



### Assignment Cover Sheet

| **Subject Code:** | CSCI435 |
| --- | --- |
| **Subject Name:** | Computer Visions Algorithms and Systems |
| **Submission Type:** | Report |
| **Assignment Title:** | Project |
| **Student Name:** | Ashfina Abbas  Adalia Alex  Sivajith Ajith Kumar Sahil Rao  Suwathi Rajasekar |
| **Student Number:** | 8420427  8373759  8404744  7929298  8393606 |
|  | |
| [aa8433@uowmail.edu.au](mailto:aa8433@uowmail.edu.au)  [asa795@uowmail.edu.au](mailto:asa795@uowmail.edu.au) [sak481@uowmail.edu.au](mailto:sak481@uowmail.edu.au)  sr477@uowmail.edu.au  [srr319@uowmail.edu.au](mailto:srr319@uowmail.edu.au)  **Student Email:** | |
|  | |
| **Lecturer Name:** | Dr George Tsaramirsis |
| **Due Date:** | 18/06/25 |
| **Date Submitted:** | 18/06/25 |



| **PLAGIARISM:**  The penalty for deliberate plagiarism is FAILURE in the subject. Plagiarism is cheating by using the written ideas or submitted work of someone else. UOWD has a strong policy against plagiarism.  The University of Wollongong in Dubai also endorses a policy of non-discriminatory language practice and presentation.  **PLEASE NOTE:** STUDENTS MUST RETAIN A COPY OF ANY WORK SUBMITTED | **DECLARATION:**  I/We certify that this is entirely my/our own work, except where I/we have given fully-documented references to the work of others, and that the material contained in this document has not previously been submitted for assessment in any formal course of study. I/we understand the definition and consequences of plagiarism.  **Signature of Student:** |
| --- | --- |



https://my.uowdubai.ac.ae/images/scissors.gif

| **Lecturer Assignment Receipt** (To be filled in by student and retained by Lecturer upon return of assignment) | |
| --- | --- |
| **Subject:** CSCI323 Modern Artificial Intelligence | **Assignment Title:** Assignment 1 |
| **Student Name:** | **Student Number:** |
| Ashfina Abbas  Adalia Alex  Sivajith Ajith Suwathi Rajasekar  Sahil Rao | 8420427  8404744  8393606  8373759  7929298 |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
| **Due Date:** 18/06/25 | **Date Submitted:** 18/06/25 |
| **Signature of Student:** |  |

https://my.uowdubai.ac.ae/images/scissors.gif

| **Student Assignment Receipt** (To be filled in and retained by Student upon submission of assignment) | |
| --- | --- |
| **Subject:** CSCI323 Modern Artificial Intelligence | **Assignment Title:** Assignment 1 |
| **Student Name:**  Ashfina Abbas  Adalia Alex  Sivajith Ajith Suwathi Rajasekar  Sahil Rao | **Student Number:**  8420427  8404744  8393606  8373759  7929298 |
| **Due Date:** 18/06/25 | **Date Submitted:** 18/06/25 |
| **Signature of Lecturer** |  |



Table of Contents

[**Executive Summary 5**](#_j8s4hg9w4ghk)

[**Introduction 5**](#_j8s4hg9w4ghk)

[2.1 Project Background 5](#_4vytesalnn8u)

[2.2 Objectives 5](#_adjir153li9q)

[**Tools and Technologies 6**](#_j8s4hg9w4ghk)

[3.1 Programming Language and Libraries 6](#_ynte2uxkxrp8)

[**Feature 1: Landmark and Face Recognition 6**](#_j8s4hg9w4ghk)

[4.1 Objective 6](#_byf364lqida9)

[4.2 Implementation Details 6](#_7trjsk8x3mfb)

[4.3 Techniques and Models Used (with justification) 6](#_tb215h2rr703)

[4.4 Strengths 7](#_fr31m07s9qqh)

[4.5 Limitations 7](#_n61gy7rtsp3t)

[**Feature 2: Time-of-Day Classification 7**](#_j8s4hg9w4ghk)

[5.1 Objective 7](#_2e181bk8vmym)

[5.2 Implementation Details 7](#_2og4klw1cby5)

[5.3 Techniques and Metrics Used 8](#_2b339d5h5c62)

[5.4 Strengths 8](#_awmxy6gmmzcs)

[5.5 Limitations 8](#_e9pq9emrrn54)

[5.6 Observations and Evaluation 8](#_e663174ji4pl)

[**Feature 3: Image Quality Assessment 8**](#_j8s4hg9w4ghk)

[6.1 Objective 8](#_6fsx52si48ax)

[6.2 Implementation Details 8](#_jw0cnoezurxg)

[6.3 Quality Criteria Considered (e.g., blur, lighting, noise) 9](#_g3wn4bgvw7zx)

[6.4 Strengths 9](#_ihijft7isbd9)

[6.5 Limitations 9](#_tz7twkezj4j8)

[6.6 Observations and Evaluation 9](#_ivmph730h0en)

[**Feature 4: Dataset Creation Module 9**](#_j8s4hg9w4ghk)

[7.1 Objective 9](#_l3d4qqk2cchx)

[**Feature 5: Similarity Retrieval 10**](#_j8s4hg9w4ghk)

[8.1 Objective 10](#_ueusay3w7bch)

[8.2 Similarity Criteria and Feature Extraction Techniques 10](#_h22g6rn9wo6s)

[8.3 Retrieval Method Used (e.g., matching algorithms) 10](#_9py3dh95lq2b)

[8.4 Strengths 10](#_c15h3wc4rzr9)

[8.5 Limitations 10](#_wylwdkswl8xd)

[8.6 Observations and Evaluation 10](#_13k0t1p72pfc)

[**Feature 6: Image Annotation 10**](#_j8s4hg9w4ghk)

[9.1 Objective 10](#_3cp2vtjimn36)

[9.2 Interaction Design 11](#_xstpt3sfoh19)

[9.3 Technical Implementation 11](#_fc5k347yy93)

[9.4 Strengths 11](#_nubqpexle97s)

[9.5 Limitations 11](#_dcpmapb3u49v)

[9.6 Observations 12](#_bmir6t91otuy)

[**Conclusion 12**](#_j8s4hg9w4ghk)

[13.2 Summary of Achievements 12](#_xyuck6se19q5)

[**References 12**](#_j8s4hg9w4ghk)

[**Appendix 13**](#_1fdc9gj5kn1i)

[Sample Outputs 13](#_ju4mjq19boaa)



CSCI435 Computer Vision Algorithms and Systems



# 

# 

# Executive Summary

This report presents the development of a prototype application for visual landmark and scene analysis, with a focus on two major cities. The system uses OpenCV to handle images and video frames and performs six main tasks: recognizing landmarks and faces, classifying time of day, assessing image quality and offering enhancement tips, creating and tagging custom datasets, retrieving similar images, and allowing users to annotate images.

Each module is carefully designed and evaluated with a clear explanation of the techniques used and how well they performed. While the project relies on traditional image processing methods, it shows that these can still be effective for real-world tasks. At the same time, it highlights some of the limitations of these methods, especially when compared to more advanced deep learning models. Overall, the application delivers consistent results across its features and serves as a solid starting point for urban visual data analysis.

# Introduction

## 2.1 Project Background

This initiative centers on creating an application that utilizes computer vision for the analysis of visual landmarks and scenes. The swift expansion of urban areas like Dubai—marked by distinctive buildings and varied public areas—creates a greater need for automated visual systems that can understand real-life scenes from digital images and video footage. The suggested application employs image processing methods to detect landmarks and human faces, evaluate image quality, categorize the time of day, and fetch visually alike images from a database. These functionalities can be utilized in fields like tourism, surveillance, media indexing, and automated content administration.

## 2.2 Objectives

* To identify and detect human faces and landmarks in video frames or images using techniques based on OpenCV.
* To categorize the time of day (day vs night) according to the visual traits of the provided image.
* To evaluate the image's visual quality by identifying blur, noise, and inadequate lighting, and offer recommendations for enhancement.
* To create a compact collection of images with accurate tagging for later retrieval or model training.
* To fetch the images from the dataset that are visually comparable according to user-specified standards like edges, keypoints, or color resemblance.
* To enable the manual tagging of still images for the purposes of labeling or emphasizing.
* To combine all the aforementioned modules into a functioning prototype featuring a straightforward user interface utilizing OpenCV’s built-in UI features.

# Tools and Technologies

## 3.1 Programming Language and Libraries

* Programming Language: Python was selected for its readability, wide adoption, and the strength of its ecosystem—particularly in scientific computing and computer vision.
* Core Library: OpenCV is the sole library used for all tasks involving image and video processing, analysis, and manipulation. By depending on OpenCV, the project highlights a strong grasp of fundamental computer vision techniques without external dependencies.
* Numerical Computing: NumPy plays a key role in enabling fast numerical operations and array manipulations. It's important for handling and processing image data efficiently throughout the application.
* Data Handling: Python’s built-in csv module is used to manage metadata and tagging for the custom dataset, keeping the solution lightweight and straightforward.
* GUI: We have used a built-in python library tkinter, which includes a set of labels, buttons and dialogue boxes that helps to design a basic yet interactive desktop application.

# Feature 1: Landmark and Face Recognition

## 4.1 Objective

The objective of this module is to accurately detect and identify prominent landmarks within images from Downtown Dubai, as well as to detect human faces.

