

# INTRODUCTION

## OVERVIEW:

Chatbots are conversational tools that perform routine tasks efficiently. People like them because they help them get through those tasks quickly so they can focus their attention on high-level, strategic, and engaging activities that require human capabilities that cannot be replicated by machines.

## PURPOSE:

The purpose of public welfare chatbot is to enhance public access to essential services and information, promote efficiency in government operations, and improve the well-being of individuals and communities.

## PROPOSED SOLUTION:

The proposed solution involves leveraging IBM Watson Assistant in combination with Python to develop an intelligent chatbot. It will be designed to assist users with inquiries, provide product information, and route complex issues to human agents. The chatbot will use Watson Assistant's natural language processing capabilities to understand user intents and entities, and Python will facilitate seamless communication between users and the chatbot. The solution includes a robust escalation mechanism to transfer users to human agents when necessary, and it will be continuously improved through testing, training, and monitoring for optimal customer support.

# PROJECT CREATION , PATH SETUP AND APP

## CREATION IN COMMAND PROMPT

- 1.Create a New folder with a name
- 2.Get into folder and open command prompt
- 3.Django installation

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.10586]
(c) 2015 Microsoft Corporation. All rights reserved.

C:\Users\user\Desktop\Chatbot>pip install django
Requirement already satisfied: django in c:\users\user\appdata\local\programs\python\python311\lib\site-packages (4.2.2)
Requirement already satisfied: asgiref<4,>=3.6.0 in c:\users\user\appdata\local\programs\python\python311\lib\site-packages (from django) (3.7.2)
Requirement already satisfied: sqlparse>=0.3.1 in c:\users\user\appdata\local\programs\python\python311\lib\site-packages (from django) (0.4.4)
Requirement already satisfied: tzdata in c:\users\user\appdata\local\programs\python\python311\lib\site-packages (from django) (2023.3)
```

- 4.Project creation

```
C:\Users\user\Desktop\Chatbot>django-admin startproject mychatbot
```

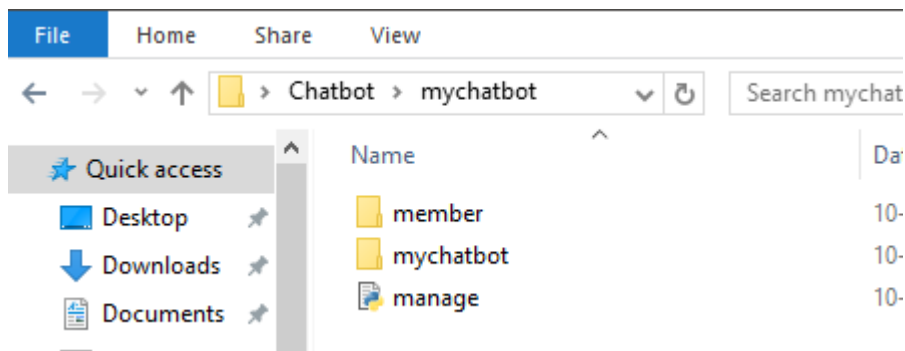
- 5.Path setup

```
C:\Users\user\Desktop\Chatbot>cd mychatbot
C:\Users\user\Desktop\Chatbot\mychatbot>
```

- 6.App creation

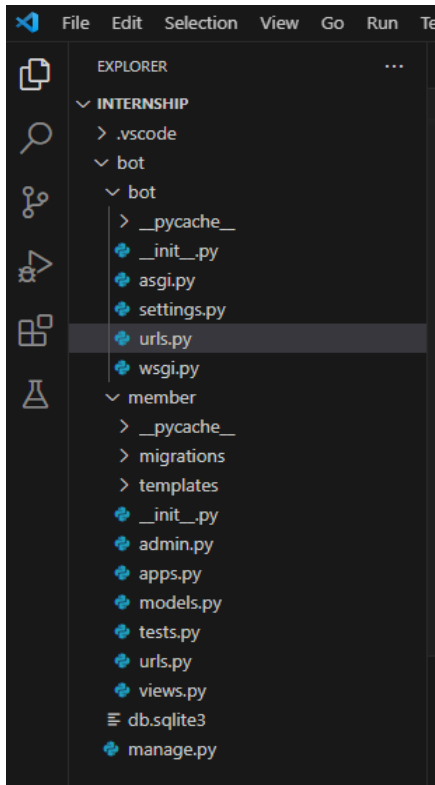
```
C:\Users\user\Desktop\Chatbot\mychatbot>django-admin startapp member
```

- 7.Check app and project are created inside the folder



## IMPORTING INCLUDE() FUNCTION AND ADDING URLPATTERNS

1. Open visual studio code and check whether these are created inside the folder



2. Import the function and add urlpatterns in the project urls.py

