School of Engineering

Diploma in AI & Data Engineering

EGT215 - Computer Vision

SMARTGUARD User Manual

AY2023S2

Done By:

Contents

Purpose of SMARTGUARD

1. **Entrance Counting System**:
   Detect people from a live camera, count the number of people entering the worksite. People count could be used in emergency management, for headcount during an emergency.
2. **Attendance system**:
   Using facial recognition, it will detect registered and authorized personnel who are trained and aware of safety protocols, and then they will be allowed to enter the worksite thus minimizing accidents and ensuring areas have secured access.
3. **Object Detection**:
   PPE Verification detecting whether individuals entering worksites are wearing required PPE that can be selected based on user needs/requirements.

Features and Functionality

1. Detect PPE via object detection for access controls and to draw bounding boxes in the GUI to show what is detected:
    a. Hard Hats
        i. Yellow: Workers
        ii. Blue: Safety Officers / Electrical works
        iii. White: Managerial roles / Visitors
    b. Safety Vest
    c. Gloves (white)
    d. Safety Shoes (black)
2. Detecting non-compliant attire via object detection to draw bounding boxes in the GUI to inform personnel what they are missing:
    a. No Gloves
    b. No Vest,
    c. No Safety Shoe

    d. No Helmet

3. Face recognition for live identification of workers to allow authorized personnel to enter the workspace.
4. Provide the count of people entering the workspace for emergency headcount or for safety reasons.
5. Access Control:
   Allow or deny entry based on PPE compliance and detect whether personnel are wearing the right PPE for the job they are doing.
6. Automatic Updating to ensure deployment of a self-sustaining service:
   Users are allowed to update the firebase automatically, and changes will take effect at a specified time once deployed.
7. Firebase Cloud Storage to store and retrieve personnel information for Face Recognition to work:
   Cloud Storage ensures that the service is scalable and also prevents hardware storage from being a problem.
8. User Needs Customization
   Allows for the user to select what PPE needs to be detected and can be modified quickly to suit needs or rapid changes.
9. Speech Support
    a. A toggle option for user input selection **only**
    b. A **permanent** feature for identification and access controls
10. Bounding Boxes Classification of classes to standardize bounding boxes' colour for easier identification of missing/detected PPE.
11. Bounding Boxes Label to identify the PPE/Object detected.

Installation Guide

System requirement:

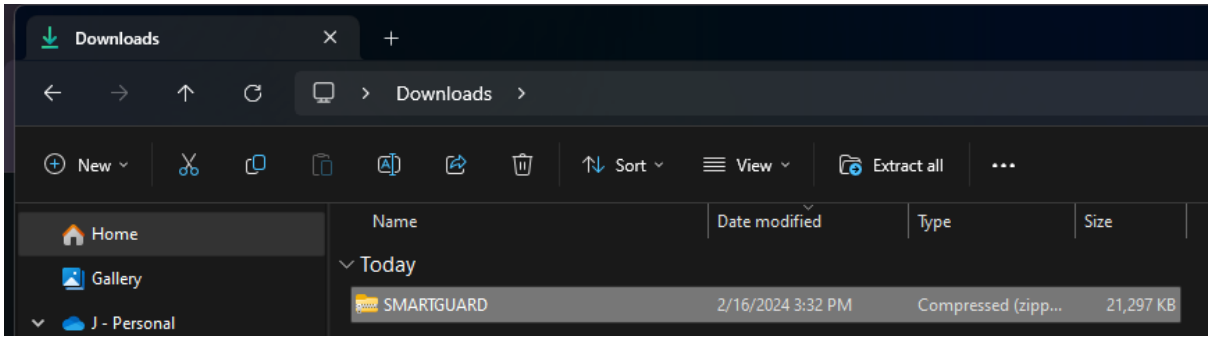
Windows 10 and above

Python IDE:

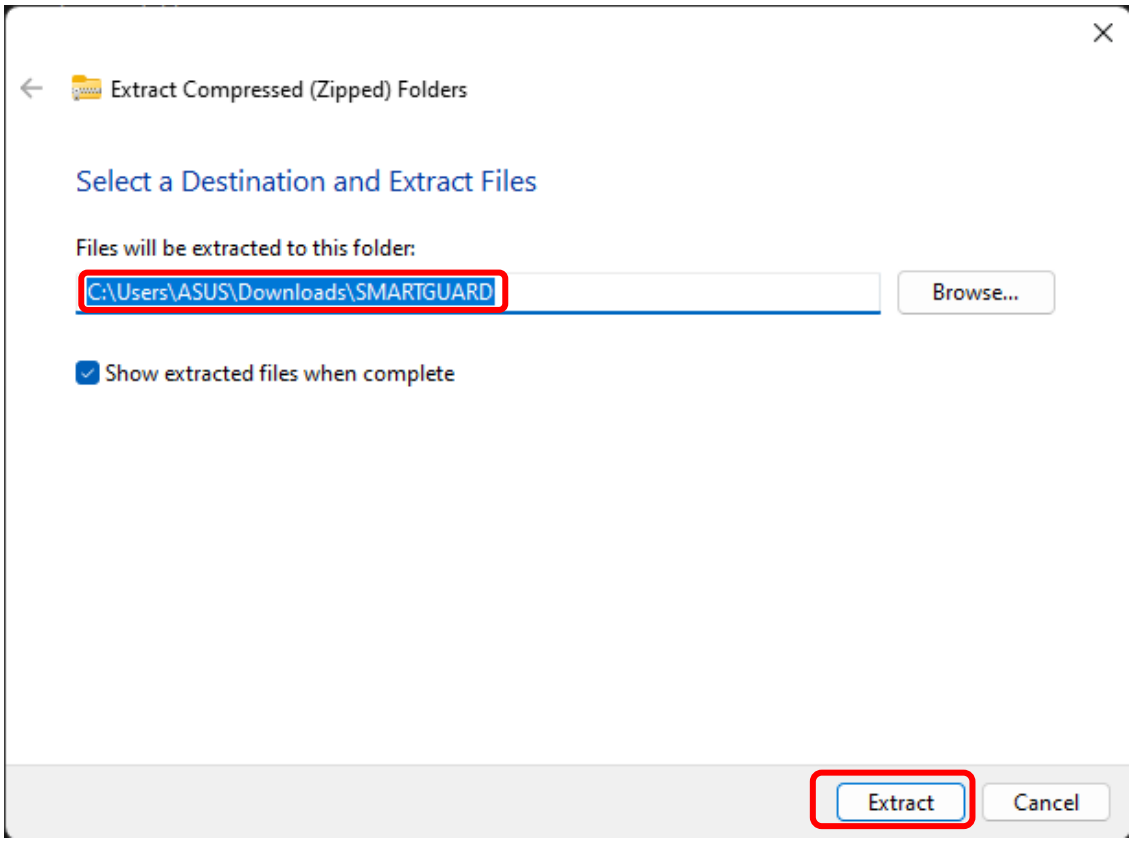
Pycharm 2023.3.3

Install SMARTGUARD:

Download and unzip the provided file, take note of the location where you unzipped the file, it will be used later.



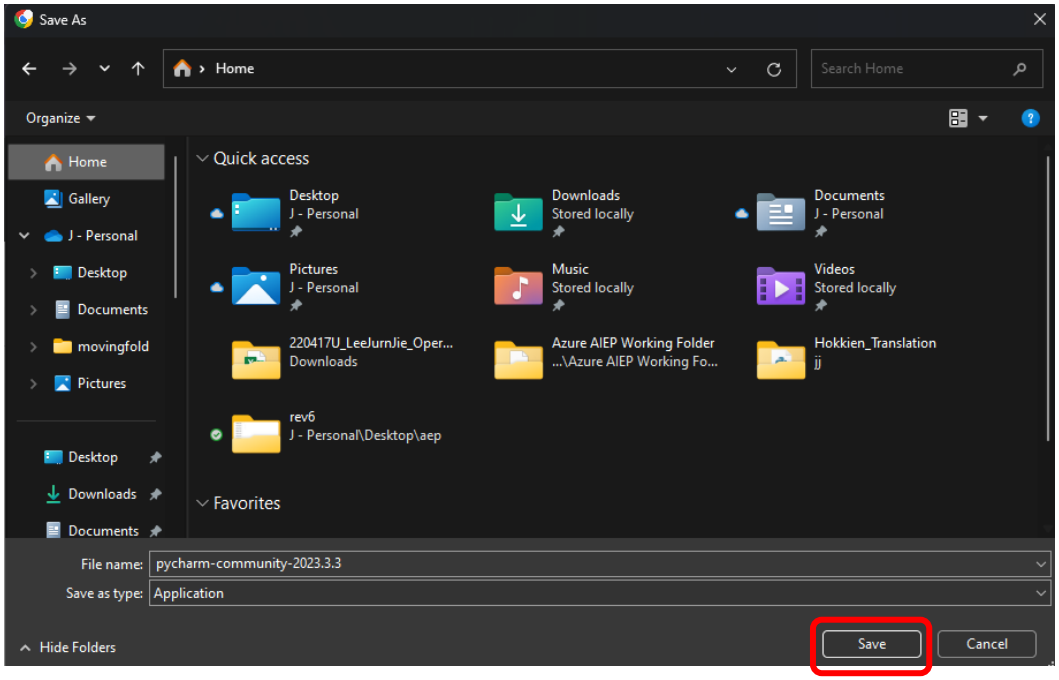
Ctrl + C, copy this folder directory and record it down. Click extract.



