

## Project Development Phase

### Delivery of Sprint - 4

Date	24 November 2022
Team ID	PNT2022TMID50971
Project Name	AI based discourse for Banking Industry

#### Creating Assistant & Integrate With Flask Web Page

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

In the flask application, users will interact with the chat bot, and based on the user queries they will get the chatbotresponses.

#### Building Python Code

##### 1: Importing Libraries

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

```
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

```
app = Flask(__name__)
```

### 3: Routing to the Html Page

Here, the declared constructor 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 web server is opened in the browser, the HTML page will be rendered.

```
@app.route('/')  
def bot():  
    return render_template('chatbot.html')
```

### Main Function

This is used to run the application in local host.

```
if __name__ == '__main__':  
    app.run()
```

### Building HTML Code

We have used HTML to create the front-end part of the web page.

Here, we have created "index.html" displays the home page which gets integrated with WatsonAssistant.

Auto-generated source code which contains the Integration ID of IBM Watson Assistants is copied and pasted inside the body tag.

### Run the application

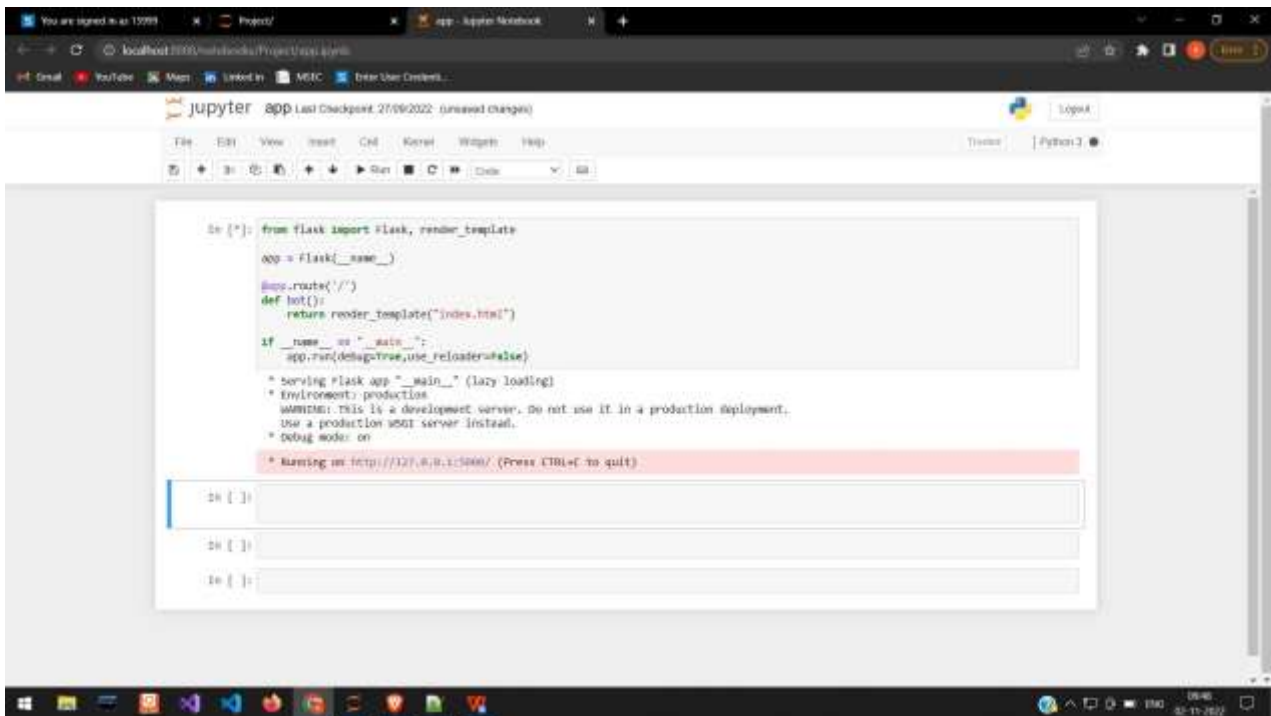
Open Jupyter notebook (anaconda3)

Navigate to the folder where app.ipynb resides.

Run the python code

Open a browser and type this URL <http://127.0.0.1:5000/>

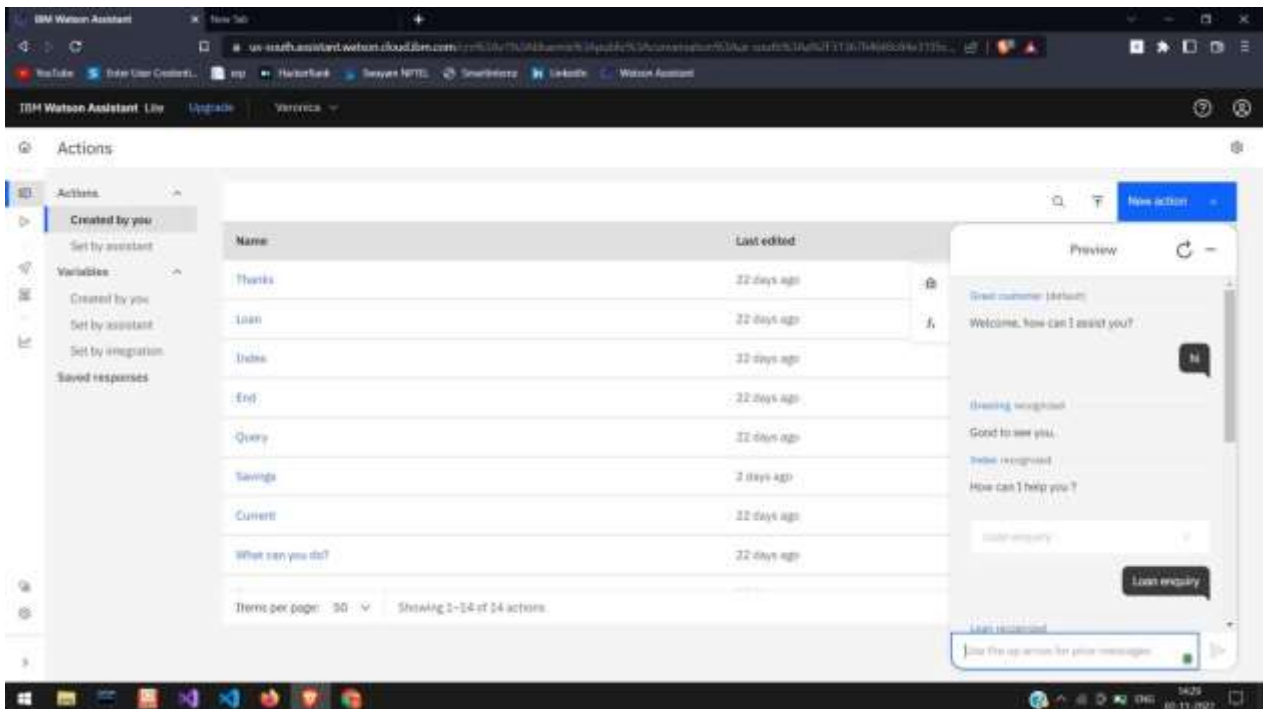
It launches the application integrated with IBM Watson Assistant.



The screenshot shows a Jupyter Notebook window titled 'app' with a Python 3 kernel. The code in the notebook is as follows:

