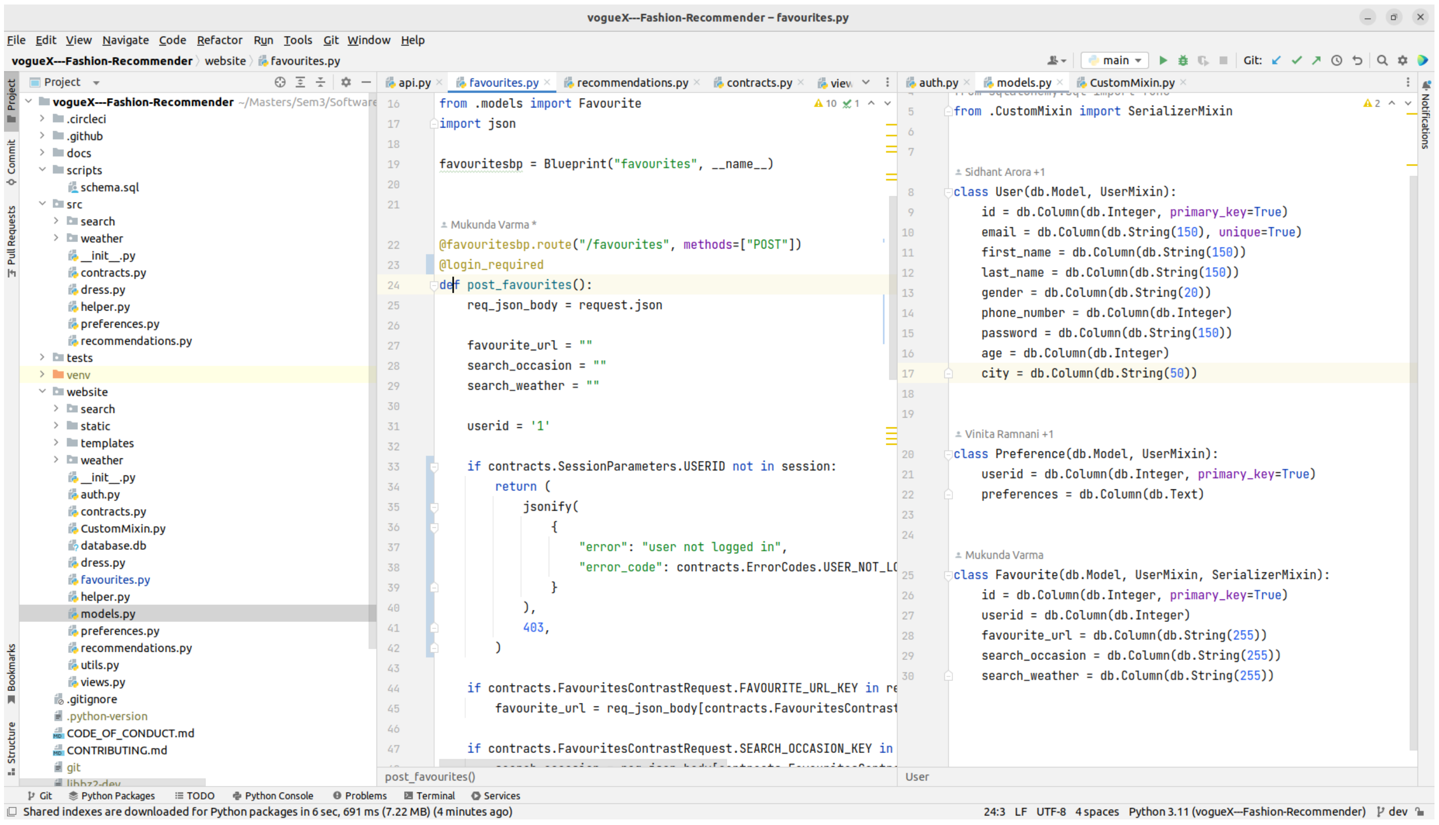
1.Mukund

Pycharm

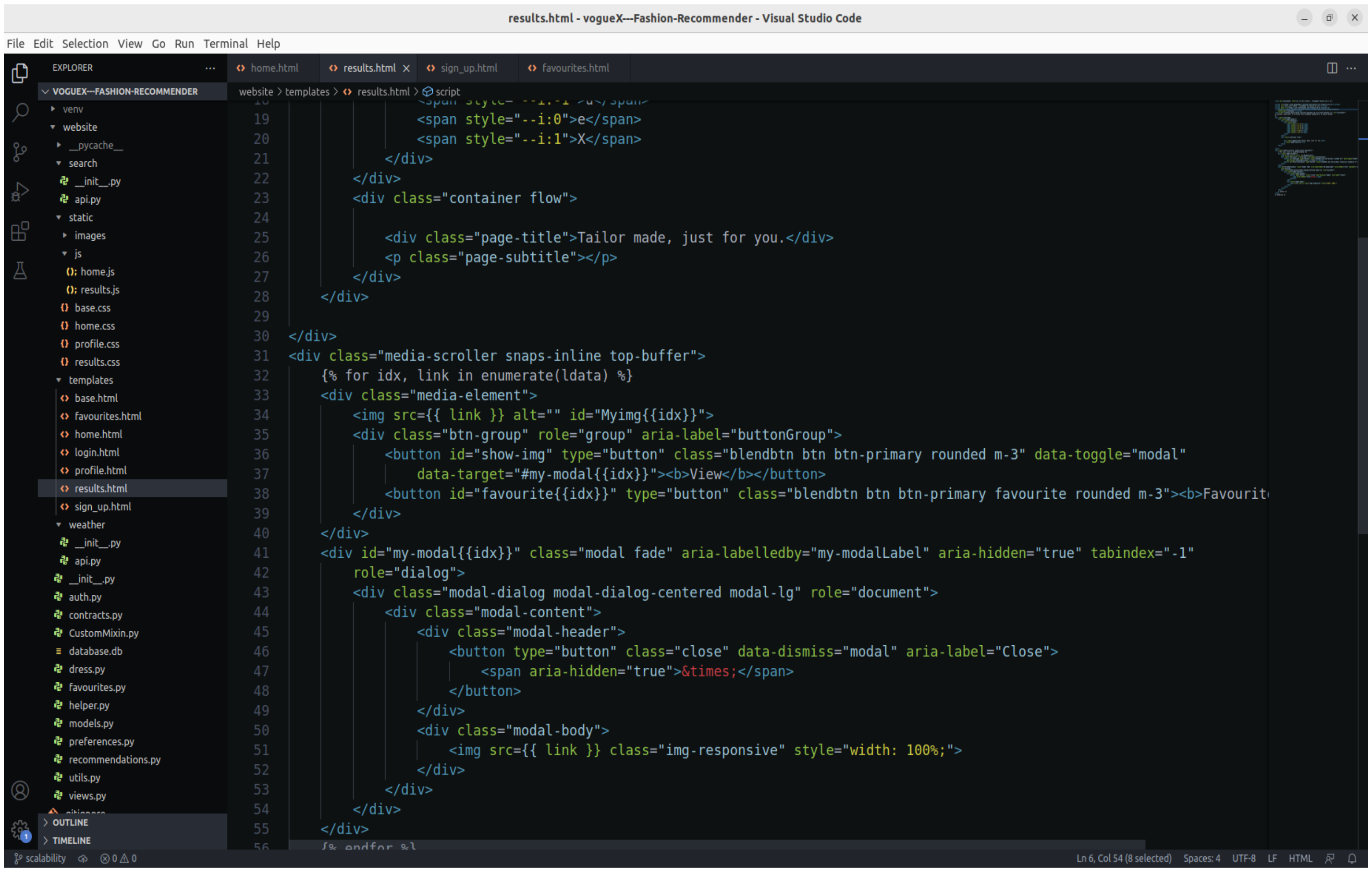


```python
from .models import Favourite
import json

favouritesbp = Blueprint("favourites", __name__)


@favouritesbp.route("/favourites", methods=["POST"])
@login_required
def post_favourites():
    req_json_body = request.json

    favourite_url = ""
    search_occasion = ""
    search_weather = ""

    userid = '1'

    if contracts.SessionParameters.USERID not in session:
        return (
            jsonify(
                {
                    "error": "user not logged in",
                    "error_code": contracts.ErrorCodes.USER_NOT_LO
                }
            ),
            403,
        )

    if contracts.FavouritesContrastRequest.FAVOURITE_URL_KEY in re
        favourite_url = req_json_body[contracts.FavouritesContrast

    if contracts.FavouritesContrastRequest.SEARCH_OCCASION_KEY in
```

```python
from .CustomMixin import SerializerMixin


class User(db.Model, UserMixin):
    id = db.Column(db.Integer, primary_key=True)
    email = db.Column(db.String(150), unique=True)
    first_name = db.Column(db.String(150))
    last_name = db.Column(db.String(150))
    gender = db.Column(db.String(20))
    phone_number = db.Column(db.Integer)
    password = db.Column(db.String(150))
    age = db.Column(db.Integer)
    city = db.Column(db.String(50))


class Preference(db.Model, UserMixin):
    userid = db.Column(db.Integer, primary_key=True)
    preferences = db.Column(db.Text)


class Favourite(db.Model, UserMixin, SerializerMixin):
    id = db.Column(db.Integer, primary_key=True)
    userid = db.Column(db.Integer)
    favourite_url = db.Column(db.String(255))
    search_occasion = db.Column(db.String(255))
    search_weather = db.Column(db.String(255))
```

Shared indexes are downloaded for Python packages in 6 sec, 691 ms (7.22 MB) (4 minutes ago)

