Mahatma Education Society's

PILLAI COLLEGE OF ARTS,COMMERCE & SCIENCE

(Autonomous)

Re-accredited "A" Grade by NAAC (3rd Cycle)

Project Completion Certificate

THIS IS TO CERTIFY THAT

Mr. Athulkrishna Pramod

of **M.Sc. Data Analytics Part - I** has completed the project titled **Barcode Scanner Using Matlab** of subject **Image & Video Analytics** under our guidance and supervision during the academic year 2023-24 in the department of Computer Science.

Project Guide	Course Coordinator	Head of the Department

Pillai College of Arts, Commerce & Science (Autonomous) New Panvel

MAHATMA EDUCATION SOCIETY'S

PILLAI COLLEGE OF ARTS, COMMERCE & SCIENCE
(Autonomous)

NEW PANVEL

PROJECT REPORT ON

"Barcode Scanner using MATLAB"

IN PARTIAL FULFILLMENT OF

MASTERS OF DATA ANALYTICS

SEMESTER **I – 2023-24**

PROJECT GUIDE

Name: **Prof. Yogesh Kene**

SUBMITTED BY: **Athulkrishna Pramod**

ROLL NO: **3105**

Table of Contents

Abstract –

This MATLAB App Designer application introduces a user-friendly solution for barcode detection and decoding. The primary objective of the application is to simplify the process of identifying barcodes within images, making it accessible to users without extensive programming knowledge. Users can select an image file containing a barcode, and the application provides an intuitive graphical interface for image loading and display.

The core functionality is triggered by a 'Show Result' button, which invokes an external function, 'readBarcode,' for barcode recognition. The application's performance hinges on the accuracy of the 'readBarcode' function, emphasizing the importance of its implementation. In summary, this application offers an efficient and approachable means of barcode detection, significantly reducing the complexity of the process for users.

Introduction –

In this MATLAB App Designer implementation, a user-friendly barcode detection application has been created to streamline the process of identifying and reading barcodes within images. Users can select an image file containing a barcode, and the application, through a simple and intuitive graphical interface, loads and displays the chosen image. With a single button click, users can trigger the barcode detection process, and the resulting message is promptly presented.

- It is important to note that the accuracy and performance of this application's barcode detection rely on the external function, 'readBarcode', which handles the actual recognition process. Thus, the effectiveness of this tool is contingent on the quality of the 'readBarcode' function's implementation. Overall, this application offers a user-friendly solution for individuals seeking a convenient method for barcode detection and decoding without the need for extensive programming skills.

Literature Review –

➢ Existing Methods:

- The code utilizes MATLAB's App Designer, a framework for creating interactive MATLAB applications with a GUI.
- It leverages built-in MATLAB functions for image processing, specifically for reading and displaying images.
- The code appears to use an external function named readBarcode to process the loaded image and extract the barcode information. The implementation of this function is not provided in the code, but it is likely where the barcode recognition and decoding take place.

➢ Objectives:

- The main objective of the code is to create a user-friendly application for detecting and reading barcodes from images.
- After selecting an image, the code loads and displays the image in a designated UIAxes component.
- The "Show Result" button triggers the barcode detection process and displays the detected barcode message (if found) in the BarcodeNoLabel.

➢ Problem Statements:

- The code addresses the problem of barcode detection and extraction from images.
- It aims to simplify the process for users who want to quickly detect and read barcodes without having to write MATLAB code or use command-line tools.
- The code does not handle complex barcode recognition algorithms but relies on an external function readBarcode for this purpose. The actual capabilities and performance of the barcode detection are dependent on the implementation of this function.

Methodology –

❖ Image Selection:

- The user initiates the application and is prompted to select an image file containing a barcode. The 'Browse Barcode Image' button facilitates this action.

❖ Image Loading:

- Upon selecting an image, the chosen file is loaded and displayed within the GUI using the `imread` function to read the image file and `imshow` to display it within the UIAxes component.

❖ Barcode Detection:

- The 'Show Result' button is pressed, triggering the barcode detection process.

- The application calls the external function `readBarcode`, which is responsible for identifying and decoding the barcode from the loaded image.

