MAJOR PROJECT REPORT

ON

Information System for Student Performance

Submitted by

E. Keerthi Aishwarya (13B81A0566)

G. Manoj Kumar (13B81A0583)

Pritheesh Panchmahalkar (13B81A05B6)

Under Supervision of

Dr. A. Vani Vathsala

Professor of Department of Computer Science

CVR College of Engineering

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

CVR COLLEGE OF ENGINEERING

An Autonomous Institution

(Affiliated to JNTU University, Hyderabad)

Mangalpally, Ibrahimpatnam-501 510

2016-2017

CERTIFICATE

This is to certify that the Project Report entitled **"Information System on students Performance"** is a bonafide record of work carried out by **E.Keerthi Aishwarya (13B81A0566)**, **G. Manoj Kumar (13B81A0583)** and **Pritheesh Panchmahalkar (13B81A05B6)** under my guidance and supervision in partial fulfillment of the requirement for the award of the degree of Bachelor of Technology in Computer Science Engineering of Jawaharlal Nehru Technological University, Hyderabad during the academic year 2016-2017.

Dr.A.Vani Vathsala
Project Guide
Professor
Department of CSE
CVR college of Engineering

Prof .L.C. Siva Reddy
Head of the Department
Department of CSE
CVR college of Engineering

External Examiner

DECLARATION

We hereby declare that the project report entitled “Information System on students Performance " submitted by us to CVR College of Engineering, in partial fulfillment of the requirement for the award of the degree of B.Tech, in Computer Science Engineering is a record of bonafide project work carried out by us under the guidance of **Dr.A.Vani Vathsala**. We further declare that the work reported in this project has not been submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Signature of the Student

E.Keerthi Aishwarya

Signature of the Student

G. Manoj Kumar

Signature of the Student

Pritheesh Panchmahalkar

ACKNOWLEDGEMENT

Apart from the efforts of the team, the success of any project depends largely on the encouragement and guidelines of many other. We take this opportunity to express our gratitude to the people who have been instrumental in the successful completion of this project.

We are also thankful to the Head of the Department **Prof. L.C. Siva Reddy** for providing excellent infrastructure and a nice atmosphere for completing this project successfully.

We take this opportunity to express our profound gratitude and deep regards one more time to our guide **Dr.A.Vani Vathsala** of CSE department for her exemplary guidance, monitoring and constant encouragement throughout the project work. The blessing, help and guidance given by her, time to time shall carry us a long way in the journey of life on which we are about to embark.

The guidance and support received from the faculty members of department of CSE of **CVR College of Engineering** who contributed to this project, was vital for the success of the project. We are grateful for their constant support and help.

Finally, we would like to take this opportunity to thank our families for their constant encouragement without which this assignment would not be possible. We sincerely acknowledge and thank all those who gave support directly or indirectly in completion of this project.

E.Keerthi Aishwarya (13B81A0566)

G. Manoj Kumar (13B81A0583)

Pritheesh Panchmahalkar (13B81A05B6)

ABSTRACT

INFORMATION SYSTEM ON STUDENTS PERFORMANCE

With the widespread use of Internet and Information technology, it is becoming inevitable for educational institutions to provide digital campus facility. The credibility of an esteemed college is easily reflected by the efficiency of its operational practices. Digitizing information associated with key functionalities is a crucial part of such process.

In order to start the movement towards achieving such digitization, an incremental procedure is recommended. As the first step of this process, we propose to develop a web application that enables parents to keep track of their ward's performance in each semester online. This application acts as a centralized resource of the performance of all the students. We achieve the above by categorizing key functionalities into the following six modules: parent registration module, student registration module, student performance information module for parents, student performance information module for students, achievements module for parents, achievements module for students.

TABLE OF CONTENTS

1. INTRODUCTION

.

Every citizen is digitally empowered and all information is digitally available. Since the world is on its way to a complete digital age, it'll be beneficial to have easy access to digital records of everything. Most of the prestigious colleges provide a very elegant interface on the internet for accessing all the information. Implementing digital campus enables parents and faculty to keep track of student's performance in academics. Data conversion of historic data (transcripts, attendance, etc.) for both current and past students can also be a significant issue when considering a transition to a newer student information system. Since most colleges are required to keep historical data of past students, considerations should be given to what information will be converted and what will be archived. Hence, we propose this project as a step in that direction for our college.

Parent registration module: Parents will have to provide the details of their email IDs, mobile numbers and wards details whose performance they would like to monitor. Parents will also have to provide login credentials like username and password. After submitting these details, they will be validated.

Student registration module: Students will have to provide the details of their mobile numbers and emails IDs which they have given at the time of admission. Students will also have to provide login credentials like username and password. After submitting these details, they will be validated.

Student performance Information module for Parents: Parents should log-in using their user id and password given during registration. After successfully logging in, semester wise mid marks and semester end exam marks of their respective wards will be displayed. Thus, parents can access the performance of their ward from anywhere. The system assures ease of use for all the users, as users need not remember the roll Number of their ward. If more than one child of a user is studying in the same college, information of both the children would be displayed. The web-application also rules out the associated disadvantaged with SMS, like information/messages regarding marks not reaching the parents.

Student performance Information module for students: Students should log-in using their user id and password given during registration. After successfully logging in, semester wise mid marks and semester end exam marks will be displayed. Thus, even students can access the results from anywhere.

Achievements module for parents: Parents can view the achievements of their wards' if they have any. E.g.: If their ward secures highest marks in the semester or highest marks in a subject for that semester.

Achievements module for students: Students can view their achievements if they have secure highest marks in the semester or highest marks in a subject for that semester.

2. SOFTWARE REQUIREMENT SPECIFICATIONS

2.1 System Environment

Presentation Layer	:	HTML, Bootstrap
Network Layer	:	TCP/IP
Web Server Layer	:	JSP
Languages	:	Python
Databases	:	MySql Workbench
Framework	:	Pycharm (IDE for Django)
Operating Systems	:	Windows 7, MacOS
Browsers	:	Chrome, Firefox, Opera, Safari

2.2 FUNCTIONAL REQUIREMENTS

Application must provide the following functionality

- The User Interface should be user friendly to the person who uses the online login and sign-up.
- Admin should be able to do the required operations such as entering students and parents information, semester results and updating them.
- User should be able to view the Semester wise results with ease.
- User should be able to view their achievements if they have secured highest marks in the semester or highest marks in a subject for that semester.

2.3 MODULES

- Parents registration module
- Students registration module
- Students performance module for parents
- Students performance module for students
- Achievements module for parents
- Achievements module for students

Parent registration module:

- Parents will have to provide the details of their email IDs, mobile numbers and wards details whose performance they would like to monitor.
- Parents will also have to provide login credentials like username and password.
- After submitting these details, they will be validated.
  The following validations take place:
  1) Mobile number verification is done.
  2) Passwords should contain more than 8 characters.
  3) Emails should be a valid one (i.e. it should contain '@' field).
  4) Username should be unique.

Student registration module:

- Students will have to provide the details of their mobile numbers and emails IDs which they have given at the time of admission.
- Students will also have to provide login credentials like username and password.
- After submitting these details, they will be validated.
  The following validations take place:
  1) Mobile number verification is done.

2) Passwords should contain more than 8 characters.
3) Emails should be a valid one (i.e. it should contain ‘@’ field).
4) Username should be unique.

Student performance Information module for Parents:

- Parents should log-in using their user ID and password given during registration.
- After successfully logging in, semester wise mid marks (internal exam) and semester end exam marks (external exam) of their respective wards will be displayed.
- Thus, parents should be able to access the performance of their ward from anywhere.
- The system assures ease of use for all the users, as users need not remember the Roll Number of their ward.
- If more than one child of a user is studying in the same college, information of both the children would be displayed.
- The web-application also rules out the associated disadvantaged with SMS, like information/messages regarding marks not reaching the parents.

Student performance Information module for students:

- Students should log-in using their user ID and password given during registration.
- After successfully logging in, semester wise mid marks and semester end exam marks will be displayed.
- Thus, even students should be able to access the results from anywhere.

Achievements module for parents:

- Parents can view the achievements of their wards’ if they have any.
- E.g.: If their ward secures highest marks in the semester or highest marks in a subject for that semester.

Achievements module for students:

- Students can view their achievements if they have secure highest marks in the semester or highest marks in a subject for that semester.

2.4 NON-FUNCTIONAL REQUIREMENTS

Scalability: System must be scalable to handle the load on the server.

Reliability: System must provide reliable data backup.

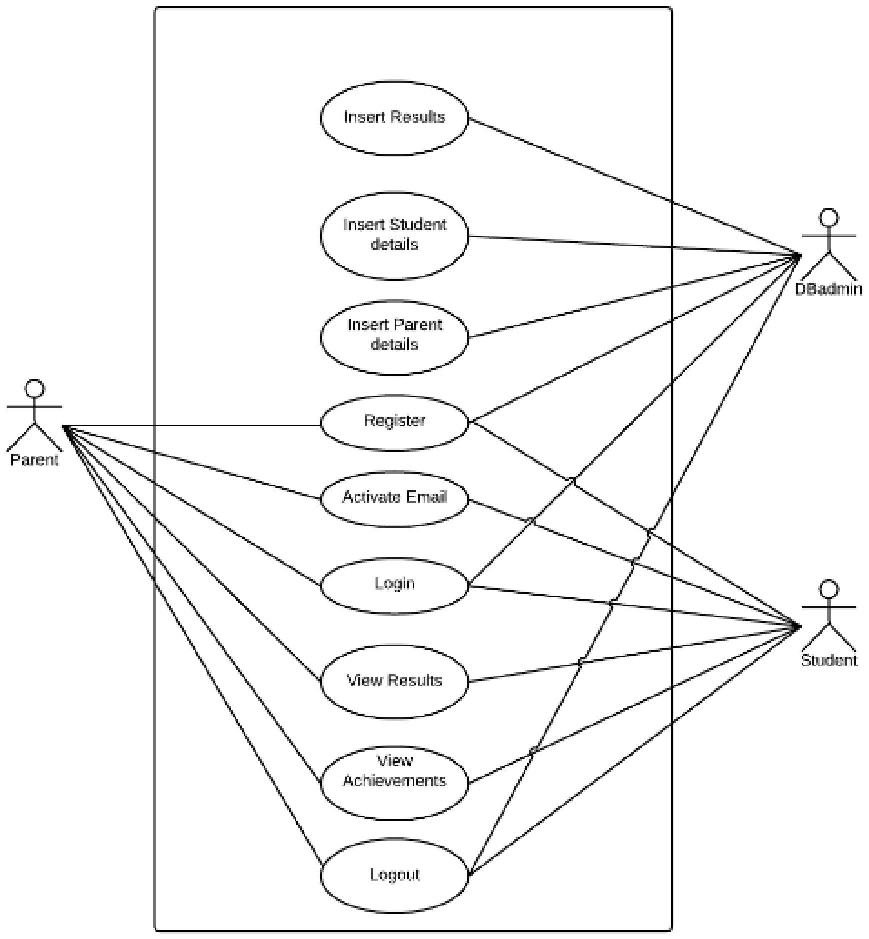
Availability: Application should be available at anytime.

3. DESIGN

3.1 Use Case Diagram

The below use-case shows list of actions between admin , students and parent to achieve a goal. Parent and student share the activation, registration, login, viewing results, viewing achievements and logout modules.

Use case diagram

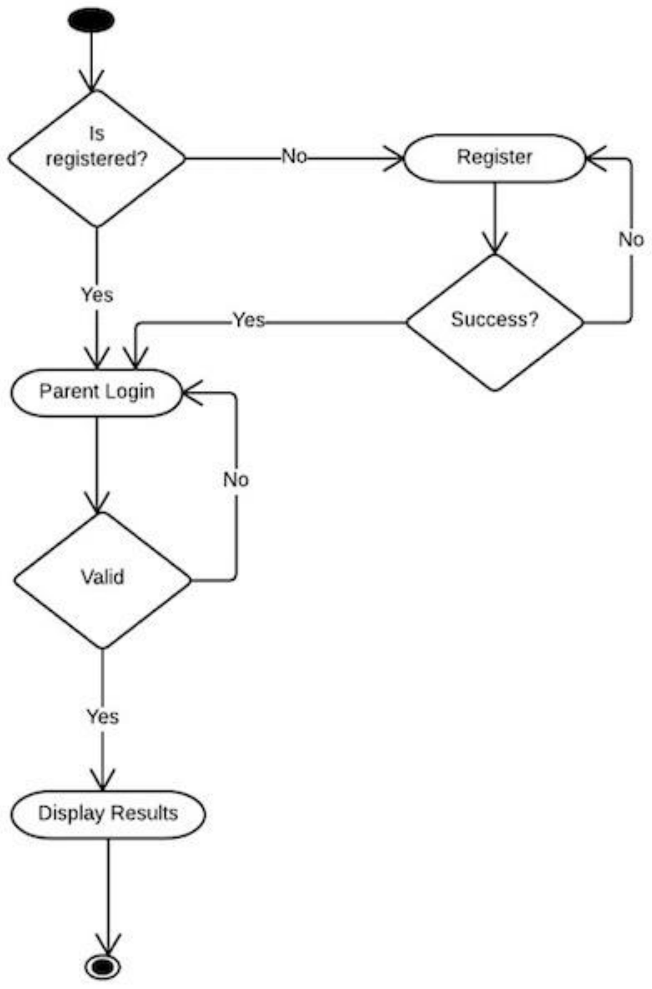


3.2 Activity Diagrams

.

The below activity diagram shows the flow of control from one activity to another activity of parent starting from registration till displaying results.

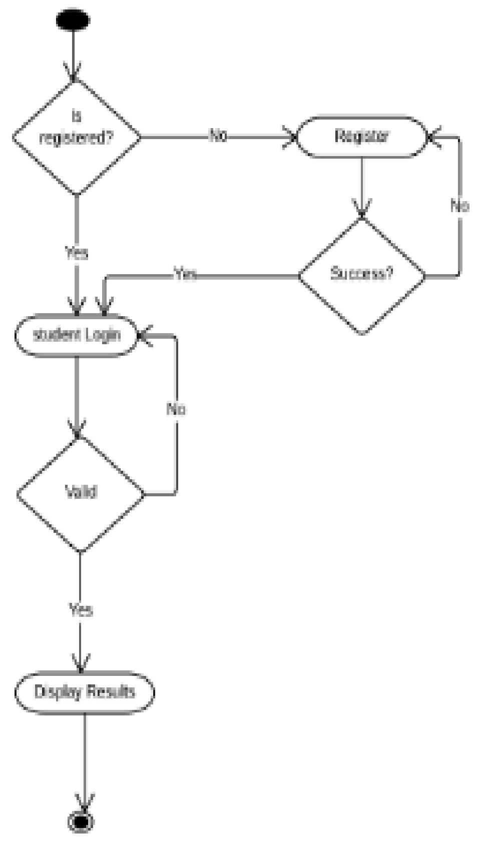
Activity Diagram for Parent:



The user is first shown a login page if the user is not registered he/she may register using the given registration link. After successful registration the user will be given an activation link sent to his/her mail. Once he is activated he will be redirected to login page where he has to login. If the user is successfully logged in he can view the results and achievement.

Activity Diagram for Student:

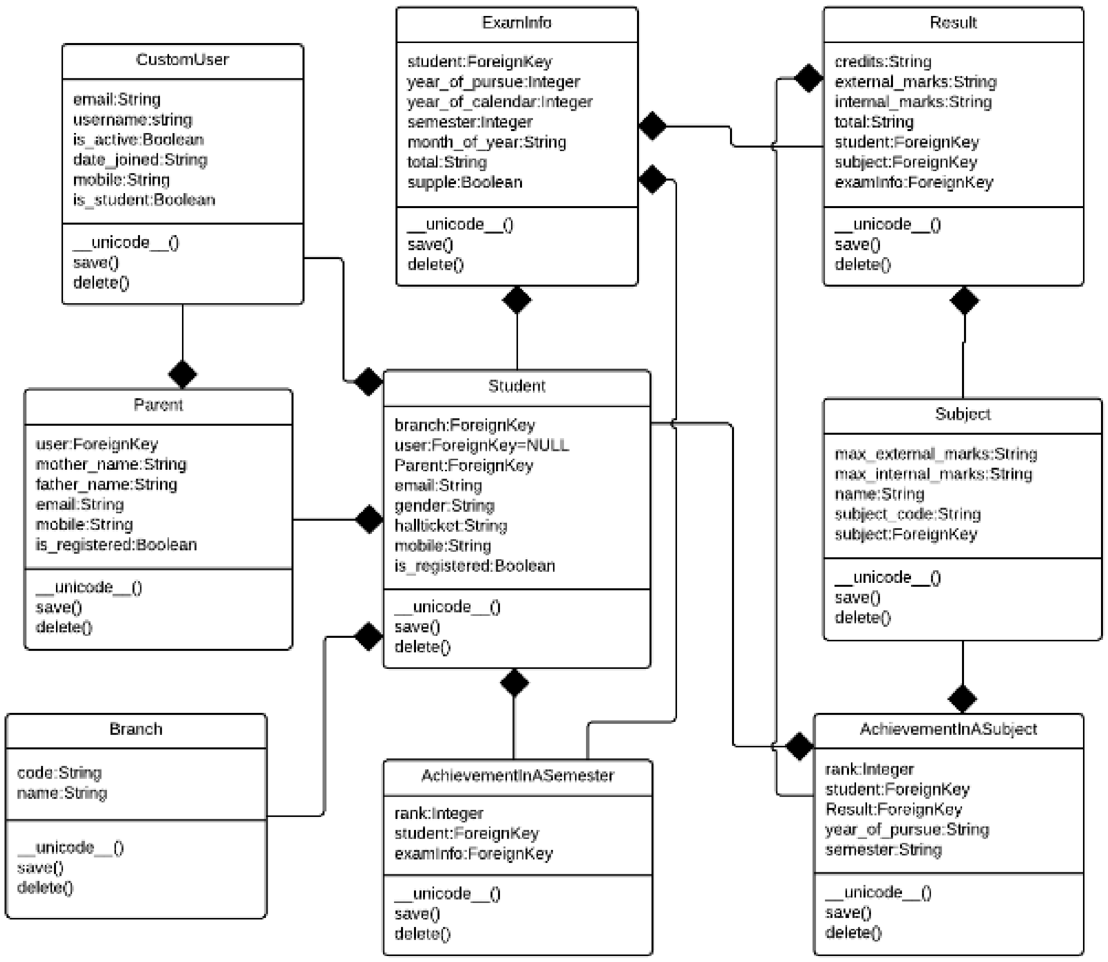
The below activity diagram shows the flow of control from one activity to another activity of parent starting from registration till displaying results.



The user is first shown a login page if the user is not registered he/she may register using the given registration link. After successful registration the user will be given an activation link sent to his/her mail. Once he is activated he will be redirected to login page where he/she has to login. If the user is successfully logged in he can view the results and achievements.

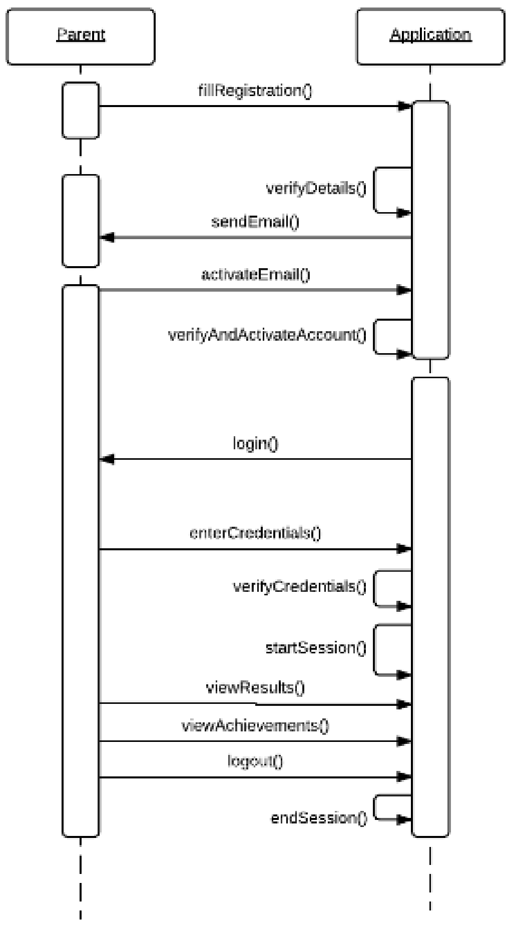
3.3 Class Diagram

A class diagram is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods) and the relationships among objects.



