In this case, we are explicitly telling `pip` to install Django 1.11.2, which is the latest version of Django 1.11 LTS at the time of writing. If you are installing Django, it’s good practice to check the Django Project website for the latest version of Django 1.11 LTS.

Also note my computer didn’t need to download anything as I have another virtual environment I installed today and Windows used the cached version.

And finally, note that Django 1.11 requires the Python Timezone package (`pytz`), so `pip` installs that in your virtual environment as well.

For some post-installation positive feedback, take a moment to test whether the installation worked. At your virtual environment command prompt, start the Python interactive interpreter by typing `python` and hitting enter. If the installation was successful, you should be able to import the module `django`:

```
(venv) C:\Users\Bharat\Documents\Project>
```

## 7.4 Starting a Project

Once you’ve installed Python, you can take the first step in developing a Django application by creating a *project*. A project is a collection of settings for an instance of Django. If this is your first time using Django, you’ll have to take care of some initial setup.

Namely, you’ll need to auto-generate some code that establishes a Django project. The auto-generated code contains a collection of settings for an instance of Django, including database configuration, Django-specific options and application-specific settings.

From your virtual environment command line, run the following command:

```
django-admin startproject AlumniPortal
```

This command will automatically create a `AlumniPortal` directory in your project directory as well as all the necessary files for a basic, but fully functioning Django website.

Let's look at what `startproject` created:

```
AlumniPortal/
  manage.py
  AlumniPortal/
    __init__.py
    settings.py
    urls.py
    wsgi.py
```

These files are:

- The outer `AlumniPortal/` root directory. It's just a container for your project. Its name doesn't matter to Django; you can rename it to anything you like.
- `manage.py`. A command-line utility that lets you interact with your Django project in various ways. You can read all the details about `manage.py` on the Django Project website.
- The inner `AlumniPortal/` directory. It's the Python package for your project. It's the name you'll use to import anything inside it (e.g. `AlumniPortal.urls`).
- `AlumniPortal/__init__.py`. An empty file that tells Python that this directory should be considered a Python package. (Read more about packages in the official Python docs if you're a Python beginner.).
- `AlumniPortal/settings.py`. Settings/configuration for this Django project. Appendix D will tell you all about how settings work.
- `AlumniPortal/urls.py`. The URL declarations for this Django project; a "table of contents" of your Django-powered site. You can read more about URLs in Chapters 2 and 7.
- `AlumniPortal/wsgi.py`. An entry-point for WSGI-compatible web servers to serve your project. See Chapter 13 for more details.

## 7.5 Setting Up a Database

Django includes a number of applications by default (e.g. the admin program and user management and authentication). Some of these applications make use of at least one database table, so we need to create tables in a database before we can use them. To do that, change into the `AlumniPortal` folder created in the last step (type `cd AlumniPortal` at the command prompt) and run the following command:

```
python manage.py migrate
```

The `migrate` command creates a new SQLite database and any necessary database tables according to the settings file created by the `startproject` command (more on the settings file later in the book). If all goes to plan, you'll see a message for each migration it applies:

```
(venv)C:\Users\Bharat\Documents\AlumniPortal\>
```

```
python manage.py migrate

Operations to perform:
Apply all migrations: admin, auth, contenttypes, sessions
Running migrations:
Applying contenttypes.0001_initial... OK
Applying auth.0001_initial... OK
Applying admin.0001_initial... OK
### several more migrations (not shown)
```

## 7.6 The Development Server

Let's verify your Django project works. Change into the outer `AlumniPortal` directory, if you haven't already, and run the following commands:

```
python manage.py runserver
```

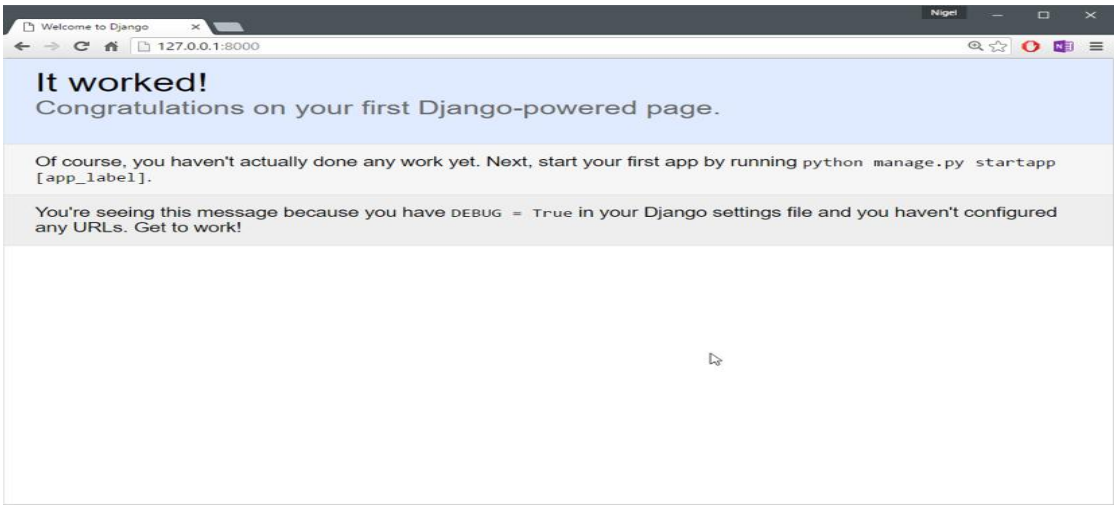
You'll see the following output on the command line:

```
Performing system checks...

System check identified no issues (0 silenced).
May 16, 2017 - 16:48:29
Django version 1.11, using settings 'AlumniPortal.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

You've started the Django development server, a lightweight Web server written purely in Python. Django's creators included this with Django so you can develop things rapidly, without having to deal with configuring a production server – such as Apache – until you're ready for production.

Now's a good time to note: **don't** use this server in anything resembling a production environment. **It's intended only for use while developing**. Now that the server's running, visit `http://127.0.0.1:8000/` with your Web browser. You'll see a "Welcome to Django" page in pleasant, light-blue pastel (Figure 1-3). It worked!



It worked!
Congratulations on your first Django-powered page.

Of course, you haven't actually done any work yet. Next, start your first app by running `python manage.py startapp [app_label]`.

You're seeing this message because you have `DEBUG = True` in your Django settings file and you haven't configured any URLs. Get to work!

Image 3 Django's Welcome Page

## 7.7 Coding Django project

**Create a Django App**
Creating Django App will automatically get views, and models for project, run command inside the Django project as