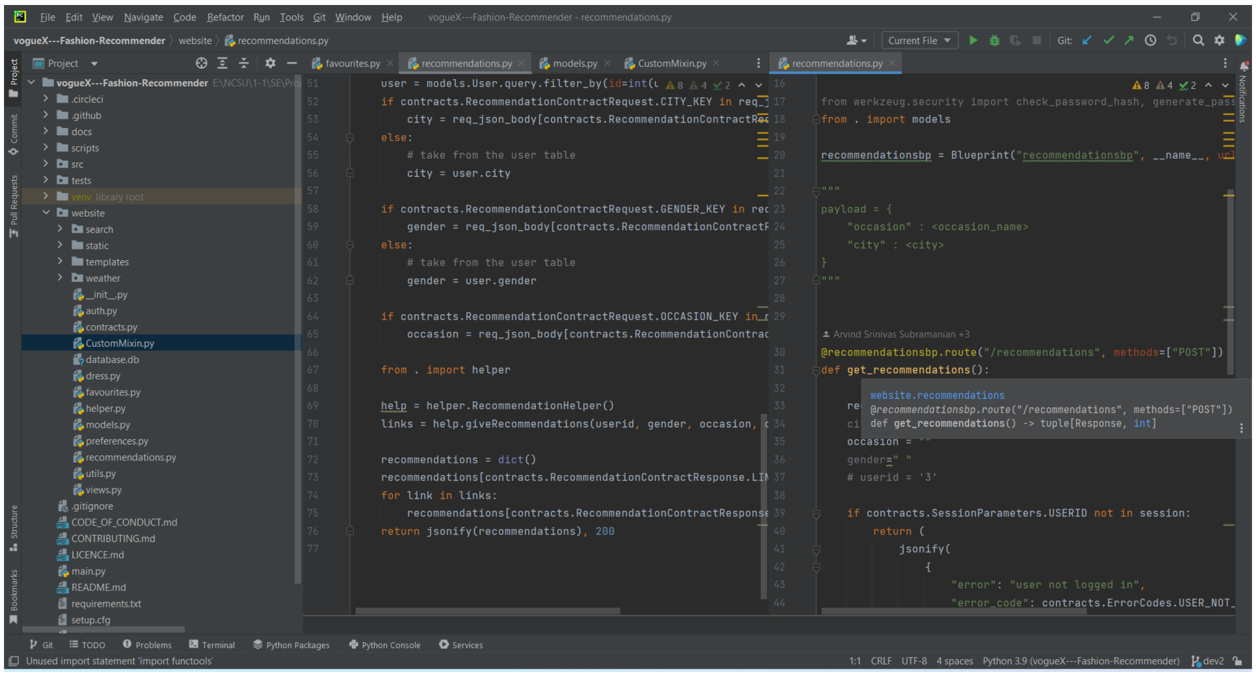
VS Code



```
                <span style="--i:0">e</span>
                <span style="--i:1">X</span>
            </div>
        </div>
        <div class="container flow">

            <div class="page-title">Tailor made, just for you.</div>
            <p class="page-subtitle"></p>
        </div>
    </div>

</div>
<div class="media-scroller snaps-inline top-buffer">
    {% for idx, link in enumerate(ldata) %}
    <div class="media-element">
        <img src={{ link }} alt="" id="Myimg{{idx}}">
        <div class="btn-group" role="group" aria-label="buttonGroup">
            <button id="show-img" type="button" class="blendbtn btn btn-primary rounded m-3" data-toggle="modal"
                data-target="#my-modal{{idx}}"><b>View</b></button>
            <button id="favourite{{idx}}" type="button" class="blendbtn btn btn-primary favourite rounded m-3"><b>Favourit
        </div>
    </div>
    <div id="my-modal{{idx}}" class="modal fade" aria-labelledby="my-modalLabel" aria-hidden="true" tabindex="-1"
        role="dialog">
        <div class="modal-dialog modal-dialog-centered modal-lg" role="document">
            <div class="modal-content">
                <div class="modal-header">
                    <button type="button" class="close" data-dismiss="modal" aria-label="Close">
                        <span aria-hidden="true">&times;</span>
                    </button>
                </div>
                <div class="modal-body">
                    <img src={{ link }} class="img-responsive" style="width: 100%;">
                </div>
            </div>
        </div>
    </div>
    {% endfor %}
```

