**PROJECT TRAINING REPORT**

**DEFENCE LABORATORY**

**DEFENCE RESEARCH AND DEVELOPMENT ORGANISATION,**

**JODHPUR**



**Project Training Report on**

***“App development for visualizing Aerial surveillance data on map using MATLAB”***

*(From June 30, 2024 to July 15, 2024)*

***Submitted By: Under the Guidance of:***

Abhijeet Goswami Mr. Vivek Bhatnagar

Electronics and Computer Eng. Scientist ‘E

II Year (23UECC8003) DLJ, DRDO

M.B.M. University, Jodhpur Jodhpur

**Preface**

Practical training is important constituent of any curriculum and the B.Tech. course is no exception to this general rule. A practical training helps a student in getting acquainted with the manner in which his knowledge in being practically used outside his institute and this is normally different from what he has learnt from books. Hence, when one switches from the purpose of learning to that of implementing his knowledge he finds an abrupt change. This is exactly why practical training session during B.Tech. curriculum becomes all the more important. This provision offers a very good opportunity to the students to supplement their knowledge and skills by working in industrial or corporate environment. This opportunity should be utilized for developing and enhancing our skills as practicing engineers.

This report describes in detail about my project on Familiarization with MATLAB Introduction to Programming with MATLAB, App development for visualizing aerial survillence data on map using MATLAB which is completed at Defence Laboratory (DRDO), Jodhpur.

**CERTIFICATE**

This is to certify that Mr.Abhijeet Goswami student of BTech , Electronics and Communication Engineering (ECE) from MBM University has undergone practical training on ***“App development for visualizing Aerial surveillance data on map using MATLAB”*** at Defence Laboratory (DRDO), Jodhpur from *May 31, 2024 to July 15, 2024.*

He successfully completed his project app development for visualizing Aerial surveillance data on map using MATLAB .

**Signature**

**(Sh. Vivek Bhatnagar)Scientist ‘E’**

**DRDO , Jodhpur**

**Acknowledgement**

I have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals and organizations. I would like to extend my sincere thanks to all of them.

I am very grateful to Sh. Shri RV Hara Prasad Outstanding Scientist &Director, Defence Laboratory, Jodhpur for providing us with this opportunity to carry out our training in this esteemed and pioneering institution.

I am highly indebted toMr. Vivek Bhatnagar, Scientist ‘E’ Defence Laboratory, Jodhpur for his guidance and constant supervision as well as for providing necessary information regarding the project and also their support in completing the project.

I would also like to express my gratitude towards my parents and members of Defence Laboratory, Jodhpur for their kind co-operation and encouragement which helped me in completion of this project.

**Project Introduction**

The Project was undertaken as a part of Practical Training held from *May 31, 2024 to July 15, 2024*. The following Activities have been carried out:-

1. Familiarization with MATLAB.
2. Familiarization with Programming on MATLAB
3. APP Development for plotting UAV Flight path data on Map

4. Visualizing Aerial Surveillance data on Map

**Index :**

**1. MATLAB**

**1.1 Introduction 7**

**1.2 Objective 8**

**1.3 Applications 8**

**1.4 Uses 9**

**2. MATLAB APP DESIGNER 10-49**

**Task -1 a }: Making of a Simple Dice 17-20**

**b}: Graphical User Interface(GUI) App 20-23**

**Task -2 : Text File Browser App 24-28**

**Task -3: Text File Browser and Plotting App 29-34**

**Task -4 : Concentric Circles Plotting App 35-38**

**Task -5: *Visualize the Aerial Surveillence Data on Map 39*-49**

1.MATLAB

**1.1  *INTRODUCTION***

MATLAB (short for "MATRIX LABORATORY") is a high-level programming language and interactive environment developed by MathWorks. It is widely used for numerical computation, visualization, and programming.

MATLAB is particularly popular among engineers and scientists for its ease of use and powerful tools for data analysis, algorithm development, and modeling and simulation.

Key Features of MATLAB:

1. High-Level Language: MATLAB provides a high-level programming language with built-in functions and toolboxes that simplify complex mathematical and engineering computations.
2. Interactive Environment: MATLAB's interactive environment allows users to experiment, iterate, and visualize data and results quickly.
3. Toolboxes:MATLAB offers a wide range of toolboxes for specific applications, including signal processing, control systems, image processing, and more.
4. Visualization: MATLAB has robust visualization capabilities, enabling users to create 2D and 3D plots, graphs, and animations to represent data and results visually.
5. Integration: MATLAB can interface with other programming languages like C, C++, Java, and Python, and it can also integrate with hardware devices, making it versatile for various applications.
6. Simulink:MATLAB includes Simulink, a block diagram environment for modeling, simulating, and analyzing dynamic systems.

**1.2 *OBJECTIVE***

The primary objective of MATLAB is to provide a comprehensive environment for technical computing that combines a high-level programming language with robust computational, visualization, and development tools.

The goals can be broken down into several key objectives:

1. Simplifying Numerical Computation

2. Facilitating Algorithm Development

3. Enhancing Data Analysis and Visualization

4. Supporting Modeling and Simulation

5. Promoting Integration and Interoperability

6. Enabling Collaboration and Education

1.3 ***APPLICATIONS OF MATLAB***

1 Engineering: MATLAB is widely used in electrical, mechanical, civil, and aerospace engineering for tasks such as control system design, signal processing, and structural analysis.

2 Finance:MATLAB is used for financial modeling, risk management, portfolio optimization, and quantitative analysis.

3 Education: MATLAB is extensively used in academic institutions for teaching mathematics, engineering, and science courses due to its interactive and visual approach to problem-solving.

4 Biotechnology and Healthcare:MATLAB is used for bioinformatics, medical imaging, and the development of medical devices and diagnostic tools.

5 Automotive Industry:MATLAB and Simulink are used for designing and testing automotive systems, such as engine control units (ECUs), autonomous driving algorithms, and vehicle dynamics.

**1.4 *USES***

1) Numerical Computation: MATLAB is used for performing numerical analysis, including matrix operations, linear algebra, numerical integration, and differential equations.

2) Data Analysis: MATLAB provides tools for data acquisition, preprocessing, analysis, and visualization, making it ideal for statistical analysis and data mining.

3) Algorithm Development: MATLAB's high-level language and extensive function library make it suitable for developing and testing algorithms.

4) Modeling and Simulation: Engineers use MATLAB and Simulink for modeling and simulating physical systems, such as control systems, mechanical systems, and electrical circuits.

5) Scientific Research: Researchers use MATLAB for prototyping, developing experiments, and analyzing experimental data.

***MATLAB App Designer***

MATLAB App Designer is a drag-and-drop environment where you can design a user interface and write the code that drives its behavior.

Here's a step-by-step tutorial to get you started:

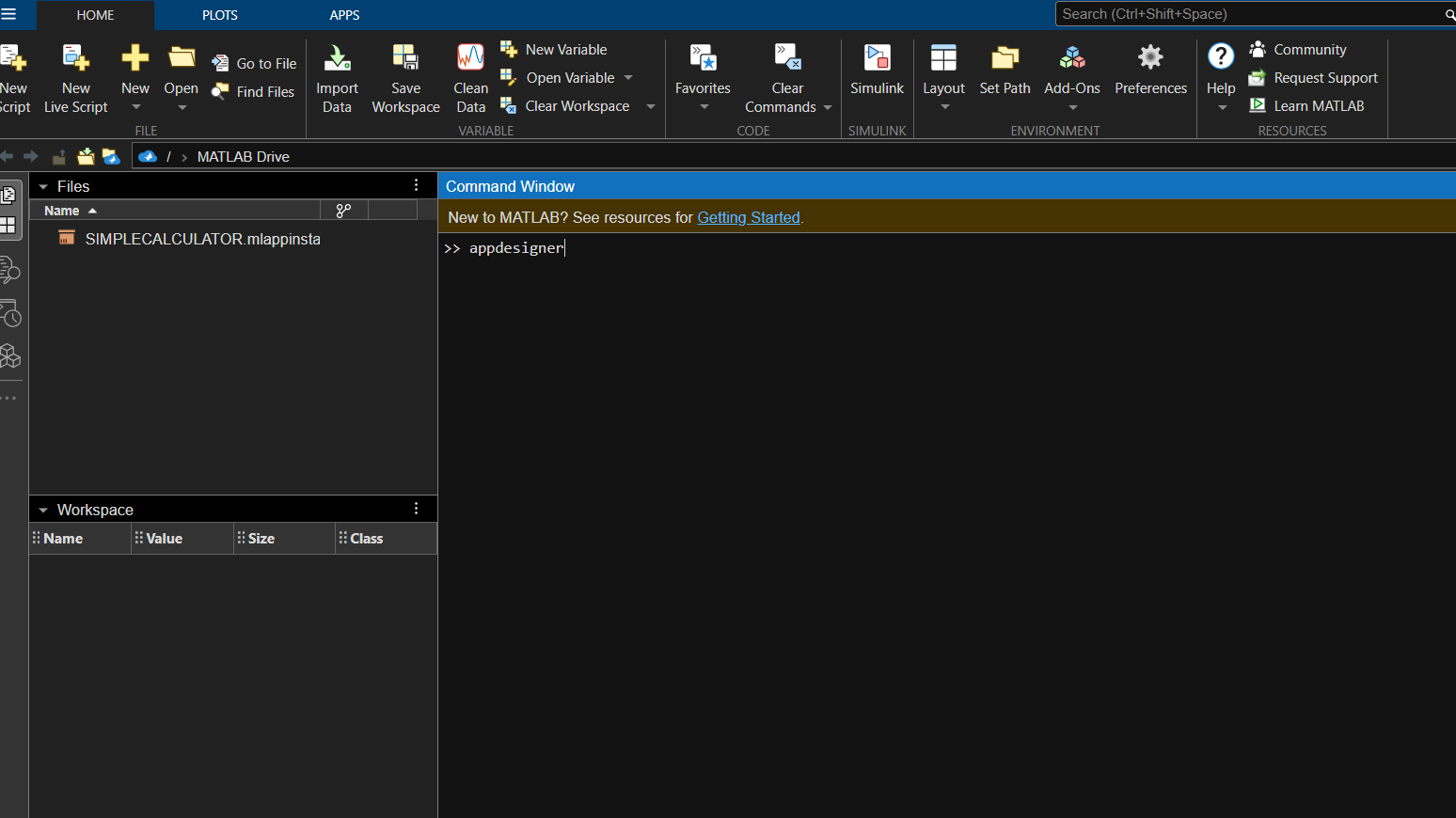
**Step 1** :

Open MATLAB App Designer

Open MATLAB.

In the MATLAB Command Window, type appdesigner and press Enter.

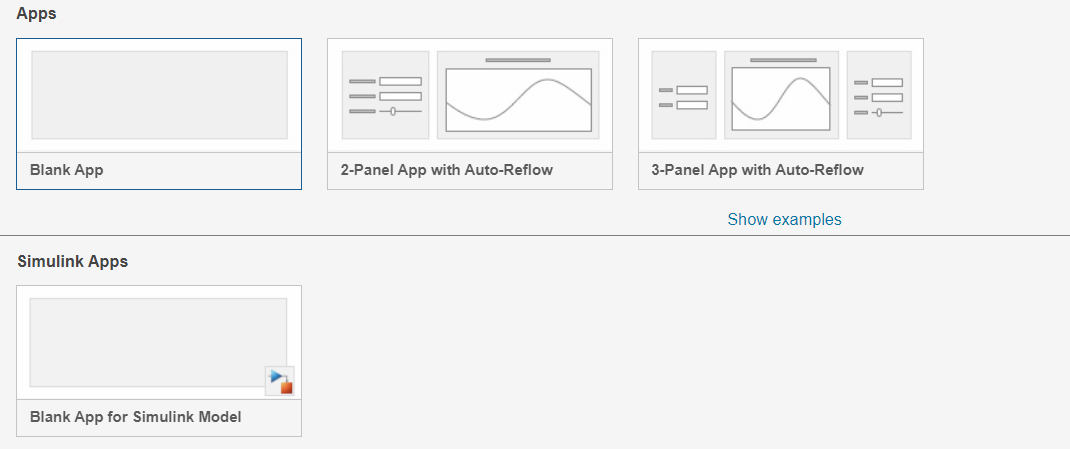
This will open the App Designer window.



**Step 2** :

Create a New App

Click on "Blank App" to create a new app from scratch.



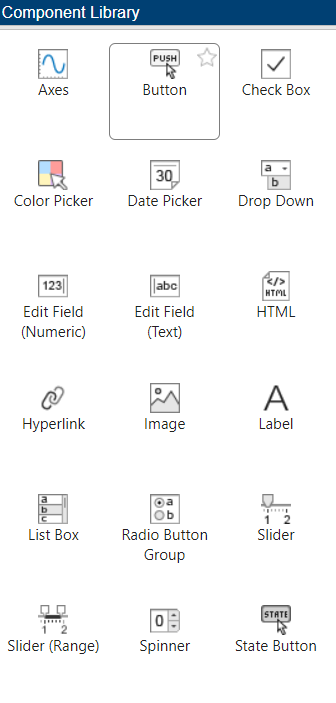
**Step 3** :

Design the User Interface

The central area is the canvas where you will design your UI.

On the right, you have the Component Library with various UI components like buttons, sliders, labels, etc.

On the left, you have the Component Browser and the Property Inspector.



**Step 4**:

Add Callbacks

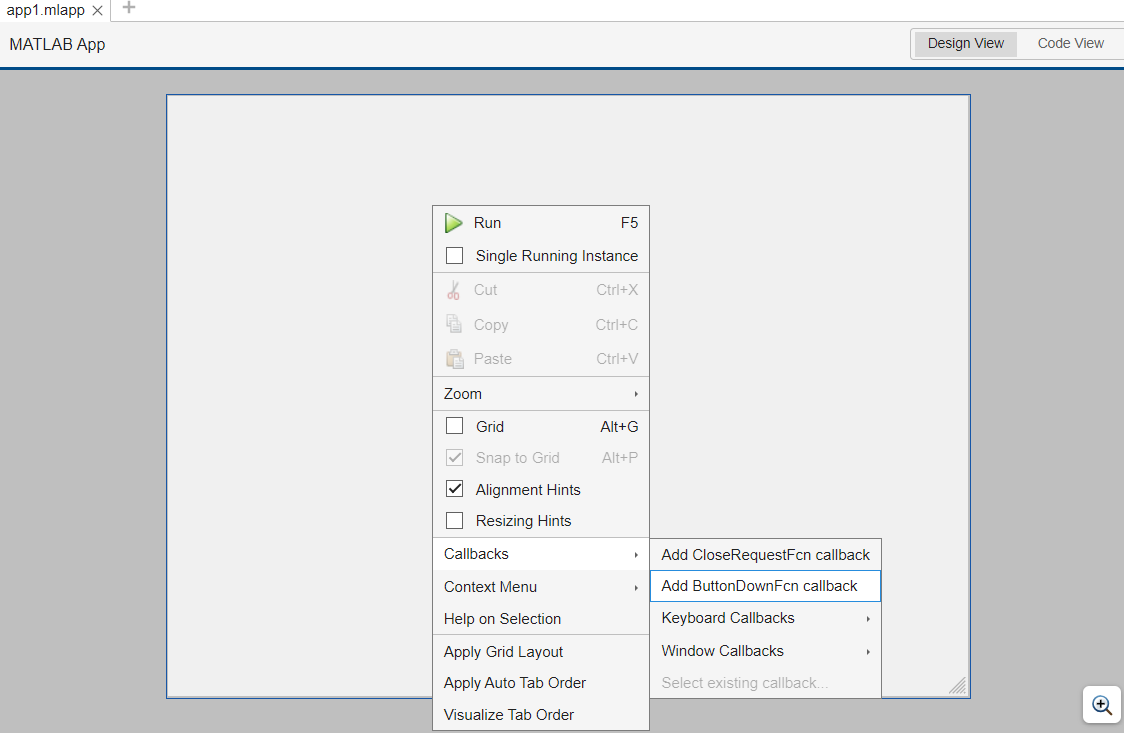
Callbacks are functions that are executed in response to user actions like clicking a button.

Select the Button on the canvas.

In the Property Inspector, scroll down to the Callbacks section.

Click the "+" next to ButtonPushedFcn to create a new callback for the button's pushed event.

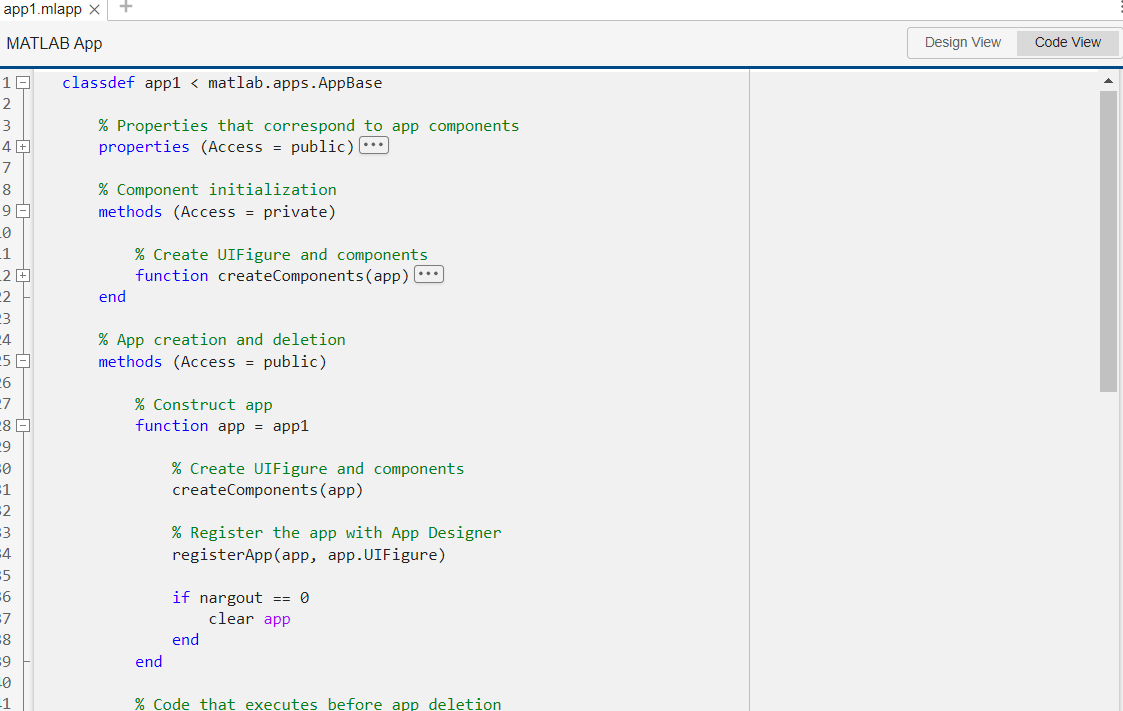
This will switch you to the Code View, and MATLAB will generate a function template for the button's pushed event.