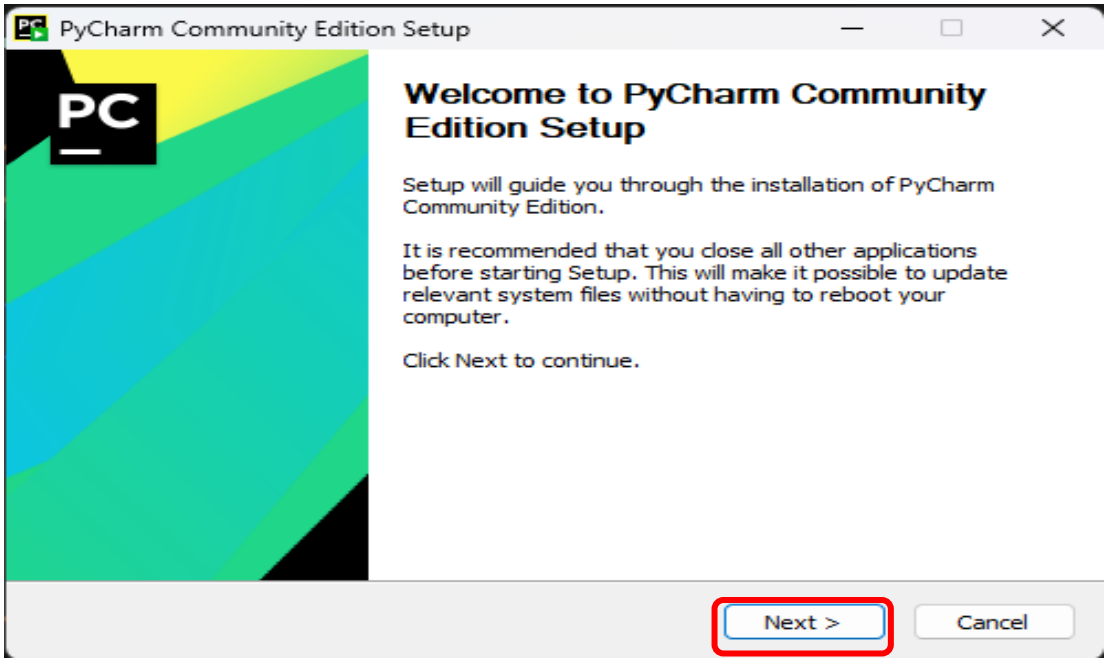
Install PyCharm:

Navigate to PyCharm, PyCharm installation can be found in the link provided. Follow the default installation instructions. [Install PyCharm here]

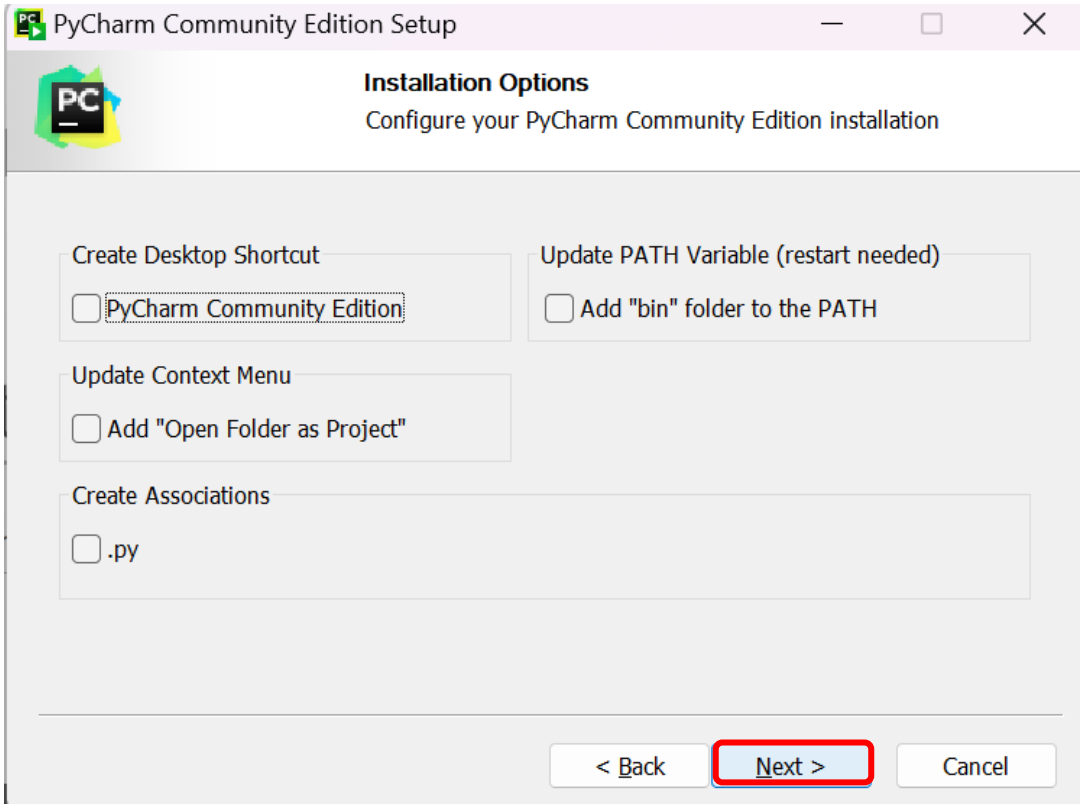
Click Save.



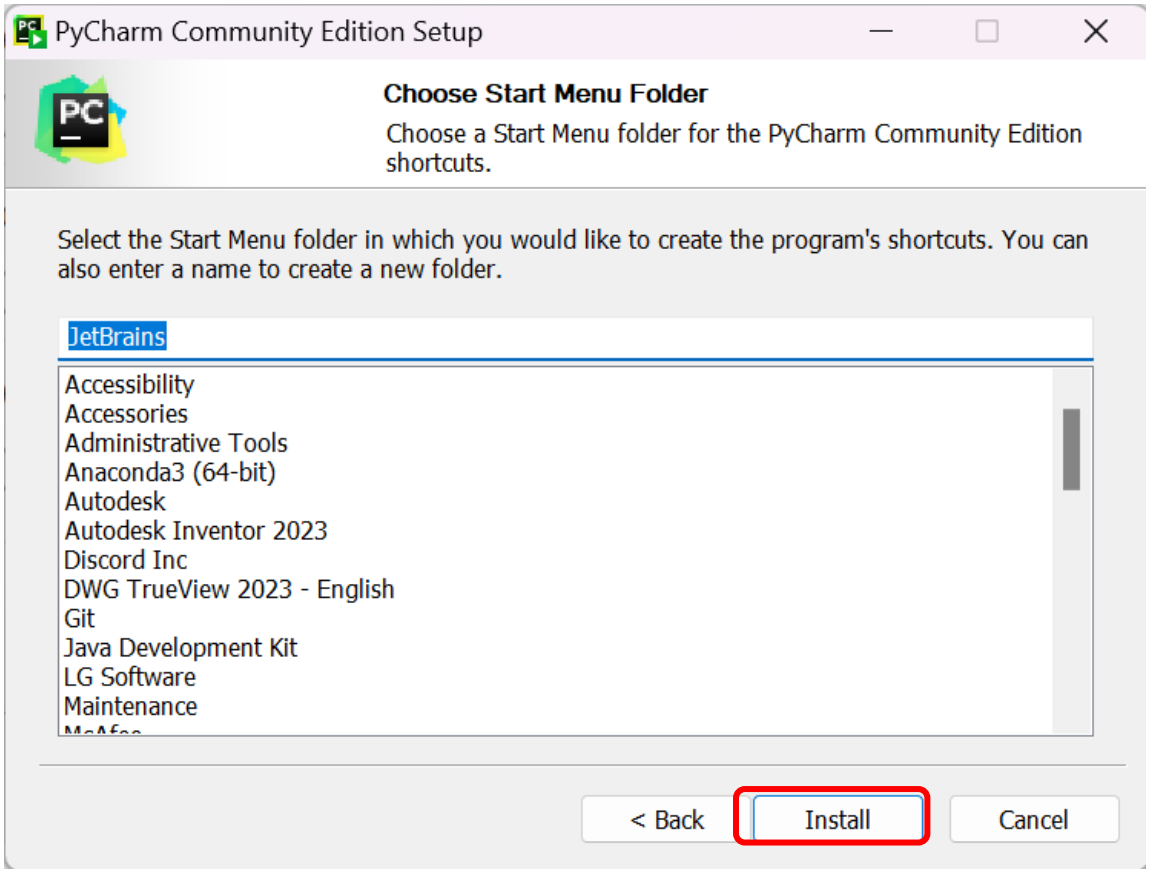
Click Next.