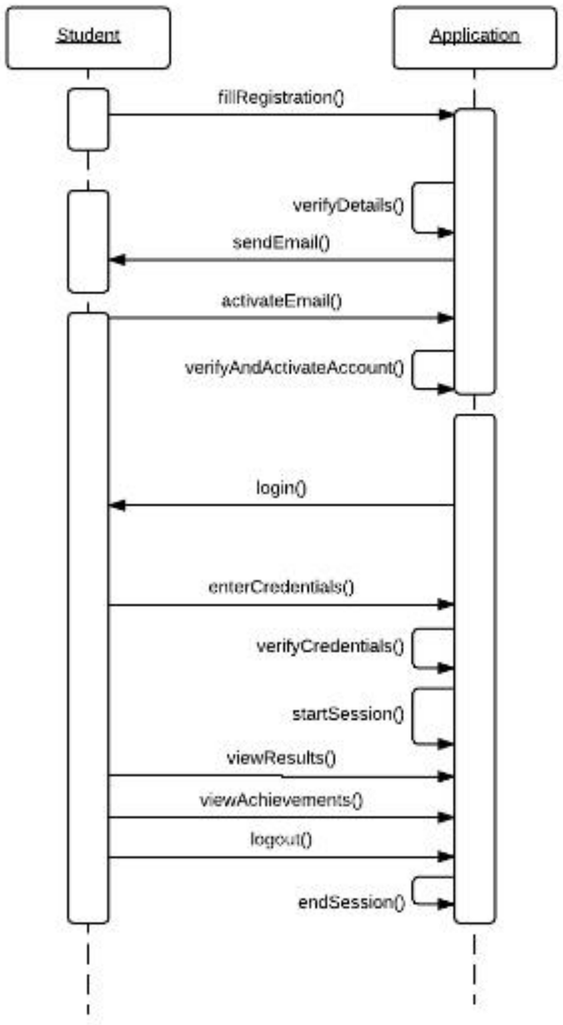
3.4 Sequence Diagram

Sequence diagram for parent



The above sequence diagram shows the sequence of actions done by a parent from registration till logout. The user is first shown a login page if the user is not registered he/she may register using the given registration link. After successful registration the user will be given an activation link sent to his/her mail. Once he is activated he will be redirected to login page where he/she has to login. If the user is successfully logged in he can view the results and achievements.

Sequence diagram for student



The above sequence diagram shows the sequence of actions done by the student from registration till logout. The user is first shown a login page if the user is not registered he/she may register using the given registration link. The student needs to check the check box in the registration link. After successful registration the user will be given an activation link sent to his/her mail. Once he is activated he will be redirected to login page where he/she has to login. If the user is successfully logged in he can view the results and achievements.

3.5 Tables for modeling the database

The tables used for modeling the database are

CustomUser table

Attribute	Type	Primary Key	Foreign Key
Id	Int	Yes	No
Username	String	No	No
Password	String	No	No
Email	String	No	No
Is_active	Boolean	No	No
Date_joined	String	No	No
Mobile	String	No	No
Is_student	Boolean	No	No

This table stores user information about user which is used for login and registration. The is_active field is used to check if the user account is activated or not. The is_student field is used to check if the user is parent or student. Date_joined is used when the user created his account. Password s are stretched by using PBKDF2 algorithm with a shah256 hash.

Parent table

Attribute	Type	Primary Key	Foreign Key
Id	Int	Yes	No
Mother_name	String	No	No
Father_name	String	No	No
Mobile	String	No	No
Email	String	No	No
Is_registered	Boolean	No	No
CustomUser	Int	No	Yes

This table stores information about parent details which is used for validation of parent details. If the is_registered is true for a parent then he/she cannot register again. The mobile field is validated during parent registration . A mail is sent to the parent's email for activation without which parent cannot login. The parent table is populated using the provided excel sheet.

Branch table

Attribute	**Type**	**Primary Key**	**Foreign Key**
Id	Int	Yes	No
Code	String	No	No
Name	String	No	No

The Branch table is used for storing information related to branches in the college to which the students belong to.

Student table

Attribute	**Type**	**Primary Key**	**Foreign Key**
Id	Int	Yes	No
Parent	Int	No	Yes
CustomUser	Int	No	Yes
Branch	Int	No	Yes
Name	String	No	No
Hall_ticket	String	No	No
Gender	String	No	No
Mobile	String	No	No
Email	String	No	No
Is _registered	Boolean	No	No

The student table stores the information about the students studying in the college. Since the student can register and login the table is related to CustomUser table. The branch information of the student is obtained from the branch table. If the is_registered is true for a parent then he/she cannot register again .The mobile field and email field are validated during student registration . A mail is sent to the student's email for activation without which parent cannot login. The student table is populated using the provided excel sheet.

Subject table

Attribute	**Type**	**Primary Key**	**Foreign Key**
Id	Int	Yes	No
Subject_code	String	No	No
Name	String	No	No
Max_internal_marks	Int	No	No
Max_external_marks	Int	No	No

The subject table stores the information about the subjects in various branches in various semesters. The subject table is populated using the provided excel sheet.

ExamInfo table

Attribute	**Type**	**Primary Key**	**Foreign Key**
Id	Int	Yes	No
Student	String	No	Yes
Year_of_pursue_roman	String	No	No
Semester_roman	String	No	No
Year_of_pursure	String	No	No
Semester	Int	No	No
Year_of_calendar	Int	No	No
Month_of_year	String	No	No
Supple	Boolean	No	No
Total	String	No	No

This table stores the information of a semester exam about a student. It is related to student table by the foreign key, student. The total field stores the semester total of the student in that semester. The supple field denotes whether the exam is main or supple. The two fields with their names ending with roman are used for templating.

Result table

Attribute	Type	Primary Key	Foreign Key
Id	Int	Yes	No
Subject	Int	No	Yes
Examinfo	Int	No	Yes
Internal_marks	String	No	No
External_marks	String	No	No
Total	String	No	No
Results	String	No	No
Credits	Int	No	No

This table stores the information of the results about a student for each subject in a semester. It is related to the ExamInfo table by the foreign key, examinfo. It is also related to the Subject table by the foreign key, subject. The fields internal_marks and external_marks specify the marks secured by student.

AchievementInASemester table

Attribute	Type	Primary Key	Foreign Key
Id	Int	Yes	No
Rank	Int	No	No
Student	Int	No	Yes
Examinfo	Int	No	Yes

This table stores the information about the rank obtained by the student in a semester based on the total marks obtained by the student in that semester. It is related to the tables Student and Examinfo by the foreign keys student and examinfo respectively.

AchievementInASubject table

Attribute	Type	Primary Key	Foreign Key
Id	Int	Yes	No
Rank	Int	No	No
Student	Int	No	Yes
Result	String	No	No
Year_of_pursue_roman	String	No	No
Semester_roman	String	No	No
Year_of_pursure	Int	No	No
Semester	Int	No	No

This table stores the information about the rank obtained by the student in a subject in a semester based on marks obtained. It is related to the Student table by the foreign key, student.

SaltForActivation table

Attribute	Type	Primary Key	Foreign Key
Id	Int	Yes	No
User	Int	No	Yes

This table stores the information about the user who has registered and is used during activation process. If the user is activated the tuple related to the user in this table is deleted.

4. IMPLEMENTATION

Models.py –

This file is used to create tables in the databases. In our web application we have 8 tables, each model maps to a single database table. Each attribute of the model represents a database field.

```
from __future__ import unicode_literals

from django.contrib.auth.models import AbstractUser
from django.db import models

# Create your models here.
class CustomUser(AbstractUser):
    mobile = models.CharField(max_length=15)
    is_student = models.BooleanField(verbose_name="Student", default=False)


class Parent(models.Model):
    user = models.OneToOneField(CustomUser, null=True)
    mother_name = models.CharField(max_length=128)
    father_name = models.CharField(max_length=128)
    mobile = models.CharField(max_length=15, unique=True)
    email = models.CharField(max_length=128, null=True)
    is_registered = models.BooleanField(default=False)

    def __unicode__(self):
        return self.father_name


class Branch(models.Model):
    code = models.CharField(max_length=2)
    name = models.CharField(max_length=64)

    def __unicode__(self):
        return self.name


class Student(models.Model):
    user = models.OneToOneField(CustomUser, null=True)
    parent = models.ForeignKey(Parent)
    branch = models.ForeignKey(Branch)
    name = models.CharField(max_length=128)
    hall_ticket = models.CharField(max_length=10, unique=True)
    gender = models.CharField(max_length=6)
    mobile = models.CharField(max_length=15, null=True)
    email = models.CharField(max_length=128, null=True)
    is_registered = models.BooleanField(default=False)

    def __unicode__(self):
        return self.hall_ticket




class Subject(models.Model):
    subject_code = models.CharField(max_length=10, unique=True)
    name = models.CharField(max_length=128)
    max_internal_marks = models.IntegerField(null=True)
    max_external_marks = models.IntegerField(null=True)

    def __unicode__(self):
        return self.name


class ExamInfo(models.Model):
    student = models.ForeignKey(Student, related_name='examinfo')
    year_of_pursue_roman = models.CharField(max_length=5)
    semester_roman = models.CharField(max_length=2)
    year_of_pursue = models.IntegerField()
    semester = models.IntegerField()
    year_of_calendar = models.IntegerField()
    month_of_year = models.CharField(max_length=15)
    supple = models.BooleanField()
    total = models.CharField(max_length=3, default='0')

    def __unicode__(self):
        if self.supple == False:
            desc = ' Main'
        else:
            desc = ' Supple'
        return self.year_of_pursue_roman+" "+self.semester_roman+desc

    class Meta:
        ordering = ['year_of_pursue', 'semester']


class Result(models.Model):
    subject = models.ForeignKey(Subject, related_name='subjects')
    examinfo = models.ForeignKey(ExamInfo, related_name='result')
    internal_marks = models.CharField(max_length=3)
    external_marks = models.CharField(max_length=3)
    total = models.CharField(max_length=3)
    results = models.CharField(max_length=5)
    credits = models.IntegerField()

    def __unicode__(self):
        return self.subject.name



class AchievementInASemester(models.Model):
    rank = models.IntegerField()
    student = models.ForeignKey(Student)
    examinfo = models.ForeignKey(ExamInfo)

    def __unicode__(self):
        return self.student.hall_ticket+" "+self.examinfo.year_of_pursue_roman+"
"+self.examinfo.semester_roman+" "+str(self.rank)

class AchievementInASubject(models.Model):
    rank = models.IntegerField()
    student = models.ForeignKey(Student)
    result = models.ForeignKey(Result)
    year_of_pursue_roman = models.CharField(max_length=5)
    semester_roman = models.CharField(max_length=2)
    semester = models.IntegerField()
    year_of_pursue = models.IntegerField()

    def __unicode__(self):
        return self.student.hall_ticket+" "+self.result.subject.name+"
"+str(self.rank)
```

forms.py

This file is used to create form such as registration form, login form and validating them.

```
from django import forms
import re

from django.contrib.auth import authenticate, login
from django.contrib.auth.models import User
from django.core.validators import validate_email

from InfoSystem.models import Student, Parent
from .models import CustomUser

class UserForm(forms.Form):
    email = forms.EmailField()
    mobile =forms.CharField()

    class Meta:
        model = CustomUser
        fields = ('username', 'email', 'password', 'mobile', 'is_student')


class UserRegistrationForm(forms.Form):
    username = forms.CharField(max_length=30)
    email = forms.EmailField()
    mobile = forms.CharField(max_length=15)
    isstudent = forms.NullBooleanField(required=False)
    password1 = forms.CharField(widget=forms.PasswordInput())
    password2 = forms.CharField(widget=forms.PasswordInput())

    def clean_username(self):
        username = self.cleaned_data['username']
        if not re.search(r'^\w+$', username):
            raise forms.ValidationError("Username can only contain alphanumeric 
characters")
        try:
            CustomUser.objects.get(username=username)
        except:
            return username
        raise forms.ValidationError("Username already exists")

    def clean_email(self):
        email = self.cleaned_data['email']
        try:
            validate_email(email)
            return email
        except:
            raise forms.ValidationError("Enter correct Email")

    def clean_mobile(self):
        mobile = self.cleaned_data['mobile']
        if 'isstudent' in self.data:
        #     is_student = True
        # else:
        #     is_student = False
        # if is_student is True:
            try:
                stud = Student.objects.get(mobile=mobile)
            except:
                raise forms.ValidationError("User with given mobile number doesn't 
exist")
            if stud.is_registered is True:
                raise forms.ValidationError("Student with given mobile number 
already exists")
```

```
            return mobile
        else:
            try:
                par = Parent.objects.get(mobile=mobile)
            except:
                raise forms.ValidationError("User with given mobile number doesn't
exist")
            if par.is_registered is True:
                raise forms.ValidationError("Parent with given mobile number
already exists")
            return mobile


    def clean_password2(self):
        if 'password1' in self.cleaned_data:
            password1 = self.cleaned_data['password1']
            password2 = self.cleaned_data['password2']
            if password1 == password2:
                return password2
            raise forms.ValidationError("Passwords do not match")

    # def clean(self):
    #     self.clean_password2()
    #     self.clean_mobile()
    #     self.clean_email()
    #     self.clean_username()

class UserLoginForm(forms.Form):
    username = forms.CharField(max_length=30)
    password = forms.CharField(widget=forms.PasswordInput())

    def clean(self):
        user = authenticate(username = self.cleaned_data['username'],
password=self.cleaned_data['password'])
        if user is None:
            raise forms.ValidationError("Invalid username or password. Please try
again!")
```

urls.py

This file is used to create urls which we see in our browser address bar. This file is a pure python code which maps between urls patterns and python functions**.**

```
from django.conf.urls import url
from django.contrib.auth.decorators import login_required
from django.contrib.auth import views as auth_views
from InfoSystem import api_views
from InfoSystem import serializers_views as s_views
from InfoSystem import views
from rest_framework.authtoken import views as rest_views

urlpatterns = [
    # url(r'^api/register/$', api_views.register, name='api-register'),
    # url(r'^api/login/$', api_views.login_user, name='api-login'),
    # url(r'^$', views.index, name='home'),
    url(r'^accounts/login/$', api_views.login_user, name='login'),
    # url(r'^api/logout/$', api_views.logout_view, name='api-logout'),
    url(r'^api/results/$', s_views.StudentList.as_view()),
    url(r'^api/register/$', s_views.UserRegisterView.as_view(), name='api-
register'),
    # url(r'^api/register/student', s_views.StudentRegisterView.as_view(),
name='api-student-register'),


    # url(r'^results/$', api_views.result_view, name='result-view'),
    url(r'^$', auth_views.login, {'redirect_authenticated_user': True},
name='login'),
    url(r'^login/$', auth_views.login, {'redirect_authenticated_user': True},
name='login'),
    url(r'^register/$', views.register, name='register'),
    url(r'^results/$', login_required(views.result_view), name='result-view'),
    url(r'^logout/$', views.logout_view, name='logout'),
]
```

api_views.py

```
from django.contrib.auth import authenticate, login
from django.contrib.auth.decorators import login_required
from django.contrib.auth.views import logout
from django.http.response import HttpResponseRedirect
from django.shortcuts import render, redirect
from django.urls.base import reverse

from InfoSystem.forms import UserForm
from InfoSystem.models import Parent

def index(request):
    return render(request, 'index.html')


def register(request):
    form = UserForm(request.POST or None)
    template_name = 'registration/register.html'

    if form.is_valid():
        user = form.save(commit=False)
        username = form.cleaned_data['username']
        password = form.cleaned_data['password']
        mobile = form.cleaned_data['mobile']
        email = form.cleaned_data['email']
        user.set_password(password)
        par = Parent.objects.get(mobile__exact=mobile)
        if(par.is_registered == False):
            user.save()
            par.is_registered =  True
            par.user = user
            par.email = email
            par.save()
        else:
            return render(request, template_name, {'error_message': 'You are
already registered', 'form':form})
        user = authenticate(username=username, password=password)

        if user is not None:
            if user.is_active:
                login(request, user)
                return redirect('login')

    return render(request, template_name, {'form': form})


def login_user(request):
    if request.user.is_authenticated():
        return HttpResponseRedirect(reverse('result-view'))
    template_name = 'registration/login.html'
    if request.method=='POST':
        username = request.POST['username']
        password = request.POST['password']
        user = authenticate(username=username, password=password)
        if user is not None:
            if user.is_active:
                login(request, user)
                return redirect('result-view')
            else:
                return render(request, template_name, {'error_message': 'Your
account has been disabled'})
        else:
            return render(request, template_name, {'error_message': 'Wrong
Username/Password. Try again'})

    return render(request, template_name)


@login_required
def result_view(request):
    template_name = 'result-view.html'
    par = Parent.objects.get(user_id=request.user.id)
    students = par.student_set.all()
    return render(request, template_name, {'students': students})


def logout_view(request):
    logout(request)
    return redirect('login')
```

settings.py

This file contains all the configuration of Django installation.

```
"""
Django settings for MajorProject1 project.


import os

# Build paths inside the project like this: os.path.join(BASE_DIR, ...)
from django.urls.base import reverse_lazy

BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))


SECRET_KEY = 'jntx+%9@)#(g2n@jnu+mv-cv)_g*$uuaskdy%xx&ccn=3nqg&s'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = []


# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'InfoSystem',
    'rest_framework',
    'rest_framework.authtoken',
    'djoser',
]
```

```
DJOSER = {
    'SERIALIZERS': {
        'user_registration': 'InfoSystem.serializers.UserRegisterSerializer',
    },
}

REST_FRAMEWORK = {
    'DEFAULT_PERMISSION_CLASSES': (
        'rest_framework.permissions.IsAuthenticated',
    ),
    'DEFAULT_AUTHENTICATION_CLASSES': (
        'rest_framework.authentication.TokenAuthentication',
        'rest_framework.authentication.BasicAuthentication',
        'rest_framework.authentication.SessionAuthentication',
    ),
}


MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF = 'MajorProject1.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [os.path.join(BASE_DIR, 'templates')]
        ,
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]

WSGI_APPLICATION = 'MajorProject1.wsgi.application'


# Database
# https://docs.djangoproject.com/en/1.10/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'infotestdb',
        'USER': 'root',
        'PASSWORD': '2812',
        'HOST': 'localhost',
        'PORT': '3306'
    }
}

AUTH_USER_MODEL = 'InfoSystem.CustomUser'

LOGIN_REDIRECT_URL=reverse_lazy('result-view')
```

```
LOGIN_URL = reverse_lazy('login')

# Password validation
# https://docs.djangoproject.com/en/1.10/ref/settings/#auth-password-validators

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME':
'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]


# Internationalization
# https://docs.djangoproject.com/en/1.10/topics/i18n/

LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'UTC'

USE_I18N = True

USE_L10N = True

USE_TZ = True


# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/1.10/howto/static-files/

STATIC_URL = '/static/'
```

Populate.py

This file is used to populate the database with parent, student information and the student results.

```
from _mysql import IntegrityError

import MySQLdb
import xlrd
from InfoSystem.models import Parent, Student, Subject, Result, ExamInfo, Branch
from MajorProject1 import settings
import sys
import os
import django

django.setup()

os.environ.setdefault("DJANGO_SETTINGS_MODULE", "MajorProject1.settings")
#populating branch table
branch = Branch(code='01', name='Civil Engineering')
branch.save()
branch = Branch(code='02', name='Electrical and Electronic Engineering')
branch.save()
```

```
branch = Branch(code='03', name='Mechanical Engineering')
branch.save()
branch = Branch(code='04', name='Electronic and Communication Engineering')
branch.save()
branch = Branch(code='05', name='Computer Science Engineering')
branch.save()
branch = Branch(code='10', name='Electronic and Instrumentation Engineering')
branch.save()
branch = Branch(code='12', name='Information Technology')
branch.save()

#populate parent table
# Open the workbook and define the worksheet
book = xlrd.open_workbook("info.xls")


for i in range(0, 7):
    sheet = book.sheet_by_index(i)
    for row in range(5, sheet.nrows):
        father_name = str(sheet.cell(row, 9).value).strip().title()
        try:
            father_mobile = str(int(sheet.cell(row, 11).value)).strip()
        except:
            father_mobile = None
        print "Father: ", father_name
        mother_name = str(sheet.cell(row, 10).value).strip().title()
        try:
            par = Parent(father_name=father_name, mobile=father_mobile,
mother_name=mother_name)
            par.save()
        except:
            pass

#populate student table
for i in range(0, 7):
    sheet = book.sheet_by_index(i)
    for row in range(5, sheet.nrows):
        try:
            father_mobile = str(int(sheet.cell(row, 11).value)).strip()
        except:
            father_mobile=None
        hall_ticket = str(sheet.cell(row, 1).value).strip()
        print hall_ticket
        name = str(sheet.cell(row, 2).value).strip().title()
        # print "student: ", name
        try:
            gender = int(sheet.cell(row, 8).value)
        except:
            gender = None
        if gender == 0:
            gender = 'MALE'
        elif gender == 1:
            gender = 'FEMALE'
        try:
            student_mobile = str(int(sheet.cell(row, 12).value)).strip()
        except:
            student_mobile = 0
        branch = Branch.objects.get(code__exact=hall_ticket[6:8])
        email = str(sheet.cell(row, 13).value).strip()
        object = Parent.objects.filter(mobile=father_mobile)[0]
        stud = Student(name=name, hall_ticket=hall_ticket, gender=gender,
mobile=student_mobile, email=email,
                       parent=object, branch=branch)
        stud.save()


#populate subject table
book = xlrd.open_workbook("results.xls")
```

```
sheet = book.sheet_by_name("TSheet")

for row in range(4, sheet.nrows):
    sub_code = str(sheet.cell(row, 2).value).strip()
    sub_name = str(sheet.cell(row, 3).value).strip()
    sub = Subject.objects.filter(subject_code=sub_code)
    if sub.count()==0:
        sub = Subject(subject_code=sub_code, name=sub_name)
        sub.save()


#populate ExamInfo table
# exam_object = ExamInfo(year_of_calendar=2016, month_of_year=4, year_of_pursue=3,
semester=2, supple=False)
# exam_object.save()

#populate result table
months = ['January', 'February', 'March', 'April', 'May', 'June', 'July', 'August',
'September', 'October', 'November',
          'December']
hall2 = ""
for row in range(4, sheet.nrows):
    try:
        int_marks = str(int(sheet.cell(row, 4).value))
    except:
        int_marks = str(sheet.cell(row, 4).value)
    try:
        ext_marks = str(int(sheet.cell(row, 5).value))
    except:
        ext_marks = str(sheet.cell(row, 5).value)
    total_marks = 0
    try:
        total_marks += int(int_marks)
    except:
        print "the guy was absent buddy for internal"
    try:
        total_marks += int(ext_marks)
    except:
        print "the guy was absent buddy for external"
    res = str(sheet.cell(row, 7).value).strip()
    credits = int(sheet.cell(row, 8).value)
    try:
        hall_ticket = str(sheet.cell(row, 0).value).strip()
        hall1 = hall_ticket

        print hall_ticket, " in results table"
        stud = Student.objects.get(hall_ticket=hall_ticket)

        if hall1 != hall2: #Populating the ExamInfo table
            exam_object = ExamInfo(year_of_calendar=2016, month_of_year=months[3],
year_of_pursue_roman='III',
                                   semester_roman='II', year_of_pursue=3,
semester=2, supple=False, student=stud)
            exam_object.save()

        hall2 = hall1

        sub_code = str(sheet.cell(row, 2).value).strip()
        sub = Subject.objects.get(subject_code=sub_code)
        #Modify the below code accordingly
        exam_object = ExamInfo.objects.get(student=stud, year_of_pursue=3,
semester=2)
        exam_object.total = str(int(exam_object.total) + total_marks)
        exam_object.save()
        result = Result(subject=sub, internal_marks=int_marks,
external_marks=ext_marks, results=res,
                        credits=credits, examinfo=exam_object,
total=str(total_marks))
```

```
        result.save()
    except:
        print sys.exc_info()
```