```
python manage.py startapp dashboard
```

The *dashboard* directory will now have all views, models etc. Files, Create view for the *Portal* inside *views.py*

***views.py***

```
from __future__ import unicode_literals

import os
from django.contrib.auth.hashers import make_password,
check_password
from django.http import HttpResponse
from django.shortcuts import render, redirect

# Create your views here.
from AlumniPortal.settings import BASE_DIR
from imgurpython import ImgurClient

from .forms import PostForm, SignUpForm, LoginForm

from .forms import LikeForm, CommentForm
from .models import UserModel, LikeModel, PostModel, CommentModel,
SessionToken

#Signup view starts here
def SignupView(request):
    if request.method == "POST":
        form = SignUpForm(request.POST)
        if form.is_valid():
            username = form.cleaned_data['username']
            name = form.cleaned_data['name']
            email = form.cleaned_data['email']
            password = form.cleaned_data['password']
            # saving data to DB
            user = UserModel(name=name,
password=make_password(password), email=email, username=username)
            user.save()
            return render(request, 'signup_success.html')
            # return redirect('login/')
    else:
        form = SignUpForm()

    return render(request, 'signup.html', {'form': form})

# Login view starts here
def LoginView(request):
    response_data = {}
    if request.method == "POST":
        form = LoginForm(request.POST)
        if form.is_valid():
            username = form.cleaned_data.get('username')
            password = form.cleaned_data.get('password')
            user =
UserModel.objects.filter(username=username).first()

            if user:
                if check_password(password, user.password):
                    token = SessionToken(user=user)
                    token.create_token()
                    token.save()
                    response = redirect('/')
                    response.set_cookie(key='session_token',
value=token.session_token)
                    return response
                else:
                    response_data['message'] = 'Incorrect Password!
Please try again!'
            else:
                response_data['message'] = 'User does not exist!!!!'

    elif request.method ==  'GET':
        form = LoginForm()

    response_data['form'] = form
    return render(request, 'login.html', response_data)

# Feed view starts here
def FeedView(request):
    user = CheckValidation(request)
    if user:
        posts = PostModel.objects.all().order_by('-created_on')
        for post in posts:
            existing_like = LikeModel.objects.filter(post_id=post.id,
user=user).first()
            if existing_like:
                post.has_liked = True

        return render(request, 'feed.html', {'posts': posts})
    else:

        return redirect('/login/')

# Post view starts here
def PostView(request):
    user = CheckValidation(request)
    if user:
        if request.method == 'POST':
            form = PostForm(request.POST, request.FILES)
            if form.is_valid():
                image = form.cleaned_data.get('image')
                caption = form.cleaned_data.get('caption')

                # saving post in database

                post = PostModel(user=user, image=image,
caption=caption)
                post.save()

                path = os.path.join(BASE_DIR, post.image.url)

                client = ImgurClient('c83158842a9256e',
'ba219c35073b2a80347afaf222e1ebc28dcc8e1a')
                post.image_url = client.upload_from_path(path,
anon=True)['link']

                #post.image_url = cloudinary.uploader.upload(path)
                post.save()
                return redirect('/')

        else:
            form = PostForm()
        return render(request, 'post.html', {'form': form})
    else:

       return redirect('/login/')
```

```
# Like view starts here
def LikeView(request):
    user = CheckValidation(request)
    if user and request.method == 'POST':
        form = LikeForm(request.POST)
        if form.is_valid():
            post_id = form.cleaned_data.get('post').id
            existing_like = LikeModel.objects.filter(post_id=post_id,
user=user).first()
            if not existing_like:
                LikeModel.objects.create(post_id=post_id, user=user)
            else:
                existing_like.delete()
            return redirect('/')
        else:
            return redirect('/login/')

# Comment starts here
def CommentView(request):
    user = CheckValidation(request)
    if user and request.method == 'POST':
        form = CommentForm(request.POST)
        if form.is_valid():
            post_id = form.cleaned_data.get('post').id
            comment_text = form.cleaned_data.get('comment_text')
            comment = CommentModel.objects.create(user=user,
post_id=post_id, comment_text=comment_text)
            comment.save()
            return redirect('/')
        else:
            return redirect('/')
    else:
      return redirect('/login')

# Validation of the session
def CheckValidation(request):
    if request.COOKIES.get('session_token'):
        session =
SessionToken.objects.filter(session_token=request.COOKIES.get('sessi
on_token')).first()
        if session:
           return session.user
    else:
        return None

def LogoutView(request):
    response = redirect("/")
    response.delete_cookie("session_token")
    return response

def ProfileView(request):
    user = CheckValidation(request)
    if user:
        return HttpResponse("Work in progress. Come back later.")
    return redirect("/login/")
def MembersView(request):
    user = CheckValidation(request)
    if user:
        return render(request, 'members.html')
def EventsView(request):
    user = CheckValidation(request)
    if user:
        return render(request,'events.html')
def AboutView(request):
    user = CheckValidation(request)
    if user:
        return render(request, 'about_us.html')
def ContactView(request):
    user = CheckValidation(request)
    if user:
        return render(request, 'contact_us.html')
```

Now create models and forms for the database and session control in *models.py* and *forms.py* files

***models.py***