Step 5:

Write Code

In the generated function template, you can write the code that should execute when the button is clicked.



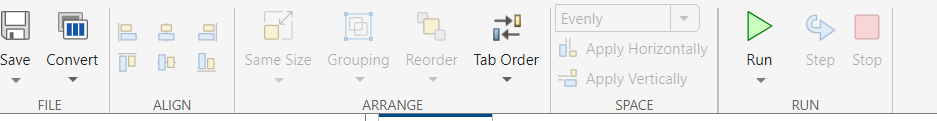
**Step 6**:

Run the App

Click the Runbutton (green triangle) at the top.

This will launch your app in a separate window.

Interact with your app (e.g., click the button) to see it in action.



**Step 7**:

Save the App

Save your app by clicking the Save button or using Ctrl + S.

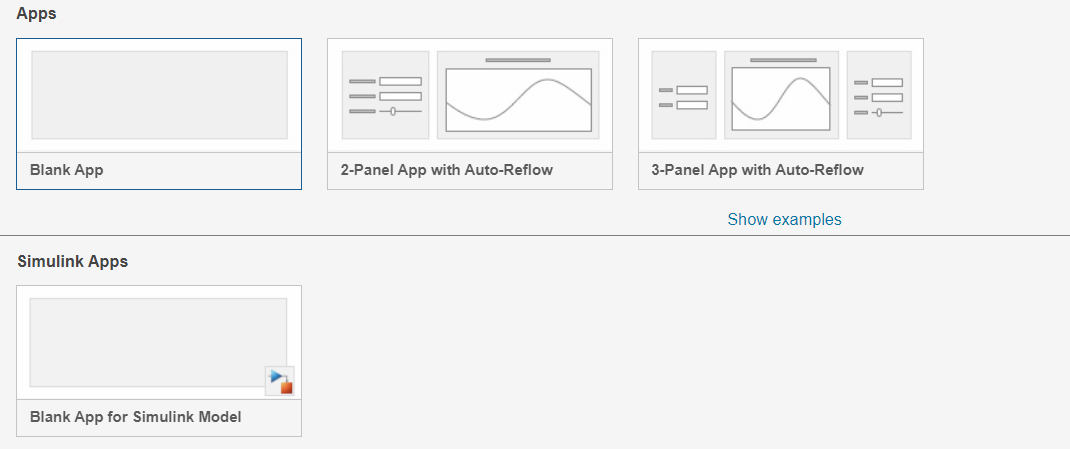
Give your app a name, and MATLAB will save it with a .mlappextension.

**Task1 {a} - Making of a Simple Dice**

Creating a simple Dice in MATLAB involves several steps, including designing the user interface (if desired), handling user inputs, and implementing the basic arithmetic operations. Here's a step-by-step guide to creating a basic text-based calculator in MATLAB:

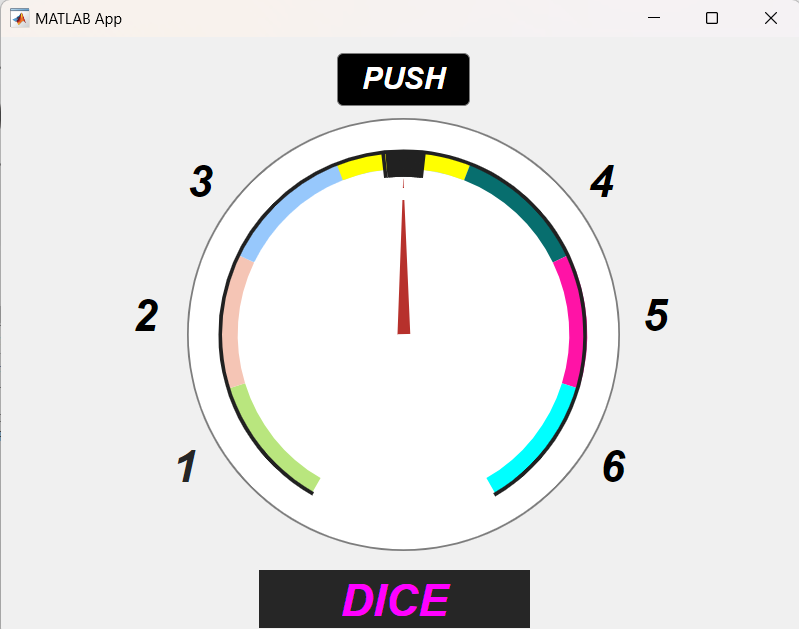
***Step 1:* Create a new App in MATLAB App Designer**

1. **Open MATLAB**: Launch MATLAB software.
2. **Open App Designer**: Type appdesigner in the MATLAB command window and press Enter.
3. **Create New App**: In the App Designer, click on "New App" to create a new app.



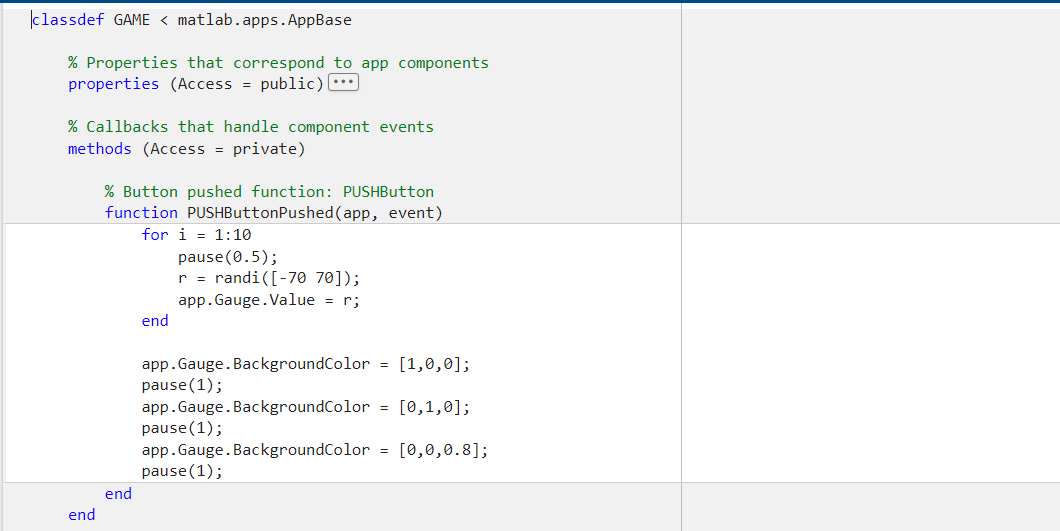
**Step 2: Design the GUI**

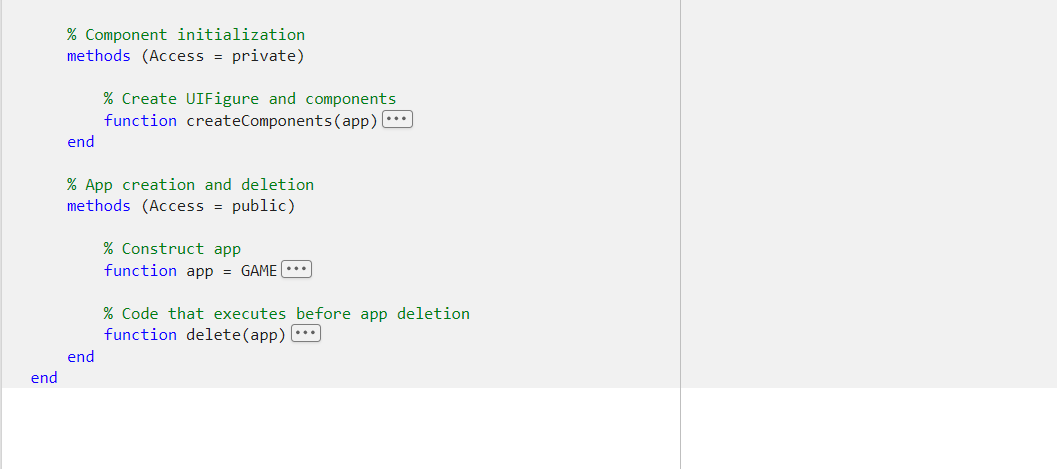
1. **Layout Components**: Drag and drop components onto the blank canvas to design interface.
   * Drag a "Button" (ButtonPushedFcn will be used to define what happens when the button is clicked).
   * Drag a "Label" (Text property will be used to display the dice result).
2. **Arrange Components**: Arrange the button and label on the design canvas as prefer.



***Step 3:* Add Callback Functionality**

1. **Define Callback Function for the Button**:
   * Double-click on the button to open the Code View for the callback function.
   * MATLAB will create a function like ButtonPushed(app, event) where app is your app's handle.
2. **Write MATLAB Code for Generating Random Number**:
   * Inside the ButtonPushed function, add the following MATLAB code:





***Step 4:* Run and Test Your App**

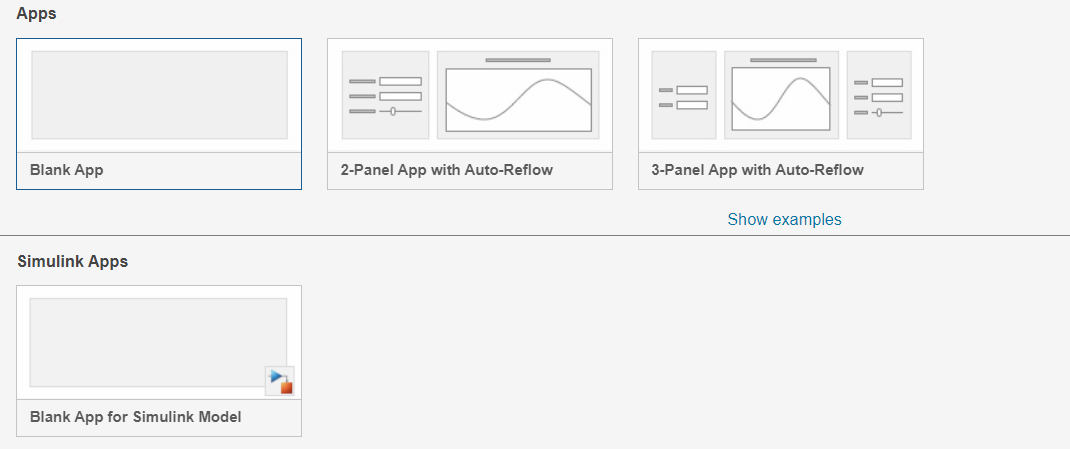
1. **Save the App**: Save your app by clicking the save icon in the App Designer.
2. **Run the App**: Click on the green "Run" button in the App Designer to run in app.
3. **Test the Dice Roll**: Click the button in app to generate a random number (simulate rolling a dice).



***Task1 {b }* – *GUI (Graphical User Interface) App***

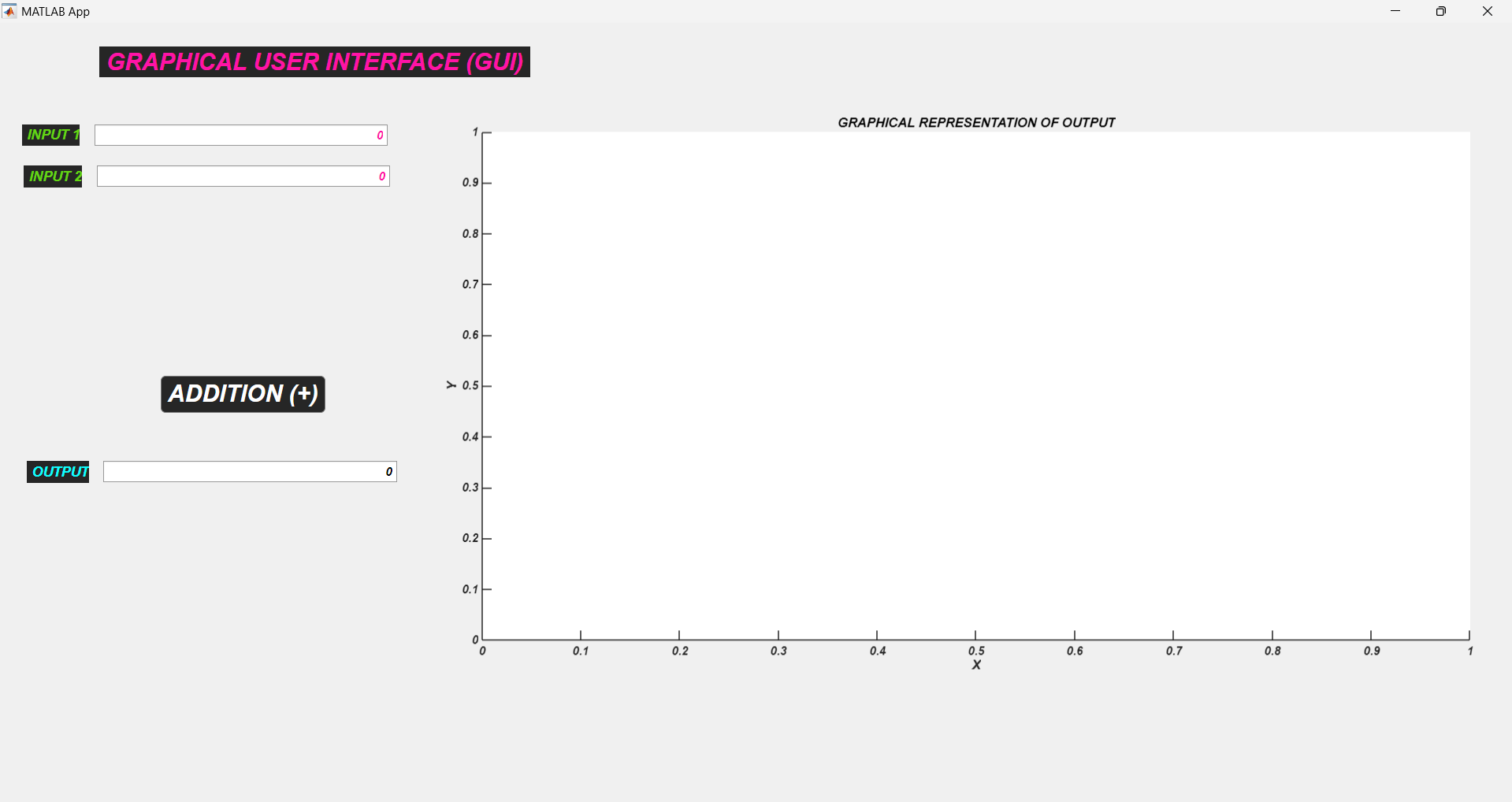
GUI : It is a “ graphical user interface” example in MATLAB App Designer. In this we give two numeric input values and  it gives an output which is the sum of these two numeric values by the help of ADDITION Button ,also it represents the output graphically  in 3D on a two dimensional plane using UIAxes feature involves several steps. Below is a detailed step-by-step guide to achieve this:

***Step 1:* Create a New App in MATLAB App Designer**

1. **Open MATLAB**: Launch MATLAB software.
2. **Open App Designer**: Type appdesigner in the MATLAB command window and press Enter.
3. **Create New App**: In the App Designer, click on "New App" to create a new app.
4. 