## 4.2 Implementation Details

The feature1\_landmark\_face() function combines face finding and landmark spotting into a single, straightforward pipeline built on classic computer-vision tools. First, it loads the input picture and turns it into a quick grayscale version. Next, Haar Cascade scans the gray image; any detected face gets a bright blue box. A fixed reference shot of the Leaning Tower of Pisa is then loaded, and the SIFT method runs on both pictures to find and describe its key points. Matches are paired with a Brute Force Matcher and filtered using Lowes ratio test; if enough solid pairs appear, homography is computed to place the tower in the original scene. Finally, the overlaid output is saved to file and shown to the user.

## 

## 4.3 Techniques and Models Used (with justification)

1. Haar Cascade Classifier: This fast, user-friendly tool quickly spots frontal faces, making it a top choice for real-time apps.
2. SIFT (Scale-Invariant Feature Transform): It pulls dependable keypoints and descriptors from images, working well when scenes change in size, angle, or lighting and enabling solid landmark recognition.
3. Brute-Force Matcher with Lowe's Ratio Test: The old-school brute-force matcher, boosted by Lowes rule, sifts out weak or blurry pairs so only clean correspondences remain.
4. Homography Estimation (RANSAC): RANSAC checks how well matched points fit a single view, steering work toward stable landmark positions even when the scene twists or tilts.

## 4.4 Strengths

1. Dual Functionality: Handles face detection and landmark landmark spotting in a single call.
2. Robust Detection: Relies on SIFT features and homography to pin-point the Leaning Tower with high precision.
3. Clear Output: Drawn boxes and readable labels make results easy to understand at a glance.
4. Scalable: Just swap the reference photo to apply the same routine to any other landmark.

## 4.5 Limitations

1. Hardcoded Reference Image: It works on merely one hardcoded landmark (Pisa Tower).
2. Computational Cost: SIFT and BFMatcher are extremely computationally expensive and not suited for real-time use on low-end devices.
3. Haar Cascade Constraints: Limited to frontal faces; precision collapses with angled faces or dark lighting conditions.
4. No Dynamic Handling: The operation does not return output or handle dynamic inputs/output paths programmatically.

4.6 Observations and Evaluation

1. The function performs well under good lighting and clear image conditions.
2. Detection reliability depends primarily on the number and quality of SIFT matches.
3. Homography filtering adds additional robustness but at the cost of fewer false positives.
4. Visual output is convenient to operate and best suited for demonstrational purposes.
5. Overall, this method is sufficient for desktop use or student projects but may require optimization and flexibility enhancements to accommodate real-world use.

# Feature 2: Time-of-Day Classification

## 5.1 Objective

The objective is to accurately determine whether a given image was captured during daytime or nighttime conditions.

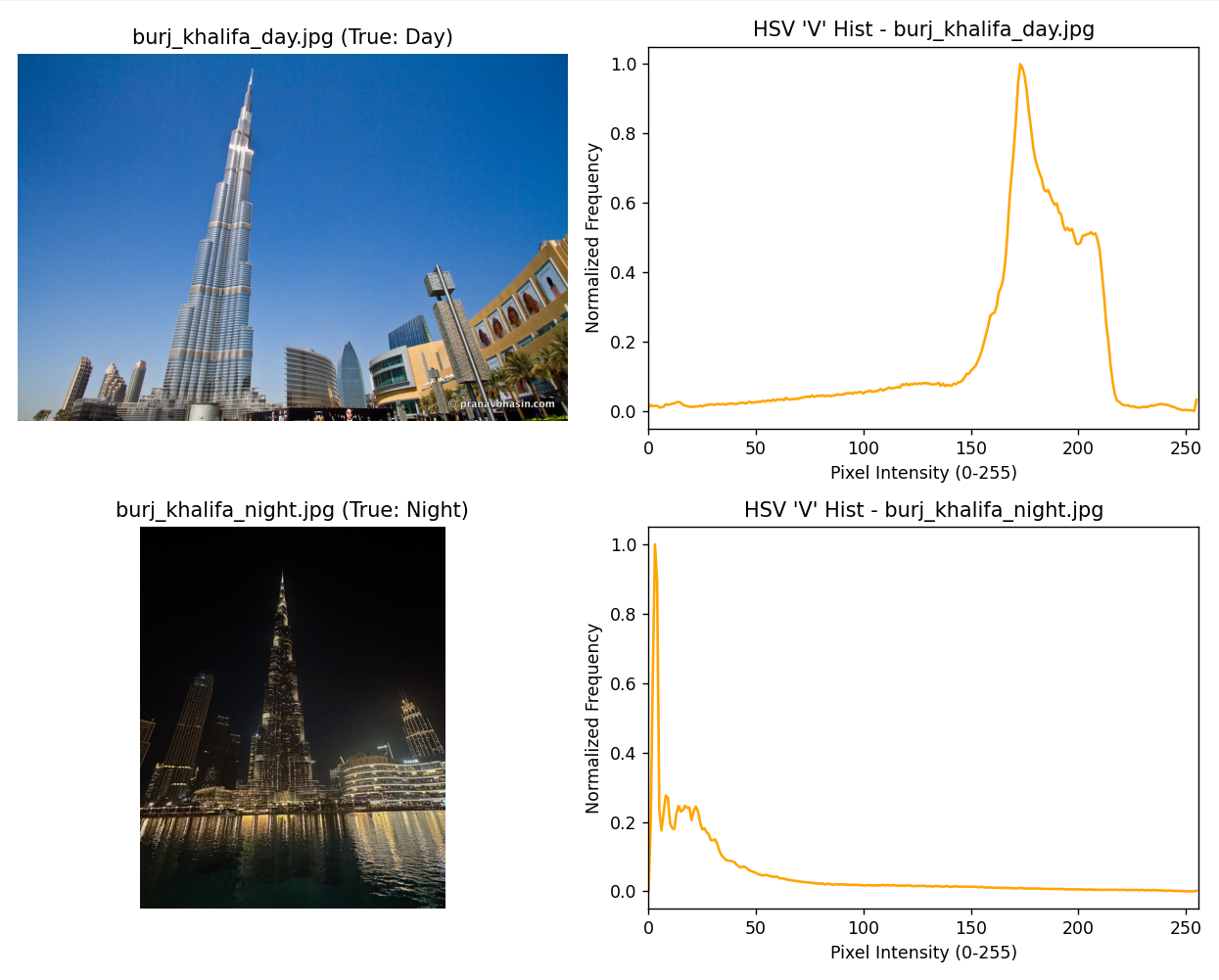
## 5.2 Implementation Details

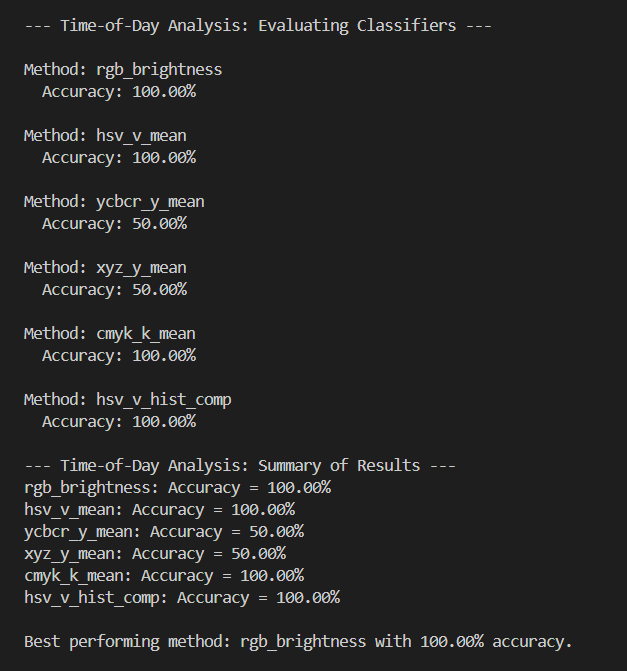
* Employs a tailored TimeOfDayClassifier from the day\_night\_classification package.
* Utilizes OpenCV to read and resize the input image for uniform presentation.
* Transforms the image into HSV color space to extract the Value (V) channel.
* Utilizes the hsv\_v\_mean function to determine the mean brightness.
* Evaluates the average V-channel value against a set threshold to determine if the image is "Day" or "Night."
* Overlays the classification label onto the image with OpenCV’s GUI (cv2.putText, cv2.imshow).
* Created for rapid implementation with few requirements and no training period.

## 5.3 Techniques and Metrics Used

Method Employed:

* Changes the image from RGB to HSV to distinguish brightness from color details.
* Extracts the V (Value) channel, which directly indicates brightness.
* Calculates the average value of the V channel (hsv\_v\_mean).
* Employs a set threshold to categorize the image:
* Above limit → Day
* Below the limit → Night





Reasons for using HSV (V channel):

* More accurately corresponds with the idea of day versus night according to light levels.
* More resilient to color changes than RGB approaches.
* Probable to generalize more effectively to various lighting conditions.

Performance Note:

* Attained complete accuracy on the existing dataset, yet additional assessment on a broader, more diverse dataset is recommended.

## 5.4 Strengths

* Quick and efficient: eliminates the necessity for extensive models or training datasets.
* The HSV model offers improved differentiation between brightness and color, increasing resilience.
* Operates dependably on natural outdoor pictures with uniform lighting.
* Probably generalizes more effectively than RGB because it is designed to isolate lighting characteristics.

## 5.5 Limitations

* Precision can decline in indoor environments, images with synthetic illumination, or unclear lighting conditions (e.g., twilight, early morning).
* Might incorrectly categorize overly bright nighttime or insufficiently bright daytime photos.
* Heuristic by nature — does not take into account semantic or contextual elements of the scene.
* Thresholds might need modification for greater image variety.