```
from django.contrib import admin
from django.urls import include, path

urlpatterns = [
    path('', include('member.urls')),
    path('admin/', admin.site.urls),
]
```

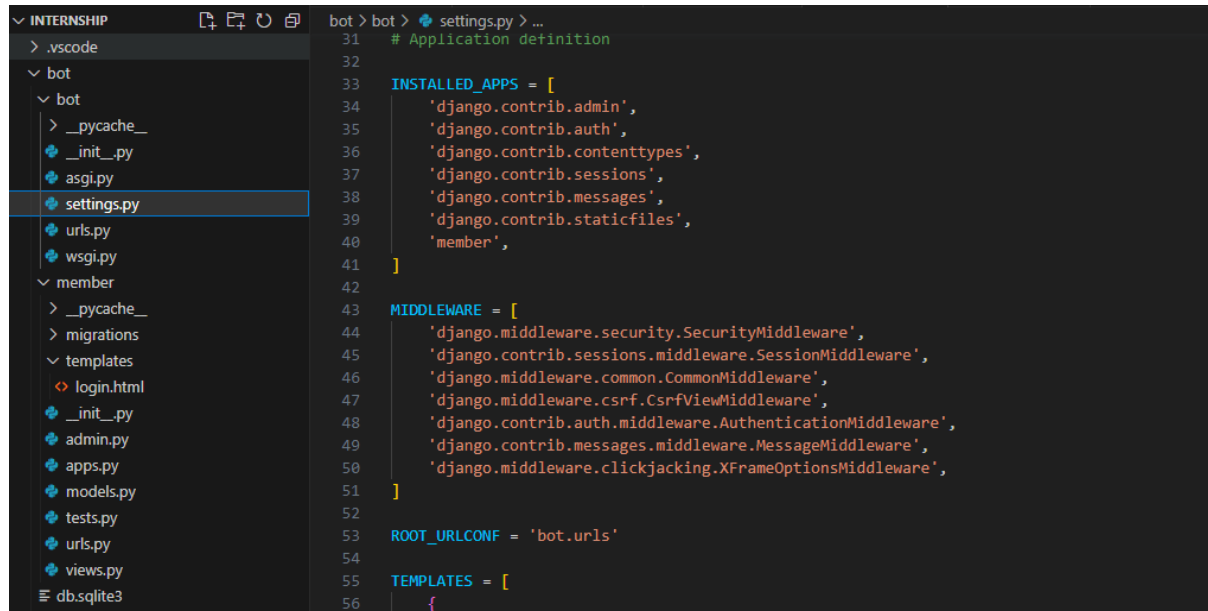
3. Import the function and add urlpatterns in the app urls.py

```
from django.urls import path
from . import views
from django.contrib import admin

urlpatterns=[
    path('', views.index),
```

]

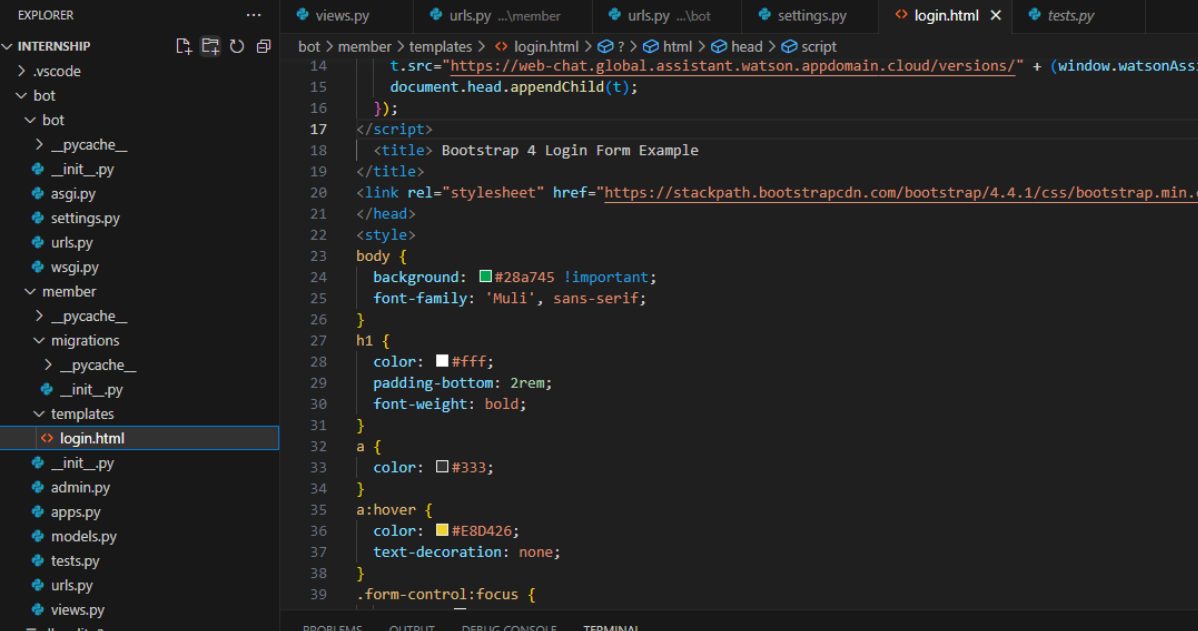
#### 4. Add the app name in the settings.py