populate2.py

This script is used to populate the database with other semesters information.

```
import xlrd
from InfoSystem.models import Parent, Student, Subject, Result, ExamInfo, Branch
from MajorProject1 import settings
import sys
import os
import django

django.setup()

os.environ.setdefault("DJANGO_SETTINGS_MODULE", "MajorProject1.settings")
book = xlrd.open_workbook("results2.xls")

sheet = book.sheet_by_name("Sheet1")
for row in range(sheet.nrows):
    sub_code = str(sheet.cell(row, 2).value).strip()
    sub_name = str(sheet.cell(row, 3).value).strip()
    sub = Subject.objects.filter(subject_code=sub_code)
    if sub.count() == 0:
        sub = Subject(subject_code=sub_code, name=sub_name)
        sub.save()

months = ['January', 'February', 'March', 'April', 'May', 'June', 'July', 'August',
'September', 'October', 'November',
          'December']
hall2 = ""
i = 0
for row in range(sheet.nrows):
    try:
        int_marks = str(int(sheet.cell(row, 4).value))
    except:
        int_marks = str(sheet.cell(row, 4).value)
    try:
        ext_marks = str(int(sheet.cell(row, 5).value))
    except:
        ext_marks = str(sheet.cell(row, 5).value)

    total_marks = 0
    try:
        total_marks += int(int_marks)
    except:
        print "buddy was absent for internal"
    try:
        total_marks += int(ext_marks)
    except:
        print "Buddy was absent for external"
    res = str(sheet.cell(row, 7).value).strip()
    credits = int(sheet.cell(row, 8).value)
    try:
        hall_ticket = str(sheet.cell(row, 0).value).strip()
        # hall1 = hall_ticket

        print hall_ticket, " in results table"
        stud = Student.objects.get(hall_ticket=hall_ticket)

        # if hall1 != hall2:  # Populating the ExamInfo table
        if i == 0:
            exam_object = ExamInfo(year_of_calendar=2013, month_of_year=months[11],
year_of_pursue_roman='I',
                                        semester_roman='I', year_of_pursue=1,
semester=1, supple=False, student=stud)
            exam_object.save()
            i=1

        # hall2 = hall1

        sub_code = str(sheet.cell(row, 2).value).strip()
        sub = Subject.objects.get(subject_code=sub_code)
        # Modify the below code accordingly
        exam_object = ExamInfo.objects.get(student=stud, year_of_pursue=1,
semester=1)
        exam_object.total = str(int(exam_object.total) + total_marks)
        exam_object.save()
        result = Result(subject=sub, internal_marks=int_marks,
external_marks=ext_marks, results=res,
                        credits=credits, examinfo=exam_object,
total=str(total_marks))
        result.save()
    except:
        print sys.exc_info()

sheet = book.sheet_by_name("Sheet2")
for row in range(sheet.nrows):
    sub_code = str(sheet.cell(row, 2).value).strip()
    sub_name = str(sheet.cell(row, 3).value).strip()
    sub = Subject.objects.filter(subject_code=sub_code)
    if sub.count() == 0:
        sub = Subject(subject_code=sub_code, name=sub_name)
        sub.save()

# months = ['January', 'February', 'March', 'April', 'May', 'June', 'July',
'August', 'September', 'October',
#                 'November', 'December']
# hall2 = ""
i=0
for row in range(sheet.nrows):
    try:
        int_marks = str(int(sheet.cell(row, 4).value))
    except:
        int_marks = str(sheet.cell(row, 4).value)
    try:
        ext_marks = str(int(sheet.cell(row, 5).value))
    except:
        ext_marks = str(sheet.cell(row, 5).value)
    total_marks = 0
    try:
        total_marks += int(int_marks)
    except:
        print "buddy was absent for internal"
    try:
        total_marks += int(ext_marks)
    except:
        print "Buddy was absent for external"
    res = str(sheet.cell(row, 7).value).strip()
    credits = int(sheet.cell(row, 8).value)
    try:
        hall_ticket = str(sheet.cell(row, 0).value).strip()
        # hall1 = hall_ticket

        print hall_ticket, " in results table"
        stud = Student.objects.get(hall_ticket=hall_ticket)

        # if hall1 != hall2:  # Populating the ExamInfo table
        if i == 0:
            exam_object = ExamInfo(year_of_calendar=2014, month_of_year=months[3],
year_of_pursue_roman='I',
                                           semester_roman='II', year_of_pursue=1,
semester=2, supple=False, student=stud)
            exam_object.save()
            i=1

        # hall2 = hall1

        sub_code = str(sheet.cell(row, 2).value).strip()
        sub = Subject.objects.get(subject_code=sub_code)
        # Modify the below code accordingly
        exam_object = ExamInfo.objects.get(student=stud, year_of_pursue=1,
semester=2)
        exam_object.total = str(int(exam_object.total) + total_marks)
        exam_object.save()
        result = Result(subject=sub, internal_marks=int_marks,
external_marks=ext_marks, results=res,
                               credits=credits, examinfo=exam_object,
total=str(total_marks))
        result.save()
    except:
        print sys.exc_info()

sheet = book.sheet_by_name("Sheet3")
for row in range(sheet.nrows):
    sub_code = str(sheet.cell(row, 2).value).strip()
    sub_name = str(sheet.cell(row, 3).value).strip()
    sub = Subject.objects.filter(subject_code=sub_code)
    if sub.count() == 0:
        sub = Subject(subject_code=sub_code, name=sub_name)
        sub.save()

# months = ['January', 'February', 'March', 'April', 'May', 'June', 'July',
'August', 'September', 'October',
#                'November', 'December']
# hall2 = ""
i=0
for row in range(sheet.nrows):
    try:
        int_marks = str(int(sheet.cell(row, 4).value))
    except:
        int_marks = str(sheet.cell(row, 4).value)
    try:
        ext_marks = str(int(sheet.cell(row, 5).value))
    except:
        ext_marks = str(sheet.cell(row, 5).value)
    total_marks = 0
    try:
        total_marks += int(int_marks)
    except:
        print "buddy was absent for internal"
    try:
        total_marks += int(ext_marks)
    except:
        print "Buddy was absent for external"
    res = str(sheet.cell(row, 7).value).strip()
    credits = int(sheet.cell(row, 8).value)
    try:
        hall_ticket = str(sheet.cell(row, 0).value).strip()
        # hall1 = hall_ticket

        print hall_ticket, " in results table"
        stud = Student.objects.get(hall_ticket=hall_ticket)

        # if hall1 != hall2:
        #  Populating the ExamInfo table
        if i == 0:
            exam_object = ExamInfo(year_of_calendar=2014, month_of_year=months[7],
year_of_pursue_roman='II',
                                             semester_roman='I', year_of_pursue=2,
semester=1, supple=False, student=stud)
            exam_object.save()
            i=1

        # hall2 = hall1

        sub_code = str(sheet.cell(row, 2).value).strip()
        sub = Subject.objects.get(subject_code=sub_code)
        # Modify the below code accordingly
        exam_object = ExamInfo.objects.get(student=stud, year_of_pursue=2,
semester=1)
        exam_object.total = str(int(exam_object.total) + total_marks)
        exam_object.save()
        result = Result(subject=sub, internal_marks=int_marks,
external_marks=ext_marks, results=res,
                                   credits=credits, examinfo=exam_object,
total=str(total_marks))
        result.save()
    except:
        print sys.exc_info()

sheet = book.sheet_by_name("Sheet4")
for row in range(sheet.nrows):
    sub_code = str(sheet.cell(row, 2).value).strip()
    sub_name = str(sheet.cell(row, 3).value).strip()
    sub = Subject.objects.filter(subject_code=sub_code)
    if sub.count() == 0:
        sub = Subject(subject_code=sub_code, name=sub_name)
        sub.save()

i = 0
# months = ['January', 'February', 'March', 'April', 'May', 'June', 'July',
'August', 'September', 'October',
#                'November', 'December']
# hall2 = ""
for row in range(sheet.nrows):
    try:
        int_marks = str(int(sheet.cell(row, 4).value))
    except:
        int_marks = str(sheet.cell(row, 4).value)
    try:
        ext_marks = str(int(sheet.cell(row, 5).value))
    except:
        ext_marks = str(sheet.cell(row, 5).value)
    total_marks = 0
    try:
        total_marks += int(int_marks)
    except:
        print "buddy was absent for internal"
    try:
        total_marks += int(ext_marks)
    except:
        print "Buddy was absent for external"
    res = str(sheet.cell(row, 7).value).strip()
    credits = int(sheet.cell(row, 8).value)
    try:
        hall_ticket = str(sheet.cell(row, 0).value).strip()
        # hall1 = hall_ticket

        print hall_ticket, " in results table"
        stud = Student.objects.get(hall_ticket=hall_ticket)

        # if hall1 != hall2:  # Populating the ExamInfo table
        if i == 0:
            exam_object = ExamInfo(year_of_calendar=2015, month_of_year=months[3],
year_of_pursue_roman='II',
                                                 semester_roman='II', year_of_pursue=2,
semester=2, supple=False, student=stud)
            exam_object.save()
            i=1

        # hall2 = hall1

        sub_code = str(sheet.cell(row, 2).value).strip()
        sub = Subject.objects.get(subject_code=sub_code)
        # Modify the below code accordingly

        exam_object = ExamInfo.objects.get(student=stud, year_of_pursue=2,
semester=2)
        exam_object.total = str(int(exam_object.total) + total_marks)
        exam_object.save()
        result = Result(subject=sub, internal_marks=int_marks,
external_marks=ext_marks, results=res,
                            credits=credits, examinfo=exam_object,
total=str(total_marks))
        result.save()
    except:
        print sys.exc_info()

sheet = book.sheet_by_name("Sheet5")
for row in range(sheet.nrows):
    sub_code = str(sheet.cell(row, 2).value).strip()
    sub_name = str(sheet.cell(row, 3).value).strip()
    sub = Subject.objects.filter(subject_code=sub_code)
    if sub.count() == 0:
        sub = Subject(subject_code=sub_code, name=sub_name)
        sub.save()

# months = ['January', 'February', 'March', 'April', 'May', 'June', 'July',
'August', 'September', 'October',
#                'November', 'December']
i=0
# hall2 = ""
for row in range(sheet.nrows):
    try:
        int_marks = str(int(sheet.cell(row, 4).value))
    except:
        int_marks = str(sheet.cell(row, 4).value)
    try:
        ext_marks = str(int(sheet.cell(row, 5).value))
    except:
        ext_marks = str(sheet.cell(row, 5).value)
    total_marks = 0
    try:
        total_marks += int(int_marks)
    except:
        print "buddy was absent for internal"
    try:
        total_marks += int(ext_marks)
    except:
        print "Buddy was absent for external"
    res = str(sheet.cell(row, 7).value).strip()
    credits = int(sheet.cell(row, 8).value)
    try:
        hall_ticket = str(sheet.cell(row, 0).value).strip()
        # hall1 = hall_ticket

        print hall_ticket, " in results table"
        stud = Student.objects.get(hall_ticket=hall_ticket)

        # if hall1 != hall2:  # Populating the ExamInfo table
        if i == 0:
            exam_object = ExamInfo(year_of_calendar=2015, month_of_year=months[8],
year_of_pursue_roman='III',
                                         semester_roman='I', year_of_pursue=3,
semester=1, supple=False, student=stud)
                exam_object.save()
                i=1
        # hall2 = hall1

        sub_code = str(sheet.cell(row, 2).value).strip()
        sub = Subject.objects.get(subject_code=sub_code)
        # Modify the below code accordingly
        exam_object = ExamInfo.objects.get(student=stud, year_of_pursue=3,
semester=1)
        exam_object.total = str(int(exam_object.total) + total_marks)
        exam_object.save()
        result = Result(subject=sub, internal_marks=int_marks,
external_marks=ext_marks, results=res,
                            credits=credits, examinfo=exam_object,
total=str(total_marks))
        result.save()
    except:
        print sys.exc_info()

sheet = book.sheet_by_name("Sheet6")
for row in range(sheet.nrows):
    sub_code = str(sheet.cell(row, 2).value).strip()
    sub_name = str(sheet.cell(row, 3).value).strip()
    sub = Subject.objects.filter(subject_code=sub_code)
    if sub.count() == 0:
        sub = Subject(subject_code=sub_code, name=sub_name)
        sub.save()

# months = ['January', 'February', 'March', 'April', 'May', 'June', 'July',
'August', 'September', 'October',
#                'November', 'December']
# hall2 = ""
i=0
for row in range(sheet.nrows):
    try:
        int_marks = str(int(sheet.cell(row, 4).value))
    except:
        int_marks = str(sheet.cell(row, 4).value)
    try:
        ext_marks = str(int(sheet.cell(row, 5).value))
    except:
        ext_marks = str(sheet.cell(row, 5).value)
    total_marks = 0
    try:
        total_marks += int(int_marks)
    except:
        print "buddy was absent for internal"
    try:
        total_marks += int(ext_marks)
    except:
        print "Buddy was absent for external"
    res = str(sheet.cell(row, 7).value).strip()
    credits = int(sheet.cell(row, 8).value)
    try:
        hall_ticket = str(sheet.cell(row, 0).value).strip()
        # hall1 = hall_ticket

        print hall_ticket, " in results table"
        stud = Student.objects.get(hall_ticket=hall_ticket)

        # if hall1 != hall2:  # Populating the ExamInfo table
        if i == 0:
            exam_object = ExamInfo(year_of_calendar=2016, month_of_year=months[10],
year_of_pursue_roman='IV',
                                    semester_roman='I', year_of_pursue=4,
semester=1, supple=False, student=stud)
            exam_object.save()
            i=1

        # hall2 = hall1

        sub_code = str(sheet.cell(row, 2).value).strip()
        sub = Subject.objects.get(subject_code=sub_code)
        # Modify the below code accordingly
        exam_object = ExamInfo.objects.get(student=stud, year_of_pursue=4,
semester=1)
        exam_object.total = str(int(exam_object.total) + total_marks)
        exam_object.save()
        result = Result(subject=sub, internal_marks=int_marks,
external_marks=ext_marks, results=res,
                            credits=credits, examinfo=exam_object,
total=str(total_marks))
        result.save()
    except:
        print sys.exc_info()
```

scripts.py

This script is used for populating database with achievements in a semester.

```
from InfoSystem.models import *


# semester wise
ei = ExamInfo.objects.all().filter(supple=False)

for i in range(1, 5):
    for j in range(1, 3):
        sem = ei.filter(year_of_pursue=i, semester=j)
        results_total = []
        for p in sem:
            results_total.append(int(p.total))
            # results_total[p] = int(p.total)
            #results_total.append({p.student: int(p.total)})
        if results_total != []:
            results_total= list(set(results_total))
            results_total.sort(reverse=True)
            # results_total = sorted(results_total.items(), key= lambda x:x[1], 
reverse=True)
            temp = sem.filter(total=results_total[0])
            for lmn in temp:
                achievement_in_semester = AchievementInASemester(rank=1, 
student=lmn.student, examinfo=lmn)
                achievement_in_semester.save()
                print achievement_in_semester, lmn.total

            try:
                temp = sem.filter(total=results_total[1])
                if temp != []:
                    for lmn in temp:
                        achievement_in_semester = AchievementInASemester(rank=2, 
student=lmn.student, examinfo=lmn)
                        achievement_in_semester.save()
                        print achievement_in_semester, lmn.total

            except:
                print "not there"
```

scripts2.py

This script is used for populating the database with achievements in every subject in a semester.

```
from InfoSystem.models import Subject, Result, AchievementInASubject

subjects = Subject.objects.all()

for sub in subjects:
    results = Result.objects.filter(subject = sub)
    results = results.filter(examinfo__supple=False)
    sub_totals = []
    for res in results:
        sub_totals.append(int(res.total))
    sub_totals= list(set(sub_totals))
    sub_totals.sort(reverse=True)
    try:
        max_res = results.filter(total=sub_totals[0])
        if len(max_res) > 0:
            for lmn in max_res:
                magic = lmn.examinfo
                achievement = AchievementInASubject(rank=1, student=magic.student,
result=lmn, year_of_pursue=magic.year_of_pursue,

year_of_pursue_roman=magic.year_of_pursue_roman, semester=magic.semester,

semester_roman=magic.semester_roman)
                achievement.save()
                print achievement.student.name, achievement.result.subject,
achievement.result.total
                # print lmn.total, lmn.subject, lmn.examinfo.student
    except:
        print "get lost"

    try:
        max_res = results.filter(total=sub_totals[1])
        if len(max_res) > 0:
            for lmn in max_res:
                magic = lmn.examinfo
                achievement = AchievementInASubject(rank=2, student=magic.student,
result=lmn,

year_of_pursue=magic.year_of_pursue,

year_of_pursue_roman=magic.year_of_pursue_roman,
                                                    semester=magic.semester,

semester_roman=magic.semester_roman)
                achievement.save()
                print achievement.student.name, achievement.result.subject,
achievement.result.total
                # print lmn.total, lmn.subject, lmn.examinfo.student
    except:
        print "lost"
```

urls2.py

```
urlpatterns = [
    url(r'^admin/', admin.site.urls),
    url(r'^', include('InfoSystem.urls')),
    url(r'auth/', include('djoser.urls.authtoken', namespace='djoser'))

]
```

login.html

This is a django template which is used for displaying login page for user**.**

```
<html>
{% load static %}
<head>
    <title>User Login</title>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
          integrity="sha384-
BVYiiSIFeK1dGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+PmSTsz/K68vbdEjh4u"
crossorigin="anonymous">
</head>
```

```
<body>
<form class="form-horizontal" style="width: 40%;margin-left: auto;margin-right:
auto" method="post" action=".">
    {% csrf_token %}
    <fieldset>
        <legend style="font-weight: bolder;margin-bottom: 20px; padding:
20px">Login</legend>
        <div class="col-lg-10 col-lg-offset-2">
            {% if error_message %}
                <p><strong>{{ error_message }}</strong></p>
            {% endif %}
        </div>
        <div class="col-sm-offset-2 col-sm-10">
                <span class="text-danger small">
                    {% if form.errors %}
                        {{  form.non_field_errors }}
                    {% endif %}
                </span>
        </div>

        <div class="form-group">
            <label for="id_username" class="col-lg-2 control-label">Username:
</label>
            <div class="col-lg-10">
                <input type="text" class="form-control" id="id_username"
name="username" placeholder="User Name"
                       required>
            </div>
        </div>
        <div class="form-group">
            <label for="id_password" class="col-lg-2 control-label">Password:
</label>
            <div class="col-lg-10">
                <input type="password" class="form-control" id="id_password"
name="password" placeholder="Password"
                       required>
            </div>
        </div>

        <div class="col-lg-10 col-lg-offset-2">
            <input type="submit" class="btn btn-primary" value="Login"/>
            <div class="panel-footer">
                Don't have an account? <a href="{% url 'register' %}">Click
here</a> to register.
            </div>
        </div>
    </fieldset>
</form>
</body>
</html>
```

