Pune Institute of Computer Technology, Pune

2020-21

**Database Management Systems Lab**

**Mini-Project Report**

**Interview Preparation Web Application**

Done By:

Roll No. Name

31161 Shreya Rani  
 31164 Prathamesh Sonawane

31170 Prathamesh Thombre

**Title**

Interview Preparation Web Application

**Abstract**

We created a user friendly web application especially for Students who are preparing for placements. It contains Subject and topic wise questions which are frequently asked by the companies during interviews. We have used HTML, CSS and Bootstrap as Frontend and Django as Backend and MongoDB as a database to store our data.

**Introduction**

This project aims at reducing the manual process of finding different resources for preparation of an interview. Everything is just a click away! It saves time and increases learning efficiency.

**Scope**

The scope of the project is to give a simple and attractive application to simplify the work as well as to reduce the efforts while doing it offline or we can say by doing it with old methods.

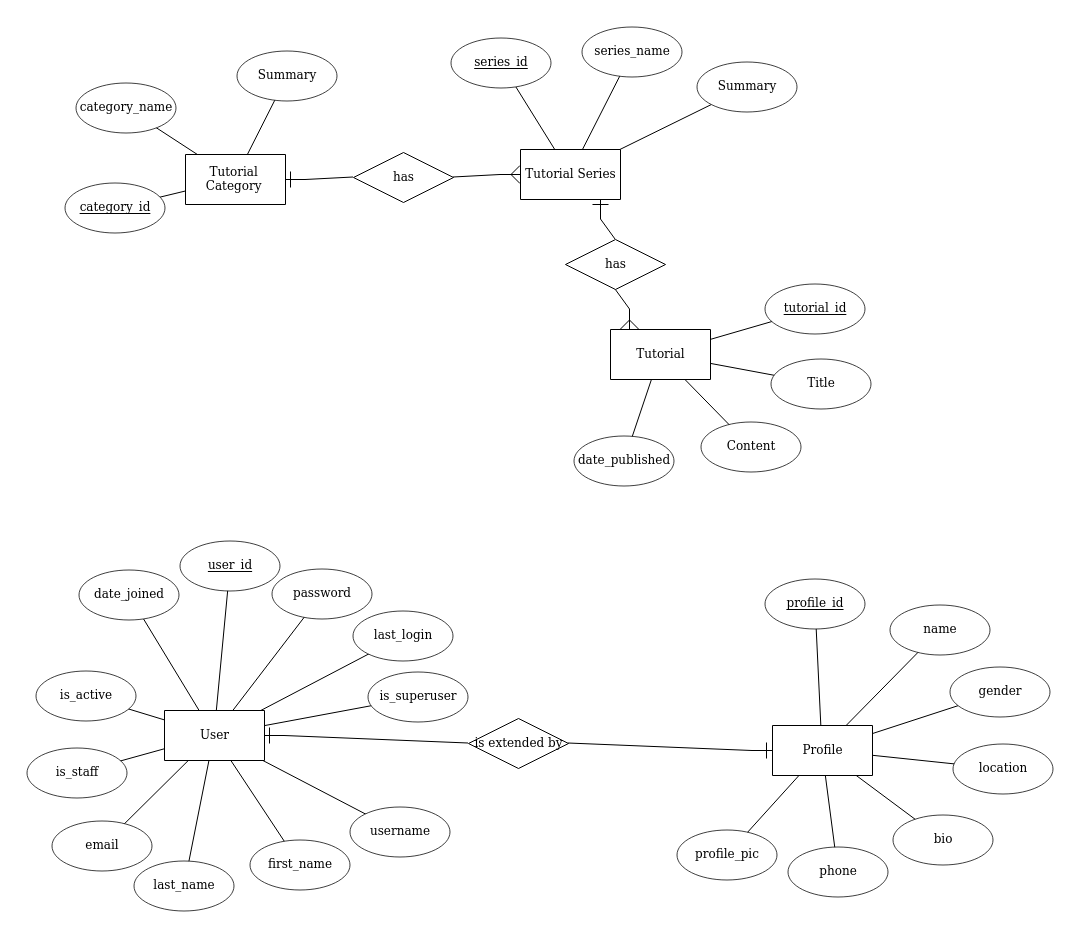
We can add a Chat Bot in future to help the students. Students can even clear their doubts from experts through chat.

**Requirements**OS - Windows 10/ Ubuntu

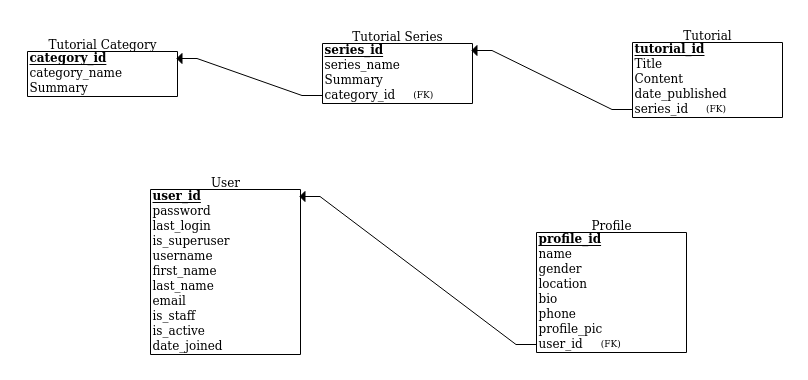
Hardware - i7 processor, 8 GB RAM  
Software - MongoDB 4.4.1, PyCharm Community Edition, Django  
Web Browser - Google Chrome, Microsoft Edge, Mozilla Firefox

**Data Modeling Features**

**ER-Diagram**

****

**Physical Data Model**

****

**Data Dictionary**

**Collection Name: auth\_user**  
> db.auth\_user.find().pretty()

{

"\_id" : ObjectId("5fc3fe47a014758ef149fddc"),

"id" : 1,

"password" : "pbkdf2\_sha256$150000$agJYHVZuSLco$mrrnsmHBLdIDh9TqLksR4NGjxhHsQNIpZnvdrHLsB1c=",

"last\_login" : ISODate("2020-09-02T05:14:47.840Z"),

"is\_superuser" : true,

"username" : "pratt3000",

"first\_name" : "",

"last\_name" : "",

"email" : "pratt3000@gmail.com",

"is\_staff" : true,

"is\_active" : true,

"date\_joined" : ISODate("2020-09-02T05:00:08.074Z")

}

{

"\_id" : ObjectId("5fc3fe47a014758ef149fddd"),

"id" : 2,

"password" : "pbkdf2\_sha256$180000$aTo0euYGC0EL$R0jMEXhwqMLpQHO+ItWX/Yq2GZPAsY3NsL/25Dj8dIA=",

"last\_login" : ISODate("2020-11-30T03:40:39.698Z"),

"is\_superuser" : true,

"username" : "admin",

"first\_name" : "",

"last\_name" : "",

"email" : "",

"is\_staff" : true,

"is\_active" : true,

"date\_joined" : ISODate("2020-11-24T14:37:36.941Z")

}

{

"\_id" : ObjectId("5fc3fe48a014758ef149fdde"),

"id" : 8,

"password" : "pbkdf2\_sha256$180000$RPUdiZi2n8R1$nkL3vpHAf4svN2X3qtSsWtJt89HPHUvJBpYHRd++BZc=",

"last\_login" : ISODate("2020-11-26T13:53:19.737Z"),

"is\_superuser" : false,

"username" : "pratt",

"first\_name" : "",

"last\_name" : "",

"email" : "pratt30010@gmail.com",

"is\_staff" : false,

"is\_active" : true,

"date\_joined" : ISODate("2020-11-26T13:51:51.138Z")

}

{

"\_id" : ObjectId("5fc3fe48a014758ef149fddf"),

"id" : 10,

"password" : "pbkdf2\_sha256$180000$f1kV2h2eMfYz$G5fUF3kSGrkxbLqxbuThwRBBNN2AHioDmb+2ww7swZA=",