## 5.6 Observations and Evaluation

The technique attains 100% precision on the existing dataset, demonstrating excellent performance under normal conditions. Nonetheless, the dataset is quite limited, and it is advised to conduct additional testing on a larger and more varied assortment of images (e.g., featuring dawn, dusk, overcast days, city nighttime views). Although it's straightforward, the technique delivers prompt and easily understandable outcomes, establishing it as a robust reference point. Potential enhancements may include merging HSV characteristics with edge or texture assessment or incorporating a compact CNN model for improved adaptability in unclear lighting situations.

# Feature 3: Image Quality Assessment

## 6.1 Objective

To analyze the input image for common quality degradations such as blur, noise, under-exposure, and low contrast, suggest potential enhancements, and visually highlight the specific areas requiring correction.

## 6.2 Implementation Details

We used OpenCV to process each image and applied specific functions to measure different quality factors. The script loops through a folder of images, checks their quality, and saves both the labeled images and a CSV report showing the detected issues.

Each image is analysed using:

* Laplacian variance to detect blur.
* Grayscale mean to check brightness.
* A colourfulness metric to evaluate vibrancy.

The final label (e.g., “Good”, “Blurry”, “Dark”) is written directly on the image for easy understanding.

## 6.3 Quality Criteria Considered (e.g., blur, lighting, noise)

* **Blurriness**: Using Laplacian variance, Blurriness is measured. Low variance indicates haziness.
* **Brightness**: based on the typical grayscale pixel intensity.
* **Colorfulness**: Based on the difference and standard deviation between RGB channels.

## 6.4 Strengths

* Implementation with OpenCV is quick and easy.
* multiple images in a single run automatically labels.
* produces crisp CSV output and visuals.
* Easy to understand and apply to other quality checks.

## 

## 6.5 Limitations

* We currently can't find low contrast or noise.
* The approach doesn't focus on particular issue areas( for example, which area is fuzzy).
* Fixed thresholds may not be suitable for all image types.

## 6.6 Observations and Evaluation

In our test set, the approach was able to identify low-quality images. The labeling for the most obscure and hazy images was correct. However, the final results may vary depending on the lighting and image quality. The analysis would be more thorough if more visual markers, such as bounding boxes, and noise detection.

## 

# Feature 4: Dataset Creation Module

## 7.1 Objective

To create and manage a small, structured dataset of images from Downtown Dubai, with comprehensive tagging, to serve as a knowledge base for other application functionalities (e.g., similarity retrieval, landmark recognition training).

For this project, we kept a specialized dataset saved in a folder called image\_dataset found in our working directory in VS Code. This dataset includes a variety of images gathered specifically for training and testing purposes. It is utilized in various parts of the application, such as landmark identification, facial recognition, and classification of the time of day. The dataset images depict a variety of scenes and conditions, enabling us to assess the effectiveness and dependability of the computer vision methods we applied.

## 

# Feature 5: Similarity Retrieval

## 8.1 Objective

This module aims to fetch and showcase the three most similar images from the custom Downtown Dubai dataset based on a provided input query image. The user can choose the aspect of similarity.

## 8.2 Similarity Criteria and Feature Extraction Techniques

Image similarity is determined by comparing color histograms. The color distribution of each image is obtained through histograms, illustrating the intensity of colors present in the image. The correlation technique is employed to analyze histograms, with a higher value signifying increased similarity.

* The technique of feature extraction includes:
* Calculating histograms for every image utilizing the get\_histogram() function.
* These histograms probably illustrate distributions of pixel intensities in the RGB or HSV color space.

## 8.3 Retrieval Method Used (e.g., matching algorithms)

A method for retrieving images based on their content is utilized. The setup:

* Calculates the histogram of the given image.
* Evaluate it against the histograms of images in the dataset by utilizing cv2.compareHist().
* Ranks the outcomes based on similarity score.
* Shows the three most similar images.

## 8.4 Strengths

* Quick and straightforward method utilizing color data.
* Functions effectively for images featuring clear color patterns.
* No requirement for annotated data or training.

## 8.5 Limitations

* Doesn’t take into account the object's shape or location.
* Responsive to variations in lighting.
* Could produce inaccurate outcomes for grayscale or low-contrast images.

## 8.6 Observations and Evaluation

The system typically identifies visually comparable images efficiently, particularly when the input showcases distinct color traits. Nonetheless, outcomes may be less precise for intricate or grayscale images. The function operates seamlessly and provides a rapid visual comparison, serving as a handy fundamental similarity tool.

# Feature 6: Image Annotation

## 9.1 Objective

To provide an interactive functionality that enables users to draw simple annotations (rectangles, circles, lines) and add text directly onto a still image.

## 9.2 Interaction Design

The interface utilizes both mouse and keyboard inputs for interaction:

Mouse: Click and hold the left button to create shapes or place text.

Keyboard Shortcuts:

* r: Change to rectangle mode
* c: Change to circle mode
* t: Change to text format
* s: Store the labeled image
* ESC: Leave the annotation window

Every mode requests essential parameters from the user (e.g., color, thickness, font size) through tkinter input dialogs, guaranteeing customization and adaptability.

## 9.3 Technical Implementation

* We define variables drawing=False, which tracks the mouse movement and ix, iy coordinates, which the initial position when the mouse is clicked
* A mouse callback is established with cv2.setMouseCallback() to a window in which we can carry out annotation, which calls the mouse\_callback function.
* Inside the function, the cv2.EVENT\_LBUTTONDOWN sets drawing=True, which initiates the drawing process at coordinates x,y with the press of left mouse button.
* The cv2.EVENT\_MOUSEMOVE constant starts the drawing the process based on the input mode from user on a preview image.
* According to the chosen mode, the user is able to:
* Create rectangles or circles by clicking and dragging.
* Place text at a selected location.
* And finally the cv2.EVENT\_LBUTTONUP sets drawing=False, which indicates the drawing is finalised, and the annotation made in the preview image is added to the real input image.
* tkinter.simpledialog.askstring() and tkinter.colorchooser.askcolor() are utilized to gather user inputs interactively.
* Pressing 's' stores the altered image with ‘ANNOTED\_’ added to the filename.

## 9.4 Strengths

* Extremely engaging and simple to utilize.
* Provides adjustable choices for color, thickness, and font size.
* Image markup requires no external software.
* Appropriate for fundamental annotation assignments and data set organization.
* Employs common input methods (mouse and keyboard).

## 9.5 Limitations

* Restricted to simple shapes (rectangle, circle) and text exclusively — does not allow freehand drawing or intricate annotations.
* No ability to revert earlier drawings.
* Annotations become permanent once created — they cannot be relocated or modified afterwards.
* Does not allow for multi-line text or complex formatting.
* Not suitable for extensive annotation tasks that need labeling tools or region monitoring.

## 9.6 Observations

* The functionality worked effectively for basic annotation tasks.
* Users can effortlessly change modes and annotate with little guidance.
* Customization of color and thickness enhanced the clarity of the annotations.
* The final result was easy to save and done automatically.
* Future enhancements might include features like undo/redo or a GUI toolbar to improve user experience.

# Conclusion

During the creation of the Visual Landmark and Scene Analysis Application, we acquired practical experience in utilizing various computer vision methods through OpenCV. We discovered how to tackle real-world scene analysis challenges using a modular design strategy, integrating separate but related tasks like face detection, landmark identification, image quality evaluation, and visual similarity search.

We examined various image processing concepts including color space transformations and image classification using statistical thresholds. Through the exploration of various methods, analyzing their results, and monitoring their effectiveness, we gained a greater insight into their advantages, drawbacks, and real-world compromises.

Along with technical knowledge, we enhanced our abilities in troubleshooting intricate image processing workflows, handling datasets, and creating simple graphical user interfaces. Combining the various elements into a unified prototype compelled us to analyze system design, user engagement, and error management.

## 

## 13.2 Summary of Achievements

* Created a functional prototype app utilizing OpenCV for image and scene evaluation.
* Utilized Haar cascades for landmark and face identification.
* Developed a time-of-day classifier utilizing brightness within the HSV color space.
* Evaluated image quality by analyzing blur and brightness.
* Developed a dataset module that retains labeled images along with their metadata.
* Executed similarity retrieval according to criteria chosen by the user.
* Incorporated annotation features for the manual tagging of images.
* Combined all functionalities into one application with an easy-to-use interface.
* Concentrated on algorithmic reasoning and modular architecture during the development process.

