# FIT3164 - User Guide

Prepared by MDS20
Cheng Tze Pin 32870663
Lim Jun Yi 32625510
Prateek Tripathi 31835643

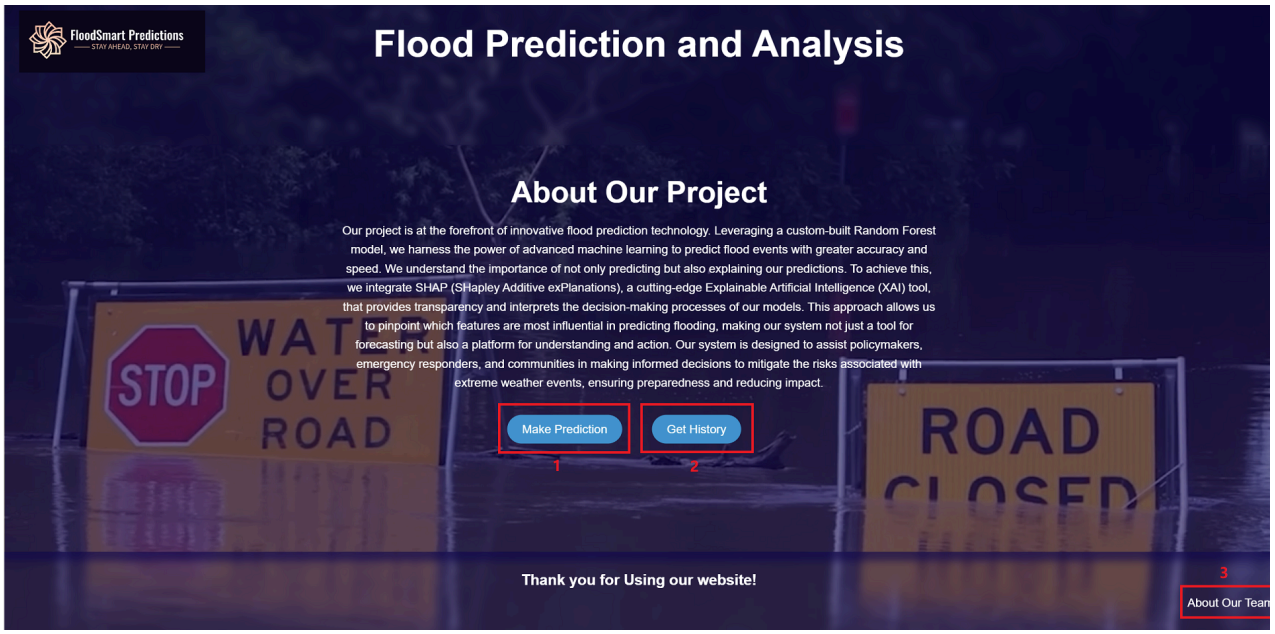
# Table of Contents

# 1. End User Guide

Welcome to the FloodSmart Predictions User Guide! This guide is crafted to assist you in navigating and utilising our Flood Prediction and Analysis web application effectively. Whether you're a policymaker, emergency responder, or a concerned citizen, this guide will provide detailed instructions on accessing and interpreting the vital information our tool offers. Our application leverages advanced machine learning technology to provide accurate flood event predictions, helping you make informed decisions and prepare for potential flood risks. Follow the instructions in this guide to learn how to make the most out of our innovative prediction platform.

## 1.1 Home Page

Navigate around the menu




(Please note that due to a formatting issue with the website, when zooming in/out, there will be website elements formatting bug. For a better user experience, keep the website's zoom at between 75% - 125%.)

Once you launch our Flood Prediction webpage, You will see a Home Page. This is the landing page for our prediction web application, in this page, you will see there are 3 interactive buttons.

- **1. "Make Predictions"**

  This button will lead you to the prediction sub-page, where flood prediction results are being shown.

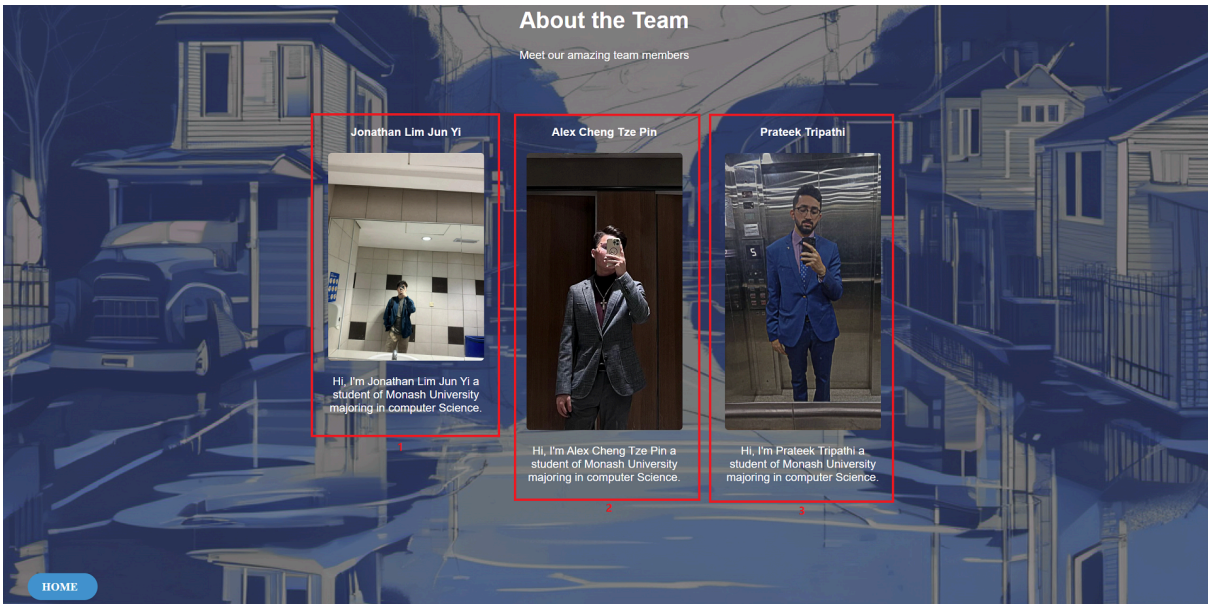
- **2. "Get History"**

  This button will lead you to a page where you can get the prediction history of all previous prediction attempts.

- **3. “About Our Team”**

  This link will lead you to an About page where the description of the developers and contributors of this flood prediction application is located.

## 1.2 About Our Team Page




On this page, you'll meet the dedicated team members behind our Flood Prediction and Analysis application. This section provides a brief overview of each individual contributing to the development and success of our application.

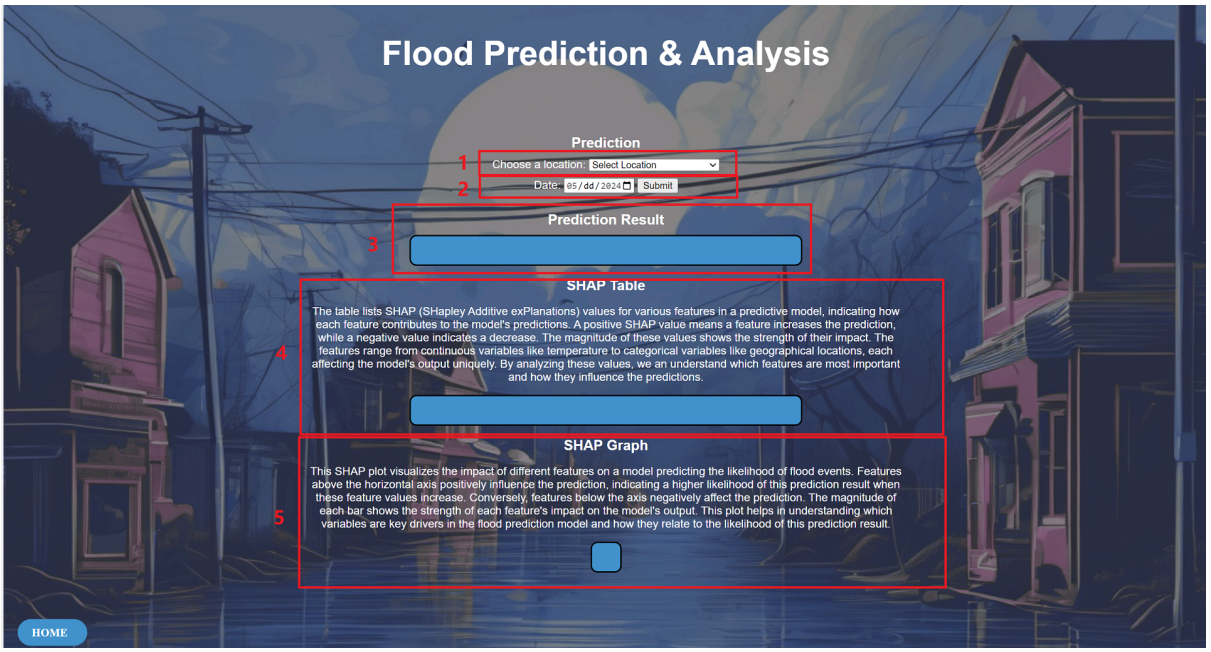
- **1. Lim Jun Yi (model and XAI)**
- **2. Cheng Tze Pin (API and database)**
- **3. Prateek Tripathi (Web UI)**

To return to the Home page, click on the “HOME” button at the bottom left corner of the website.

To exit the Home page, click on the “x” mark on your browser.

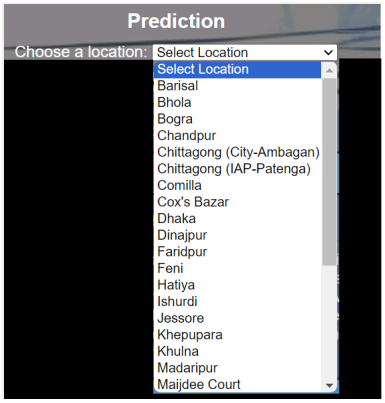
The detailed guides on the Prediction page and History page will be shown in the next section.

## 1.3 Prediction & Analysis Page



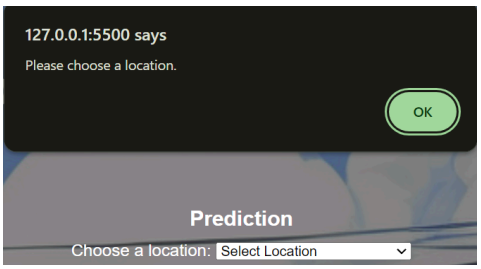

Upon entering the prediction page, you will see there are some attributes on the websites with descriptions, don't worry if you're confused, we will walk you through them.

- **1. "Choose a location"**



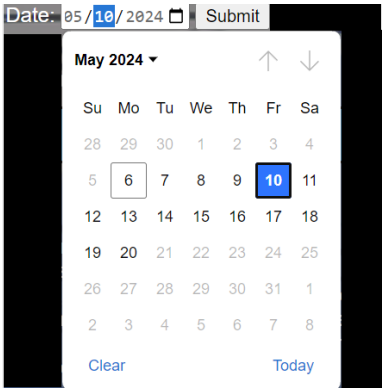

For this attribute, you will need to choose the location/city in Bangladesh that you want to make a flood prediction for. For example, if you wanted to predict the flood status for "Dhaka", you would select "Dhaka" in the location attribute.

(note that due to dataset and API limitations, not all cities in Bangladesh are included in this prediction software and hence no prediction for those locations is available.)



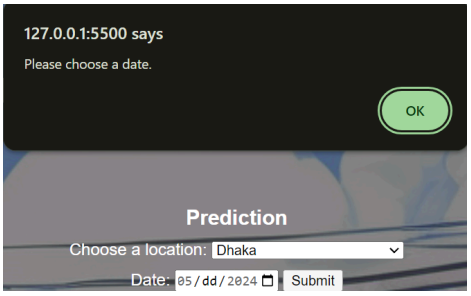

If you leave the location attribute blank, an error message will appear. A location must be chosen for the prediction to proceed.

- **2. “Date”**




For this attribute, you will need to select the date that you want to predict, that is, the flood prediction model will try to predict whether it will have a flood or not during that particular date with a confidence score(ranging from 0-1). Let’s say you wanted to predict a flood event for “Dhaka” on **10 May 2024**, then you would click the corresponding date on the date selector.

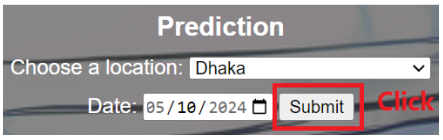
(note that the prediction date can only be today or up to the next 2 weeks, this is due to that Weather API can only supply the next 2 weeks of forecast data.)



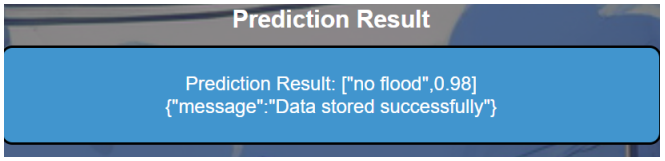

If you leave the date attribute blank, an error message will appear. A date must be chosen for the prediction to proceed.

- **3. Prediction Result**

Assume you predicted a flood event in “Dhaka” on **10 May 2024** and clicked the **Submit button**.

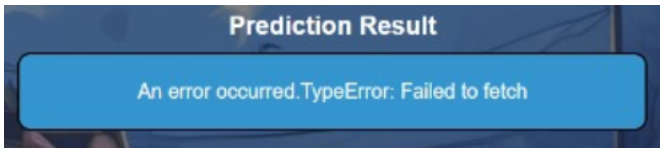



The below will be the Result output:

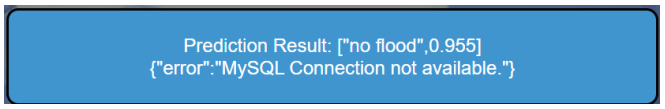



The prediction result will return 2 lines of string message. The first line of string indicates the result of a flood event. In this case, the flood prediction result for “Dhaka” on **10 May 2024** is there will be no flood with a confidence of 0.98.

The second line of string indicates if the prediction result has been successfully stored in the database. In this example, our result for “Dhaka” on **10 May 2024** has been stored in the database and later can be retrieved in the History Page.

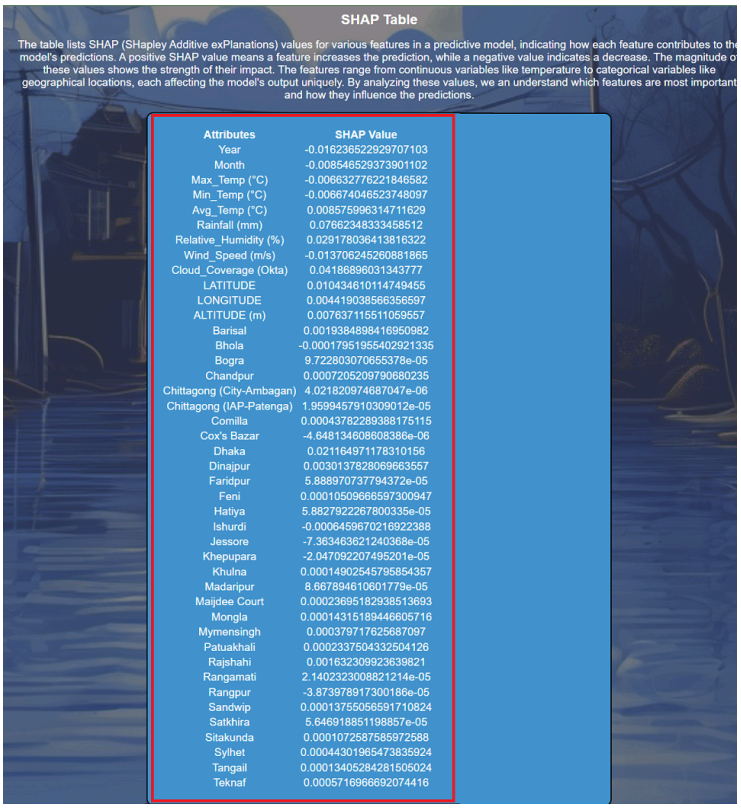



An error message will occur if the Administrator has not set up the prediction API. “Failed to fetch”, showing that there isn’t a connection with the prediction model and hence cannot get the prediction result. To deal with this situation, please contact the software administrator or run the “PredictionModelAPI.py” file.




An error message will occur if the Administrator has not set up the database API. “MySQL connection not available”, showing that there isn’t a connection with the database and hence cannot store the result. To deal with this situation, please contact the software administrator or run the “DatabaseAPI.py” file.

- **4. SHAP Table**



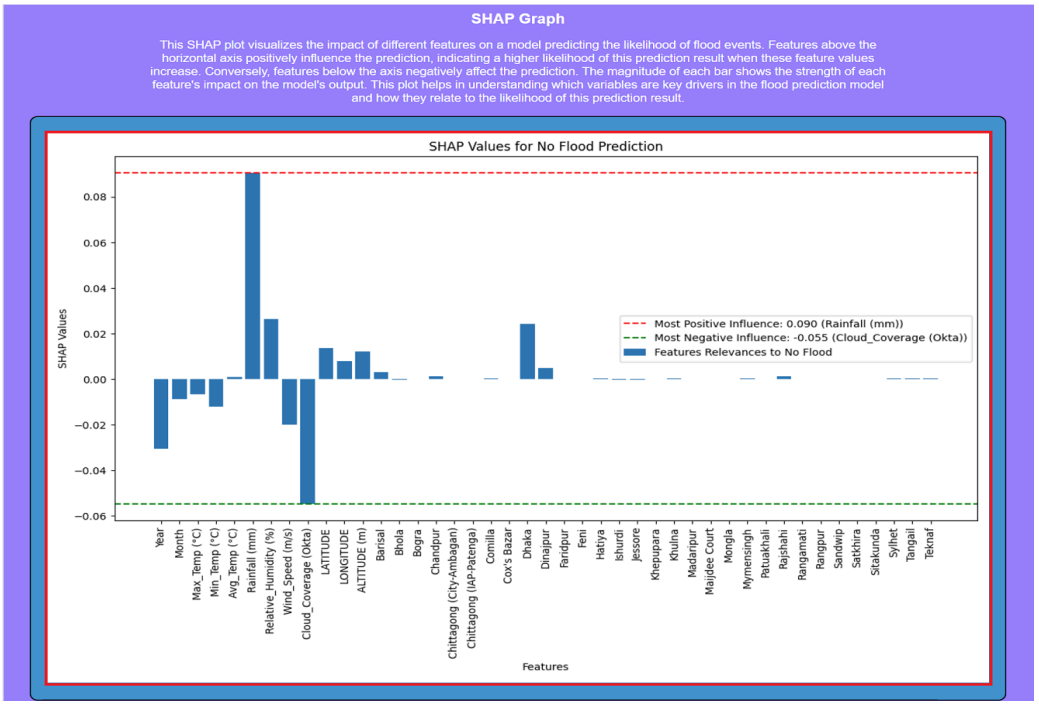
SHAP Table

The table lists SHAP (SHapley Additive exPlanations) values for various features in a predictive model, indicating how each feature contributes to the model's predictions. A positive SHAP value means a feature increases the prediction, while a negative value indicates a decrease. The magnitude of these values shows the strength of their impact. The features range from continuous variables like temperature to categorical variables like geographical locations, each affecting the model's output uniquely. By analyzing these values, we an understand which features are most important and how they influence the predictions.

| Attributes | SHAP Value |
|---|---|
| Year | -0.016236522929707103 |
| Month | -0.008546529373901102 |
| Max_Temp (°C) | -0.006632776221846582 |
| Min_Temp (°C) | -0.006674046523748097 |
| Avg_Temp (°C) | 0.008575996314711629 |
| Rainfall (mm) | 0.07662348333458512 |
| Relative_Humidity (%) | 0.029178036413816322 |
| Wind_Speed (m/s) | -0.013706245260881865 |
| Cloud_Coverage (Okta) | 0.04186896031343777 |
| LATITUDE | 0.010434610114749455 |
| LONGITUDE | 0.004419038566356597 |
| ALTITUDE (m) | 0.007637115511059557 |
| Barisal | 0.0019384898416950982 |
| Bhola | -0.0001795195540292133 5 |
| Bogra | 9.722803070655378e-05 |
| Chandpur | 0.0007205209790680235 |
| Chittagong (City-Ambagan) | 4.021820974687047e-06 |
| Chittagong (IAP-Patenga) | 1.959945791030901 2e-05 |
| Comilla | 0.00043782289388175115 |
| Cox's Bazar | -4.648134608608386e-06 |
| Dhaka | 0.021164971178310156 |
| Dinajpur | 0.0030137828069663557 |
| Faridpur | 5.888970737794372e-05 |
| Feni | 0.0001050966659730094 7 |
| Hatiya | 5.882792226780033 5e-05 |
| Ishurdi | -0.0006459670216922388 |
| Jessore | -7.363463621240368e-05 |
| Khepupara | -2.047092207495201e-05 |
| Khulna | 0.0001490254579585435 7 |
| Madaripur | 8.667894610601779e-05 |
| Maijdee Court | 0.0002369518293851369 3 |
| Mongla | 0.0001431518944605 7716 |
| Mymensingh | 0.000379717625687097 |
| Patuakhali | 0.0002337504332504126 |
| Rajshahi | 0.001632309923639821 |
| Rangamati | 2.1402323008821214e-05 |
| Rangpur | -3.8739789173001 86e-05 |
| Sandwip | 0.0001375505659171 0824 |
| Satkhira | 5.646918851198857e-05 |
| Sitakunda | 0.000107258758597258 8 |
| Sylhet | 0.0004430196547383592 4 |
| Tangail | 0.0001340528428150502 4 |
| Teknaf | 0.0005716966692074416 |

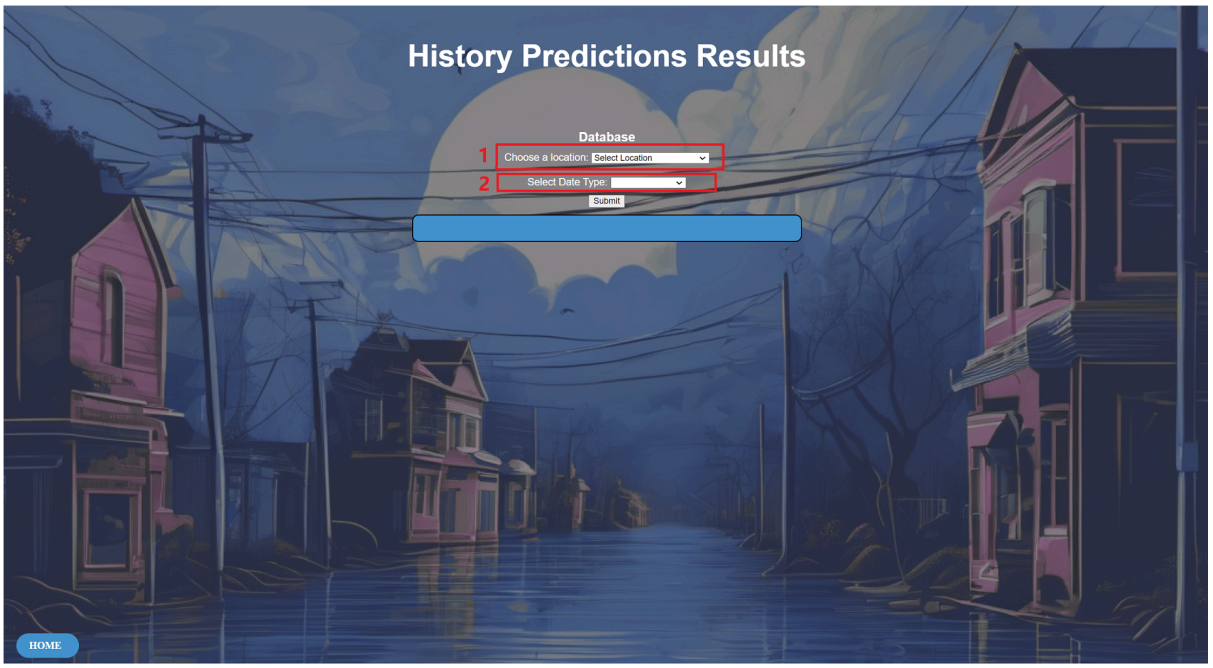
This should be the SHAP(SHapley Additive exPlanations)/XAI table output by our prediction model for a flood event prediction for “Dhaka” on **10 May 2024**. The SHAP table will tell you the information about how each attribute contributed to the prediction result, a positive value means positive influence and vice versa, a detailed description of how to interpret this table is shown on the website under “SHAP Table ".