```
# -*- coding: utf-8 -*-
from __future__ import unicode_literals

from django.db import models
import uuid

# Create your models here.
class UserModel(models.Model):
    name = models.CharField(max_length=120)
    username = models.CharField(max_length=120)
    email = models.EmailField()
    password = models.CharField(max_length=40)
    created_on = models.DateTimeField(auto_now_add=True)
    updated_on = models.DateTimeField(auto_now=True)

class SessionToken(models.Model):
    user = models.ForeignKey(UserModel)
    session_token = models.CharField(max_length=255)
    last_request_on = models.DateTimeField(auto_now=True)
    created_on = models.DateTimeField(auto_now_add=True)
    is_valid = models.BooleanField(default=True)

    def create_token(self):
        self.session_token = uuid.uuid4()

class PostModel(models.Model):
    user = models.ForeignKey(UserModel)
    image = models.FileField(upload_to='user_images')
    image_url = models.CharField(max_length=255)
    caption = models.CharField(max_length=240)
    created_on = models.DateTimeField(auto_now_add=True)
    updated_on = models.DateTimeField(auto_now=True)
    has_liked = False

    @property
    def like_count(self):
        return len(LikeModel.objects.filter(post=self))
    @property
    def comments(self):
        return CommentModel.objects.filter(post=self).order_by('-
created_on')

class LikeModel(models.Model):
    user = models.ForeignKey(UserModel)
    post = models.ForeignKey(PostModel)
    created_on = models.DateTimeField(auto_now_add=True)
    updated_on = models.DateTimeField(auto_now=True)

class CommentModel(models.Model):
    user = models.ForeignKey(UserModel)
    post = models.ForeignKey(PostModel)
    comment_text = models.CharField(max_length=555)
    created_on = models.DateTimeField(auto_now_add=True)
    updated_on = models.DateTimeField(auto_now=True)
```

***forms.py***

```
from django import forms
from .models import UserModel, PostModel, LikeModel, CommentModel

class SignUpForm(forms.ModelForm):
    class Meta:
        model = UserModel
        fields=['email','username','name','password']

class LoginForm(forms.ModelForm):
    class Meta:
        model = UserModel
        fields = ['username', 'password']

class PostForm(forms.ModelForm):
    class Meta:
        model = PostModel
        fields=['image', 'caption']


class LikeForm(forms.ModelForm):

    class Meta:
        model = LikeModel
        fields=['post']


class CommentForm(forms.ModelForm):

    class Meta:
        model = CommentModel
        fields = ['comment_text', 'post']
```

For templates create a directory as *templates* in *dashboard* and store static files as in *static* directory (like js,css, and images). Write html inside templates directory.

***login.html***

```
<!Doctype html>
<html lang="en">
    <head>
        {% include "head.html"%}

        <title>Login</title>
    </head>
    <body class="home">
        {% include "navbar.html" %}

        <!--Login Form starts here-->
<div class="container form-top-margin-100">
    <div class="row">
        <div class="col-sm-3">

        </div>
        <div class="col-xs-12 col-sm-6">

        </div>
        <div class="col-sm-3">

        </div>
    </div>
    <div class="row">
        <div class="col-sm-5">
            <img src="../static/images/alumni-first-logo.png"
width="30%" height="30%">

        </div>

            <form class="form-horizontal" method="post">
              {% csrf_token %}
                        <div class="col-xs-12 col-sm-3 text-center">
              <div class="form-group">
                <label for="username" class="control-label"></label>
                <div class="col-sm-12" id="username-div">
                    <div class="input-group">
                        <span class="input-group-addon" id="usename-
span">@</span>
                        <input type="text" class="form-control"
id="username" name="username" value="{{ username }}"
placeholder="Username" aria-describedby="username-span">
                    </div>
                </div>
                  <div>Not a member yet ? <a href="/signup">
Register Now</a></div>
              </div>                 </div>
                <div class="col-xs-12 col-sm-3 text-center">
              <div class="form-group">
                <label for="password" class="control-label"></label>
                <div class="col-sm-12" id="password-div">
                    <div class="input-group">
                        <span class="input-group-addon"><i
class="glyphicon glyphicon-option-horizontal"></i></span>
                        <input type="password" class="form-control"
id="password" name="password" value="{{ password }}"
placeholder="Password"></div>
                    </div>
                  <div><a href="/profile"> Forgot
Password</a></u></div>

              </div></div>
                            <div class="col-xs-12 col-sm-1 text-
center">
              <div class="form-group">
                <div class="col-sm-offset-2 col-sm-3">
                  <button type="submit" class="btn btn-success form-
btn">Sign in</button>
                </div>
              </div></div>
            </form>
        </div>
        <div class="col-sm-3">
            <p class="error-message">{{ message }}</p>
        </div>
    </div>
        <div>
        <img src="../static/images/Lets-Stay-Connected-300x47.png">
        </div>
        <div align="right">
         <div class="container">
               <div class="row">
                       <div class="[ col-xs-12 col-sm-offset-7 col-
sm-5 ]">
                               <ul class="event-list">
                                       <li>
                                               <time datetime="2014-
07-20">
                                                       <span
class="day">4</span>
                                                       <span
class="month">Jul</span>
                                                       <span
class="year">2018</span>
```

```
                                                        <span
class="time">ALL DAY</span>
                                                </time>
                                                <img alt="Independence
Day"
src="https://farm4.staticflickr.com/3100/2693171833_3545fb852c_q.jpg
" />
                                                <div class="info">
                                                        <h2
class="title">Delhi Alumni Meet</h2>
                                                        <p
class="desc">We will be meet at delhi. </p>
                                                </div>
                                                <div class="social">
                                                        <ul>
                                                                <li
class="facebook" style="width:33%;"><a href="#facebook"><span
class="fa fa-facebook"></span></a></li>
                                                                <li
class="twitter" style="width:34%;"><a href="#twitter"><span
class="fa fa-twitter"></span></a></li>
                                                                <li
class="google-plus" style="width:33%;"><a href="#google-plus"><span
class="fa fa-google-plus"></span></a></li>
                                                        </ul>
                                                </div>
                                        </li>

                                        <li>
                                                <time datetime="2014-
07-20 0000">
                                                        <span
class="day">8</span>
                                                        <span
class="month">Jul</span>
                                                        <span
class="year">2018</span>
                                                        <span
class="time">12:00 AM</span>
                                                </time>
                                                <div class="info">
                                                        <h2
class="title">Pune Alumni Meet</h2>
```