"last\_login" : ISODate("2020-11-30T03:51:32.652Z"),

"is\_superuser" : false,

"username" : "timmy",

"first\_name" : "",

"last\_name" : "",

"email" : "timmy@gmail.com",

"is\_staff" : false,

"is\_active" : true,

"date\_joined" : ISODate("2020-11-26T17:50:02.337Z")

}

**Collection Name: main\_profile**> db.main\_profile.find().pretty()

{

"\_id" : ObjectId("5fc3fe48a014758ef149fe06"),

"id" : 1,

"user\_id" : 2,

"name" : "Admin",

"gender" : "Male",

"location" : "Pune",

"bio" : "This is admin bio",

"phone" : "8888889897",

"profile\_pic" : "images/profile/profile1.png"

}

{

"\_id" : ObjectId("5fc3fe49a014758ef149fe07"),

"id" : 5,

"user\_id" : 8,

"name" : "pratt",

"gender" : "Male",

"location" : null,

"bio" : null,

"phone" : null,

"profile\_pic" : "images/profile/Screenshot\_from\_2020-10-19\_09-53-56.png"

}

{

"\_id" : ObjectId("5fc3fe49a014758ef149fe08"),

"id" : 7,

"user\_id" : 10,

"name" : "timmy",

"gender" : "Male",

"location" : "New York",

"bio" : "Actor",

"phone" : "7541578454",

"profile\_pic" : "images/profile/tim.jpg"

}

**Collection Name: main\_tutorialcategory**> db.main\_tutorialcategory.find().pretty()

{

"\_id" : ObjectId("5fc3fe48a014758ef149fde4"),

"id" : 3,

"tutorial\_category" : "Database Management Systems",

"category\_summary" : "DBMS Interview Questions",

"category\_slug" : "dbms"

}

{

"\_id" : ObjectId("5fc3fe48a014758ef149fde5"),

"id" : 5,

"tutorial\_category" : "Object Oriented Programming (JAVA)",

"category\_summary" : "OOP Interview Questions",

"category\_slug" : "oop"

}

{

"\_id" : ObjectId("5fc3fe48a014758ef149fde6"),

"id" : 6,

"tutorial\_category" : "Data Structures and Algorithms",

"category\_summary" : "DSA Interview Questions",

"category\_slug" : "dsa"

}  
  
**Collection Name: main\_tutorialseries**> db.main\_tutorialseries.find().pretty()

{

"\_id" : ObjectId("5fc3fe48a014758ef149fde7"),

"id" : 2,

"tutorial\_series" : "DBMS Questions",

"tutorial\_category\_id" : 3,

"series\_summary" : "General, SQL Query Based Questions"

}

{

"\_id" : ObjectId("5fc3fe48a014758ef149fde8"),

"id" : 9,

"tutorial\_series" : "JAVA",

"tutorial\_category\_id" : 5,

"series\_summary" : "General Interview Questions"

}

{

"\_id" : ObjectId("5fc3fe48a014758ef149fde9"),

"id" : 14,

"tutorial\_series" : "DSA Questions",

"tutorial\_category\_id" : 6,

"series\_summary" : "General Interview Questions"

}

**Collection Name: main\_tutorial**> db.main\_tutorial.find().pretty()

{

"\_id" : ObjectId("5fc3fe48a014758ef149fdea"),

"id" : 17,

"tutorial\_title" : "What do you understand by Java virtual machine?",

"tutorial\_content" : "<p>Java Virtual Machine is a virtual machine that enables the computer to run the Java program. JVM acts like a run-time engine which calls the main method present in the Java code. JVM is the specification which must be implemented in the computer system. The Java code is compiled by JVM to be a Bytecode which is machine independent and close to the native code.</p>",

"tutorial\_published" : ISODate("2020-11-25T09:24:09Z"),

"tutorial\_series\_id" : 9,

"tutorial\_slug" : "java\_1"

}

{

"\_id" : ObjectId("5fc3fe48a014758ef149fdeb"),

"id" : 18,

"tutorial\_title" : "How many types of memory areas are allocated by JVM?",

"tutorial\_content" : "<ol>\r\n<li><strong>Class(Method) Area:</strong> Class Area stores per-class structures such as the runtime constant pool, field, method data, and the code for methods.</li>\r\n<li><strong>Heap:</strong> It is the runtime data area in which the memory is allocated to the objects</li>\r\n<li><strong>Stack:</strong> Java Stack stores frames. It holds local variables and partial results, and plays a part in method invocation and return. Each thread has a private JVM stack, created at the same time as the thread. A new frame is created each time a method is invoked. A frame is destroyed when its method invocation completes.</li>\r\n<li><strong>Program Counter Register:</strong> PC (program counter) register contains the address of the Java virtual machine instruction currently being executed.</li>\r\n<li><strong>Native Method Stack:</strong> It contains all the native methods used in the application.</li>\r\n</ol>",

"tutorial\_published" : ISODate("2020-11-25T11:46:49Z"),

"tutorial\_series\_id" : 9,

"tutorial\_slug" : "java\_2"

}

{

"\_id" : ObjectId("5fc3fe48a014758ef149fdec"),

"id" : 19,

"tutorial\_title" : "What is JIT compiler?",

"tutorial\_content" : "<p><strong>Just-In-Time(JIT) compiler:</strong> It is used to improve the performance. JIT compiles parts of the bytecode that have similar functionality at the same time, and hence reduces the amount of time needed for compilation. Here the term &ldquo; compiler & rdquo; refers to a translator from the instruction set of a Java virtual machine (JVM) to the instruction set of a specific CPU.</p>",

"tutorial\_published" : ISODate("2020-11-25T11:47:47Z"),

"tutorial\_series\_id" : 9,

"tutorial\_slug" : "java\_3"

}

{

"\_id" : ObjectId("5fc3fe48a014758ef149fded"),

"id" : 20,

"tutorial\_title" : "What are the various access specifiers in Java?",

"tutorial\_content" : "<p>In Java, access specifiers are the keywords which are used to define the access scope of the method, class, or a variable. In Java, there are four access specifiers given below.</p>\r\n<ul class=\"points\">\r\n<li><strong>Public</strong> The classes, methods, or variables which are defined as public, can be accessed by any class or method.</li>\r\n<li><strong>Protected</strong> Protected can be accessed by the class of the same package, or by the sub-class of this class, or within the same class.</li>\r\n<li><strong>Default</strong> Default are accessible within the package only. By default, all the classes, methods, and variables are of default scope.</li>\r\n<li><strong>Private</strong> The private class, methods, or variables defined as private can be accessed within the class only.</li>\r\n</ul>",

"tutorial\_published" : ISODate("2020-11-25T11:48:51Z"),

"tutorial\_series\_id" : 9,

"tutorial\_slug" : "java\_4"

}

{

"\_id" : ObjectId("5fc3fe48a014758ef149fdee"),

"id" : 21,

"tutorial\_title" : "What is the purpose of static methods and variables?",

"tutorial\_content" : "<p>The methods or variables defined as static are shared among all the objects of the class. The static is the part of the class and not of the object. The static variables are stored in the class area, and we do not need to create the object to access such variables. Therefore, static is used in the case, where we need to define variables or methods which are common to all the objects of the class.</p>\r\n<p>For example, In the class simulating the collection of the students in a college, the name of the college is the common attribute to all the students. Therefore, the college name will be defined as <strong>static</strong>.</p>",

"tutorial\_published" : ISODate("2020-11-25T12:48:16Z"),

"tutorial\_series\_id" : 9,

"tutorial\_slug" : "java\_5"

}

…..