- **5. SHAP Graph**



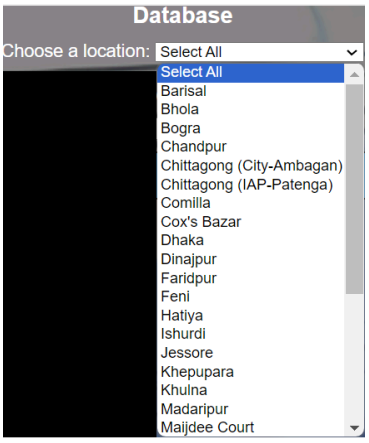
The red circled plot will be the example SHAP values plot for the flood prediction of "Dhaka" on **10 May 2024**. This plot visualised the SHAP table above to provide a better & easier understanding of the different attribute's SHAP values. A detailed description of how to interpret the plot is shown on the website under "SHAP Graph".

## 1.4 Prediction History Page



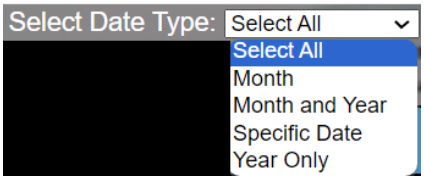

Upon entering the prediction history page, there will be mainly 2 attributes that control how you access the historical prediction data.

- **1. "Choose a location"**

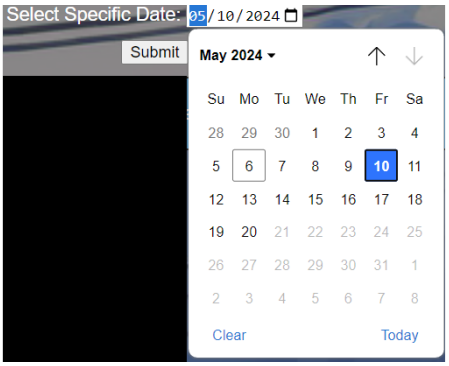



For this attribute, you can choose which location you want to filter the historical prediction data on. You can either leave this field on "Select All" and all the locations will be shown or choose a specific location. For example, we choose "Dhaka".

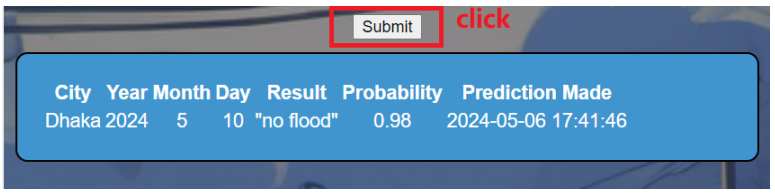
- **2. “Select Date Type”**




For this attribute, you can choose the date format type that you want to filter the historical prediction data on. You can either leave this field on “Select All” and all the data in the database for the selected city will be shown or choose a specific date format. For example, choosing “Specific Date”:




A calendar date selector will pop out after choosing the “Specific Date” format. Here you can filter the prediction result based on a specific date you like. For example, we want to see all predictions made for “Dhaka” on **10 May 2024**.



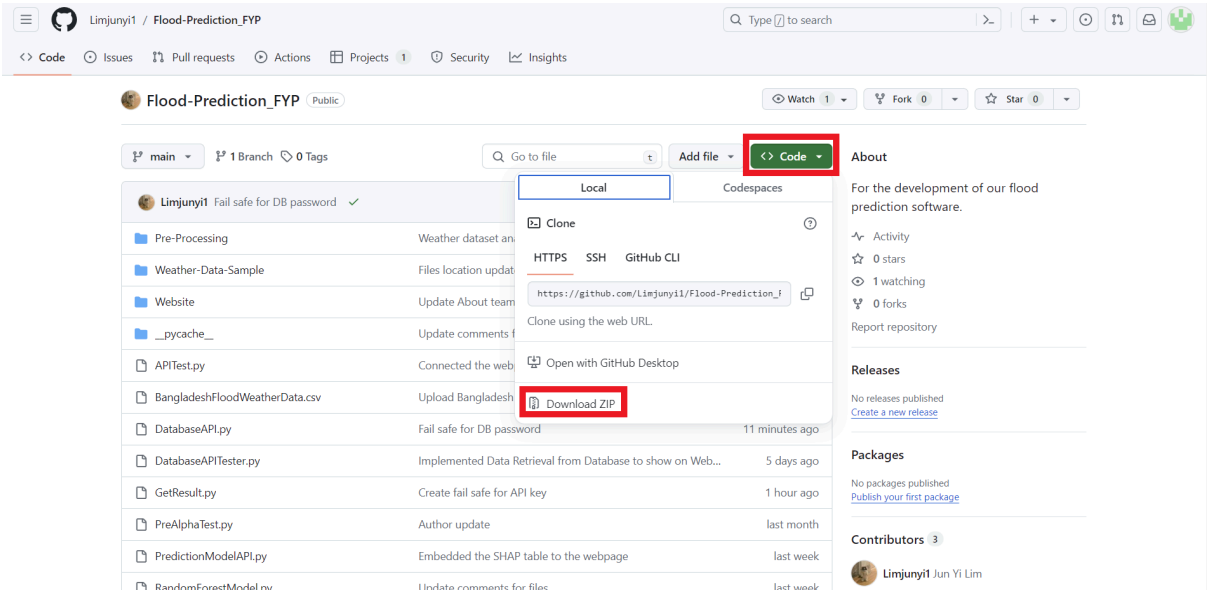

After clicking submit, we can see the historic result for “Dhaka” on **10 May 2024** that we did earlier.

# 2. Technical Guide

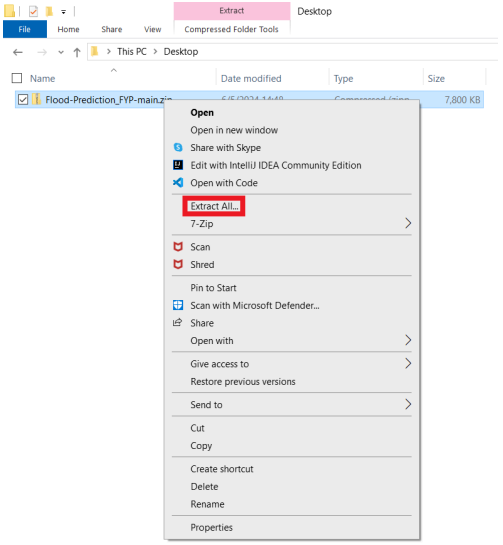
## 2.1 Accessing the Application

IMPORTANT NOTE!!!
You need to install the libraries used in the files if you do not have them using the following command by replacing the LIBRARY_NAME with the actual library:
pip install LIBRARY_NAME
If you wish to start the application without knowing the codes, please jump to 2.5 Running the Application as a Whole
To access the Application's source code, the administrator should follow the steps below:

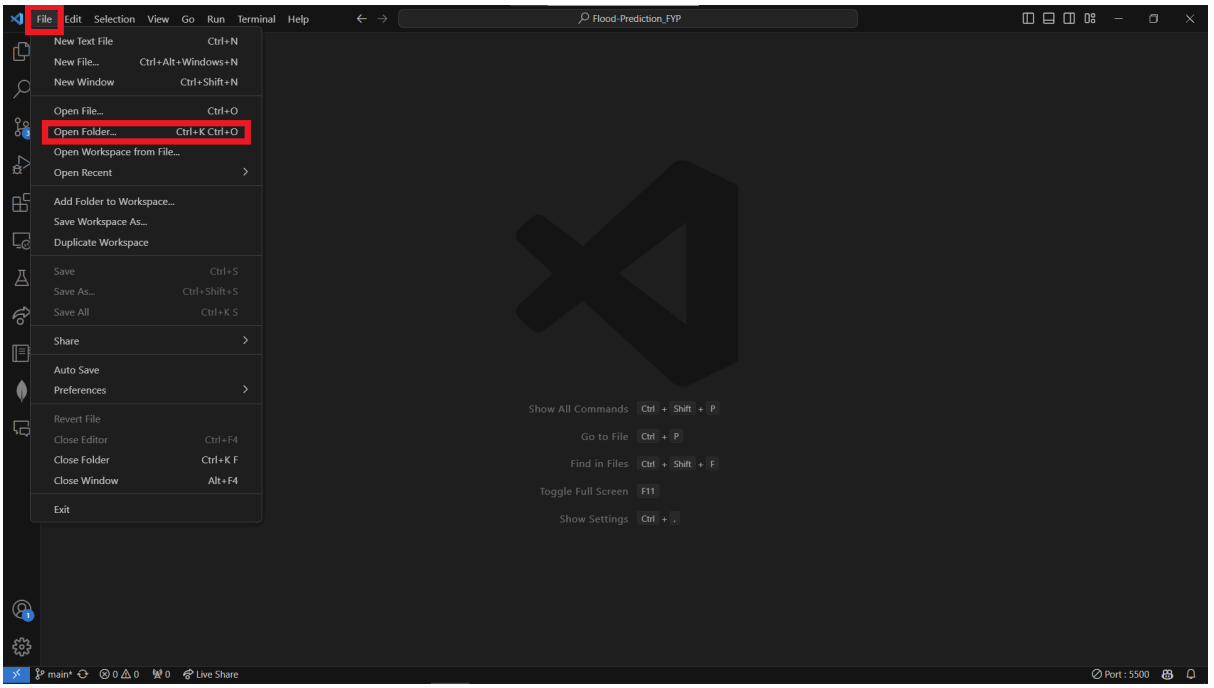
1. Go to GitHub and access our repository via https://github.com/Limjunyi1/Flood-Prediction_FYP.
2. Download the codes as a zip folder




3. Unzip the zip folder

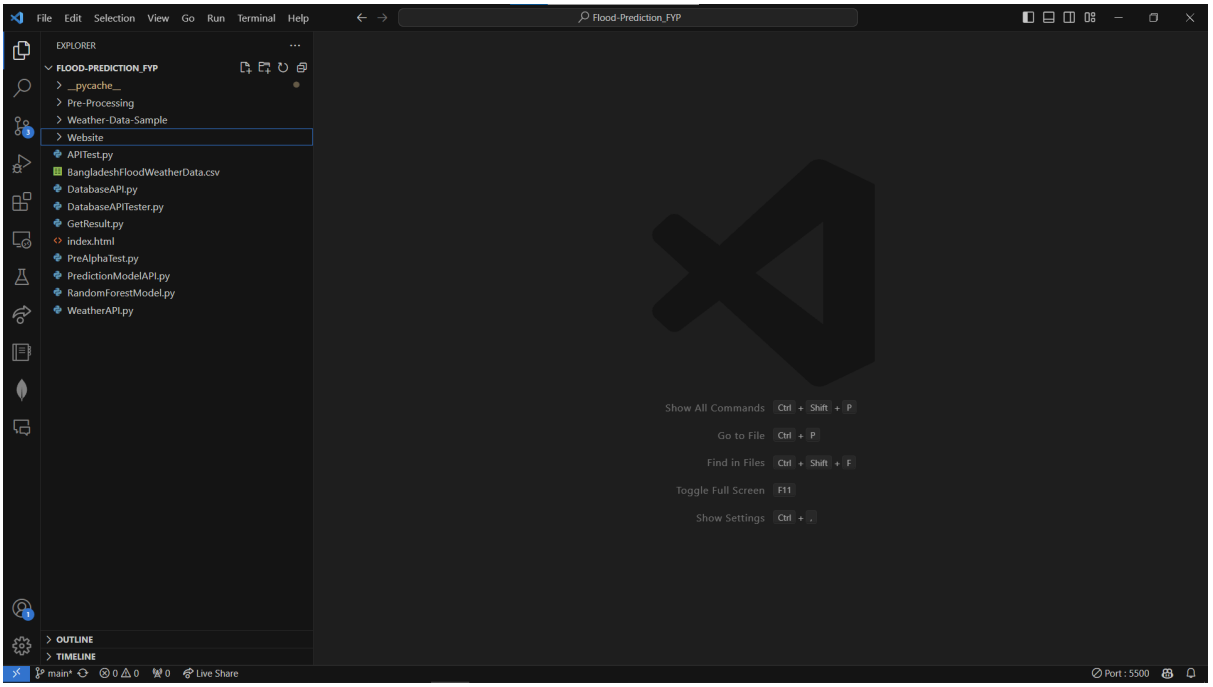


4. Open the folder in an IDE of your choice e.g. Visual Studio Code




Need an IDE? Follow the link to [Download Visual Studio Code - Mac, Linux, Windows]

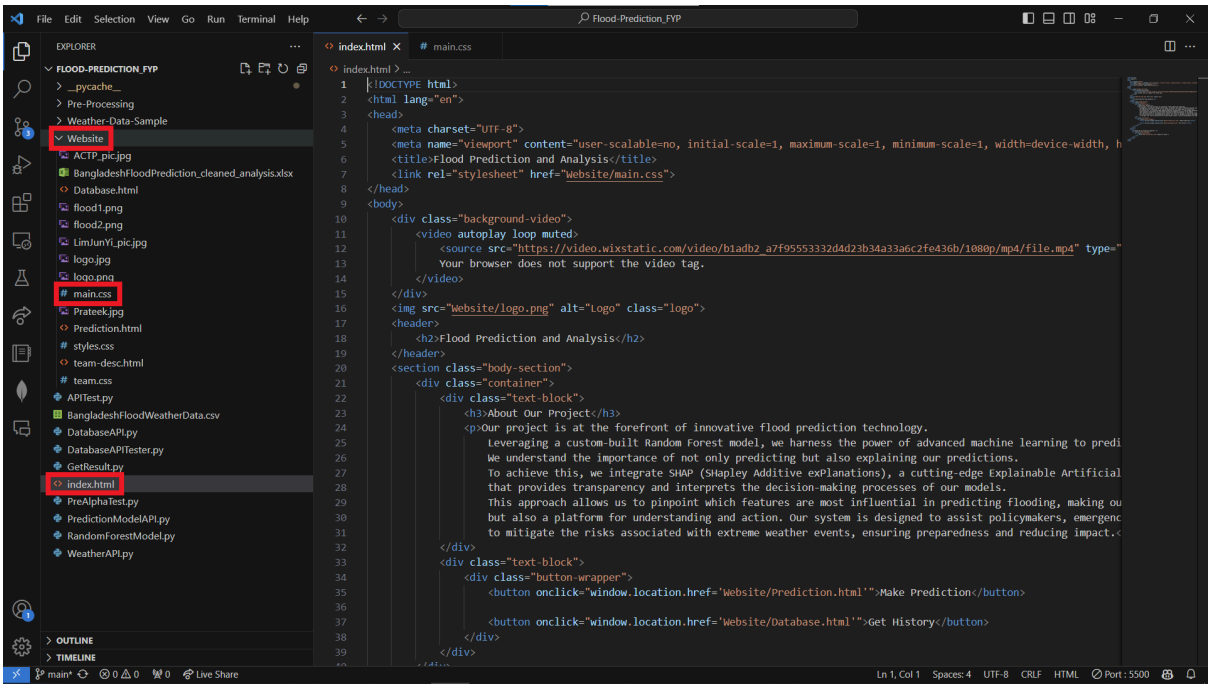
This is what you should expect!



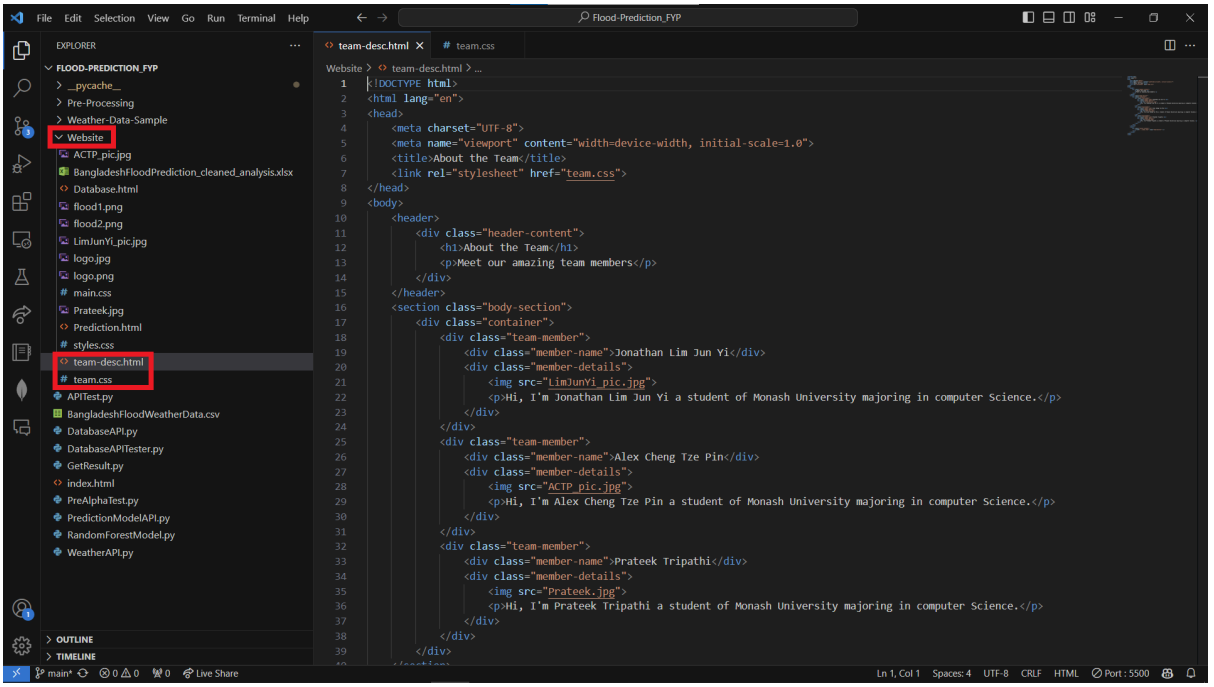
### 2.1.1 Webpage

As shown in the End User Guide previously, there will be 4 web pages for the application:

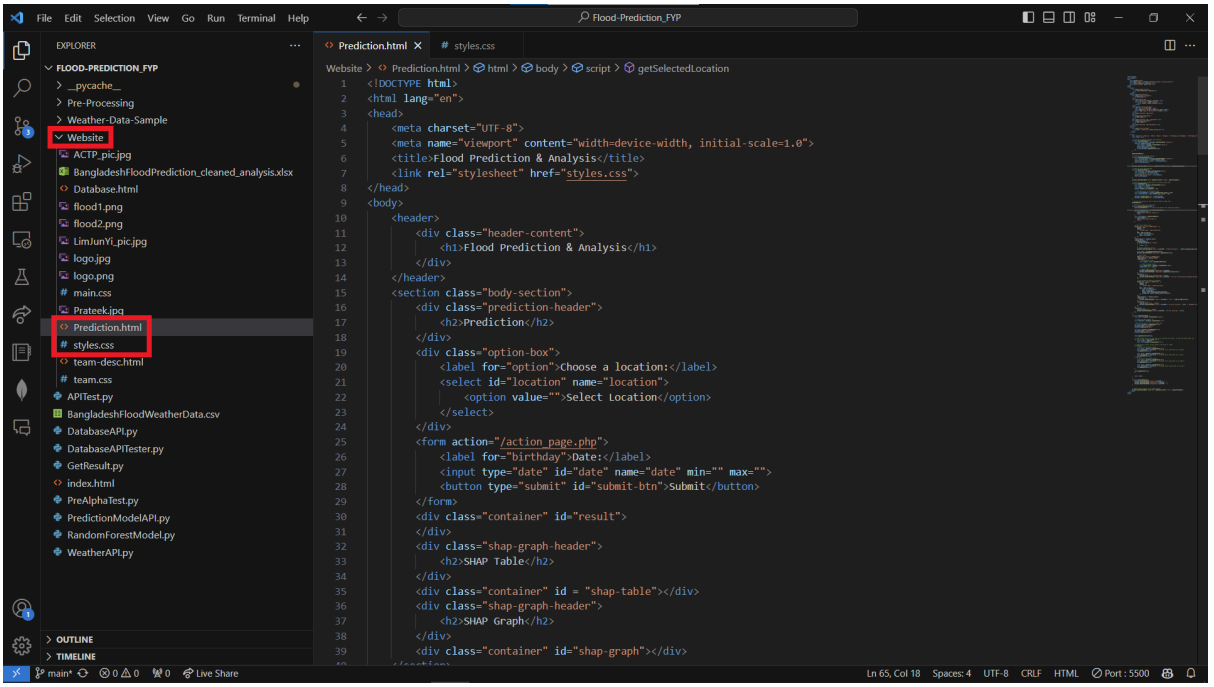
1. Home Page - The source code is stored in a html file called index.html. A CSS file named main.css is used to style the webpage.



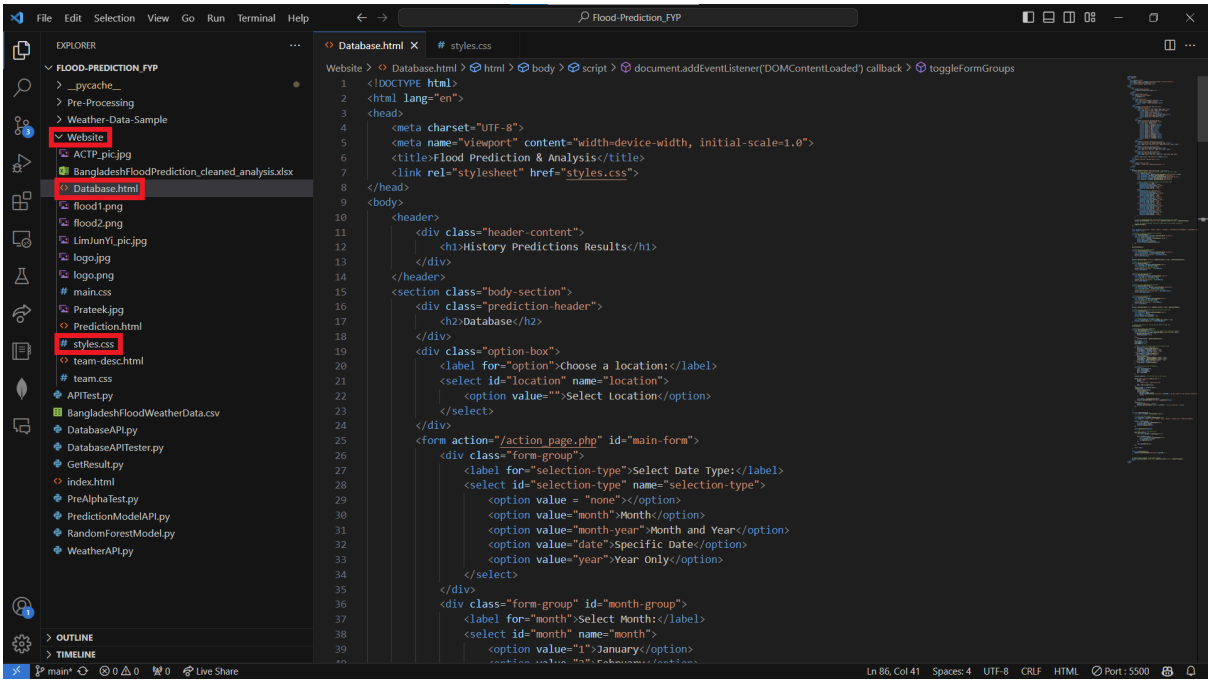

2. Team Description Page - The source code is stored in a html file called team-desc.html. A CSS file named team.css is used to style the webpage.