```
                                                        <p
class="desc">All alumnus please come at : 56 Street, Ramesh
Nagar</p>
                                                        <ul>
                                                                <li
style="width:50%;"><a href="#website"><span class="fa fa-
globe"></span> Register for event</a></li>
                                                                <li
style="width:50%;"><span class="fa fa-money"></span> $5.00 </li>
                                                        </ul>
                                                </div>
                                                <div class="social">
                                                        <ul>
                                                                <li
class="facebook" style="width:33%;"><a href="#facebook"><span
class="fa fa-facebook"></span></a></li>
                                                                <li
class="twitter" style="width:34%;"><a href="#twitter"><span
class="fa fa-twitter"></span></a></li>
                                                                <li
class="google-plus" style="width:33%;"><a href="#google-plus"><span
class="fa fa-google-plus"></span></a></li>
                                                        </ul>
                                                </div>
                                        </li>

                                </ul>
                        </div>
                </div>
        </div>

        </div>
        <!--Login form ends here-->

        {% include "footer.html" %}

        {% include "scripts.html" %}
    </body>

</html>
```

***signup.html***

```
<!Doctype html>
<html lang="en">
```

```
    <head>
        {% include "head.html"%}

        <title>Login</title>
    </head>
    <body class="home">
        {% include "navbar.html" %}

        <!--Login Form starts here-->
<div class="container form-top-margin-100">
    <div class="row">
        <div class="col-sm-3">

        </div>
        <div class="col-xs-12 col-sm-6">

        </div>
        <div class="col-sm-3">

        </div>
    </div>
    <div class="row">
        <div class="col-sm-5">
            <img src="../static/images/alumni-first-logo.png"
width="30%" height="30%"><br>
            <div>
        <img src="../static/images/Lets-Stay-Connected-300x47.png">
        </div>
        </div>

         <div class="col-xs-12 col-sm-6">
    <img src="../static/images/register_now_button_2.png"
width="50%" height="50%">
    <form class="form-horizontal" method="post">
              {% csrf_token %}
              <div class="form-group">
                <label for="username" class="col-sm-2 control-
label"></label>
                <div class="col-sm-10" id="username-div">
                    <div class="input-group">
                        <span class="input-group-addon"
id="username-span">@</span>
```

```
                        <input type="text" class="form-control"
id="username" name="username" value="{{ username }}"
placeholder="Username" aria-describedby="username-span">
                    </div>
              </div>
              <div class="form-group">
                <label for="name" class="col-sm-2 control-
label"></label>
                <div class="col-sm-10" id="name-div"><br>
                    <div class="input-group">
                        <span class="input-group-addon"><i
class="glyphicon glyphicon-user"></i></span>
                        <input type="text" class="form-control"
id="name" name="name" value="{{ name }}" placeholder="Name"  >
                    </div>
                </div>
              </div>
              <div class="form-group">
                <label for="email" class="col-sm-2 control-
label"></label>
                <div class="col-sm-10" id="email-div">
                    <div class="input-group">
                        <span class="input-group-addon"><i
class="glyphicon glyphicon-envelope"></i></span>
                        <input type="email" class="form-control"
id="email" name="email" value="{{ email }}" placeholder="Email">
                    </div>
                </div>
              </div>
              <div class="form-group" id="password-div">
                <label for="password" class="col-sm-2 control-
label"></label>
                <div class="col-sm-10">
                    <div class="input-group">
                        <span class="input-group-addon"><i
class="glyphicon glyphicon-option-horizontal"></i></span>
                        <input type="password" class="form-control"
id="password" name="password" value="{{ password }}"
placeholder="Password">
                    </div>
                </div>
              </div>
              <div class="form-group">
                <div class="col-sm-offset-2 col-sm-5">
                <button type="submit" class="btn btn-success form-
btn">Register</button>
               </div>
               <div class="col-sm-5">
                <a href="/login"> <button type="button" class="btn
btn-primary form-btn">Login</button></a>
               </div>
             </div>
             </div></form>
        </div>