{

"\_id" : ObjectId("5fc3fe48a014758ef149fdf6"),

"id" : 31,

"tutorial\_title" : "What are the different levels of abstraction in the DBMS?",

"tutorial\_content" : "<p>There are three levels of data abstraction in DBMS. They are:</p> <p>Physical Level: It is the lowest level of abstraction and describes how the data is stored. <br> Logical Level: This is the next level of abstraction after the Physical level. This layer determines what data is stored in the database, and what is the relationship between the data points. <br> View Level: The View Level is the highest level of abstraction and it describes only a part of the entire database. </p>",

"tutorial\_published" : ISODate("2020-11-29T17:01:28Z"),

"tutorial\_series\_id" : 2,

"tutorial\_slug" : "5"

}

**Relational Database Design**1. auth\_user

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| id | password | last\_ login | is\_ superuser | username | first\_ name | last\_ name | email | is\_staff | is\_active | date\_ joined |

2. main\_profile

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| id | user\_id | name | gender | location | bio | phone | profile\_pic |

3. main\_tutorialcategory

|  |  |  |  |
| --- | --- | --- | --- |
| id | tutorial\_category | category\_summary | category\_slug |

4. main\_tutorialseries

|  |  |  |  |
| --- | --- | --- | --- |
| id | tutorial\_series | tutorial\_category\_id | series\_summary |

5. main\_tutorial

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| id | tutorial\_title | tutorial\_content | tutorial\_ published | tutorial\_ series\_id | tutorial\_slug |

**Graphical User Interface**The application is very user friendly and uses a GUI interface implemented in Django and HTML to communicate with the user. Various features are self - explanatory. Registered users can login and view the questions subject wise. Unauthorized users first need to register. There is a navigation bar to help the users navigate to home, profile page, about us, logout. They can easily update their profile and can also upload their profile photo.

**Source Code**

1. messages.html

{% if messages %}

{% for message in messages %}

{% if message.tags == 'success'%}

<script>M.toast({html: "{{message}}", classes: 'green rounded', displayLength:2000});</script>

{% elif message.tags == 'info'%}

<script>M.toast({html: "{{message}}", classes: 'blue rounded', displayLength:2000});</script>

{% elif message.tags == 'warning'%}

<script>M.toast({html: "{{message}}", classes: 'orange rounded', displayLength:10000});</script>

{% elif message.tags == 'error'%}

<script>M.toast({html: "{{message}}", classes: 'red rounded', displayLength:10000});</script>

{% endif %}

{% endfor %}

{% endif %}

1. navbar.html

<nav>

<div class="nav-wrapper">

<a href="/" class="brand-logo" >Interview Prep</a>

<ul id="nav-mobile" class="right hide-on-med-and-down">

<li><a href="/">Home</a></li>

<li> <a href="/aboutus/">About Us</a></li>

{% if user.is\_authenticated%}

<li><a href="/profilepage/">{{user.username}}</a> </li>

<li><a href="/logout/">Logout</a></li>

{% else %}

<li><a href="/register/">Register</a></li>

<li><a href="/login/">Login</a></li>

{% endif %}

</ul>

</div>

</nav>

1. aboutus.html

{% extends "main/header.html" %}

{% block content %}

<form method="POST">

{% csrf\_token%}

<div class="card">

<div class="card-header">

<h4>Team Members</h4>

</div>

<div class="card-body">

<form>

<div class="input-group form-group">

<div class="input-group-prepend">

<span class="input-group-text"><i class="fas fa-user"></i> </span>

</div>

<b> Shreya Rani </b>

</div>

<div class="input-group form-group">

<div class="input-group-prepend">

<span class="input-group-text"><i class="fas fa-user"></i> </span>

</div>

<b> Prathamesh Thombre </b>

</div>

<div class="input-group form-group">

<div class="input-group-prepend">

<span class="input-group-text"><i class="fas fa-user"></i> </span>

</div>

<b> Prathamesh Sonawane</b>

</div>

</form>

</div>

<div class="card-footer">

<div class="d-flex justify-content-center links">

Students of PICT, Pune

</div>

</div>

</div>

</form>

{% endblock %}

1. categories.html

{% extends 'main/header.html' %}

{% block content %}

<div class="row">

{% for cat in categories %}

<div class="col s12 m6 l4">

<div style="margin: 0 auto">

<a href="{{cat.category\_slug}}" style="color:#000000;">

<div class="card hoverable">

<div class="card-content">

<div class="card-title">{{cat.tutorial\_category}}</div>

<p>{{cat.category\_summary}}</p>

</div>

</div>

</a>

</div>

</div>

{% endfor %}

</div>

{% endblock %}

1. category.html

{% extends 'main/header.html' %}

{% block content %}

<div class="row">

{% for tut, partone in part\_ones.items %}

<div class="col s12 m6 l4">

<div style="text-align: center">

<a href="{{partone}}", style="color:#000000;">

<div class="card hoverable">

<div class="card-content">

<div class="card-title">{{tut.tutorial\_series}}</div>

<p>{{tut.series\_summary}}</p>

</div>

</div>

</a>

</div>

</div>

{% endfor %}

</div>

{% endblock %}

1. header.html

<head>

{% load static %}

<link href="{% static 'tinymce/css/prism.css' %}" rel="stylesheet">

*<!-- Compiled and minified CSS -->*

<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css" integrity="sha384-MCw98/SFnGE8fJT3GXwEOngsV7Zt27NXFoaoApmYm81iuXoPkFOJwJ8ERdknLPMO" crossorigin="anonymous">

<link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.3.1/css/all.css" integrity="sha384-mzrmE5qonljUremFsqc01SB46JvROS7bZs3IO2EmfFsd15uHvIt+Y8vEf7N7fWAU" crossorigin="anonymous">

<link href="{% static 'main/css/materialize.css' %}" rel="stylesheet">

*<!-- Compiled and minified JavaScript -->*

<script src="https://cdnjs.cloudflare.com/ajax/libs/materialize/1.0.0/js/materialize.min.js"></script>

*<!--Custom styles-->*

<link rel="stylesheet" type="text/css" href=" {% static 'main/css/styles.css' %}">

</head>

<body>

{% include "main/includes/navbar.html" %}

{% include "main/includes/messages.html" %}

<div class = "container-fluid">

<div class="d-flex justify-content-center h-100">

<br>

{% block content %}

{% endblock%}

</div>

</div>

</body>

<script src="{% static 'tinymce/js/prism.js' %}"></script>

<script>M.AutoInit();</script>

1. home.html

{% extends "main/header.html" %}

{% block content %}

<div class="row">

{% for tut in tutorials %}

<div class="col s12 m6 l4">

<div class="card blue-grey darken-1">

<div class="card-content white-text">

<span class="card-title">{{tut.tutorial\_title}}</span>

<p style = "font-size :70%">{{tut.tutorial\_published}}</p>

<p>{{tut.tutorial\_content|safe}}</p>

</div>

<div class="card-action">

<a href="#">Visit</a>

</div>

</div>

</div>

{% endfor %}

</div>

{% endblock %}

1. login.html

{% extends "main/header.html" %}

{% block content %}

<form method="POST">

{% csrf\_token%}

<div class="card">

<div class="card-header">

<h4>Log In</h4>

</div>

<div class="card-body">

<form>

<div class="input-group form-group">

<div class="input-group-prepend">

<span class="input-group-text"><i class="fas fa-user"></i> </span>

</div>

<b> Username </b> {{form.username}}

</div>

<div class="input-group form-group">

<div class="input-group-prepend">

<span class="input-group-text"><i class="fas fa-key"></i></span>

</div>

<b> Password </b> {{form.password}}

</div>

<div class="form-group">

<input type="submit" value="Login" class="btn float-right login\_btn">

</div>

</form>

</div>

<div class="card-footer">

<div class="d-flex justify-content-center links">

Don't have an account?<a href="/register/"><b> Sign Up </b></a>

</div>

</div>

</div>

</form>

{% endblock %}

1. profile.html

{% extends "main/header.html" %}

{% block content %}

<div class="row-12" style="margin: 20px">

<div class="card card-body">

*<!-- <a class="btn btn-warning" href="/"> &#8592; Back to Home</a> -->*

<hr/>

<h3 style="text-align: center">Profile Settings</h3>

<hr/>

<form method="POST" action="" enctype="multipart/form-data">

<img class="profile-pic" src="{{ request.user.profile.profile\_pic.url }}" style="height: 20vh; width: 13vw">