2. Saketh

Pycharm



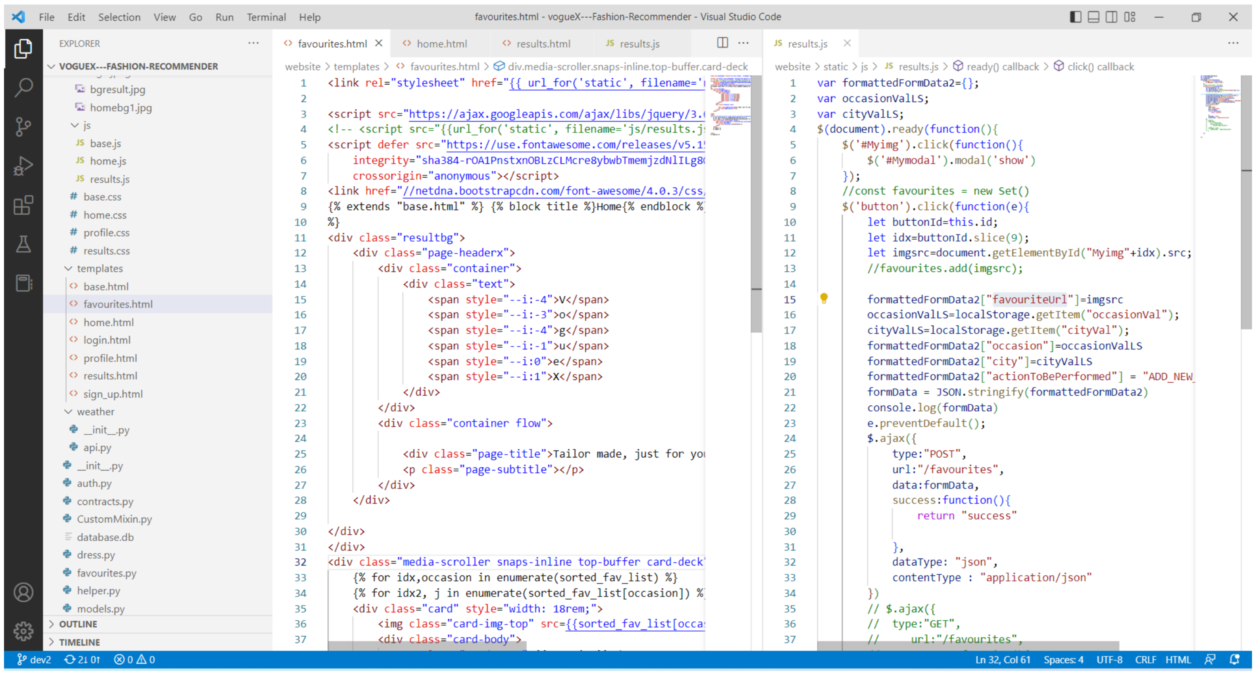
```python
    user = models.User.query.filter_by(id=int(u
    if contracts.RecommendationContractRequest.CITY_KEY in req_j
        city = req_json_body[contracts.RecommendationContractReq
    else:
        # take from the user table
        city = user.city

    if contracts.RecommendationContractRequest.GENDER_KEY in req
        gender = req_json_body[contracts.RecommendationContractR
    else:
        # take from the user table
        gender = user.gender

    if contracts.RecommendationContractRequest.OCCASION_KEY in_r
        occasion = req_json_body[contracts.RecommendationContrac

    from . import helper

    help = helper.RecommendationHelper()
    links = help.giveRecommendations(userid, gender, occasion, c

    recommendations = dict()
    recommendations[contracts.RecommendationContractResponse.LIN
    for link in links:
        recommendations[contracts.RecommendationContractResponse
    return jsonify(recommendations), 200
```

```python
from werkzeug.security import check_password_hash, generate_pass
from . import models

recommendationsbp = Blueprint("recommendationsbp", __name__, url

"""
payload = {
    "occasion" : <occasion_name>
    "city" : <city>
}
"""


@recommendationsbp.route("/recommendations", methods=["POST"])
def get_recommendations():

    re
    ci
    occasion = ""
    gender=" "
    # userid = '3'

    if contracts.SessionParameters.USERID not in session:
        return (
            jsonify(
                {
                    "error": "user not logged in",
                    "error_code": contracts.ErrorCodes.USER_NOT_
```

```
website.recommendations
@recommendationsbp.route("/recommendations", methods=["POST"])
def get_recommendations() -> tuple[Response, int]
```

Unused import statement 'import functools'

VSCode

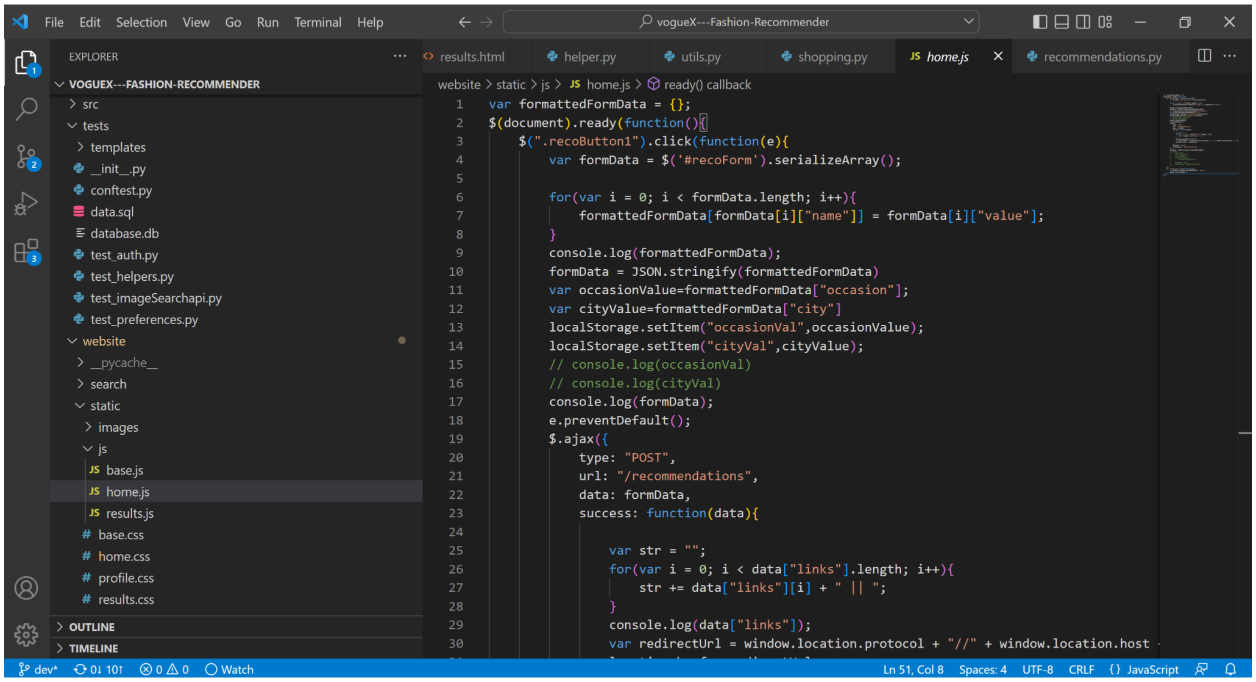


```html
<link rel="stylesheet" href="{{ url_for('static', filename='r

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.(
<!-- <script src="{{url_for('static', filename='js/results.js
<script defer src="https://use.fontawesome.com/releases/v5.15
    integrity="sha384-rOA1PnstxnOBLzCLMcre8ybwbTmemjzdNlILg8(
    crossorigin="anonymous"></script>