3. Prediction Page - The source code is stored in a html file called prediction.html. A CSS file named styles.css is used to style the webpage.

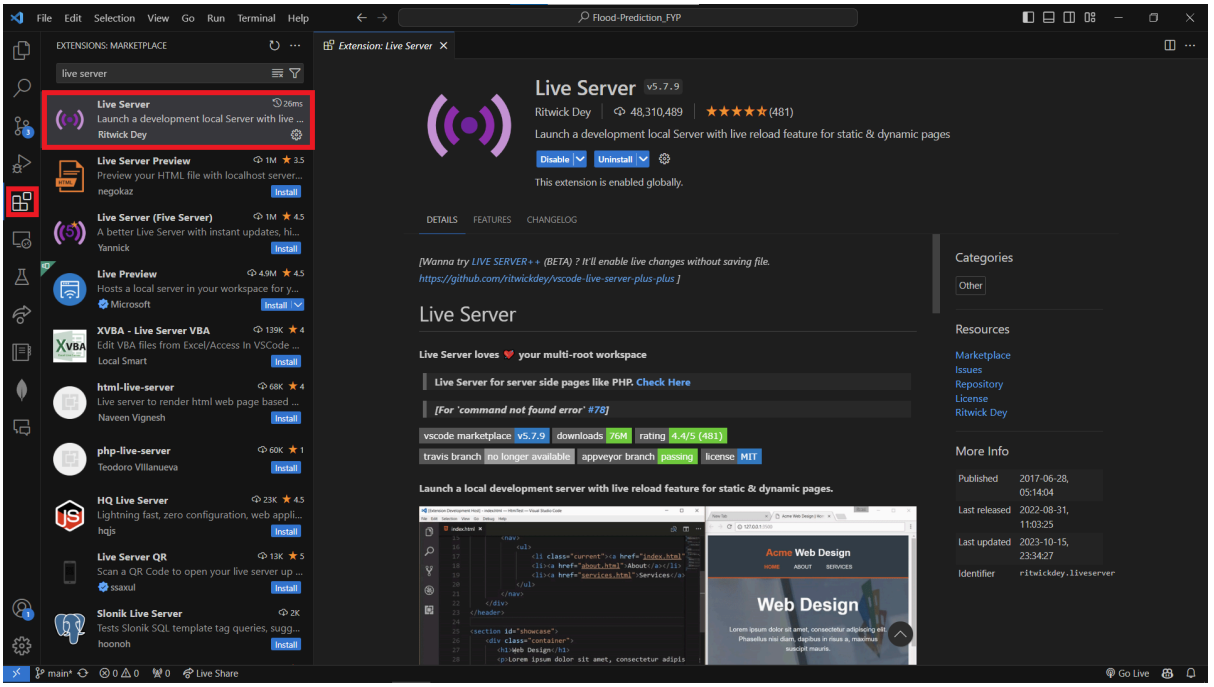



4. Past Results Page - The source code is stored in a html file called database.html. A CSS file named styles.css is used to style the webpage.

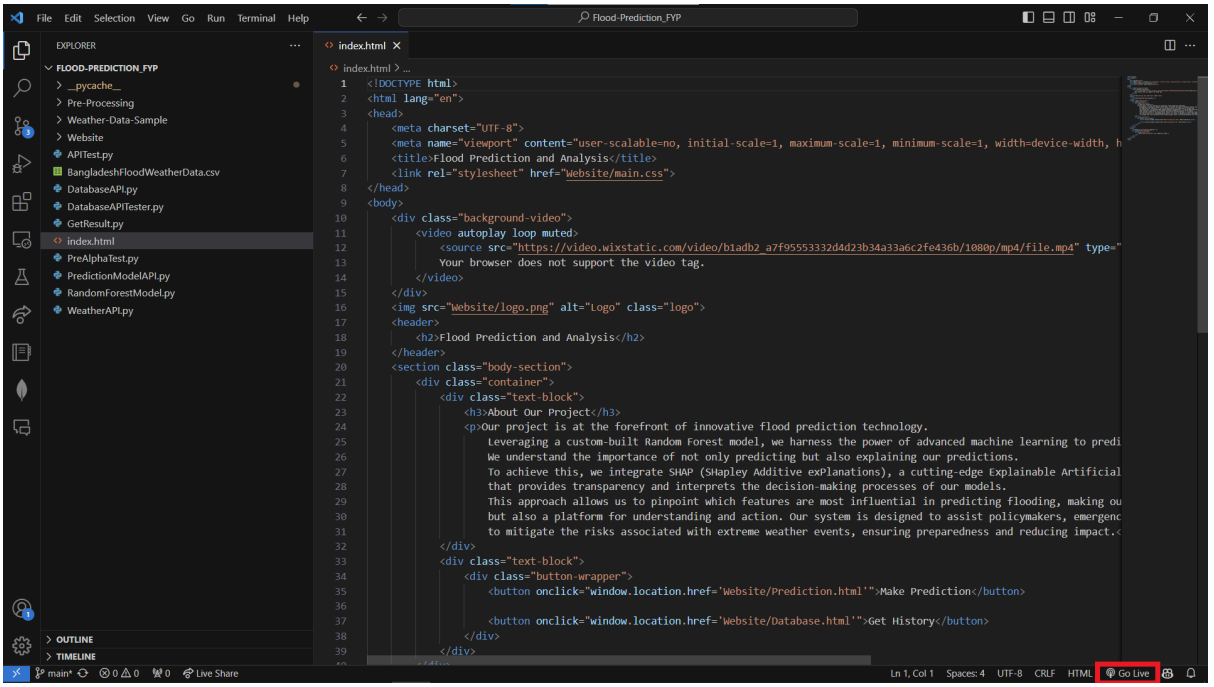


To host the webpage locally, follow these steps:

1. In the Visual Studio Code, download the live server extension



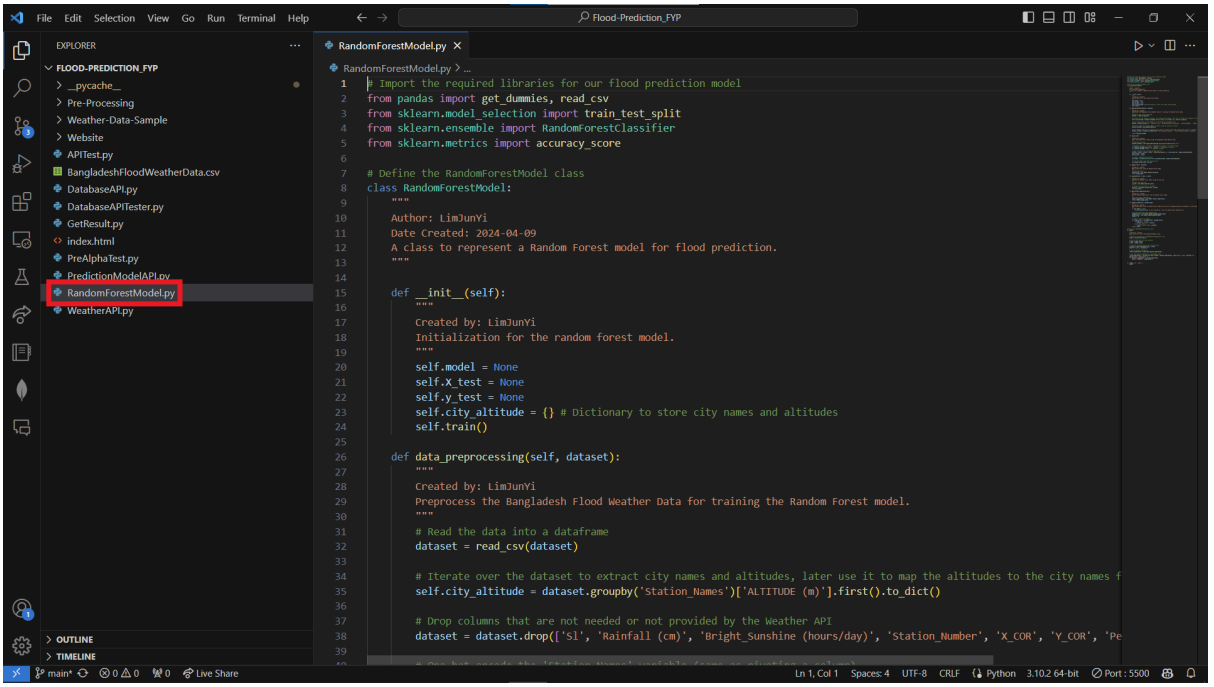

2. Click the Go Live button at the bottom right



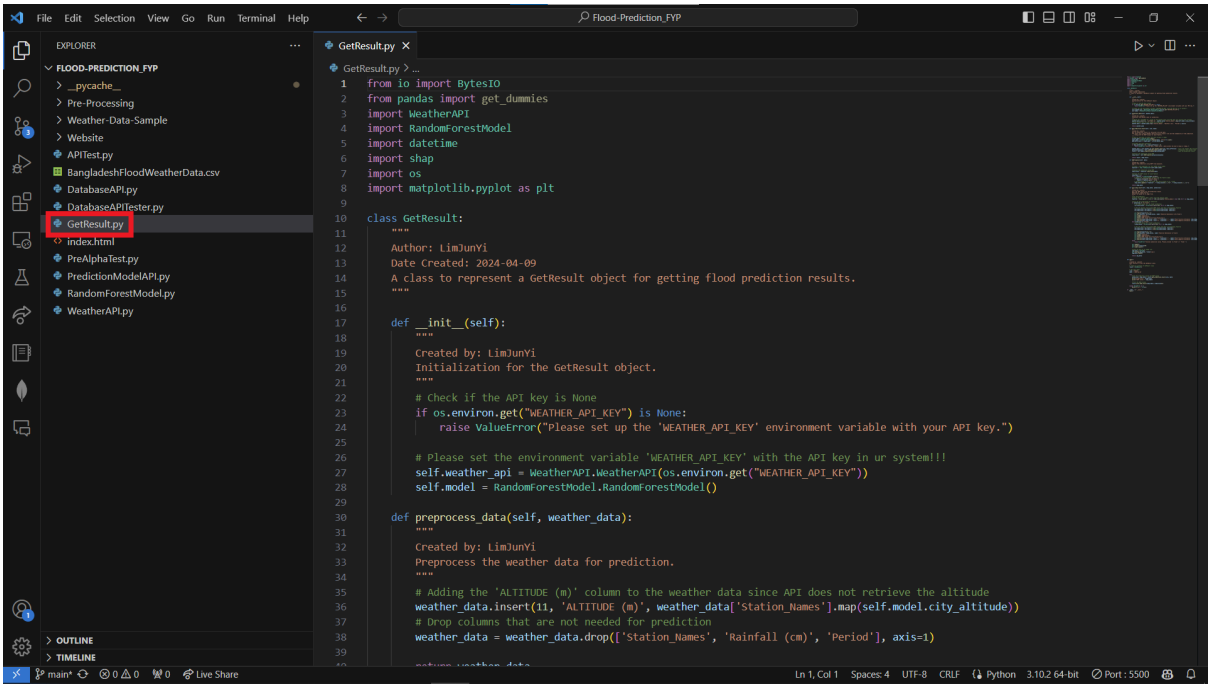
### 2.1.2 Prediction Model

The prediction model consists of two Python files with one being the model itself and another one storing the codes to get the prediction results:

1. RandomForestModel.py - This file defines the Random Forest Model Class to train, test as well as use the prediction model.



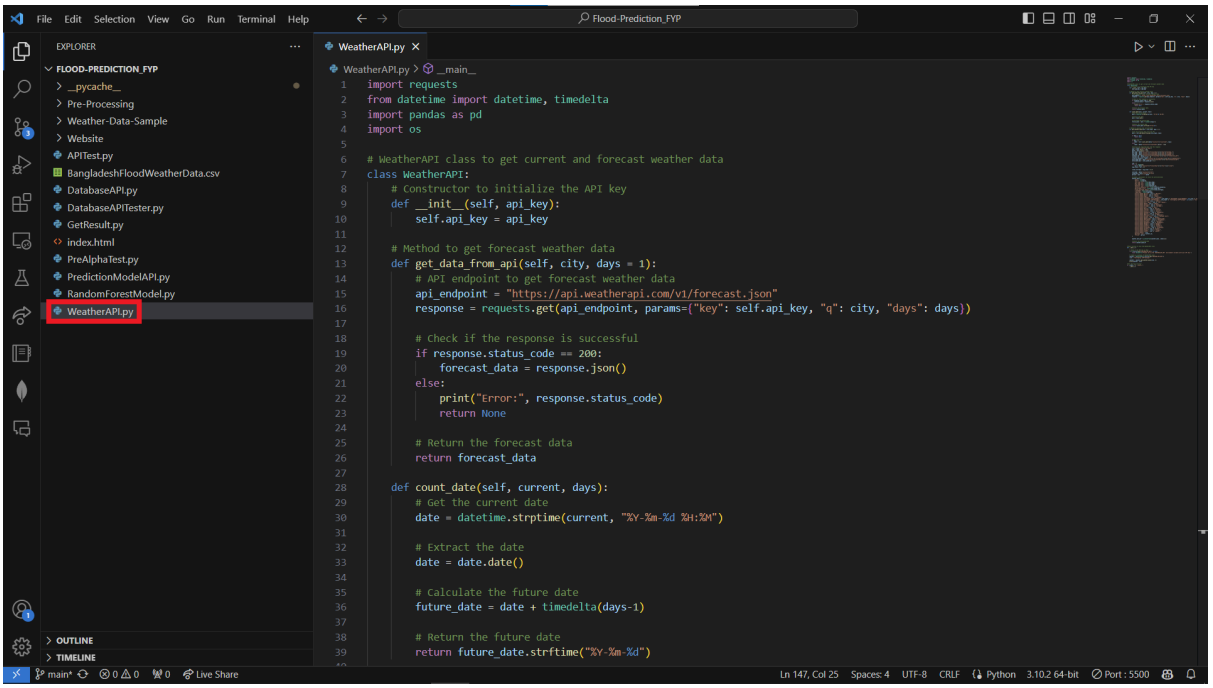

2. GetResult.py - This file contains the codes to get forecast data, make predictions and also the XAI model to describe the importance of the attributes.



Note: If you are not able to run the GetResult.py, please follow the instructions in 2.2 Accessing to the Weather API, then restart your IDE, you should be able to run it without problem (providing the city and date are valid).

## 2.2 Accessing the Weather API

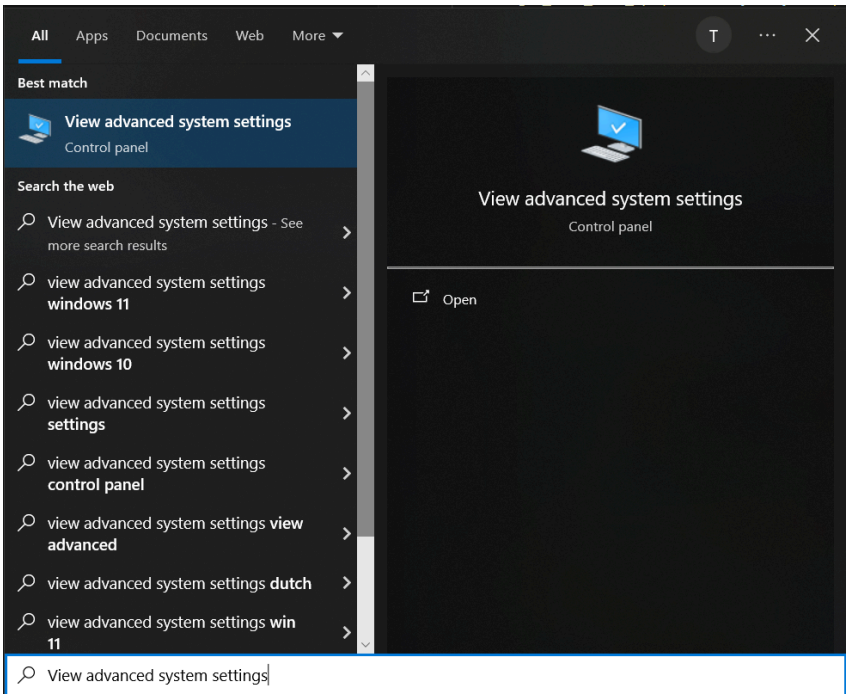
The WeatherAPI class is defined in a Python file named WeatherAPI.py. The file will connect to the WeatherAPI and process the response to fit in the prediction model.



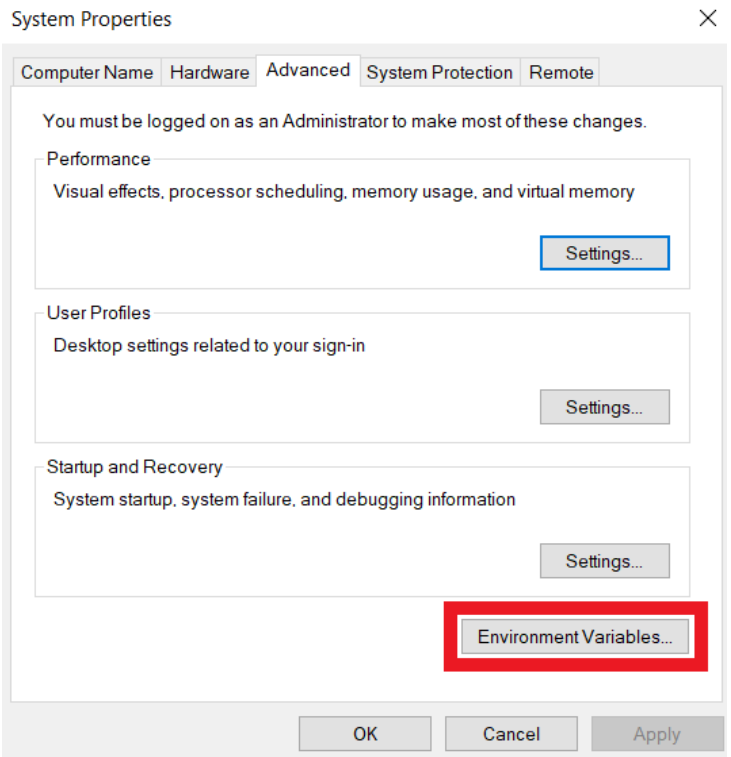

For more information on the WeatherAPI, please refer to Weather and Geolocation API JSON and XML - WeatherAPI.com

To use the Weather API, the administrator will have to do the prerequisite steps below to store the API key locally:

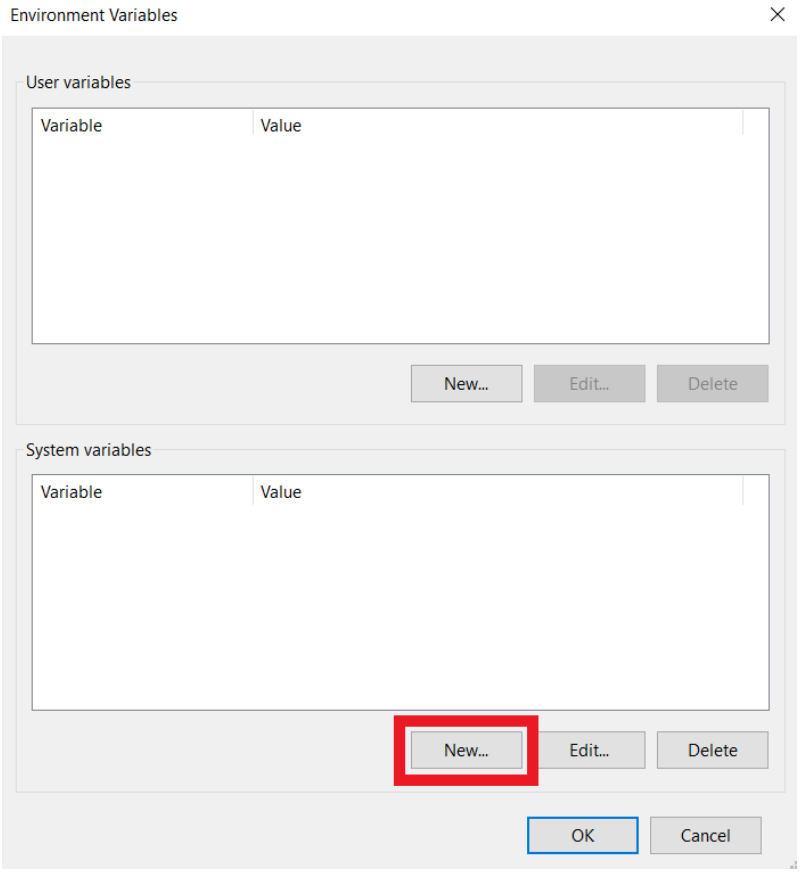
1. Go to “View advanced system settings”



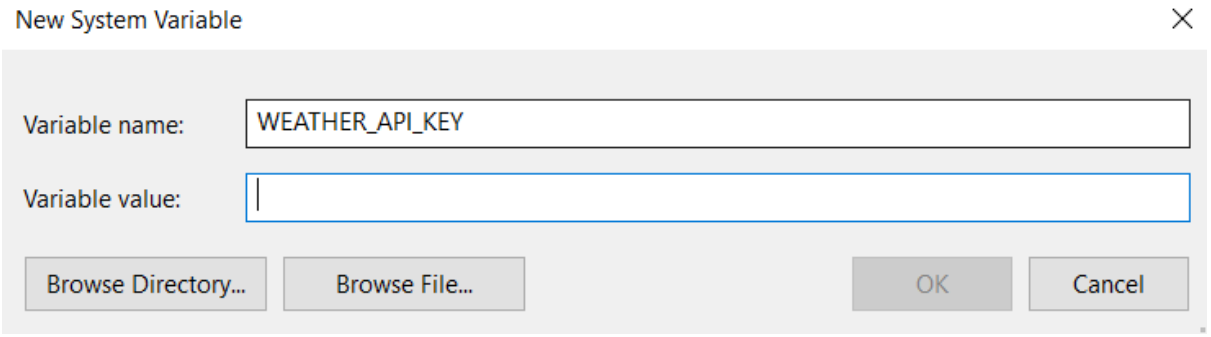

2. Select Environment Variables



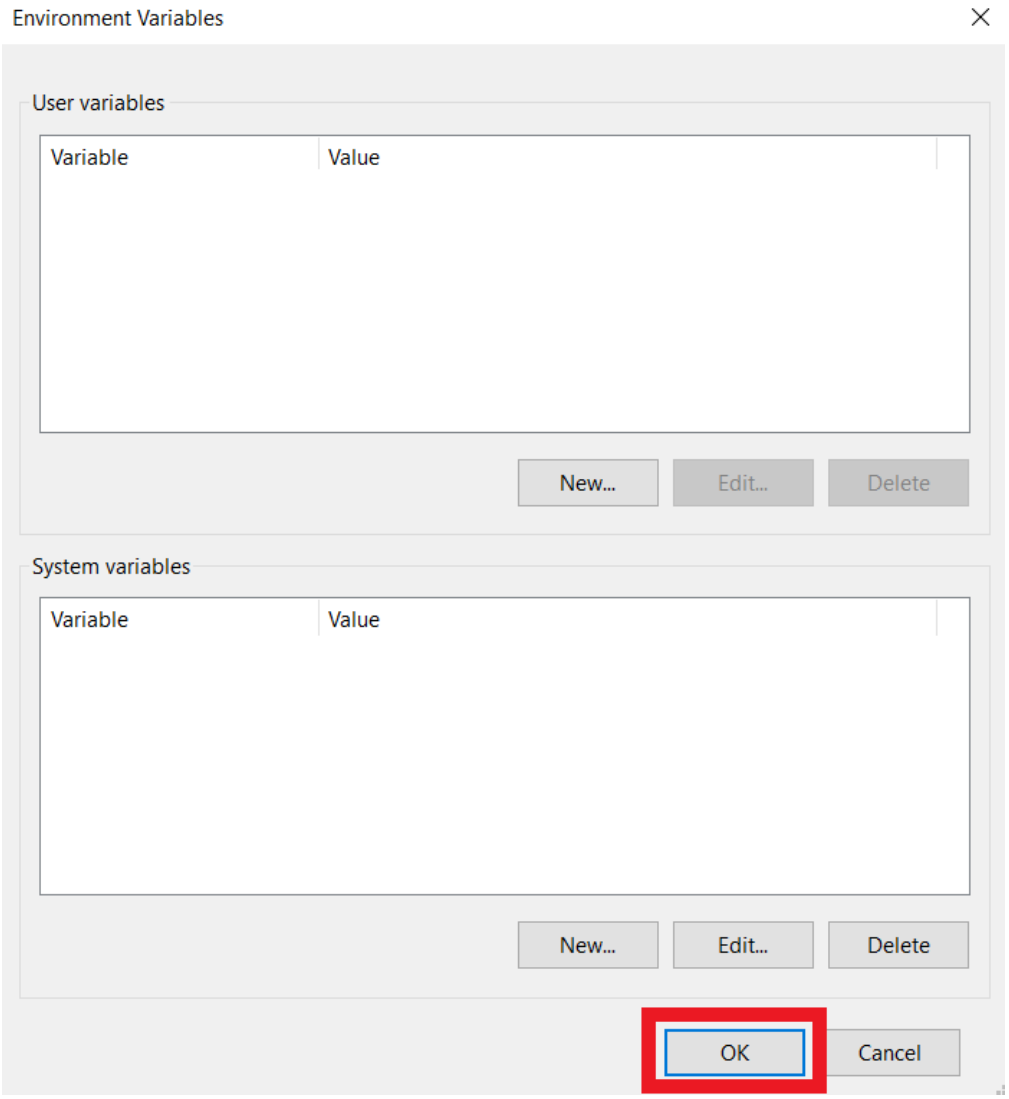
3. Select New for System variables




4. Enter the variable name as “WEATHER_API_KEY” and the key
   Note: The key will be given once you have been assigned as an administrator