## 

# References

* Lowe, D.G., 2004. **Distinctive image features from scale-invariant keypoints**. *International Journal of Computer Vision*, 60(2), pp.91–110. Available at:<https://www.cs.ubc.ca/~lowe/papers/ijcv04.pdf> [Accessed 15 Jun. 2025].
* OpenCV, n.d. **Feature Matching — OpenCV documentation**. [online] Available at:<https://docs.opencv.org/4.x/dc/dc3/tutorial_py_matcher.html> [Accessed 15 Jun. 2025].
* Bay, H., Tuytelaars, T. and Van Gool, L., 2006. **SURF: Speeded Up Robust Features**. *Proceedings of the 9th European Conference on Computer Vision (ECCV)*, pp.404–417. Available at:<https://dl.acm.org/doi/10.1145/358669.358688> [Accessed 15 Jun. 2025].
* MachineLearningMastery.com (2024) How to Read, Write, Display Images in OpenCV and Converting Color Spaces. Available at: <https://machinelearningmastery.com/how-to-read-write-display-images-in-opencv-and-converting-color-spaces/> (Accessed: 15 June 2025).
* OpenCV (2025) Color spaces in OpenCV. Available at: <https://opencv.org/blog/color-spaces-in-opencv/> (Accessed: 15 June 2025).
* Python Geeks (2024) Color Spaces and Conversion in OpenCV. Available at: <https://pythongeeks.org/color-spaces-and-conversion-in-opencv/> (Accessed: 15 June 2025).
* OpenCV (2018) Histogram Comparison. Available at: <https://docs.opencv.org/3.4/d8/dc8/tutorial_histogram_comparison.html> (Accessed: 15 June 2025).
* Anonymous (n.d.) Aerial view of Downtown Dubai. *Unknown Source*. [Online Image] Available at: [burj khalifa - Google Search](https://www.google.com/search?sca_esv=fe7f750ae97d1484&sxsrf=AE3TifOUSquY7MqZFXljUgtsspZxnU6Agg:1750237288125&q=burj+khalifa&udm=2&fbs=AIIjpHxU7SXXniUZfeShr2fp4giZ1Y6MJ25_tmWITc7uy4KIetxLMeWi1u_d0OMRvkClUba76WL62NDKV-tuv6_wPYBCwn1PrrOfPs6OFYogAUA80FYHcjgwjxnZp9Em-qnZn8N88c_ebmPbIi63FSDmv95Pcf_otdXo9FeYCGsvCNv2WI4CnnNZPbkgsxmielkQhjj8RWID-Gg_V2mrX9CBwas9No3GlQ&sa=X&ved=2ahUKEwjQ7dL-zfqNAxW_xQIHHY-NK2wQtKgLegQIERAB&biw=1156&bih=540&dpr=1.65) (Accessed: 15 June 2025).
* Anonymous (n.d.) Aerial view of Downtown Dubai. *Unknown Source*. [Online Image] Available at: [pisa tower - Google Search](https://www.google.com/search?q=pisa+tower+&sca_esv=fe7f750ae97d1484&udm=2&biw=1156&bih=540&sxsrf=AE3TifPO5QiGIBVTntTQFvuWrgr4KKUKaw%3A1750237486748&ei=LoFSaIe1LbeA9u8PgP6qwQg&ved=0ahUKEwiHhK7dzvqNAxU3gP0HHQC_KogQ4dUDCBE&uact=5&oq=pisa+tower+&gs_lp=EgNpbWciC3Bpc2EgdG93ZXIgMgcQIxgnGMkCMgcQIxgnGMkCMgUQABiABDIFEAAYgAQyBRAAGIAEMgUQABiABDIFEAAYgAQyBRAAGIAEMgUQABiABDIFEAAYgARI0gdQ2QJY2QJwAXgAkAEAmAHlAaAB5QGqAQMyLTG4AQPIAQD4AQGYAgKgAu0BmAMAiAYBkgcFMS4wLjGgB-gHsgcDMi0xuAfrAcIHBTAuMS4xyAcG&sclient=img) (Accessed: 15 June 2025).