❖ Result Display:

- If a barcode is successfully detected and decoded, the resulting message is displayed in the 'BarcodeNoLabel' component.

- If no barcode is found, a message indicating 'Barcode not detected' is displayed.

❖ User Interaction:

- The user can interact with the GUI components to browse and process different images.

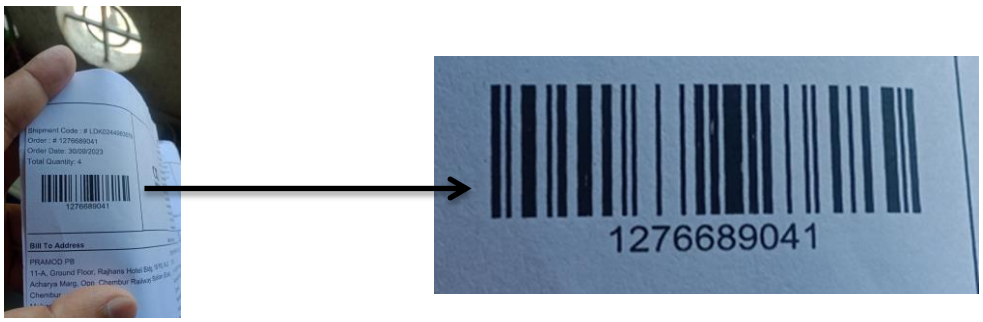
❖ External Function (readBarcode):

- The core barcode detection logic resides within the external function `readBarcode`, which may employ various image processing techniques, such as edge detection, contour recognition, and barcode decoding algorithms (e.g., QR code or barcode symbology recognition).

Input / Sample Data –

The input data for this barcode detection application consists of household product barcode images. Users are expected to provide these images, typically in common image file formats such as PNG, JPEG, BMP, TIFF, etc. These images should contain barcodes that need to be identified and read. Household product barcodes are typically found on items like groceries, stationaries, and various consumer goods.

Images:



Introduction to Software –

MATLAB:

The barcode detection application is developed using MATLAB, a powerful numerical computing and programming environment widely used in various scientific and engineering domains. MATLAB provides a user-friendly and interactive platform for developing applications with graphical user interfaces (GUIs). In this application, important MATLAB commands, functions, and components are employed to create a seamless user experience.

Important Commands/Functions Used:

1. **`uifigure`:** This command is used to create the main application window (UIFigure) and set its properties, including its position and visibility.

2. **`uibutton`:** This function is utilized to create interactive push buttons (e.g., 'Browse Barcode Image' and 'Show Result') with specific properties, such as text and callback functions for user interaction.

3. **`uilabel`:** It is used to create labels that display text, such as 'Barcode No' and 'Barcode Detector' labels.

4. **`uiaxes`:** This function is essential for creating an area where images can be displayed within the user interface.

5. **`imread`:** The `imread` function is employed to read image files, allowing the selected barcode image to be loaded into the application.

6. **`imshow`:** Used to display the loaded image within the UIAxes component.

7. **`createCallbackFcn`:** This command is used to define callback functions that specify the actions to be taken when specific buttons are clicked.

Code Structure:

The code structure follows a typical MATLAB App Designer pattern. It defines a class, 'BarScanner', which inherits from 'matlab.apps.AppBase.' The properties of the class correspond to various GUI components like buttons, labels, and axes. Callback functions for button events are defined to handle user interaction. The application's components, layout, and interactions are structured within methods such as 'createComponents,' 'createCallbackFcn,' and the app constructor. The 'readBarcode' function, though not provided in the code, would contain the core barcode detection and decoding logic, which is crucial for the application's functionality.