5. Click OK

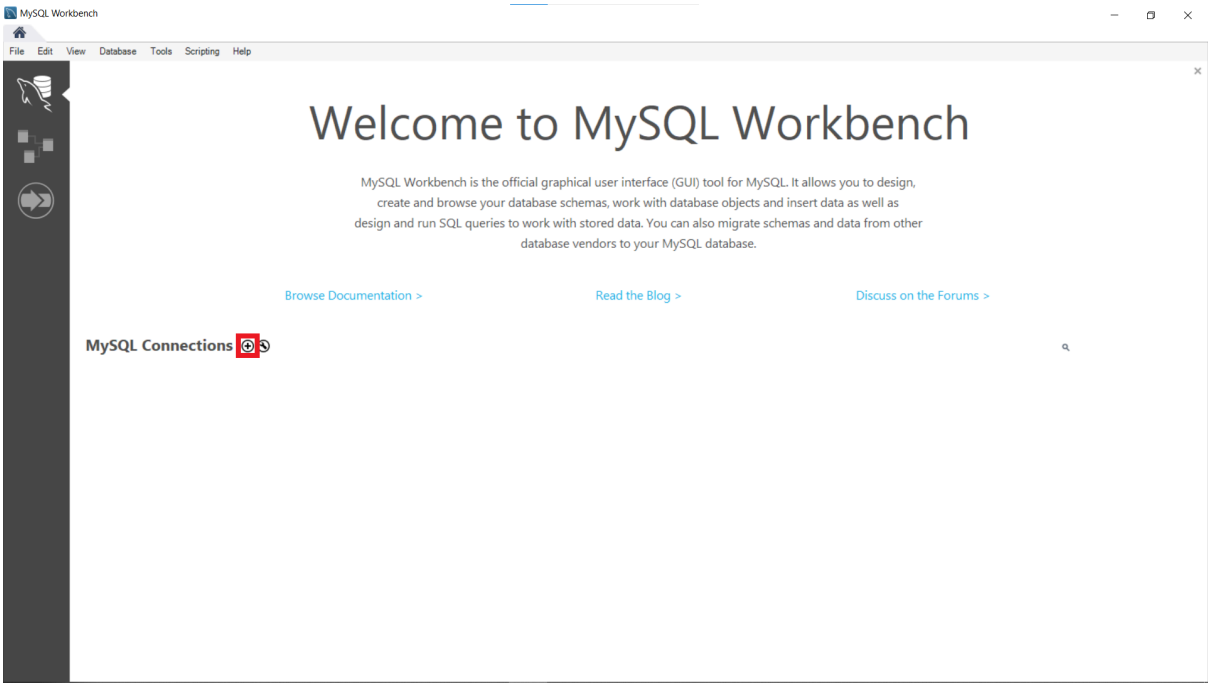



After doing this, you should be able to access the WeatherAPI.

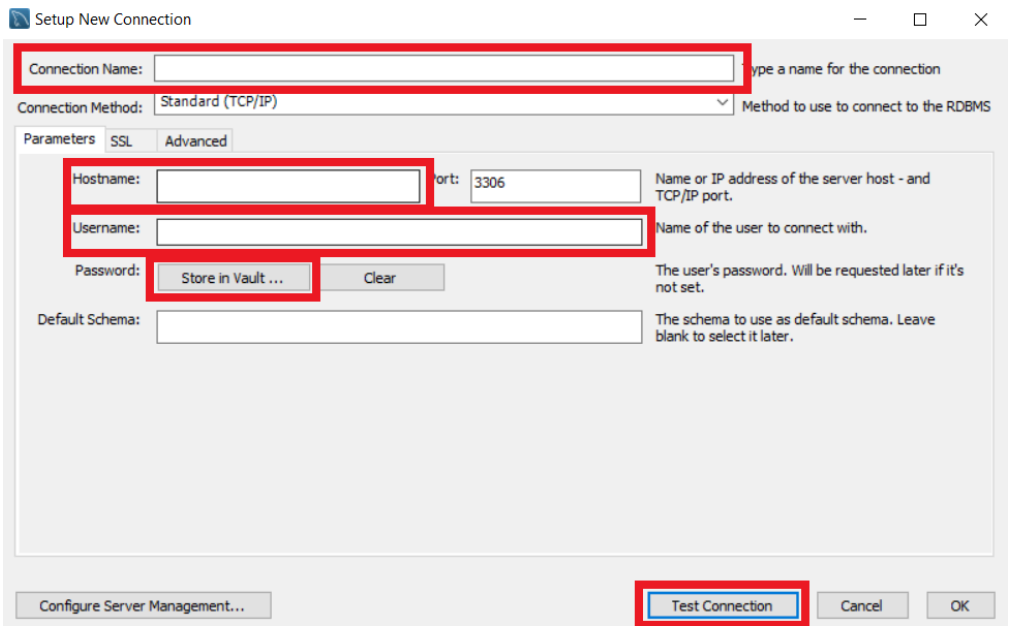
## 2.3 Accessing the MySQL Database

To access the MySQL Database, you will need to follow the steps below:

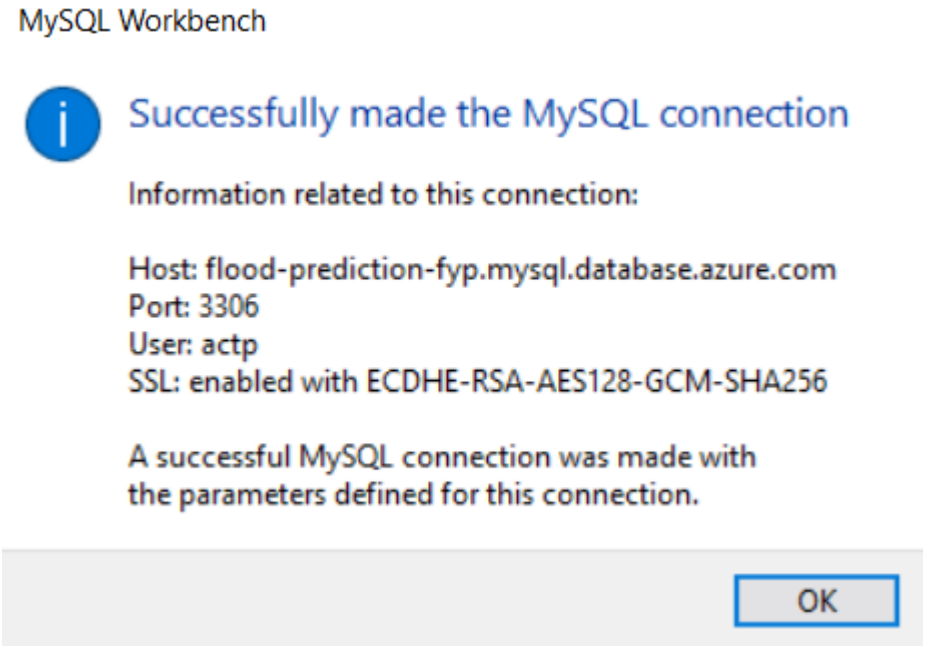
1. Download and Install the MySQL Workbench via Download MySQL Workbench.
   Note: You will need an Oracle account to download. If you do not have one, you can sign up for an account at Oracle | Create Account
2. In the MySQL Workbench, click on the plus sign



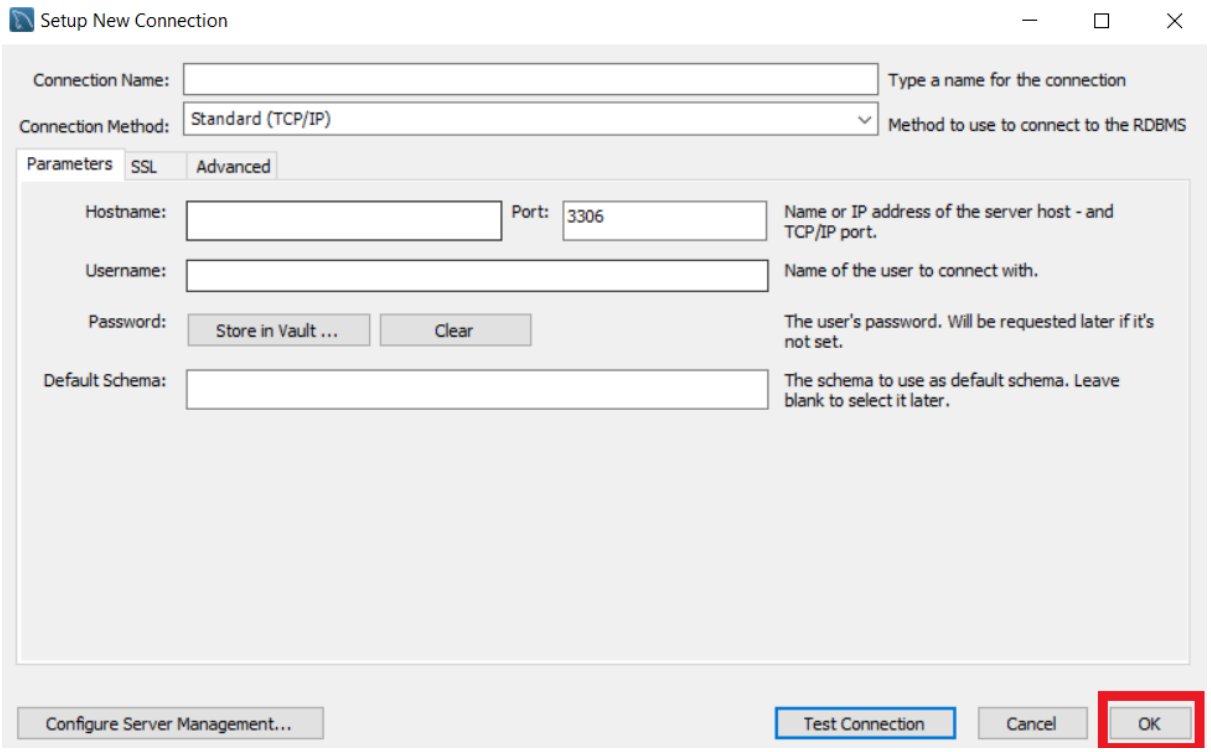

3. Fill in the details below, then click the Test Connection button:
    a. Connection Name: Any name of your choice
    b. Hostname: flood-prediction-fyp.mysql.database.azure.com
    c. Username and password: To be given once you have been assigned as an administrator



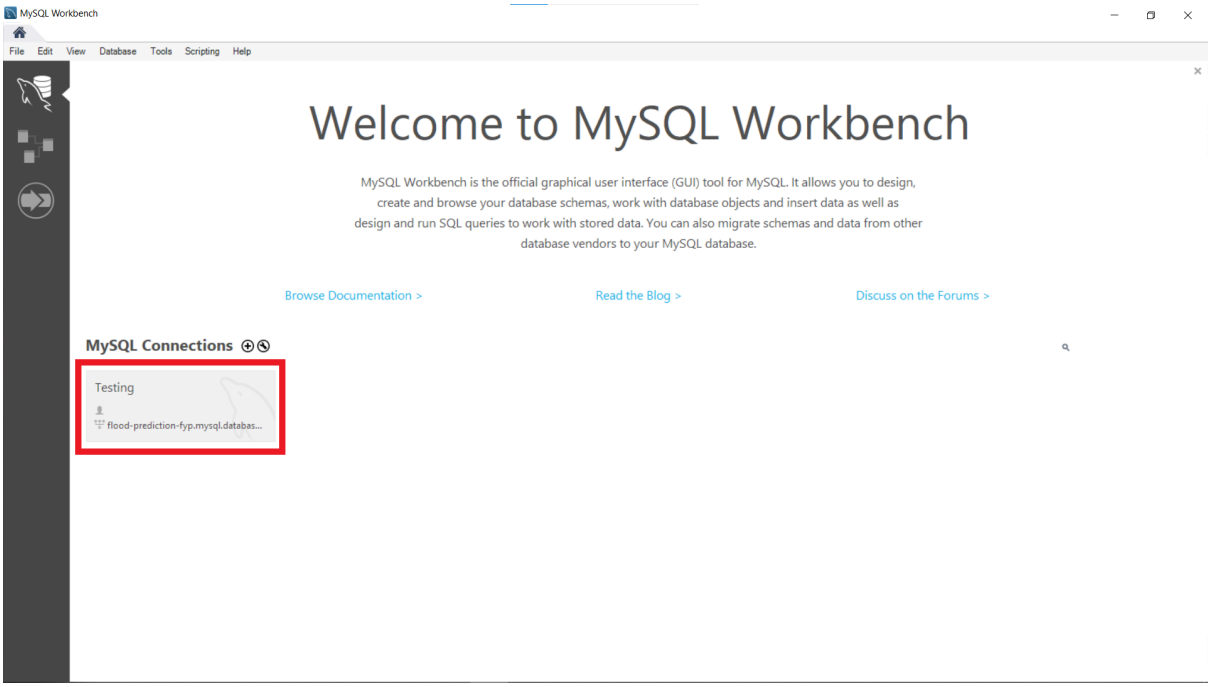

4. You will receive this if you have connected successfully




5. Click the OK button



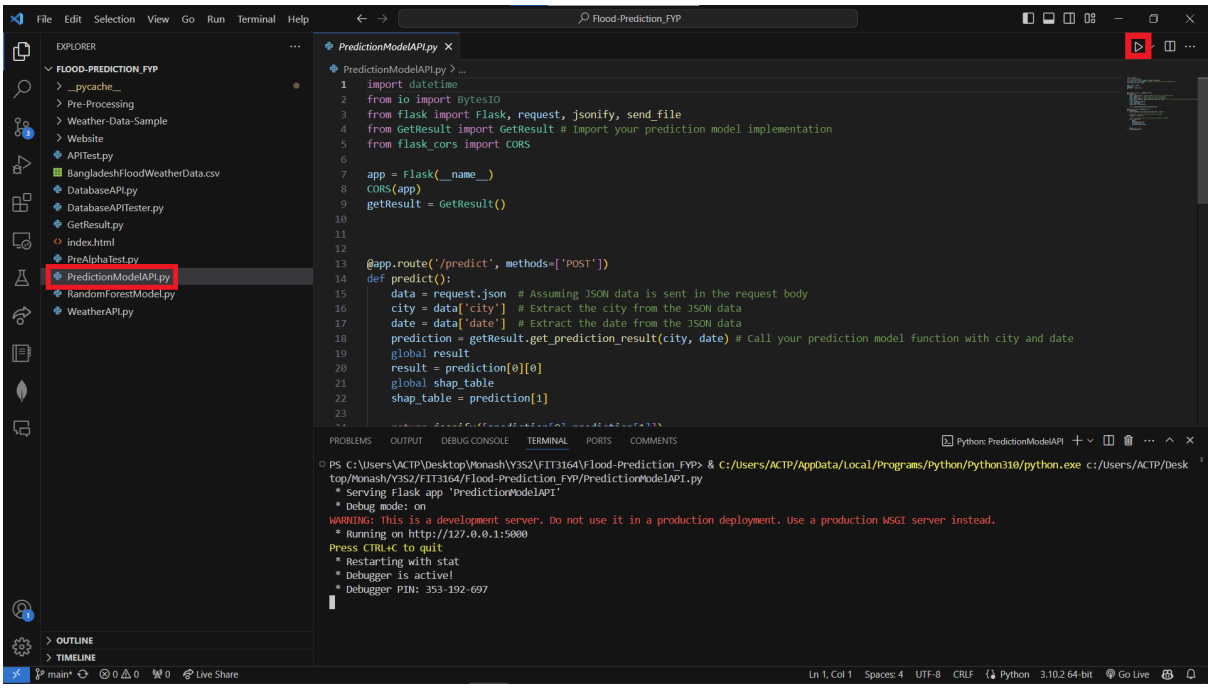

6. The connection will appear on the home page of the workbench




## 2.4 Accessing the Application APIs

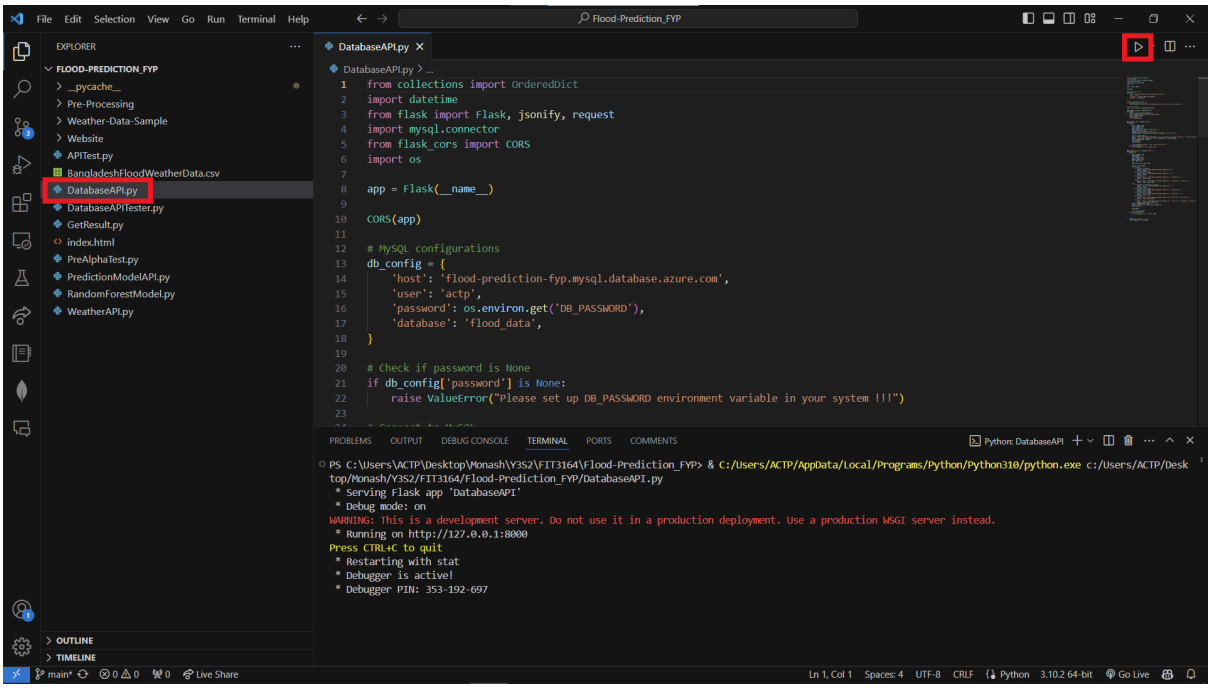
### 2.4.1 Prediction Model API

The source code of the Prediction Model API is stored in a Python file called PredictionModelAPI.py



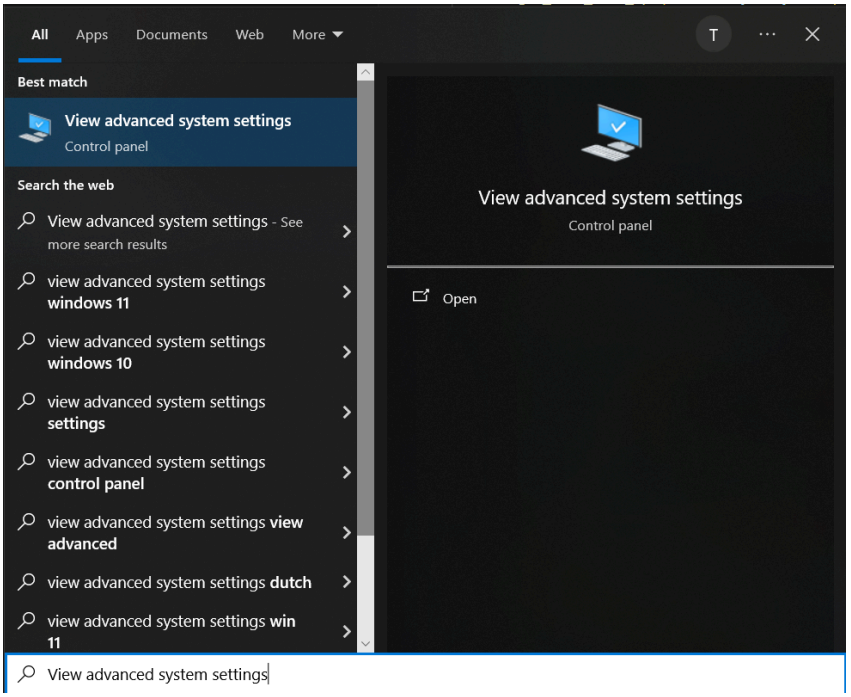

### 2.4.2 Database API

The source code of the Database API is stored in a Python file named DatabaseAPI.py. This file is used to connect the database and the web page. You will have to run the file to use the API.