<link href="//netdna.bootstrapcdn.com/font-awesome/4.0.3/css/
{% extends "base.html" %} {% block title %}Home{% endblock %
%}
<div class="resultbg">
    <div class="page-headerx">
        <div class="container">
            <div class="text">
                <span style="--i:-4">V</span>
                <span style="--i:-3">o</span>
                <span style="--i:-4">g</span>
                <span style="--i:-1">u</span>
                <span style="--i:0">e</span>
                <span style="--i:1">X</span>
            </div>
        </div>
        <div class="container flow">

            <div class="page-title">Tailor made, just for you
            <p class="page-subtitle"></p>
        </div>
    </div>

</div>
</div>
<div class="media-scroller snaps-inline top-buffer card-deck'
    {% for idx,occasion in enumerate(sorted_fav_list) %}
    {% for idx2, j in enumerate(sorted_fav_list[occasion]) %
    <div class="card" style="width: 18rem;">
        <img class="card-img-top" src={{sorted_fav_list[occas
        <div class="card-body">
```

```javascript
var formattedFormData2={};
var occasionValLS;
var cityValLS;
$(document).ready(function(){
    $('#Myimg').click(function(){
        $('#Mymodal').modal('show')
    });
    //const favourites = new Set()
    $('button').click(function(e){
        let buttonId=this.id;
        let idx=buttonId.slice(9);
        let imgsrc=document.getElementById("Myimg"+idx).src;
        //favourites.add(imgsrc);

        formattedFormData2["favouriteUrl"]=imgsrc
        occasionValLS=localStorage.getItem("occasionVal");
        cityValLS=localStorage.getItem("cityVal");
        formattedFormData2["occasion"]=occasionValLS
        formattedFormData2["city"]=cityValLS
        formattedFormData2["actionToBePerformed"] = "ADD_NEW_
        formData = JSON.stringify(formattedFormData2)
        console.log(formData)
        e.preventDefault();
        $.ajax({
            type:"POST",
            url:"/favourites",
            data:formData,
            success:function(){
                return "success"

            },
            dataType: "json",
            contentType : "application/json"
        })
        // $.ajax({
        //  type:"GET",
        //     url:"/favourites",
```