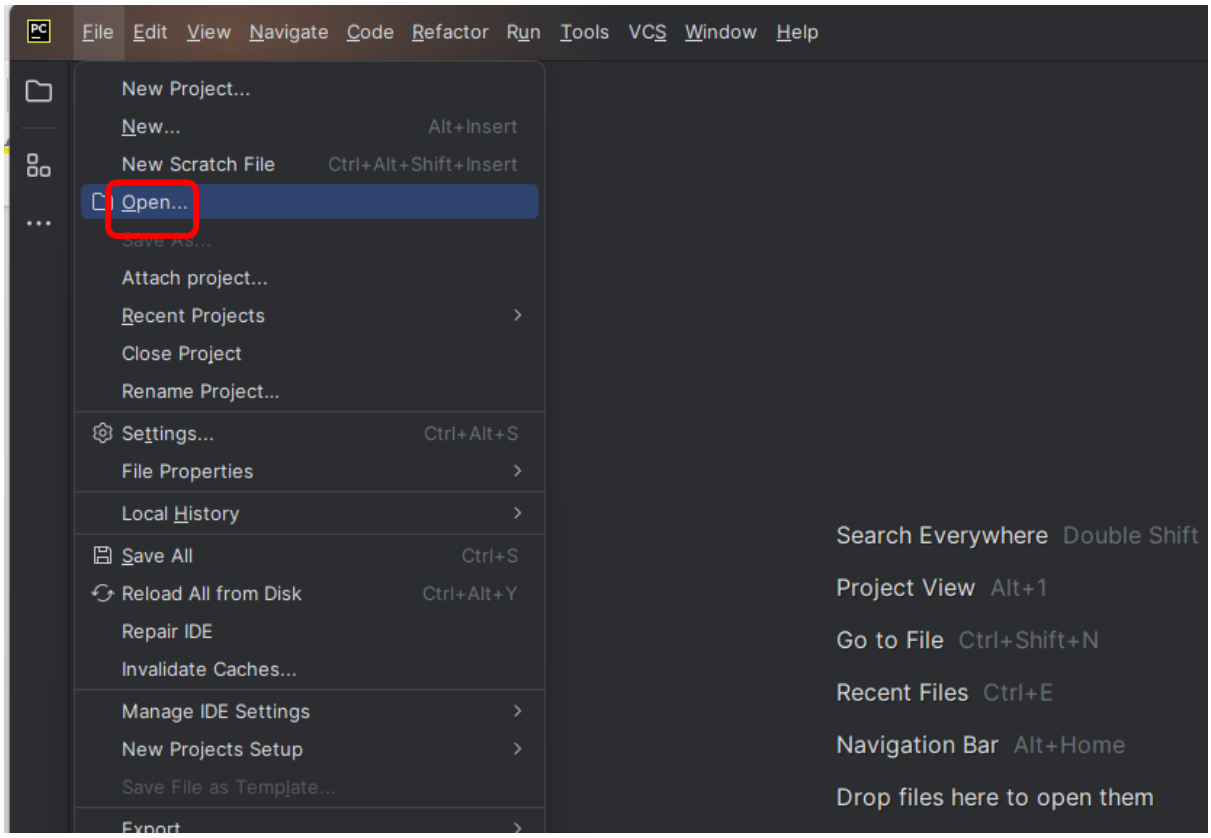
Click Next



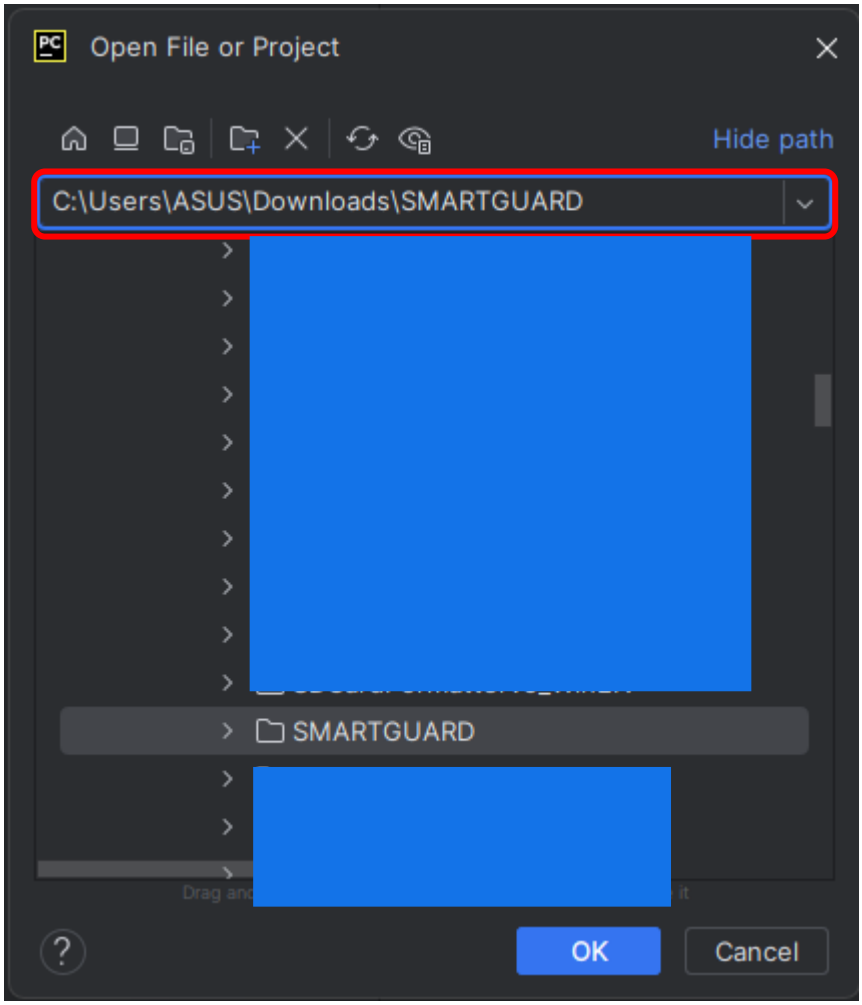
Click Install



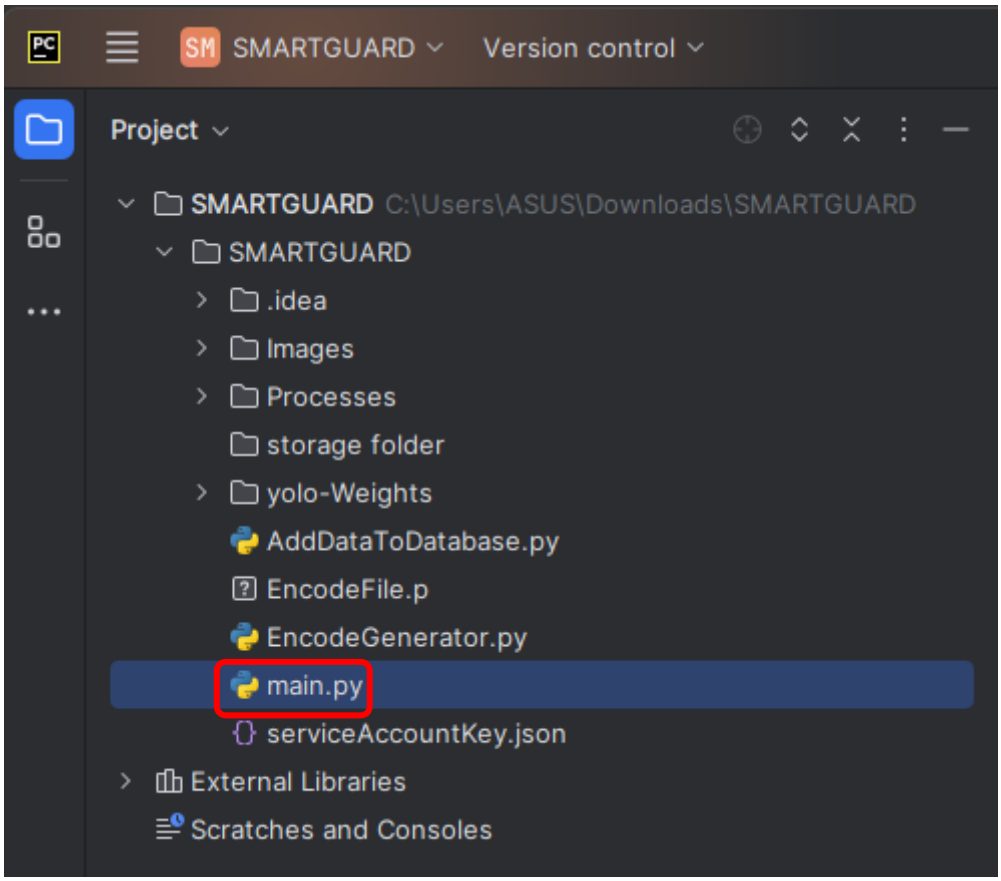
After the pop-up window appears after installation, click Open.



Ctrl + V, paste the earlier copied sentence into the search bar. Click ok.

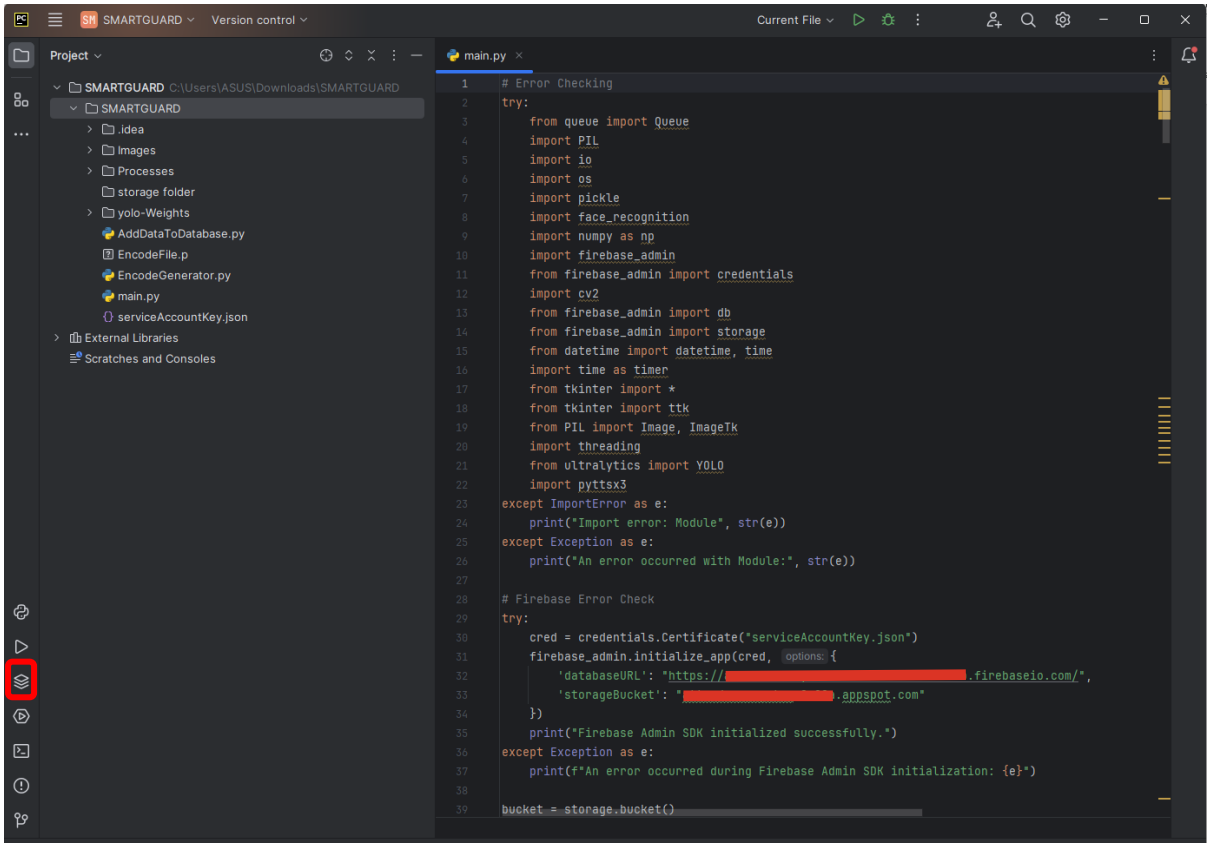


Double click main.py



Install Libraries/Dependencies