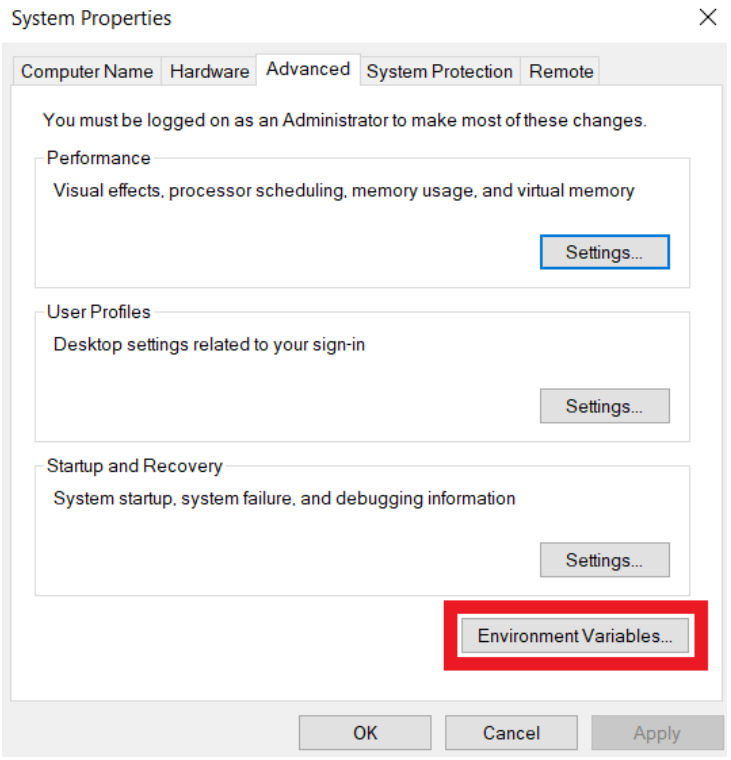



If you have not set up the environment variable, please follow the steps below:

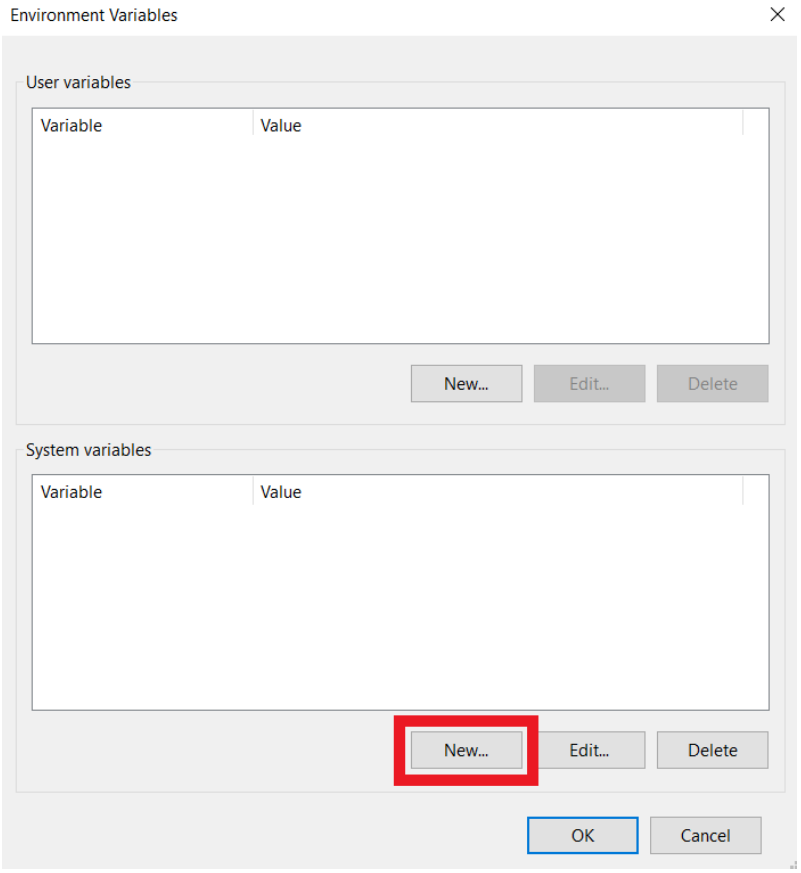
1. Go to “View advanced system settings”



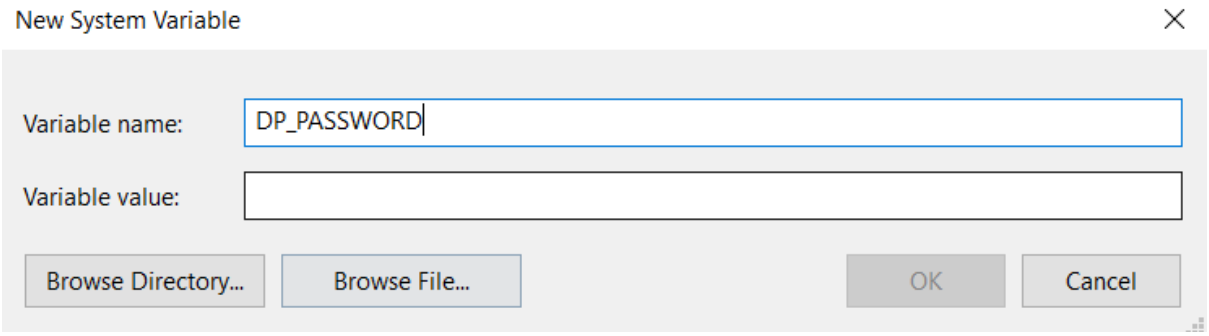
2. Select Environment Variables



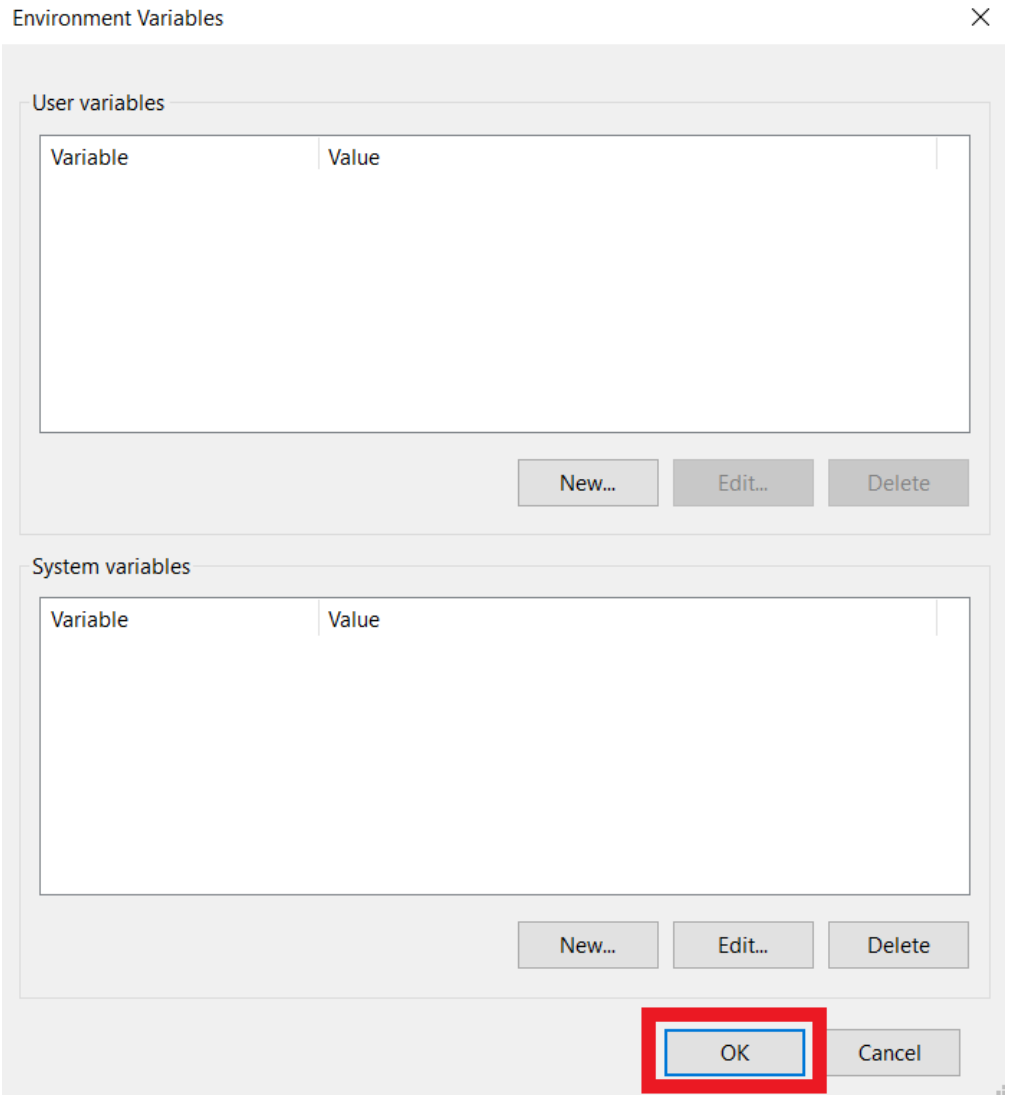

3. Select New for System variables



4. Enter the variable name as “DB_PASSWORD” and the key
   Note: The key will be given once you have been assigned as an administrator



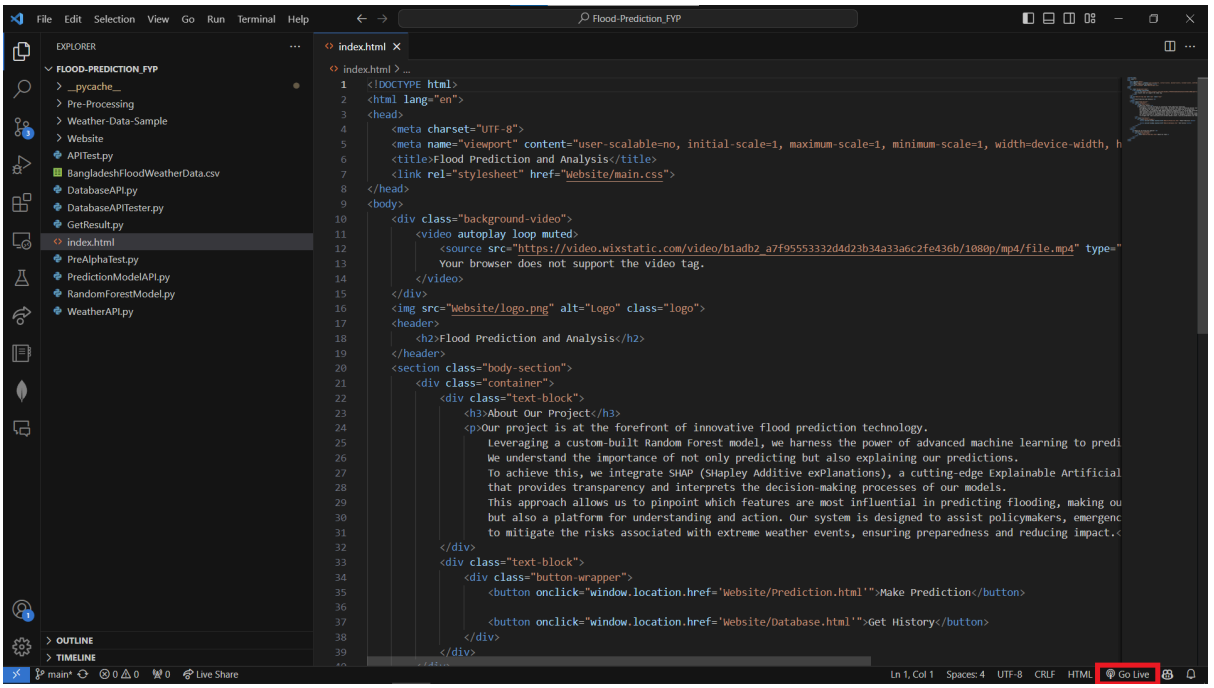

5. Click OK



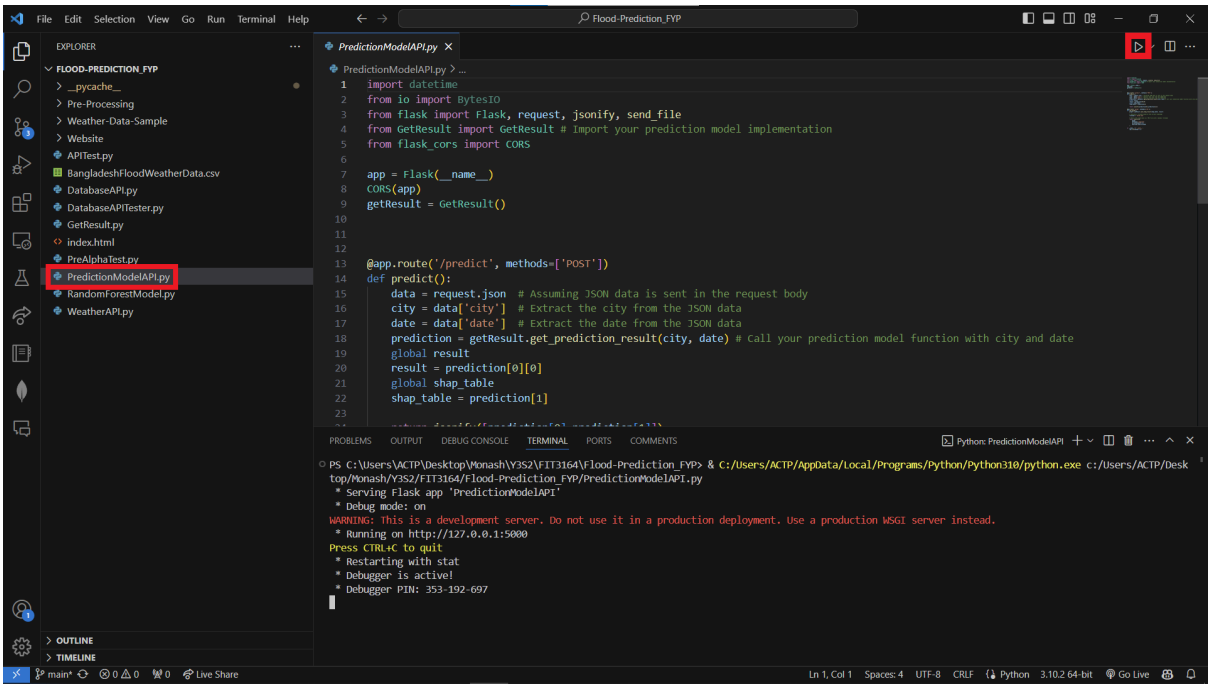
## 2.5 Running the Application as a Whole

There are 3 steps to run the application:

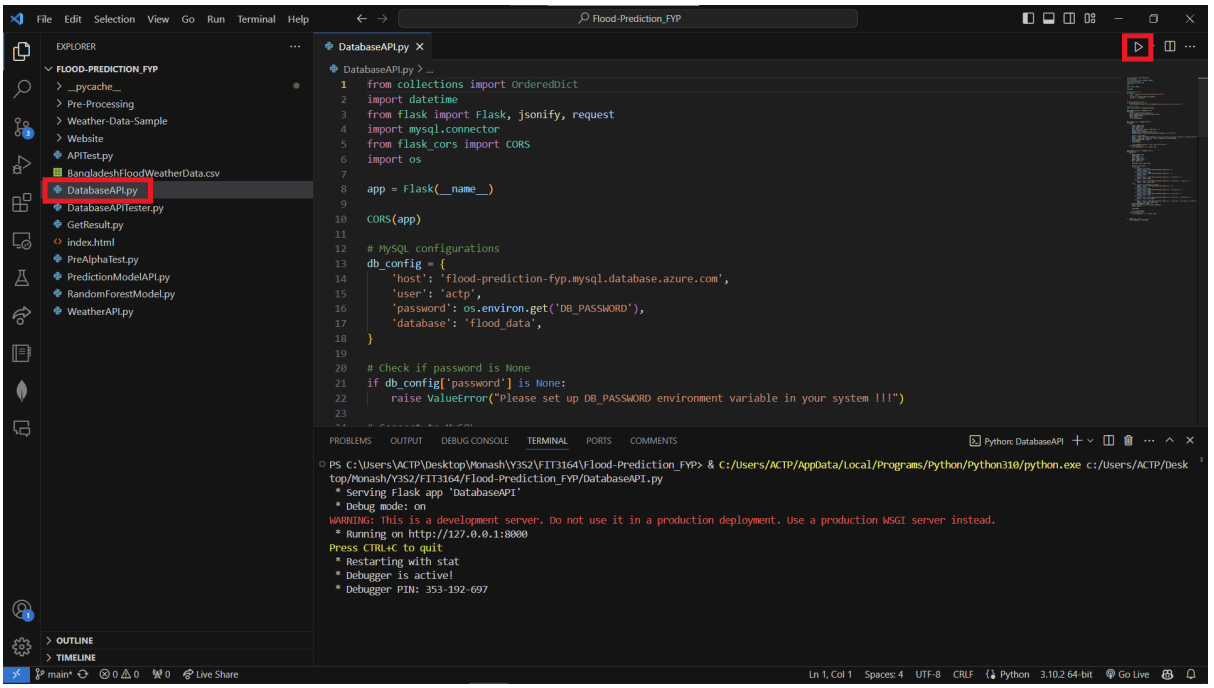
1. Run the live server to host the webpage




2. Run the Prediction Model API



3. Run the Database API



# FIT3164 - Data Science Software Project

# Software Test / QA Report

Prepared by MDS20
Prateek Tripathi 31835643
Lim Jun Yi 32625510
Cheng Tze Pin 32870663

# Table of Contents

# 1. Introduction

In the evolving landscape of software development, assuring the quality and dependability of supplied products is critical. The main objective of our project is to build a predictive model that can predict floods, and be able to explain the key factors responsible for the prediction. To do this, we use the random forest prediction model and SHAP explainable AI (XAI) as they seem to be the candidates that offer the best and most consistent results for the project. The outcome of the project is provided in the form of a website, where a user can specify location and date to get the predictions required. The Test Report provided below is a cornerstone in our pursuit of excellence, precisely developed to assess the software's performance against set requirements and specifications. This study attempts to give a transparent and thorough review of the software's functionality, robustness, and user experience by employing rigorous testing procedures and detailed analyses.

Through this report, we aim to show that the developed software matches the requirements set for the software.

## 2. Description of Test Approach

The approach implemented by our team on our software is Manual Testing. This approach was chosen over an automated testing approach because the software is still relatively in its beginning stages and thus would be unable to profit from automated testing platforms like PyTest, or other such platforms. Our testing is done in 3 parts:  Black Box Testing, Integration Testing and lastly Usability Testing. The Black Box testing is mainly conducted on our website. It focuses on the functional aspect of the software, ensuring that all specified functionalities work as expected. We can validate core features, like user input handling, API integrations, and data retrieval by simulating various user scenarios and comparing the outputs with the expected results. The aim of conducting the integration testing is to ensure all components of the software work together seamlessly. This consists of testing interactions between the front end and back end, communicating with other APIs, and integrating the prediction model into the user experience. Lastly, usability testing is conducted to ensure that the software is intuitive, easy to navigate, and provides a satisfactory user experience. This is conducted by volunteers, who shall be given the link to our website along with our user guide and their comments and feedback are taken into account.

## 3. Black Box Testing

### 3.1: Prediction Page

The **test input screenshots** and the **results screenshots** are added in the **appendix** with the appropriate **S. No**.

| Test S. No | What is being tested? | How is it being tested? | What are the expected outputs? | What are the actual outputs being observed? |
|---|---|---|---|---|
| **3.1** | Error handling for invalid date input. | Submitting invalid input, such as not selecting the date and verifying the system's response. | Upon clicking the submit button an error message is shown that provides a suggested action for the user to take. | After clicking the submit button with an empty date field, the user is prompted to select a date. |
| **3.2** | Error handling for invalid location input. | Submitting invalid input, such as not selecting a location and verifying the system's response. | Upon clicking the submit button an error message is shown that provides a suggested action for the user to take. | After clicking the submit button with an empty location field, the user is prompted to select a location. |
| **3.3** | Display of results for the specific prediction. | Submitting a selected location, and a specified date, then verifying the system’s response. | A text stating the prediction result and the accuracy of the prediction between 0 and 1, where 1 stands for 100% accuracy and 0 stands for 0% accuracy. | After submitting the input, a valid prediction made using the prediction model is shown in the result section, with a confidence score alongside it. |
| **3.4** | Display of SHAP graph for a specific prediction. | Submitting a selected location, and a specified date, then verifying the system’s response | Upon submitting the input, the SHAP graph should be displayed, illustrating the contribution of each feature to the prediction outcome. | After submitting the input, a valid SHAP graph for the selected location and prediction made is outputted |

| Test S. No | What is being tested? | How is it being tested? | What are the expected outputs? | What are the actual outputs being observed? |
|---|---|---|---|---|
| **3.5** | Display of SHAP table for a specific prediction. | Submitting a selected location, and a specified date, then verifying the system's response. | Upon submitting the input, the SHAP table should be displayed, illustrating the contribution of each feature to the prediction outcome. | After submitting the input, a valid SHAP table for the selected location and prediction made is outputted. |

### 3.2: Database Page

Similarly to the above section, the **test input screenshots** and **results screenshots** are added in the **appendix** with the appropriate **S. No**.

| Test S. No | What is being tested? | How is it being tested? | What are the expected outputs? | What are the actual outputs being observed? |
|---|---|---|---|---|
| **3.6** | Retrieval of flood prediction history for a specific month. | Submitting a selected month, and verifying the system's response. | Upon clicking the submit button, all the predictions made for the specified month that have been stored in the database will be shown | After submitting the month, the user is shown the predictions made related to the specified month, regardless of location |
| **3.7** | Retrieval of flood prediction history for a specific month and year. | Submitting a selected month and year, then verifying the system's response. | Upon clicking the submit button, all the predictions made for the specified month in the specified year, that have been stored in the database, will be shown. | After submitting the specified month and the year, the user is shown the predictions made related to the specified details, regardless of location. |
| **3.8** | Retrieval of flood prediction history for a specific year. | Submitting a selected year, then verifying the system's response. | Upon clicking the submit button, all the predictions made for the specified year, that have been stored in the database, will be shown. | Upon clicking the submit button, all the predictions made for the specified year, that have been stored in the database, will be shown. |
| **3.9** | Retrieval of flood prediction history for a specific date. | Submitting a selected date, then verifying the system's response. | Upon clicking the submit button, all the predictions for the specified date, in this case, 1st May 2024, that have been stored in the database, will be shown. | After submitting the specified date, the user is shown the predictions made related to the specified date, regardless of location |
| **3.10** | Retrieval of flood prediction history for a specified location in a specific month. | Submitting a selected location, and a selected month, with appropriate inputs, then verifying the system's response. | Upon clicking the submit button, all the predictions made for the specified location in the specified month, that have been stored in the database, will be shown. | After submitting the specified month and location, the user is shown the predictions made in the specified month, for the specified location. |

| Test S. No | What is being tested? | How is it being tested? | What are the expected outputs? | What are the actual outputs being observed? |
|---|---|---|---|---|
| **3.11** | Retrieval of flood prediction history for a specified location in a specific month and year. | Submitting a selected location, and a selected month and year, with appropriate inputs, then verifying the system's response. | Upon clicking the submit button, all the predictions made for the specified location in the specified month and year, that have been stored in the database, will be shown. | After submitting the specified location, month and year, the user is shown the predictions made in the specified month and year, for the specified location. |
| **3.12** | Retrieval of flood prediction history for a specified location in a specific year. | Submitting a selected location, and a selected year, with appropriate inputs, then verifying the system's response. | Upon clicking the submit button, all the predictions made for the specified location in the specified year, that have been stored in the database, will be shown. | After submitting the specified location and year, the user is shown the predictions made in the specified year, for the specified location. |
| **3.13** | Retrieval of flood prediction history for a specified location on a specific date. | Submitting a selected location, and a selected date, with appropriate inputs, then verifying the system's response. | Upon clicking the submit button, all the predictions made for the specified location on the specified date, that have been stored in the database, will be shown. | After submitting the specified location and date, the user is shown the predictions made on the specified date, for the specified location. |
| **3.14** | Retrieval of flood prediction history for a specified location. | Submitting a selected location, with appropriate inputs, then verifying the system's response. | Upon clicking the submit button, all the predictions made for the specified location, that has been stored in the database, will be shown. | After submitting the specified location, the user is shown all the predictions made for the specified location that are available in the database. |

## 4. Integration Testing

Integration testing is an essential step in the software testing life cycle that focuses on ensuring that different modules and components of a system work together seamlessly. The goal of integration testing is to identify any discrepancies or issues that may arise when individual units of the application are combined and to validate that they work together as expected.