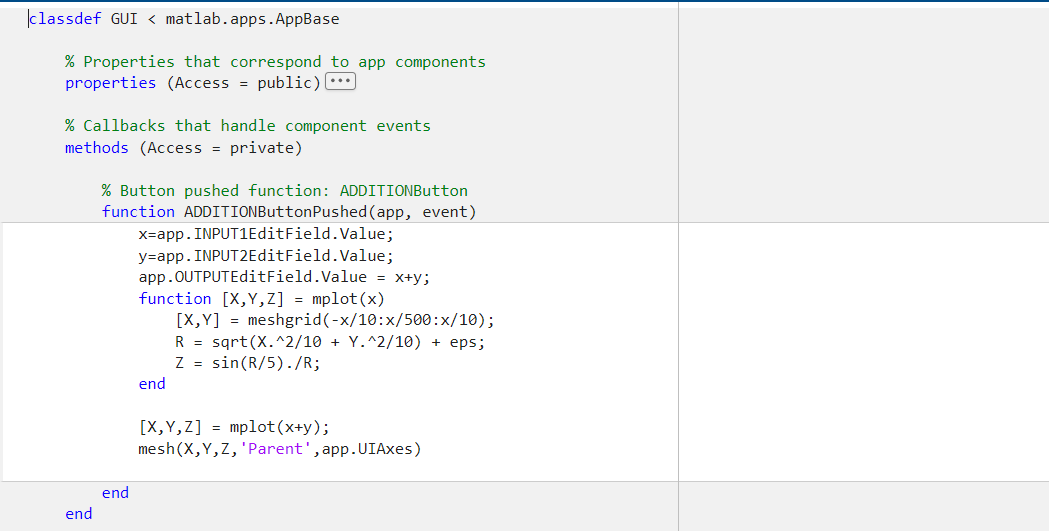
***Step 2:* Design the GUI**

1. **Layout Components**: Drag and drop components onto the blank canvas to design interface.
   * Drag two "NumericEditField" components (NumericEditField will be used to input numbers).
   * Drag a "Button" (ButtonPushedFcn will be used to define what happens when the button is clicked).
   * Drag a "UIAxes" (UIAxes will be used to plot the 3D graph).
2. **Arrange Components**: Arrange the components on the design canvas as prefer**.**



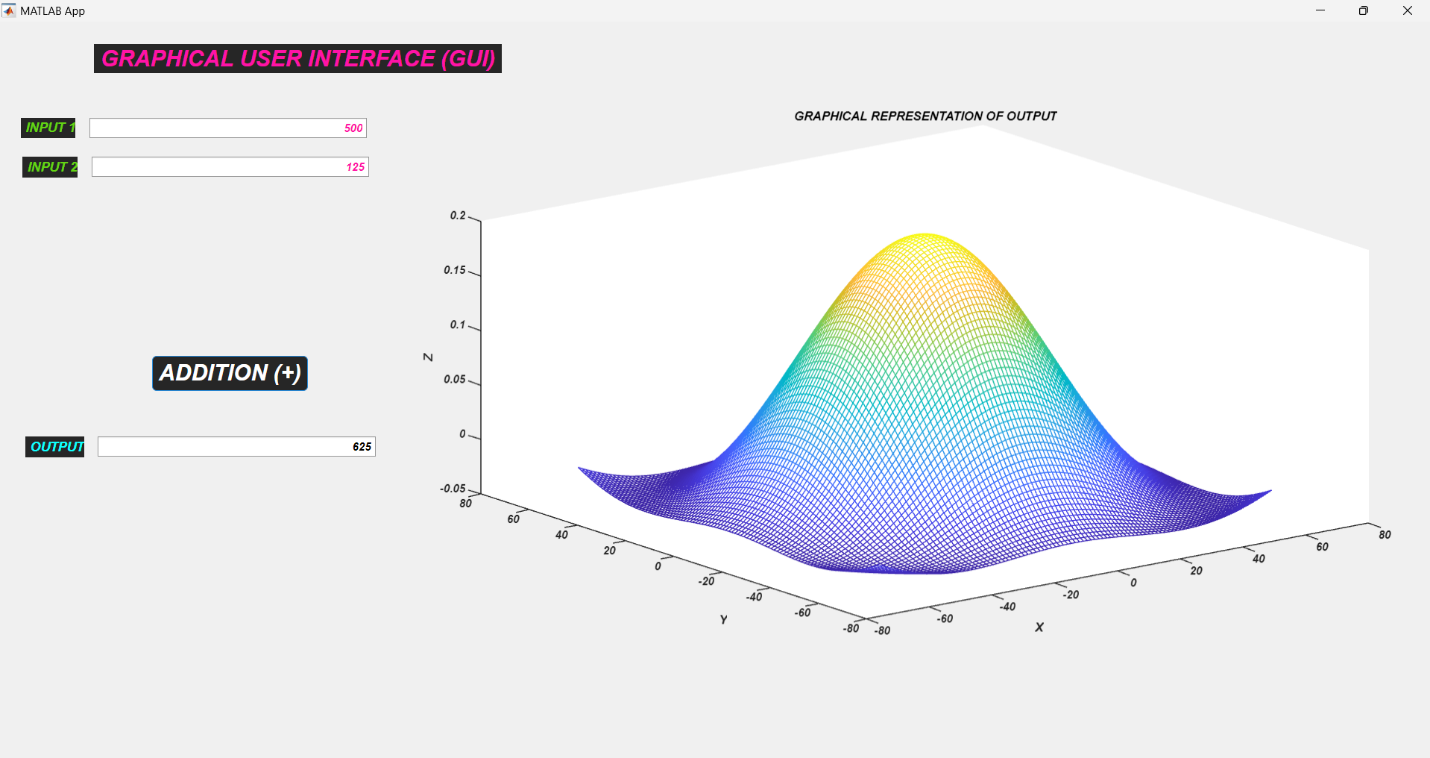
***Step 3:* Add Callback Functionality**

1. **Define Callback Function for the Button**:
   * Double-click on the button to open the Code View for the callback function.
   * MATLAB will create a function like ButtonPushed(app, event) where app is handle.
2. **Write MATLAB Code for Adding and Plotting**:
   * Inside the ButtonPushed function, add the following MATLAB code:



**Step 4: Run and Test Your App**

1. **Save Your App**: Save the app by clicking the save icon in the App Designer.
2. **Run Your App**: Click on the green "Run" button in the App Designer to run the app.
3. **Test the Functionality**: Enter numbers into the numeric edit fields and click the button to see the 3D plot based on the addition of those numbers.



**TASK-2: *Text File Browser App***

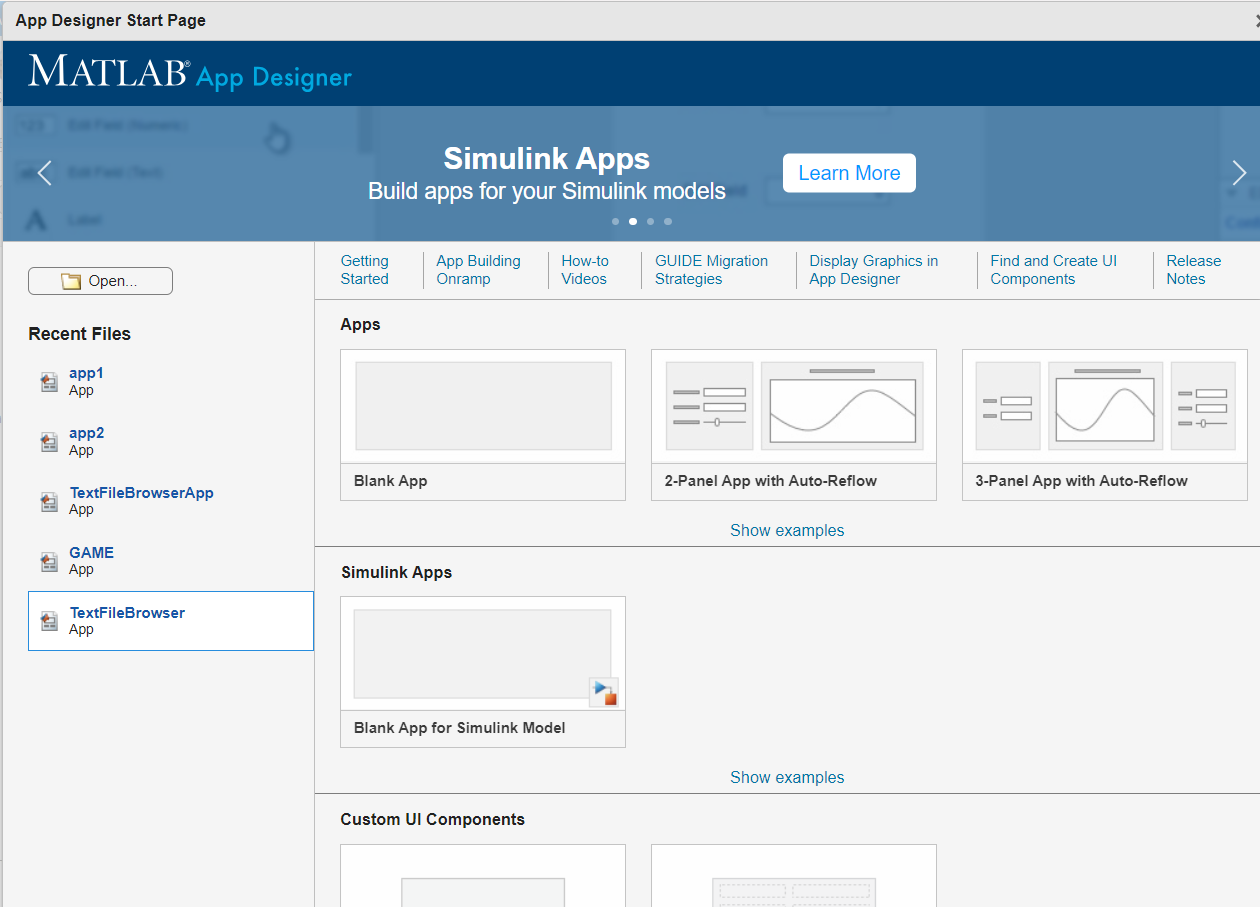
**Step 1 :**

Open MATLAB.

Open App Designer

In the MATLAB Command Window, type appdesigner and press Enter.

Create a new blank app by selecting "Blank App".



***Step 2 :***

Design the User Interface

UI Components: Add the following components from the Component Library to the canvas:

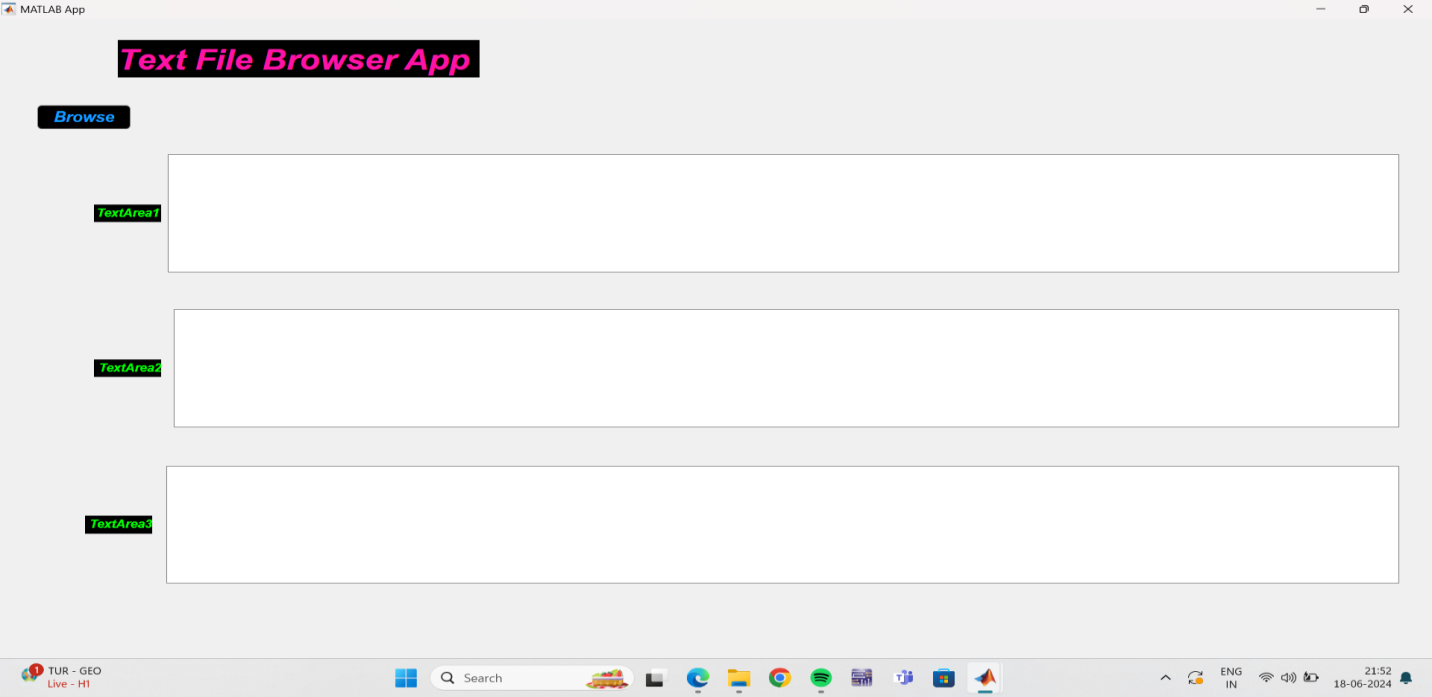
A Button to open the file selection dialog.

A Text Area to display the content of the selected file.

Customize Components:

Select the Button and change its text to "BROWSE".

Adjust the size and position of the Text Area to occupy a large portion of the app window, making it suitable for displaying text content.



***Step 3:***

Add Callback Functions

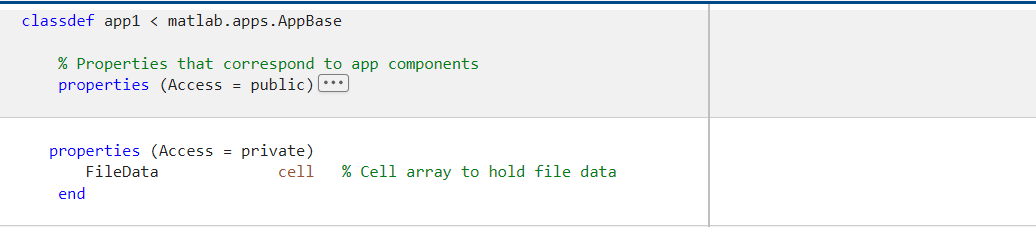
Button Callback: Select the Button on the canvas.

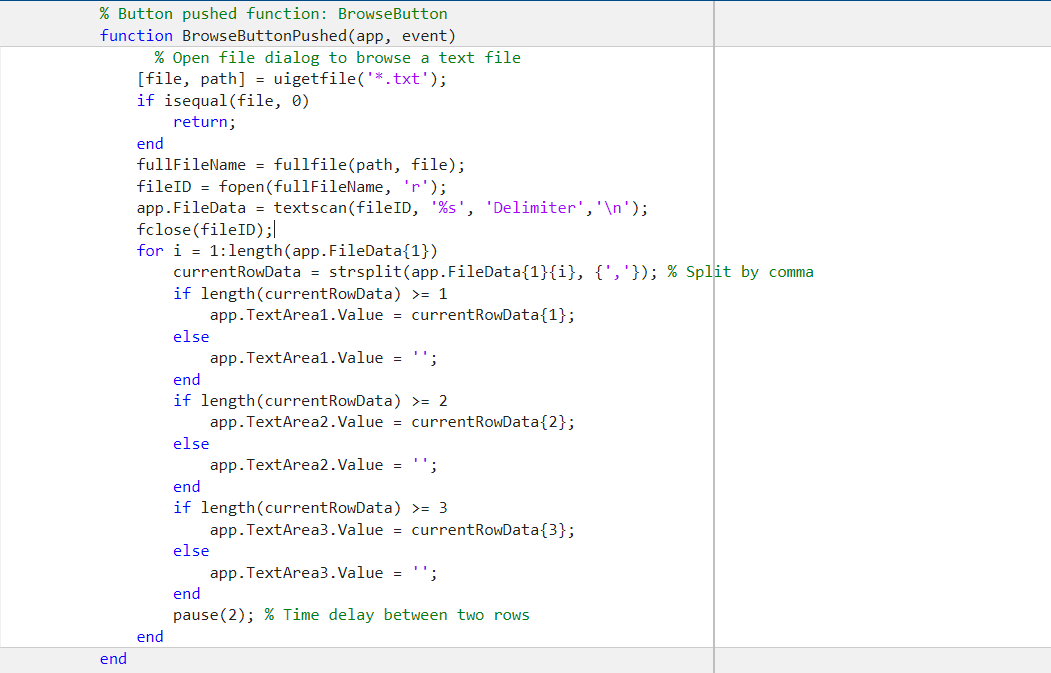
In the Property Inspector, scroll down to the Callbacks section.

Click the "+" next to ButtonPushedFcn to create a new callback for the button's pushed event.

This will switch you to the Code View, and MATLAB will generate a function template for the button's pushed event.

***Step 4:***





***Step 5:***

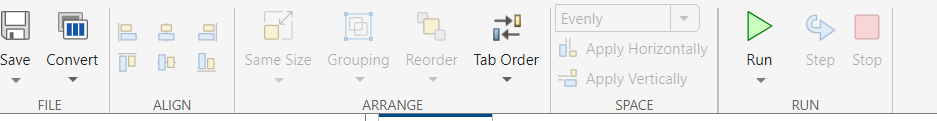
Save and Run the App

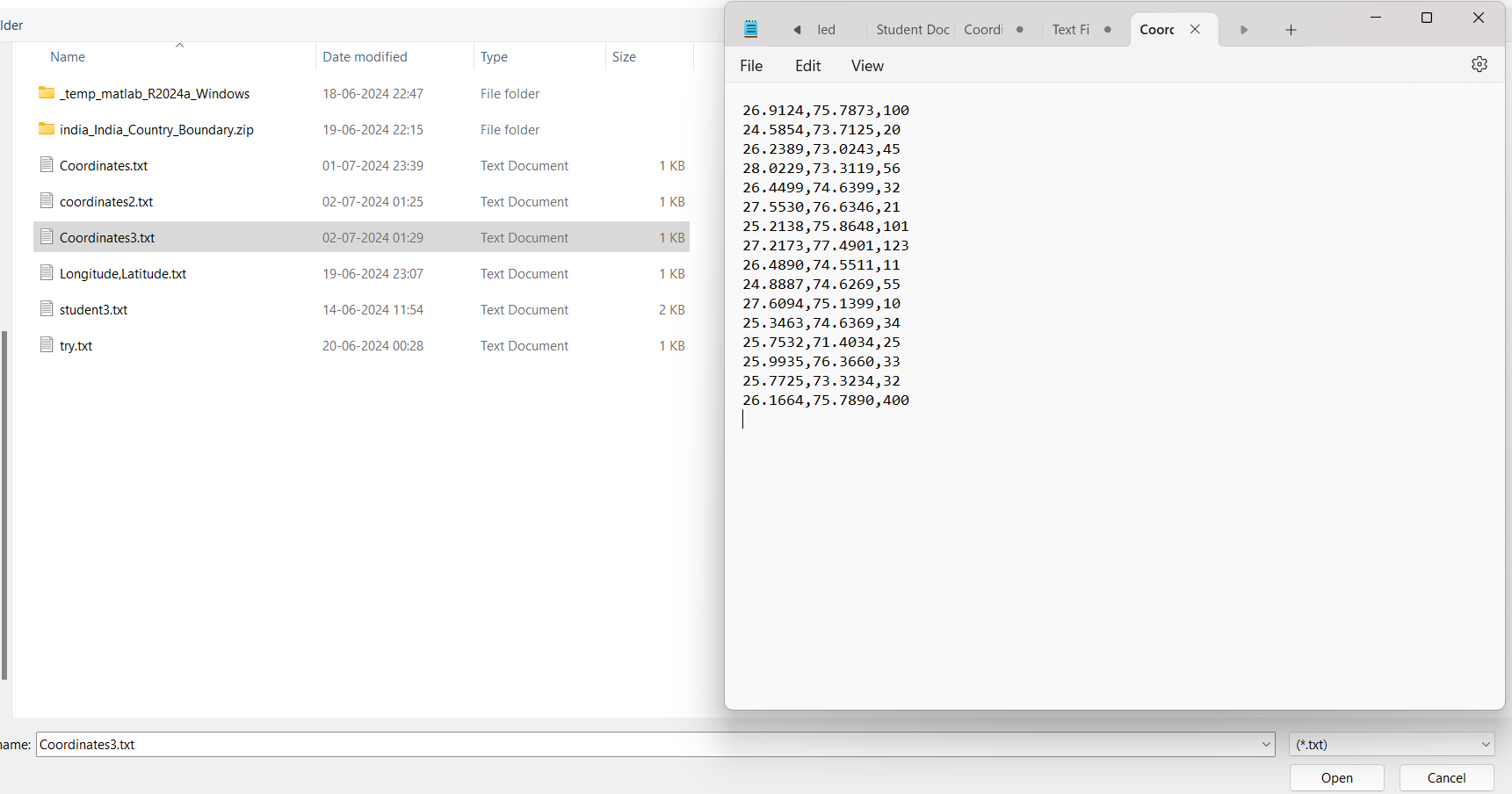
Save your app by clicking the Save button or using Ctrl + S.