    </div>
        </div>
        <!--Login form ends here-->

        {% include "footer.html" %}

        {% include "scripts.html" %}
    </body>

</html>
```

All other *html pages* should also be coded in the same manner inside the *templates* directory.

## 8. Testing

Automated testing is an extremely useful bug-killing tool for the modern Web developer. You can use a collection of tests – a test suite – to solve, or avoid, a number of problems:

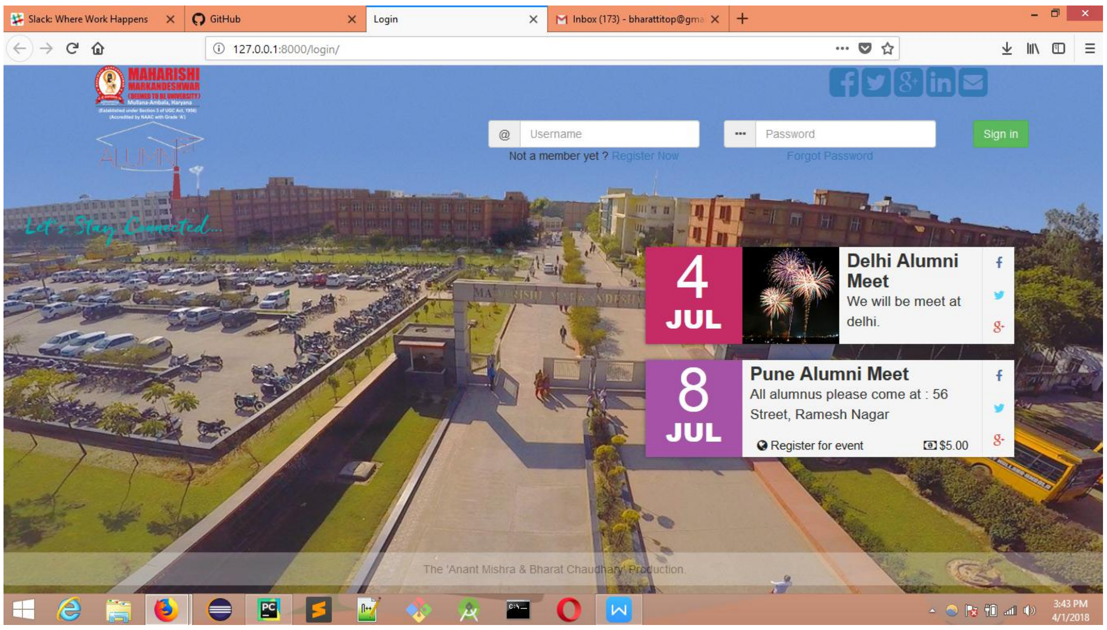
- When you're writing new code, you can use tests to validate your code works as expected.
- When you're refactoring or modifying old code, you can use tests to ensure your changes haven't affected your application's behavior unexpectedly.

Testing a Web application is a complex task, because a Web application is made of several layers of logic – from HTTP-level request handling, to form validation and processing, to template rendering. With Django's test-execution framework and assorted utilities, you can simulate requests, insert test data, inspect your application's output and generally verify your code is doing what it should be doing.