<p>{{form.profile\_pic}}<br></p>

<br>

{% csrf\_token %}

<p>Name: {{form.name}}</p>

<p>Gender: {{form.gender}}</p>

<p>Location: {{form.location}}</p>

<p>Bio: {{form.bio}}</p>

<p>Phone: {{form.phone}}</p>

<input class="btn btn-primary" type="submit" name="Update Information"/>

</form>

</div>

</div>

{% endblock %}

1. register.html

{% extends "main/header.html" %}

{% block content %}

<form method="POST">

{% csrf\_token%}

<div class="card">

<div class="card-header">

<h4>Register</h4>

</div>

<div class="card-body">

<form>

<div class="input-group form-group">

<b> Enter Username </b> {{form.username}}

</div>

<div class="input-group form-group">

<b> Enter Email-id </b> {{form.email}}

</div>

<div class="input-group form-group">

<b> Enter Password </b> {{form.password1}}

</div>

<div class="input-group form-group">

<b> Confirm Password </b> {{form.password2}}

</div>

<div class="form-group">

<input type="submit" value="Sign Up" class="btn float-right login\_btn">

</div>

</form>

</div>

<div class="card-footer">

<div class="d-flex justify-content-center links">

Already have an account?<a href="/login/"><b> Log In </b></a>

</div>

</div>

</div>

</form>

{% endblock %}

1. tutorial.html

{% extends 'main/header.html' %}

{% block content %}

<div class="row" style="width: 100%">

<div class="col-md-5 " style="background-color: #464866;">

<div class="collapsible popout">

{% for tutorial in sidebar %}

{% if forloop.counter0 == this\_tutorial\_idx %}

<li class="active" style="background-color: #000000;background-image: linear-gradient(147deg, #000000 0%, #04619f 74%); margin-top: 2vh;">

<div class="collapsible-header">{{tutorial.tutorial\_title}}<br>(currently viewing)</div>

</li>

{% else %}

<li style="background-color: #000000;background-image: linear-gradient(147deg, #000000 0%, #04619f 74%);

margin-top: 2vh">

<div class="collapsible-header">{{tutorial.tutorial\_title}}</div>

<div class="collapsible-body" style="background-color: #EDF6FF">

<p><a href="/{{tutorial.tutorial\_slug}}">

<button class="btn waves-effect waves-light right-align"

style="color:black">Go

</button>

</a></p>

</div>

</li>

{% endif %}

{% endfor %}

</div>

</div>

<div class="col-md-7">

<div id="container1" class="container1">

<h4 class="box1">{{tutorial.tutorial\_title}}</h4>

<p class="box2" style="font-size:70%">Published {{tutorial.tutorial\_published}}</p>

<div class="box3" style="background-color: #F7F9FB">

<p>{{tutorial.tutorial\_content|safe}}</p>

</div>

</div>

</div>

</div>

{% endblock %}

1. models.py

from django.db import models

from datetime import datetime

from django.utils import timezone

from django.contrib.auth.models import User

*# Create your models here.*

*# migration need to be done when you add models*

class TutorialCategory(models.Model):

tutorial\_category = models.CharField(max\_length=200)

category\_summary = models.CharField(max\_length=200)

category\_slug = models.CharField(max\_length=200, default=1)

class Meta:

*# Gives the proper plural name for admin*

verbose\_name\_plural = **"Categories"**

def \_\_str\_\_(self):

return self.tutorial\_category

class TutorialSeries(models.Model):

tutorial\_series = models.CharField(max\_length=200)

tutorial\_category = models.ForeignKey(TutorialCategory, default=1, verbose\_name=**"Category"**, on\_delete=models.SET\_DEFAULT)

series\_summary = models.CharField(max\_length=200)

class Meta:

*# otherwise we get "Tutorial Seriess in admin"*

verbose\_name\_plural = **"Series"**

def \_\_str\_\_(self):

return self.tutorial\_series

class Tutorial(models.Model):

tutorial\_title = models.CharField(max\_length=200)

tutorial\_content = models.TextField()

tutorial\_published = models.DateTimeField(**'date published'**, default=timezone.now)

*#https://docs.djangoproject.com/en/2.1/ref/models/fields/#django.db.models.ForeignKey.on\_delete*

tutorial\_series = models.ForeignKey(TutorialSeries, default=1, verbose\_name=**"Series"**, on\_delete=models.SET\_DEFAULT)

tutorial\_slug = models.CharField(max\_length=200, default=1)

def \_\_str\_\_(self):

return self.tutorial\_title

class Profile(models.Model):

GENDER = (

(**'Male'**, **'Male'**),

(**'Female'**, **'Female'**),

(**'Other'**, **'Other'**),

)

user = models.OneToOneField(User, null=True, on\_delete=models.CASCADE)

name = models.CharField(max\_length=100, null=True, blank=True)

gender = models.CharField(max\_length=20, null=True, choices=GENDER)

location = models.CharField(max\_length=100, null=True, blank=True)

bio = models.CharField(max\_length=100, null=True, blank=True)

phone = models.CharField(max\_length=12, null=True, blank=True)

profile\_pic = models.ImageField(upload\_to=**'images/profile'**, default=**'profile1.png'**)

def \_\_str\_\_(self):

return str(self.user)

1. views.py

from django.shortcuts import render, redirect

from django.http import HttpResponse

from .models import Tutorial, TutorialCategory, TutorialSeries, Profile

from django.contrib.auth.forms import AuthenticationForm

*# for forgot pass add email later in UserCreationForm, its not there by default*

from django.contrib.auth import login, logout, authenticate

from django.contrib import messages

from .forms import NewUserForm, ProfileForm

from django.contrib.auth.decorators import login\_required

@login\_required(login\_url=**'main:login'**)

def single\_slug(request, single\_slug):

categories = [c.category\_slug for c in TutorialCategory.objects.all()]

if single\_slug in categories:

matching\_series = TutorialSeries.objects.filter(tutorial\_category\_\_category\_slug=single\_slug)

series\_urls = {}

for m in matching\_series.all():

part\_one = Tutorial.objects.filter(tutorial\_series\_\_tutorial\_series=m.tutorial\_series).earliest(**"tutorial\_published"**)

series\_urls[m] = part\_one.tutorial\_slug

return render(

request,

**"main/category.html"**,

{**"part\_ones"**:series\_urls}

)

tutorials = [t.tutorial\_slug for t in Tutorial.objects.all()]

if single\_slug in tutorials:

this\_tutorial = Tutorial.objects.get(tutorial\_slug=single\_slug)

tutorials\_from\_series = Tutorial.objects.filter(tutorial\_series\_\_tutorial\_series=this\_tutorial.tutorial\_series).order\_by(**"tutorial\_published"**)

this\_tutorial\_idx = list(tutorials\_from\_series).index(this\_tutorial)

return render(request = request,

template\_name=**'main/tutorial.html'**,

context = {**"tutorial"**:this\_tutorial,

**"sidebar"**:tutorials\_from\_series,

**"this\_tutorial\_idx"**:this\_tutorial\_idx,

})

return HttpResponse(**f"**{single\_slug} **does not correspond to anything"**)

@login\_required(login\_url=**'main:login'**)

def homepage(request):

return render(request=request, template\_name=**"main/categories.html"**, context = {**"categories"**: TutorialCategory.objects.all})

def register(request):

*# to register page*

if request.user.is\_authenticated:

return redirect(**'main: homepage'**)

if request.method == **"POST"**:

form = NewUserForm(request.POST)

if form.is\_valid():

user = form.save()

username = form.cleaned\_data.get(**'username'**)