# 

# 

# 

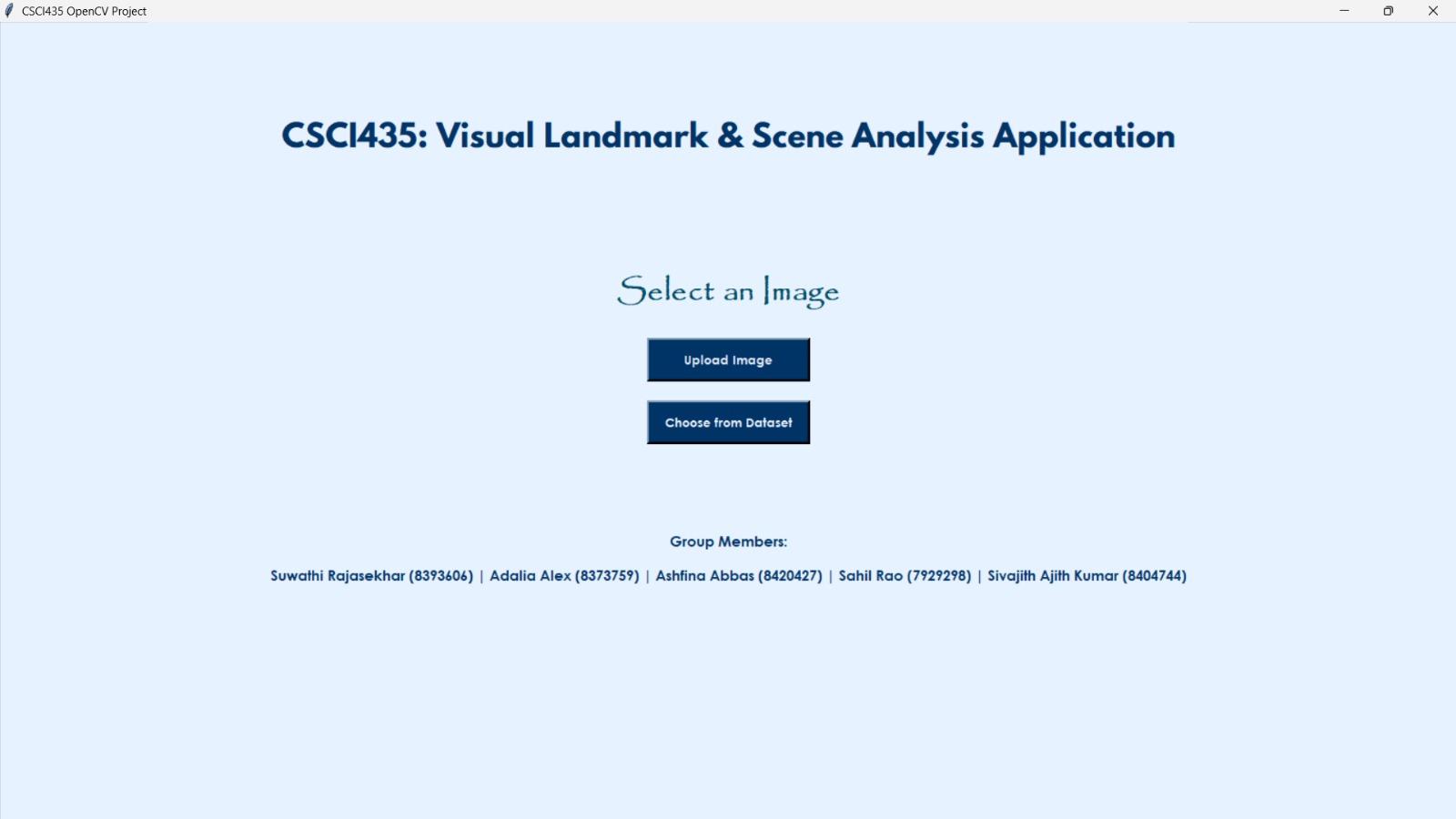
# 

# 

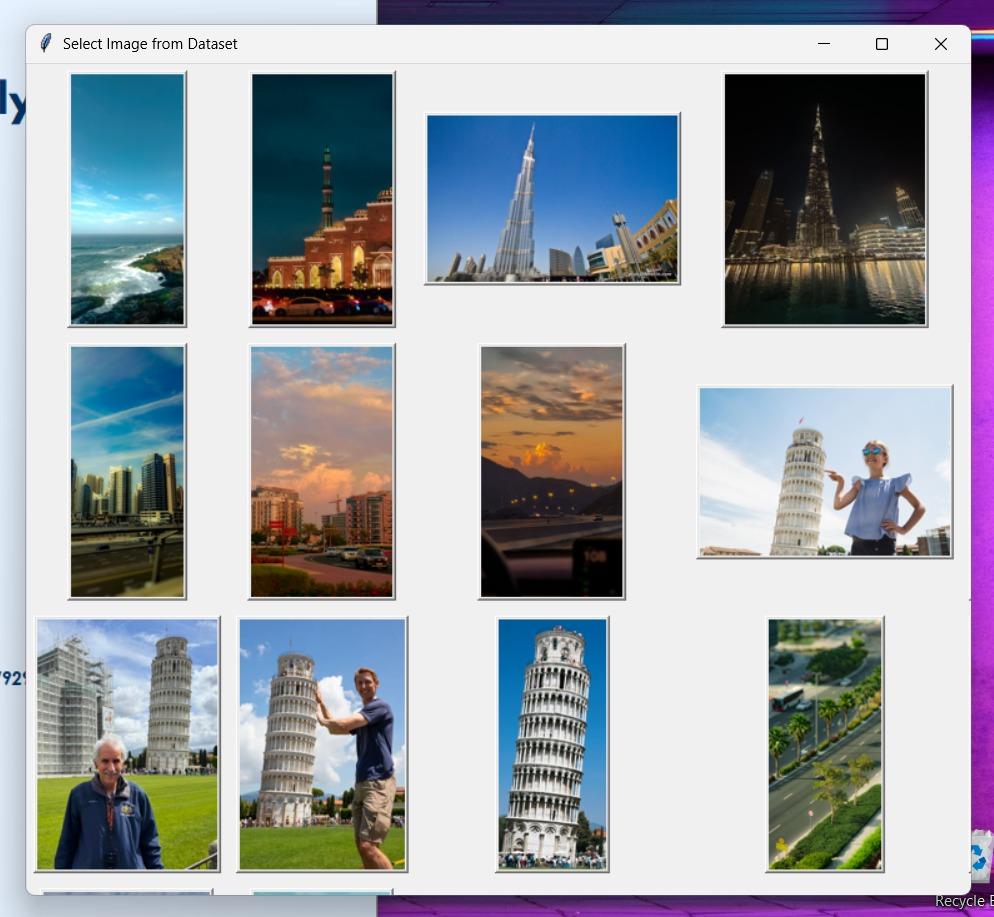
# Appendix

# Sample Outputs

Home



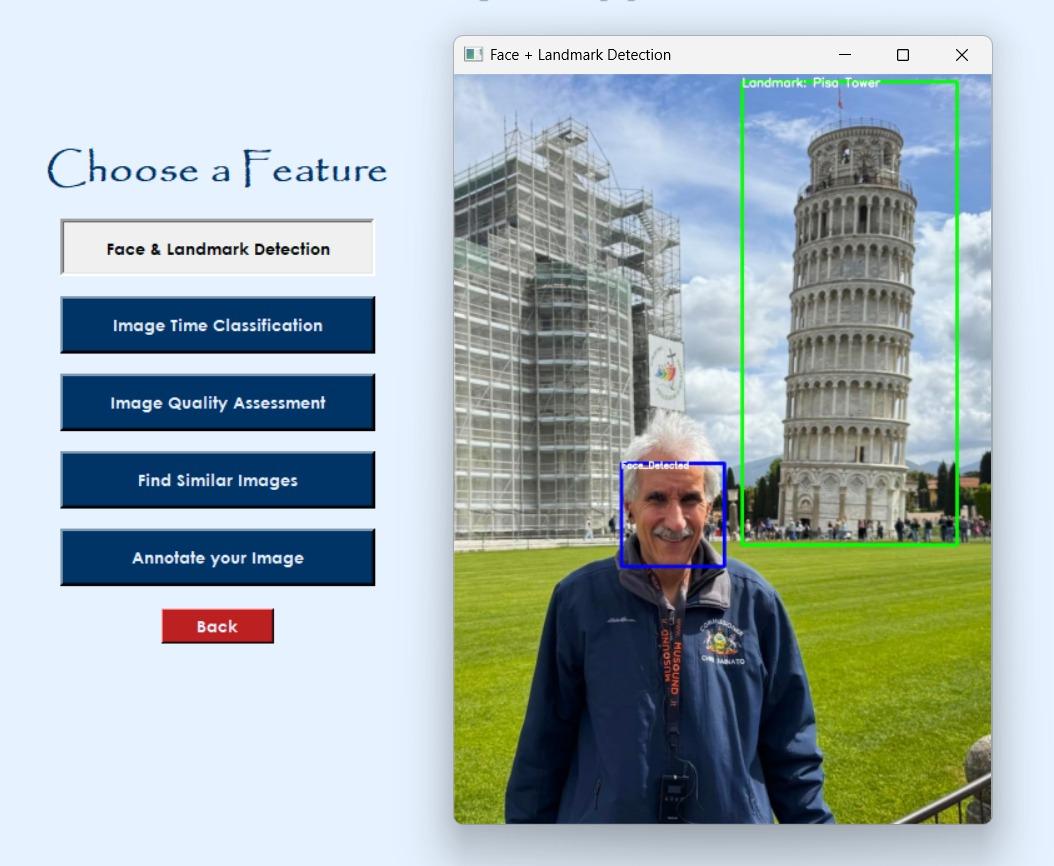
Data Selection



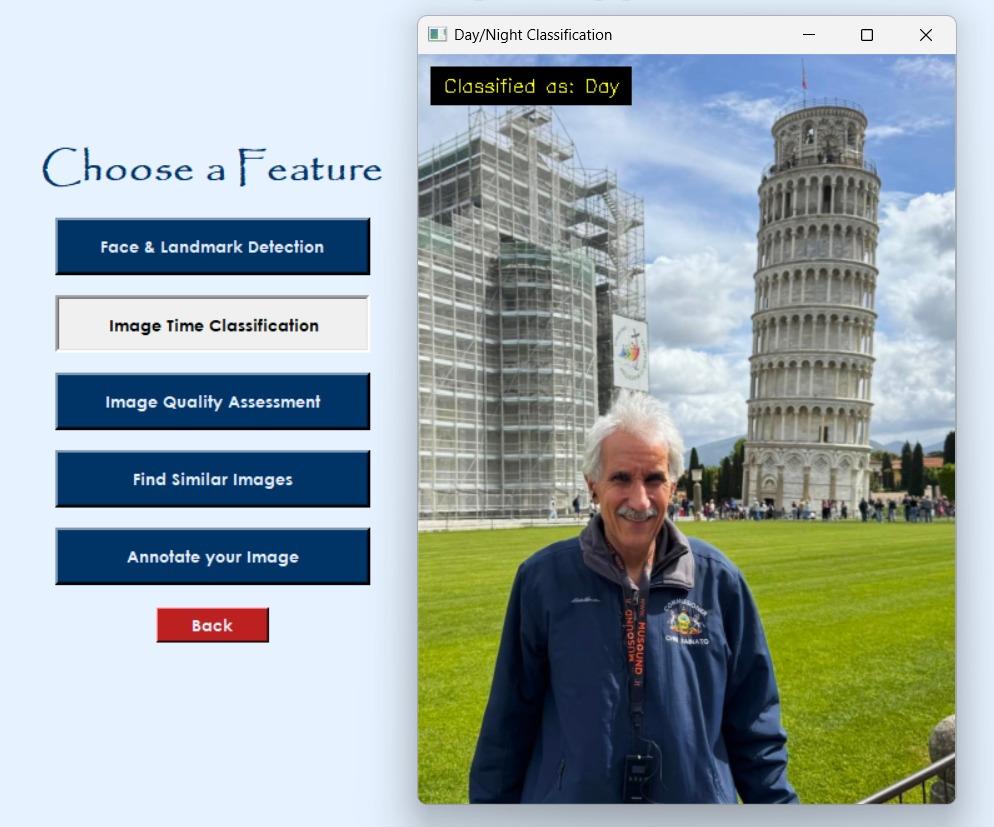
Features Screen