Register.html

This is a django template which is used for displaying register page for user**.**

```
<!DOCTYPE html>
{% load static %}
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Sign Up</title>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet"
```

```
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
          integrity="sha384-
BVYiiSIFeK1dGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+PmSTsz/K68vbdEjh4u"
crossorigin="anonymous">
    <style type="text/css">
        body{
            background-image: url("{% static 'InfoSystem/yoyo.jpeg' %}");
            background-size: cover;
            background-position: center;
            background-attachment: fixed;
        }

    </style>
</head>
<body>
{#<h3>Register an Account</h3>#}
<form class="form-horizontal" style="width: 50%;margin-left: auto;margin-right:
auto" method="post">
    {% csrf_token %}
  <fieldset>
    <legend style="font-weight: bolder;margin-bottom: 20px; padding: 20px;text-
align: center">Sign Up</legend>
{#      {% for field in form %}#}
{#        <div class="form-group">#}
{#            <div class="col-sm-offset-2 col-sm-10">#}
{#                <span class="text-danger small">{{ field.errors }}</span>#}
{#                {{ form.username.name }}#}
{#            </div>#}
{#            <label class="col-lg-2 control-label" >{{ field.label_tag
}}</label>#}
{#            <div class="col-lg-10">{{ field }}</div>#}
{#        </div>#}
{#      {% endfor %}#}
        <div class="form-group">
            <div class="col-sm-offset-2 col-sm-10">
                <span class="text-danger small">{{ form.username.errors }}</span>
            </div>
          <label for="id_username" class="col-lg-4 control-label">Username:
</label>
          <div class="col-lg-6">
            <input type="text" class="form-control" id="id_username"
name="username" placeholder="User Name" required>
          </div>
        </div>
        <div class="form-group">
            <div class="col-sm-offset-2 col-sm-10">
                <span class="text-danger small">{{ form.email.errors }}</span>
            </div>
        <label for="id_email" class="col-lg-4 control-label">Email: </label>
          <div class="col-lg-6">
            <input type="email" class="form-control" id="id_email"  name="email"
placeholder="E-mail" required>
{#                {{ form.email }}#}
          </div>
        </div>
        <div class="form-group">
            <div class="col-sm-offset-2 col-sm-10">
                <span class="text-danger small">{{ form.mobile.errors }}</span>
            </div>
          <label for="id_mobile" class="col-lg-4 control-label">Mobile: </label>
          <div class="col-lg-6">
            <input type="text" class="form-control" id="id_mobile" name="mobile"
placeholder="Mobile number" required>
{#                {{ form.mobile }}#}
          </div>
        </div>
        <div class="form-group">
            <div class="col-sm-offset-2 col-sm-10">
```

```
                <span class="text-danger small">{{ form.password1.errors }}</span>
            </div>
          <label for="id_password1" class="col-lg-4 control-label">Enter Password:
</label>
          <div class="col-lg-6">
            <input type="password" class="form-control" id="id_password1"
name="password1" placeholder="Password" required>
          </div>
        </div>
        <div class="form-group">
            <div class="col-sm-offset-2 col-sm-10">
                <span class="text-danger small">{{ form.password2.errors }}</span>
            </div>
          <label for="id_password2" class="col-lg-4 control-label">Enter Password
again: </label>
          <div class="col-lg-6">
            <input type="password" class="form-control" id="id_password2"
name="password2" placeholder="Password" required>
          </div>
        </div>
        <div class="form-group">
          <label for="id_isstudent" class="col-lg-4 control-label">Student?
</label>
          <div class="col-lg-6">
            <input style="vertical-align: middle" type="checkbox"
id="id_isstudent" name="isstudent">
          </div>
        </div>


      <div class="col-sm-11" align="center">
        <button type="reset" class="btn btn-danger">Cancel</button>
        <input type="submit" class="btn btn-success" value="Submit"/>
      </div>

    {% if error_message %}
        <p><strong>{{ error_message }}</strong></p>
    {% endif %}
  <div align="center">
      <br><br>Already have an account? <a href="{% url 'login' %}">Click here</a>
to log in.
  </div>

  </fieldset>
</form>
</body>
</html>
```

Results_parent.py

This is a django template which is used for displaying results page for parent**.**

```
<!DOCTYPE html>
{% load static %}
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
{#    <link rel="stylesheet" href="{% static 'InfoSystem/bootstrap.css' %}">#}
    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
          integrity="sha384-
BVYiiSIFeK1dGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+PmSTsz/K68vbdEjh4u"
```

```
crossorigin="anonymous">
    <title>Student Results Page</title>


    <style type="text/css">
        .my_container {
            position: absolute;
{#              z-index: 1;#}
            visibility: visible;
        }

        .my_yet_other_container {
            position: absolute;
{#              z-index: 1;#}
            visibility: hidden;
        }

        .my_outer_container {
            position: absolute;
{#              z-index: 1;#}
            visibility: visible;
        }

        .my_other_outer_container {
            position: absolute;
{#              z-index: 1;#}
            visibility: hidden;
        }
    </style>
</head>
<body>
<nav class="navbar navbar-default" style="background-color: #1f7e9a">
    <div class="container-fluid">
        <div class="navbar-header">
            <a class="navbar-brand" href="#" style="color: white">Welcome {{
user.username }}</a>
        </div>
        <ul class="nav navbar-nav" style="background-color: #1f7e9a">
            <li class="active"><a style="color: white;background-color: #1f7e9a;"
href="https://cvr.ac.in">Home</a></li>
        </ul>
        <ul class="nav navbar-nav navbar-right">
            <li><a style="color: white;background-color: #1f7e9a;" href="{% url
'logout' %}"><span style="color: white;background-color: #1f7e9a;" class="glyphicon
glyphicon-log-out"></span> Logout</a></li>
        </ul>
    </div>
</nav>

<div style="position: relative">
    <div class="container">
        <h4 style="float: left">Father Name: {{ parent.father_name }}</h4>
        <h4 style="float: right">Mobile No: {{ parent.mobile }}</h4><br>
    </div>
    <div class="container">
        <h4>Mother Name: {{ parent.mother_name }}</h4><br>
    </div>
    <div>
        <div class="container" id="buttons">
            {% for student in students %}
                {#                {% for i in num_buttons %}#}
                <button class="btn btn-success"
                        id="button{{ forloop.counter }}"
                        onclick="onClickOuter({{ forloop.counter }})">
                    {{ student.name }}
                </button>
                {#                {% endfor %}#}
                {% endfor %}
        </div>
        <div class="container" style="position: relative">
            {% for student, examinfo in my_dict.items %}
                <div class="
                    {% if forloop.counter0 == 0 %}
                        my_outer_container
                        {% else %}
                            my_other_outer_container
                        {% endif %}
                         "
                        id="div{{ forloop.counter }}">
                    <div>
                        <div class="container">
                            <h4 style="float: left">Name: {{ student.name }}</h4>
                            <h4 style="float: right">Roll No: {{
student.hall_ticket }}</h4><br>
                        </div>
                        <div>
                            <div class="container" id="buttons">
                                {% for sem, ei in examinfo.items %}
                                    <button class="btn btn-success"
                                            id="button{{ forloop.parentloop.counter
}}{{ ei.0.year_of_pursue }}{{ ei.0.semester }}"
                                            onclick="onClick({{
forloop.parentloop.counter }}, {{ ei.0.year_of_pursue }}{{ ei.0.semester }})">
                                        {{ ei.0.year_of_pursue_roman }} - {{
ei.0.semester_roman }}
                                    </button>
                                {% endfor %}
                            </div>
                            {% for key, ei in examinfo.items %}
                                <div class="container
                                    {% if ei.0.year_of_pursue == 1 and
ei.0.semester == 1 and forloop.parentloop.counter0 == 0 %}
                                        my_container
                                    {% else %}
                                        my_yet_other_container
                                    {% endif %}
                                      "
                                     id="div{{ forloop.parentloop.counter }}{{
ei.0.year_of_pursue }}{{ ei.0.semester }}">

                                    {% for e in ei %}
                                        <h5 style="float: left">Examination held
during {{ e.month_of_year }}

                                            / {{ e.year_of_calendar }}</h5>
                                        <h5 style="float: right"> B.Tech {{
e.year_of_pursue_roman }}

                                            Year {{ e.semester_roman }}
                                            Semester
                                             {% if e.supple is False %}
                                                 Main
                                                 {% else %}
                                                 Supple
                                             {% endif %}
                                        </h5><br/><br>
                                        <table class="table table-hover table-
responsive table-striped">

                                            <tr>
                                                <th>#</th>
                                                <th>Subject</th>
                                                <th>Internal Exams</th>
                                                <th>External Exams</th>
                                                <th>Total</th>
                                                <th>Result</th>
                                                <th>Credits</th>
                                            </tr>
                                            {% for result in e.result.all %}
                                                <tr>
                                                    <td>{{ forloop.counter }}</td>
                                                    <td>{{ result.subject }}</td>
                                                    <td>{{ result.internal_marks
}}</td>
                                                    <td>{{ result.external_marks
}}</td>
                                                    <td>{{
result.internal_marks|add:result.external_marks }}</td>
                                                    <td>{{ result.results }}</td>
                                                    <td>{{ result.credits }}</td>
                                                </tr>
                                            {% endfor %}
                                        </table>
                                        <h4>Total Marks: {{ e.total }}</h4>

                                        {% if e.supple == False and
list_of_sem|get_item:student|get_item:key %}
                                            <h3>Achievements: </h3>
                                            {% for ach in
list_of_sem|get_item:student|get_item:key %}
                                                <h4>Your ward {{ student.name }}
has secured rank {{ ach.rank }} in this Semester<br>
                                            {% endfor %}
                                            <h4>Achievements desk</h4>
                                            <table class="table table-hover table-
responsive table-striped">

                                                <tr>
                                                    <th>Rank</th>
                                                    <th>Subject</th>
                                                </tr>
                                                {% for ach in
list_of_subs|get_item:student|get_item:key %}
                                                    <tr>
                                                        <td>{{ ach.rank }}</td>
                                                        <td>{{ ach.result.subject
}}</td>
                                                    </tr>
                                                {% endfor %}
                                            </table>
                                        {% endif %}
                                    {% endfor %}

                                </div>
                            {% endfor %}
                        </div>
                    </div>
                </div>
            {% endfor %}
        </div>
    </div>
</div>
<script type="text/javascript" src="{% static "InfoSystem/results_parent.js"
%}"></script>
</body>
</html>

{% load static %}
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
          integrity="sha384-
BVYiiSIFeK1dGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+PmSTsz/K68vbdEjh4u"
crossorigin="anonymous">
```

```
    <title>Student Results Page</title>


    <style type="text/css">
        .my_container {
            position: absolute;
            z-index: 1;
            visibility: visible;
        }

        .my_yet_other_container {
            position: absolute;
            z-index: 1;
            visibility: hidden;
        }
    </style>
</head>
<body>
<nav class="navbar navbar-default" style="background-color: #1f7e9a">
    <div class="container-fluid">
        <div class="navbar-header">
            <a class="navbar-brand" style="color: white" href="#">Welcome {{
user.username }}</a>
        </div>
        <ul class="nav navbar-nav">
            <li class="active"><a style="color: white;background-color: #1f7e9a;"
href="https://cvr.ac.in">Home</a></li>
        </ul>
        <ul class="nav navbar-nav navbar-right">
            <li><a style="color: white;background-color: #1f7e9a;" href="{% url
'logout' %}">
                <span class="glyphicon glyphicon-log-out" style="color:
white;background-color: #1f7e9a;"></span> Logout</a></li>
        </ul>
    </div>
</nav>
<div style="position: relative">
    <div class="container">
        <h4 style="float: left">Name: {{ students.0.name }}</h4>
        <h4 style="float: right">Roll No: {{ students.0.hall_ticket }} </h4><br>
    </div>

    <div class="container">
        <div class="container" id="buttons">
            {% for key, value in stud_res_dict %}
                {#                  {% for i in num_buttons %}#}
                <button class="btn btn-success"
                        id="button{{ value.0.year_of_pursue }}{{ value.0.semester
}}"
                        onclick="onClick({{ value.0.year_of_pursue }}{{
value.0.semester }})">
                    {{ value.0.year_of_pursue_roman }} - {{ value.0.semester_roman
}}
                </button>
                {#                  {% endfor %}#}
            {% endfor %}
        </div>
        {% for key, examinfo in stud_res_dict %}
            <div class="container
                {% if examinfo.0.year_of_pursue == 1 and examinfo.0.semester == 1
%}
                  my_container
                  {% else %}
                  my_yet_other_container
                {% endif %}
                 "
                id="div{{ examinfo.0.year_of_pursue }}{{ examinfo.0.semester }}">
                {% for ei in examinfo %}
```

```
{#                    {{ sem_dict.keys }}#}
                    <h5 style="float: left">Examination held during {{
ei.month_of_year }}
                        / {{ ei.year_of_calendar }}</h5>
                    <h5 style="float: right"> B.Tech {{ ei.year_of_pursue_roman }}
                        Year {{ ei.semester_roman }}
                        Semester
                        {% if ei.supple %}
                            Supple
                        {% else %}
                            Main
                        {% endif %}
                    </h5><br/><br>
                    <table class="table table-hover table-responsive table-
striped">
                        <tr>
                            <th>#</th>
                            <th>Subject</th>
                            <th>Internal Exams</th>
                            <th>External Exams</th>
                            <th>Total</th>
                            <th>Result</th>
                            <th>Credits</th>
                        </tr>

                        {% for result in ei.result.all %}
                            <tr>
                                <td>{{ forloop.counter }}</td>
                                <td>{{ result.subject }}</td>
                                <td>{{ result.internal_marks }}</td>
                                <td>{{ result.external_marks }}</td>
                                <td>{{ result.total }}</td>
                                <td>{{ result.results }}</td>
                                <td>{{ result.credits }}</td>
                            </tr>
                        {% endfor %}
                    </table>
                    <h4>Total Marks: {{ ei.total }}</h4>
                    {% if ei.supple == False and sem_dict %}
                        <h3>Achievements: </h3>

                            {% for ach in sem_dict|get_item:key %}
                                <h4>You have secured rank {{ ach.rank }} in this
Semester.</h4><br>
                            {% endfor %}
                            <h4>Achievements desk</h4>
                            <table class="table table-hover table-responsive table-
striped">
                            <tr>
                                <th>Rank</th>
                                <th>Subject</th>
                            </tr>
                            {% for ach in sub_dict|get_item:key %}
                                <tr>
                                    <td>{{ ach.rank }}</td>
                                    <td>{{ ach.result.subject }}</td>
                                </tr>
                            {% endfor %}
                        </table>
                    {% endif %}
                {% endfor %}
            </div>
        {% endfor %}
    </div>
</div>
<script type="text/javascript" src="{% static "InfoSystem/results_student.js"
%}"></script>
```

```
</body>
</html>
```

Serializer.py

This file is used to convert python datatypes that can be easily rendered into JSON or XML.

```
from django.contrib.auth.models import User
from django.db import models
from rest_framework import serializers

from InfoSystem.models import Parent, Student, Result, Subject, CustomUser,
ExamInfo
from django.utils.translation import ugettext as _


class CustomUserSerializer(serializers.ModelSerializer):
    class Meta:
        model = CustomUser
        fields = ('is_student', )


class SubjectSerializer(serializers.ModelSerializer):
    class Meta:
        model = Subject
        fields = ('name', )


class ResultSerializer(serializers.ModelSerializer):
    subjects = SubjectSerializer(source='subject', read_only=True)
    class Meta:
        model = Result
        fields = ('subjects', 'internal_marks', 'external_marks', 'results',
'credits')


class ExamInfoSerializer(serializers.ModelSerializer):
    result = ResultSerializer(many=True)
    class Meta:
        model = ExamInfo
        fields = ('year_of_pursue', 'semester', 'month_of_year',
'year_of_calendar', 'supple', 'year_of_pursue_roman', 'semester_roman', 'result')


class StudentSerializer(serializers.ModelSerializer):
    examinfo = ExamInfoSerializer(many=True, read_only=True)
    class Meta:
        model = Student
        fields = ('name', 'email', 'hall_ticket', 'examinfo')


class UserRegisterSerializer(serializers.ModelSerializer):
    password = serializers.CharField(write_only=True,  style={'input_type':
'password'})
    class Meta:
        model = CustomUser
        fields = ['username', 'email', 'mobile', 'password', 'is_student']

    def create(self, validated_data):
        if validated_data['is_student'] is False:
            par = Parent.objects.get(mobile__exact=validated_data['mobile'])
            user = super(UserRegisterSerializer, self).create(validated_data)
            par.user = user
            par.email = validated_data['email']
            par.is_registered=True
            par.save()
            user.set_password(validated_data['password'])
            user.save()
            return user
        else:
            stud = Student.objects.get(mobile__exact=validated_data['mobile'])
            user = super(UserRegisterSerializer, self).create(validated_data)
            stud.user = user
            stud.is_registered = True
            stud.save()
            user.set_password(validated_data['password'])
            user.save()
            return user

    def validate(self, data):
        if data['is_student'] is False:
            try:
                par = Parent.objects.get(mobile__exact=data['mobile'])
            except:
                raise serializers.ValidationError("Invalid mobile number.")
            if par.is_registered:
                raise serializers.ValidationError("You are already registered with
the given mobile number.")
        else:
            try:
                stud = Student.objects.get(mobile__exact=data['mobile'])
            except:
                raise serializers.ValidationError("Invalid mobile number.")
            if stud.is_registered:
                raise serializers.ValidationError("You are already registered with
the given mobile number.")
        return data
```

results_student.html

This is a django template which is used for displaying results page for student.

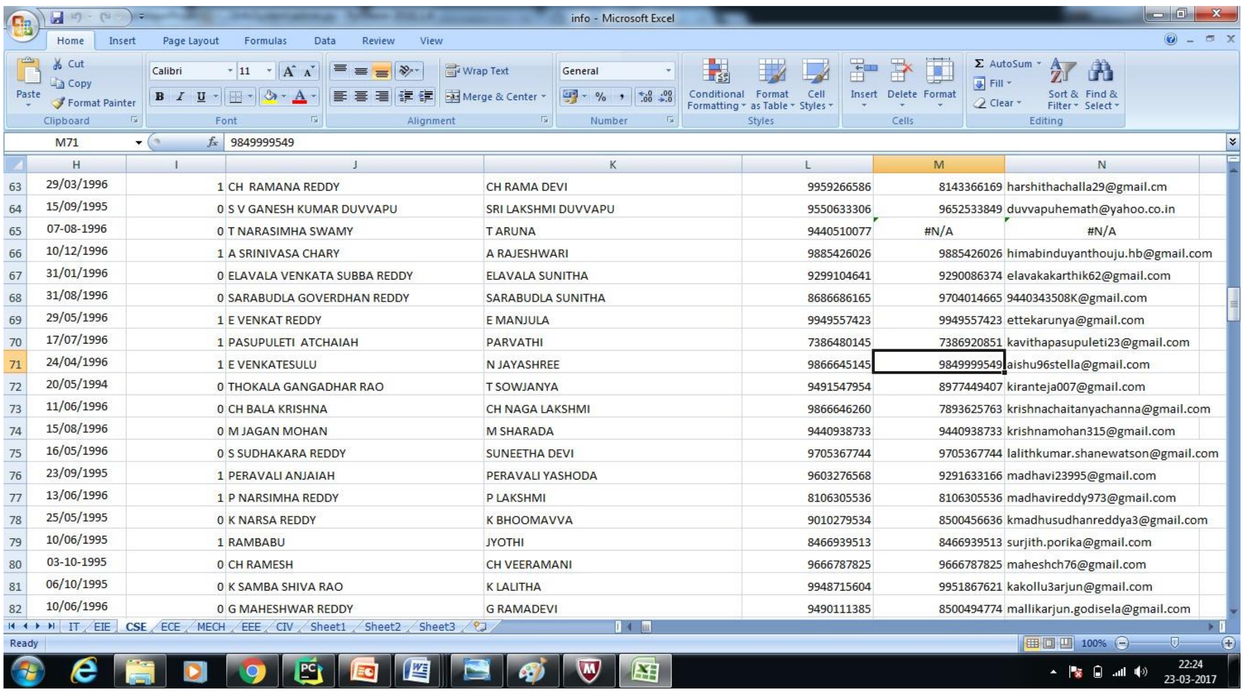
```
<!DOCTYPE html>
{% load static %}
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
          integrity="sha384-
BVYiiSIFeK1dGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+PmSTsz/K68vbdEjh4u"
crossorigin="anonymous">
    <title>Student Results Page</title>


    <style type="text/css">
        .my_container {
            position: absolute;
            z-index: 1;
            visibility: visible;
        }

        .my_yet_other_container {
            position: absolute;
            z-index: 1;
            visibility: hidden;
        }
    </style>
</head>
```

```
<body>
<nav class="navbar navbar-default" style="background-color: #1f7e9a">
    <div class="container-fluid">
        <div class="navbar-header">
            <a class="navbar-brand" style="color: white" href="#">Welcome {{
user.username }}</a>
        </div>
        <ul class="nav navbar-nav">
            <li class="active"><a style="color: white;background-color: #1f7e9a;"
href="https://cvr.ac.in">Home</a></li>
        </ul>
        <ul class="nav navbar-nav navbar-right">
            <li><a style="color: white;background-color: #1f7e9a;" href="{% url
'logout' %}">
                <span class="glyphicon glyphicon-log-out" style="color:
white;background-color: #1f7e9a;"></span> Logout</a></li>
        </ul>
    </div>
</nav>
<div style="position: relative">
    <div class="container">
        <h4 style="float: left">Name: {{ students.0.name }}</h4>
        <h4 style="float: right">Roll No: {{ students.0.hall_ticket }} </h4><br>
    </div>

    <div class="container">
        <div class="container" id="buttons">
            {% for key, value in stud_res_dict %}
                {#              {% for i in num_buttons %}#}
                <button class="btn btn-success"
                        id="button{{ value.0.year_of_pursue }}{{ value.0.semester
}}"
                        onclick="onClick({{ value.0.year_of_pursue }}{{
value.0.semester }})">
                    {{ value.0.year_of_pursue_roman }} - {{ value.0.semester_roman
}}
                </button>
                {#              {% endfor %}#}
            {% endfor %}
        </div>
        {% for key, examinfo in stud_res_dict %}
            <div class="container
                 {% if examinfo.0.year_of_pursue == 1 and examinfo.0.semester == 1
%}
                    my_container
                    {% else %}
                    my_yet_other_container
                 {% endif %}
                  "
                 id="div{{ examinfo.0.year_of_pursue }}{{ examinfo.0.semester }}">
                {% for ei in examinfo %}
{#                    {{ sem_dict.keys }}#}
                    <h5 style="float: left">Examination held during {{
ei.month_of_year }}
                        / {{ ei.year_of_calendar }}</h5>
                    <h5 style="float: right"> B.Tech {{ ei.year_of_pursue_roman }}
                        Year {{ ei.semester_roman }}
                        Semester
                        {% if ei.supple %}
                            Supple
                        {% else %}
                            Main
                        {% endif %}
                    </h5><br/><br>
                    <table class="table table-hover table-responsive table-
striped">
                        <tr>
                            <th>#</th>
```

```
                                    <th>Subject</th>
                                    <th>Internal Exams</th>
                                    <th>External Exams</th>
                                    <th>Total</th>
                                    <th>Result</th>
                                    <th>Credits</th>
                                </tr>

                                {% for result in ei.result.all %}
                                    <tr>
                                        <td>{{ forloop.counter }}</td>
                                        <td>{{ result.subject }}</td>
                                        <td>{{ result.internal_marks }}</td>
                                        <td>{{ result.external_marks }}</td>
                                        <td>{{ result.total }}</td>
                                        <td>{{ result.results }}</td>
                                        <td>{{ result.credits }}</td>
                                    </tr>
                                {% endfor %}
                            </table>
                            <h4>Total Marks: {{ ei.total }}</h4>
                            {% if ei.supple == False and sem_dict %}
                                <h3>Achievements: </h3>

                                    {% for ach in sem_dict|get_item:key %}
                                        <h4>You have secured rank {{ ach.rank }} in this
Semester.</h4><br>

                                    {% endfor %}
                                    <h4>Achievements desk</h4>
                                    <table class="table table-hover table-responsive table-
striped">

                                    <tr>
                                        <th>Rank</th>
                                        <th>Subject</th>
                                    </tr>
                                    {% for ach in sub_dict|get_item:key %}
                                        <tr>
                                            <td>{{ ach.rank }}</td>
                                            <td>{{ ach.result.subject }}</td>
                                        </tr>
                                    {% endfor %}
                                </table>
                            {% endif %}
                        {% endfor %}
                    </div>
                {% endfor %}
            </div>
</div>
<script type="text/javascript" src="{% static "InfoSystem/results_student.js"
%}"></script>
</body>
</html>
```