*# creating Profile for user*

Profile.objects.create(

user=user,

name=username

)

messages.success(request, **f"New Account created :** {username}**"**)

login(request, user)

messages.info(request, **f"you are now logged in as :** {username}**"**)

return redirect(**"main:homepage"**)

else:

*#for msg in form.error\_messages:*

*# messages.error(request, f"{msg}:{form.error\_messages[msg]}")*

password1 = form.data[**'password1'**]

password2 = form.data[**'password2'**]

email = form.data[**'email'**]

for msg in form.errors.as\_data():

if msg == **'email'**:

messages.error(request, **f"Email-id is invalid"**)

if msg == **'password2'** and password1 == password2:

messages.error(request, **f"Password is not strong enough"**)

elif msg == **'password2'** and password1 != password2:

messages.error(request, **f"Passwords do not match"**)

form = NewUserForm

return render(request, **"main/register.html"**, context = {**"form"** : form})

def logout\_request(request):

logout(request)

messages.info(request, **"Logged out succesfully"**)

return redirect(**"main:homepage"**)

def login\_request(request):

if request.user.is\_authenticated:

return redirect(**'main: homepage'**)

if request.method == **"POST"**:

form = AuthenticationForm(request, data=request.POST)

if form.is\_valid():

username = form.cleaned\_data.get(**'username'**)

password = form.cleaned\_data.get(**'password'**)

user = authenticate(username=username, password=password) *#mentioning username= & password= is necessary*

if user is not None:

login(request, user) *#order of arguements is crucial, pass request first always*

messages.info(request, **f"you are now logged in as :** {username}**"**)

return redirect(**"main:homepage"**)

else:

messages.error(request, **"Invalid username/password"**)

else:

messages.error(request, **"Invalid username/password"**)

form = AuthenticationForm()

return render(

request,

**"main/login.html"**,

{**"form"**:form}

)

def aboutus(request):

return render(request, **"main/aboutus.html"**)

@login\_required(login\_url=**'main:login'**)

def profilepage(request):

profile = request.user.profile

form = ProfileForm(instance=profile)

if request.method == **'POST'**:

form = ProfileForm(request.POST, request.FILES, instance=profile)

if form.is\_valid():

form.save()

return redirect(**'main:profilepage'**)

context = {**'form'**: form}

return render(request, **'main/profile.html'**, context)

1. forms.py

from django import forms

from django.contrib.auth.forms import UserCreationForm

from django.contrib.auth.models import User

from django.forms import ModelForm

from .models import Profile

class NewUserForm(UserCreationForm):

username = forms.CharField(label=**'Enter Username'**, min\_length=4, max\_length=15)

email = forms.EmailField(label=**'Enter email'**,required=True)

password1 = forms.CharField(label=**'Enter password'**, widget=forms.PasswordInput)

password2 = forms.CharField(label=**'Confirm password'**, widget=forms.PasswordInput)

class Meta:

model = User

fields = (**"username"**, **"email"**, **"password1"**, **"password2"**)

def save(self, commit=True):

user = super(NewUserForm, self).save(commit=False)

user.email = self.cleaned\_data[**'email'**]

if commit:

user.save()

return user

class ProfileForm(ModelForm):

class Meta:

model = Profile

fields = **'\_\_all\_\_'**

exclude = [**'user'**]

1. main/urls.py

from django.urls import path

from . import views

app\_name = **"main"**

urlpatterns = [

path(**""**, views.homepage, name = **"homepage"**),

path(**"register/"**, views.register, name=**"register"**),

path(**"logout/"**, views.logout\_request, name=**"logout"**),

path(**"login/"**, views.login\_request, name=**"login"**),

path(**"<single\_slug>"**, views.single\_slug, name=**"single\_slug"**),

path(**"aboutus/"**, views.aboutus, name=**"aboutus"**),

path(**"profilepage/"**, views.profilepage, name=**"profilepage"**)

]

1. admin.py

from django.contrib import admin

from .models import Tutorial, TutorialSeries, TutorialCategory, Profile

from tinymce.widgets import TinyMCE

from django.db import models

*# Register your models here.*

...

class TutorialAdmin(admin.ModelAdmin):

fieldsets = [

(**"Title/date"**, {**'fields'**: [**"tutorial\_title"**, **"tutorial\_published"**]}),

(**"URL"**, {**'fields'**: [**"tutorial\_slug"**]}),

(**"Series"**, {**'fields'**: [**"tutorial\_series"**]}),

(**"Content"**, {**"fields"**: [**"tutorial\_content"**]})

]

formfield\_overrides = {

models.TextField: {**'widget'**: TinyMCE(attrs={**'cols'**: 80, **'rows'**: 30})},

}

admin.site.register(TutorialSeries)

admin.site.register(TutorialCategory)

admin.site.register(Tutorial,TutorialAdmin)

admin.site.register(Profile)

1. apps.py

from django.apps import AppConfig

class MainConfig(AppConfig):

name = **'main'**

1. settings.py

import os

*# Build paths inside the project like this: os.path.join(BASE\_DIR, ...)*

BASE\_DIR = os.path.dirname(os.path.dirname(os.path.abspath(\_\_file\_\_)))

*# Quick-start development settings - unsuitable for production*

*# See https://docs.djangoproject.com/en/2.2/howto/deployment/checklist/*

*# SECURITY WARNING: keep the secret key used in production secret!*

SECRET\_KEY = **'t@$!!m^b4\_t2c2\*s8xmj\*9p1)i)y\_9dy6@qj4gbm&a6l-q8(wu'**

*# 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'**,

**'main.apps.MainConfig'**,

**'tinymce'**,

]

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 = **'mysite.urls'**

TEMPLATES = [

{

**'BACKEND'**: **'django.template.backends.django.DjangoTemplates'**,

**'DIRS'**: [],

**'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 = **'mysite.wsgi.application'**

*# Database*

*# https://docs.djangoproject.com/en/2.2/ref/settings/#databases*

DATABASES = {

**'default'**: {

**'ENGINE'**: **'djongo'**,

**'NAME'**: **'projectdb'**,

**'USER'**:**''**,

**'PASSWORD'**:**''**,

**'HOST'**:**'localhost'**,

**'PORT'**:**'27017'**

}

}

*# Password validation*

*# https://docs.djangoproject.com/en/2.2/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/2.2/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/2.2/howto/static-files/*

STATIC\_URL = **'/static/'**

MEDIA\_URL = **'/images/'**

STATICFILES\_DIRS = [

os.path.join(BASE\_DIR, **'static'**),

]

MEDIA\_ROOT = os.path.join(BASE\_DIR, **'static/images'**)