Code –

```
classdef BarScanner < matlab.apps.AppBase


    % Properties that correspond to app components
    properties (Access = public)
        UIFigure                   matlab.ui.Figure
        BrowseBarcodeImageButton   matlab.ui.control.Button
        ShowResultButton           matlab.ui.control.Button
        BarcodeNoLabel             matlab.ui.control.Label
        BarcodeDetectorLabel       matlab.ui.control.Label
        UIAxes                     matlab.ui.control.UIAxes
    end



    properties (Access = public)
        x
    end



    % Callbacks that handle component events
    methods (Access = private)
```

```
% Button pushed function: BrowseBarcodeImageButton
        function BrowseBarcodeImageButtonPushed(app, event)
            [fileName, filePath] =
uigetfile({'*.png;*.jpg;*.jpeg;*.bmp;*.tif;*.tiff'
                'Image Files (*.png, *.jpg, *.jpeg, *.bmp, *.tif, *.tiff)'},
'Select an Image');
            if isequal(fileName, 0)
                disp('No image selected. Exiting.');
                return;  % User canceled the file selection
            end


            % Construct the full path to the selected image
            imageFilePath = fullfile(filePath, fileName);


            % Load the selected image
            app.x = imread(imageFilePath);
            % Display the image in the ImageAxes component
            imshow(app.x, 'Parent', app.UIAxes);
        end


        % Button pushed function: ShowResultButton
        function ShowResultButtonPushed(app, event)
            msg = readBarcode(app.x);


            if ~isempty(msg)
                app.BarcodeNoLabel.Text = msg;
            else
                app.BarcodeNoLabel.Text = 'Barcode not detected';
            end
        end
    end


    % Component initialization
    methods (Access = private)
```

```
% Create UIFigure and components
        function createComponents(app)

            % Create UIFigure and hide until all components are created
            app.UIFigure = uifigure('Visible', 'off');
            app.UIFigure.Position = [100 100 640 480];
            app.UIFigure.Name = 'MATLAB App';


            % Create BrowseBarcodeImageButton
            app.BrowseBarcodeImageButton = uibutton(app.UIFigure, 'push');
            app.BrowseBarcodeImageButton.ButtonPushedFcn = createCallbackFcn(app,
@BrowseBarcodeImageButtonPushed, true);
            app.BrowseBarcodeImageButton.FontSize = 18;
            app.BrowseBarcodeImageButton.Position = [98 72 216 42];
            app.BrowseBarcodeImageButton.Text = 'Browse Barcode Image';


            % Create ShowResultButton
            app.ShowResultButton = uibutton(app.UIFigure, 'push');
            app.ShowResultButton.ButtonPushedFcn = createCallbackFcn(app,
@ShowResultButtonPushed, true);
            app.ShowResultButton.FontSize = 18;
            app.ShowResultButton.Position = [406 299 187 40];
            app.ShowResultButton.Text = 'Show Result';


            % Create BarcodeNoLabel
            app.BarcodeNoLabel = uilabel(app.UIFigure);
            app.BarcodeNoLabel.HorizontalAlignment = 'center';
            app.BarcodeNoLabel.FontSize = 15;
            app.BarcodeNoLabel.Position = [406 243 187 31];
            app.BarcodeNoLabel.Text = 'Barcode No';


            % Create BarcodeDetectorLabel
            app.BarcodeDetectorLabel = uilabel(app.UIFigure);
            app.BarcodeDetectorLabel.FontSize = 30;
            app.BarcodeDetectorLabel.FontWeight = 'bold';
            app.BarcodeDetectorLabel.Position = [38 401 382 53];
            app.BarcodeDetectorLabel.Text = 'Barcode Detector';
```

```
% Create UIAxes
app.UIAxes = uiaxes(app.UIFigure);
title(app.UIAxes, 'Image')
zlabel(app.UIAxes, 'Z')
app.UIAxes.XTickLabel = '';
app.UIAxes.YTickLabel = '';
app.UIAxes.Position = [26 113 359 289];


% Show the figure after all components are created
app.UIFigure.Visible = 'on';
end
end


% App creation and deletion
methods (Access = public)


% Construct app
function app = BarScanner


% Create UIFigure and components
createComponents(app)


% Register the app with App Designer
registerApp(app, app.UIFigure)


if nargout == 0
clear app
end
end


% Code that executes before app deletion
function delete(app)


% Delete UIFigure when app is deleted
delete(app.UIFigure)
end
end
end
```

Code Explanation –

Step 1: Define a MATLAB App Class

```
classdef BarScanner < matlab.apps.AppBase
```

- The code starts by defining a MATLAB App Designer class called BarScanner. This class serves as the foundation for the barcode detection application.