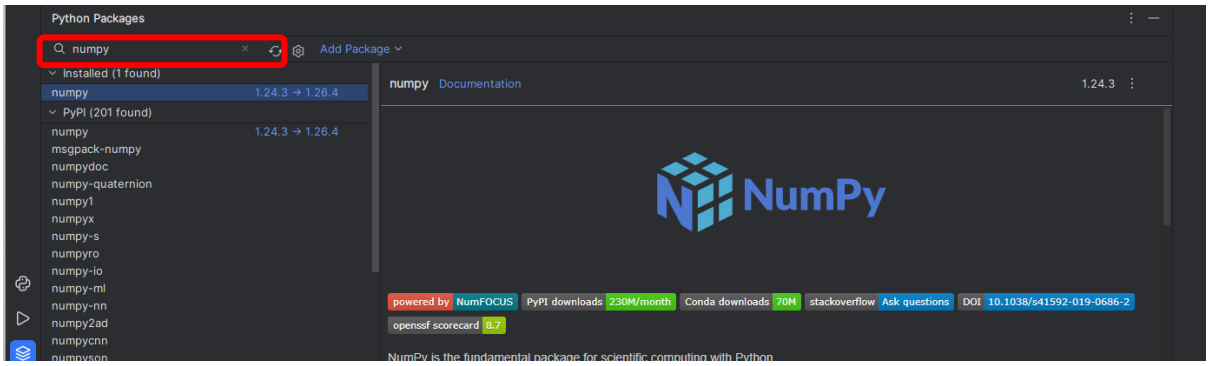
Library dependencies: click the icon located in the red box. Refer to the image below



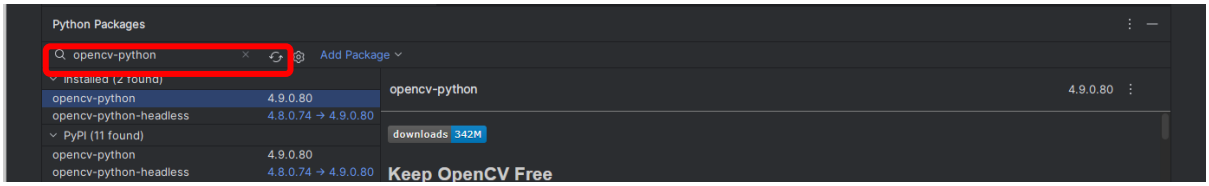
To install, click the install button(if there is one) and just install the latest version (refer to queuelib example below)

- Queue (Search “queuelib”)

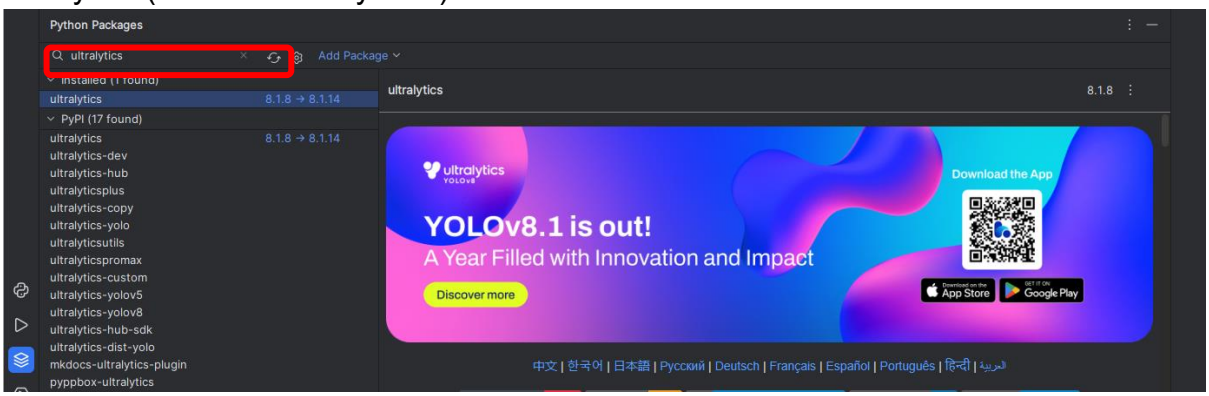
- Numpy (Search “numpy”)