Face and Landmark Detection



Day Classification



Night Classification

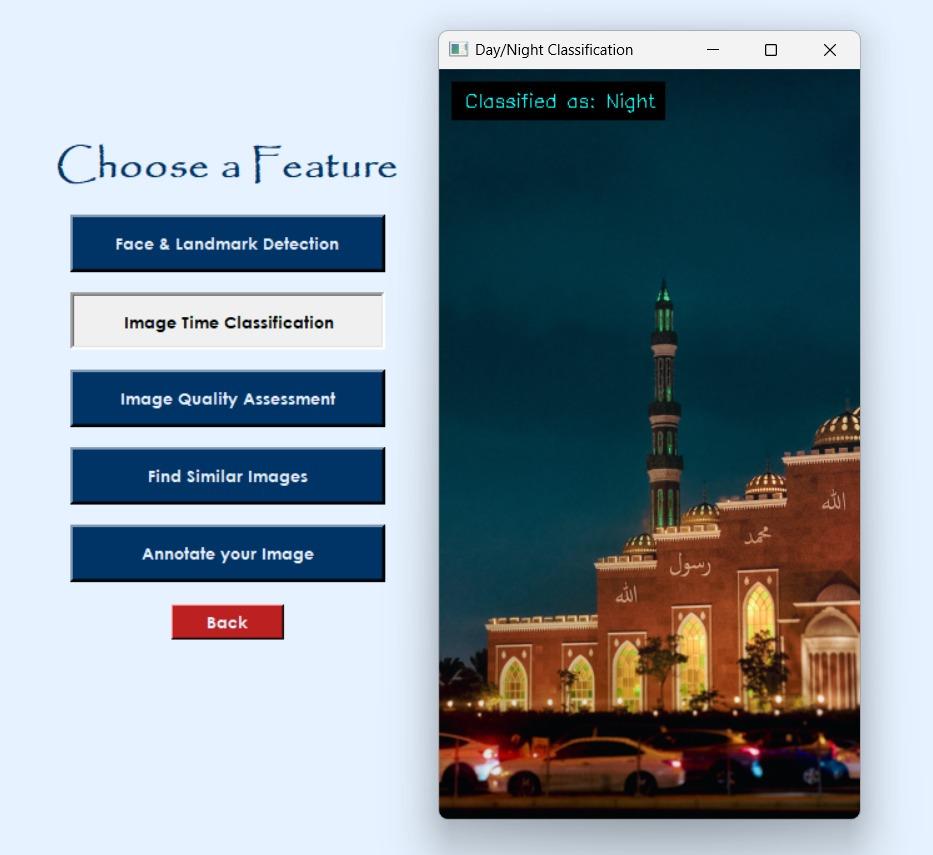
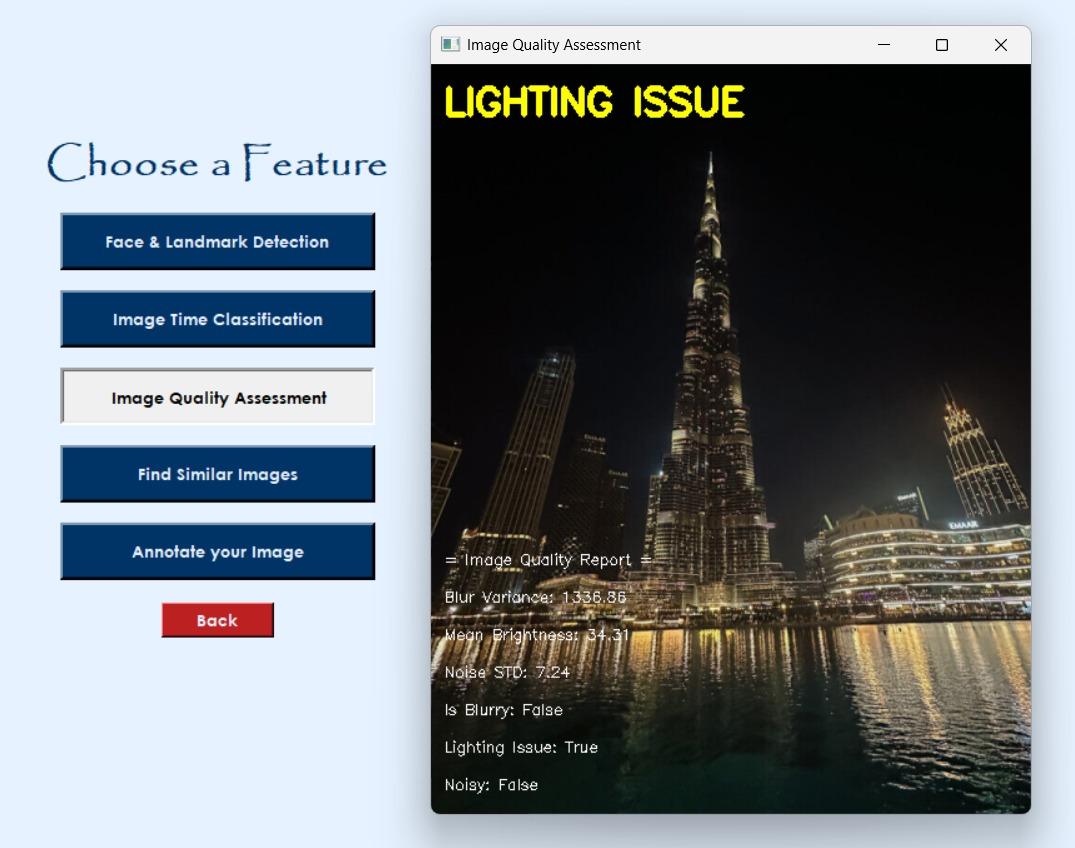
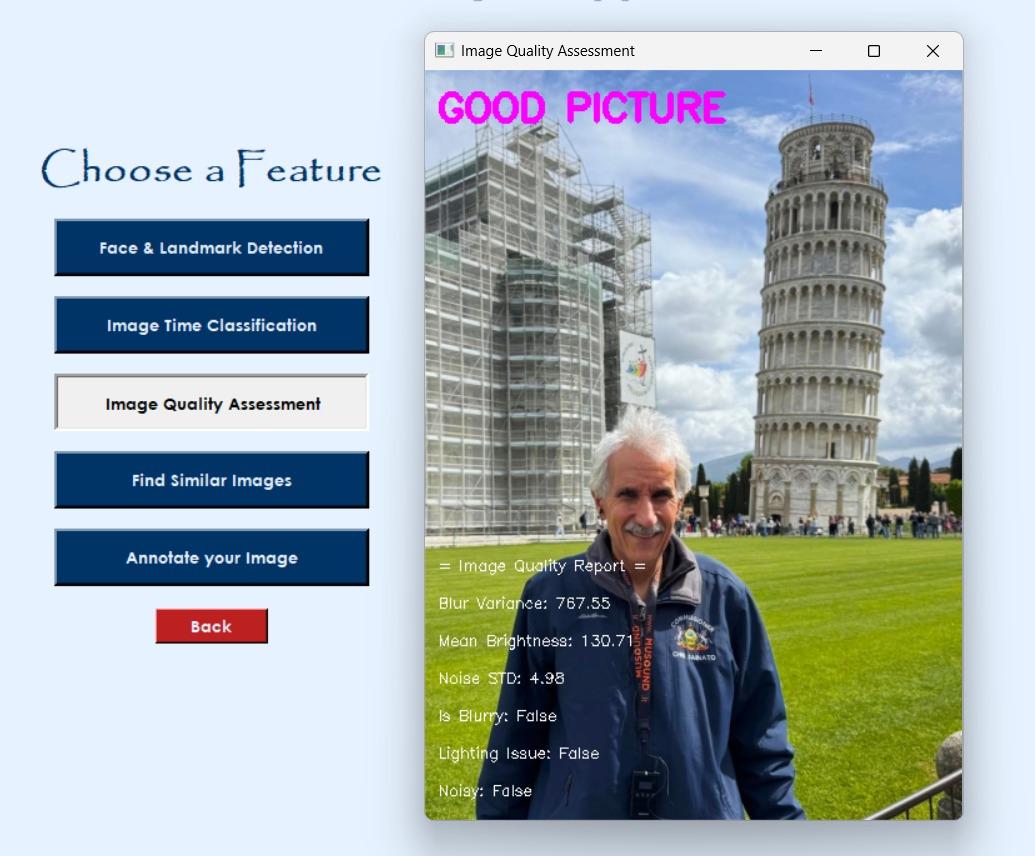
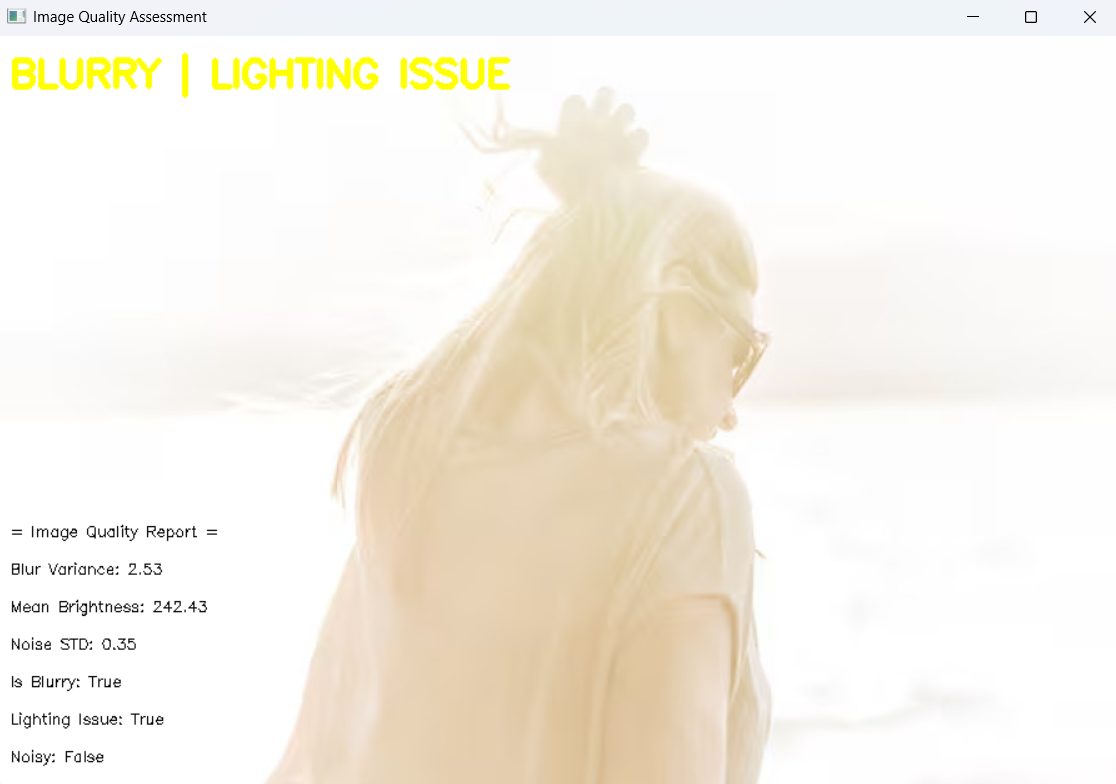
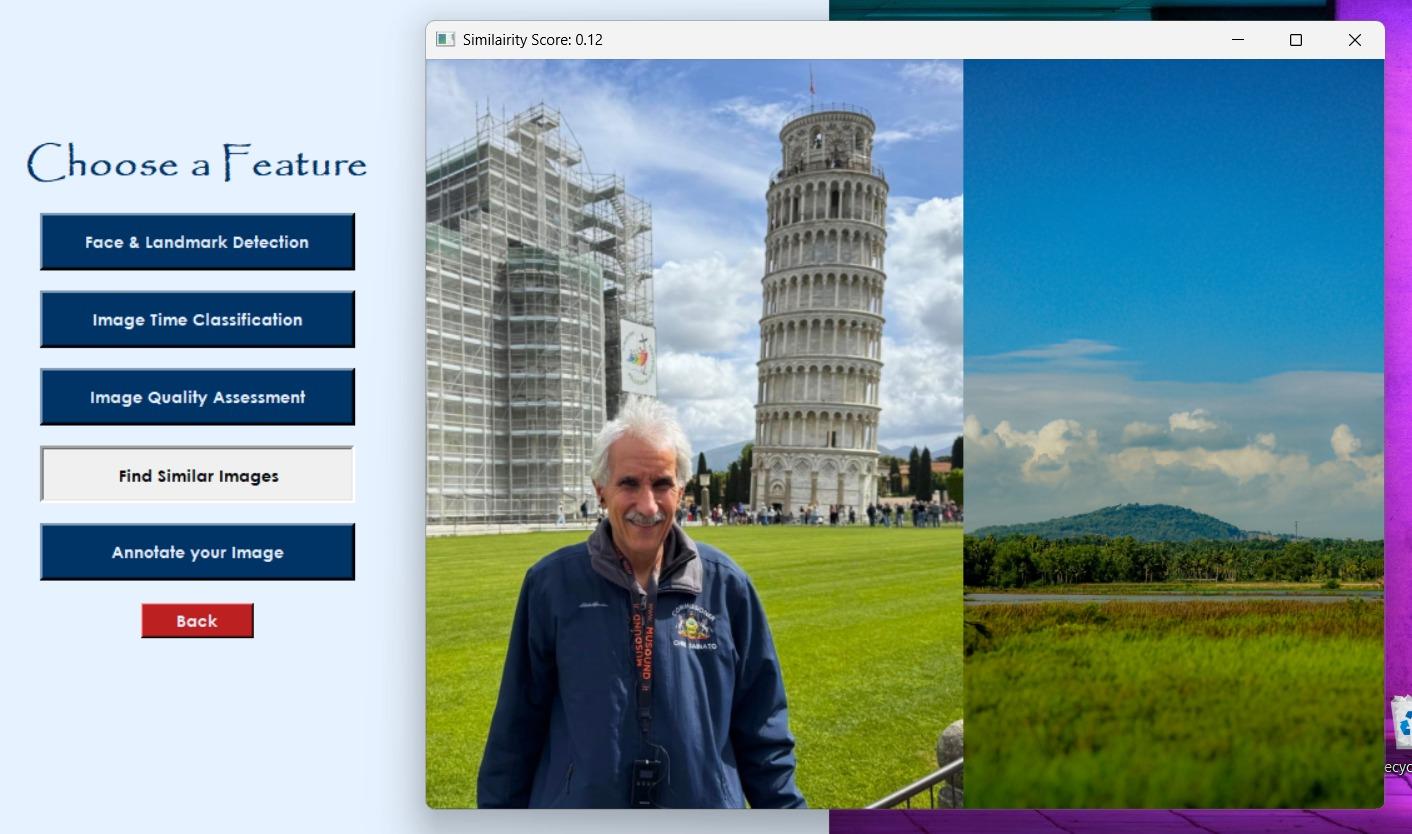