```
bot > bot > settings.py > ...
31 # Application definition
32
33 INSTALLED_APPS = [
34     'django.contrib.admin',
35     'django.contrib.auth',
36     'django.contrib.contenttypes',
37     'django.contrib.sessions',
38     'django.contrib.messages',
39     'django.contrib.staticfiles',
40     'member',
41 ]
42
43 MIDDLEWARE = [
44     'django.middleware.security.SecurityMiddleware',
45     'django.contrib.sessions.middleware.SessionMiddleware',
46     'django.middleware.common.CommonMiddleware',
47     'django.middleware.csrf.CsrfViewMiddleware',
48     'django.contrib.auth.middleware.AuthenticationMiddleware',
49     'django.contrib.messages.middleware.MessageMiddleware',
50     'django.middleware.clickjacking.XFrameOptionsMiddleware',
51 ]
52
53 ROOT_URLCONF = 'bot.urls'
54
55 TEMPLATES = [
56     {
```

# CREATING VIEWS FOR LOGIN PAGE CODE

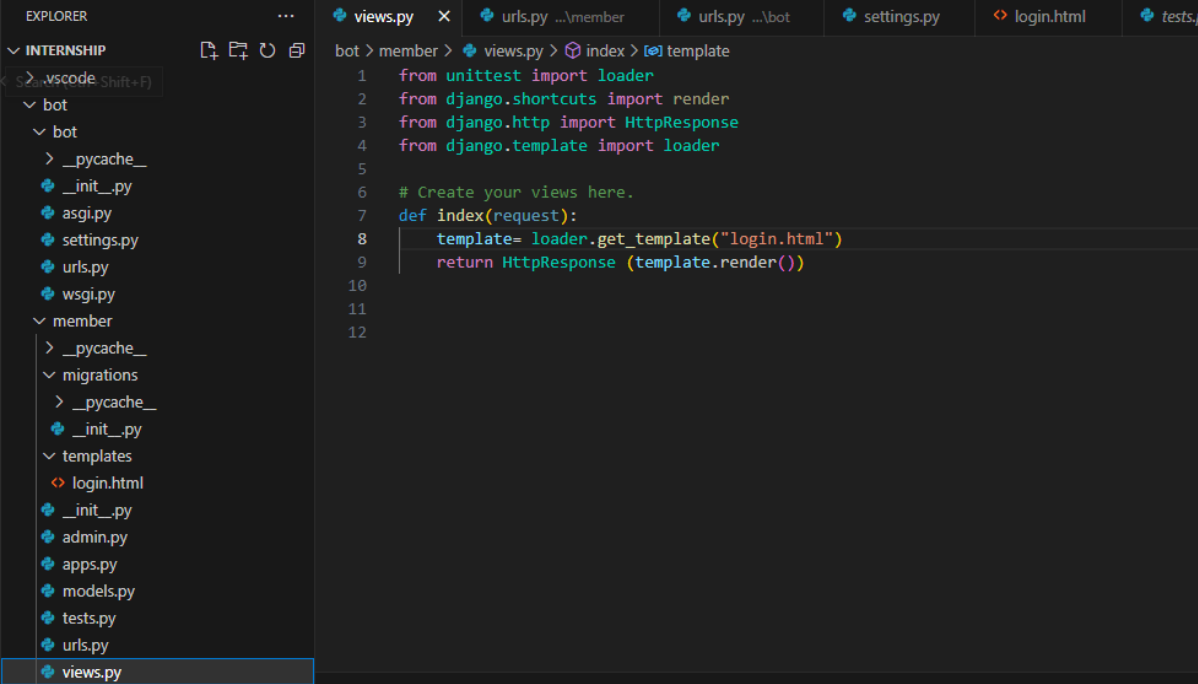
1.Create a file template inside the app and add the login page code



The screenshot shows the Visual Studio Code editor with the Explorer sidebar on the left. The 'templates' folder is expanded, and 'login.html' is selected. The main editor displays the HTML code for the login page template. The code includes a script tag for a chat widget, a title 'Bootstrap 4 Login Form Example', a link to the Bootstrap 4.4.1 CSS, and a style block with a light blue background and a serif font. The body contains a single paragraph with the text 'Welcome to the Django Project'.