```
In [*]: from flask import Flask, render_template
        app = Flask(__name__)
        @app.route('/')
        def bot():
            return render_template("index.html")
        if __name__ == "__main__":
            app.run(debug=True, use_reloader=False)
        * Serving Flask app "main_" (lazy loading)
        * Environment: production
        WARNING: This is a development server. Do not use it in a production deployment.
        Use a production WSGI server instead.
        * Debug mode: on
        * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Below the code cell, there are three empty input fields for the next steps.



The screenshot shows the IBM Watson Assistant interface. On the left, there is a sidebar with 'Actions' and 'Variables' sections. The 'Actions' section is expanded, showing a list of actions created by the user. The main area displays a table of these actions.

Name	Last edited
Thanks	22 days ago
Loan	22 days ago
Index	22 days ago
End	22 days ago
Query	22 days ago
Savings	2 days ago
Current	22 days ago
What can you do?	22 days ago

At the bottom of the table, it says 'Items per page: 50' and 'Showing 1-14 of 14 actions'.

On the right, there is a 'Preview' chat window. It shows a conversation with a 'Guest customer (default)' who says 'Welcome, how can I assist you?'. The assistant responds with 'Greeting: welcome!'. The user then says 'Good to see you.' and the assistant responds with 'Index: response'. The user then says 'How can I help you?' and the assistant responds with 'Loan: inquiry'. The user then says 'Loan: response' and the assistant responds with 'Loan: response'.