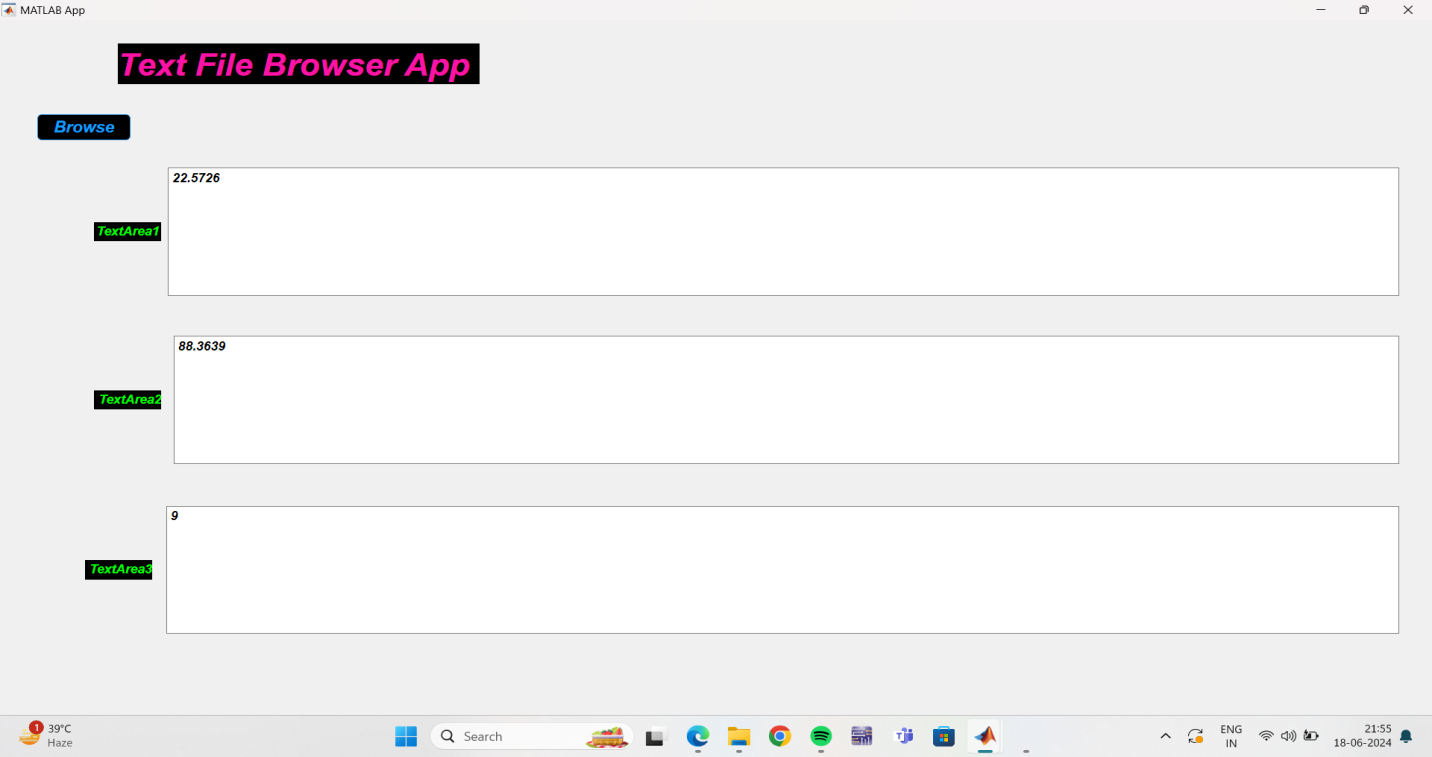
Give your app a name, and MATLAB will save it with a .mlappextension.

Click the Run button (green triangle) at the top to launch your app.

Interact with your app by clicking the "Open File" button, selecting a text file, and viewing its content in the text area.







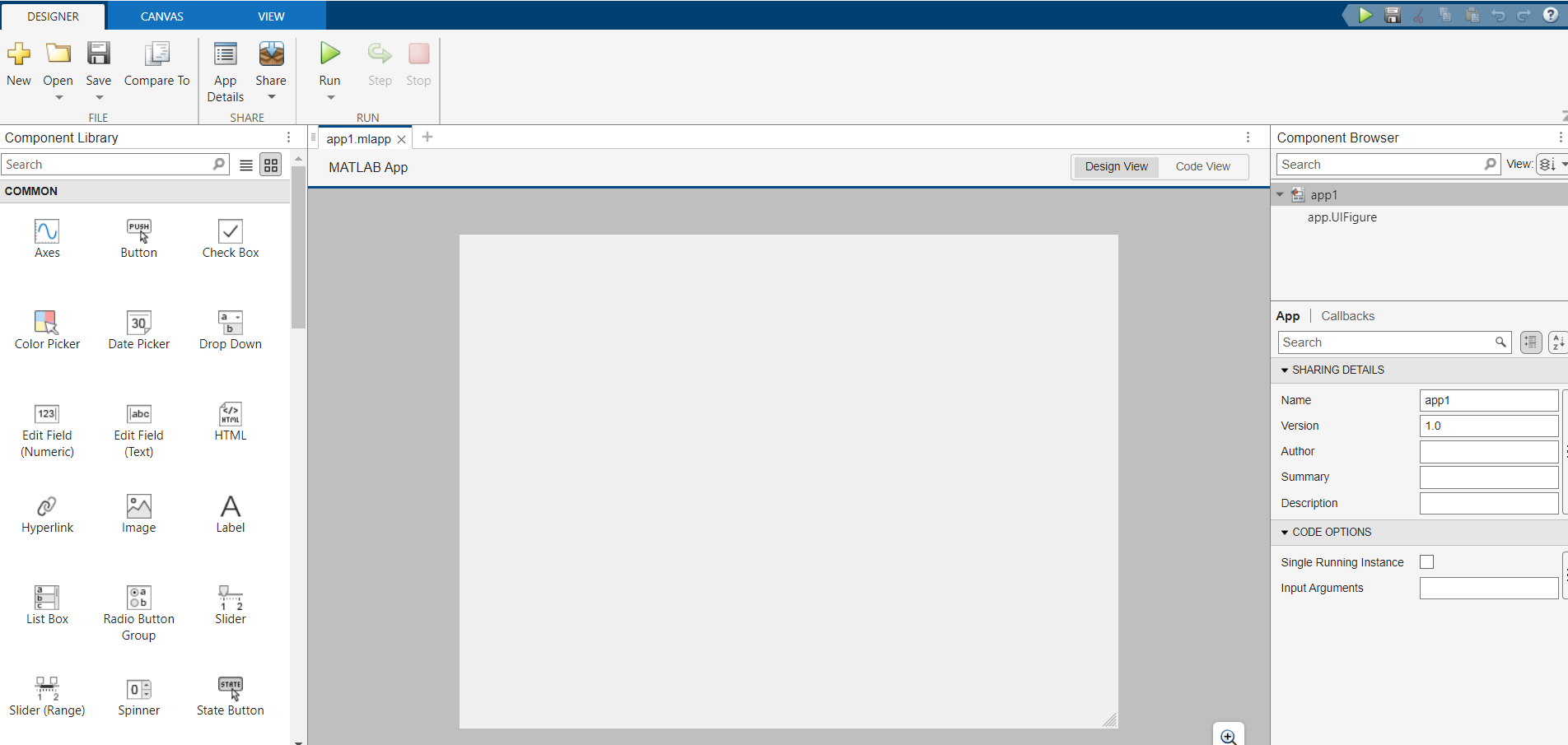
***Task 3:* *Text File Browser and Plotting App***

***Step 1:***

Setup the MATLAB Environment:

Open MATLAB.

Create a new app using App Designer.



***Step 2:***

Design the Interface:

Open App Designer: >> appdesignerin the command window.

Create a new app with a blank interface.

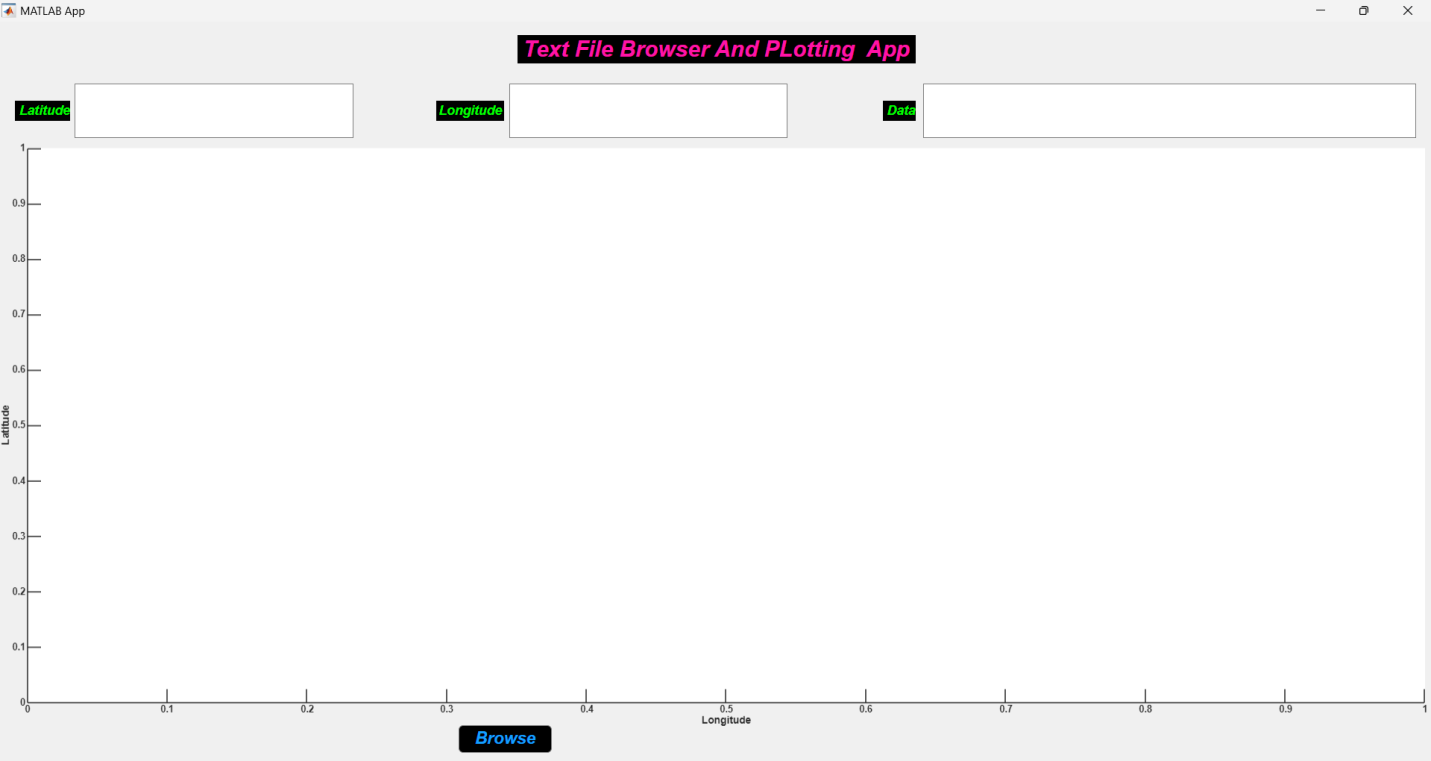
Add the necessary UI components:

Button for browsing and loading the text file.

Text area or label to display the path of the loaded file.

Axes for displaying the plot.

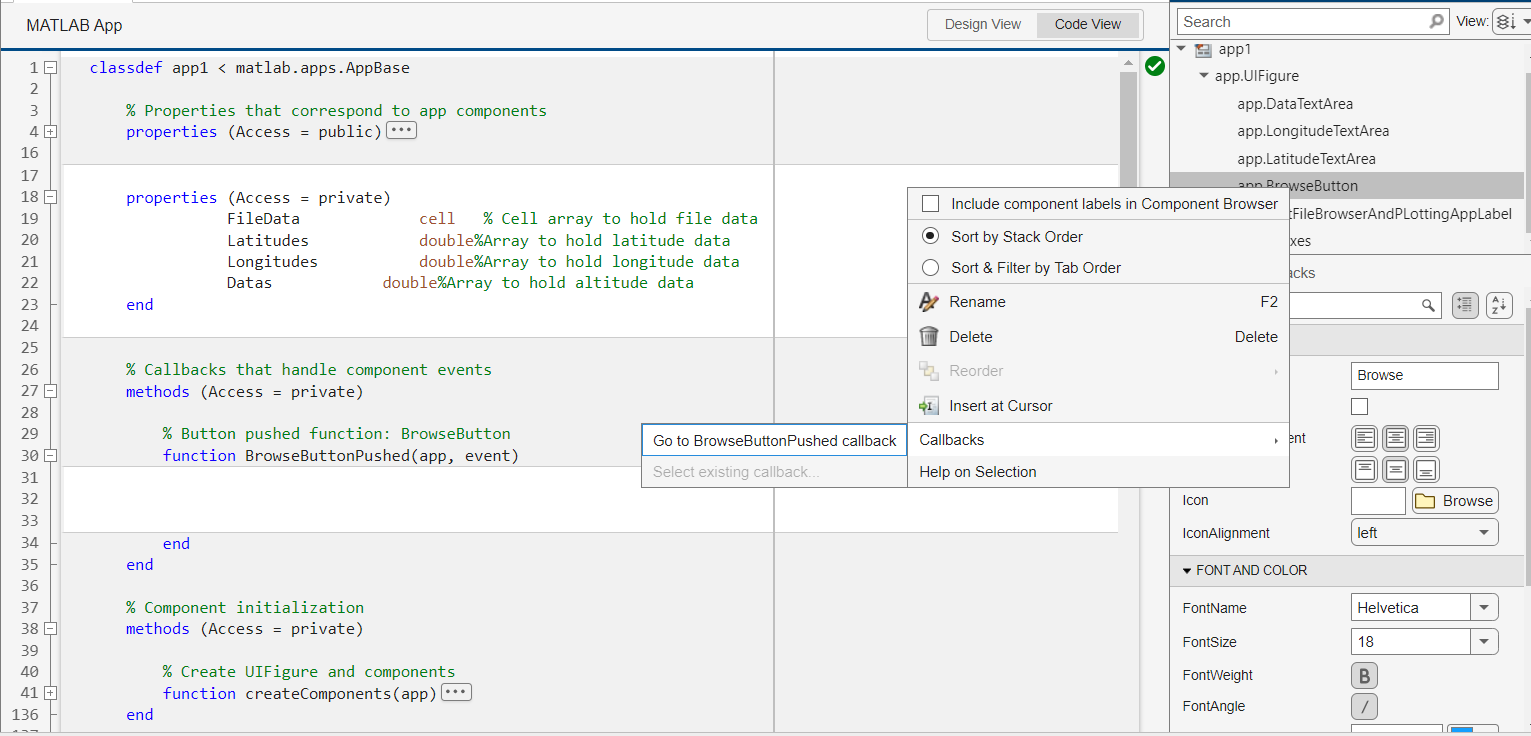
Optionally, a button to clear the plot or reset the app.



***Step 3:***

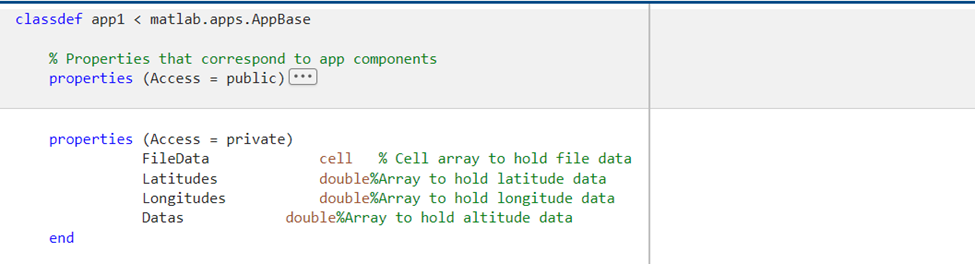
Set Up Callbacks:

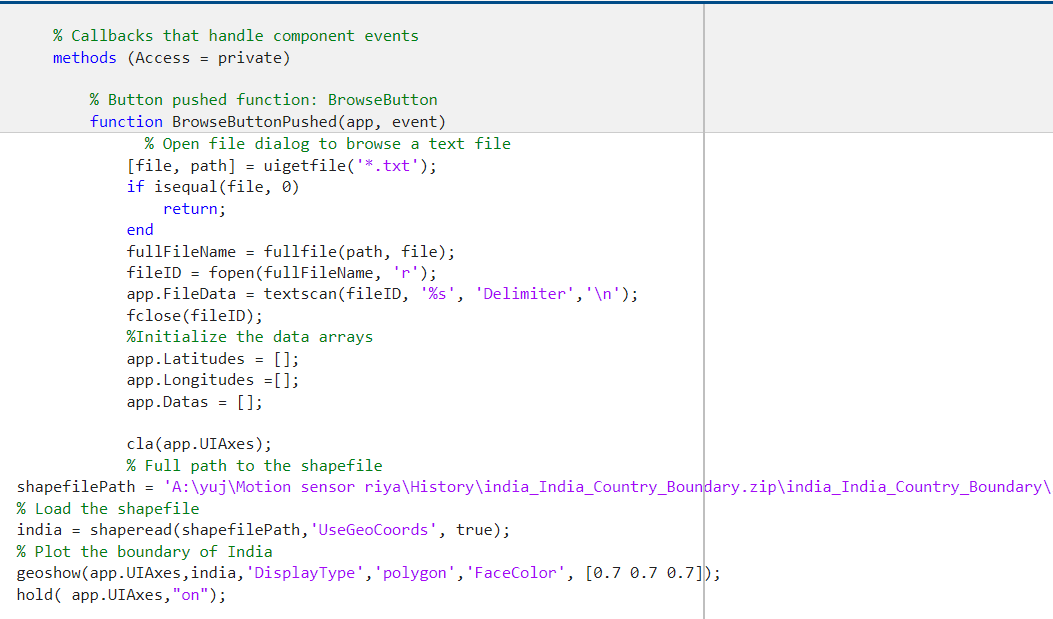
In the Code View, add callbacks to handle user interactions, such as pressing the browse button.