```
bot > member > templates > login.html > ? > html > head > script
14 | t.src="https://web-chat.global.assistant.watson.appdomain.cloud/versions/" + (window.watsonAssi
15 | document.head.appendChild(t);
16 | });
17 | </script>
18 | <title> Bootstrap 4 Login Form Example
19 | </title>
20 | <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.c
21 | </head>
22 | <style>
23 | body {
24 | background-color: #28a745;
25 | font-family: 'Mulii', sans-serif;
26 | }
27 | h1 {
28 | color: #fff;
29 | padding-bottom: 2rem;
30 | font-weight: bold;
31 | }
32 | a {
33 | color: #333;
34 | }
35 | a:hover {
36 | color: #E8D426;
37 | text-decoration: none;
38 | }
39 | .form-control:focus {
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
```

2.Create the views

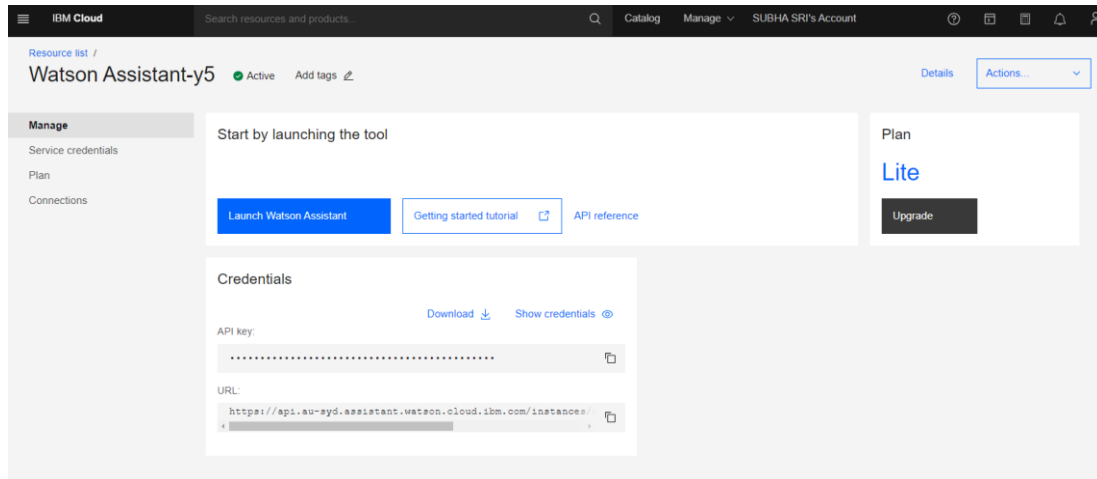