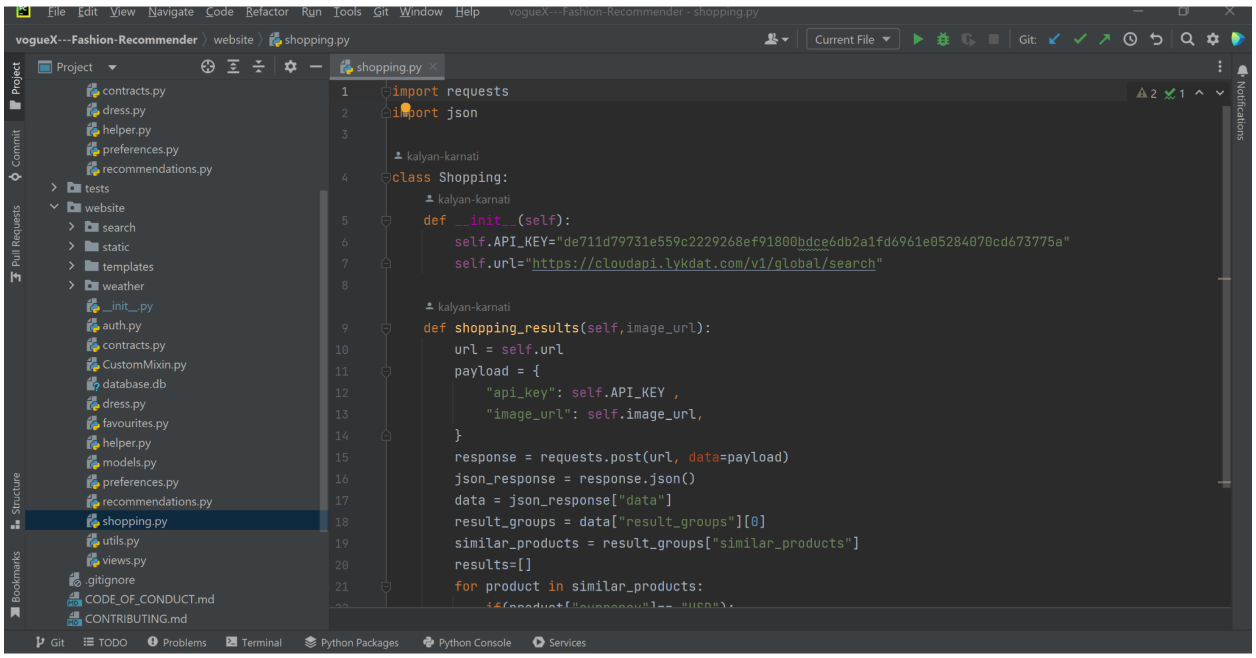
3.Kalyan

VS Code



```
var formattedFormData = {};
$(document).ready(function(){
    $(".recoButton1").click(function(e){
        var formData = $('#recoForm').serializeArray();

        for(var i = 0; i < formData.length; i++){
            formattedFormData[formData[i]["name"]] = formData[i]["value"];
        }
        console.log(formattedFormData);
        formData = JSON.stringify(formattedFormData)
        var occasionValue=formattedFormData["occasion"];
        var cityValue=formattedFormData["city"]
        localStorage.setItem("occasionVal",occasionValue);
        localStorage.setItem("cityVal",cityValue);
        // console.log(occasionVal)
        // console.log(cityVal)
        console.log(formData);
        e.preventDefault();
        $.ajax({
            type: "POST",
            url: "/recommendations",
            data: formData,
            success: function(data){

                var str = "";
                for(var i = 0; i < data["links"].length; i++){
                    str += data["links"][i] + " || ";
                }
                console.log(data["links"]);
                var redirectUrl = window.location.protocol + "//" + window.location.host
```

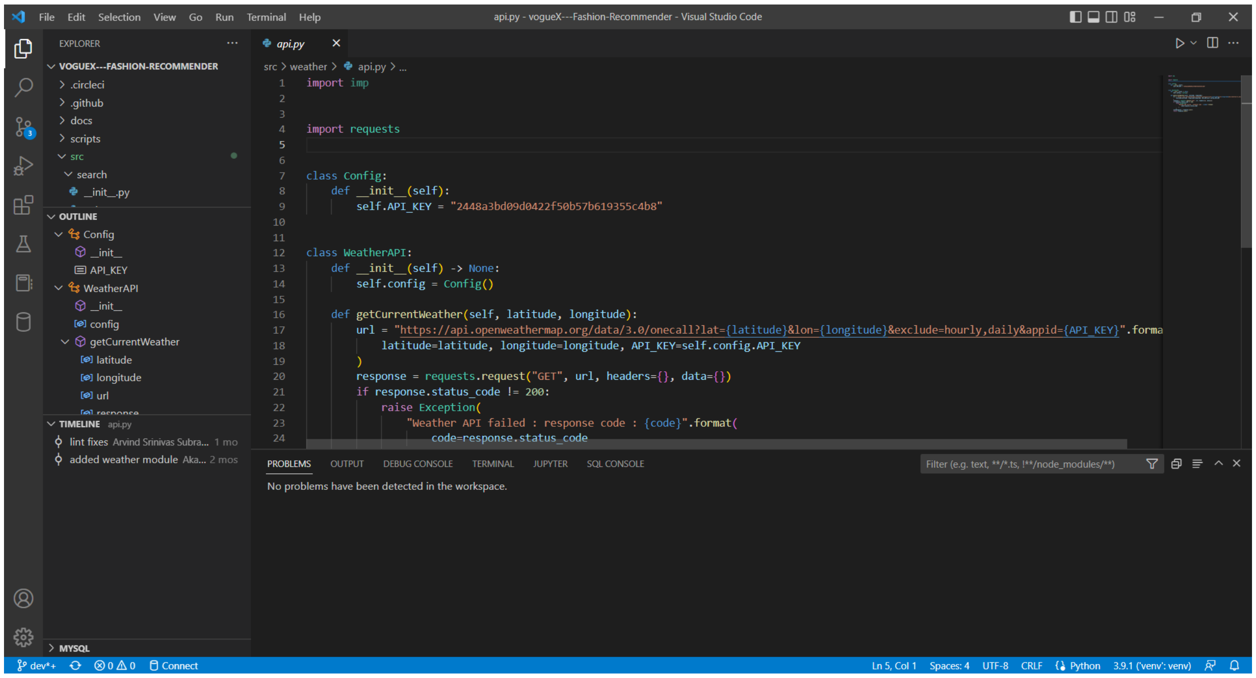
Pycharm



```
import requests
import json

class Shopping:
    def __init__(self):
        self.API_KEY="de711d79731e559c2229268ef91800bdce6db2a1fd6961e05284070cd673775a"
        self.url="https://cloudapi.lykdat.com/v1/global/search"

    def shopping_results(self,image_url):
        url = self.url
        payload = {
            "api_key": self.API_KEY ,
            "image_url": self.image_url,
        }
        response = requests.post(url, data=payload)
        json_response = response.json()
        data = json_response["data"]
        result_groups = data["result_groups"][0]
        similar_products = result_groups["similar_products"]
        results=[]
        for product in similar_products:
```