<u>4.1.Screenshots</u>

Raw data of parent details



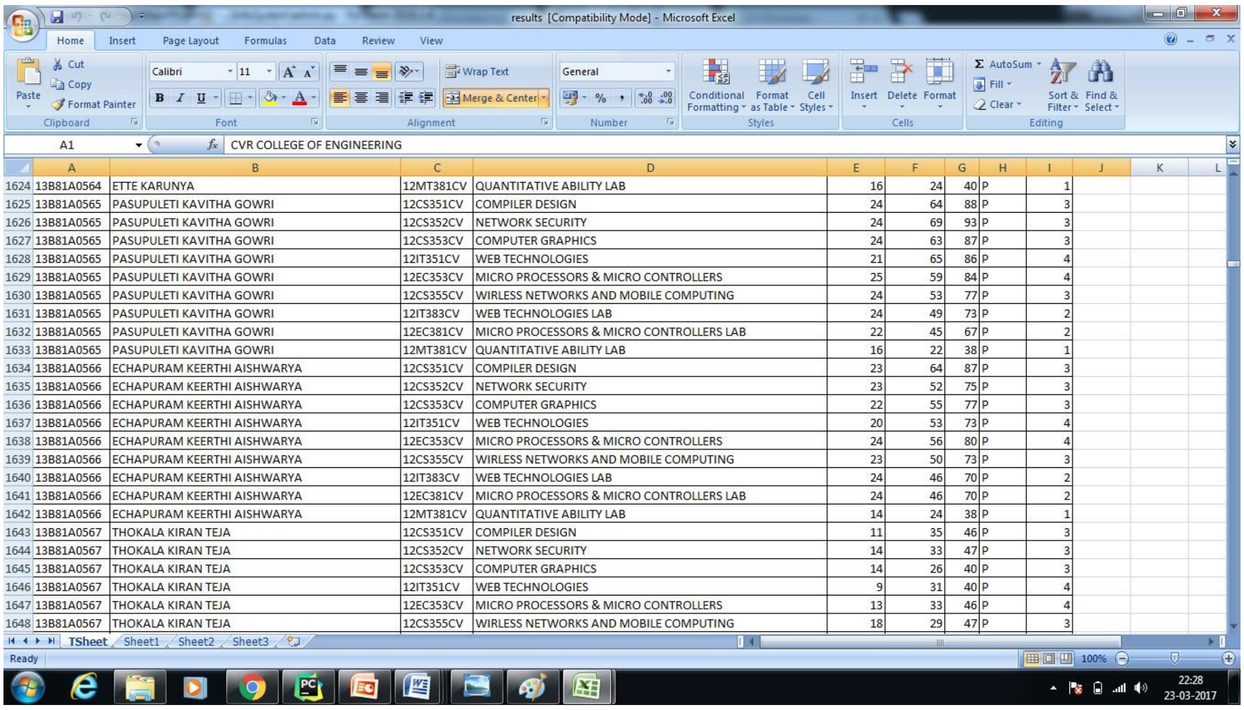
M71 9849999549

	H	I	J	K	L	M	N
63	29/03/1996	1	CH RAMANA REDDY	CH RAMA DEVI	9959266586	8143366169	harshithachalla29@gmail.cm
64	15/09/1995	0	S V GANESH KUMAR DUVVAPU	SRI LAKSHMI DUVVAPU	9550633306	9652533849	duvvapuhemath@yahoo.co.in
65	07-08-1996	0	T NARASIMHA SWAMY	T ARUNA	9440510077	#N/A	#N/A
66	10/12/1996	1	A SRINIVASA CHARY	A RAJESHWARI	9885426026	9885426026	himabinduyanthouju.hb@gmail.com
67	31/01/1996	0	ELAVALA VENKATA SUBBA REDDY	ELAVALA SUNITHA	9299104641	9290086374	elavakakarthik62@gmail.com
68	31/08/1996	0	SARABUDLA GOVERDHAN REDDY	SARABUDLA SUNITHA	8686686165	9704014665	9440343508K@gmail.com
69	29/05/1996	1	E VENKAT REDDY	E MANJULA	9949557423	9949557423	ettekarunya@gmail.com
70	17/07/1996	1	PASUPULETI ATCHAIAH	PARVATHI	7386480145	7386920851	kavithapasupuleti23@gmail.com
71	24/04/1996	1	E VENKATESULU	N JAYASHREE	9866645145	9849999549	aishu96stella@gmail.com
72	20/05/1994	0	THOKALA GANGADHAR RAO	T SOWJANYA	9491547954	8977449407	kiranteja007@gmail.com
73	11/06/1996	0	CH BALA KRISHNA	CH NAGA LAKSHMI	9866646260	7893625763	krishnachaitanyachanna@gmail.com
74	15/08/1996	0	M JAGAN MOHAN	M SHARADA	9440938733	9440938733	krishnamohan315@gmail.com
75	16/05/1996	0	S SUDHAKARA REDDY	SUNEETHA DEVI	9705367744	9705367744	lalithkumar.shanewatson@gmail.com
76	23/09/1995	1	PERAVALI ANJAIAH	PERAVALI YASHODA	9603276568	9291633166	madhavi23995@gmail.com
77	13/06/1996	1	P NARSIMHA REDDY	P LAKSHMI	8106305536	8106305536	madhavireddy973@gmail.com
78	25/05/1995	0	K NARSA REDDY	K BHOOMAVVA	9010279534	8500456636	kmadhusudhanreddya3@gmail.com
79	10/06/1995	1	RAMBABU	JYOTHI	8466939513	8466939513	surjith.porika@gmail.com
80	03-10-1995	0	CH RAMESH	CH VEERAMANI	9666787825	9666787825	maheshch76@gmail.com
81	06/10/1995	0	K SAMBA SHIVA RAO	K LALITHA	9948715604	9951867621	kakollu3arjun@gmail.com
82	10/06/1996	0	G MAHESHWAR REDDY	G RAMADEVI	9490111385	8500494774	mallikarjun.godisela@gmail.com

info - Microsoft Excel
Home Insert Page Layout Formulas Data Review View
IT EIE CSE ECE MECH EEE CIV Sheet1 Sheet2 Sheet3
Ready
22:24
23-03-2017

Data given by college will be in the form of excel sheet. This excel sheet contains details of parents and students like student name, parent name, year of completion, date of birth, phone number, email id, student mobile. Using python scripts we populate the database.

Raw data of students’ results

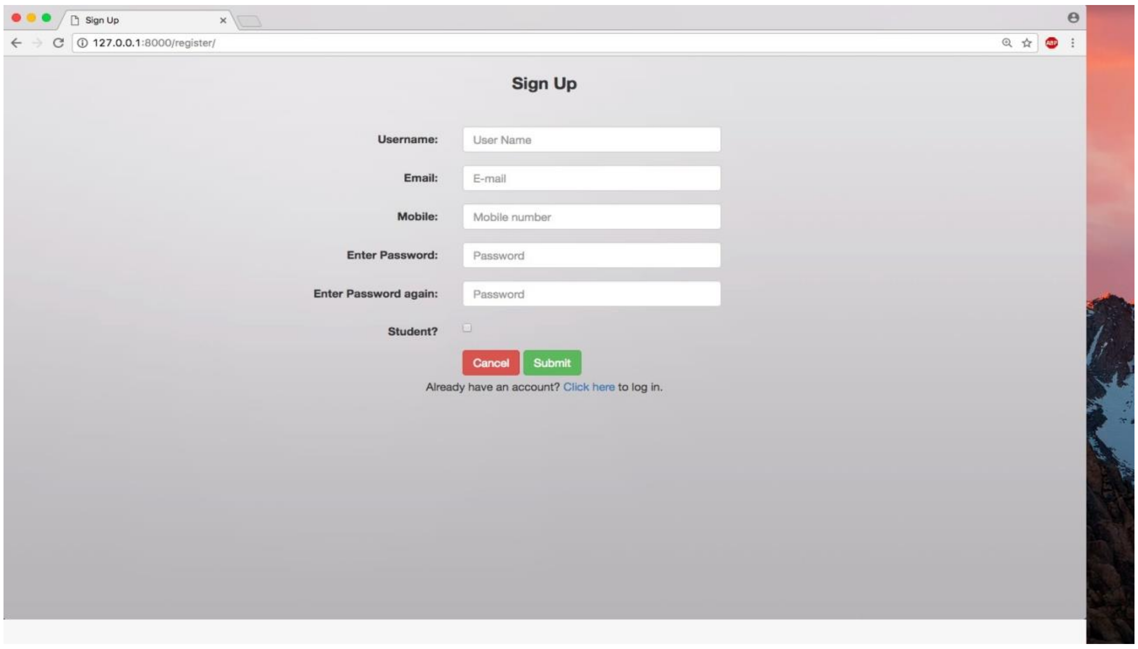


A1 CVR COLLEGE OF ENGINEERING

	A	B	C	D	E	F	G	H	I
1624	13B81A0564	ETTE KARUNYA	12MT381CV	QUANTITATIVE ABILITY LAB	16	24	40	P	1
1625	13B81A0565	PASUPULETI KAVITHA GOWRI	12CS351CV	COMPILER DESIGN	24	64	88	P	3
1626	13B81A0565	PASUPULETI KAVITHA GOWRI	12CS352CV	NETWORK SECURITY	24	69	93	P	3
1627	13B81A0565	PASUPULETI KAVITHA GOWRI	12CS353CV	COMPUTER GRAPHICS	24	63	87	P	3
1628	13B81A0565	PASUPULETI KAVITHA GOWRI	12IT351CV	WEB TECHNOLOGIES	21	65	86	P	4
1629	13B81A0565	PASUPULETI KAVITHA GOWRI	12EC353CV	MICRO PROCESSORS & MICRO CONTROLLERS	25	59	84	P	4
1630	13B81A0565	PASUPULETI KAVITHA GOWRI	12CS355CV	WIRLESS NETWORKS AND MOBILE COMPUTING	24	53	77	P	3
1631	13B81A0565	PASUPULETI KAVITHA GOWRI	12IT383CV	WEB TECHNOLOGIES LAB	24	49	73	P	2
1632	13B81A0565	PASUPULETI KAVITHA GOWRI	12EC381CV	MICRO PROCESSORS & MICRO CONTROLLERS LAB	22	45	67	P	2
1633	13B81A0565	PASUPULETI KAVITHA GOWRI	12MT381CV	QUANTITATIVE ABILITY LAB	16	22	38	P	1
1634	13B81A0566	ECHAPURAM KEERTHI AISHWARYA	12CS351CV	COMPILER DESIGN	23	64	87	P	3
1635	13B81A0566	ECHAPURAM KEERTHI AISHWARYA	12CS352CV	NETWORK SECURITY	23	52	75	P	3
1636	13B81A0566	ECHAPURAM KEERTHI AISHWARYA	12CS353CV	COMPUTER GRAPHICS	22	55	77	P	3
1637	13B81A0566	ECHAPURAM KEERTHI AISHWARYA	12IT351CV	WEB TECHNOLOGIES	20	53	73	P	4
1638	13B81A0566	ECHAPURAM KEERTHI AISHWARYA	12EC353CV	MICRO PROCESSORS & MICRO CONTROLLERS	24	56	80	P	4
1639	13B81A0566	ECHAPURAM KEERTHI AISHWARYA	12CS355CV	WIRLESS NETWORKS AND MOBILE COMPUTING	23	50	73	P	3
1640	13B81A0566	ECHAPURAM KEERTHI AISHWARYA	12IT383CV	WEB TECHNOLOGIES LAB	24	46	70	P	2
1641	13B81A0566	ECHAPURAM KEERTHI AISHWARYA	12EC381CV	MICRO PROCESSORS & MICRO CONTROLLERS LAB	24	46	70	P	2
1642	13B81A0566	ECHAPURAM KEERTHI AISHWARYA	12MT381CV	QUANTITATIVE ABILITY LAB	14	24	38	P	1
1643	13B81A0567	THOKALA KIRAN TEJA	12CS351CV	COMPILER DESIGN	11	35	46	P	3
1644	13B81A0567	THOKALA KIRAN TEJA	12CS352CV	NETWORK SECURITY	14	33	47	P	3
1645	13B81A0567	THOKALA KIRAN TEJA	12CS353CV	COMPUTER GRAPHICS	14	26	40	P	3
1646	13B81A0567	THOKALA KIRAN TEJA	12IT351CV	WEB TECHNOLOGIES	9	31	40	P	4
1647	13B81A0567	THOKALA KIRAN TEJA	12EC353CV	MICRO PROCESSORS & MICRO CONTROLLERS	13	33	46	P	4
1648	13B81A0567	THOKALA KIRAN TEJA	12CS355CV	WIRLESS NETWORKS AND MOBILE COMPUTING	18	29	47	P	3

Data given by college will be in the form of excel sheet. This sheet contains the results of students. The fields in this sheet are student roll number, student name, subject code, subject name, internal marks, external marks, pass/fail, credits. Using python scripts we populate the database.

User Registration Page


Sign Up

127.0.0.1:8000/register/

Sign Up

Username: User Name

Email: E-mail

Mobile: Mobile number

Enter Password: Password

Enter Password again: Password

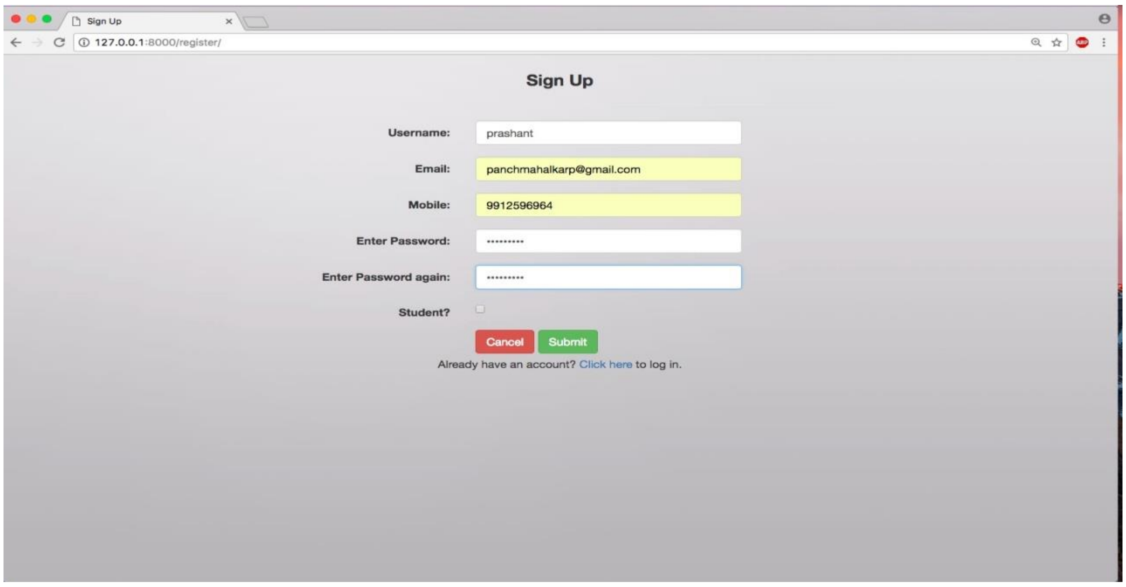
Student?

Cancel Submit

Already have an account? Click here to log in.

This is the User interface for a new user to register. In this page user has to give a unique username and mobile number which was given at the time of admission. If the user is a student he/she needs to check the checkbox.

Filled registration page


Sign Up

127.0.0.1:8000/register/

Sign Up

Username: prashant

Email: panchmahalkarp@gmail.com

Mobile: 9912596964

Enter Password: ·········

Enter Password again: ·········

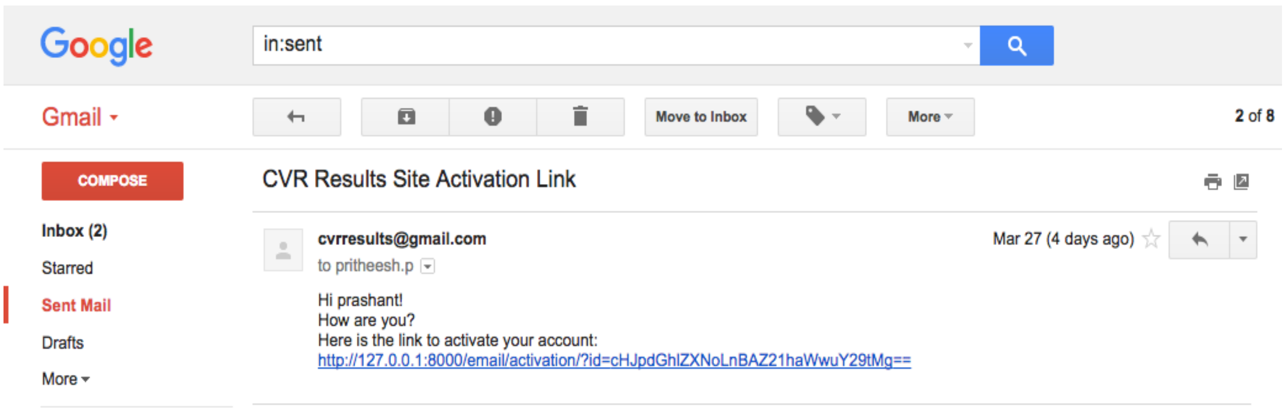
Student?

Cancel Submit

Already have an account? Click here to log in.

The above screenshot is a filled registration page which should contain the valid username, email, mobile and password which is more than 8 characters.

Activation link mail



CVR Results Site Activation Link

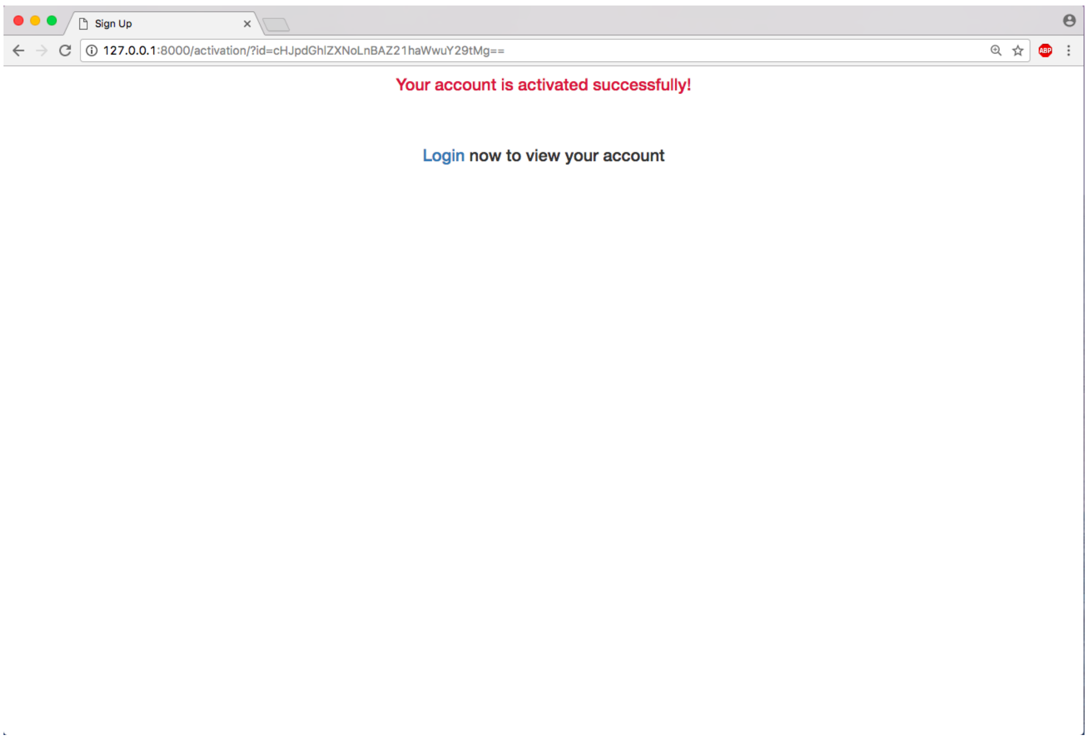
cvrresults@gmail.com
to pritheesh.p

Mar 27 (4 days ago)

Hi prashant!
How are you?
Here is the link to activate your account:
http://127.0.0.1:8000/email/activation/?id=cHJpdGhlZXNoLnBAZ21haWwuY29tMg==

After user fills the valid details in the registration form he will be receiving an email from CVR for activating his account. He needs to click on the activation link shown above.