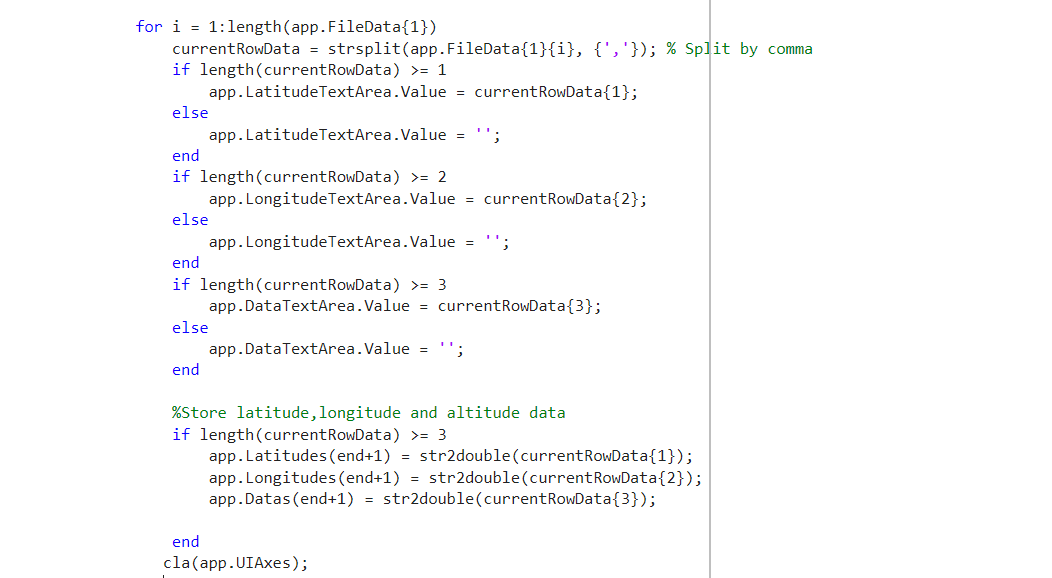


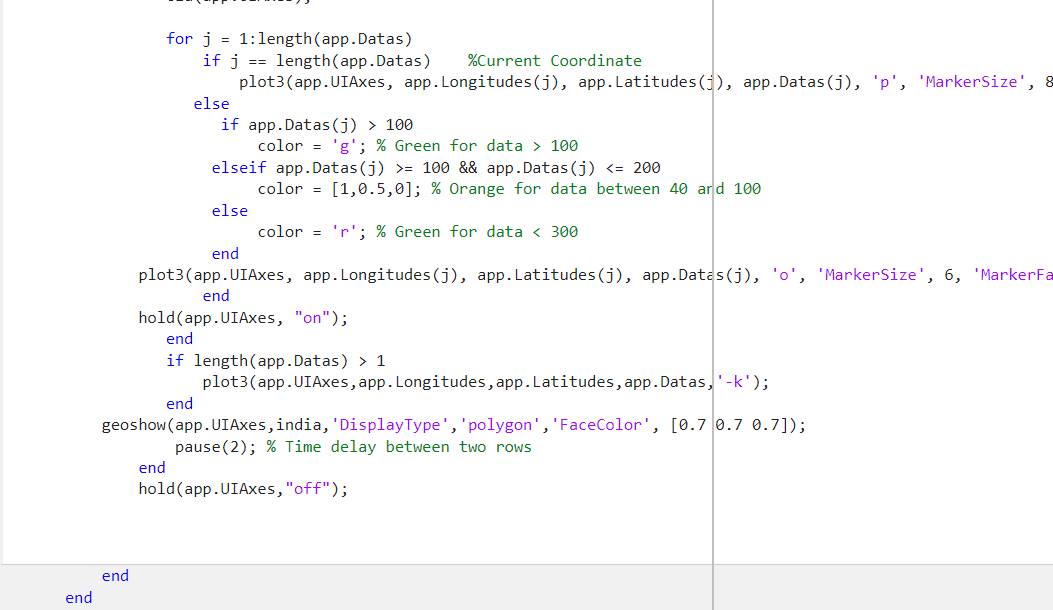
***Step 4:***

Write the Callback Functions: Define the function to open a file dialog, read the file, extract latitude and longitude data, and plot the data on a map.

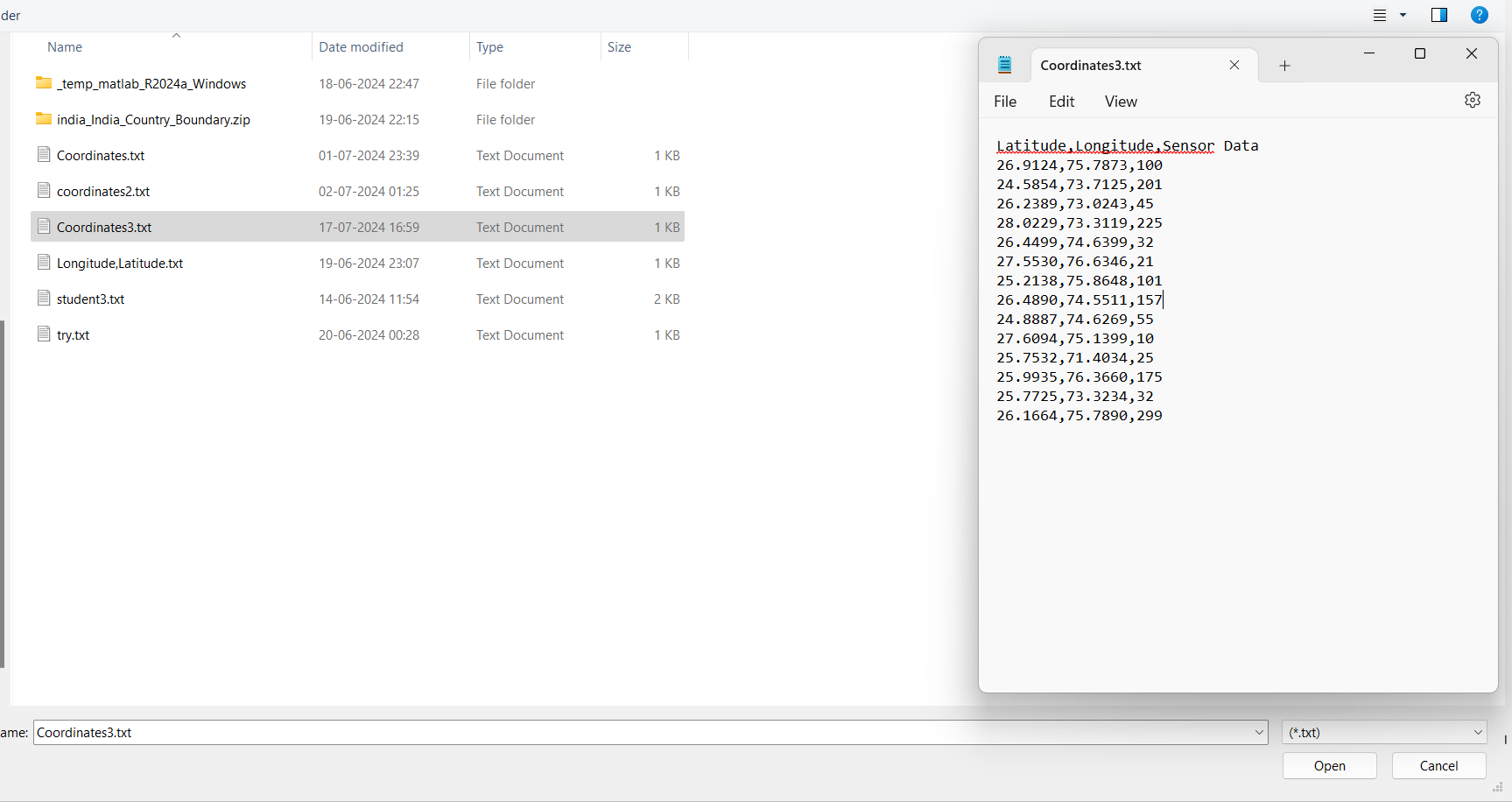


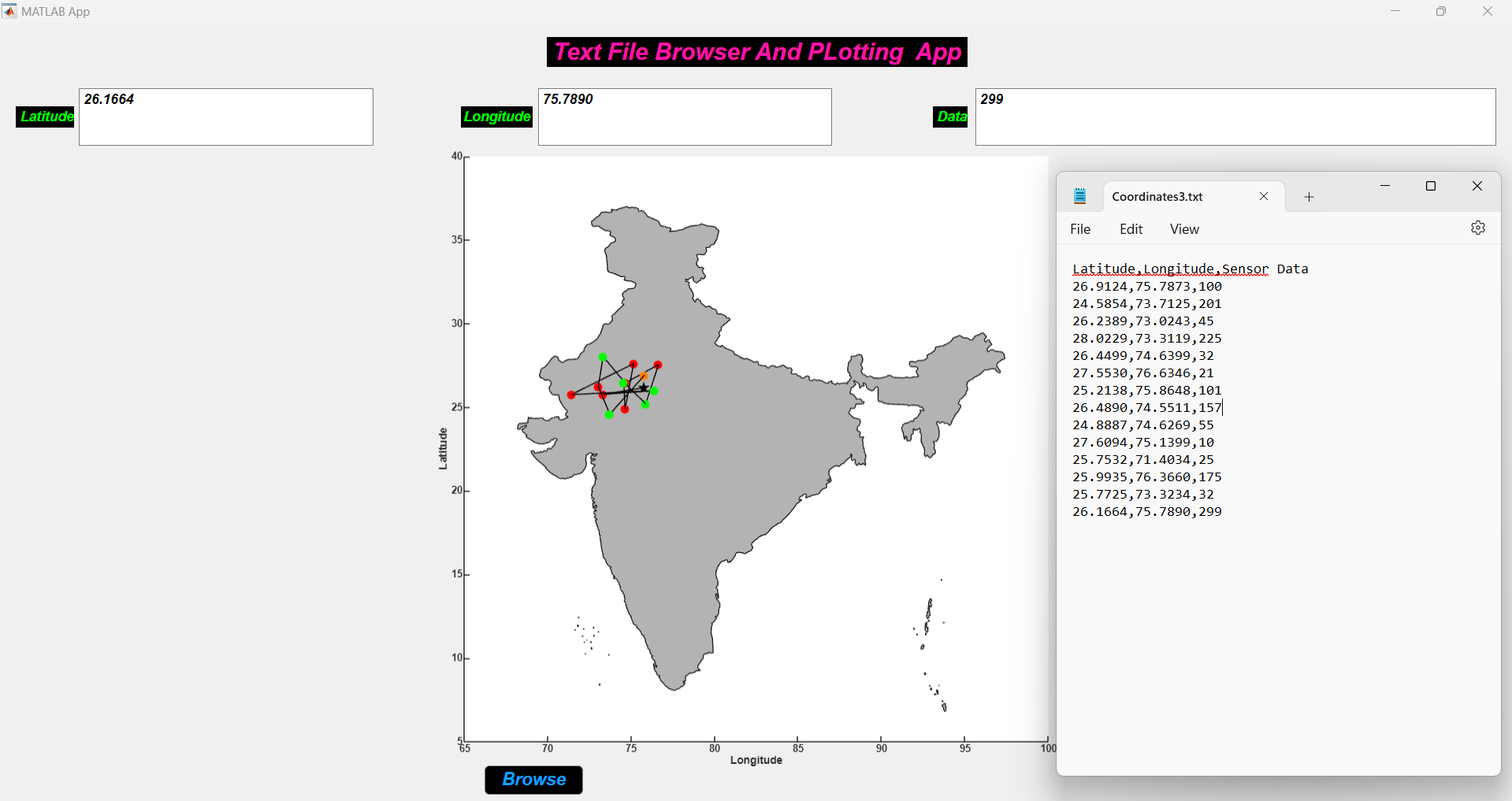


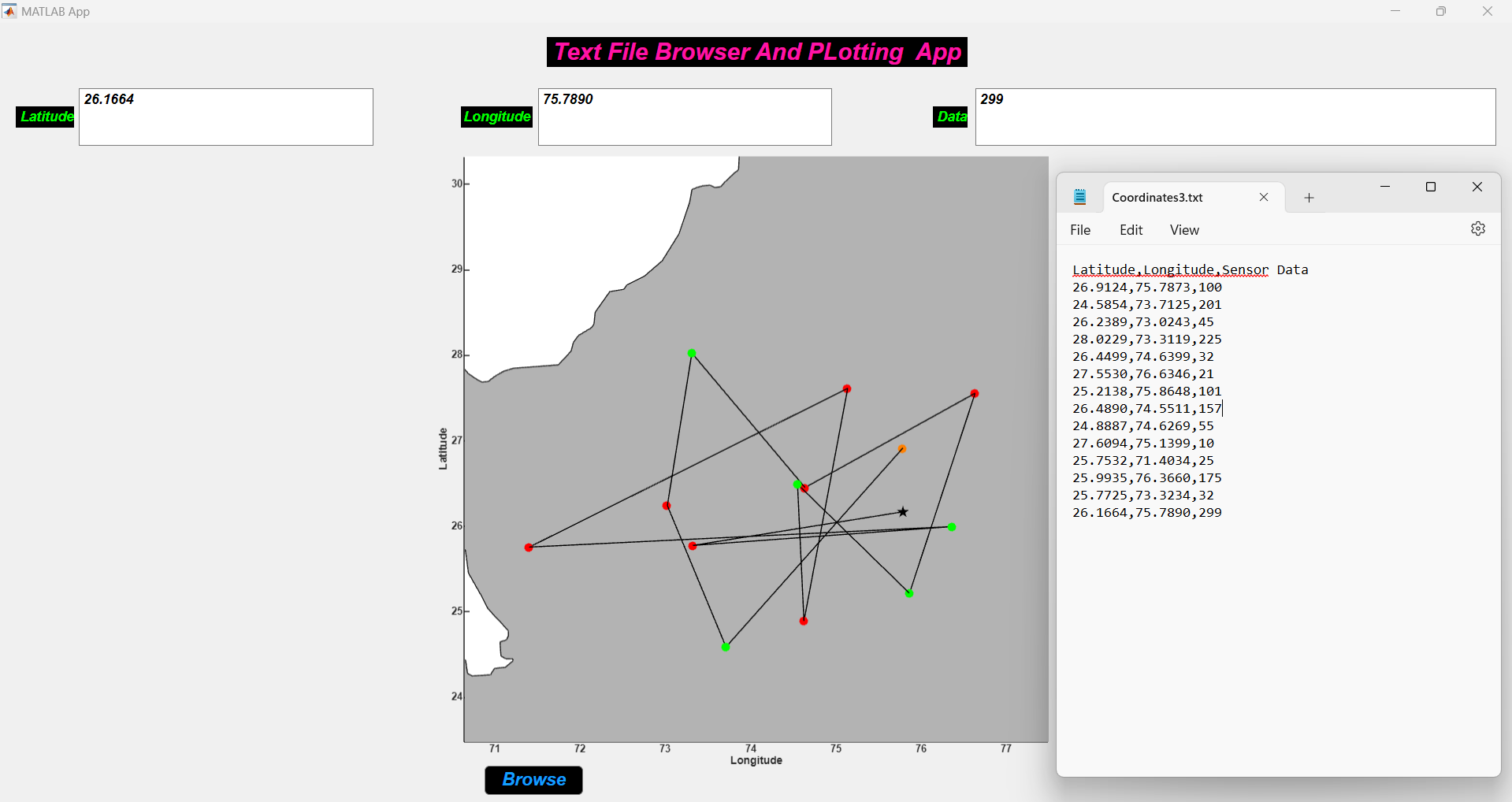




***Step 5 :*** how the app will look







***Task-4: Concentric Circles Plotting App***

***Step 1:***

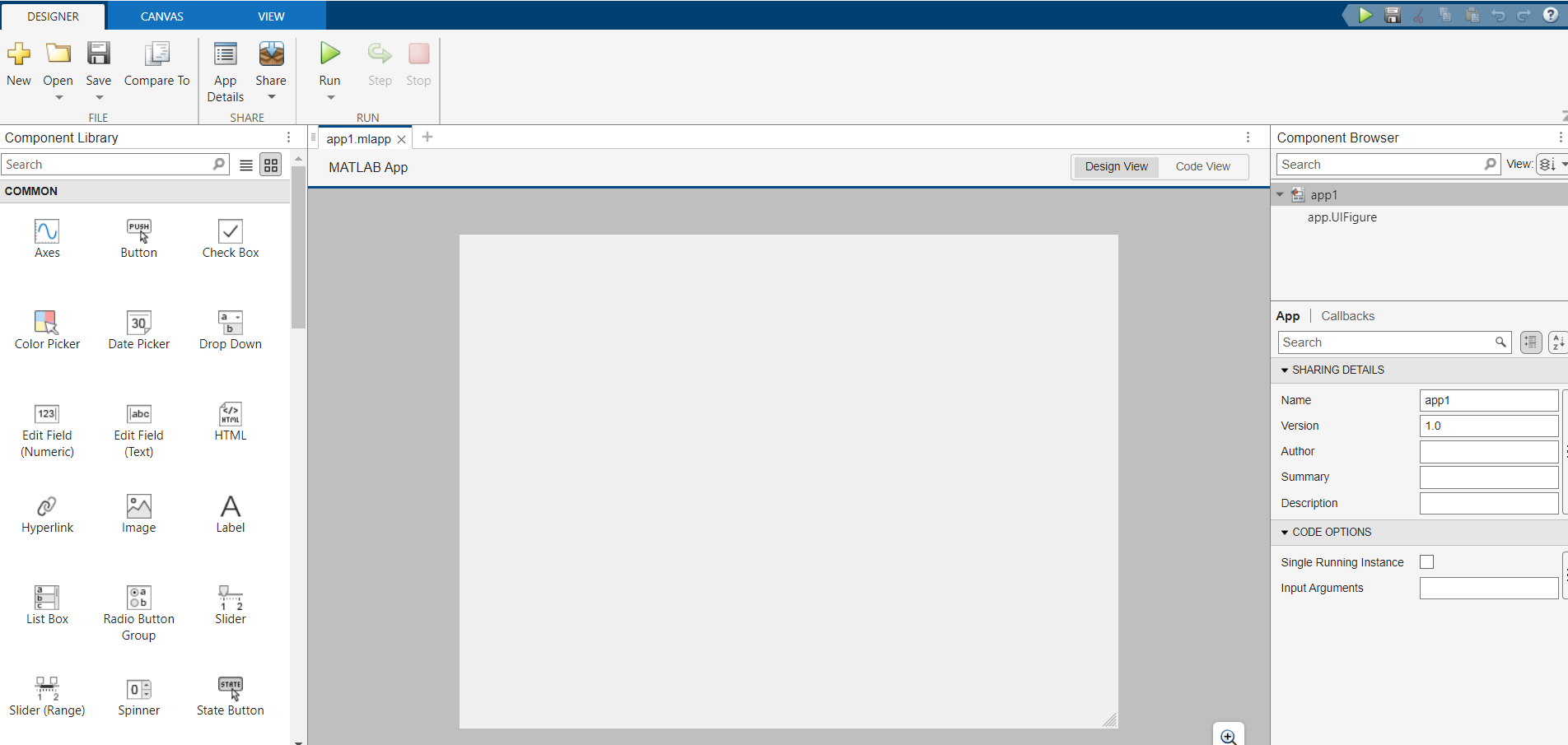
Setup the MATLAB Environment:

Open MATLAB.