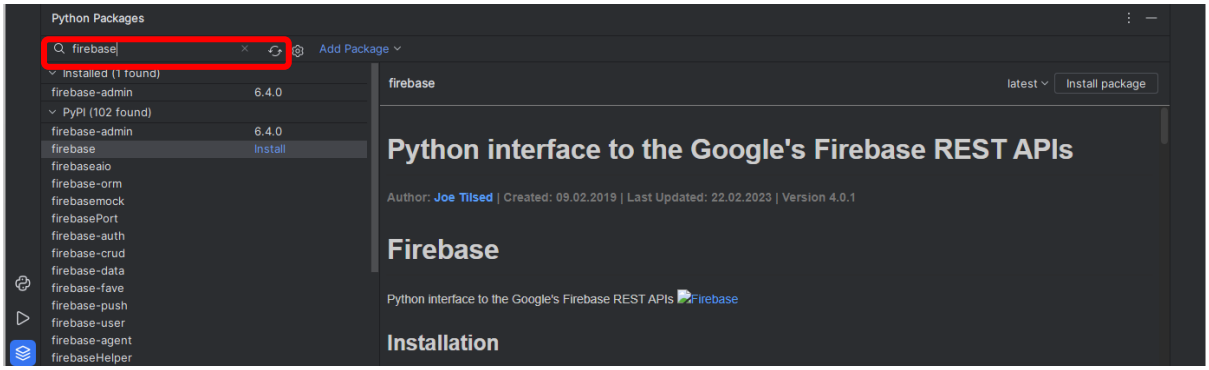
- cv2 (Search “opencv-python”)



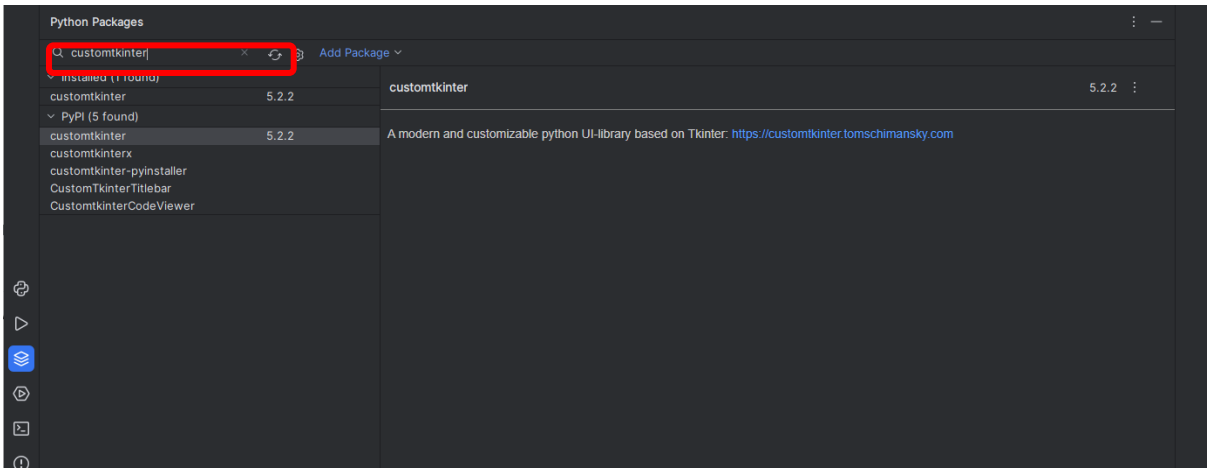
- ultralytics (Search “ultralytics”)



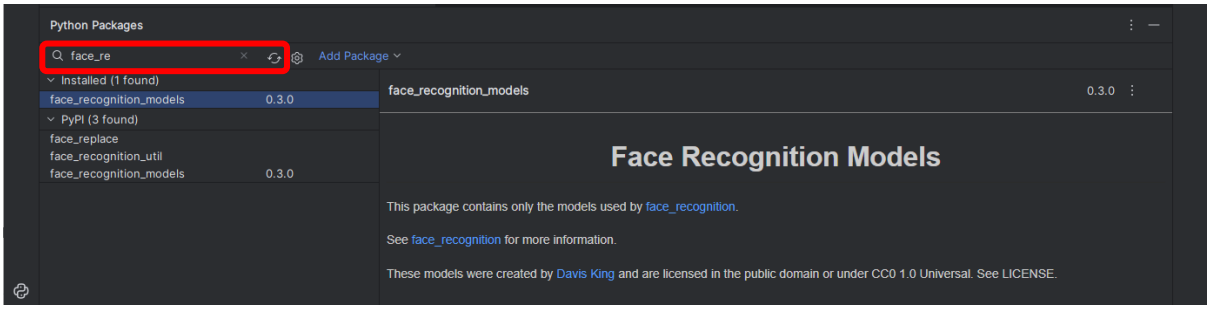
- firebase_admin (Search “firebase-admin)



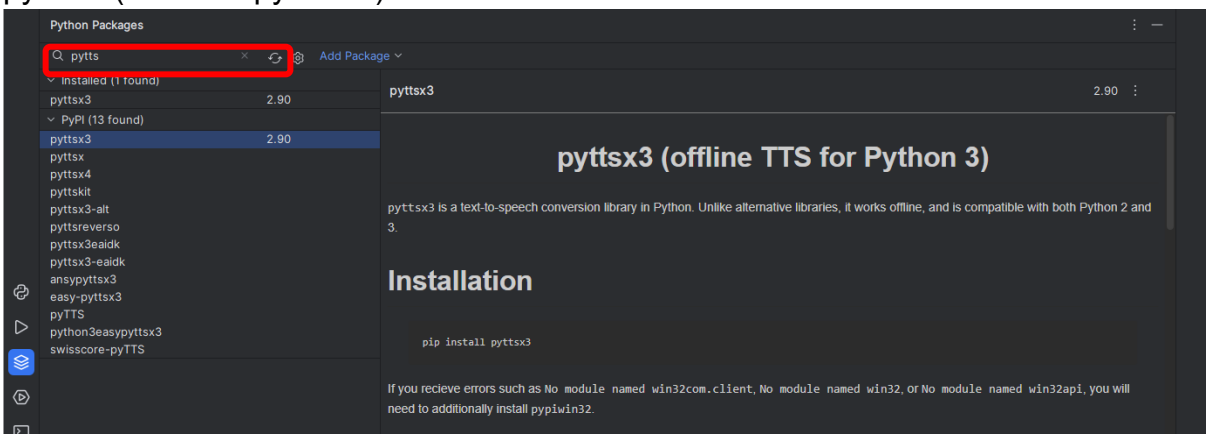
- tkinkter (Search “customtkinter”)



- face_recognition (Search “face_recognition_models”)



- pyttsx3 (Search “pyttsx3”)