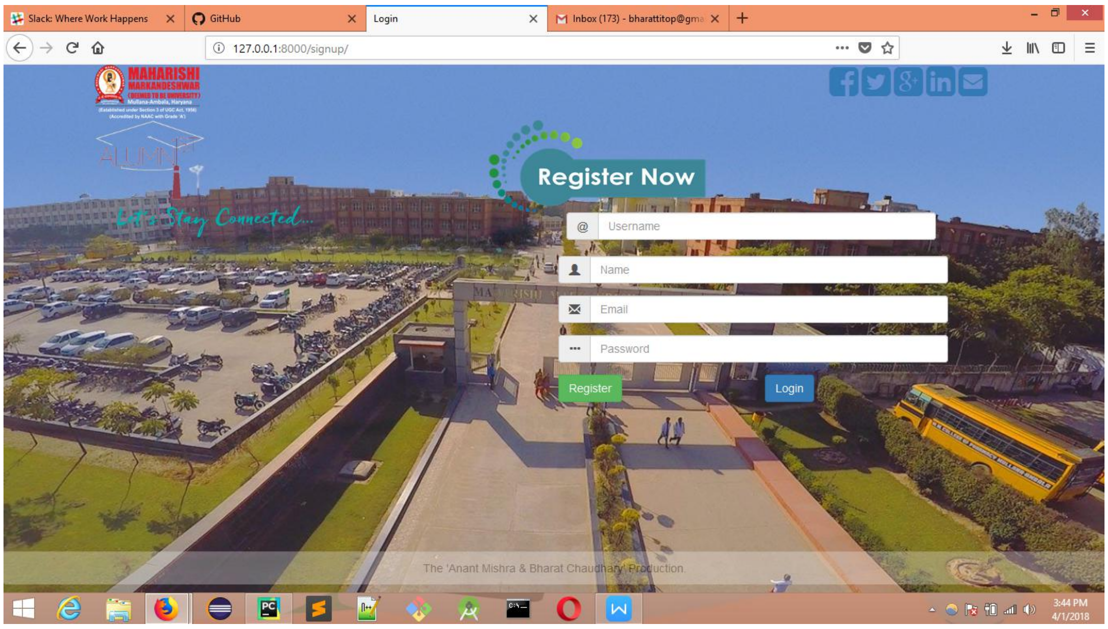
The best part is, it's really easy.

The preferred way to write tests in Django is using the *unittest* module built in to the Python standard library.

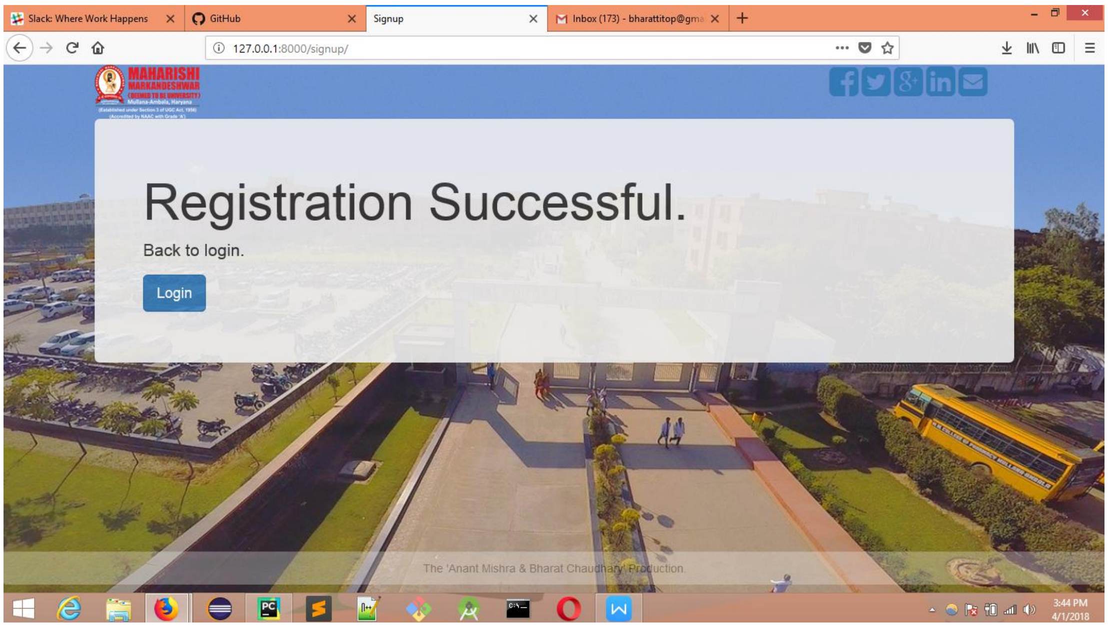
# 9. Screenshots


Slack: Where Work Happens
GitHub
Login
Inbox (173) - bharattitop@gma
127.0.0.1:8000/login/
MAHARISHI MARKANDESHWAR
ALUMNI
Let's Stay Connected...
Username
Password
Sign in
Not a member yet ? Register Now
Forgot Password
4 JUL
Delhi Alumni Meet
We will be meet at delhi.
8 JUL
Pune Alumni Meet
All alumnus please come at : 56 Street, Ramesh Nagar
Register for event
$5.00
The 'Anant Mishra & Bharat Chaudhary' Production.
3:43 PM 4/1/2018

Screenshot 9.1 Home Page



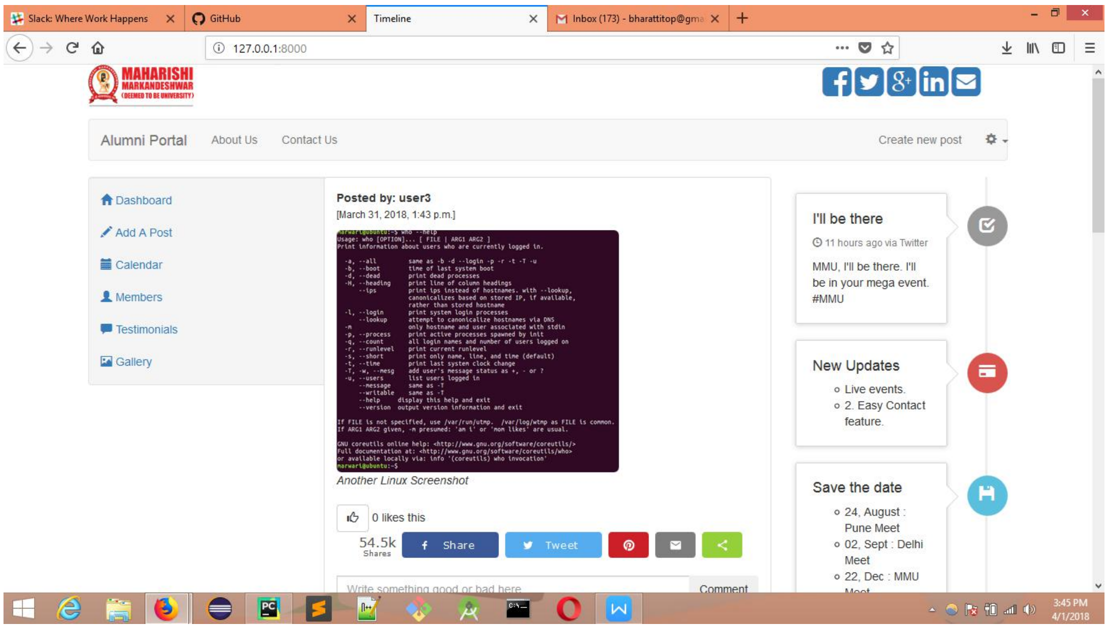
Screenshot 9.2 Registration Page


Slack: Where Work Happens
GitHub
Signup
Inbox (173) - bharattitop@gma
127.0.0.1:8000/signup/
MAHARISHI MARKANDESHWAR
Login
The 'Anant Mishra & Bharat Chaudhary' Production.
3:44 PM
4/1/2018

Registration Successful.

Back to login.

Screenshot 9.3 Registration Success Message Page



**Posted by: user3**
[March 31, 2018, 1:43 p.m.]