TINYMCE\_DEFAULT\_CONFIG = {

**'height'**: 360,

**'width'**: 1120,

**'cleanup\_on\_startup'**: True,

**'custom\_undo\_redo\_levels'**: 20,

**'selector'**: **'textarea'**,

**'theme'**: **'modern'**,

**'plugins'**: **'''**

**textcolor save link image media preview codesample contextmenu**

**table code lists fullscreen insertdatetime nonbreaking**

**contextmenu directionality searchreplace wordcount visualblocks**

**visualchars code fullscreen autolink lists charmap print hr**

**anchor pagebreak**

**'''**,

**'toolbar1'**: **'''**

**fullscreen preview bold italic underline | fontselect,**

**fontsizeselect | forecolor backcolor | alignleft alignright |**

**aligncenter alignjustify | indent outdent | bullist numlist table |**

**| link image media | codesample |**

**'''**,

**'toolbar2'**: **'''**

**visualblocks visualchars |**

**charmap hr pagebreak nonbreaking anchor | code |**

**'''**,

**'contextmenu'**: **'formats | link image'**,

**'menubar'**: True,

**'statusbar'**: True,

}

1. mysite/urls.py

from django.contrib import admin

from django.urls import path, include

from django.conf import settings

from django.conf.urls.static import static

urlpatterns = [

path(**''**, include(**'main.urls'**)),

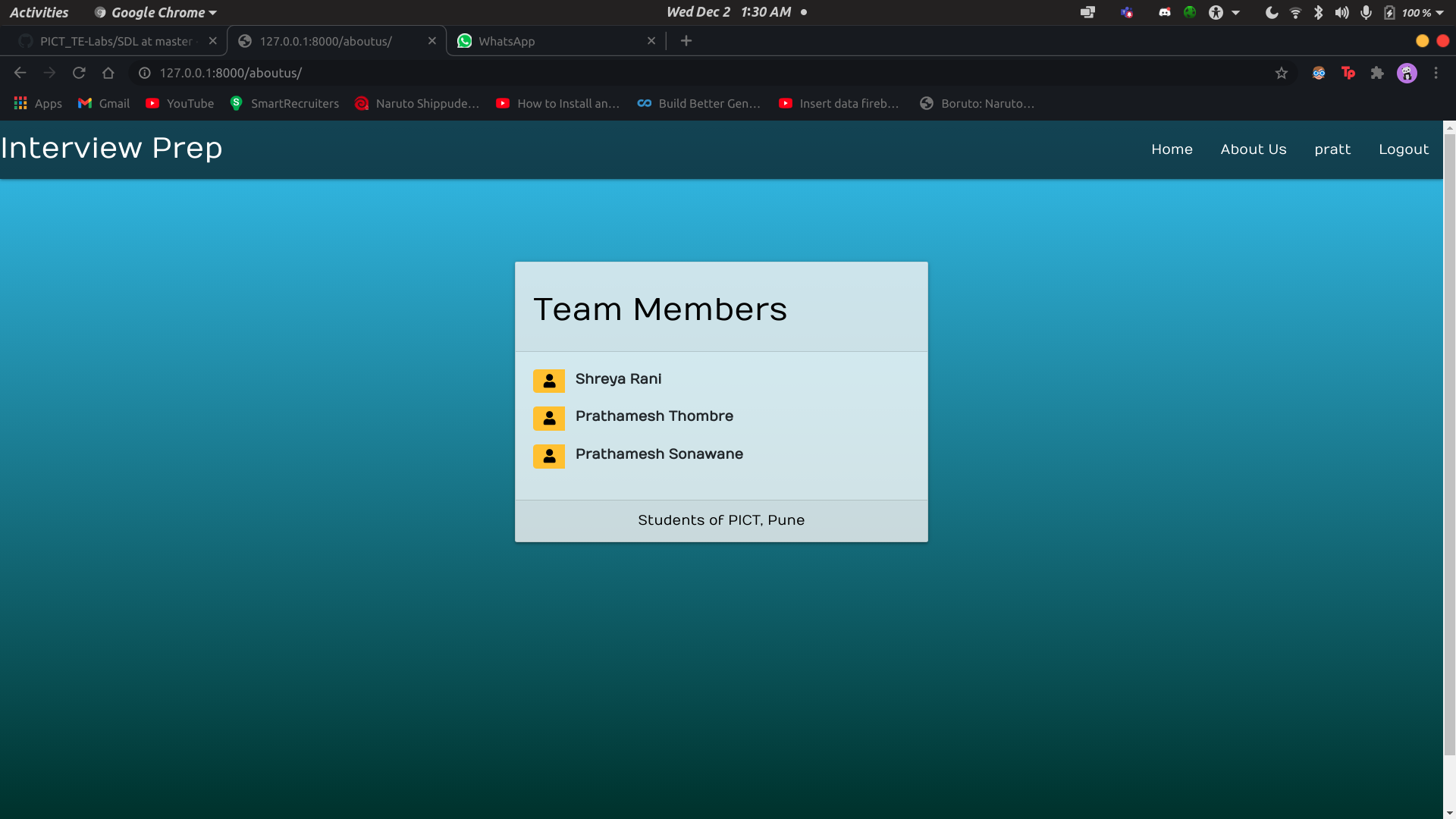
path(**'admin/'**, admin.site.urls),

path(**'tinymce'**, include(**'tinymce.urls'**)),

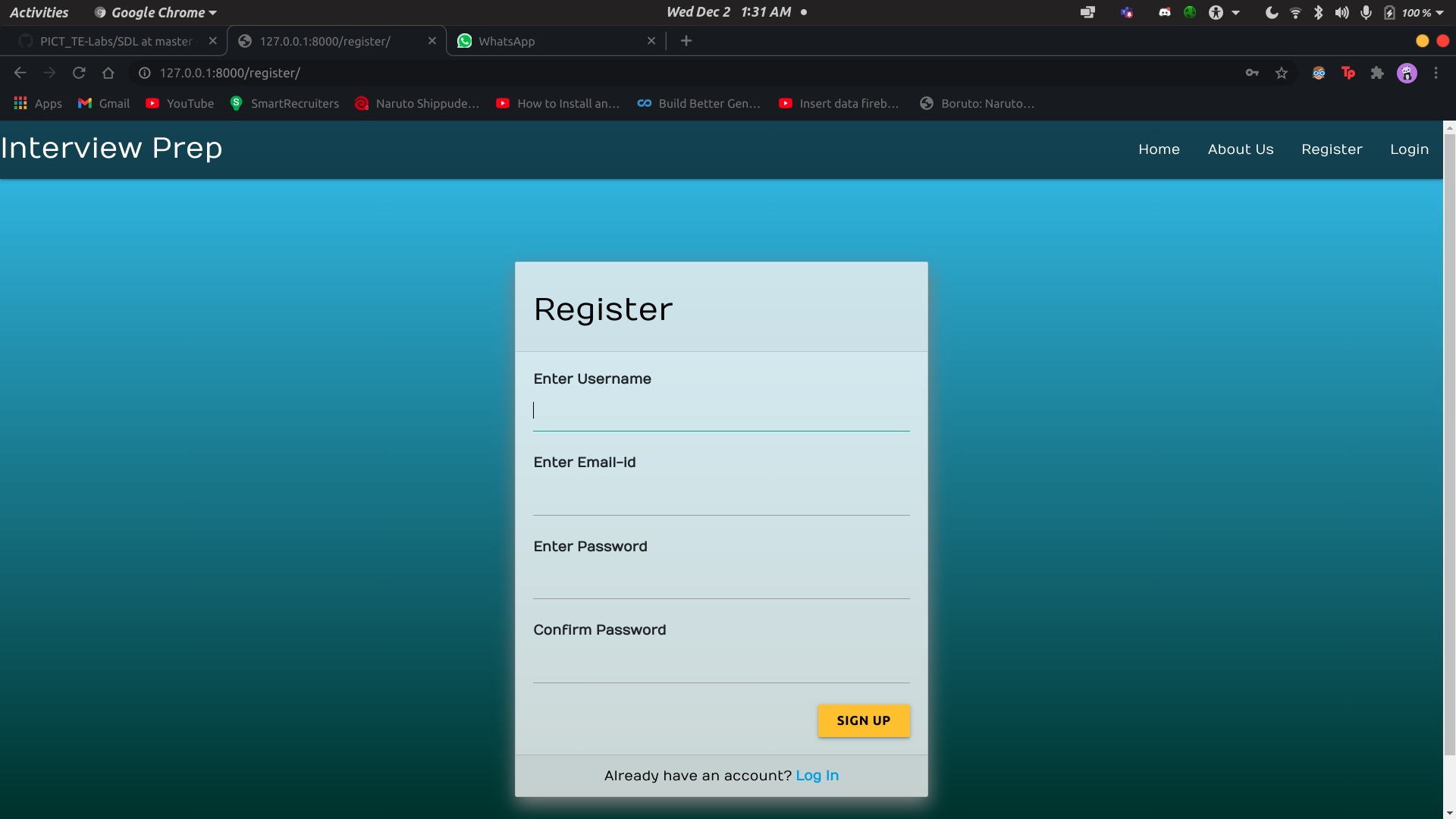
]+ static(settings.MEDIA\_URL, document\_root=settings.MEDIA\_ROOT)