Create a new script or function file.

Create the GUI:

Use App Designer to design the interface.



***Step 2 :***

Design the Interface:

Open App Designer: >> appdesignerin the command window.

Create a new app with a blank interface. Add the necessary UI components:

Text Label for instructions (e.g., "Enter initial radius:")

Numeric Edit Field for user to input the initial radius.

Text Label for instructions (e.g., "Enter difference between radii:")

Numeric Edit Field for user to input the difference between radii.

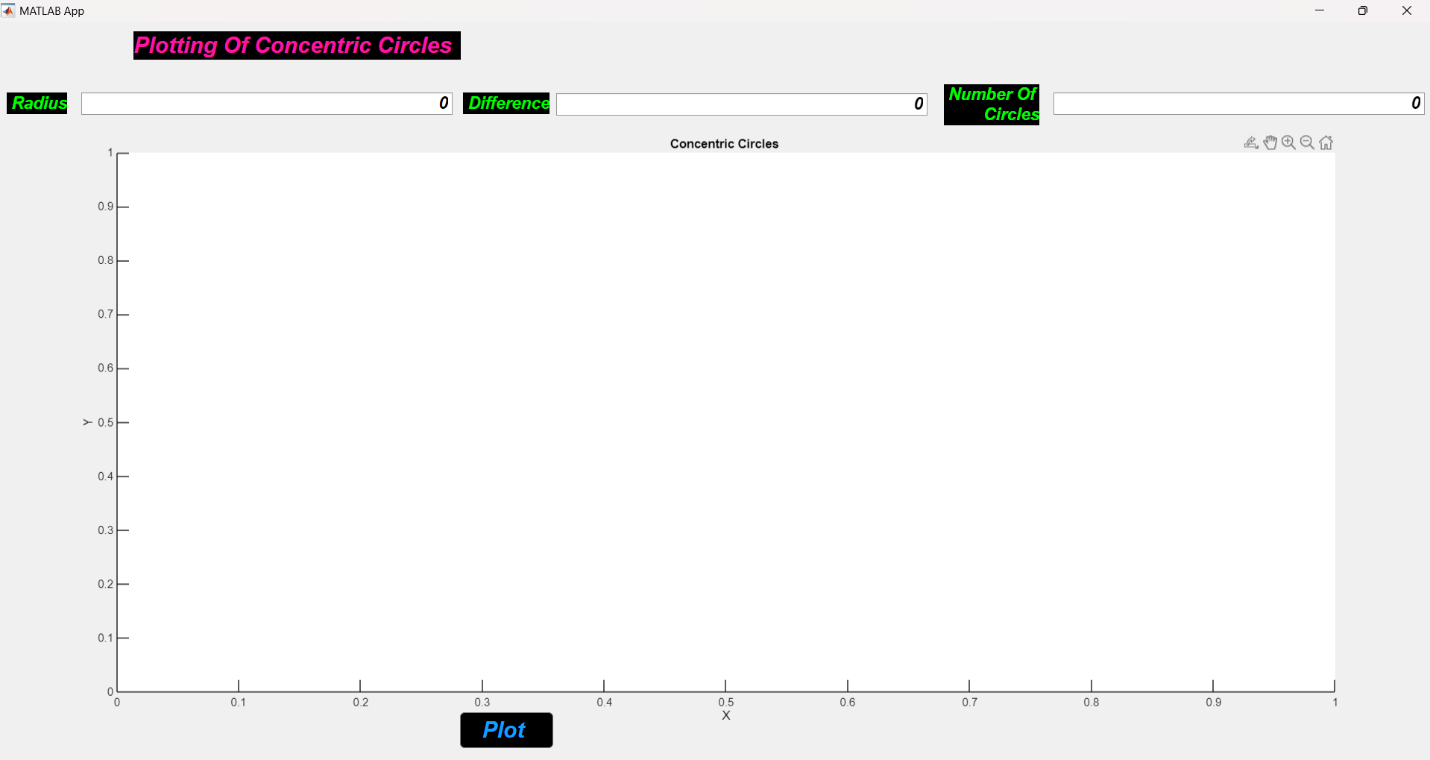
Text Label for instructions (e.g., "Enter number of circles:")

Numeric Edit Field for user to input the number of circles.

Button for plotting the circles.

Axes for displaying the plot.

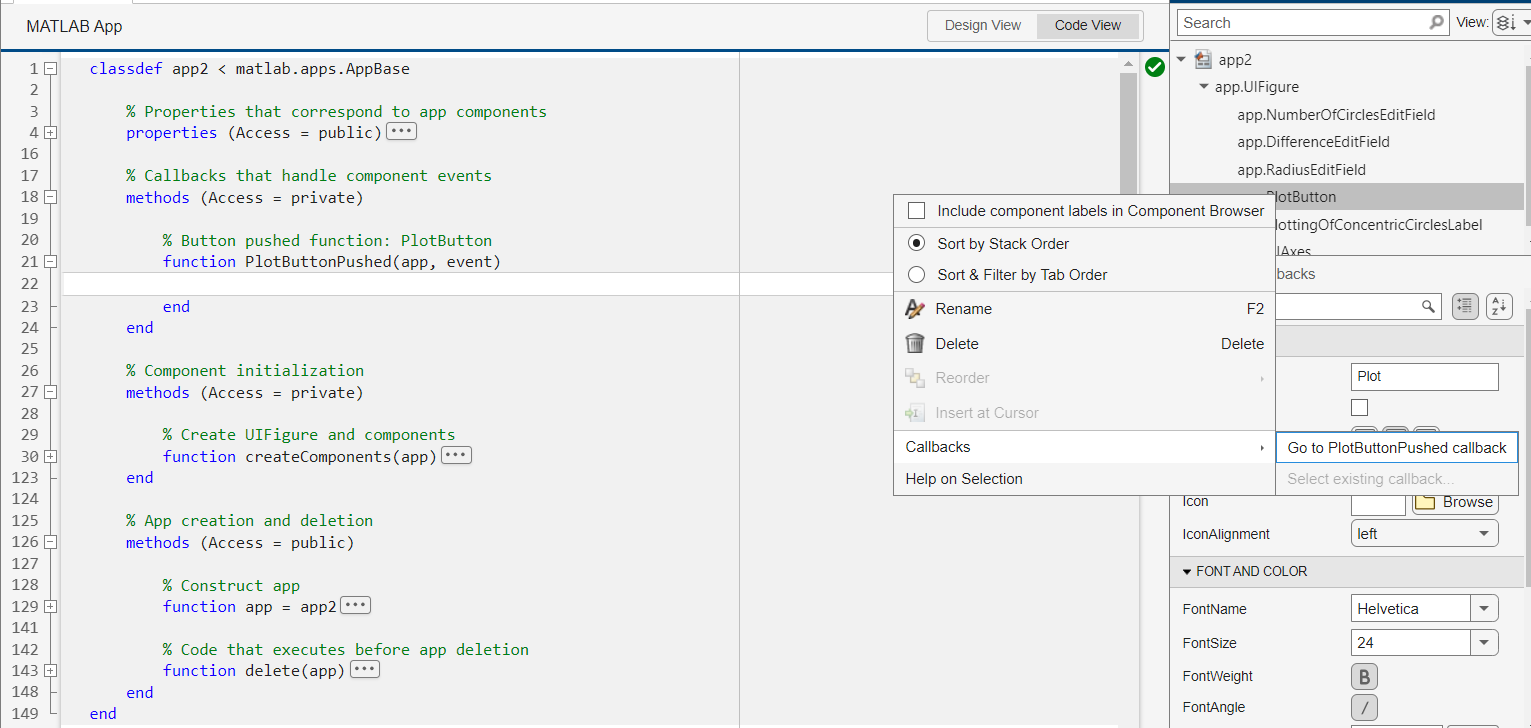
Text Label to show the number of circles.



***Step 3:***

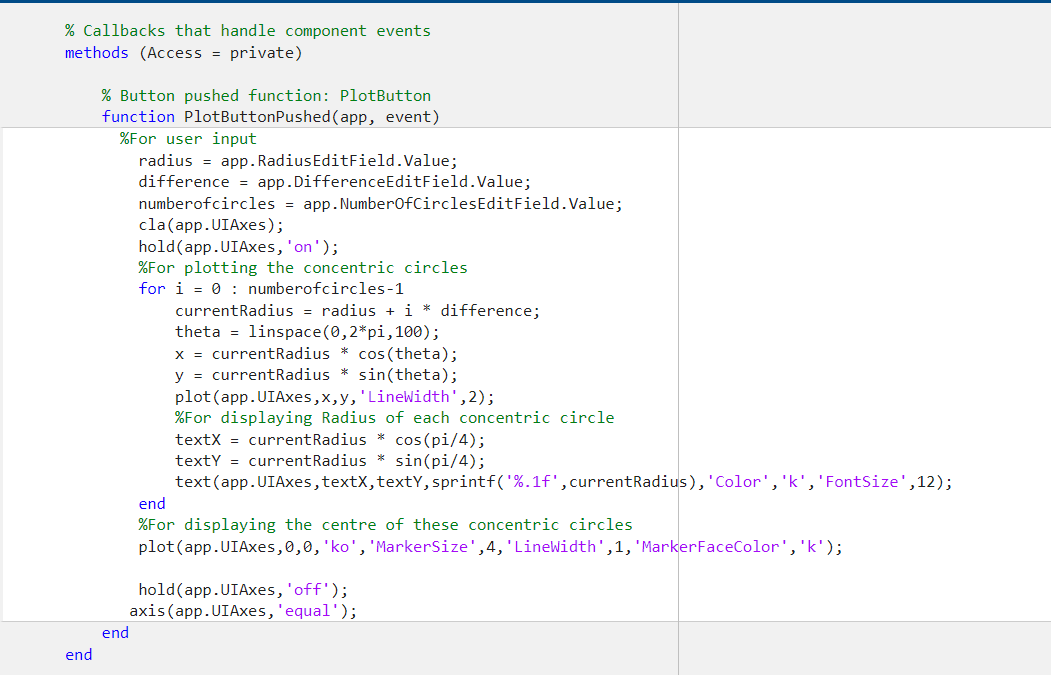
Set Up Callbacks:

In the Code View, add callbacks to handle user interactions, such as pressing the plot button.



***Step 4:***

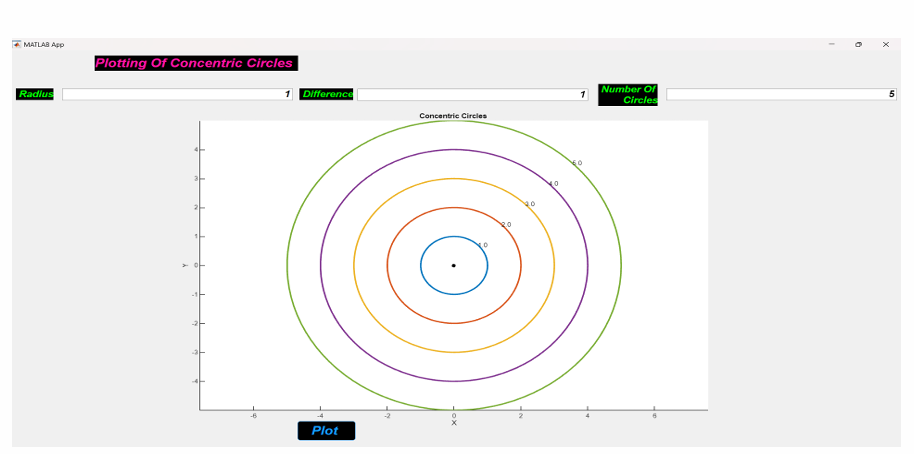
Write the Callback Functions:



***Step 5:***

Run the App:

Save and run the app. We should be able to input the initial radius, difference, and number of circles, plot the circles, and see the count. The app will look like this :



***Task-5 :*** ***Visualize the Aerial Surveillence Data on Map***

***Objective:***

The objective of this project is to visualize aerial surveillance data on a geographical map using MATLAB. This involves importing, processing, and plotting spatial data points representing surveillance activities onto a map interface. The visualization aims to provide spatial insights into surveillance patterns, coverage areas, and potential hotspots, utilizing MATLAB's Mapping Toolbox for effective data representation and analysis.

***Key Components:***

1. **Data Import and Preparation: Implementing methods to import surveillance data, ensuring it is in a suitable format for processing in MATLAB.**
2. **Mapping and Visualization: Using MATLAB's Mapping Toolbox to plot data points on a geographical map, considering aspects such as marker size, color coding based on surveillance attributes, and optional overlays such as satellite imagery for enhanced context.**
3. **Customization and Interactivity: Customizing the map appearance and potentially implementing interactive features like zooming, panning, and data point labeling to enhance user experience and data exploration.**
4. **Analysis and Insights: Leveraging spatial analysis capabilities of MATLAB to derive insights from the visualized data, such as identifying spatial trends, distribution patterns, and areas of high surveillance activity.**
5. **Documentation and Reporting: Documenting the process, including code snippets, methodologies, challenges encountered, and lessons learned, to facilitate reproducibility and knowledge transfer.**

***Step 1:***

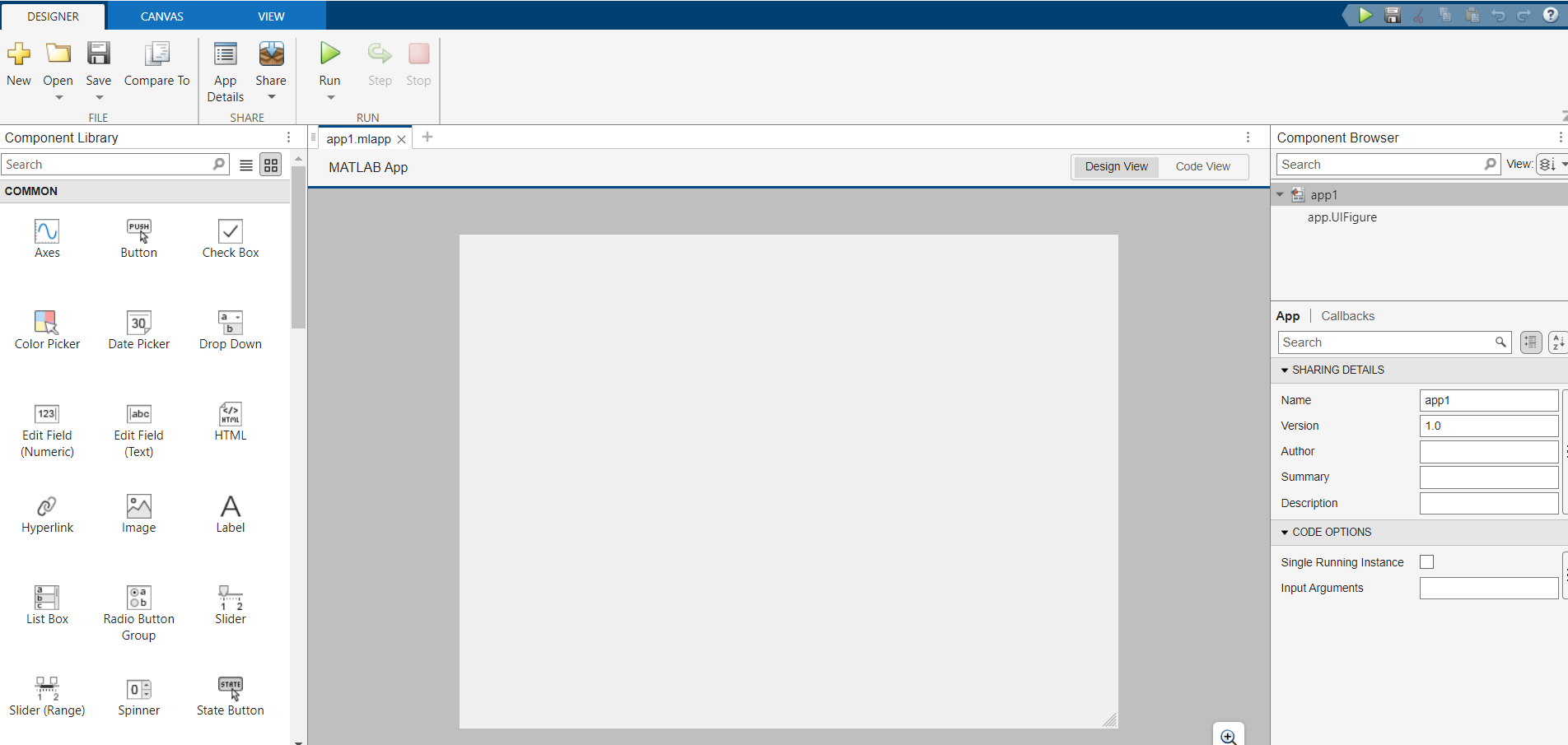
Setup the MATLAB Environment:

Open MATLAB.

