### Step-by-Step Execution Guide

**Step 1: Set Up the Preprocessing Folder**  
Start by organizing your image preprocessing scripts inside the Preprocessing folder. This folder should contain:

* color\_filtering.py to isolate specific colors (like red for tomatoes),
* flipping.py for horizontally/vertically flipping images to enhance data variation,
* normalization.py to scale pixel values (e.g., from 0–255 to 0–1),
* resizing.py to standardize image dimensions (e.g., 64x64), and
* thresholding.py to convert images into binary using threshold techniques.  
  You can run each script individually using the command python Preprocessing/<script\_name>.py. Ensure these scripts are modular and reusable to create a custom preprocessing pipeline.

**Step 2: Prepare the Templates Folder**  
The Templates folder should include home.html, which is the user-facing interface of your web application. This HTML file allows users to upload an image and displays the prediction result. Make sure this file includes a reference to your CSS for proper styling using:

<link rel="stylesheet" href="{{ url\_for('static', filename='style.css') }}">

**Step 3: Configure the Static Folder**  
Inside the Static folder, place your style.css file to style the home.html page. This file can control background colors, font families, button designs, and overall layout. Example:

body {

background-color: #f5f5f5;

font-family: Arial, sans-serif;

}

**Step 4: Train Your Models in the Training Folder**  
In the Training folder, create two training scripts: train\_cnn.py for a Convolutional Neural Network (CNN), and random\_forest.py for a classical ML approach. These scripts should load the preprocessed images, train the models, and save them to files—typically .h5 for CNN and .pkl for Random Forest. Execute training using:

python Training/train\_cnn.py

**Step 5: Build the Web Interface with app.py**  
The app.py file is a Flask web application that integrates your trained CNN model into a website. This script should:

* Load the saved CNN model (e.g., model\_cnn.h5),
* Render the home.html form for user input,
* Handle image uploads from users,
* Perform predictions and display results on the page.  
  Run the app with the command:

python app.py

**Step 6: Deploy with main.py on Raspberry Pi**  
The main.py script is used for hardware integration with the Raspberry Pi and the AI-Hat. It should:

* Access the Raspberry Pi camera for real-time image capturing,
* Use the trained CNN model to perform on-the-fly predictions,
* Optionally control GPIO pins for physical actions (e.g., moving a robotic arm).  
  Install necessary packages with:

pip install RPi.GPIO opencv-python flask tensorflow

Then run the integration using:

python3 main.py

**Step 7: Raspberry Pi Setup Instructions**

* Insert the microSD card with Raspberry Pi OS installed.
* Enable the camera module using sudo raspi-config.
* Copy the entire project folder to the Raspberry Pi.
* If needed, re-run preprocessing and training scripts on the Pi.
* Connect your AI-Hat to GPIO pins and confirm pin mappings in main.py.

**Step 8: Final Checklist**

* All preprocessing scripts are working correctly
* The HTML page is styled and functional with CSS.
* Your CNN and/or Random Forest models are successfully trained and saved.
* The Flask web app (app.py) runs and predicts correctly on the browser.
* The main.py script integrates the model with Raspberry Pi and performs real-time inference.