```
marwari@ubuntu:~$ who --help
Usage: who [OPTION]... [ FILE | ARG1 ARG2 ]
Print information about users who are currently logged in.

  -a, --all         same as -b -d --login -p -r -t -T -u
  -b, --boot        time of last system boot
  -d, --dead        print dead processes
  -H, --heading     print line of column headings
      --ips         print ips instead of hostnames. with --lookup,
                    canonicalizes based on stored IP, if available,
                    rather than stored hostname
  -l, --login       print system login processes
      --lookup      attempt to canonicalize hostnames via DNS
  -m                only hostname and user associated with stdin
  -p, --process     print active processes spawned by init
  -q, --count       all login names and number of users logged on
  -r, --runlevel    print current runlevel
  -s, --short       print only name, line, and time (default)
  -t, --time        print last system clock change
  -T, -w, --mesg    add user's message status as +, - or ?
  -u, --users       list users logged in
      --message     same as -T
      --writable    same as -T
      --help     display this help and exit
      --version  output version information and exit

If FILE is not specified, use /var/run/utmp.  /var/log/wtmp as FILE is common.
If ARG1 ARG2 given, -m presumed: 'am i' or 'mom likes' are usual.

GNU coreutils online help: <http://www.gnu.org/software/coreutils/>
Full documentation at: <http://www.gnu.org/software/coreutils/who>
or available locally via: info '(coreutils) who invocation'
marwari@ubuntu:~$
```

*Another Linux Screenshot*

0 likes this

54.5k Shares

Write something good or bad here

I'll be there

11 hours ago via Twitter

MMU, I'll be there. I'll be in your mega event. #MMU

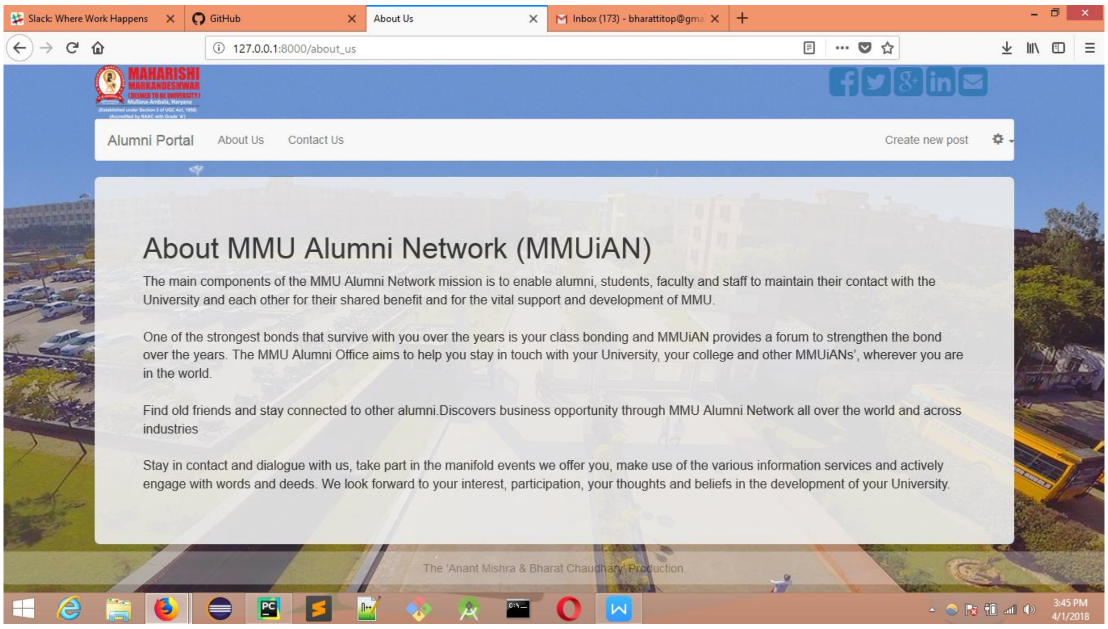
New Updates

- Live events.
- 2. Easy Contact feature.

Save the date

- 24, August : Pune Meet
- 02, Sept : Delhi Meet
- 22, Dec : MMU Meet

Screenshot 9.4 (After Login) Feed Page