Given the complexity of our application, which involves making API calls to retrieve weather information, processing this data through a trained prediction model, and displaying the results on the user interface, integration testing helps confirm that all these components function cohesively.

The key objectives of conducting the integration testing are:

1. To confirm that the inputs, provided by the user are accurately received and processed by the backend.
2. To confirm that the back-end results are correctly displayed in the front-end.
3. To ensure any errors or exceptions are handled gracefully and communicated properly to the user.

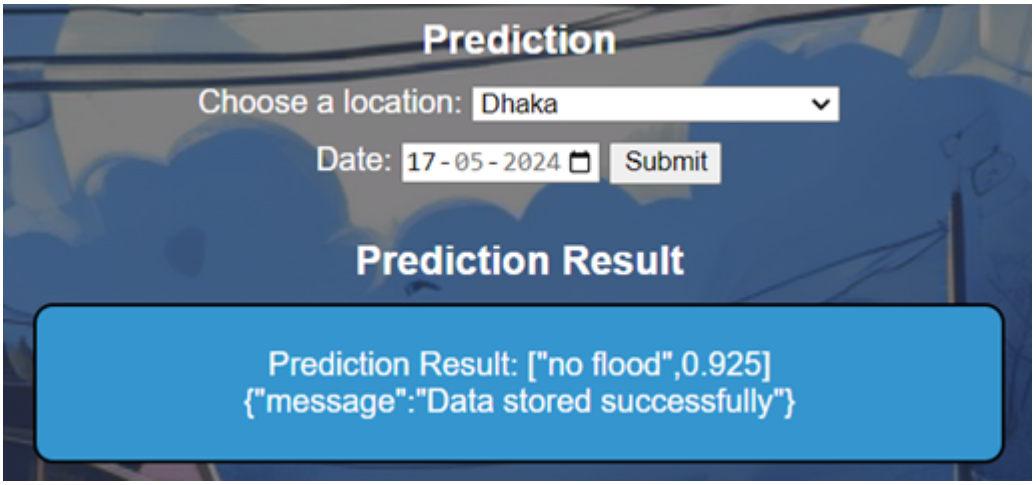
## Test Execution:

**Case 1:** Prediction Request

**Objective:** Verify that the prediction request sent from the front-end is correctly received and processed by the back-end and that the prediction result is displayed accurately on the website. Additionally, test to compare the SHAP table and graph output to ensure they align with the prediction results.

Input: {"city": "Dhaka", "date": "2024-05-17"}

**The result on the website:**



## SHAP Table

The table lists SHAP (SHapley Additive exPlanations) values for various features in a predictive model, indicating how each feature contributes to the model's predictions. A positive SHAP value means a feature increases the prediction, while a negative value indicates a decrease. The magnitude of these values shows the strength of their impact. The features range from continuous variables like temperature to categorical variables like geographical locations, each affecting the model's output uniquely. By analyzing these values, we an understand which features are most important and how they influence the predictions.

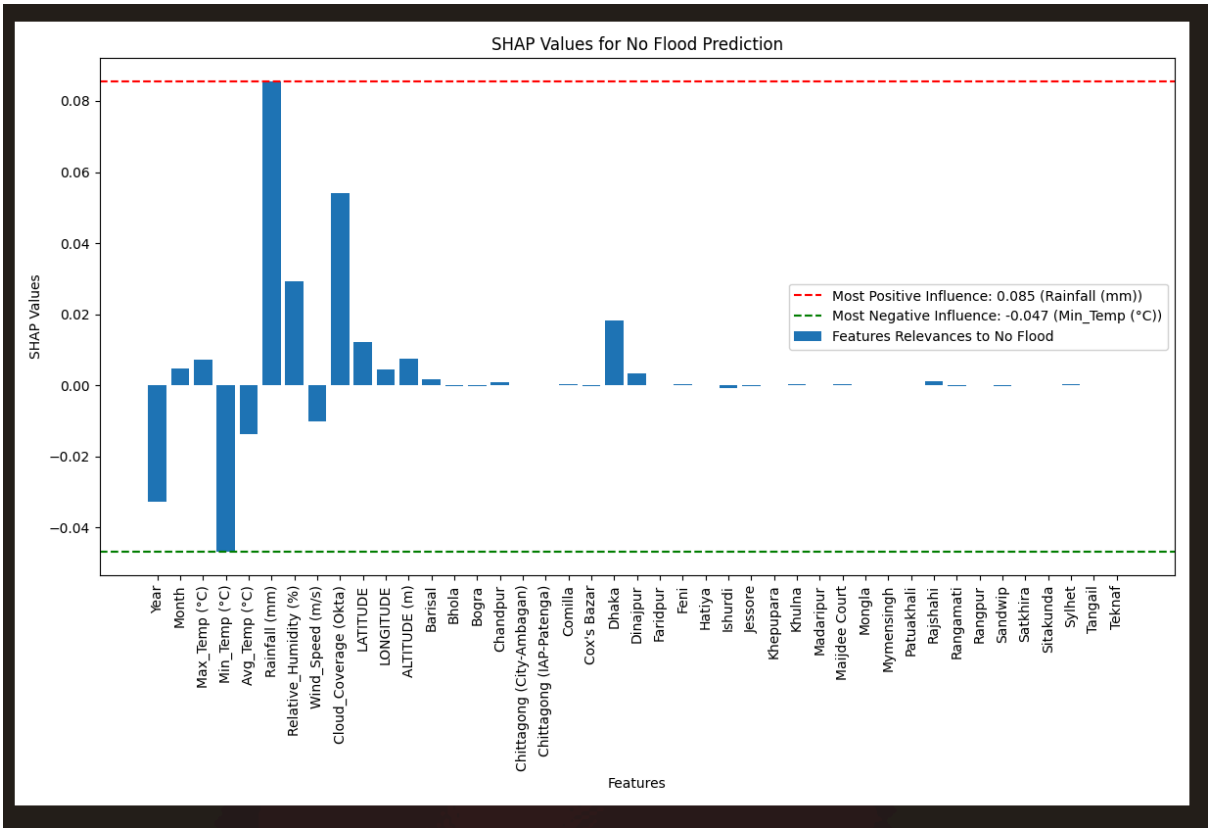
| Attributes | SHAP Value |
|---|---|
| Year | -0.032785166650268614 |
| Month | 0.00485640412325707 |
| Max_Temp (°C) | 0.007289811799944202 |
| Min_Temp (°C) | -0.046768707547127204 |
| Avg_Temp (°C) | -0.013590238911768773 |
| Rainfall (mm) | 0.08542030978541362 |
| Relative_Humidity (%) | 0.029356311428046 4 |
| Wind_Speed (m/s) | -0.01020475202618718 |
| Cloud_Coverage (Okta) | 0.05418178662060897 |
| LATITUDE | 0.012209533251422692 |
| LONGITUDE | 0.004562488998769671 |
| ALTITUDE (m) | 0.007578035933853639 |
| Barisal | 0.0018004175423582452 |
| Bhola | -0.00022031705726421354 |
| Bogra | -0.0002950130290760619 |
| Chandpur | 0.0008168883746556891 |
| Chittagong (City-Ambagan) | -6.440060356662434e-06 |
| Chittagong (IAP-Patenga) | 0.0001290078857151203 |
| Comilla | 0.0002356309997162605 |
| Cox's Bazar | -0.00015559236073517215 |
| Dhaka | 0.018342897935195373 |
| Dinajpur | 0.0033350728810156837 |
| Faridpur | 7.024255623783972e-05 |
| Feni | 0.00022611269523244067 |
| Hatiya | 0.00013424720086465067 |
| Ishurdi | -0.0006756957299972381 |
| Jessore | -0.00023127029684351282 |
| Khepupara | -4.0371005262369476e-05 |
| Khulna | 0.000256869518484564 |
| Madaripur | 7.526346196346176e-05 |
| Maijdee Court | 0.000279639065006994 |
| Mongla | 7.18048324936524e-05 |
| Mymensingh | 0.00012171347551038032 |
| Patuakhali | -2.173187567451481e-06 |
| Rajshahi | 0.001151619167943617 |
| Rangamati | -7.616857094297193e-05 |
| Rangpur | 4.5142167273525725e-06 |
| Sandwip | -0.00011003839740715541 |
| Satkhira | 0.0001797183926550974 |
| Sitakunda | -3.5703375149143216e-05 |
| Sylhet | 0.00037497505105241267 |
| Tangail | 2.396214167047119e-06 |
| Teknaf | 6.436241690373735e-05 |

**API test code:**

```python
import requests
from PIL import Image
from io import BytesIO

# Define the URLs of your Flask API endpoints
predict_url = 'http://127.0.0.1:5000/predict'
graph_url = 'http://127.0.0.1:5000/graph'

# Define the input data (city and date)
data = {
    'city': 'Dhaka',
    'date': '2024-05-17'
}

# Send a POST request to the Flask API endpoint to get the prediction
response = requests.post(predict_url, json=data)

# Check if the prediction request was successful (status code 200)
if response.status_code == 200:
    # Print the predicted result
    prediction = response.json()
    print('Prediction Result:', prediction)

    # If the prediction was successful, fetch the SHAP graph
    graph_response = requests.get(graph_url)

    # Check if the graph request was successful (status code 200)
    if graph response.status_code == 200:
        # Display the SHAP graph using PIL
        img = Image.open(BytesIO(graph_response.content))
        img.show()  # This will open the default image viewer and display the image
    else:
        # Print an error message if the graph request was not successful
        print('Error fetching graph:', graph_response.text)
else:
    # Print an error message if the prediction request was not successful
    print('Error fetching prediction:', response.text)
```

## API test result:

```
PS C:\Users\Prateek Tripathi\Desktop\Flood prediction\Flood-Prediction_FYP> & "C:/Users/Prateek Tripathi/AppData/Local/Microsoft/WindowsApps/python3
.11.exe" "c:/Users/Prateek Tripathi/Desktop/Flood prediction/Flood-Prediction_FYP/APITest.py"
Prediction Result: [['no flood', 0.925], [['Year', '-0.032785166650268614', '0.032785166650268774'], ['Month', '0.00485640412325707', '-0.0048564041
232570555'], ['Max_Temp (°C)', '0.007289811799944202', '-0.007289811799944146'], ['Min_Temp (°C)', '-0.046768707547127204', '0.0467687075471277 2'],
['Avg_Temp (°C)', '-0.013590238911768773', '0.013590238911768871'], ['Rainfall (mm)', '0.08542030978541362', '-0.08542030978541253'], ['Relative_Hum
idity (%)', '0.0293563114280464', '-0.029356311428046337'], ['Wind_Speed (m/s)', '-0.01020475202618718', '0.01020475202618724 6'], ['Cloud_Coverage (
Okta)', '0.05418178662060897', '-0.05418178662060899'], ['LATITUDE', '0.012209533251422692', '-0.012209533251422915'], ['LONGITUDE', '0.004562488998
769671', '-0.0045624889987697015'], ['ALTITUDE (m)', '0.007578035933853639', '-0.007578035933853607'], ['Barisal', '0.0018004175423582452', '-0.0018
004175423582773'], ['Bhola', '-0.00022031705726421354', '0.00022031705726421 42'], ['Bogra', '-0.0002950130290760619', '0.0002950130290760566 7'], ['C
handpur', '0.0008168883746556891', '-0.0008168883746556692'], ['Chittagong (City-Ambagan)', '-6.440060356662434e-06', '6.440060356662149e-06'], ['Chi
ttagong (IAP-Patenga)', '0.0001290078857151203', '-0.0001290078857151189 2'], ['Comilla', '0.0002356309997162605', '-0.0002356309997162590 7'], ["Cox'
s Bazar", '-0.00015559236073517215', '0.00015559236073516914'], ['Dhaka', '0.018342897935195373', '-0.018342897935195418'], ['Dinajpur', '0.00333507
28810156837', '-0.0033350728810157006'], ['Faridpur', '7.024255623783972e-05', '-7.024255623784604e-05'], ['Feni', '0.00226112695232440 67', '-0.000
22611269523244404'], ['Hatiya', '0.00013424720086465067', '-0.00013424720086464907'], ['Ishurdi', '-0.0006756957299972381', '0.000675695729997237'],
['Jessore', '-0.00023127029684351282', '0.00023127029684349816'], ['Khepupara', '-4.0371005262369476e-05', '4.037100526237481e-05'], ['Khulna', '0.
00025686951848456 4', '-0.00025686951848457 11'], ['Madaripur', '7.526346196346176e-05', '-7.526346196346 05e-05'], ['Maijdee Court', '0.00027963906500
5994', '-0.00027963906500699 02'], ['Mongla', '7.18048324936524e-05', '-7.18048324936599e-05'], ['Mymensingh', '0.00012171347551038032', '-0.00012171
347551038198'], ['Patuakhali', '-2.173187567451481e-06', '2.1731875674485807e-06'], ['Rajshahi', '0.0011516191679436 17', '-0.0011516191679436206'],
['Rangamati', '-7.616857094297193e-05', '7.616857094297 16e-05'], ['Rangpur', '4.5142167273525725e-06', '-4.514216727350071e-06'], ['Sandwip', '-0.00
011003839740715541', '0.00011003839740715449'], ['Satkhira', '0.0001797183926550974', '-0.0001797183926550993 4'], ['Sitakunda', '-3.5703375149143216
e-05', '3.5703375149148 16e-05'], ['Sylhet', '0.0003749750510524 1267', '-0.0003749750510524 1413'], ['Tangail', '2.396214167047119e-06', '-2.396214167
0534934e-06'], ['Teknaf', '6.436241690373735e-05', '-6.436241690373738e-05']]]
```




## Status: Passed

As shown above, the prediction request made from the website and the manually executed API test both result in the same prediction output as well as the SHAP table and graph. This confirms that the front-end and back-end are integrated successfully and communicate effectively, ensuring that the flood prediction feature works as intended.

**Case 2:** Database storing and Retrieval

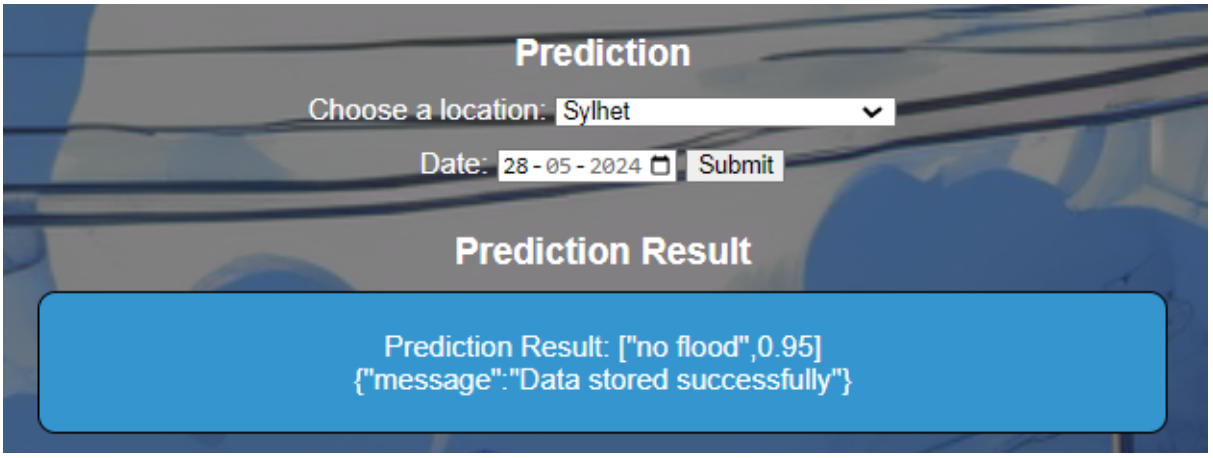
**Objective:** Verify that the prediction data is accurately retrieved and correctly stored in the database. As in the previous case, this test will involve first checking the results displayed on the website, and then confirming the same results by manually sending the inputs to the API. Additionally, we will check if the prediction in the database includes the correct timestamp to ensure that the prediction is made and recorded accurately in the database. This will ensure consistency between the frontend display and the backend data storage.

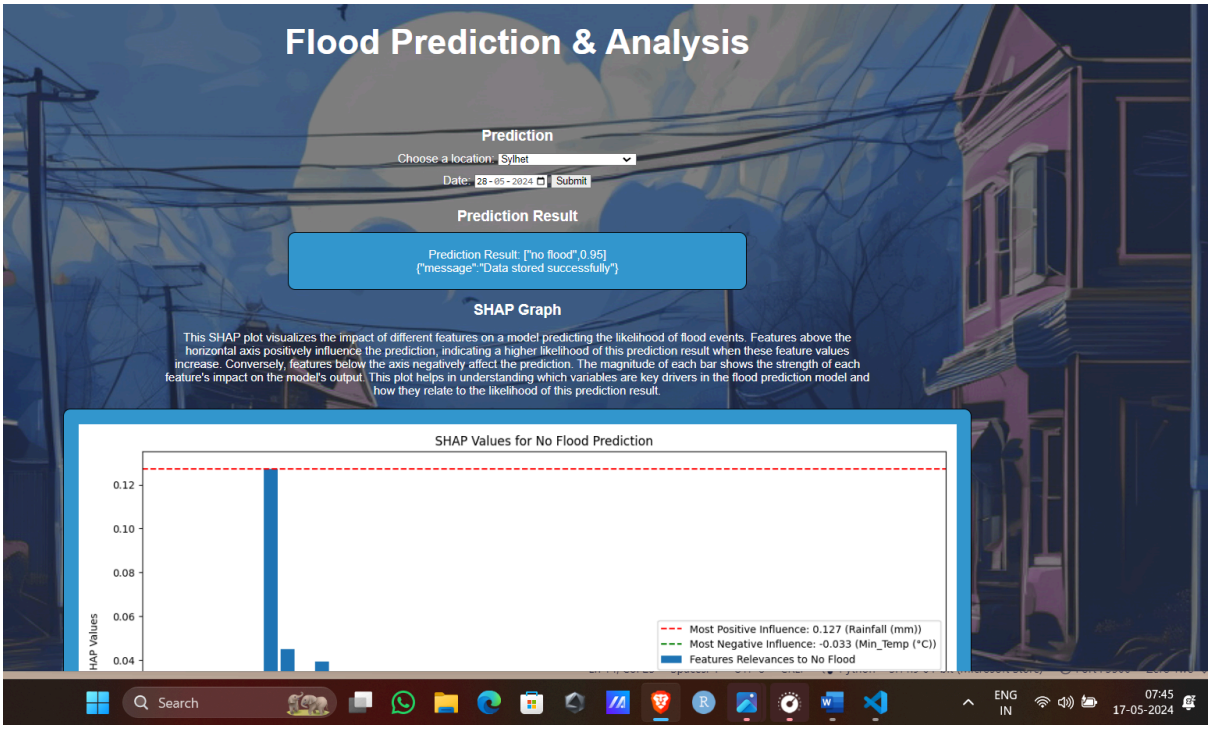
**Input:** {"city": "Sylhet", "date": "2024-05-28"} (The data specified is prediction date)

**Time:** "7:45 AM" | Date (generated): "2024-05-17"

**The result on the website:**

**Database storing (website)**



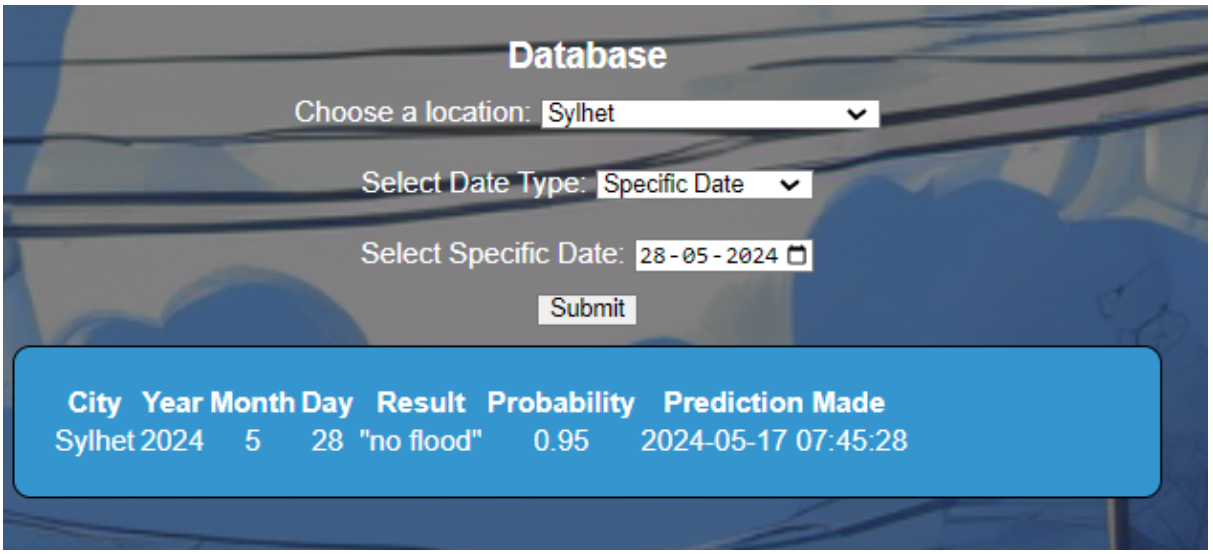
**Database Storing (database)**

```
18 •    SELECT * FROM PREDICTION_RESULT ORDER BY PREDICTION_MADE DESC;
19
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

| City | Year | Month | Day | Result | Probability | Prediction_Made |
|---|---|---|---|---|---|---|
| Chandpur | 2024 | 5 | 29 | "no flood" | 0.94 | 2024-05-17 08:11:09 |
| Sylhet | 2024 | 5 | 28 | "no flood" | 0.95 | 2024-05-17 07:45:28 |
| Bhola | 2024 | 5 | 23 | "no flood" | 0.93 | 2024-05-17 04:26:29 |

**Data Retrieval (website)**



**Test Code:**

```python
import unittest
import requests
import json

Codiumate: Add more tests
class TestGetData(unittest.TestCase):
    BASE_URL = 'http://127.0.0.1:8000'  # Replace with your actual server URL

    Codiumate: Add more tests
    def test_get_data(self):
        # Define the data to send
        data = {
            'city': 'Sylhet',
            'year': '2024',
            'month': '5',
            'day': '28'
        }

        # Send a POST request to the /get_data endpoint
        response = requests.post(f'{self.BASE_URL}/get_data', json=data)

        # Check that the status code is 200
        self.assertEqual(response.status_code, 200)

        # Check that the response is JSON
        self.assertEqual(response.headers['Content-Type'], 'application/json')

        # Parse the JSON response
        response_data = response.json()

        # Check that the response data is as expected
        # This will depend on what your function is supposed to return
        # Here's an example:
        print(response_data)

if __name__ == '__main__':
    unittest.main()
```

**Test result:**

```
PS C:\Users\Prateek Tripathi\Desktop\Flood prediction\Flood-Prediction_FYP> & "C:/Users/Prateek Tripathi/AppData/Local/Microsoft/WindowsApps/python3.11.exe" "c:/Users/Prateek Tripathi/Desktop/Flood prediction/Flood-Prediction_FYP/DatabaseAPITester.py"
[['Sylhet', 2024, 5, 28, '"no flood"', 0.95, '2024-05-17 07:45:28']]
.
----------------------------------------------------------------------
Ran 1 test in 0.145s

OK
```

**Data Retrieval (Database)**

```
18 •  SELECT *
19    FROM PREDICTION_RESULT
20    WHERE CITY = 'SYLHET'
21      AND YEAR = 2024
22      AND MONTH = 5
23      AND DAY = 28;
24
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content

| City | Year | Month | Day | Result | Probability | Prediction_Made |
|---|---|---|---|---|---|---|
| Sylhet | 2024 | 5 | 28 | "no flood" | 0.95 | 2024-05-17 07:45:28 |

**Status: Passed**

As shown above, the prediction made from the website is successfully stored in the database, and correctly retrieved. This can be further confirmed by looking at the test case results where we are able to retrieve the new prediction made. We can confirm this by checking the timestamp of the prediction made. This confirms that the front-end and back-end are integrated successfully and communicate effectively, ensuring that the database feature works as intended.