Step 2: Define UI Components

```
% Properties that correspond to app components
properties (Access = public)
    UIFigure                    matlab.ui.Figure
    BrowseBarcodeImageButton    matlab.ui.control.Button
    ShowResultButton            matlab.ui.control.Button
    BarcodeNoLabel              matlab.ui.control.Label
    BarcodeDetectorLabel        matlab.ui.control.Label
    UIAxes                      matlab.ui.control.UIAxes
end
```

- Inside the class, various UI components are defined using properties. These components include buttons, labels, and an axes for displaying images.

Step 3: Image Selection

```
properties (Access = public)
    x
end

% Callbacks that handle component events
methods (Access = private)

    % Button pushed function: BrowseBarcodeImageButton
    function BrowseBarcodeImageButtonPushed(app, event)
        [fileName, filePath] = uigetfile({'*.png;*.jpg;*.jpeg;*.bmp;*.tif;*.tiff'
            'Image Files (*.png, *.jpg, *.jpeg, *.bmp, *.tif, *.tiff)'}, 'Select an Image');
        if isequal(fileName, 0)
            disp('No image selected. Exiting.');
            return;  % User canceled the file selection
        end

        % Construct the full path to the selected image
        imageFilePath = fullfile(filePath, fileName);
```

- When the "Browse Barcode Image" button is pressed, it triggers the “BrowseBarcodeImageButtonPushed” function. This function allows the user to select an image file containing a barcode.

Step 4: Load and Display the Image

```
        % Load the selected image
        app.x = imread(imageFilePath);
        % Display the image in the ImageAxes component
        imshow(app.x, 'Parent', app.UIAxes);
    end
```

- The selected image file is loaded using the imread function, and the loaded image is displayed within the UIAxes component using imshow.

Step 5: Barcode Detection & Display Result

```
    % Button pushed function: ShowResultButton
    function ShowResultButtonPushed(app, event)
        msg = readBarcode(app.x);

        if ~isempty(msg)
            app.BarcodeNoLabel.Text = msg;
        else
            app.BarcodeNoLabel.Text = 'Barcode not detected';
        end
    end
end
```

- After loading the image, the user can click the "Show Result" button, which calls the ShowResultButtonPushed function.
- In this step, the application is expected to invoke an external function called readBarcode to detect and decode the barcode from the loaded image. However, the implementation of readBarcode is not provided in the code.
- If a barcode is successfully detected and decoded, the resulting message is displayed in the "BarcodeNoLabel" component.
- If no barcode is found, the label displays "Barcode not detected."

