

## **Project Development Phase**

### **Delivery of Sprint- 4**

|                    |   |
|--------------------|---|
| <b>Date</b>        | 14 November 2022                        |
| <b>TeamID</b>      | PNT2022TMID07052                        |
| <b>ProjectName</b> | AI-based discourse for Banking Industry |

### **Creating Assistant & Integrate with Flask Web Page**

You will be creating a banking bot in this activity that has the following capabilities

- 1) The Bot should be able to guide a customer to create a bank account.
- 2) The Bot should be able to answer loan queries.
- 3) The Bot should be able to answer general banking queries.
- 4) The Bot should be able to answer queries regarding net banking.
- 5) With the help of this bot ,you can get all the required details related to banking.

Let us build our flask application which will be running in our local browser with a user interface.

In the flask application ,users will interact with the chatbot, and based on the user queries they will get the outcomes.

This process includes the following steps:

- 1) Building python code
- 2) Building html code
- 3) Running the application

# Build Python Code

## 1. Importing Libraries

The first step is usually importing the libraries that will be needed in the program.

```
1 from flask import Flask,render_template
```

Importing the flask module into the project is mandatory. An object of the Flask class is our WSGI application. Flask constructor takes the name of the current module(name).

## 2. Creating our flask application and loading

```
2 app=Flask(__name__,template_folder='template')
```

## 3. Routing to the Html Page

Here, the declared construct or is used to route to the HTML page created earlier.

The “ route is bound with the bot function. Hence, when the home page of a webserver is opened in the browser, the HTML page will be rendered.

```
3 @app.route('/')
4 def Chatbot():
5     return render_template('Chatbot.html')
```

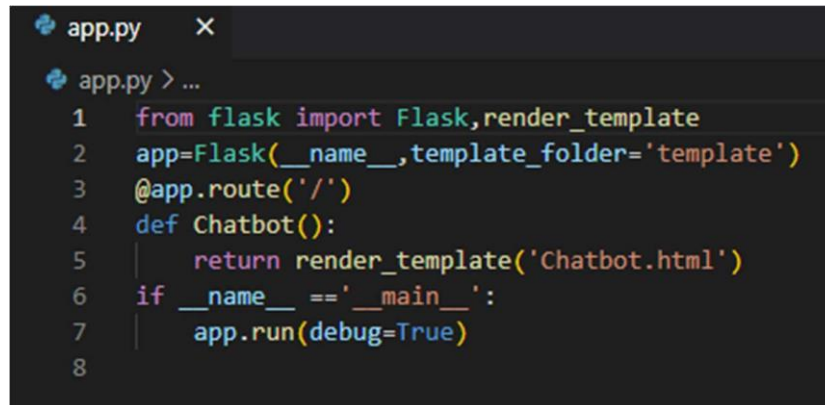
## Main Function

This is used to run the application in localhost. The debug=True property will display what kind of error we are facing while running

the application. If there are no errors ,flask will run the html file successfully.

```
6  if __name__ == '__main__':  
7      app.run(debug=True)
```

The Full Python code looks like:



```
app.py  X  
app.py > ...  
1  from flask import Flask,render_template  
2  app=Flask(__name__,template_folder='template')  
3  @app.route('/')  
4  def Chatbot():  
5      return render_template('Chatbot.html')  
6  if __name__ == '__main__':  
7      app.run(debug=True)  
8
```