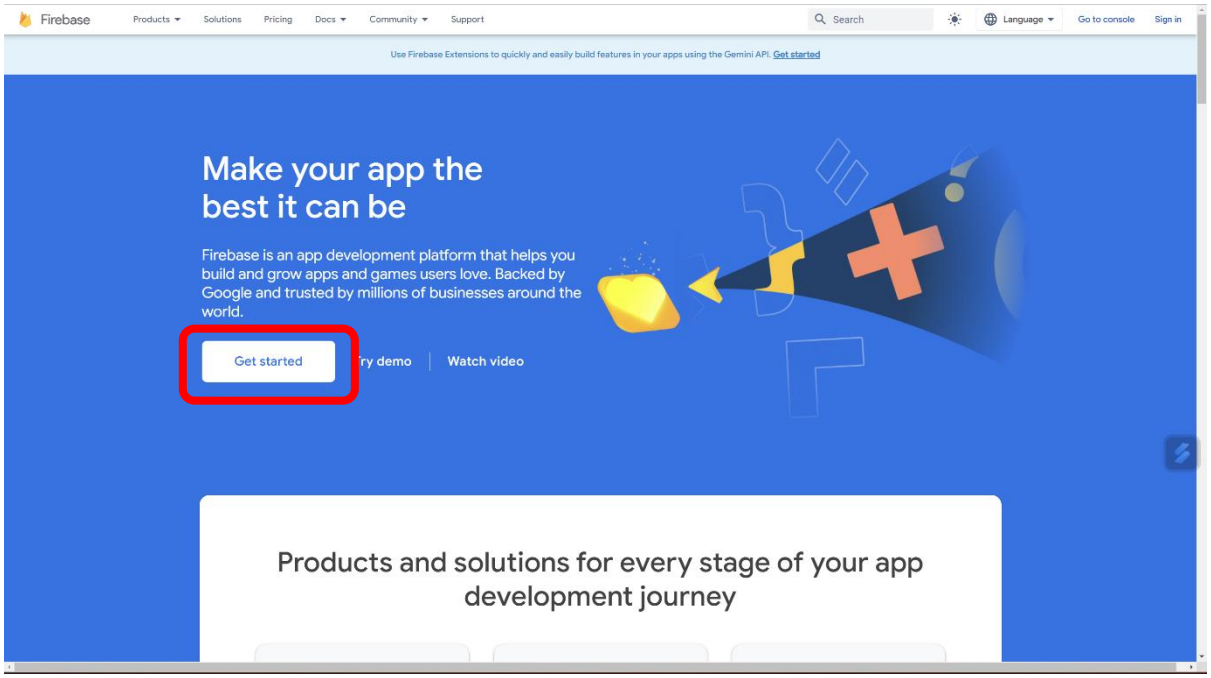


Create a Google Account

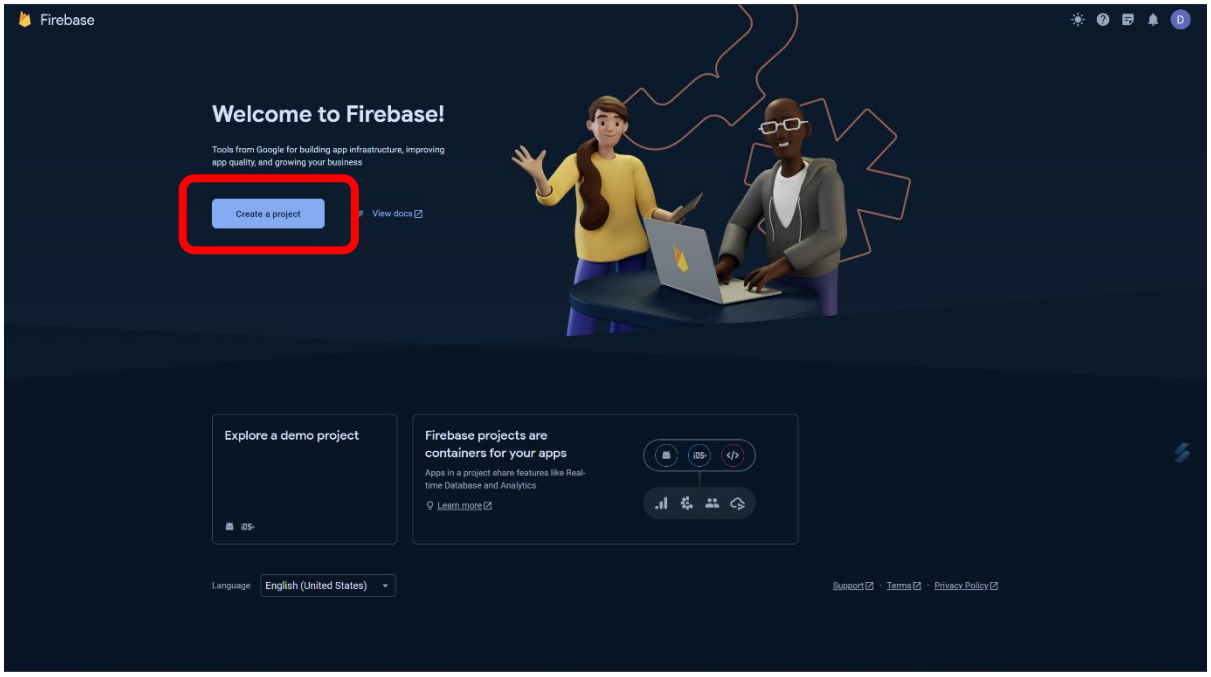
1. Navigate to https://www.hubspot.com/email-signature-generator/set-up-gmail-account and follow the instructions.

Creating Firebase Account and Database

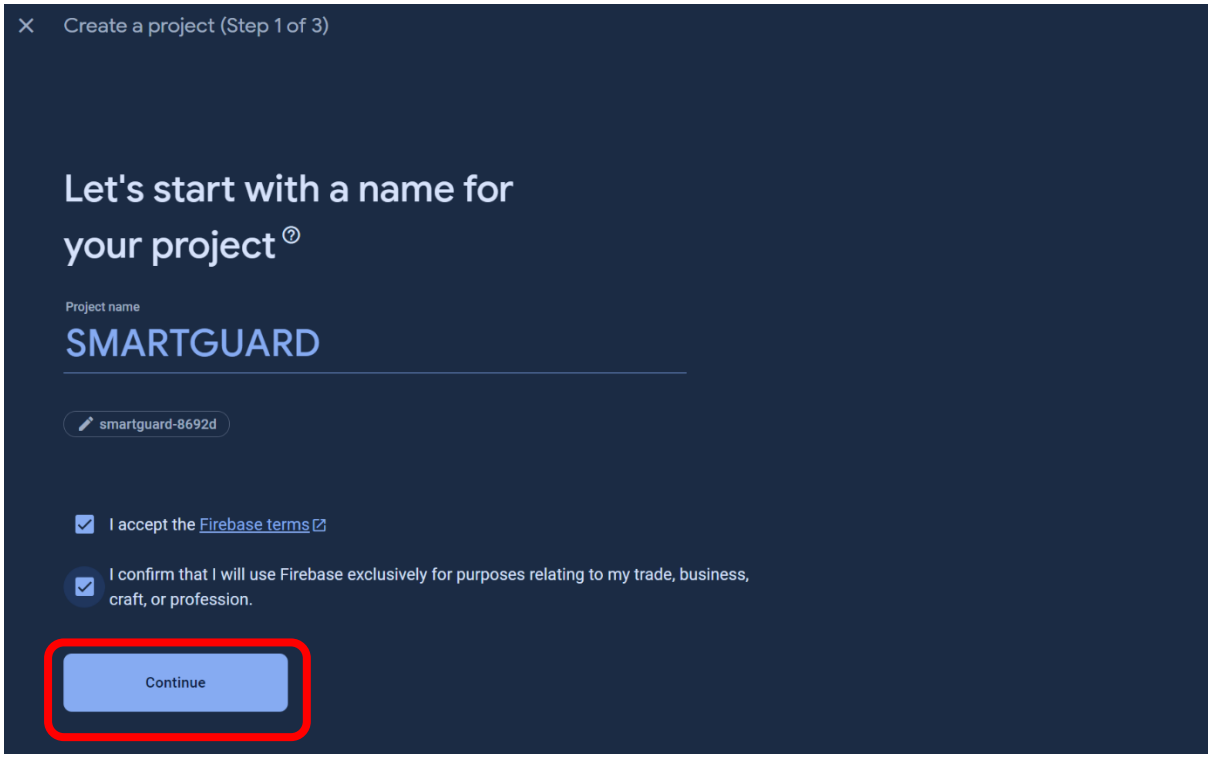
1. Navigate to https://firebase.google.com/
2. Click “Get Started”