The screenshot shows the Visual Studio Code editor with the Explorer sidebar on the left. The 'views.py' file is selected. The main editor displays the Python code for the views. The code imports 'loader' from 'django.shortcuts', 'render' from 'django.shortcuts', 'HttpResponse' from 'django.http', and 'loader' from 'django.template'. It then defines an 'index' function that takes a 'request' argument, gets the 'login.html' template, and returns an 'HttpResponse' object with the rendered template.

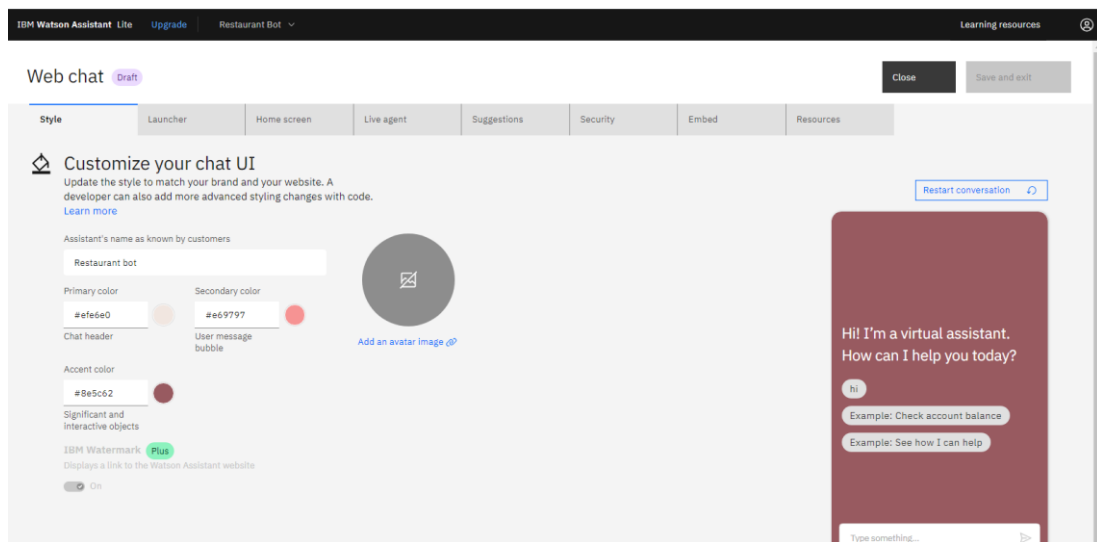
```
bot > member > views.py > index > template
1 | from django.shortcuts import loader
2 | from django.shortcuts import render
3 | from django.http import HttpResponse
4 | from django.template import loader
5 |
6 | # Create your views here.
7 | def index(request):