**OUTPUT**

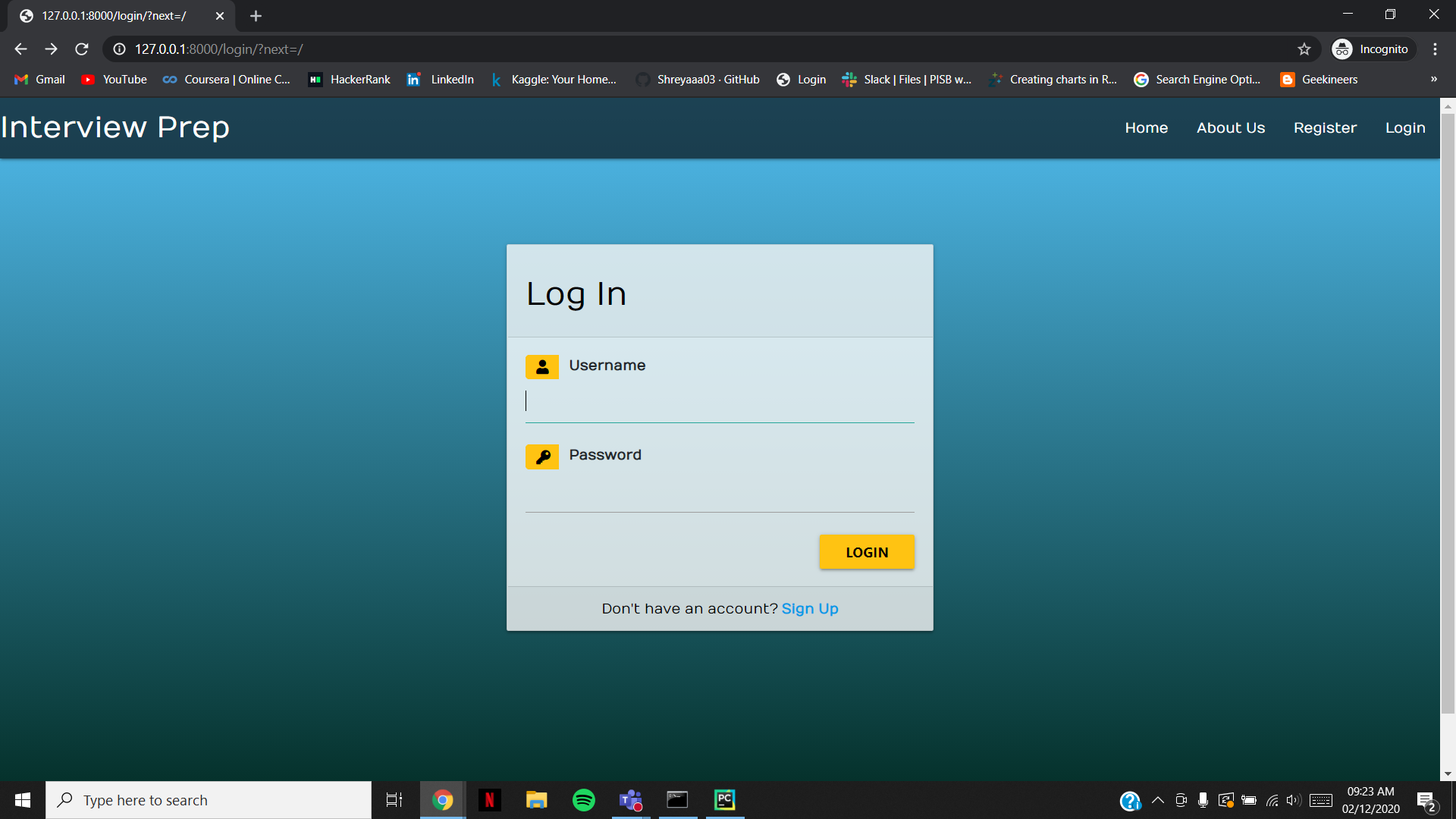
1. About us page



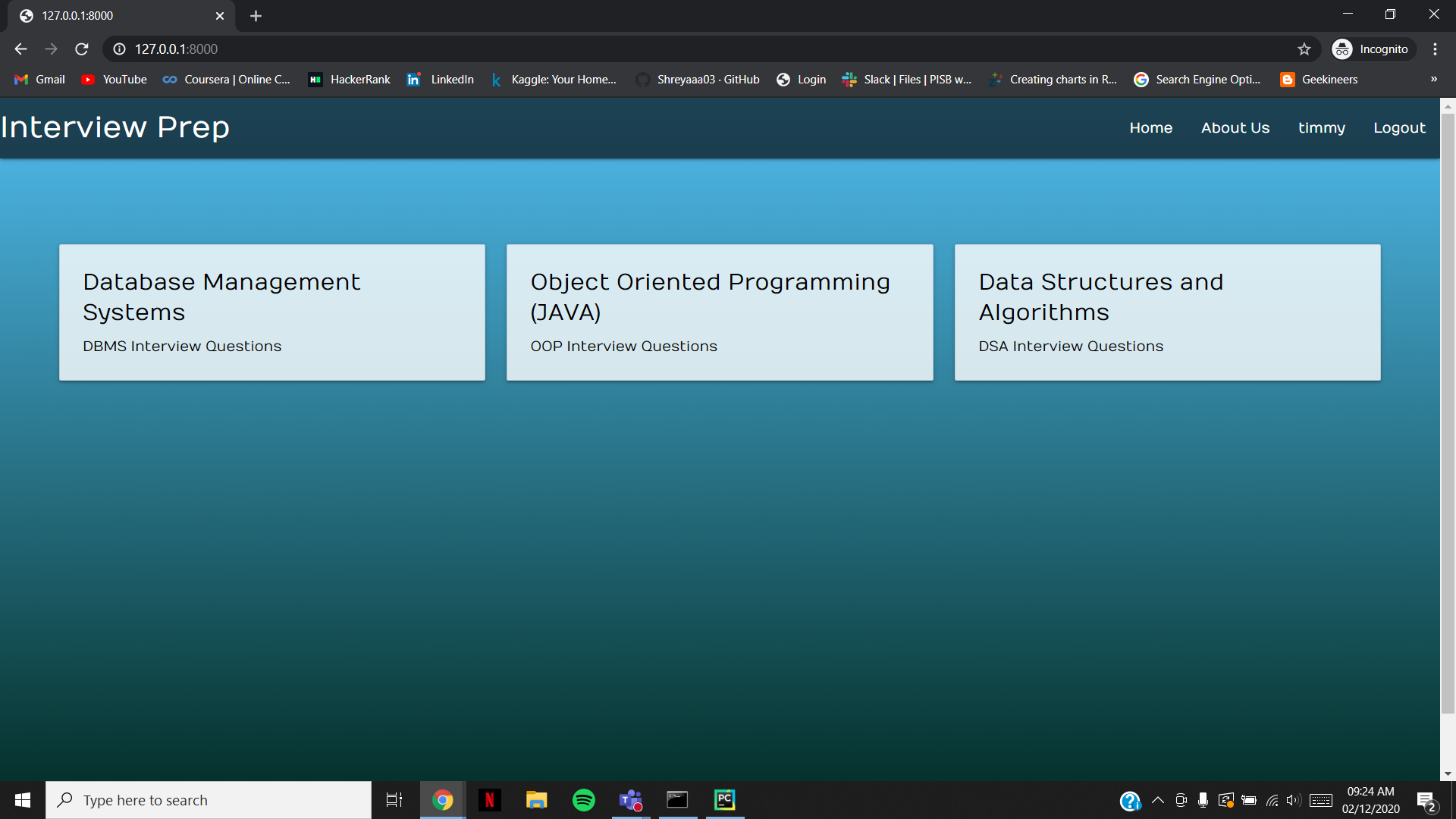
2. Register Page

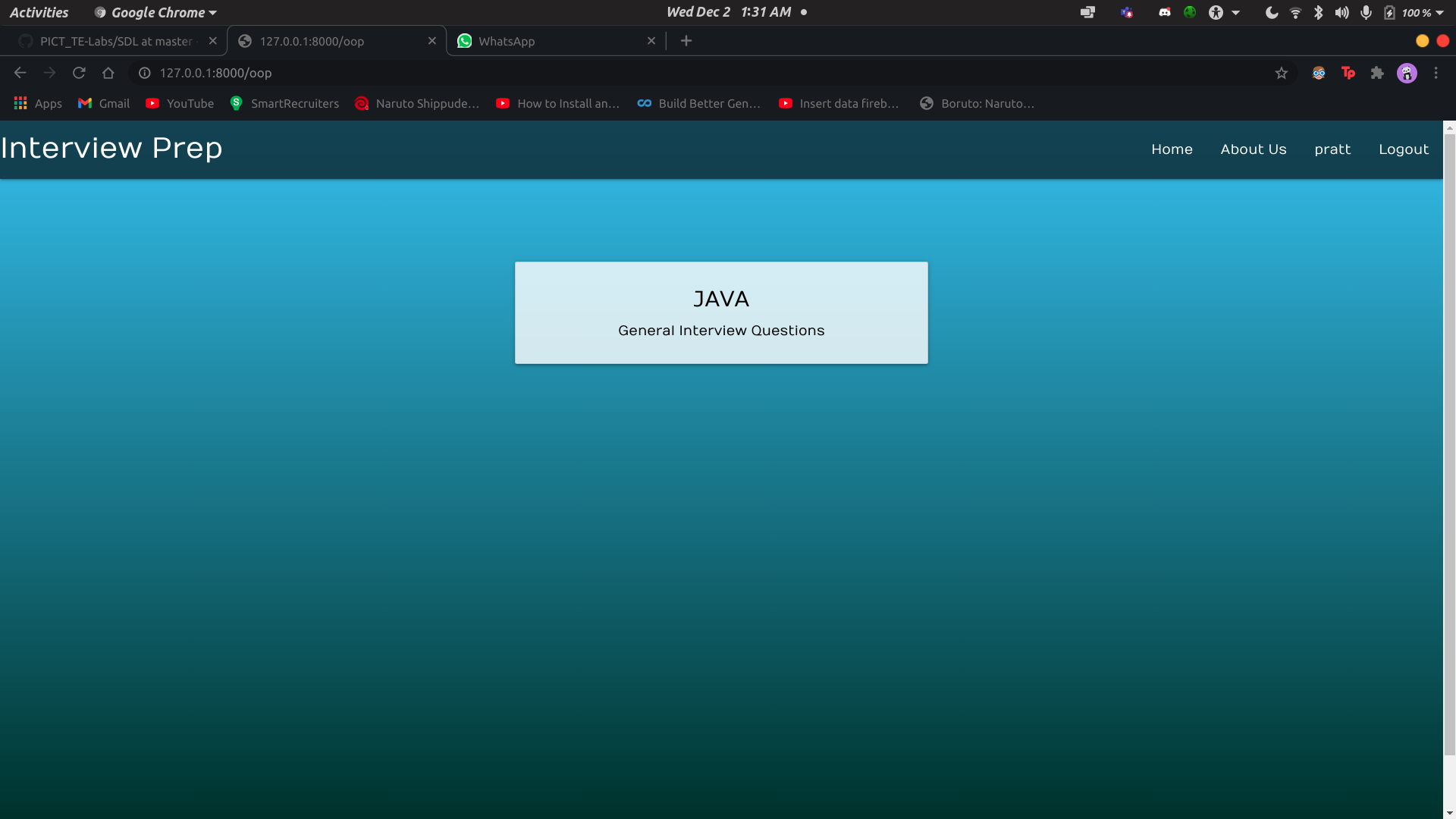