## About MMU Alumni Network (MMUiAN)

The main components of the MMU Alumni Network mission is to enable alumni, students, faculty and staff to maintain their contact with the University and each other for their shared benefit and for the vital support and development of MMU.

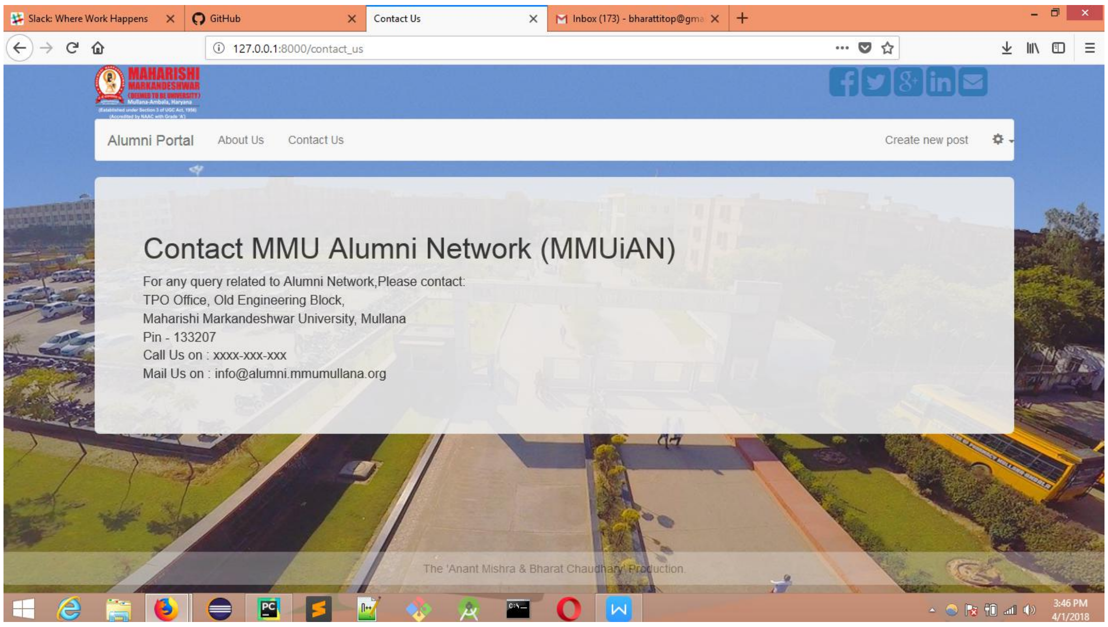
One of the strongest bonds that survive with you over the years is your class bonding and MMUiAN provides a forum to strengthen the bond over the years. The MMU Alumni Office aims to help you stay in touch with your University, your college and other MMUiANs', wherever you are in the world.

Find old friends and stay connected to other alumni.Discovers business opportunity through MMU Alumni Network all over the world and across industries

Stay in contact and dialogue with us, take part in the manifold events we offer you, make use of the various information services and actively engage with words and deeds. We look forward to your interest, participation, your thoughts and beliefs in the development of your University.

The 'Anant Mishra & Bharat Chaudhary' Production.

Screenshot 9.5 About Us Page

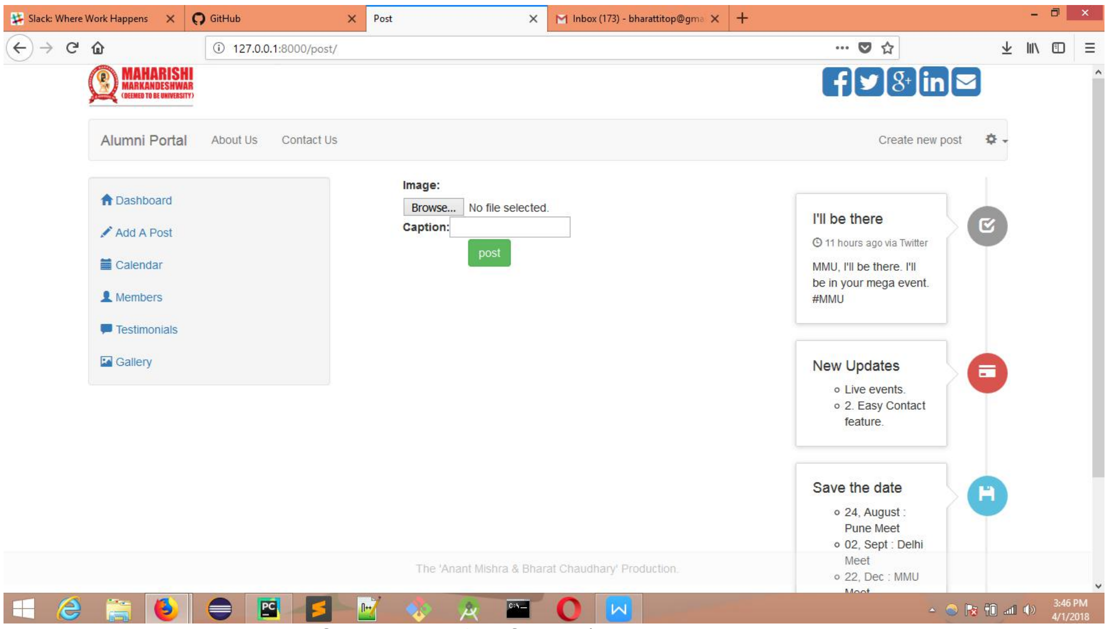

Slack: Where Work Happens
GitHub
Contact Us
Inbox (173) - bharattitop@gma
127.0.0.1:8000/contact_us
Alumni Portal
About Us
Contact Us
Create new post

## Contact MMU Alumni Network (MMUiAN)

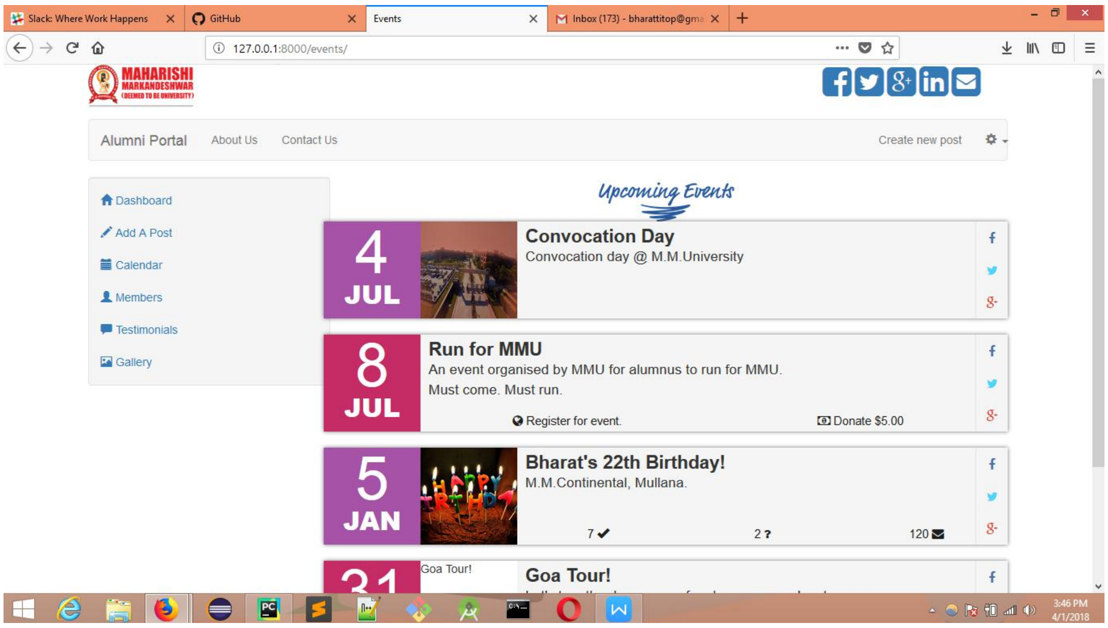
For any query related to Alumni Network,Please contact:
TPO Office, Old Engineering Block,
Maharishi Markandeshwar University, Mullana
Pin - 133207
Call Us on : xxxx-xxx-xxx
Mail Us on : info@alumni.mmumullana.org

The 'Anant Mishra & Bharat Chaudhary' Production.

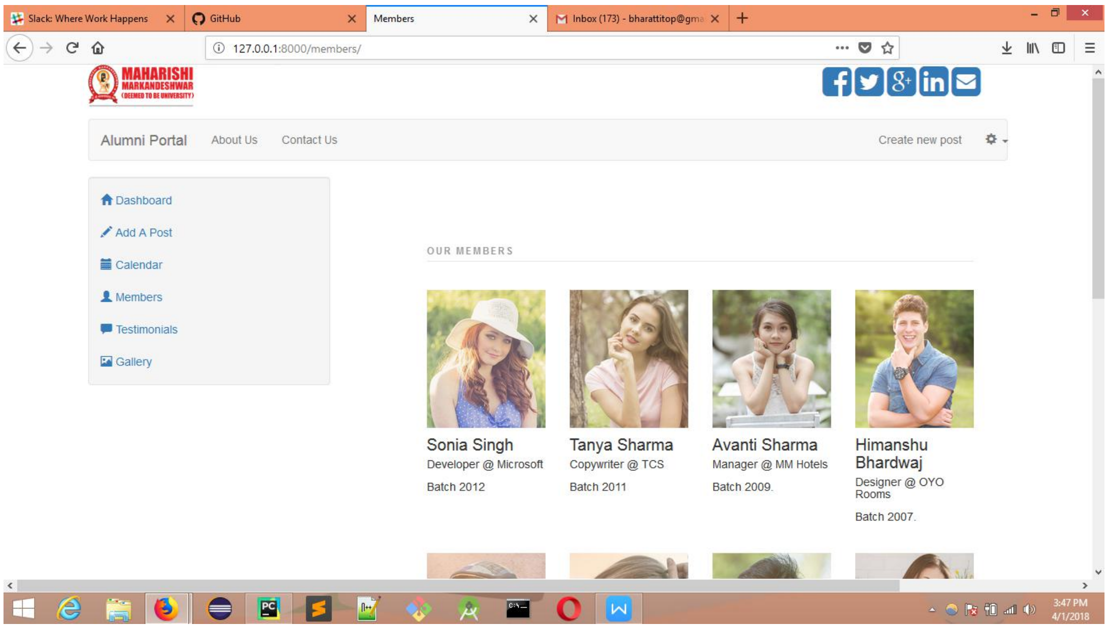
Screenshot 9.6 Contact Us Page



Screenshot 9.7 Create/Add a Post Page



Screenshot 9.8 Events Page



OUR MEMBERS

Sonia Singh
Developer @ Microsoft
Batch 2012

Tanya Sharma
Copywriter @ TCS
Batch 2011

Avanti Sharma
Manager @ MM Hotels
Batch 2009.

Himanshu Bhardwaj
Designer @ OYO Rooms
Batch 2007.

Screenshot 9.9 Members Page

## 10. References

1. Python official website ( https://www.python.org )
2. Django official website ( https://www.djangoproject.com )