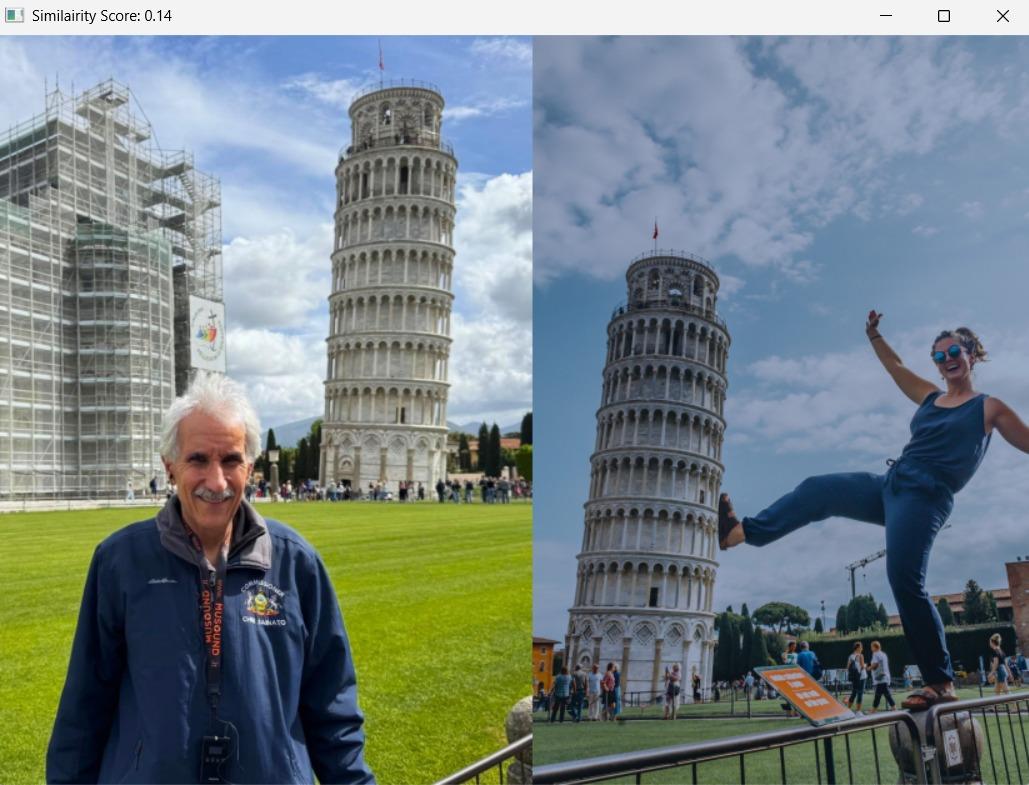
Image Assessment



Similarity Retrieval (Arranged based on the score)





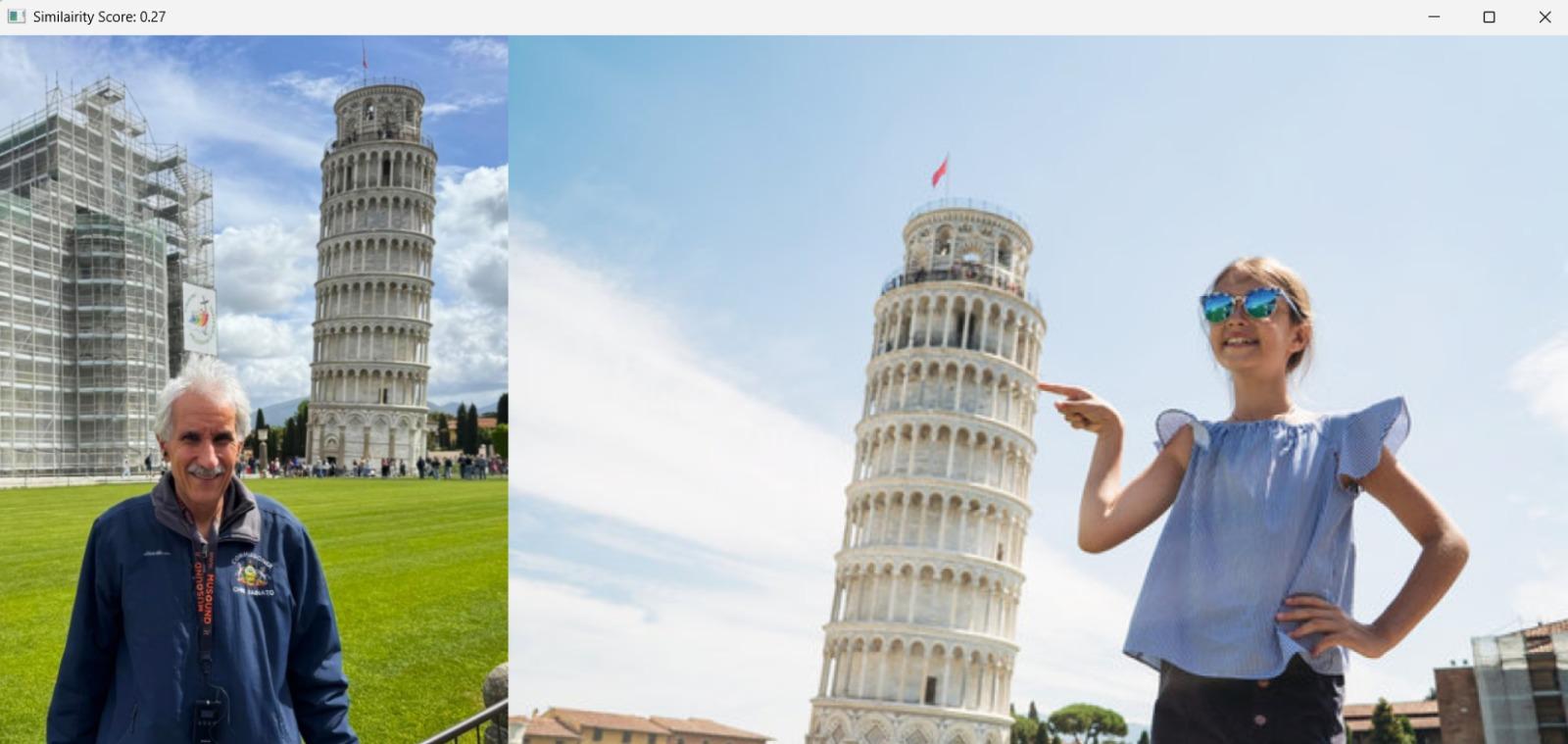
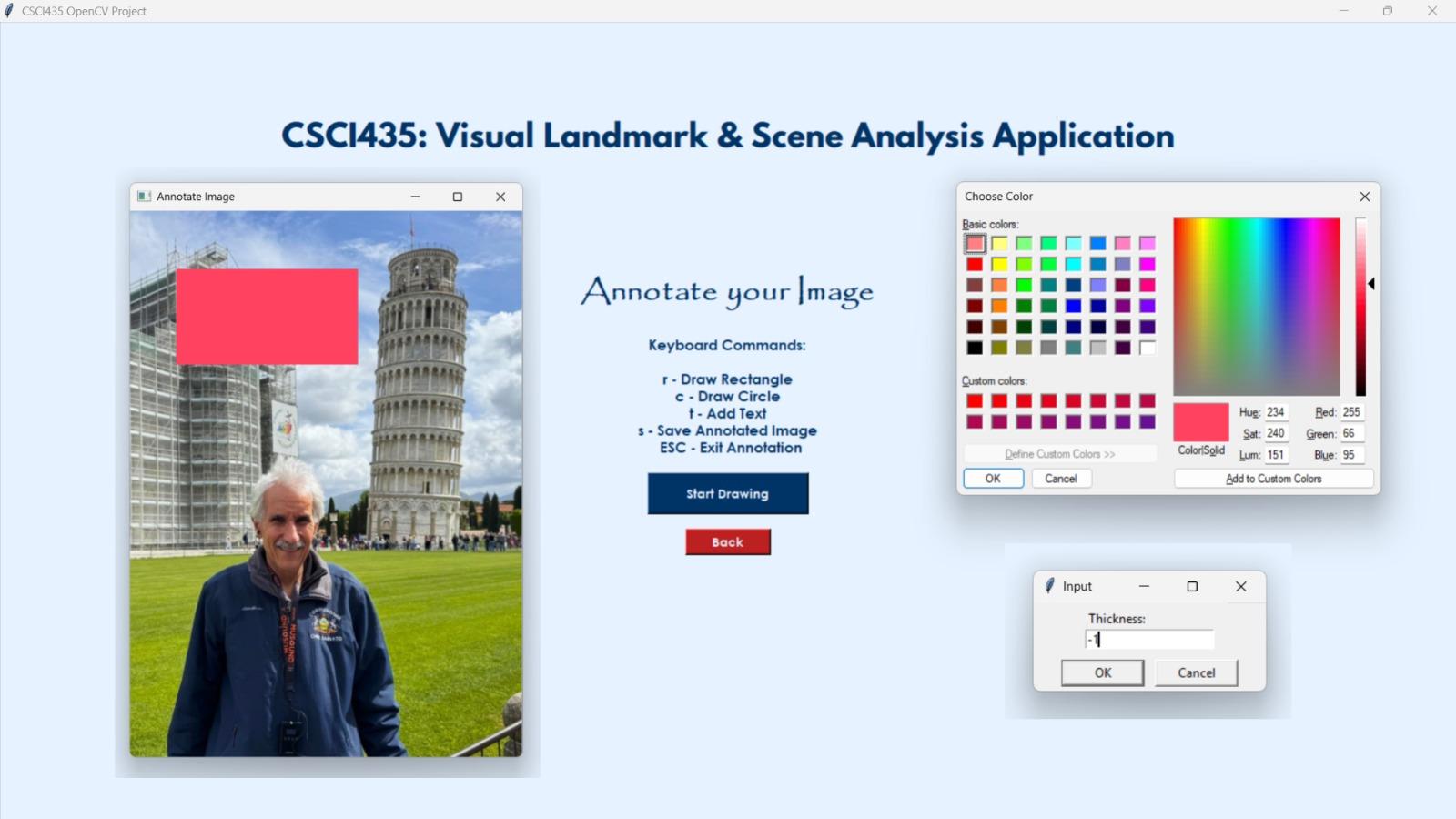
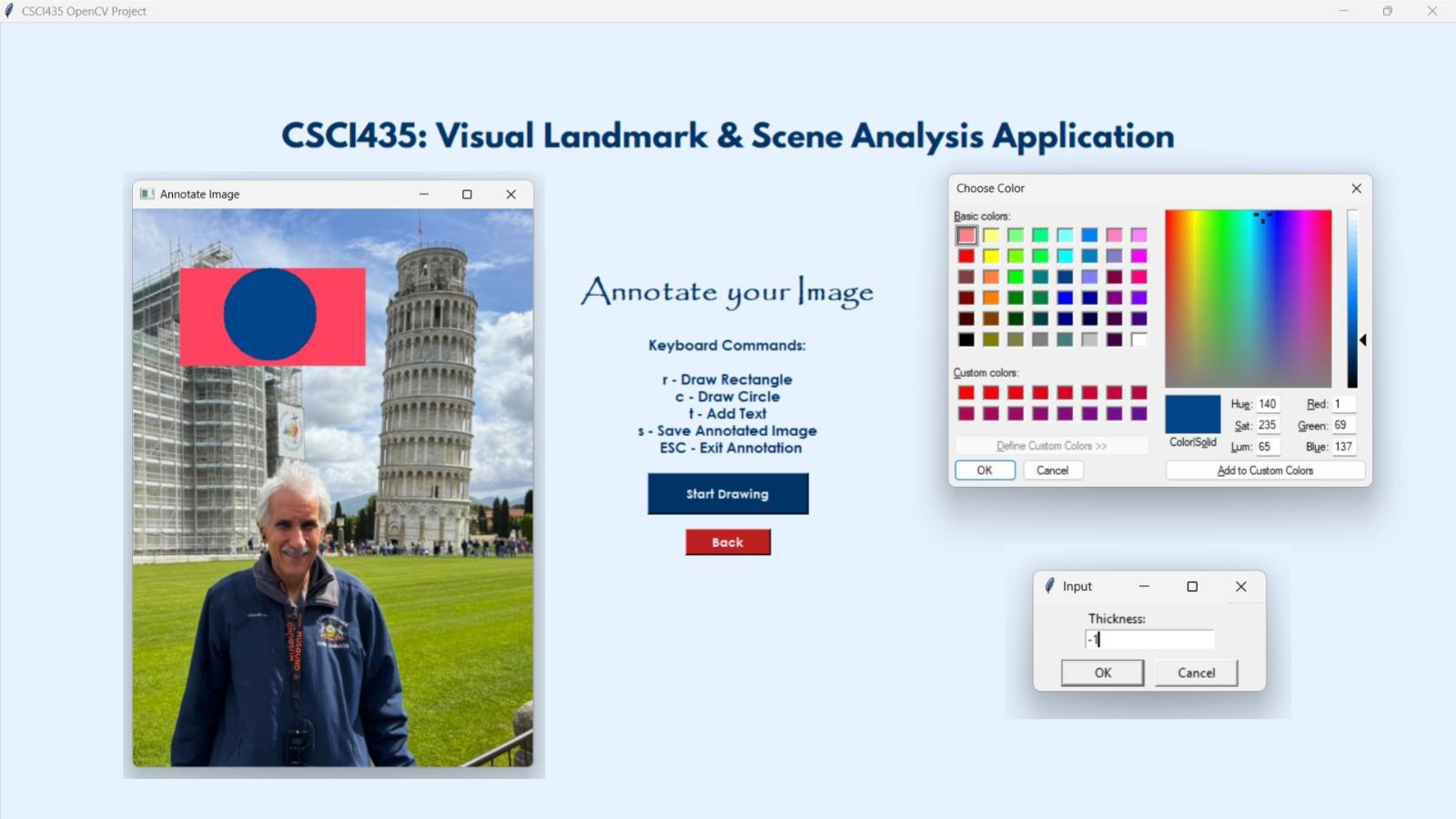