8 |     template= loader.get_template("login.html")
9 |     return HttpResponse (template.render())
10 |
11 |
12 |
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
```

# CHATBOT CREATION THROUGH WATSON ASSISTANT CATALOG

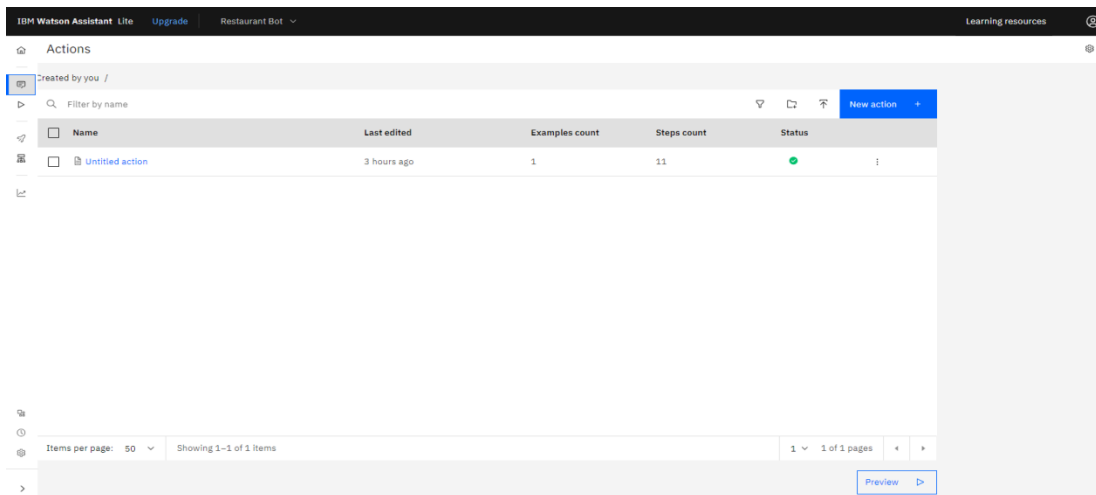
1.Login into your cloud account and search Watson Assistant in Catalog and launch it and create actions.



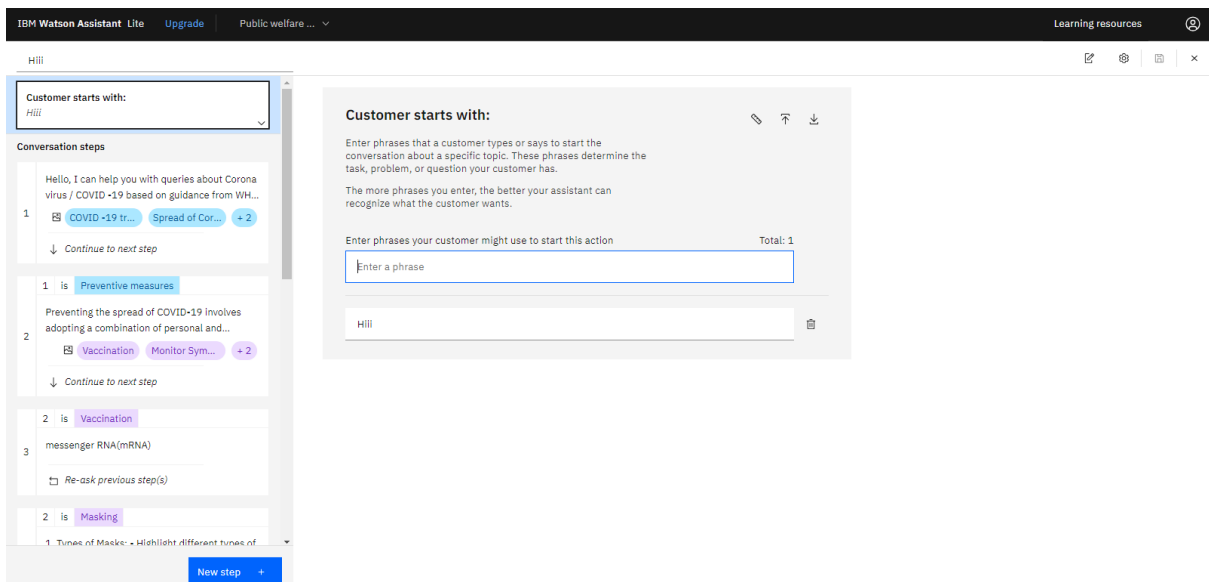
2.Customize your chatbot



3.Select Actions

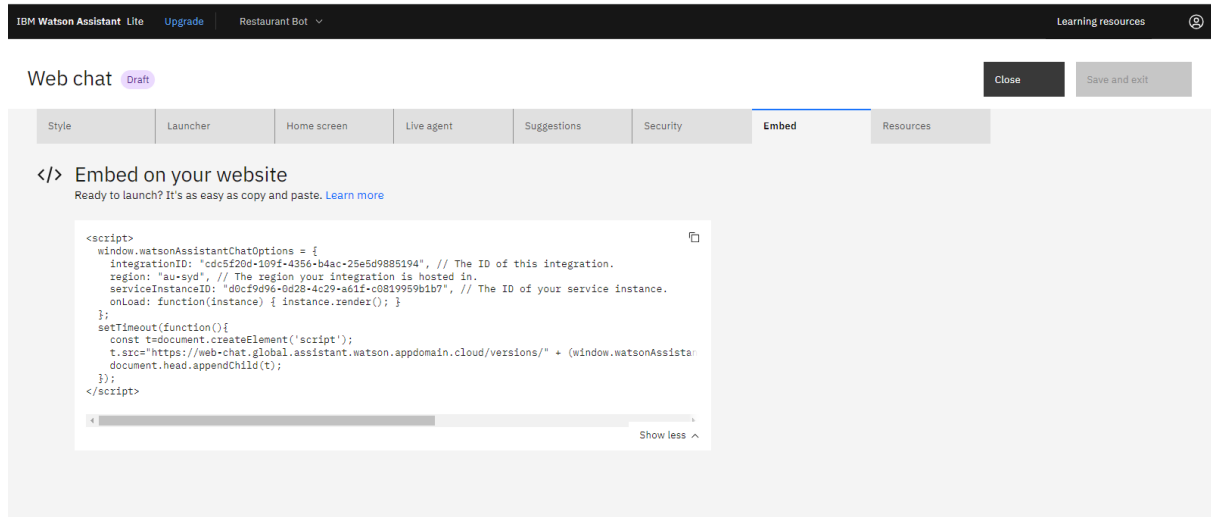


#### 4. Insert the actions for your chatbot

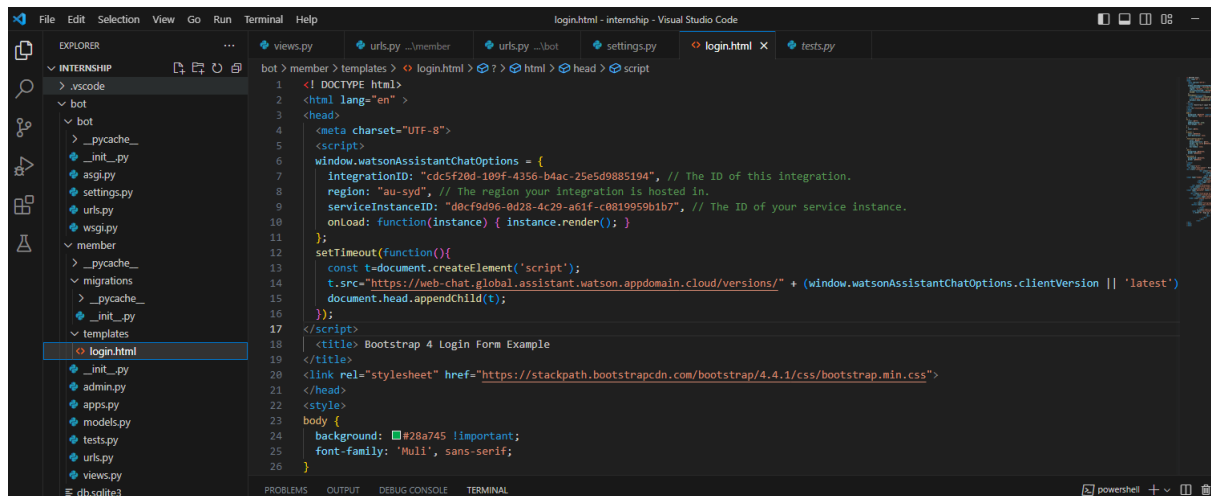


# INTEGRATING THE CHATBOT

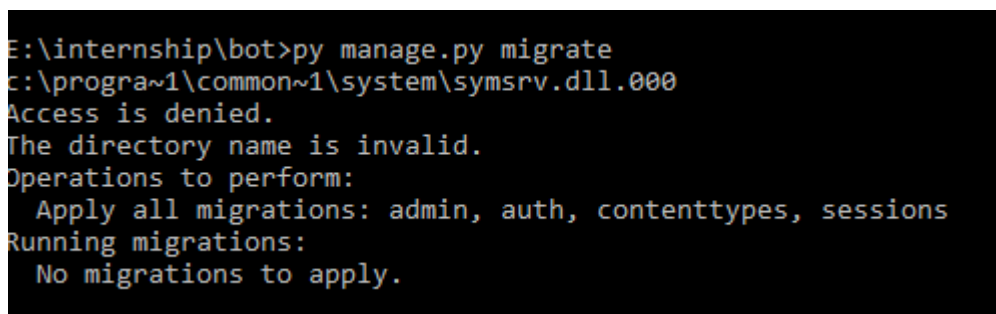
## 1.Copy the code from embed



## 2.Paste it in your home page code



## 3..Migrate it





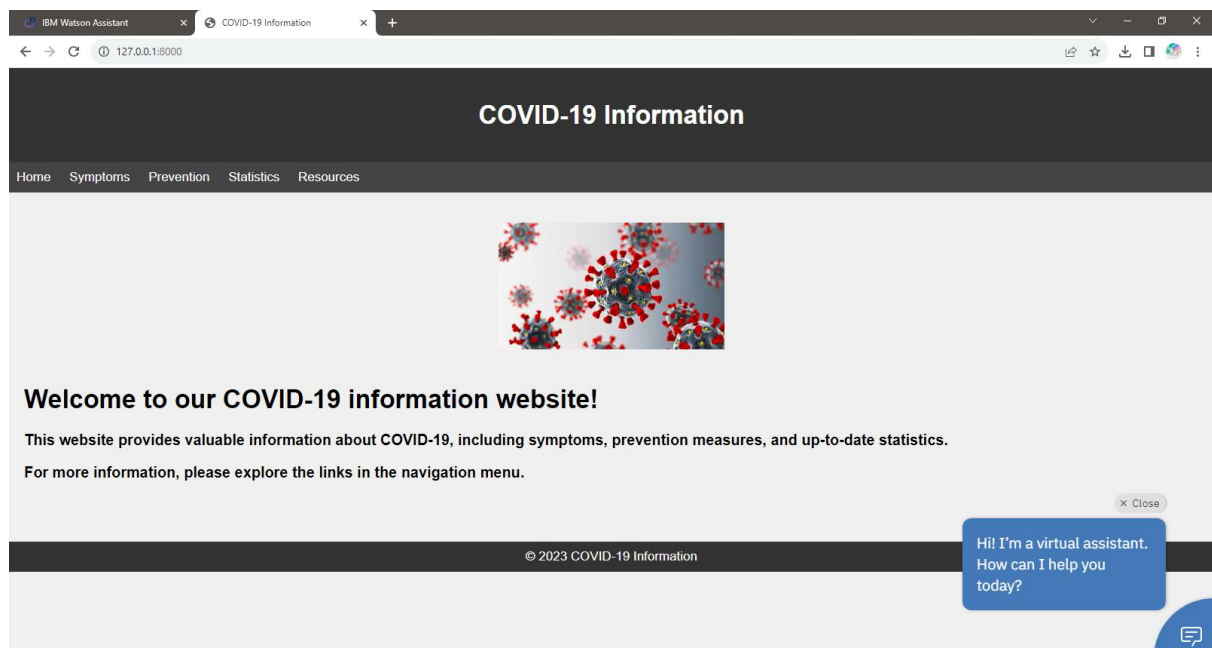
#### 4..Runserver