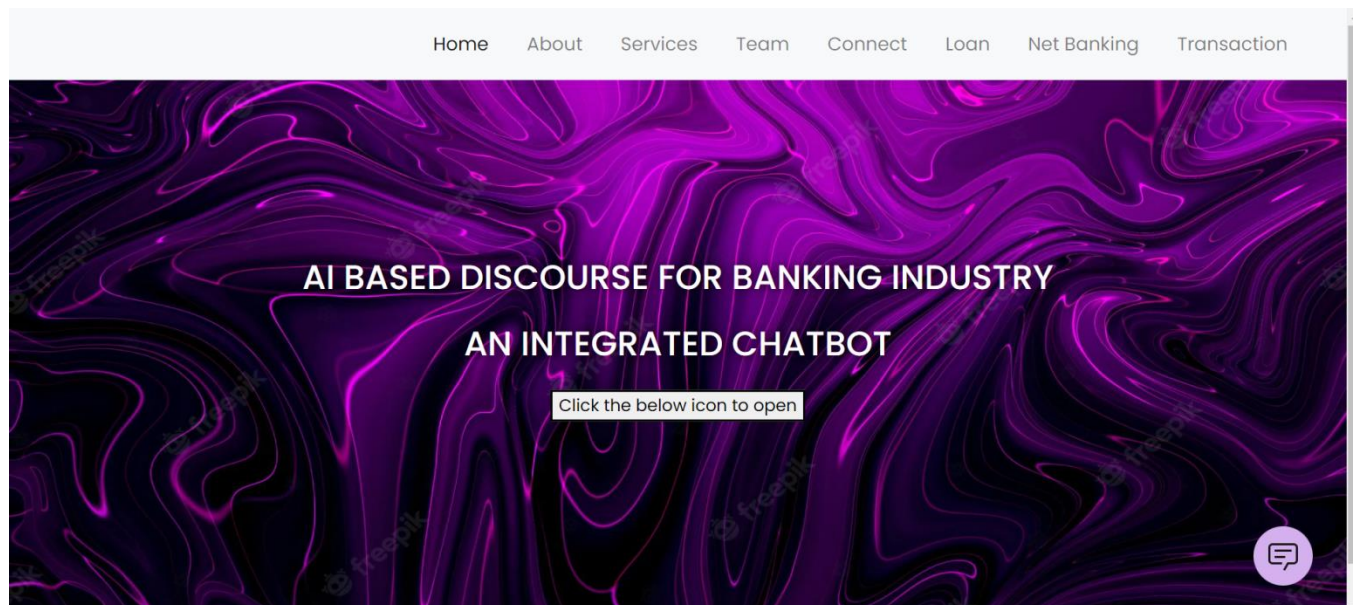
## Build HTML Code

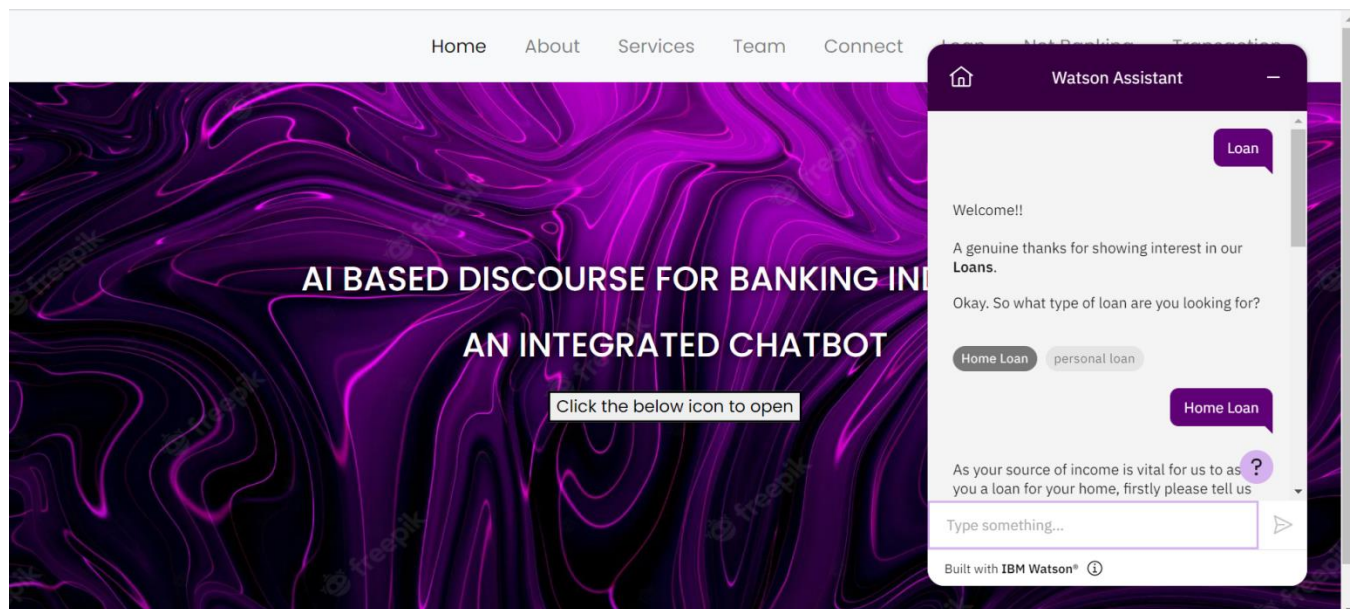
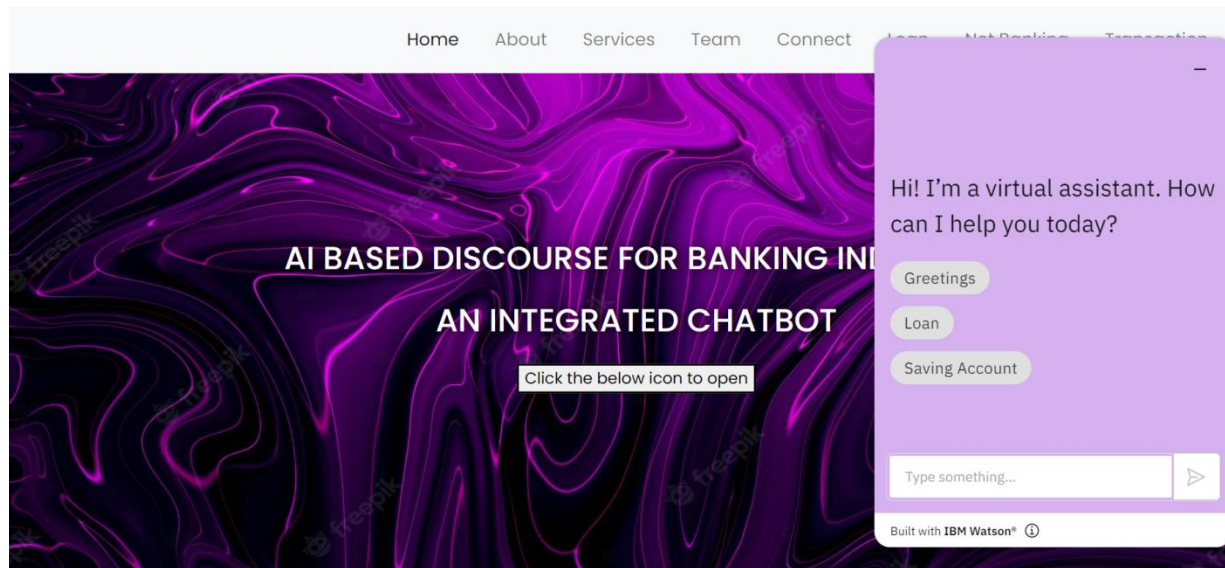
- We use HTML to create the front-end part of the webpage.
- Here, we have created 1 HTMLpage-Chatbot.html
- Chatbot.html displays the home page which integrates with Watson Assistant.
- A simple HTML page is created. Auto-generated source code from IBM Watson Assistants is copied and pasted inside the body tag.
- The links, paths and the folder names are correctly assigned as per the default names which python flask will be able to recognize.