Activation Successful

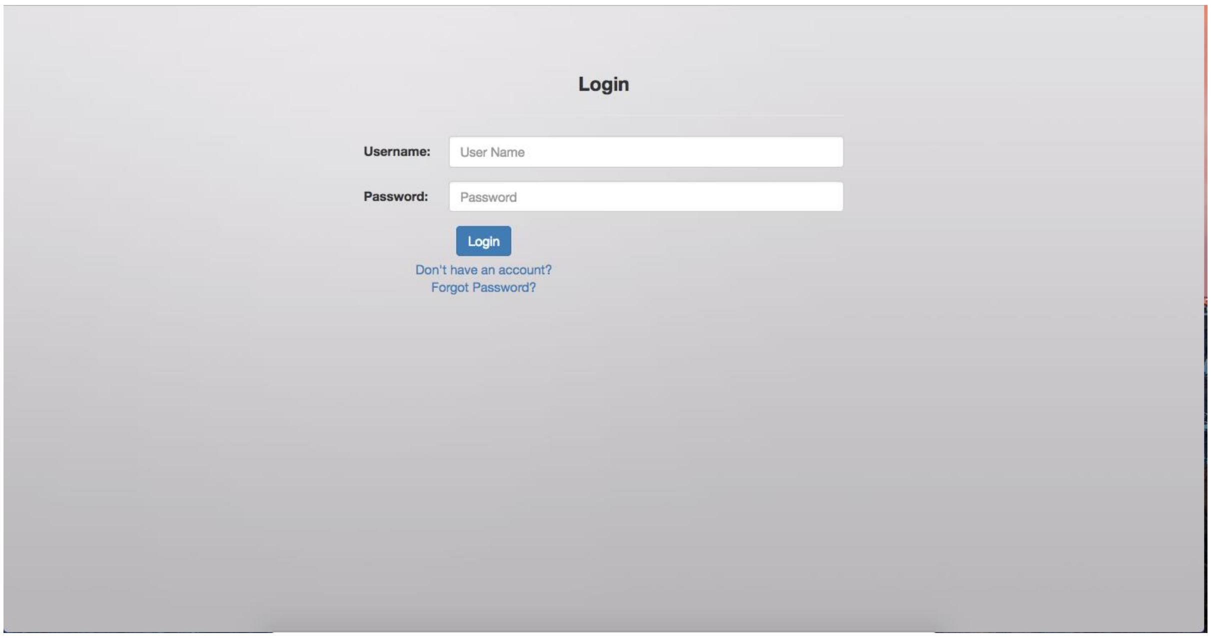

Sign Up
127.0.0.1:8000/activation/?id=cHJpdGhlZXNoLnBAZ21haWwuY29tMg==

Your account is activated successfully!

Login now to view your account

Once he clicks on the activation link he will be redirected to this page where it shows the message “Your account is activated successfully”. User needs to click on the login button.

Login page



Login

Username: User Name

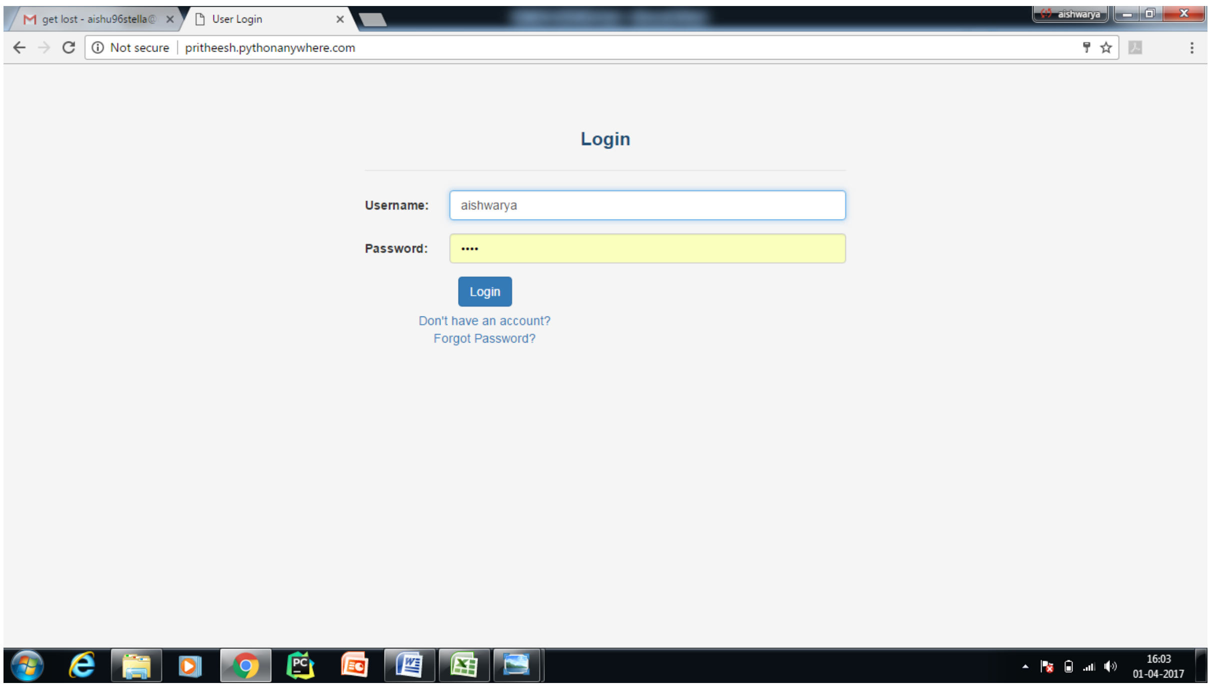
Password: Password

Login

Don't have an account?
Forgot Password?

Once the user clicks the login button he/she will be redirected to this page where he has to enter the valid username and password given at the time of registration.

Filled login page



get lost - aishu96stella@ | User Login

Not secure | pritheesh.pythonanywhere.com

Login

Username: aishwarya

Password: ••••

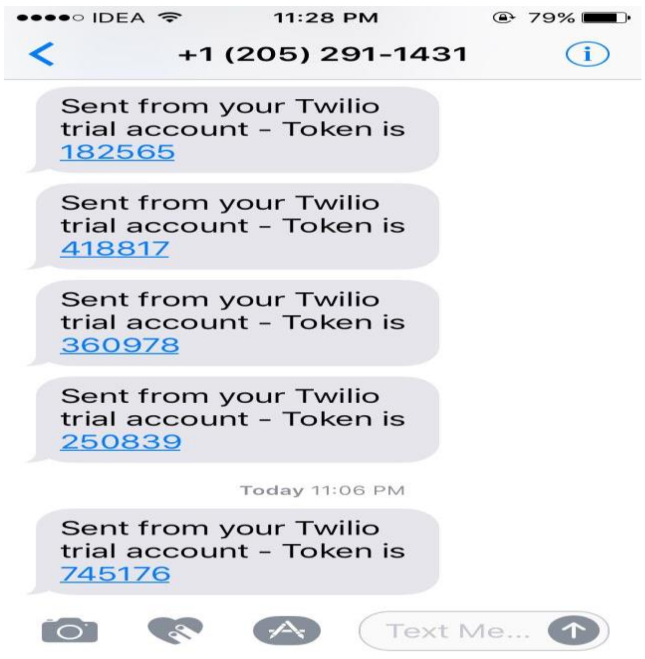
Login

Don't have an account?
Forgot Password?

16:03
01-04-2017

User signing with his credentials.

OTP sent as a message



Sent from your Twilio trial account - Token is 182565

Sent from your Twilio trial account - Token is 418817

Sent from your Twilio trial account - Token is 360978

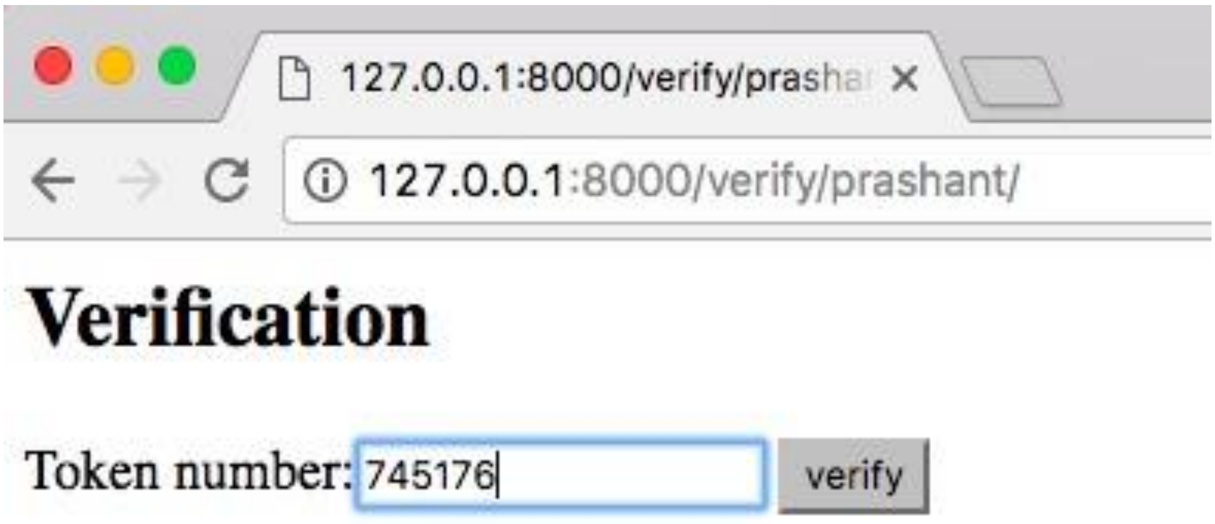
Sent from your Twilio trial account - Token is 250839

Today 11:06 PM

Sent from your Twilio trial account - Token is 745176

To activate your account you can even type the password sent to your registered mobile.

OTP(one time password) Verification

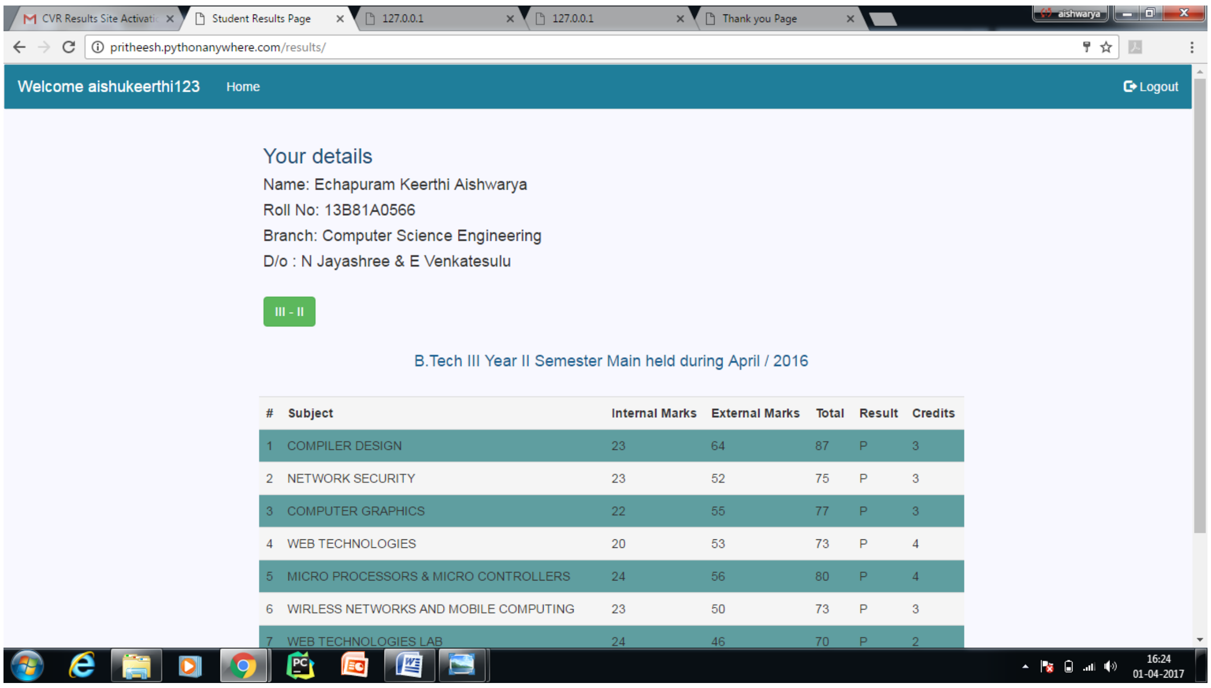


Verification

Token number: 745176

Once the user enters correct OTP, the user will be activated and will be redirected to login page.

Students Result Page



Welcome aishukeerthi123 Home Logout

Your details

Name: Echapuram Keerthi Aishwarya

Roll No: 13B81A0566

Branch: Computer Science Engineering

D/o : N Jayashree & E Venkatesulu

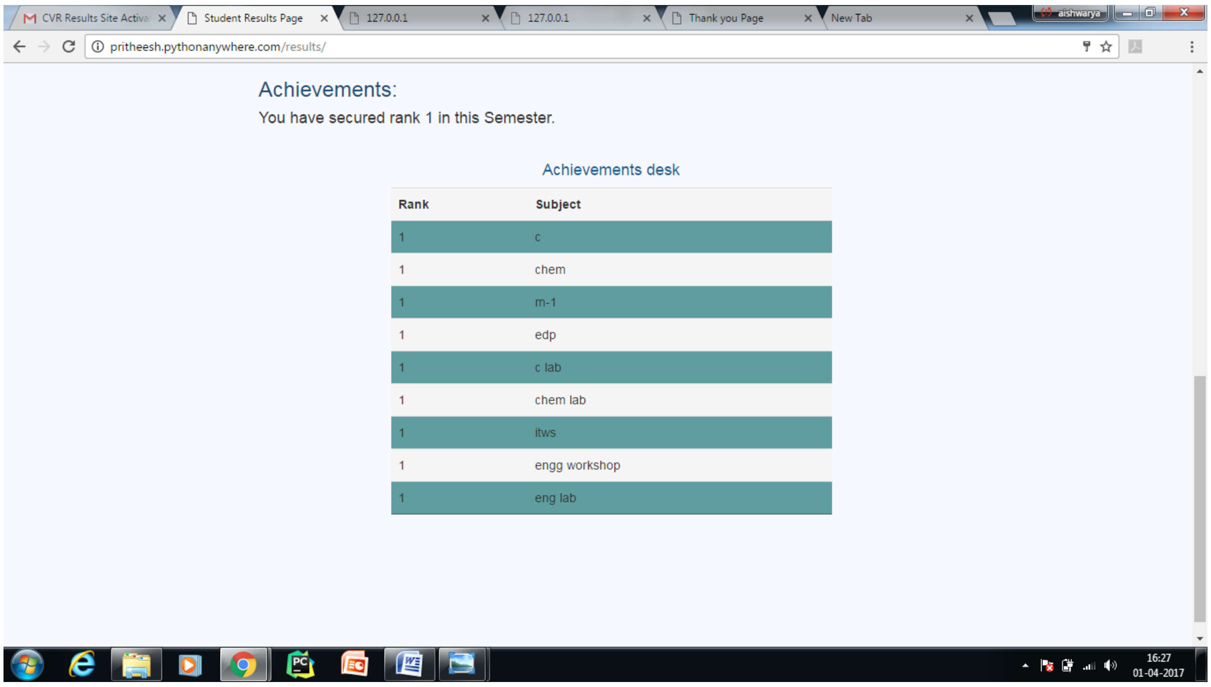
III - II

B.Tech III Year II Semester Main held during April / 2016

#	Subject	Internal Marks	External Marks	Total	Result	Credits
1	COMPILER DESIGN	23	64	87	P	3
2	NETWORK SECURITY	23	52	75	P	3
3	COMPUTER GRAPHICS	22	55	77	P	3
4	WEB TECHNOLOGIES	20	53	73	P	4
5	MICRO PROCESSORS & MICRO CONTROLLERS	24	56	80	P	4
6	WIRLESS NETWORKS AND MOBILE COMPUTING	23	50	73	P	3
7	WEB TECHNOLOGIES LAB	24	46	70	P	2

The above page will be displayed when student logs into the website to check his results.

Student Achievements



Achievements:

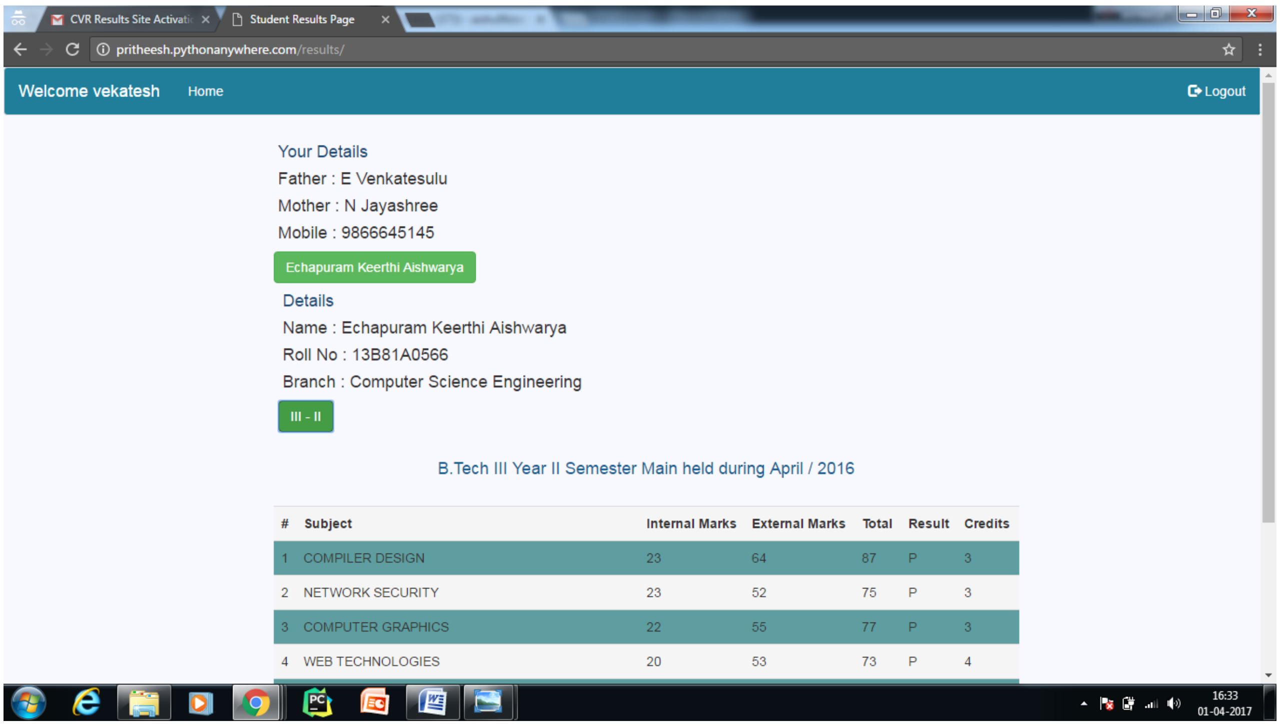
You have secured rank 1 in this Semester.

Achievements desk

Rank	Subject
1	c
1	chem
1	m-1
1	edp
1	c lab
1	chem lab
1	itws
1	engg workshop
1	eng lab

Students can view their achievements if they have secure highest marks in the semester or highest marks in a subject for that semester.