Create a new script or function file.

Create the GUI:

Use App Designer to design the interface.

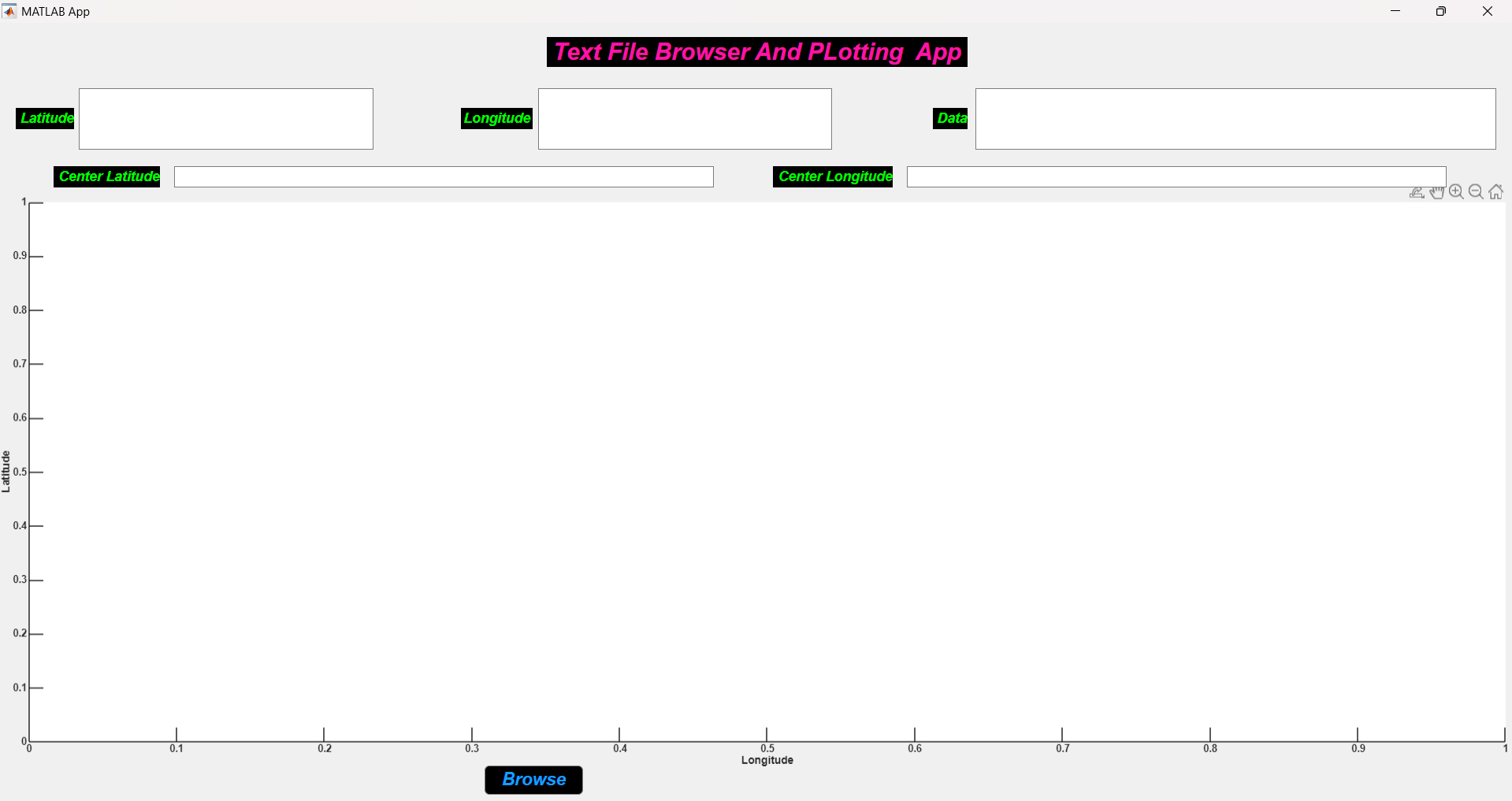


***Step 2 :***

Design the Interface:

Open App Designer: >> appdesignerin the command window.

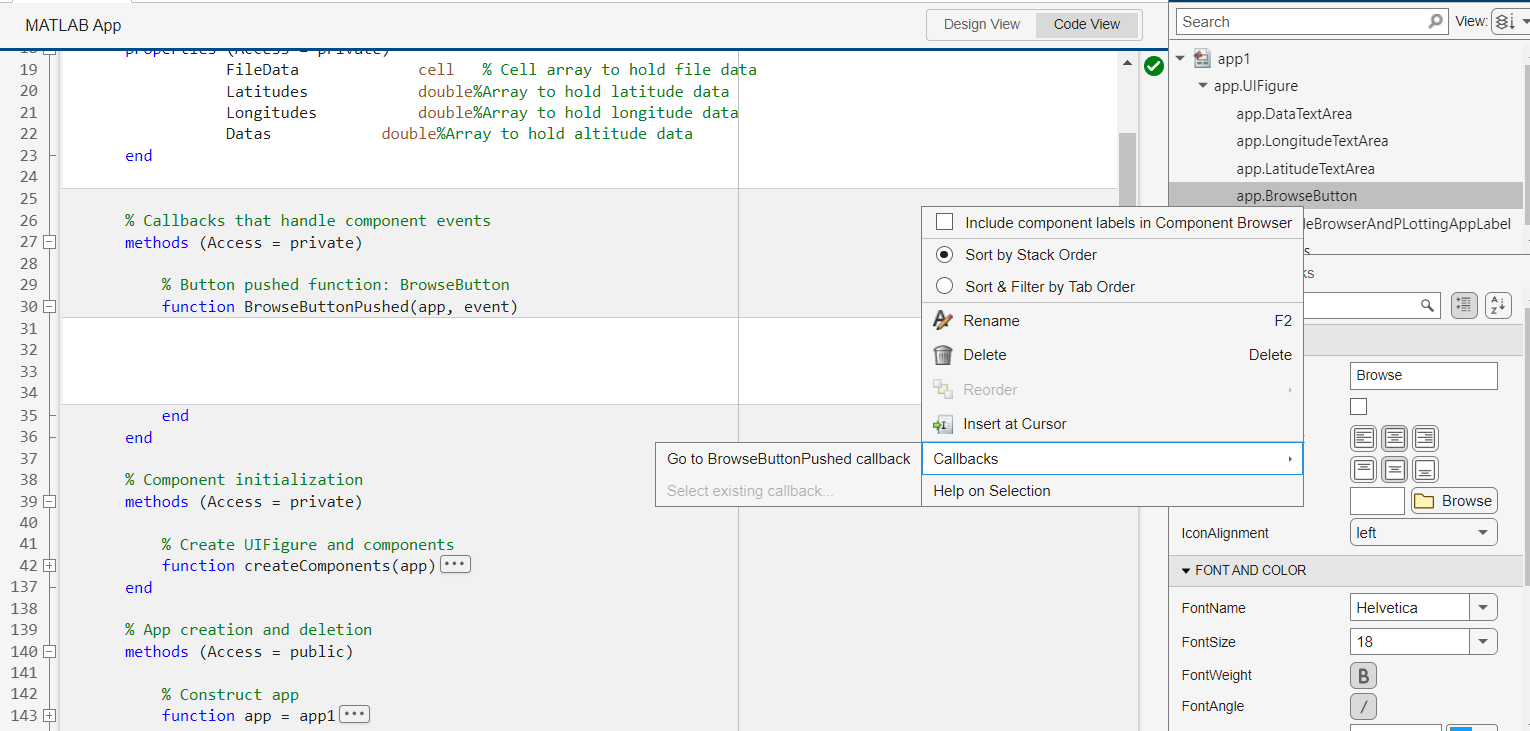
Create a new app with a blank interface. Add the necessary UI components:



***Step 3:***

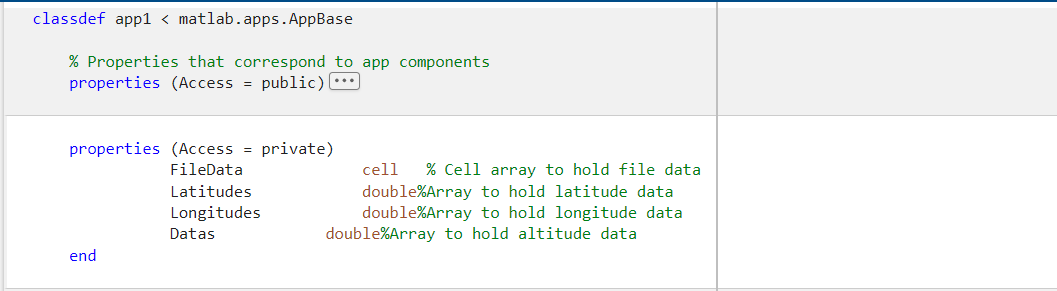
Set Up Callbacks:

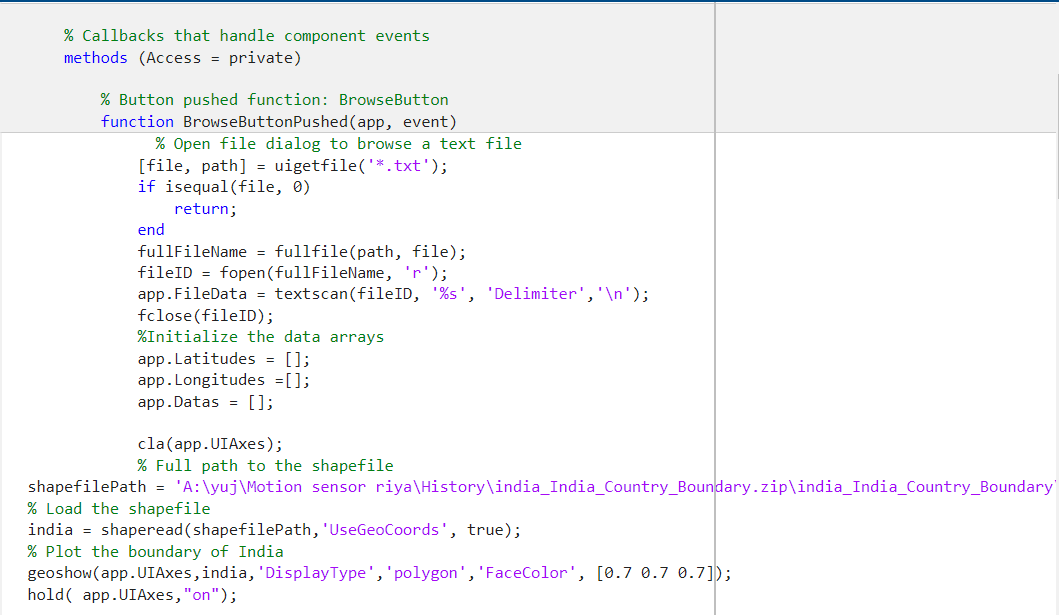
In the Code View, add callbacks to handle user interactions, such as pressing the plot button.

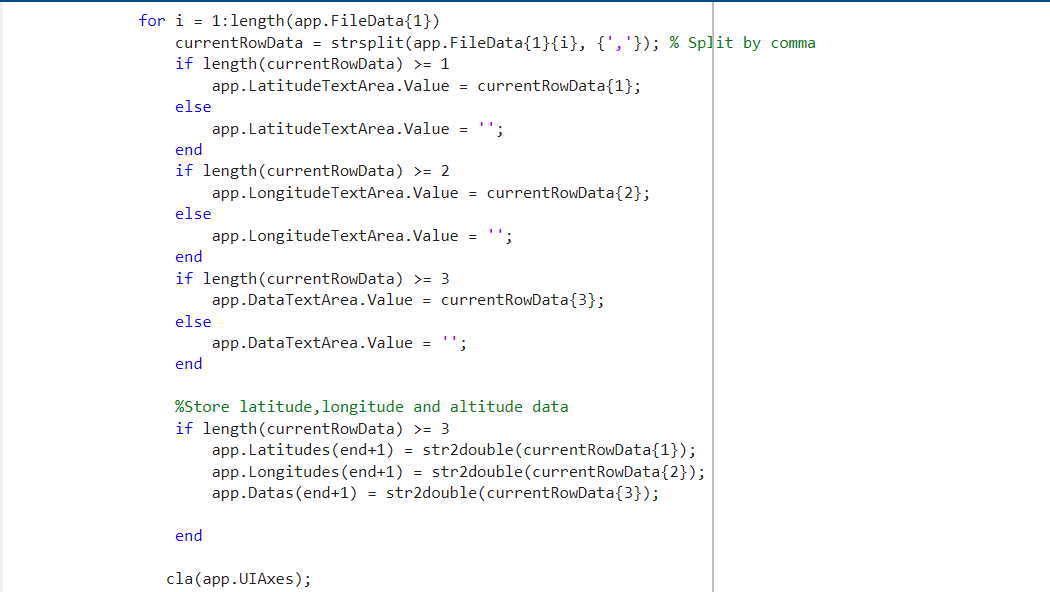


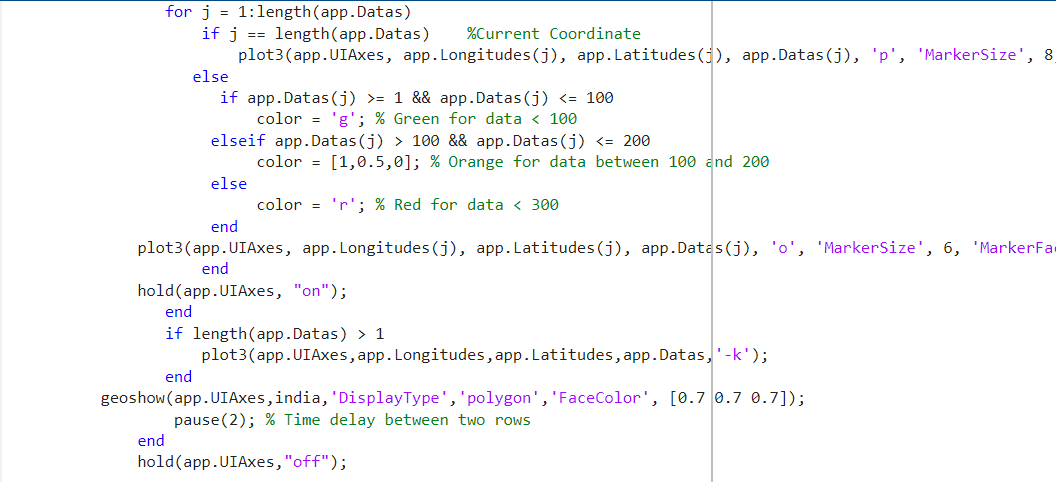
***Step 4:***

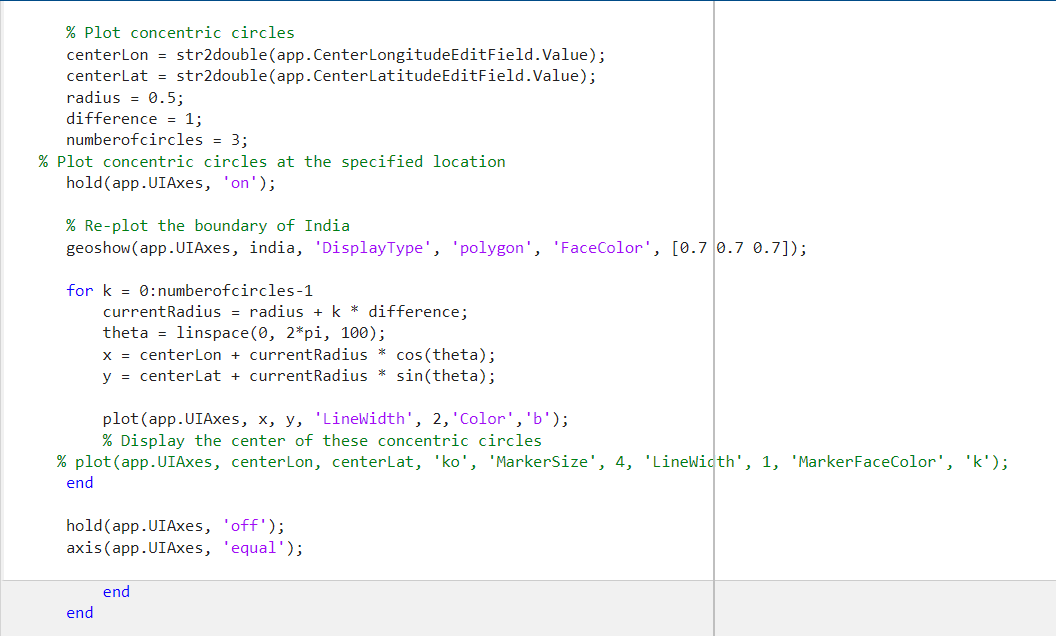
Write the Callback Functions:











***Step 5:***

Run the App:

Save and run the app. We should be able to input the initial radius, difference, and number of circles, plot the circles, and see the count. The app will look like this :

Aerial surveillance data typically consists of spatial information (latitude and longitude) combined with sensor data measurements. Here’s an explanation of the format based on your description:

***Aerial Surveillance Data Format Explanation:***

1. **Latitude and Longitude:**
   * These are geographic coordinates that specify the location where the surveillance data was collected. Latitude measures north-south position relative to the equator, and longitude measures east-west position relative to the Prime Meridian.
2. **Sensor Data:**
   * Sensor data refers to measurements taken by onboard sensors during aerial surveillance. This data could include various parameters such as temperature, humidity, radiation levels, air quality, or any other relevant environmental or situational metrics.

***Threshold Values and Color Coding:***

Threshold values are predefined limits used to categorize sensor data into different ranges. Based on your description:

* Data values between **1 and 100** are represented by **green** points on the map.
* Data values between **100 and 200** are represented by **orange** points.
* Data values between **200 and 300** are represented by **red** points.