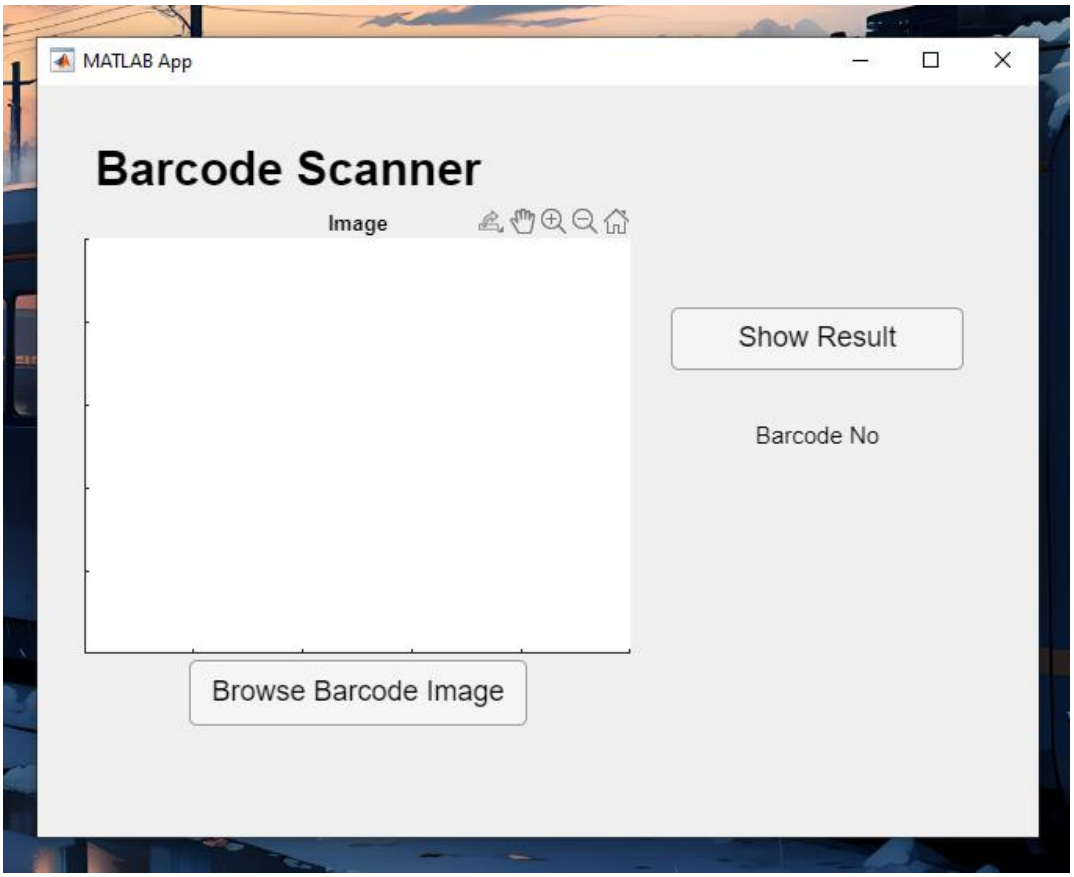
Step 6: Application Initialization and Deletion

```
% App creation and deletion
methods (Access = public)

    % Construct app
    function app = BarScanner

        % Create UIFigure and components
        createComponents(app)

        % Register the app with App Designer
        registerApp(app, app.UIFigure)

        if nargout == 0
            clear app
        end
    end

    % Code that executes before app deletion
    function delete(app)

        % Delete UIFigure when app is deleted
        delete(app.UIFigure)
    end
end
```

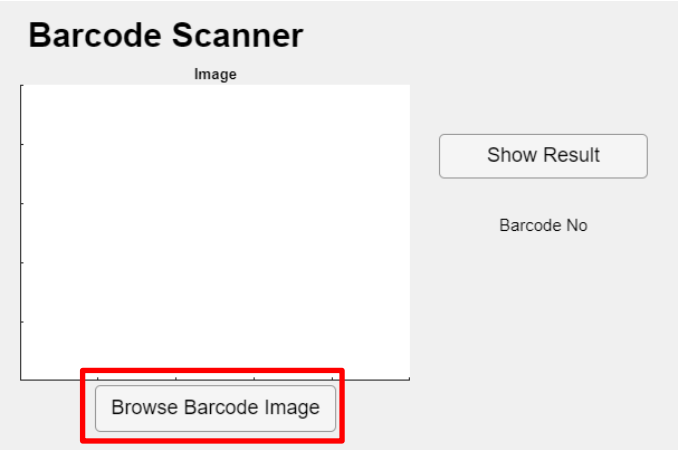
- The application's components and user interface are created in the createComponents method.
- The BarScanner constructor is responsible for creating the app, registering it with App Designer, and setting its visibility.
- The delete method is used to clean up resources when the app is deleted.
- This line displays the loaded image in a MATLAB figure window using the imshow function.