Results page for parents


Welcome vekatesh Home Logout

Your Details
Father : E Venkatesulu
Mother : N Jayashree
Mobile : 9866645145

Echapuram Keerthi Aishwarya

Details
Name : Echapuram Keerthi Aishwarya
Roll No : 13B81A0566
Branch : Computer Science Engineering

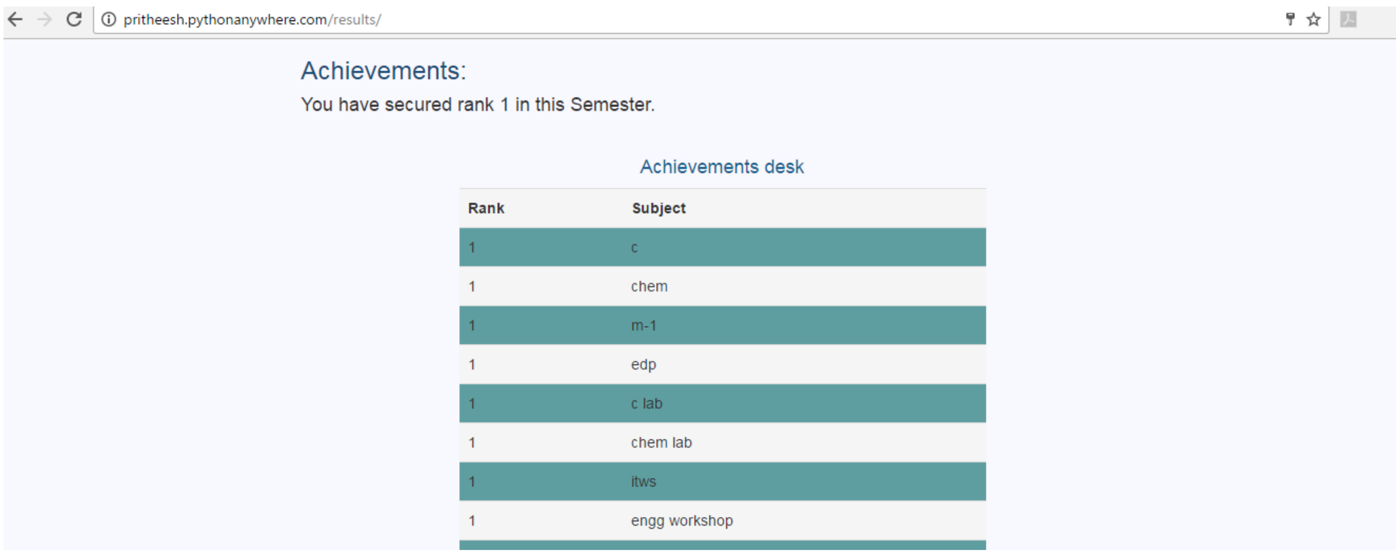
III - II

B.Tech III Year II Semester Main held during April / 2016

#	Subject	Internal Marks	External Marks	Total	Result	Credits
1	COMPILER DESIGN	23	64	87	P	3
2	NETWORK SECURITY	23	52	75	P	3
3	COMPUTER GRAPHICS	22	55	77	P	3
4	WEB TECHNOLOGIES	20	53	73	P	4

The above page is displayed when parent logs into the website to view their ward's result**.**

Achievements for Parents result page

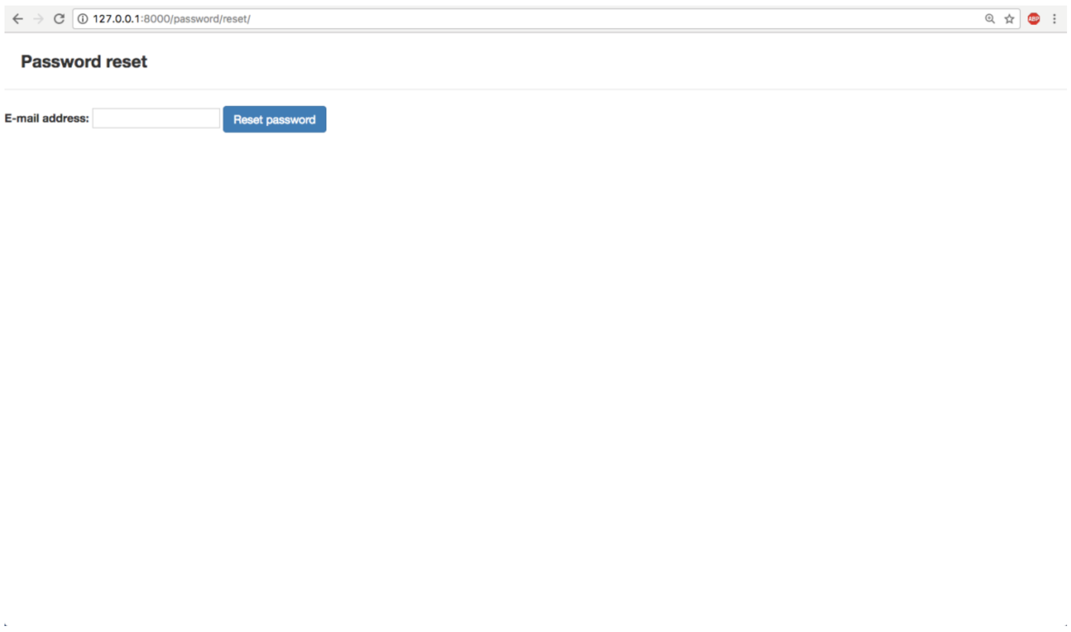

Achievements:
You have secured rank 1 in this Semester.

Achievements desk

Rank	Subject
1	c
1	chem
1	m-1
1	edp
1	c lab
1	chem lab
1	itws
1	engg workshop

Parents can view the achievements of their wards' if they have any. E.g.: If their ward secures highest marks in the semester or highest marks in a subject for that semester.

Password resetting

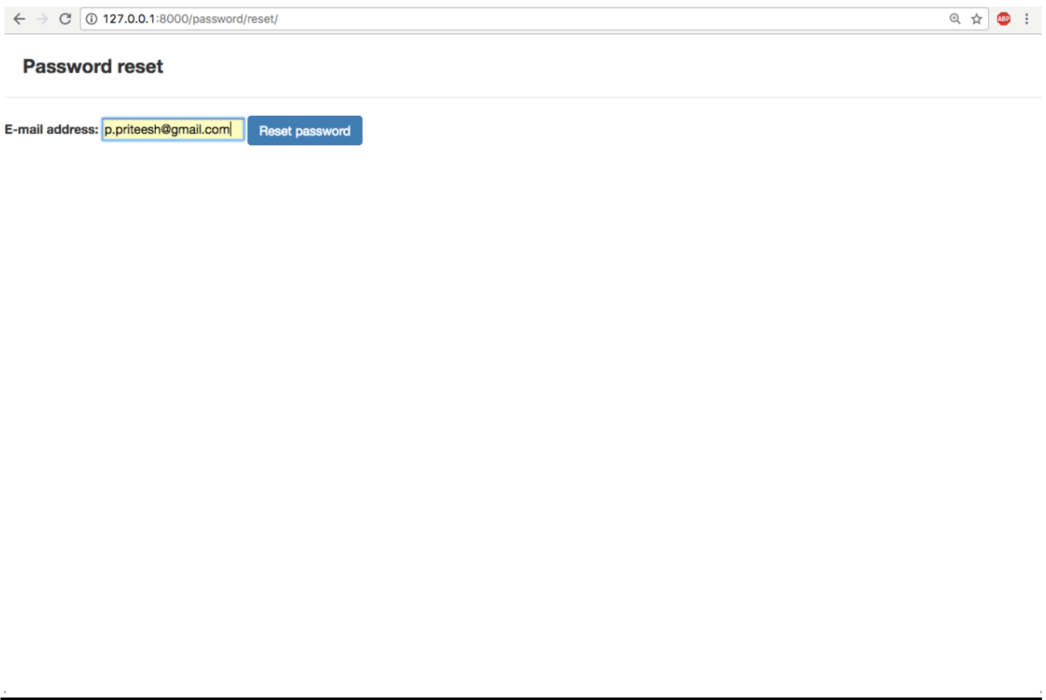

127.0.0.1:8000/password/reset/

Password reset

E-mail address:

If the user forgets the password he/she can easily retrieve by using the forgot password button. Once the button is clicked it is redirected to the above page where user needs to enter the registered email address to reset his password.

Password reset

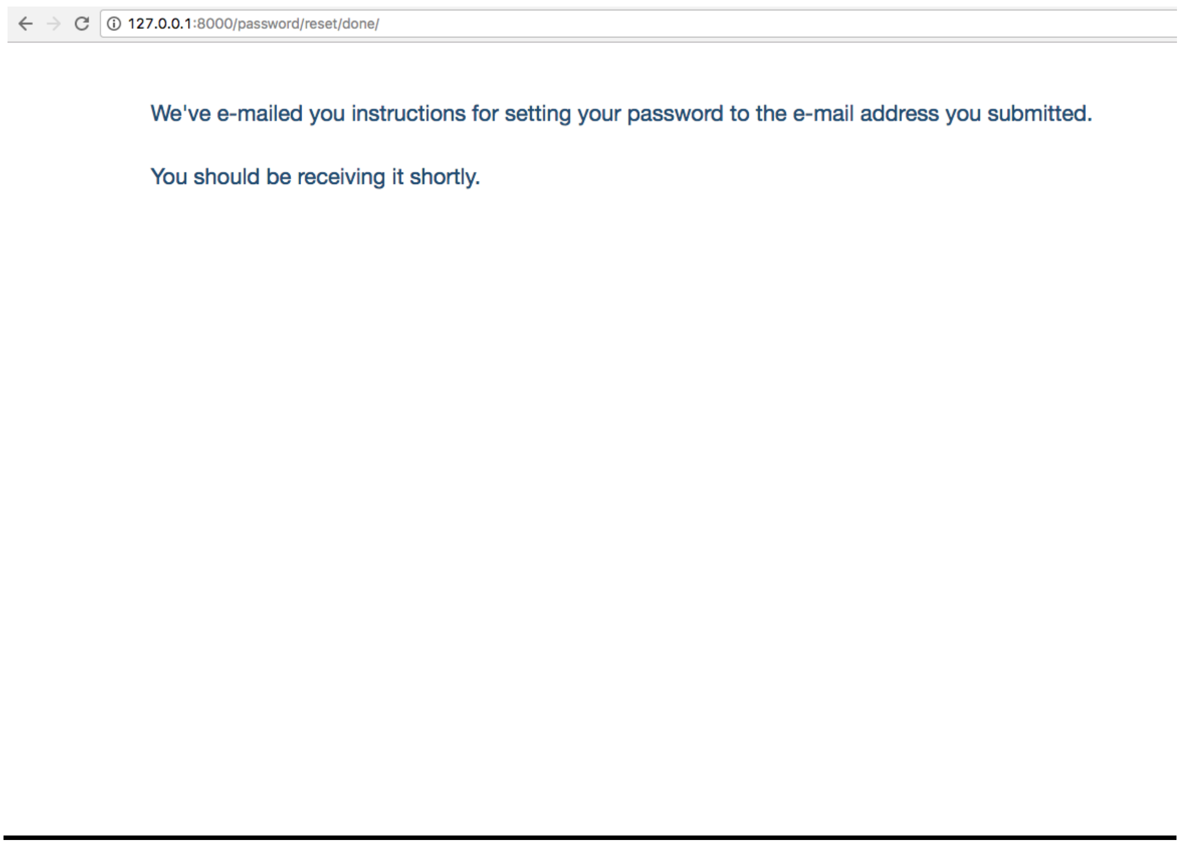

127.0.0.1:8000/password/reset/

Password reset

E-mail address: p.priteesh@gmail.com

Shows registered email filled in email address space.

Password reset confirmation page

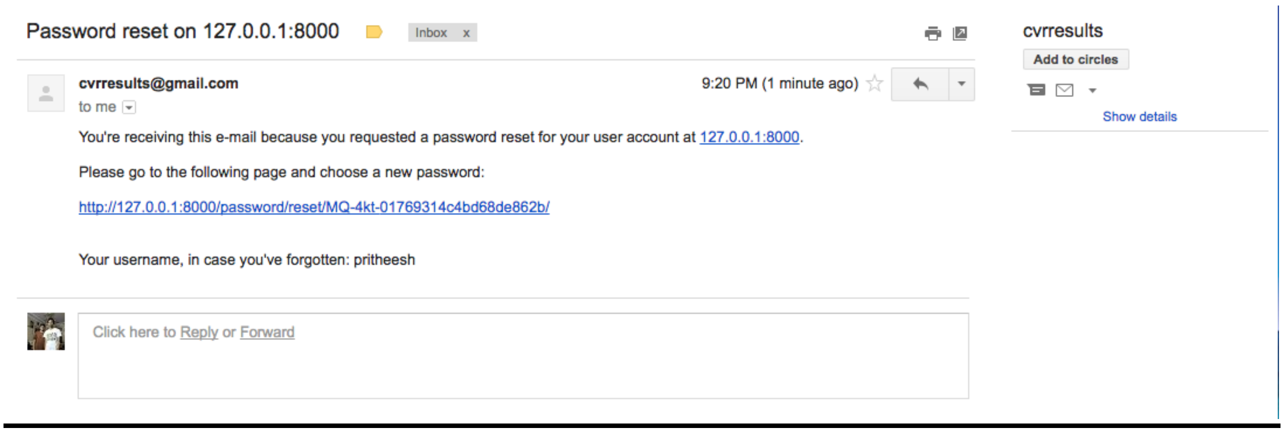

127.0.0.1:8000/password/reset/done/

We've e-mailed you instructions for setting your password to the e-mail address you submitted.

You should be receiving it shortly.

Once the user enters valid email address he/she will receive password reset link to their email address.

Password reset email


Password reset on 127.0.0.1:8000

cvrresults@gmail.com
to me

9:20 PM (1 minute ago)

You're receiving this e-mail because you requested a password reset for your user account at 127.0.0.1:8000.

Please go to the following page and choose a new password:

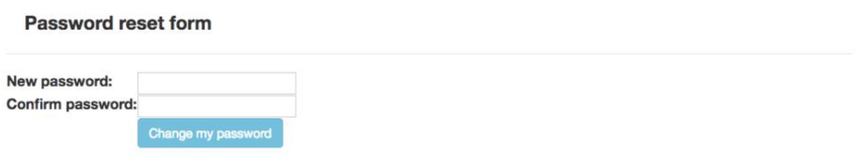
http://127.0.0.1:8000/password/reset/MQ-4kt-01769314c4bd68de862b/

Your username, in case you've forgotten: pritheesh

Click here to Reply or Forward

This is the screenshot of password rest link sent to the email of the user.

Password reset form

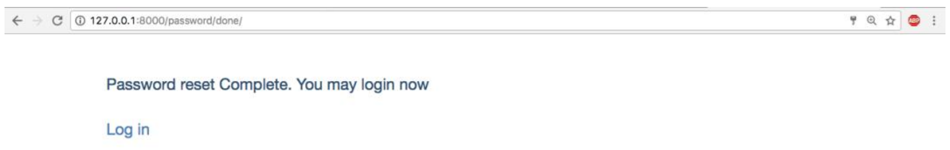

Password reset form

New password:

Confirm password:

Once the link is clicked the user will be redirected to the above page where he/she can reset the password.

Password reset confirmation



Password reset Complete. You may login now

After the user enters a new password, the user will be redirected to password reset confirmation page as shown above.

5. TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceived fault or weakness in a work product.

Test Case: Testing if email has already been used during registration earlier.

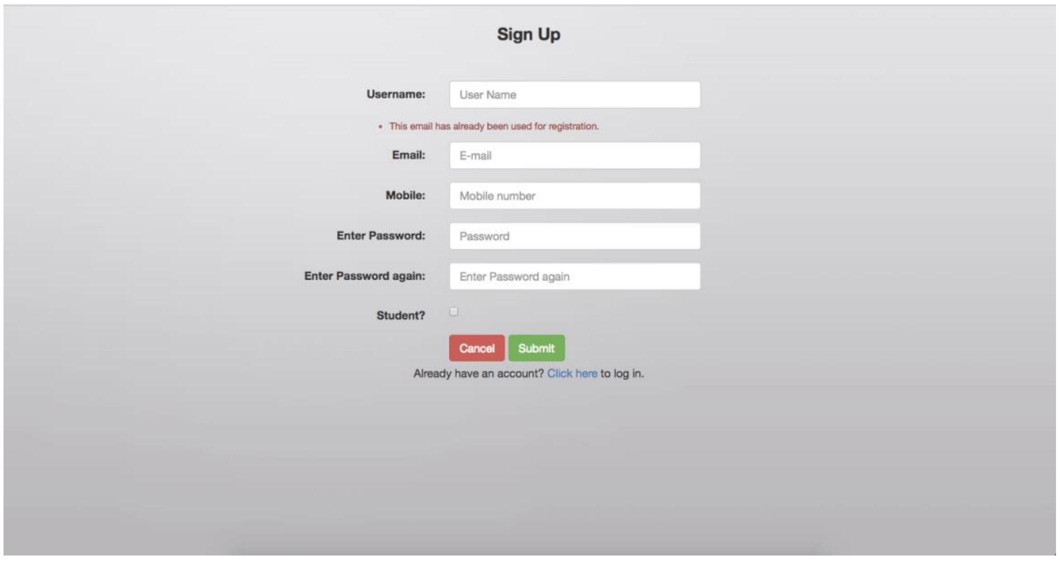
Description: An email can be used only once for registration.

Input: Email which is already registered.

Expected Outcome: User must be given an error message informing that the email has already been used for registration earlier.

Actual Outcome:

The error message is displayed when the user enters the email which has been used. The message displayed is “The email has already been used for registration”.



Sign Up

Username: User Name

- This email has already been used for registration.

Email: E-mail

Mobile: Mobile number

Enter Password: Password

Enter Password again: Enter Password again

Student?

Already have an account? Click here to log in.

Test Case: Filling required fields

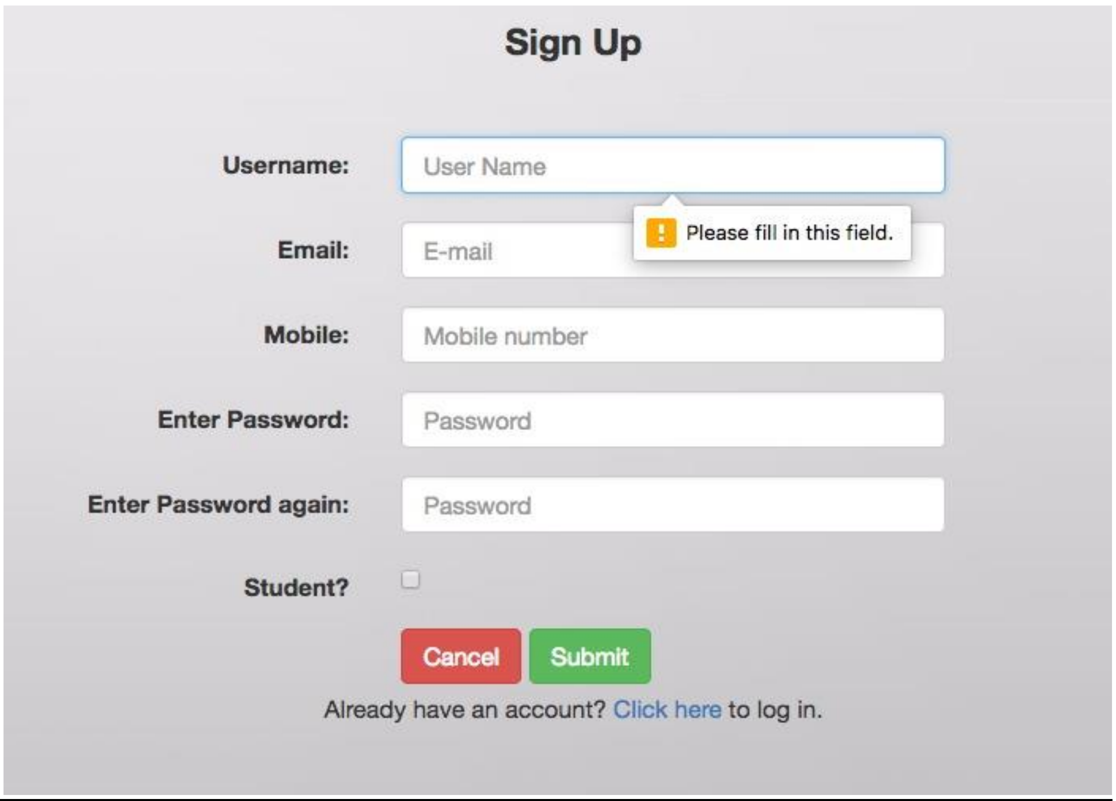
Description: Testing if all the fields in the registration form are filled by the user.

Input: Leaving the fields empty.

Expected Outcome: User must be displayed an error message informing that the field must be filled.

Actual Outcome:

All the fields in the registration form must be filled or else the user is given a message, "Please fill in this field".



Sign Up

Username:	User Name
Email:	E-mail
Mobile:	Mobile number
Enter Password:	Password
Enter Password again:	Password
Student?	☐

Please fill in this field.

Already have an account? Click here to log in.

Test Case: Testing validity of an email filled by user.

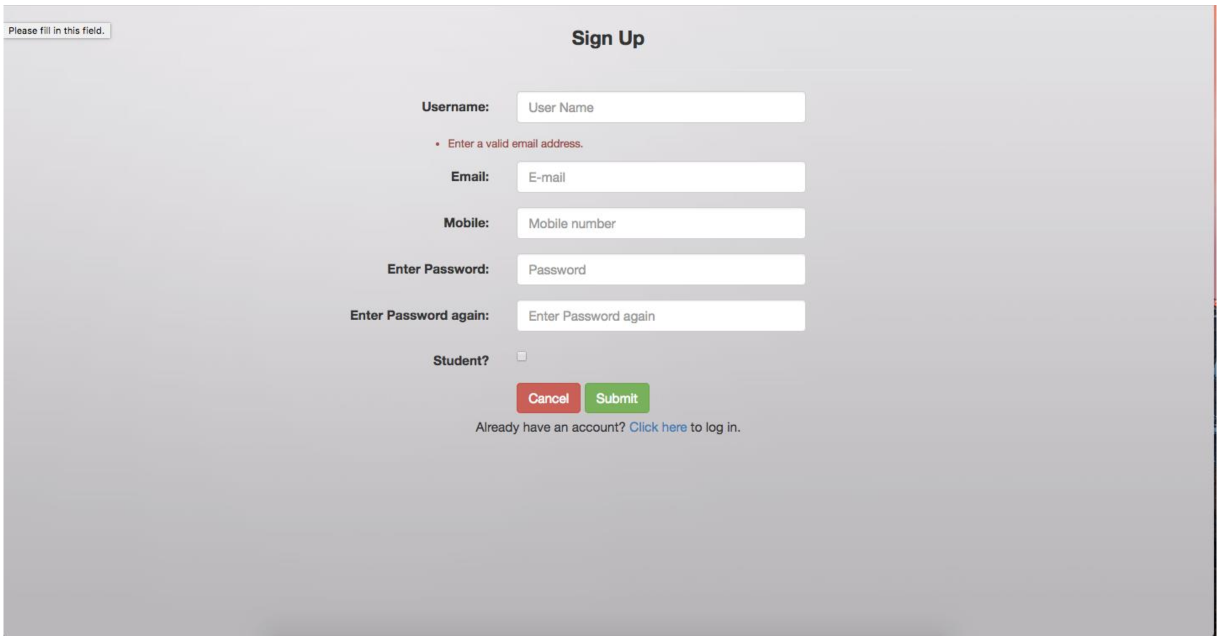
Description: The email entered by the user must be checked if it is valid. A valid email contains a sequence of alphanumeric and special characters followed by "@" and the domain followed by a "." and then the domain suffix which is usually of 2 or 3 characters.