4. Pranavi

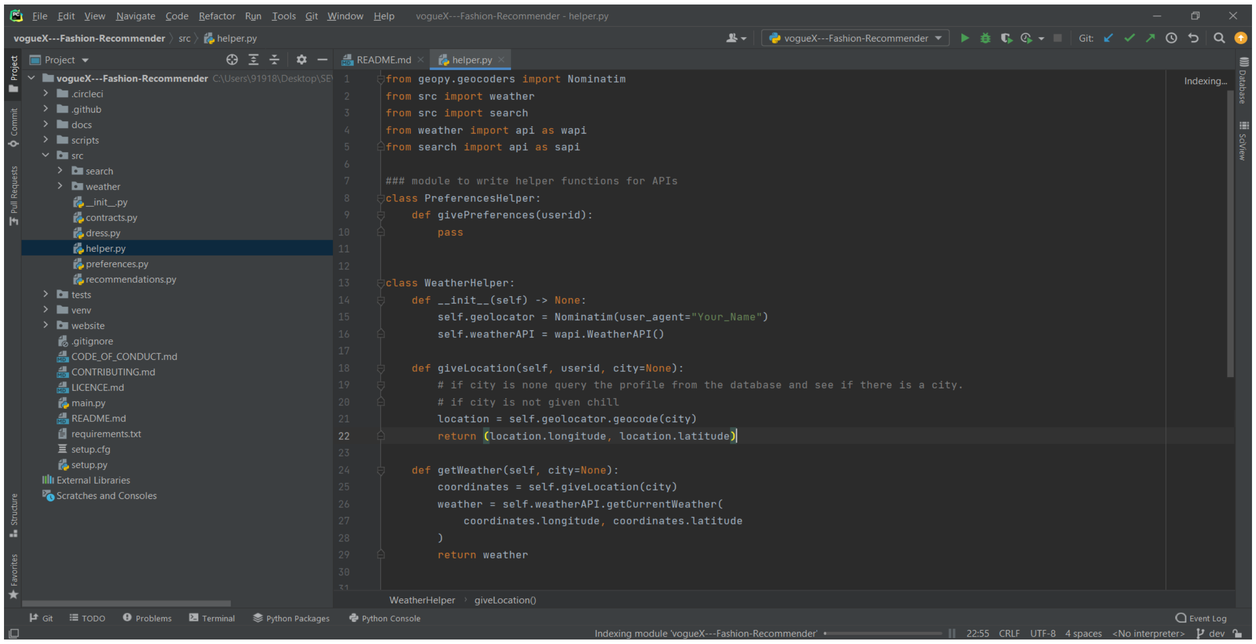
VS Code



```python
import imp


import requests


class Config:
    def __init__(self):
        self.API_KEY = "2448a3bd09d0422f50b57b619355c4b8"


class WeatherAPI:
    def __init__(self) -> None:
        self.config = Config()

    def getCurrentWeather(self, latitude, longitude):
        url = "https://api.openweathermap.org/data/3.0/onecall?lat={latitude}&lon={longitude}&exclude=hourly,daily&appid={API_KEY}".forma
            latitude=latitude, longitude=longitude, API_KEY=self.config.API_KEY
        )
        response = requests.request("GET", url, headers={}, data={})
        if response.status_code != 200:
            raise Exception(
                "Weather API failed : response code : {code}".format(
                    code=response.status_code
```

```
No problems have been detected in the workspace.
```

Pycharm

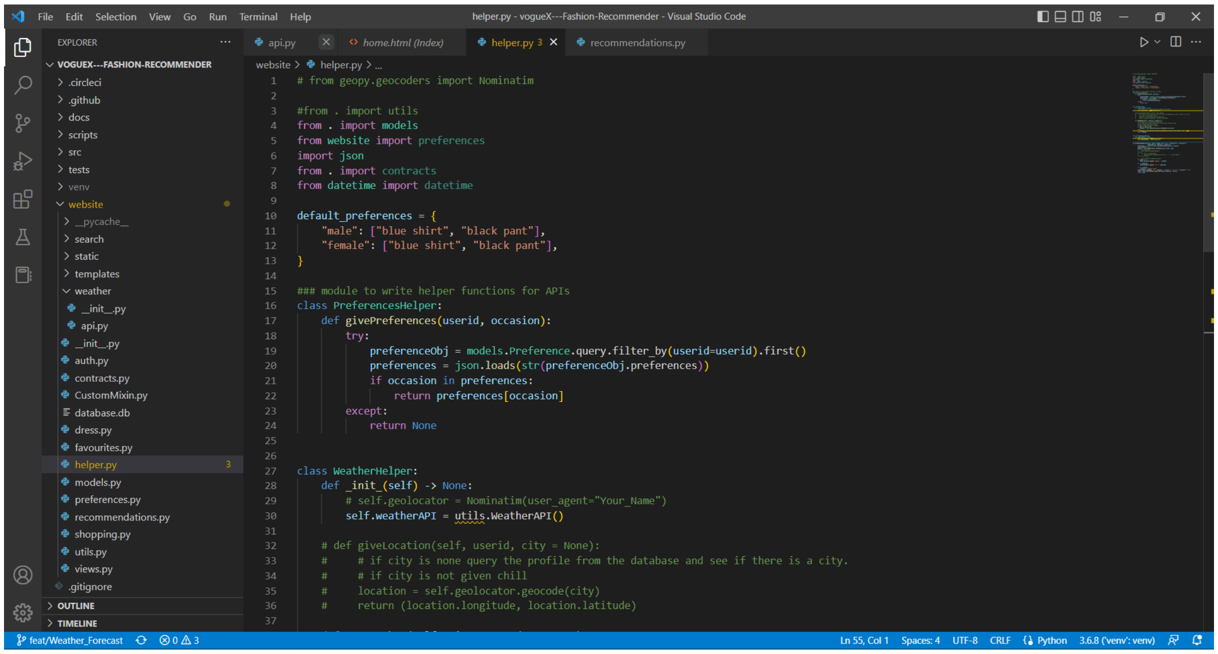


```python
from geopy.geocoders import Nominatim
from src import weather
from src import search
from weather import api as wapi
from search import api as sapi

### module to write helper functions for APIs
class PreferencesHelper:
    def givePreferences(userid):
        pass


class WeatherHelper:
    def __init__(self) -> None:
        self.geolocator = Nominatim(user_agent="Your_Name")
        self.weatherAPI = wapi.WeatherAPI()

    def giveLocation(self, userid, city=None):
        # if city is none query the profile from the database and see if there is a city.
        # if city is not given chill
        location = self.geolocator.geocode(city)
        return (location.longitude, location.latitude)

    def getWeather(self, city=None):
        coordinates = self.giveLocation(city)
        weather = self.weatherAPI.getCurrentWeather(
            coordinates.longitude, coordinates.latitude
        )
        return weather
```

```
Indexing module 'vogueX---Fashion-Recommender'
```