```
E:\internship\bot>py manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

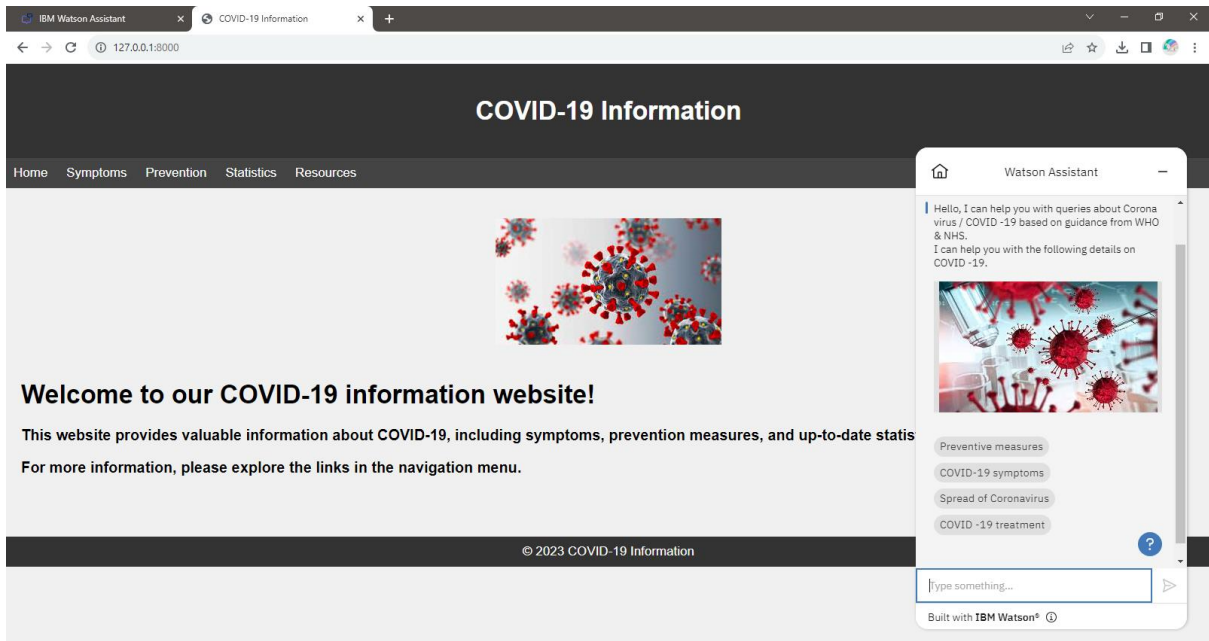
System check identified no issues (0 silenced).
June 10, 2023 - 13:47:37
Django version 4.2.2, using settings 'bot.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.