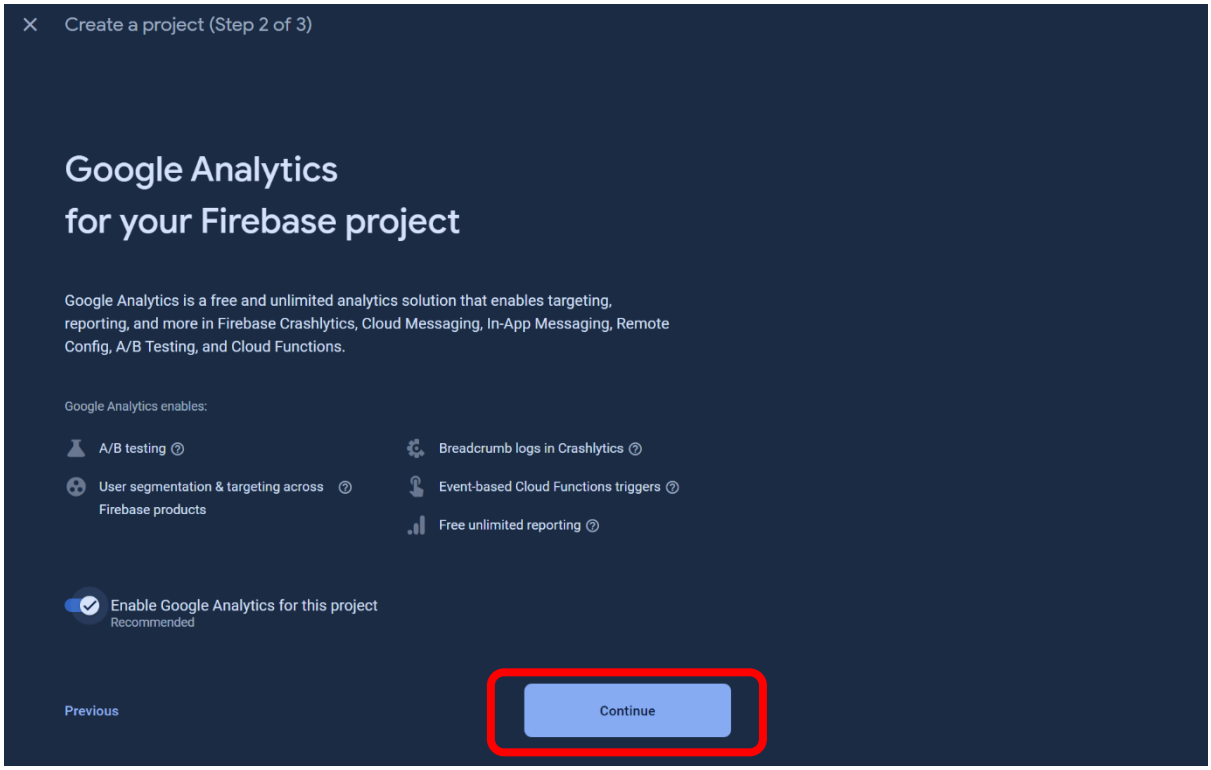
3. Sign in with your Google Account that you created.
4. Click “Create a project”



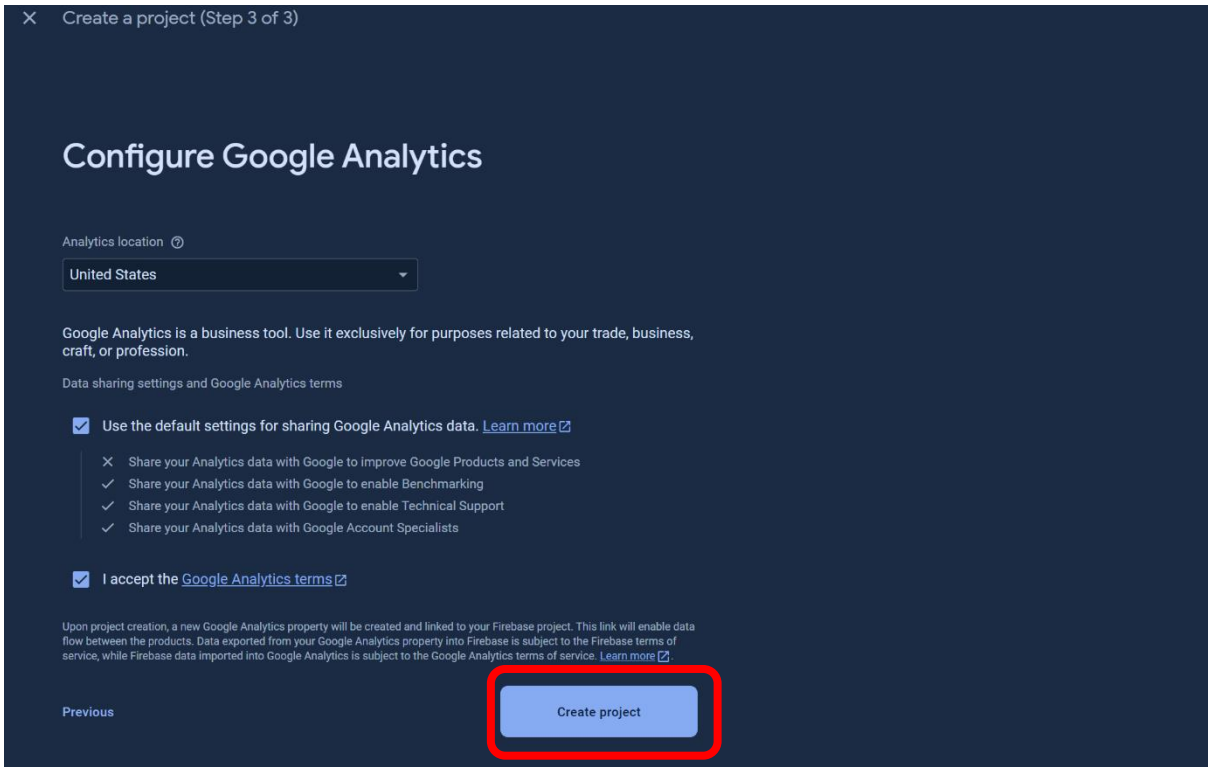
5. Enter “SMARTGUARD” under “Project Name”
6. Check “I accept the Firebase terms”
7. Check “I confirm that I will use Firebase exclusively for purposes relating to my trade, business, craft, or profession”
8. Click “Continue”



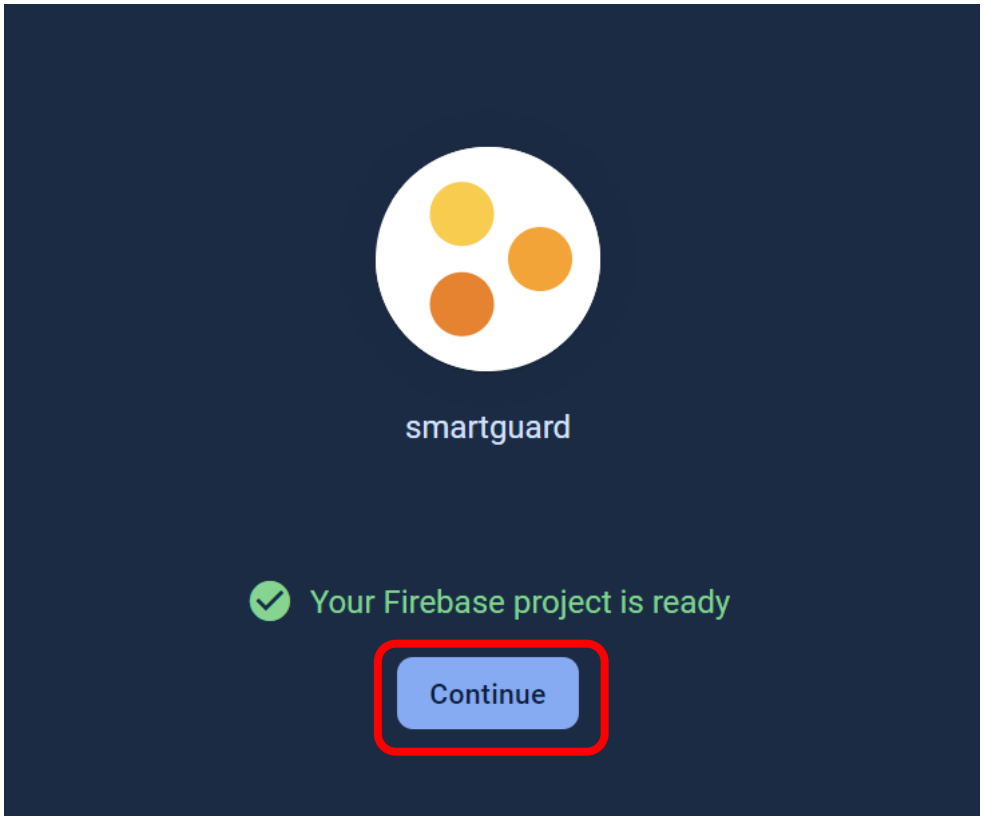
9. Click “Continue”