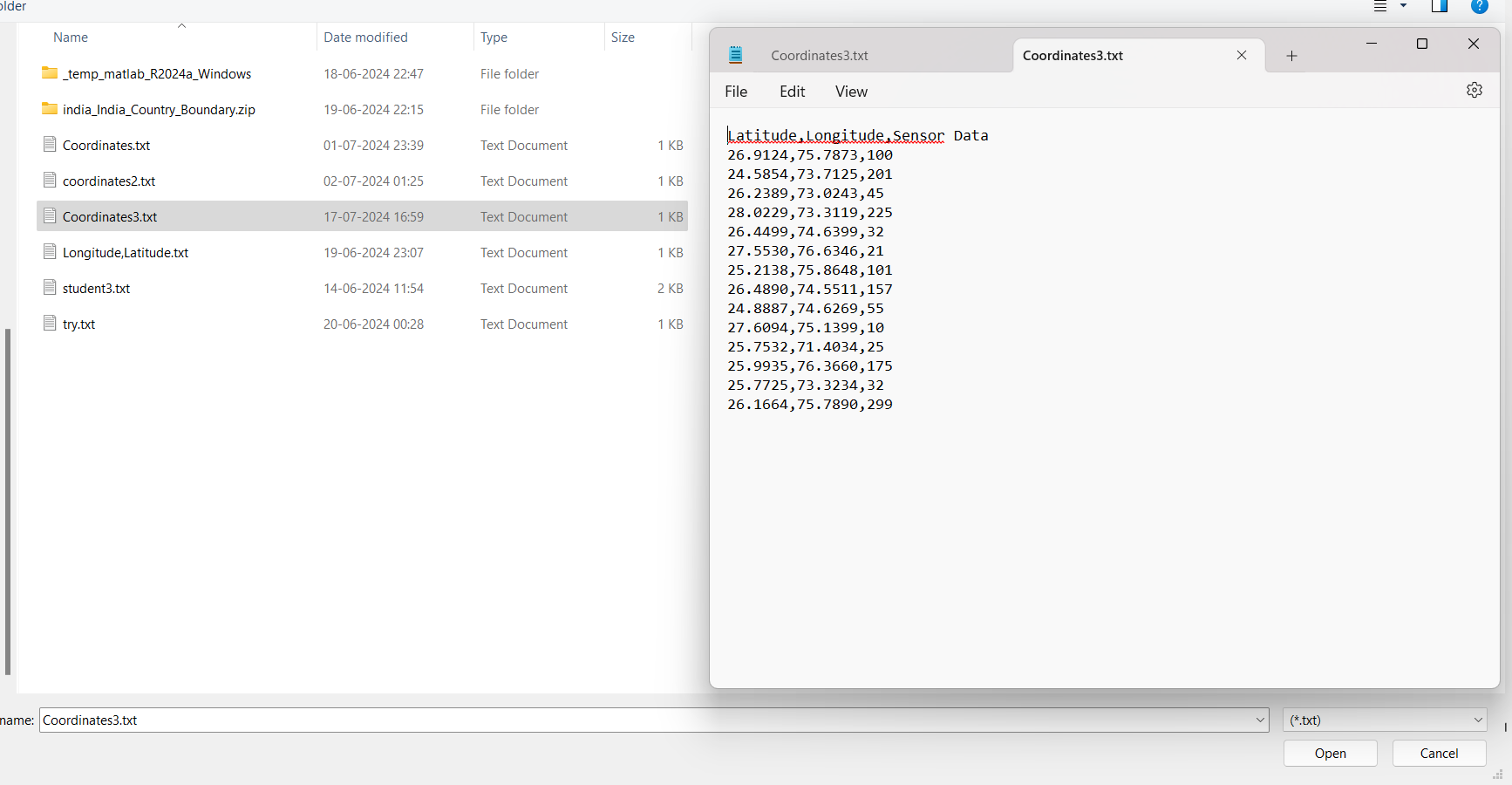
***Visual Representation on a Map:***

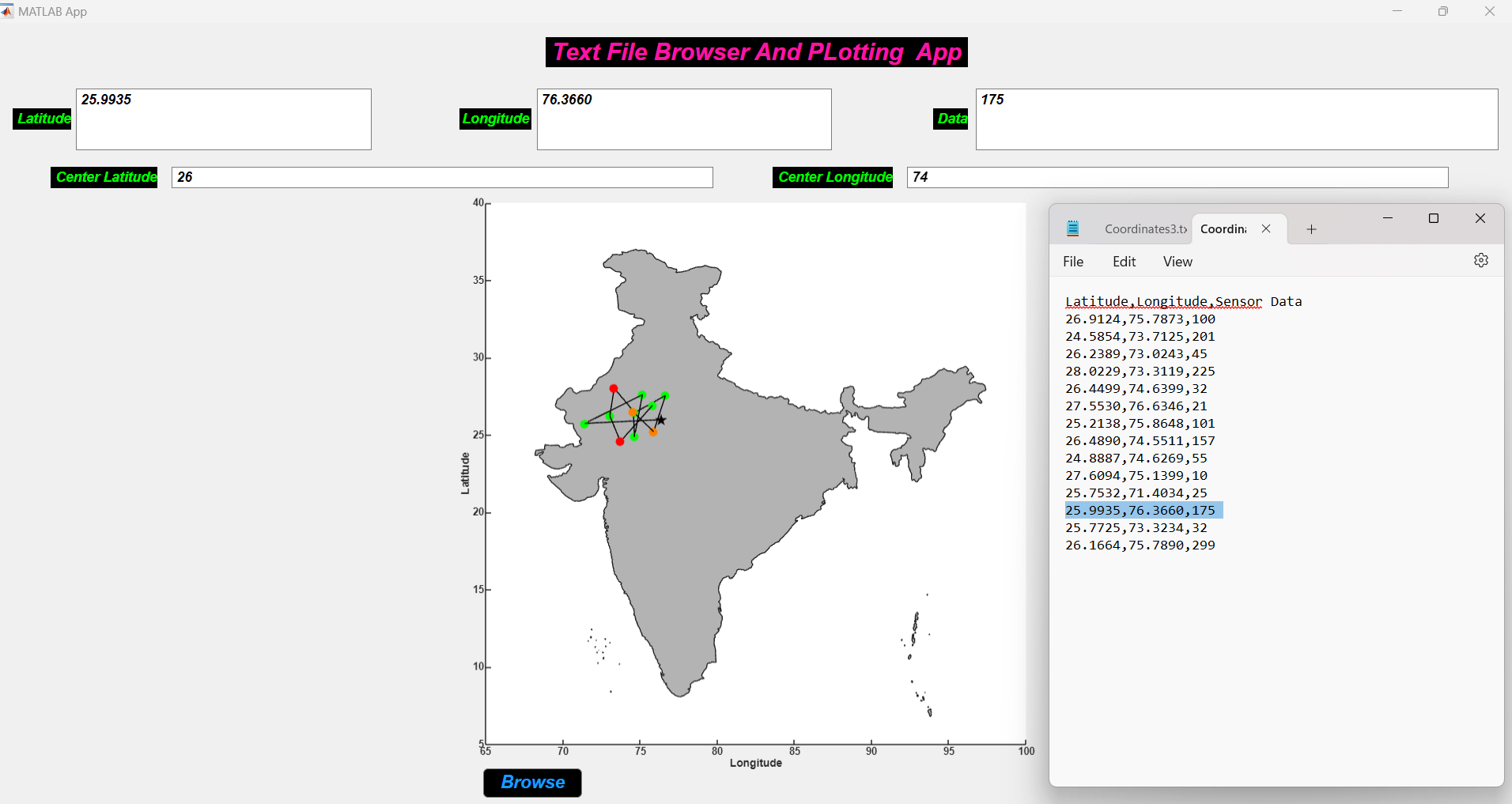
* **Green Points**: Represent areas with sensor values between 1 and 100. These could indicate normal or safe conditions.
* **Orange Points**: Represent areas with sensor values between 100 and 200. These might indicate moderate caution levels or thresholds nearing limits.
* **Red Points**: Represent areas with sensor values between 200 and 300. These could indicate critical or potentially hazardous conditions.

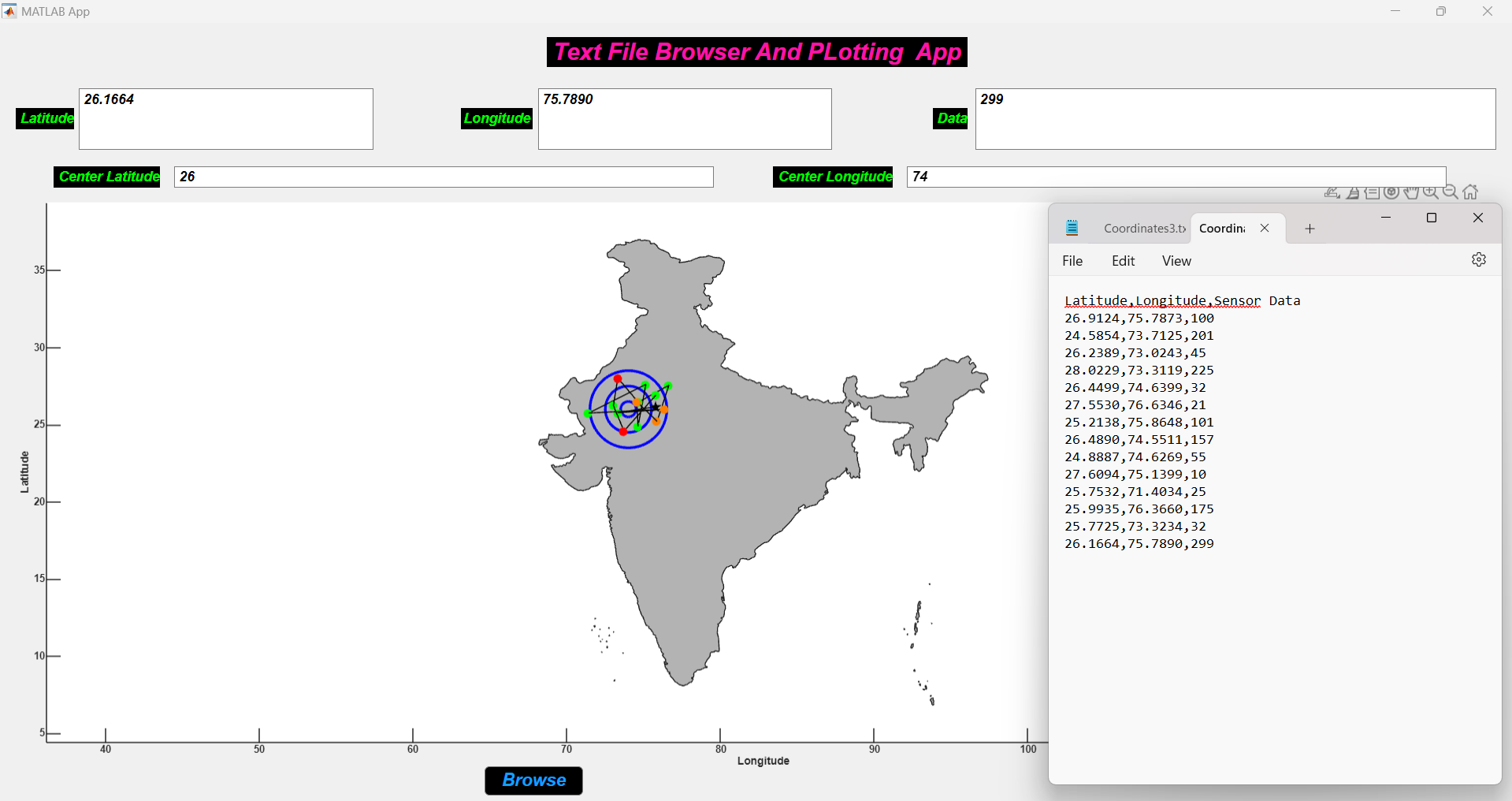
***Implementation in Mapping Tools:***

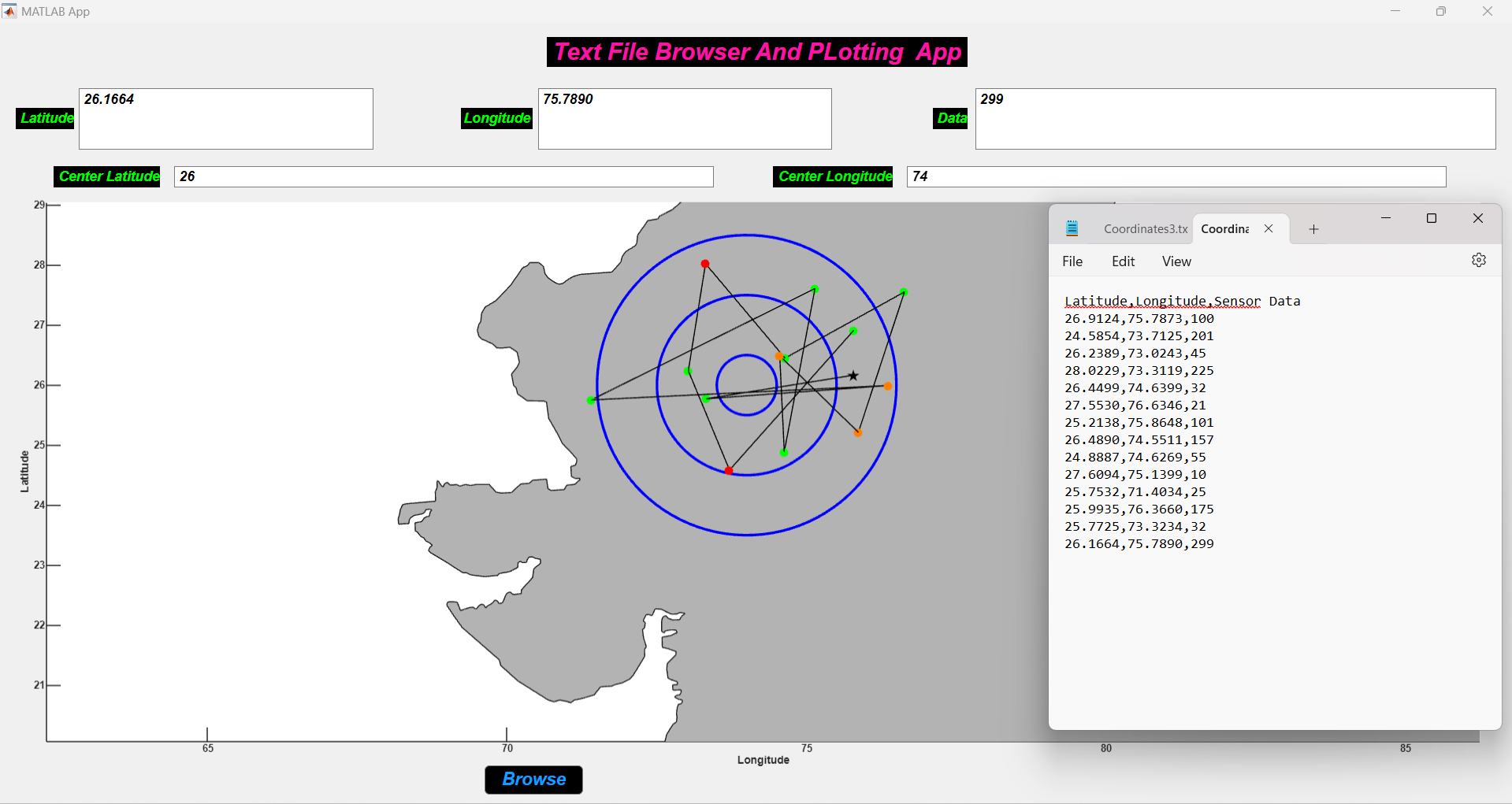
To implement this visualization in mapping tools like MATLAB, we would:

* Import your dataset with latitude, longitude, and sensor data values.
* Plot each point on a map, assigning colors (green, orange, red) based on the respective sensor value ranges.
* Customize markers or symbols to distinguish between different thresholds for clear visual differentiation.
* Optionally, add additional map layers (e.g., streets, satellite imagery) to provide context to the surveillance data.









***Conclusion***

This project aimed to visualize aerial surveillance data on a geographical map using MATLAB, leveraging its Mapping Toolbox for spatial analysis and visualization. The objectives were successfully achieved through the following key accomplishments:

1. **Data Import and Preparation:**
   * Efficiently imported and processed aerial surveillance data, ensuring compatibility with MATLAB's mapping functionalities.
2. **Mapping and Visualization:**
   * Implemented visual representations of surveillance data points on a map, utilizing markers with customizable attributes (e.g., size, color) to reflect different surveillance parameters.
3. **Customization and Interactivity:**
   * Enhanced user interaction through features like zooming, panning, and data point labeling, facilitating detailed exploration of surveillance patterns.
4. **Analysis and Insights:**
   * Utilized MATLAB’s spatial analysis capabilities to derive actionable insights from the visualized data, including identification of spatial trends and hotspot areas of surveillance activity.
5. **Documentation and Reporting:**
   * Documented the entire process, including methodologies, challenges, and solutions, ensuring transparency and reproducibility of results.

***Findings and Contributions***

Through this project, significant findings and contributions were made:

* **Spatial Patterns:** The visualization highlighted distinct spatial patterns in surveillance activities, contributing to a deeper understanding of surveillance coverage and distribution.
* **Decision Support:** The insights derived from the visualized data can support informed decision-making processes related to surveillance planning and resource allocation.
* **Technical Skills:** Enhanced proficiency in MATLAB's Mapping Toolbox, data handling, and spatial analysis techniques, applicable across various fields requiring spatial data visualization.

***Future Directions***

While this project achieved its primary objectives, several avenues for future exploration and improvement remain:

* **Integration with Real-Time Data:** Incorporating real-time data feeds for dynamic surveillance monitoring.
* **Advanced Spatial Analysis:** Exploring more sophisticated spatial analysis techniques, such as predictive modeling or clustering algorithms, to extract deeper insights from surveillance data.
* **User Interface Enhancements:** Enhancing the user interface for improved usability and accessibility, potentially developing a web-based application for broader deployment.

*Thank You !!!*