Image Annotation

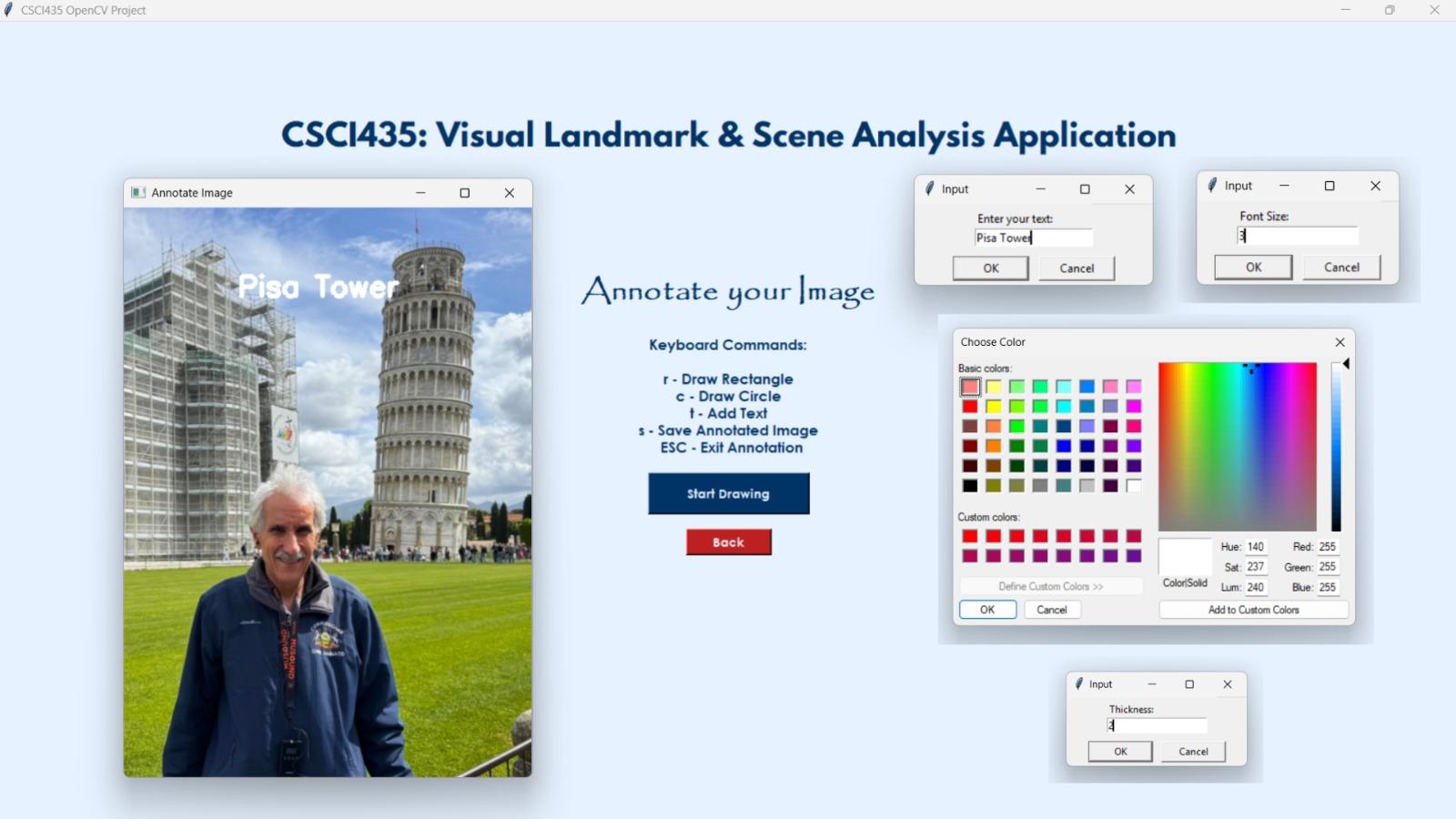
Rectangle



Circle



Text



## s