The directory name is invalid.
c:\progra~1\common~1\system\sysmrv.dll.000
Access is denied.
The directory name is invalid.
c:\progra~1\common~1\system\sysmrv.dll.000
Access is denied.
The directory name is invalid.
```

#### 5..Copy the http code and run it in the browser



# FINAL RESULT



## APPENDIX:

### SOURCE CODE:

#### urls.py

```
"""
URL configuration for bot project.

The `urlpatterns` list routes URLs to views. For more information please see:
    https://docs.djangoproject.com/en/4.2/topics/http/urls/
Examples:
Function views
    1. Add an import:  from my_app import views
    2. Add a URL to urlpatterns:  path('', views.home, name='home')
Class-based views
    1. Add an import:  from other_app.views import Home
    2. Add a URL to urlpatterns:  path('', Home.as_view(), name='home')
Including another URLconf
    1. Import the include() function: from django.urls import include, path
    2. Add a URL to urlpatterns:  path('blog/', include('blog.urls'))
"""
from django.contrib import admin
from django.urls import include, path

urlpatterns = [
    path('', include('member.urls')),
    path('admin/', admin.site.urls),
]
```

#### settings.py

```
"""
Django settings for bot project.

Generated by 'django-admin startproject' using Django 4.2.2.

For more information on this file, see
    https://docs.djangoproject.com/en/4.2/topics/settings/

For the full list of settings and their values, see
    https://docs.djangoproject.com/en/4.2/ref/settings/
"""

from pathlib import Path

# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).resolve().parent.parent

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

```
# See https://docs.djangoproject.com/en/4.2/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'django-insecure-!aga#hcx4$&f3i-k4ltxnp&2=oshp85@dg9zj2i7)r3al(44dw'

# 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',
    'member',
]

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 = 'bot.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 = 'bot.wsgi.application'
```

```
# Database
# https://docs.djangoproject.com/en/4.2/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
    }
}

# Password validation
# https://docs.djangoproject.com/en/4.2/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/4.2/topics/i18n/

LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'UTC'

USE_I18N = True

USE_TZ = True

# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/4.2/howto/static-files/

STATIC_URL = 'static/'

# Default primary key field type
# https://docs.djangoproject.com/en/4.2/ref/settings/#default-auto-field

DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'
```

## App creation:

### urls.py

```
from django.urls import path
from . import views
from django.contrib import admin

urlpatterns=[
    path('',views.index),
]
```

### views.py

```
from unittest import loader
from django.shortcuts import render
from django.http import HttpResponse
from django.template import loader

# Create your views here.
def index(request):
    template= loader.get_template("login.html")
    return HttpResponse (template.render())
```

## login page code:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>COVID-19 Information</title>
    <style>
        body {
            background-color: #f0f0f0; /* Set your desired background color */
            font-family: Arial, sans-serif;
            margin: 0;
            padding: 0;
        }

        header {
            background-color: #333; /* Header background color */
            color: #fff; /* Header text color */
            text-align: center;
            padding: 20px;
        }

        nav {
            background-color: #444; /* Navigation background color */
```

```

        padding: 10px;
    }

    nav a {
        color: #fff; /* Navigation link color */
        text-decoration: none;
        margin-right: 20px;
    }

    main {
        padding: 20px;
    }

    footer {
        background-color: #333; /* Footer background color */
        color: #fff; /* Footer text color */
        text-align: center;
        padding: 10px;
    }
</style>
</head>
<body>
    <header>
        <h1>COVID-19 Information</h1>
    </header>
    <nav>
        <a href="https://www.who.int/health-topics/coronavirus">Home</a>
        <a href="https://www.cdc.gov/coronavirus/2019-ncov/symptoms-
testing/symptoms.html">Symptoms</a>
        <a href="https://www.cdc.gov/coronavirus/2019-ncov/prevent-getting-
sick/prevention.html">Prevention</a>
        <a href="https://www.worldometers.info/coronavirus/">Statistics</a>
        <a href="https://en.wikipedia.org/wiki/Coronavirus">Resources</a>
    </nav><br><br>
    <div class="img">
        <center>
        </center></div>
    <main>
        <h1>Welcome to our COVID-19 information website!</h1>
        <p><h3>This website provides valuable information about COVID-19, including
symptoms, prevention measures, and up-to-date statistics.</h3></p>
        <p><h3>For more information, please explore the links in the navigation
menu.</h3></p>
    </main>
    <br><br>
    <footer>
        &copy; 2023 COVID-19 Information
    </footer>
</body>
</html>

```