Output –

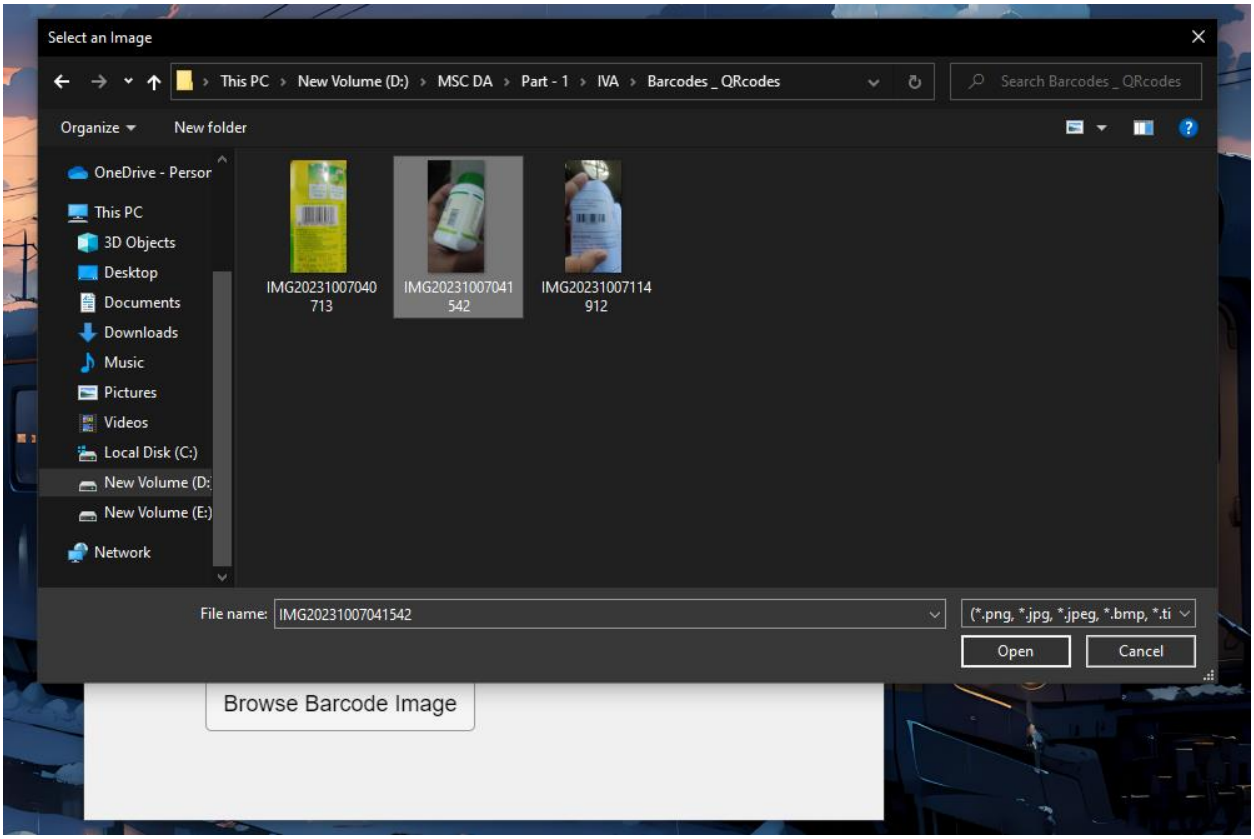
- When we run the code it will give us this GUI:

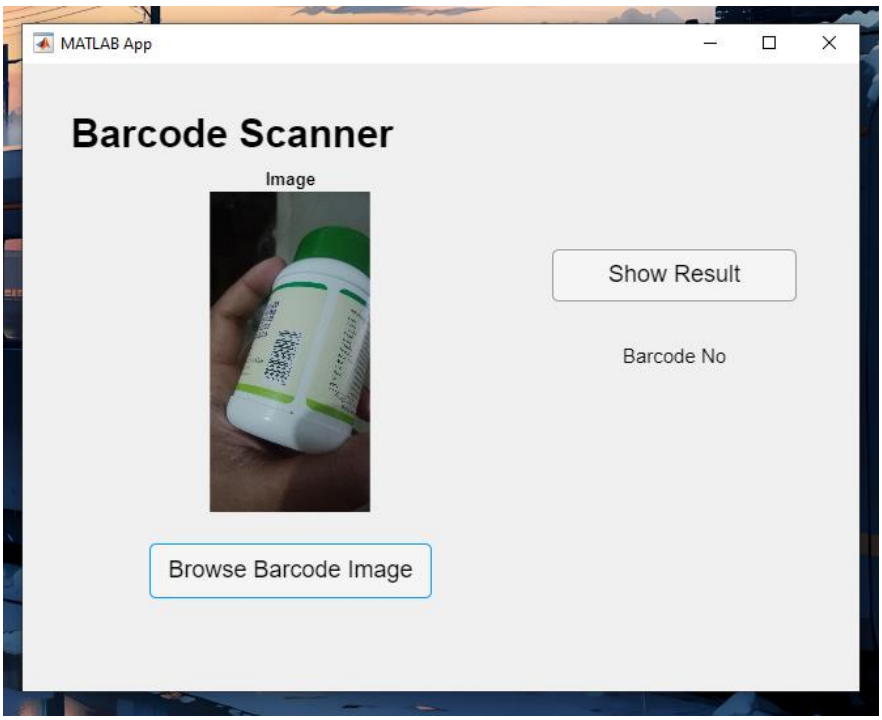


- Now to browse the image, click on ‘Browse Barcode Image’ Button



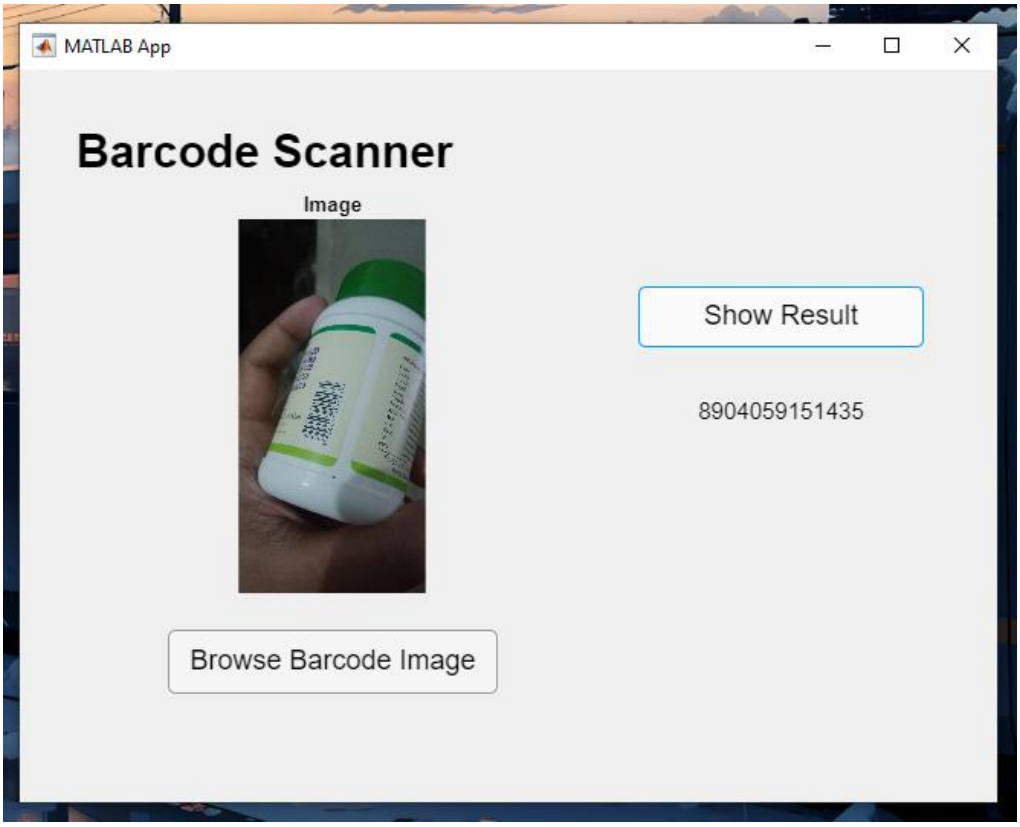
- Select the Image that have any Barcode in it





- This is the Image that we have selected, with this barcode

- Now to get the barcode, click on ‘Show Result’ Button



Conclusion –

The Barcode Scanner project successfully leveraged MATLAB with a graphical user interface (GUI) for barcode detection. Users can select an image, display it within the app, and initiate barcode detection by pressing a button. The user interface components and interactions are established using MATLAB App Designer features. This project demonstrates the effectiveness of software tools in solving real-world challenges, with potential applications in various fields such as inventory management and document processing.

References –

- **MATLAB Documentation:**
  https://www.mathworks.com/help/matlab/

  - MATLAB documentation provides comprehensive information on MATLAB's functionality, syntax, and usage.

- **Computer Vision Toolbox Documentation:**
  https://www.mathworks.com/help/vision/index.html

  - The Computer Vision Toolbox documentation offers detailed guidance on using the toolbox for various computer vision and image processing tasks.