Input: Invalid email address e.g.: aishwarya12.com

Expected Outcome: User must be given an error message, "Enter a valid email address".

Actual Outcome:

When a user enters the invalid email, the above page is displayed showing the error message, "Enter a Valid email address".



Please fill in this field.

Sign Up

Username: User Name

• Enter a valid email address.

Email: E-mail

Mobile: Mobile number

Enter Password: Password

Enter Password again: Enter Password again

Student?

Already have an account? Click here to log in.

Test Case: Validating mobile number and email entered by a student during registration.

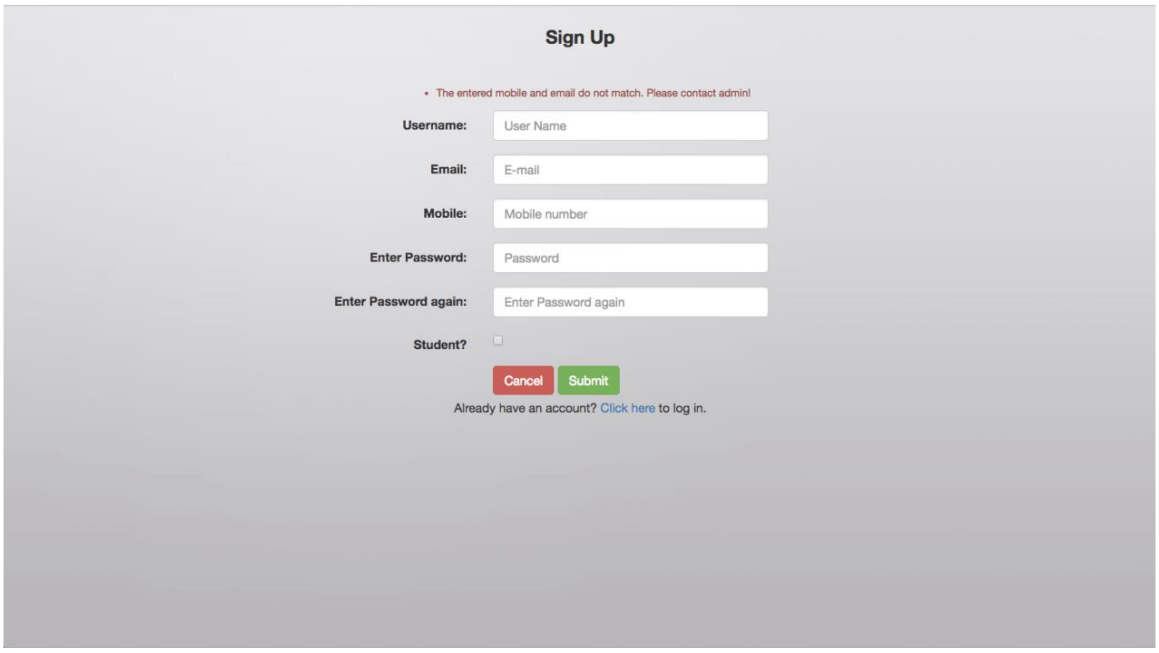
Description: When a user registers as a student, he/she must fill the details matching the information in the database, that was provided by the student during admission.

Input: Invalid details.

Expected Outcome: User must be displayed an error message, “The entered mobile and email do not match. Please contact admin”.

Actual Outcome:

When a user enters the invalid email and mobile, the above page is displayed showing the error message, “The entered mobile and email do not match. Please contact admin”.



Sign Up

- The entered mobile and email do not match. Please contact admin!

Username:	User Name
Email:	E-mail
Mobile:	Mobile number
Enter Password:	Password
Enter Password again:	Enter Password again
Student?	☐

Already have an account? Click here to log in.

Test case: Parent mobile validation

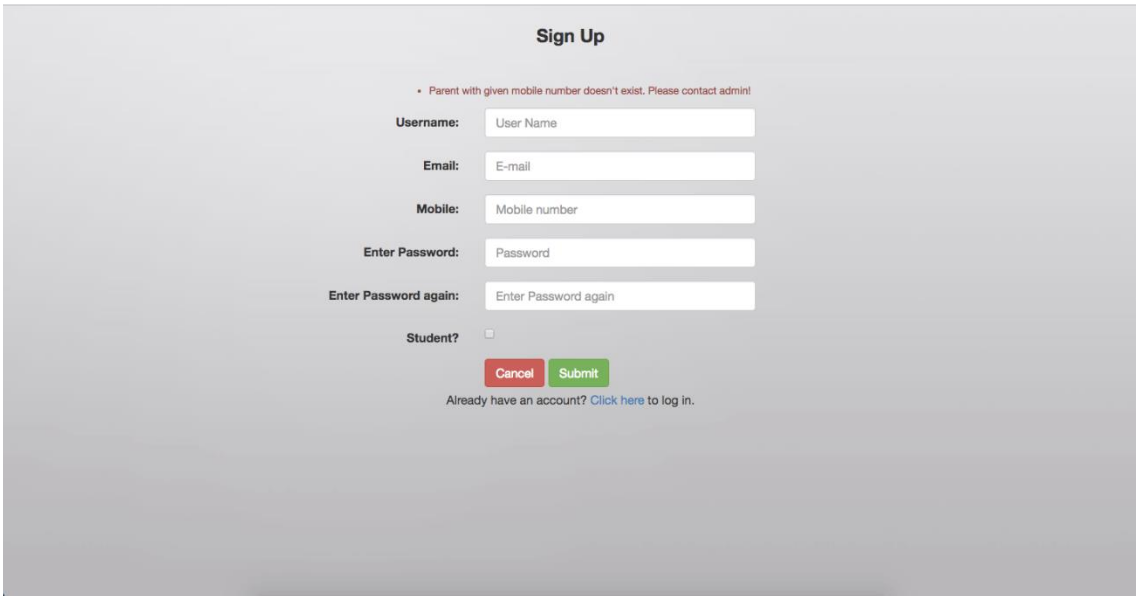
Description: When a user registers as a parent, he/she must fill the mobile number matching the information in the database, that was provided by the student during admission.

Input: entering wrong parent mobile number.

Expected Outcome: User must be displayed an error message, “Parent with a given mobile number doesn’t exist. Please contact admin”.

Actual Outcome:

When a user enters the invalid mobile, the above page is displayed showing the error message, “Parent with a given mobile number doesn’t exist. Please contact admin”.


Sign Up

- Parent with given mobile number doesn't exist. Please contact admin!

Username:	User Name
Email:	E-mail
Mobile:	Mobile number
Enter Password:	Password
Enter Password again:	Enter Password again
Student?	

Already have an account? Click here to log in.

Test Case: Weak password Validation

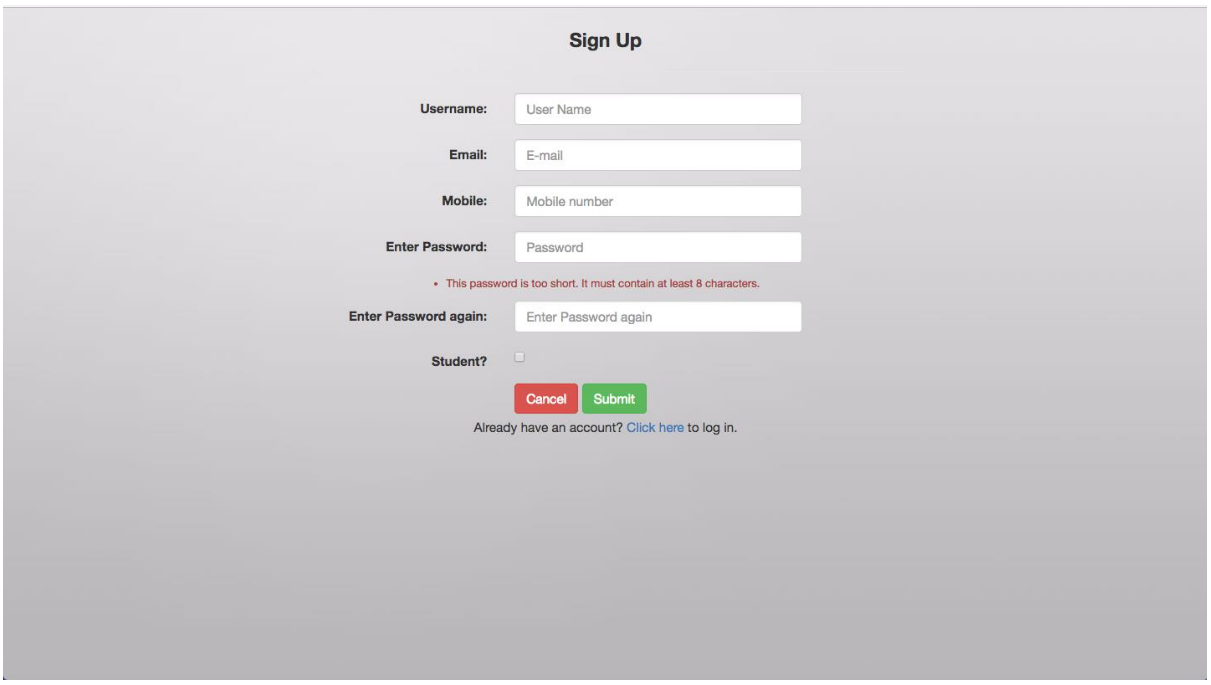
Description: Users are recommended to use stronger passwords which are difficult to be cracked.

Input: Entering password less than 8 characters.

Expected Outcome: User must be informed that the password is weak and must be suggested the ways to make the password stronger.

Actual Outcome:

When a user enters a weak password, the above page is displayed showing error message,“ The password is too short. It must at least contain 8 characters”.



Sign Up

Username: User Name

Email: E-mail

Mobile: Mobile number

Enter Password: Password

• This password is too short. It must contain at least 8 characters.

Enter Password again: Enter Password again

Student?

Already have an account? Click here to log in.

Test case: Password match validation

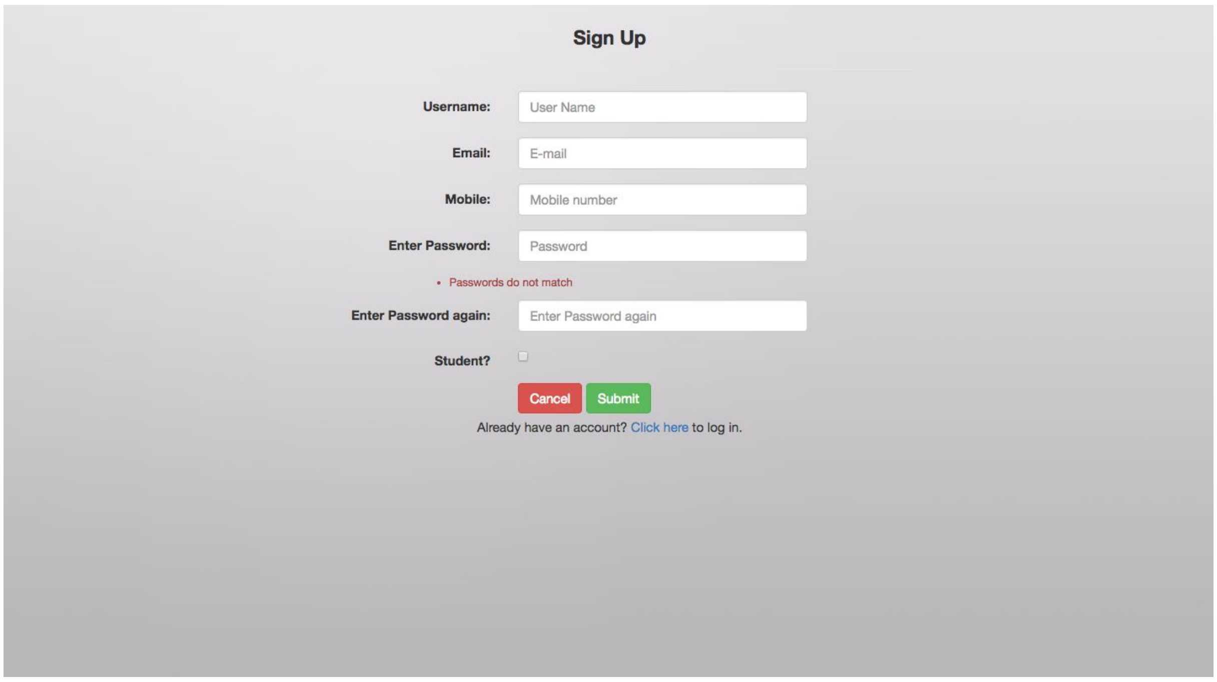
Description: The users are supposed to type same password in the enter password again field.

Input: Entering different passwords in both the password fields.

Expected Outcome: User must be given an error message , "Passwords do not match".

Actual Outcome:

When a user enters mismatched passwords, the above page is displayed showing the error message, "Passwords do not match".


Sign Up

Username: User Name

Email: E-mail

Mobile: Mobile number

Enter Password: Password

- Passwords do not match

Enter Password again: Enter Password again

Student?

Cancel Submit

Already have an account? Click here to log in.

Test case: Account activation testing

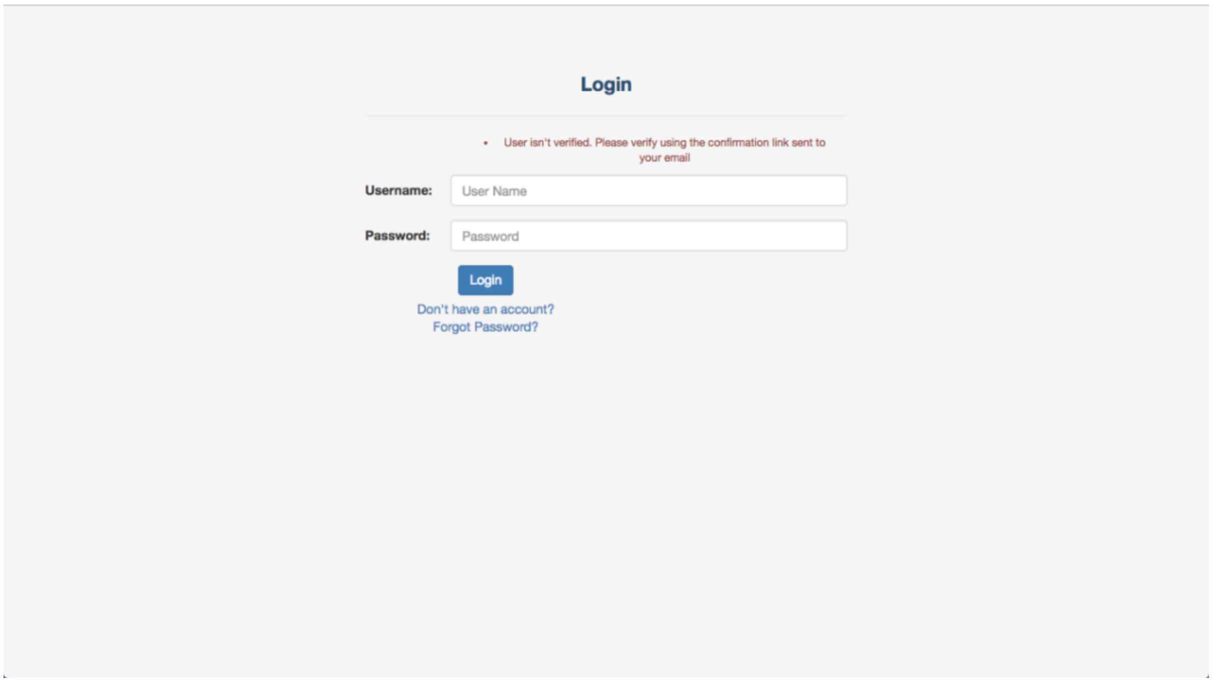
Description: A user can login only if the account is activated using the activation link sent by email.

Input: Logging in without activation.

Expected Outcome: User must be dislyed with an error message informing that the account must be activated using the link sent to the provided email during registration.

Actual Outcome:

When a user enters the credentials of the account which is not verified, the above page is displayed showing error message, “User isn’t verified. Please verify using the confirmation link sent to your email”.



Login

- User isn't verified. Please verify using the confirmation link sent to your email

Username: User Name

Password: Password

Test case: Login credentials testing

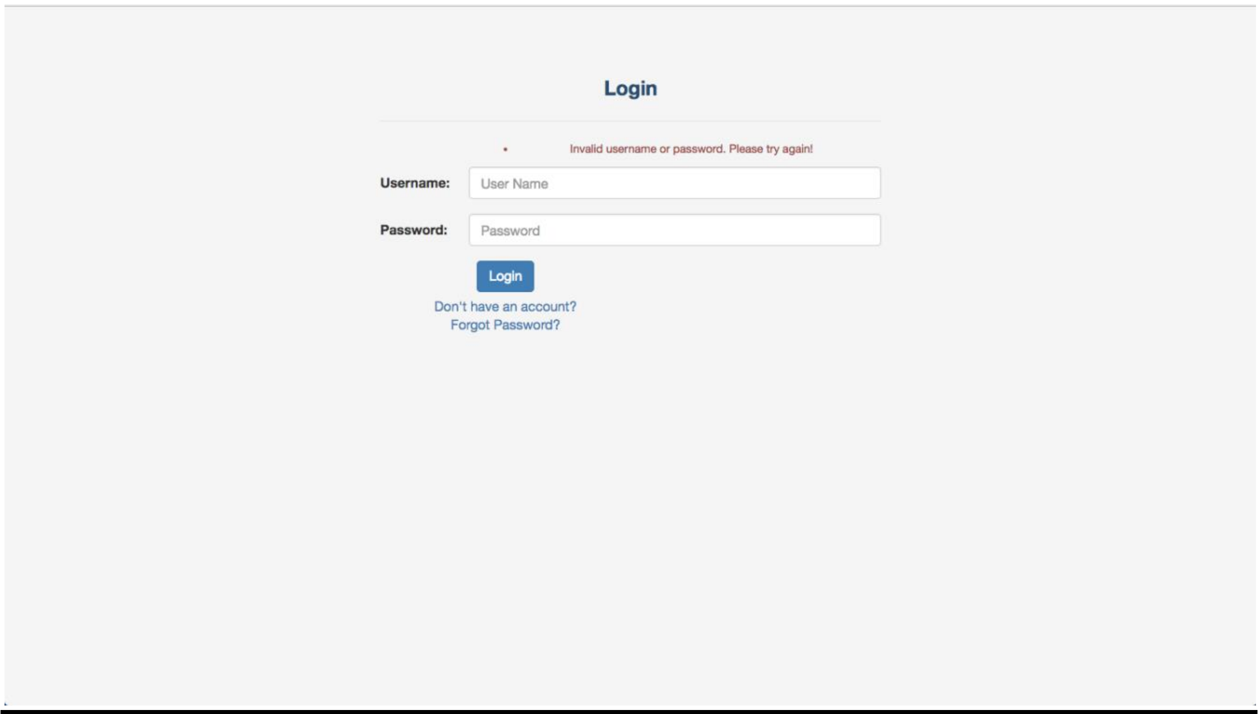
Description: A user should provide valid credentials during log in.

Input: Invalid credentials.

Expected Outcome: User must be displayed with an error message, "Invalid username or password. Please try again".

Actual Outcome:

When a user enters the invalid credentials, the above page is displayed showing the error message, "Invalid username or password. Please try again".



Login

• Invalid username or password. Please try again!

Username: User Name

Password: Password

6. Conclusion and Future scope

The Online Student Information System is an internet based application that can be accessed by all the authorized users.

User requirements keep changing as the system is being used. Some of the future enhancements that can be done to this system are:

- As the technology emerges, it is possible to upgrade the system and can be adaptable to desired environment.
- Student and faculty interaction can be implemented.
- Notes, material, files etc. can be shared by faculty
- Displaying Results Page for University.
- Extending the application for Attendance module.
- Providing API for mobile platforms.

REFERENCES

1. Brandon Lorenz Hands-On Django: Going Beyond the Polls Publication Date: March 25, 2015.
2. Daniel Roy Greenfeld and Audrey Roy Greenfeld: Two Scoops of Django: Best Practices for Django 1.8 paperback- May 15, 2015.
3. Zed Shaw's Hard Way: Learn Python the Hard Way: A Very Simple Introduction to the Terrifyingly Beautiful World of Computers and Code (3rd Edition) Publication Date: Oct 1, 2013.
4. Aidas Bendoraitis Web Development with Django Cookbook Publication Date: Oct. 16, 2014.
5. Karen M. Tracey Django 1.1 Testing and Debugging Publication Date: April 20, 2010.
6. Adrian Holovaty and Jacob Kaplan-Moss The Definitive Guide to Django: Web Development Done Right (Expert's Voice in Web Development) Publication Date: Dec. 9, 2007.
7. David Beazley and Brian K. Jones Python Cookbook, Third edition.
8. https://docs.python.org/2/
9. https://docs.djangoproject.com/en/1.10/
10. https://pypi.python.org/pypi
11. http://pythonhosted.org/
12. http://stackoverflow.com/
13. http://getbootstrap.com/
14. http://www.django-rest-framework.org/