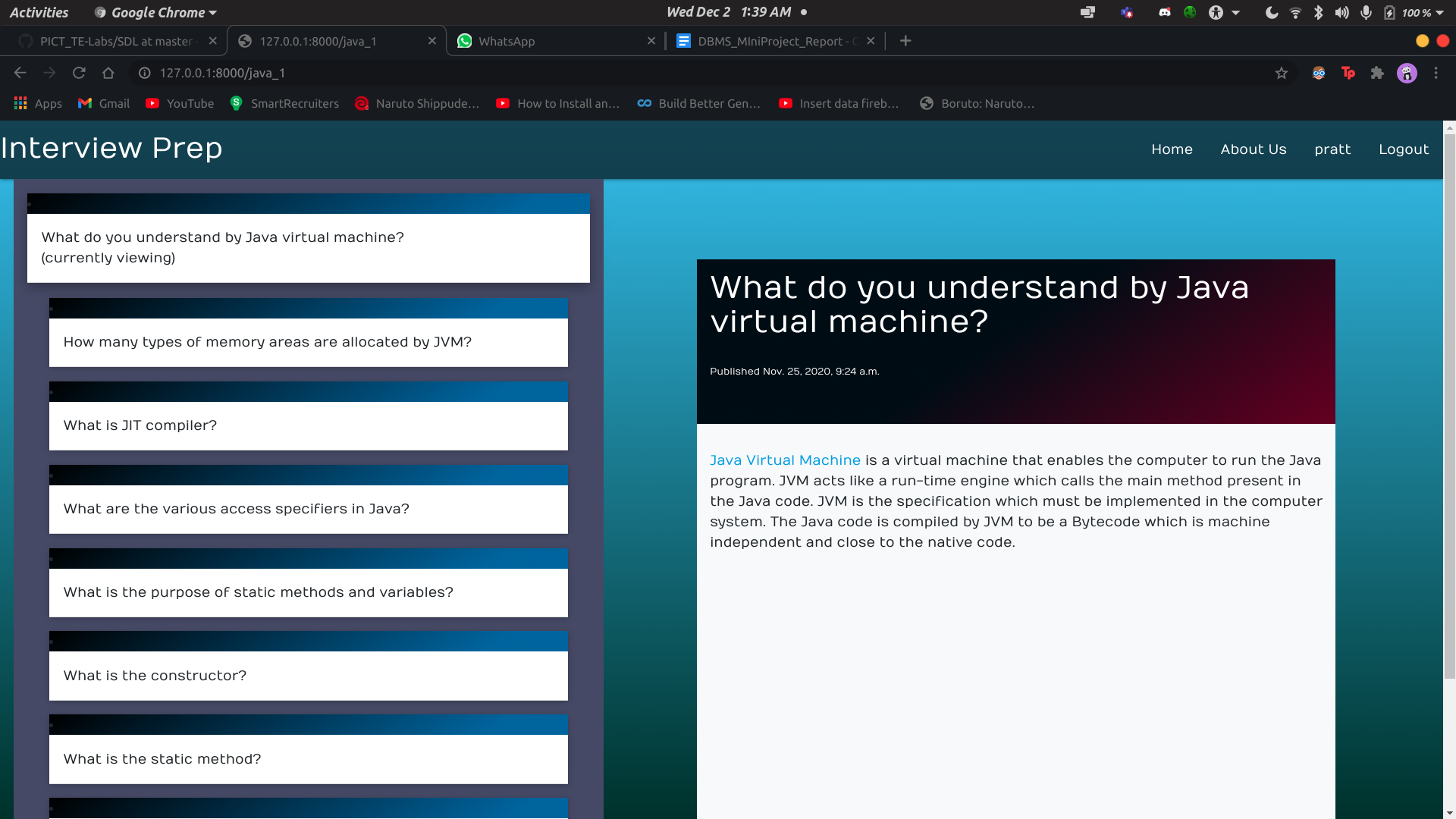
3. Login Page



4. Home Page



5. Sub-topic page  
  


6. Questions Page  
  


7. Profile Settings Page