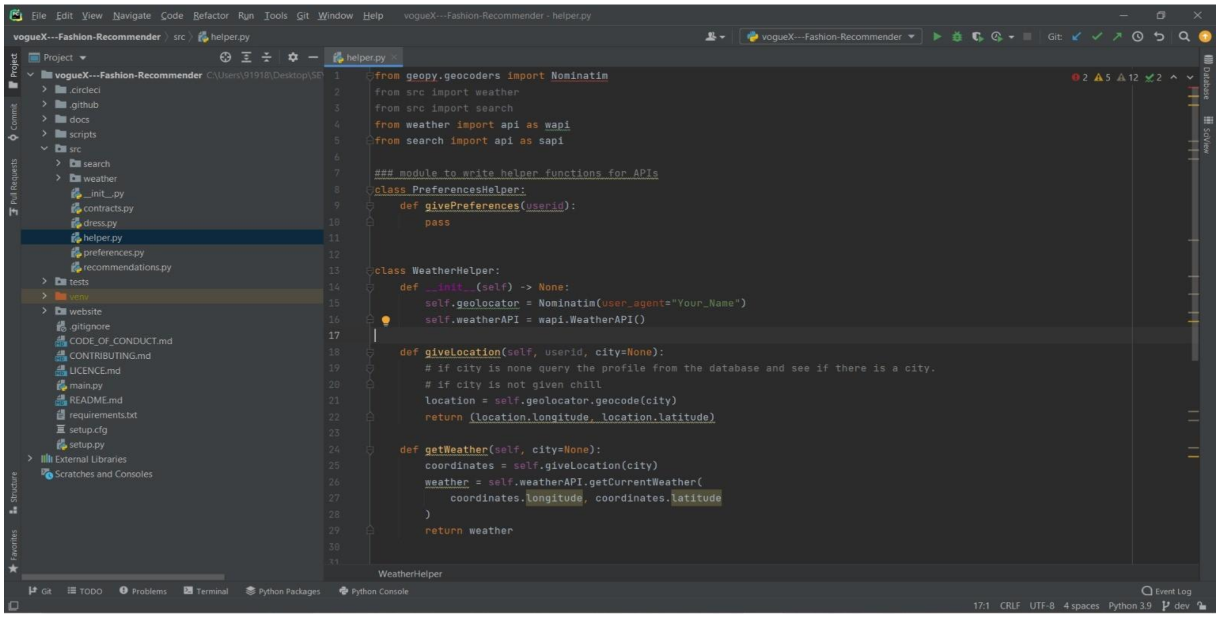
5. Srihitha

VS Code



```python
# from geopy.geocoders import Nominatim

#from . import utils
from . import models
from website import preferences
import json
from . import contracts
from datetime import datetime

default_preferences = {
    "male": ["blue shirt", "black pant"],
    "female": ["blue shirt", "black pant"],
}

### module to write helper functions for APIs
class PreferencesHelper:
    def givePreferences(userid, occasion):
        try:
            preferenceObj = models.Preference.query.filter_by(userid=userid).first()
            preferences = json.loads(str(preferenceObj.preferences))
            if occasion in preferences:
                return preferences[occasion]
        except:
            return None


class WeatherHelper:
    def _init_(self) -> None:
        # self.geolocator = Nominatim(user_agent="Your_Name")
        self.weatherAPI = utils.WeatherAPI()

    # def giveLocation(self, userid, city = None):
    #     # if city is none query the profile from the database and see if there is a city.
    #     # if city is not given chill
    #     location = self.geolocator.geocode(city)
    #     return (location.longitude, location.latitude)
```

Pycharm



```python
from geopy.geocoders import Nominatim
from src import weather
from src import search
from weather import api as wapi
from search import api as sapi

### module to write helper functions for APIs
class PreferencesHelper:
    def givePreferences(userid):
        pass


class WeatherHelper:
    def __init__(self) -> None:
        self.geolocator = Nominatim(user_agent="Your_Name")
        self.weatherAPI = wapi.WeatherAPI()

    def giveLocation(self, userid, city=None):
        # if city is none query the profile from the database and see if there is a city.
        # if city is not given chill
        location = self.geolocator.geocode(city)
        return (location.longitude, location.latitude)

    def getWeather(self, city=None):
        coordinates = self.giveLocation(city)
        weather = self.weatherAPI.getCurrentWeather(
            coordinates.longitude, coordinates.latitude
        )
        return weather
```