## 5. Usability Testing

The purpose of this usability test is to evaluate the ease of use, efficiency, and overall user satisfaction of our flood prediction application. The following outlines the usability test for our application, focusing on the prediction and historical data retrieval.

**The objectives we are trying to achieve are as follows:**

- Assess how easily users can make flood predictions using the application.
- Evaluate how effectively users can interpret prediction results and the accompanying SHAP (SHapley Additive exPlanations) graphs and tables.
- Determine the ease with which users can retrieve and interpret historical flood data.
- Collect user feedback on the overall user experience and identify any usability issues.

The main group of participants representing the target user base are students in Monash University, Friends and Family members of the development team and any volunteers that are optioned to be included in this testing.

**Our methodology:**

- Type of Test: Remote usability testing/Onsite testing
- Location: Remote (participants' environments)/Onsite(developers' environments)
- Tools: Laptops, Internet, Python IDE/VSCode

**Testing tasks:**

**Task 1: Making a Prediction**

Scenario: You want to make a flood prediction for today in Dhaka.

Steps:

1. Click on the "Make Prediction" button.
2. Select "Dhaka" as the location for the prediction.
3. Select the current date (today) as the Date for prediction.
4. Interpret the prediction result.
5. Review the SHAP graph and SHAP table for an explanation of the prediction.

Success Criteria:

The user completes all steps without significant errors.

The user accurately interprets the prediction result and the SHAP explanations.

**Task 2: Retrieving Historical Data**

Scenario: You want to view historical flood data for Dhaka.

Steps:

1. Go back to the home page.
2. Click on the "Get History" button.
3. Select "Dhaka" as the location for historical data.
4. Select "Month and Year" as the filter criteria.
5. Select May 2024.
6. Interpret the historical data results.

Success Criteria:

The user successfully navigates to and uses the historical data retrieval feature.

The user accurately interprets the historical data.

**Task 3: Learning About the Team**

Scenario: You want to learn more about the development team behind the application.

Steps:

1. Go back to the home page.
2. Click on the "About Our Team" button.

Success Criteria:

The user successfully navigates to the "About Our Team" section and reviews the information.

**Metrics**

- **User Satisfaction:** Participant feedback was gathered through post-task and post-test questionnaires.
- **Ease of Use:** Ratings on a scale of 1 to 5 for each task.

**Responses Results Table:**

The table below shows the counts of each response and mean scores.

(details breakdown and visualisations are located in the appendix)

| No | Question | 1(Very Poor) | 2(Poor) | 3(Normal) | 4(Good) | 5(Very Good) | Mean |
|---|---|---|---|---|---|---|---|
| 1 | How would you rate your overall experience with the application? | 1 | 0 | 2 | 4 | 3 | 3.8 |
| 2 | The layout of the website is visually appealing. | 1 | 2 | 1 | 3 | 3 | 3.5 |
| 3 | The website loads quickly. | 0 | 1 | 1 | 3 | 5 | 4.2 |
| 4 | How easy was it to navigate through the application? | 0 | 2 | 5 | 1 | 2 | 3.3 |
| 5 | How easy was it to make a prediction? | 0 | 0 | 2 | 1 | 7 | 4.5 |
| 6 | How well do you understand the prediction results? | 0 | 1 | 4 | 2 | 3 | 3.7 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 7 | Were the SHAP graph and SHAP table helpful in explaining the prediction? | 2 | 2 | 4 | 2 | 0 | 2.6 |
| 8 | How easy was it to retrieve historical data? | 0 | 0 | 1 | 5 | 4 | 4.3 |
| 9 | How well do you understand the historical data results? | 1 | 3 | 0 | 0 | 6 | 3.7 |
| 10 | Were the historical data results presented in a clear and understandable manner? | 0 | 2 | 3 | 3 | 2 | 3.5 |
| 11 | How easy was it to find information about the development team? | 1 | 2 | 3 | 1 | 3 | 3.3 |
| 12 | Was the information about the team sufficient and helpful? | 2 | 4 | 2 | 0 | 2 | 2.6 |

## 6. Recommendation for improvements

After performing the tests, it is crucial to understand where is the downsides of our application, based on the test results and user feedback. From that, there are a few improvements that can be made to enhance the user experience:

1. **SHAP table and graph:** As SHAP(SHapley Additive exPlanations) is not a common term and technique to explain the relevance of features, the value and the graph are not easy to understand by the public. From the usability test, the feature got a low mark on average, indicating that users do not understand how it works. Although we have a description of how to read the SHAP table and graph, it may seem too overwhelming and still does not work well. Hence, future work on this part should include a brief but more informative instruction to guide the user.
2. **Information about the team:** Based on the test results, the user does not seem to be satisfied with the information about the team provided with a low mark on average. Hence, besides focusing on the technical aspects, we should also focus on the information aspects in the future.

## 7. Limitation of Testing

Although the testing we performed has included every aspect of the application, there are still some limitations:

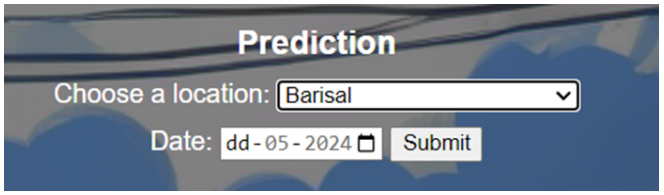
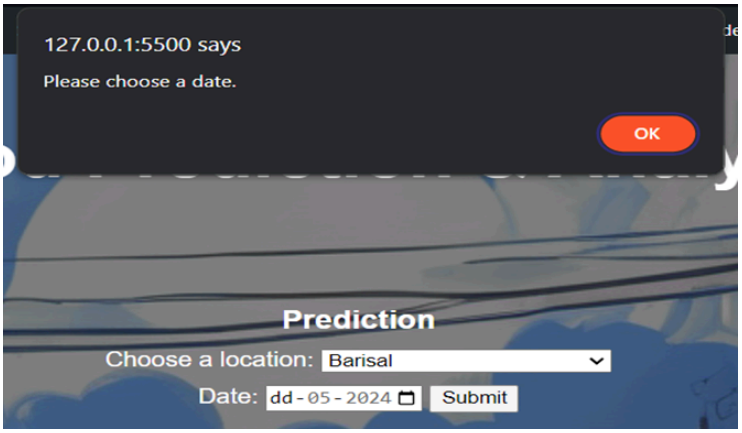
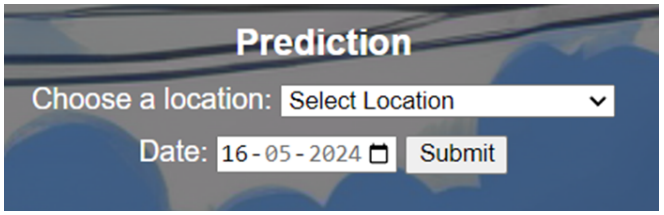
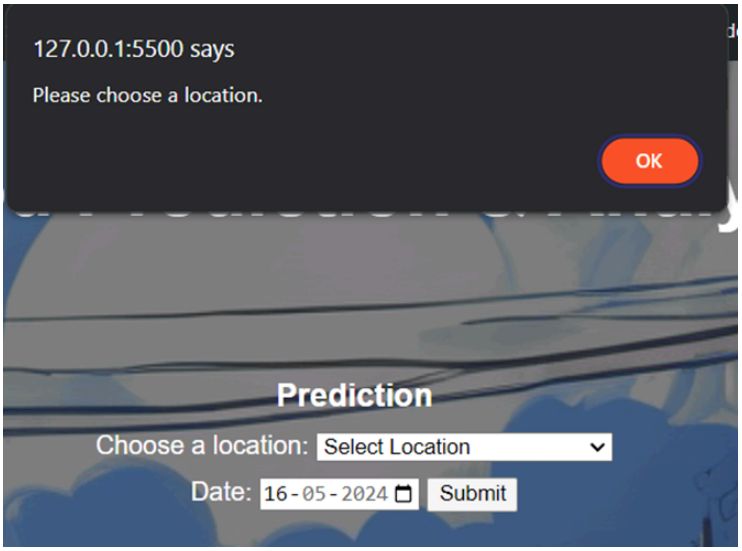
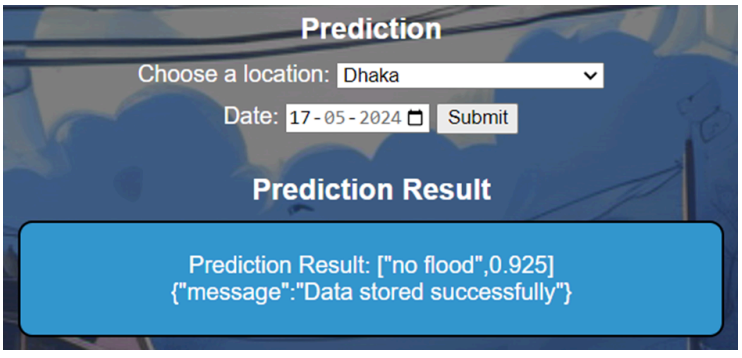
1. **Lack of Server Hosting:** The application has not been hosted on a live server. Hence, we are not able to simulate a real world where the application would be deployed and accessed by multiple users simultaneously.

2. **No Load Testing:** Due to the absence of server hosting, we are unable to perform load testing which is crucial for understanding how the application behaves under high-traffic conditions. Without this, we cannot actually gauge the performance, stability and responsiveness of the application when subjected to a large number of concurrent users.

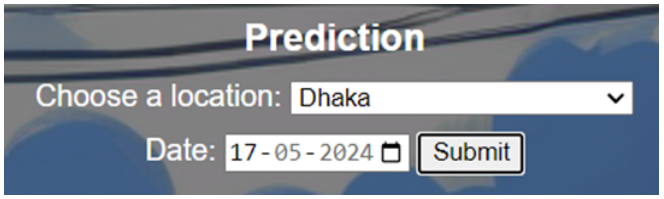
3. **Limited Security Testing:** Although our application does not allow users to manually input anything, which reduces the risk of injection attacks, there are still scenarios where injections could occur, such as through manipulating URLs or other forms of indirect input.

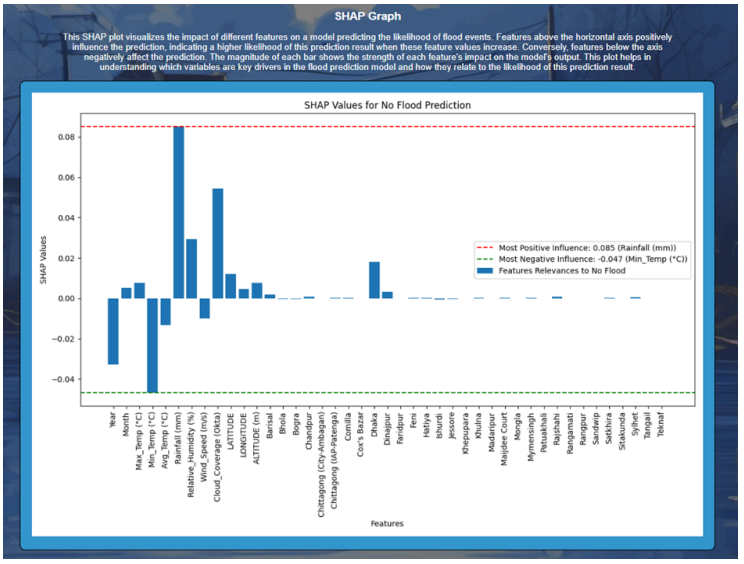
## 8. Appendix

**Screenshots of each Black Box test input and output.**

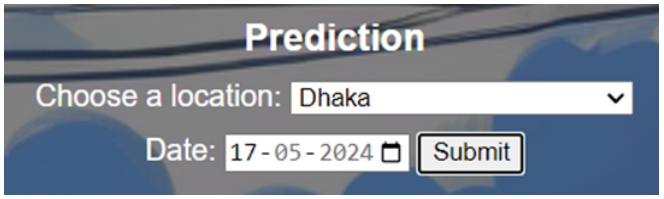
| Test S. No | Input Being Tested | Output Result |
|---|---|---|
| **3.1** |  |  |
| **3.2** |  |  |
| **3.3** |  |  |

**3.4**


Prediction
Choose a location: Dhaka
Date: 17-05-2024
Submit

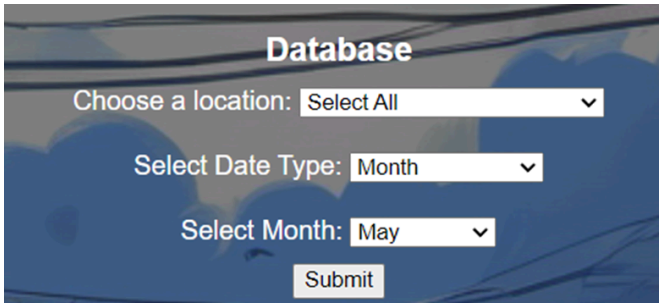

SHAP Graph
This SHAP plot visualizes the impact of different features on a model predicting the likelihood of flood events. Features above the horizontal axis positively influence the prediction, indicating a higher likelihood of this prediction result when these feature values increase. Conversely, features below the axis negatively affect the prediction. The magnitude of each bar shows the strength of each feature's impact on the model's output. This plot helps in understanding which variables are key drivers in the flood prediction model and how they relate to the likelihood of this prediction result.
SHAP Values for No Flood Prediction
SHAP Values
Features
Most Positive Influence: 0.085 (Rainfall (mm))
Most Negative Influence: -0.047 (Min_Temp (°C))
Features Relevances to No Flood

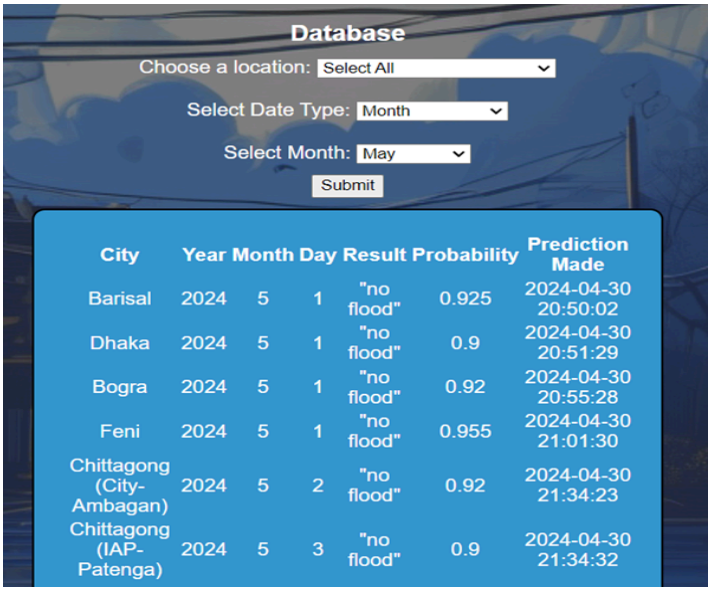
**3.5**

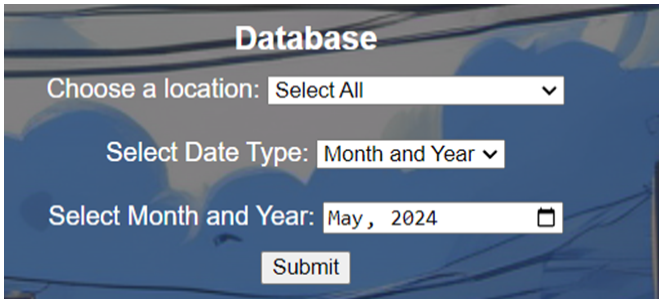
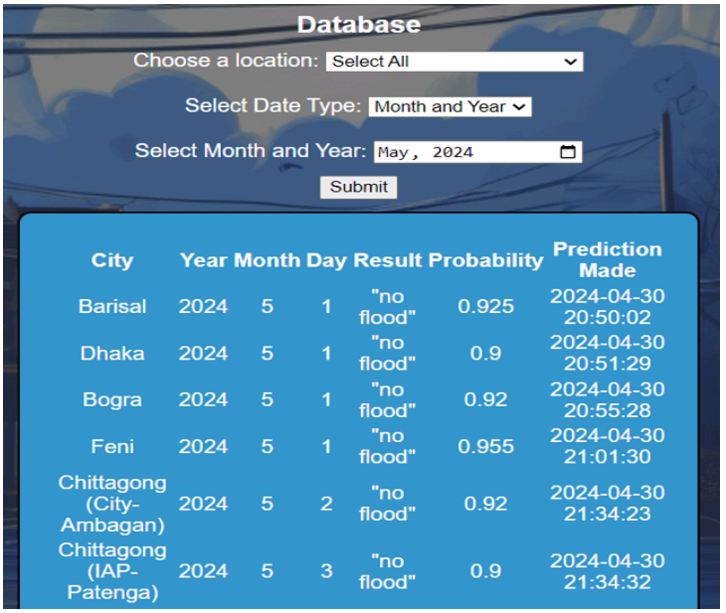
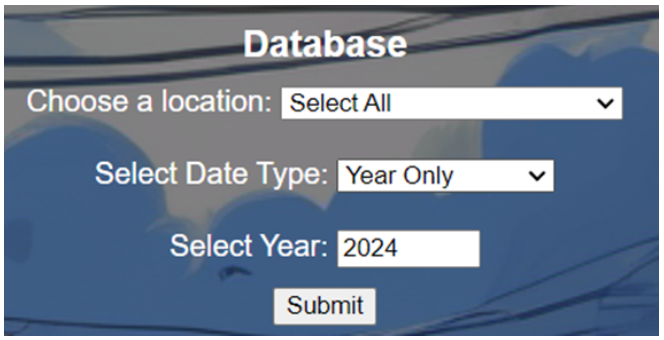
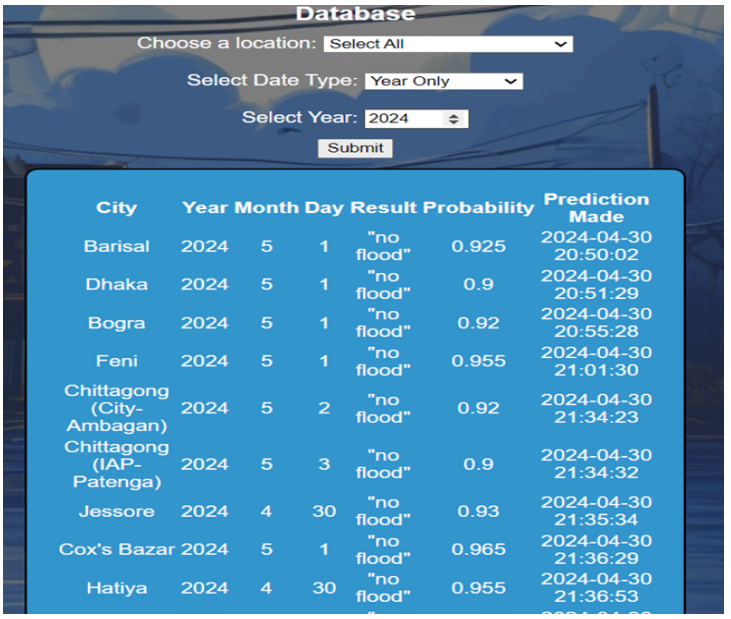

Prediction
Choose a location: Dhaka
Date: 17-05-2024
Submit


SHAP Table
The table lists SHAP (SHapley Additive exPlanations) values for various features in a predictive model, indicating how each feature contributes to the model's predictions. A positive SHAP value means a feature increases the prediction, while a negative value indicates a decrease. The magnitude of these values shows the strength of their impact. The features range from continuous variables like temperature to categorical variables like geographical locations, each affecting the model's output uniquely. By analyzing these values, we an understand which features are most important and how they influence the predictions.
Attributes
SHAP Value
Year -0.03283358697458214
Month 0.005066809688137185
Max_Temp (°C) 0.007520749221483938
Min_Temp (°C) -0.046779588152218 47
Avg_Temp (°C) -0.013219668312275108
Rainfall (mm) 0.08587495625992 78
Relative_Humidity (%) 0.029285402130988313
Wind_Speed (m/s) -0.009965095492652 37
Cloud_Coverage (Okta) 0.054356345761837 76
LATITUDE 0.012055171381934631
LONGITUDE 0.004505254080343538
ALTITUDE (m) 0.007587924788071793
Barisal 0.001816578098133061 6
Bhola -0.000219771247245895 26
Bogra -0.000289261325984169 53
Chandpur 0.000817223201778576
Chittagong (City-Ambagan) -6.440968356662434e-06
Chittagong (IAP-Patenga) 0.000128773878944997 92
Comilla 0.000235634678078502 04
Cox's Bazar -0.000154742713553687 304
Dhaka 0.018196063201804294
Dinajpur 0.003181446960911104 3
Faridpur 7.0983424779016 22e-05
Feni 0.000226027350152550 05
Hatiya 0.000130495362818736 75
Ishurdi -0.000675681715224 0828
Jessore -0.000229335619918451 37
Khepupara -4.0108067499471 7e-05
Khulna 0.000257959589700 2127
Madaripur 7.7789844187930 97e-05
Maijdee Court 0.000281647812162594 06
Mongla 7.1910180964507 74e-05
Mymensingh 0.000120241063942 4721
Patuakhali -3.014175178173663e-06
Rajshahi 0.000851067324763587 1
Rangamati -7.6073225419158 47e-05
Rangpur 3.424749124198604 6e-06
Sandwip -9.7991703753185 9e-05
Satkhira 0.000180353761083856 38
Sitakunda -3.533513492961273e-05
Sylhet 0.000376526725490657
Tangail 4.156226080924186 5e-06
Teknaf 6.449177359581018e-05

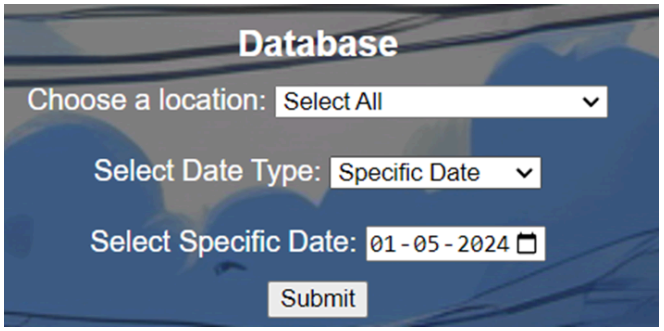
**3.6**

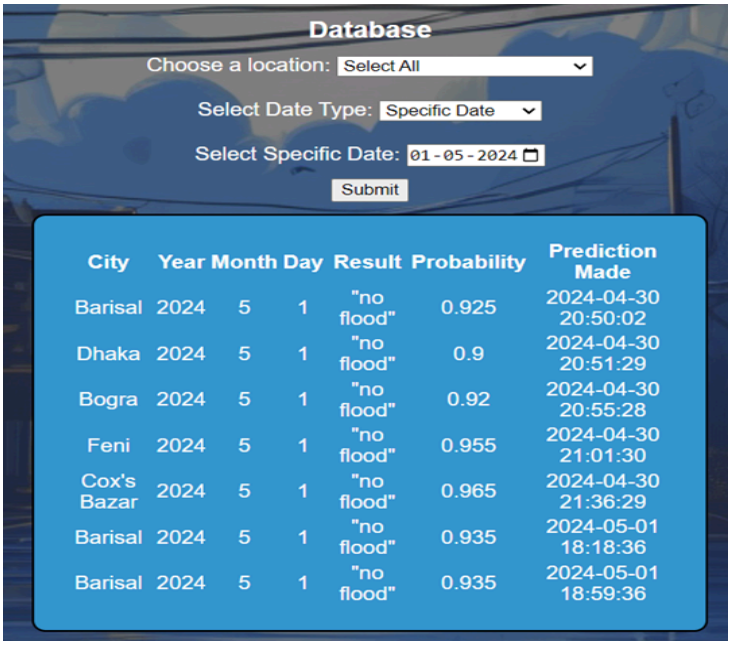

Database
Choose a location: Select All
Select Date Type: Month
Select Month: May
Submit