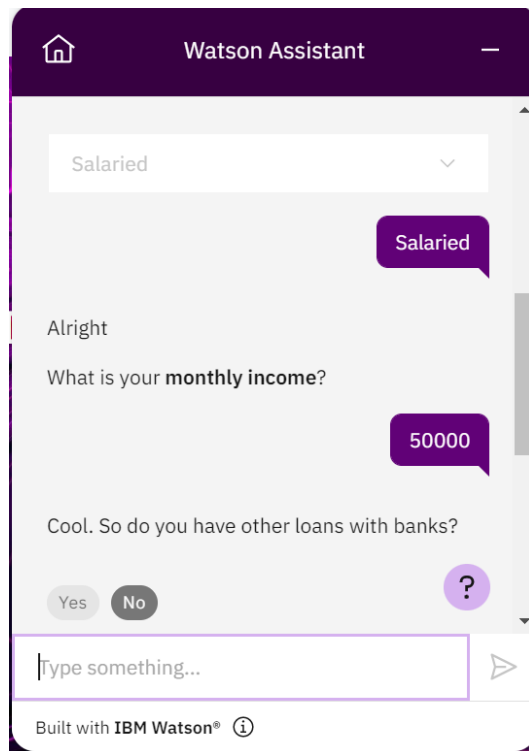
## Run the application

- Open the anaconda prompt from the start menu.
- Navigate to the folder where your app.py resides.
- Now type the “python app.py” command.
- It will show the localhost where your app is running on `http://127.0.0.1.5000/`
- Copy that localhost URL and open that URL in the browser. It does navigate me to where you can view your webpage.

## Final Output:







**Source code will be attached in Final Deliverables.**

**NOTE: SINCE NO CODES WERE USED IN CREATING ASSISTANT AND INTEGRATE WITH FLASK WEB PAGE**