Database
Choose a location: Select All
Select Date Type: Month
Select Month: May
Submit
City Year Month Day Result Probability Prediction Made
Barisal 2024 5 1 "no flood" 0.925 2024-04-30 20:50:02
Dhaka 2024 5 1 "no flood" 0.9 2024-04-30 20:51:29
Bogra 2024 5 1 "no flood" 0.92 2024-04-30 20:55:28
Feni 2024 5 1 "no flood" 0.955 2024-04-30 21:01:30
Chittagong (City-Ambagan) 2024 5 2 "no flood" 0.92 2024-04-30 21:34:23
Chittagong (IAP-Patenga) 2024 5 3 "no flood" 0.9 2024-04-30 21:34:32

| | | |
|---|---|---|
| **3.7** |  Database<br>Choose a location: Select All<br>Select Date Type: Month and Year<br>Select Month and Year: May, 2024<br>Submit |  Database<br>Choose a location: Select All<br>Select Date Type: Month and Year<br>Select Month and Year: May, 2024<br>Submit<br>City Year Month Day Result Probability Prediction Made<br>Barisal 2024 5 1 "no flood" 0.925 2024-04-30 20:50:02<br>Dhaka 2024 5 1 "no flood" 0.9 2024-04-30 20:51:29<br>Bogra 2024 5 1 "no flood" 0.92 2024-04-30 20:55:28<br>Feni 2024 5 1 "no flood" 0.955 2024-04-30 21:01:30<br>Chittagong (City-Ambagan) 2024 5 2 "no flood" 0.92 2024-04-30 21:34:23<br>Chittagong (IAP-Patenga) 2024 5 3 "no flood" 0.9 2024-04-30 21:34:32 |
| **3.8** |  Database<br>Choose a location: Select All<br>Select Date Type: Year Only<br>Select Year: 2024<br>Submit |  Database<br>Choose a location: Select All<br>Select Date Type: Year Only<br>Select Year: 2024<br>Submit<br>City Year Month Day Result Probability Prediction Made<br>Barisal 2024 5 1 "no flood" 0.925 2024-04-30 20:50:02<br>Dhaka 2024 5 1 "no flood" 0.9 2024-04-30 20:51:29<br>Bogra 2024 5 1 "no flood" 0.92 2024-04-30 20:55:28<br>Feni 2024 5 1 "no flood" 0.955 2024-04-30 21:01:30<br>Chittagong (City-Ambagan) 2024 5 2 "no flood" 0.92 2024-04-30 21:34:23<br>Chittagong (IAP-Patenga) 2024 5 3 "no flood" 0.9 2024-04-30 21:34:32<br>Jessore 2024 4 30 "no flood" 0.93 2024-04-30 21:35:34<br>Cox's Bazar 2024 5 1 "no flood" 0.965 2024-04-30 21:36:29<br>Hatiya 2024 4 30 "no flood" 0.955 2024-04-30 21:36:53 |

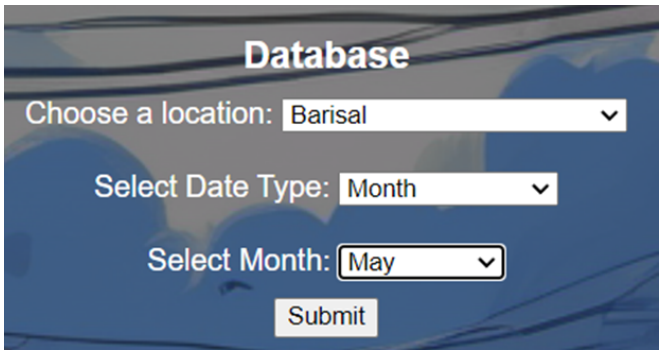
**3.9**

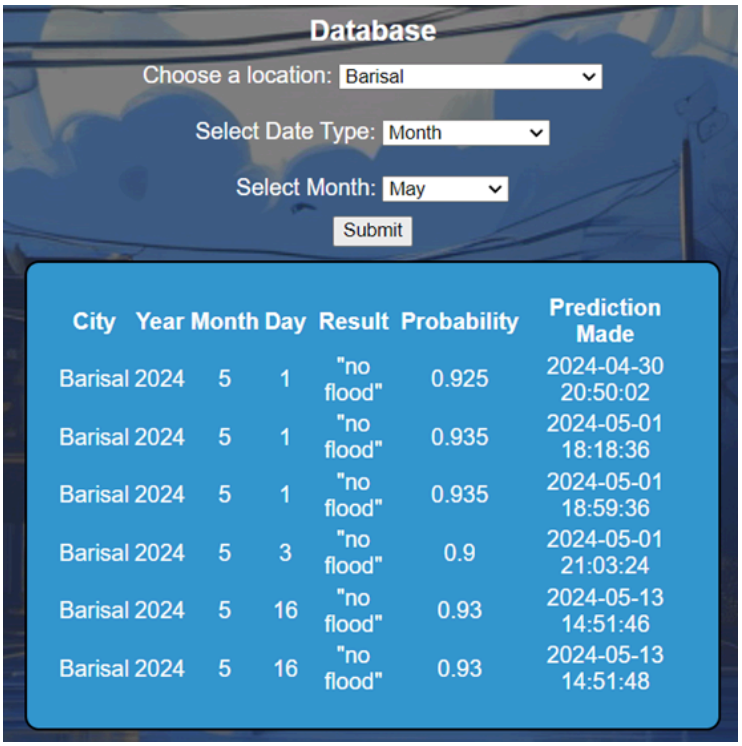



| City | Year | Month | Day | Result | Probability | Prediction Made |
|---|---|---|---|---|---|---|
| Barisal | 2024 | 5 | 1 | "no flood" | 0.925 | 2024-04-30 20:50:02 |
| Dhaka | 2024 | 5 | 1 | "no flood" | 0.9 | 2024-04-30 20:51:29 |
| Bogra | 2024 | 5 | 1 | "no flood" | 0.92 | 2024-04-30 20:55:28 |
| Feni | 2024 | 5 | 1 | "no flood" | 0.955 | 2024-04-30 21:01:30 |
| Cox's Bazar | 2024 | 5 | 1 | "no flood" | 0.965 | 2024-04-30 21:36:29 |
| Barisal | 2024 | 5 | 1 | "no flood" | 0.935 | 2024-05-01 18:18:36 |
| Barisal | 2024 | 5 | 1 | "no flood" | 0.935 | 2024-05-01 18:59:36 |

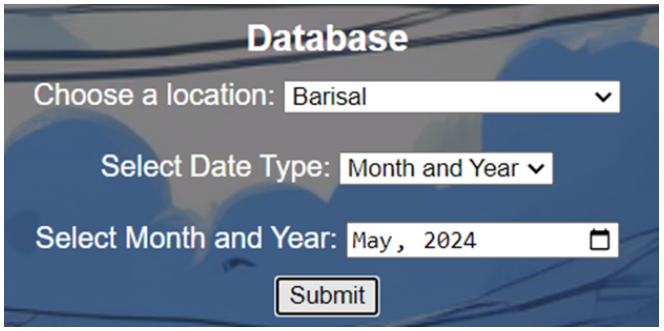
**3.10**

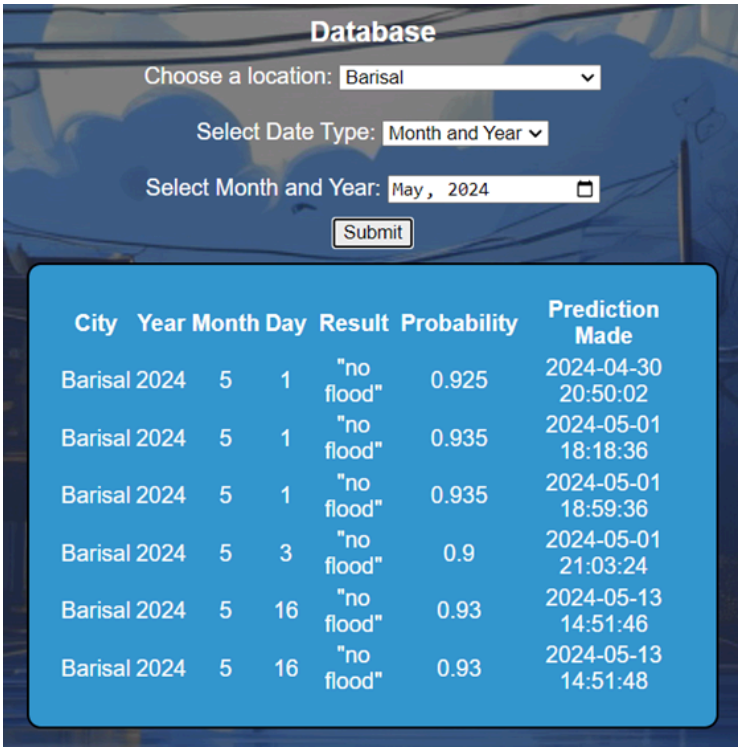



| City | Year | Month | Day | Result | Probability | Prediction Made |
|---|---|---|---|---|---|---|
| Barisal | 2024 | 5 | 1 | "no flood" | 0.925 | 2024-04-30 20:50:02 |
| Barisal | 2024 | 5 | 1 | "no flood" | 0.935 | 2024-05-01 18:18:36 |
| Barisal | 2024 | 5 | 1 | "no flood" | 0.935 | 2024-05-01 18:59:36 |
| Barisal | 2024 | 5 | 3 | "no flood" | 0.9 | 2024-05-01 21:03:24 |
| Barisal | 2024 | 5 | 16 | "no flood" | 0.93 | 2024-05-13 14:51:46 |
| Barisal | 2024 | 5 | 16 | "no flood" | 0.93 | 2024-05-13 14:51:48 |

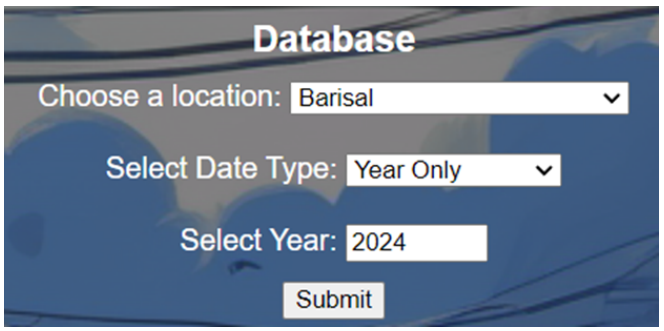
**3.11**



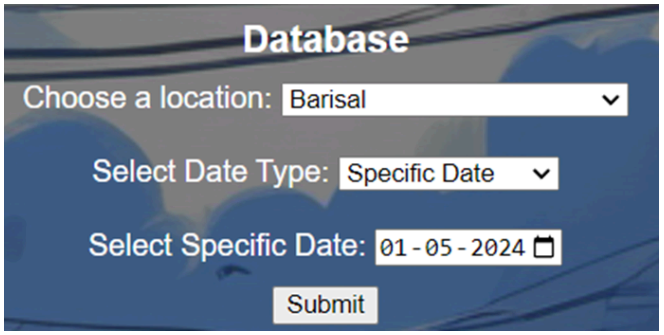
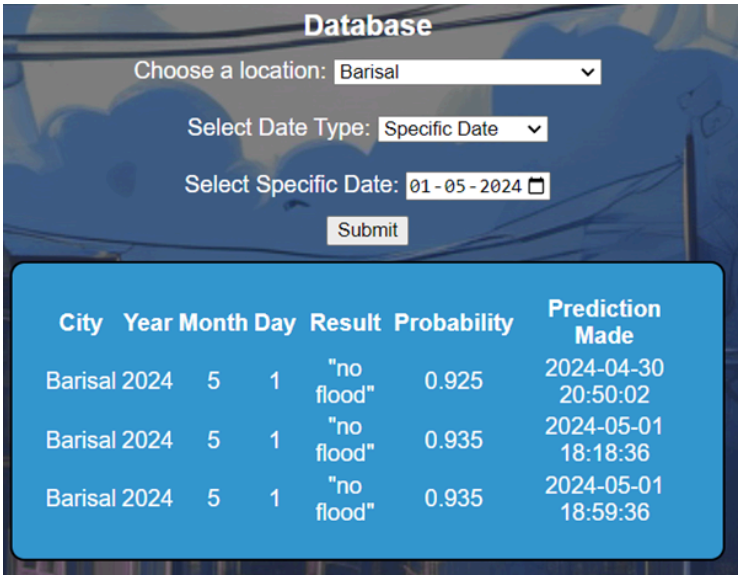
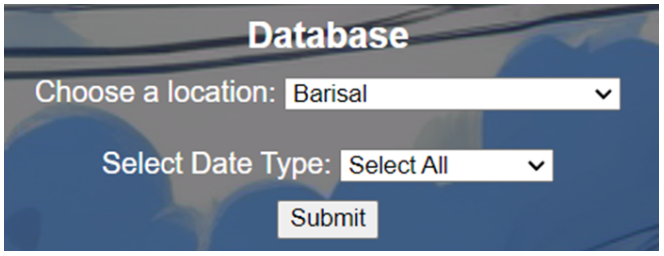

| City | Year | Month | Day | Result | Probability | Prediction Made |
|---|---|---|---|---|---|---|
| Barisal | 2024 | 5 | 1 | "no flood" | 0.925 | 2024-04-30 20:50:02 |
| Barisal | 2024 | 5 | 1 | "no flood" | 0.935 | 2024-05-01 18:18:36 |
| Barisal | 2024 | 5 | 1 | "no flood" | 0.935 | 2024-05-01 18:59:36 |
| Barisal | 2024 | 5 | 3 | "no flood" | 0.9 | 2024-05-01 21:03:24 |
| Barisal | 2024 | 5 | 16 | "no flood" | 0.93 | 2024-05-13 14:51:46 |
| Barisal | 2024 | 5 | 16 | "no flood" | 0.93 | 2024-05-13 14:51:48 |

**3.12**





| City | Year | Month | Day | Result | Probability | Prediction Made |
|---|---|---|---|---|---|---|
| Barisal | 2024 | 5 | 1 | "no flood" | 0.925 | 2024-04-30 20:50:02 |
| Barisal | 2024 | 5 | 1 | "no flood" | 0.935 | 2024-05-01 18:18:36 |
| Barisal | 2024 | 5 | 1 | "no flood" | 0.935 | 2024-05-01 18:59:36 |
| Barisal | 2024 | 5 | 3 | "no flood" | 0.9 | 2024-05-01 21:03:24 |
| Barisal | 2024 | 5 | 16 | "no flood" | 0.93 | 2024-05-13 14:51:46 |
| Barisal | 2024 | 5 | 16 | "no flood" | 0.93 | 2024-05-13 14:51:48 |

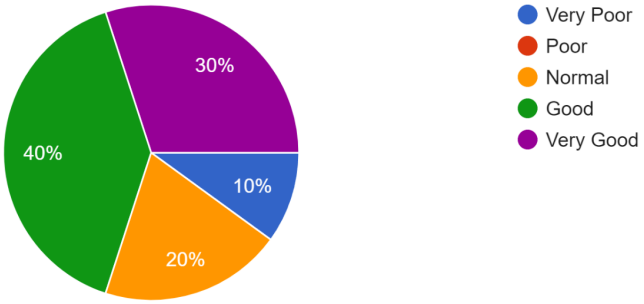
| | | |
|---|---|---|
| 3.13 |  Database<br>Choose a location: Barisal<br>Select Date Type: Specific Date<br>Select Specific Date: 01-05-2024<br>Submit |  Database<br>Choose a location: Barisal<br>Select Date Type: Specific Date<br>Select Specific Date: 01-05-2024<br>Submit<br>City Year Month Day Result Probability Prediction Made<br>Barisal 2024 5 1 "no flood" 0.925 2024-04-30 20:50:02<br>Barisal 2024 5 1 "no flood" 0.935 2024-05-01 18:18:36<br>Barisal 2024 5 1 "no flood" 0.935 2024-05-01 18:59:36 |
| 3.14 |  Database<br>Choose a location: Barisal<br>Select Date Type: Select All<br>Submit |  Database<br>Choose a location: Barisal<br>Select Date Type: Select All<br>Submit<br>City Year Month Day Result Probability Prediction Made<br>Barisal 2024 5 1 "no flood" 0.925 2024-04-30 20:50:02<br>Barisal 2024 5 1 "no flood" 0.935 2024-05-01 18:18:36<br>Barisal 2024 5 1 "no flood" 0.935 2024-05-01 18:59:36<br>Barisal 2024 5 3 "no flood" 0.9 2024-05-01 21:03:24<br>Barisal 2024 5 16 "no flood" 0.93 2024-05-13 14:51:46<br>Barisal 2024 5 16 "no flood" 0.93 2024-05-13 14:51:48 |

**Usability Testing Responses breakdown and visualisations**

Form: https://forms.gle/wy26cXUV53Txx7XJ8

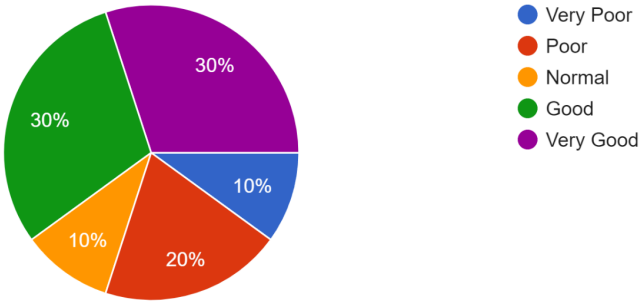
How would you rate your overall experience with the application?

10 responses



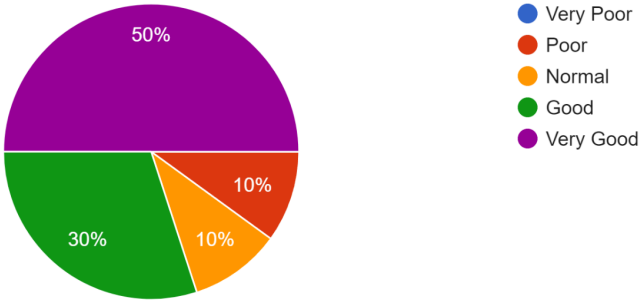

The layout of the website is visually appealing.

10 responses



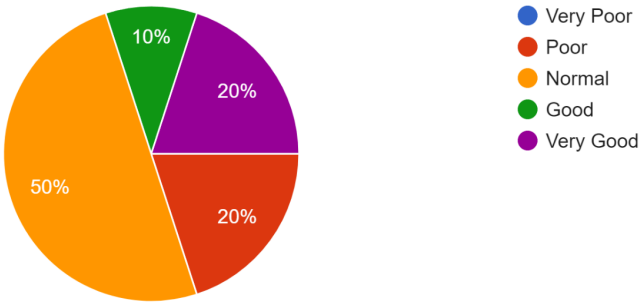

The website loads quickly.

10 responses



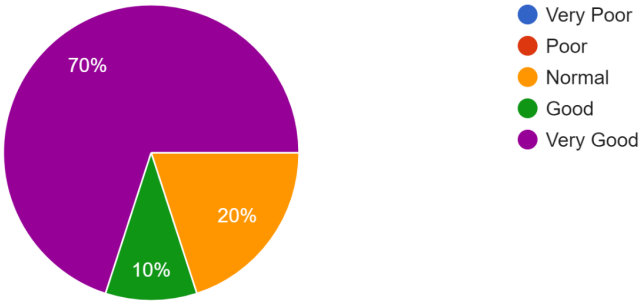
How easy was it to navigate through the application?

10 responses



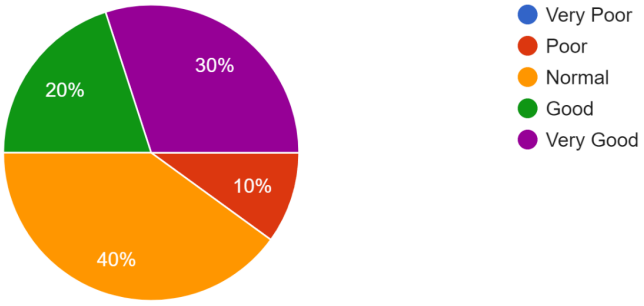

How easy was it to make a prediction?

10 responses



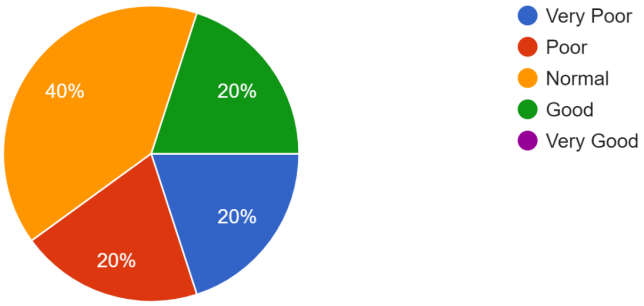

How well you understand the prediction results?

10 responses



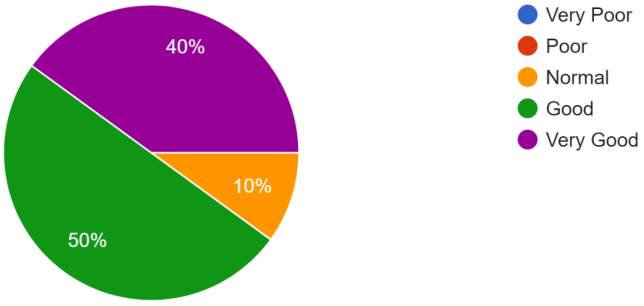
Were the SHAP graph and SHAP table helpful in explaining the prediction?

10 responses



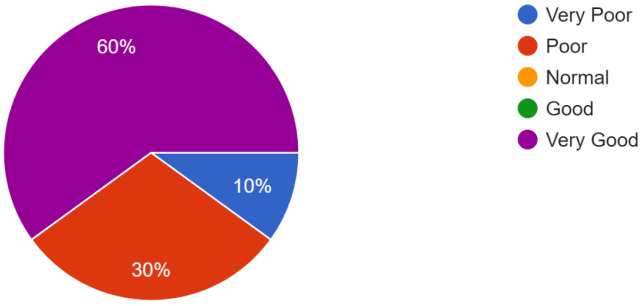

How easy was it to retrieve historical data?

10 responses



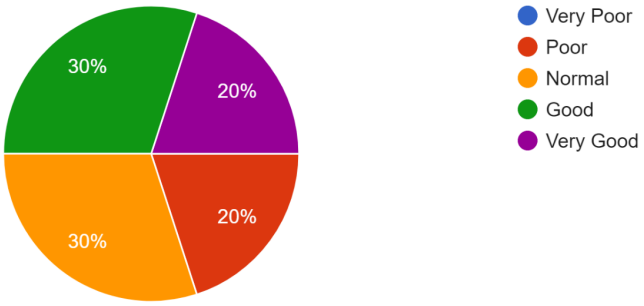

How well you understand the historical data results?

10 responses



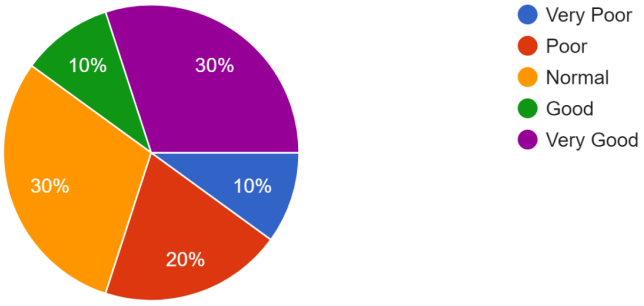
Were the historical data results presented in a clear and understandable manner?

10 responses




How easy was it to find information about the development team?

10 responses




Was the information about the team sufficient and helpful?

10 responses