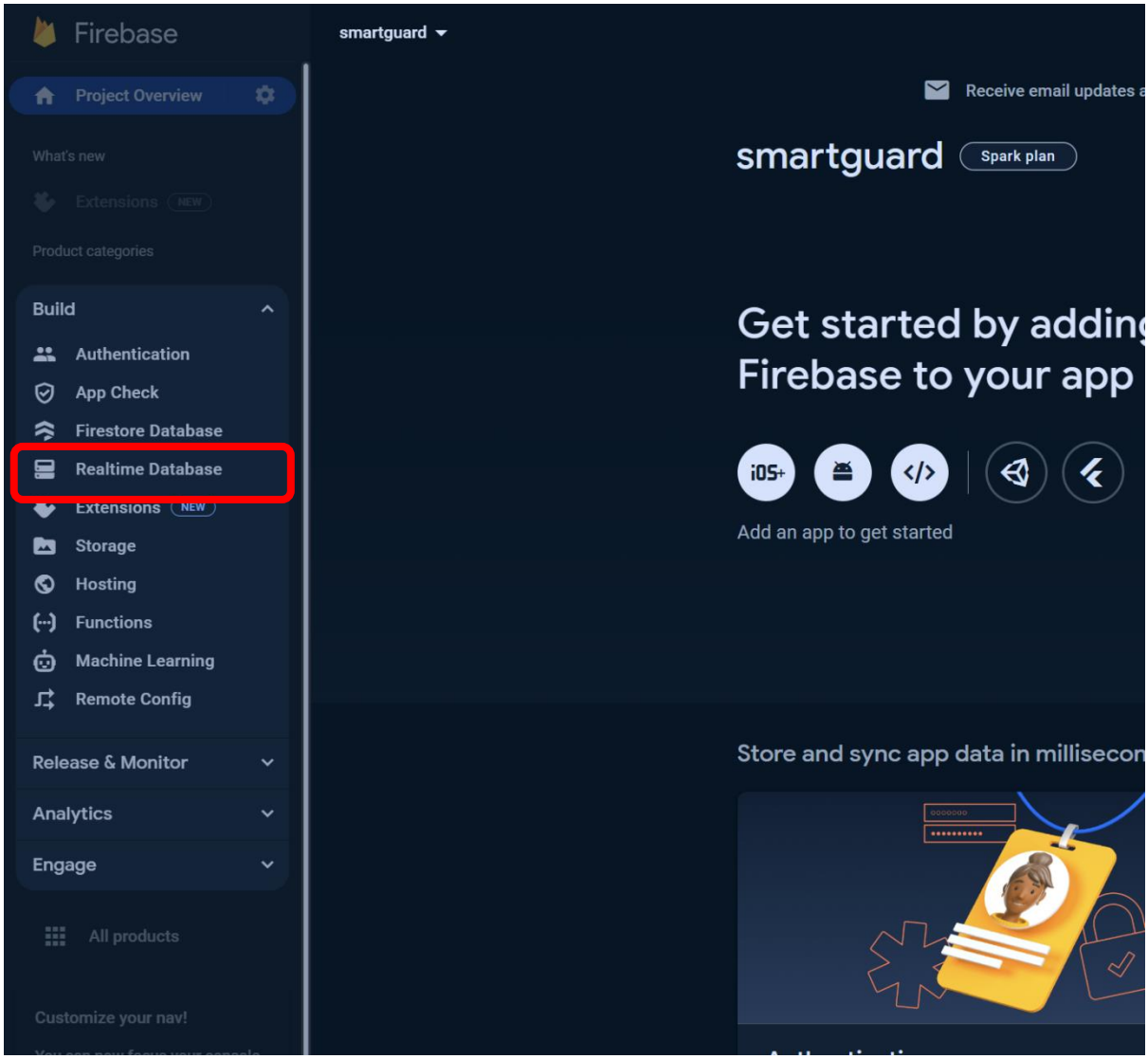
10. Check “I accept the Google Analytics terms”
11. Click “Create Project”



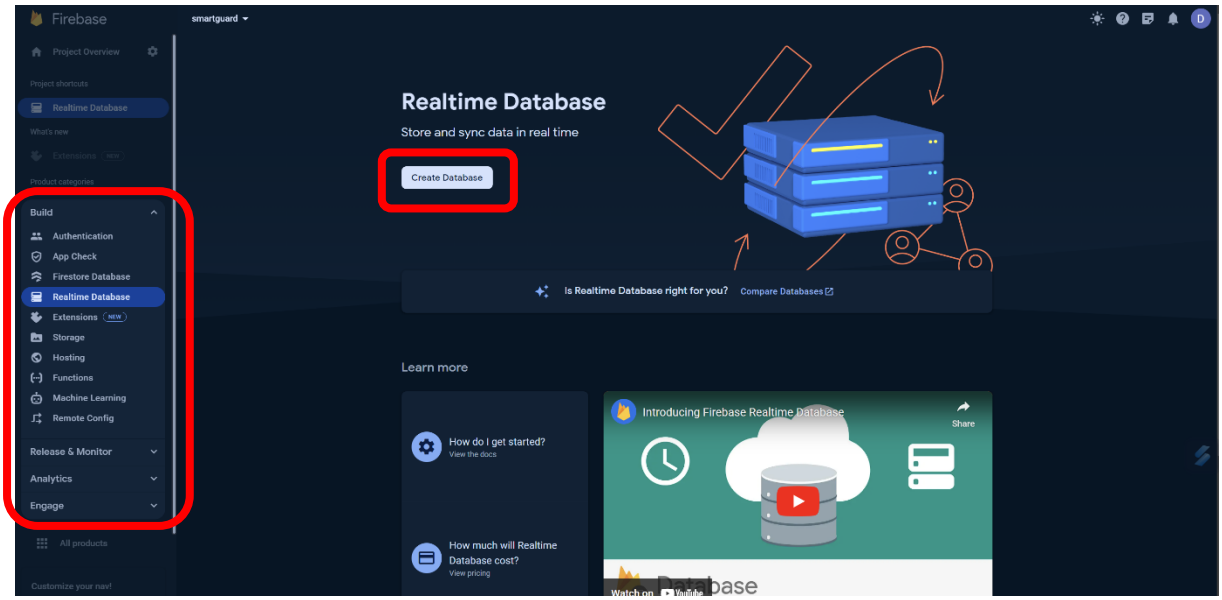
12. Once your project has been created, Click “Continue”.



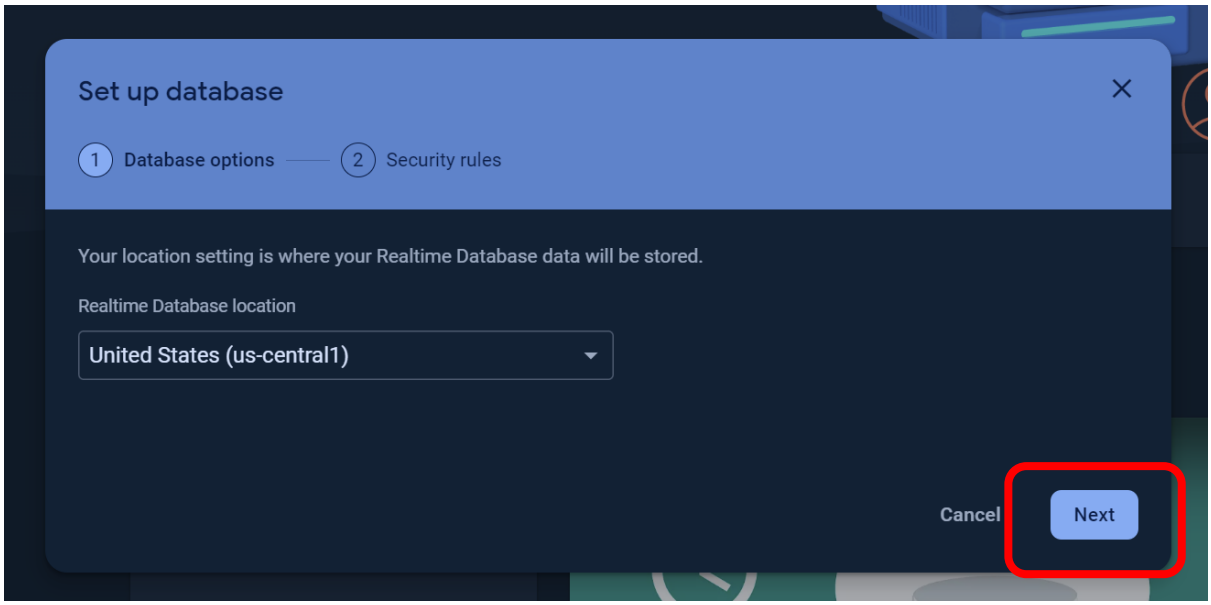
13. Navigate to the left-hand side menu of the webpage and click “Build”
14. Click “Realtime Database”



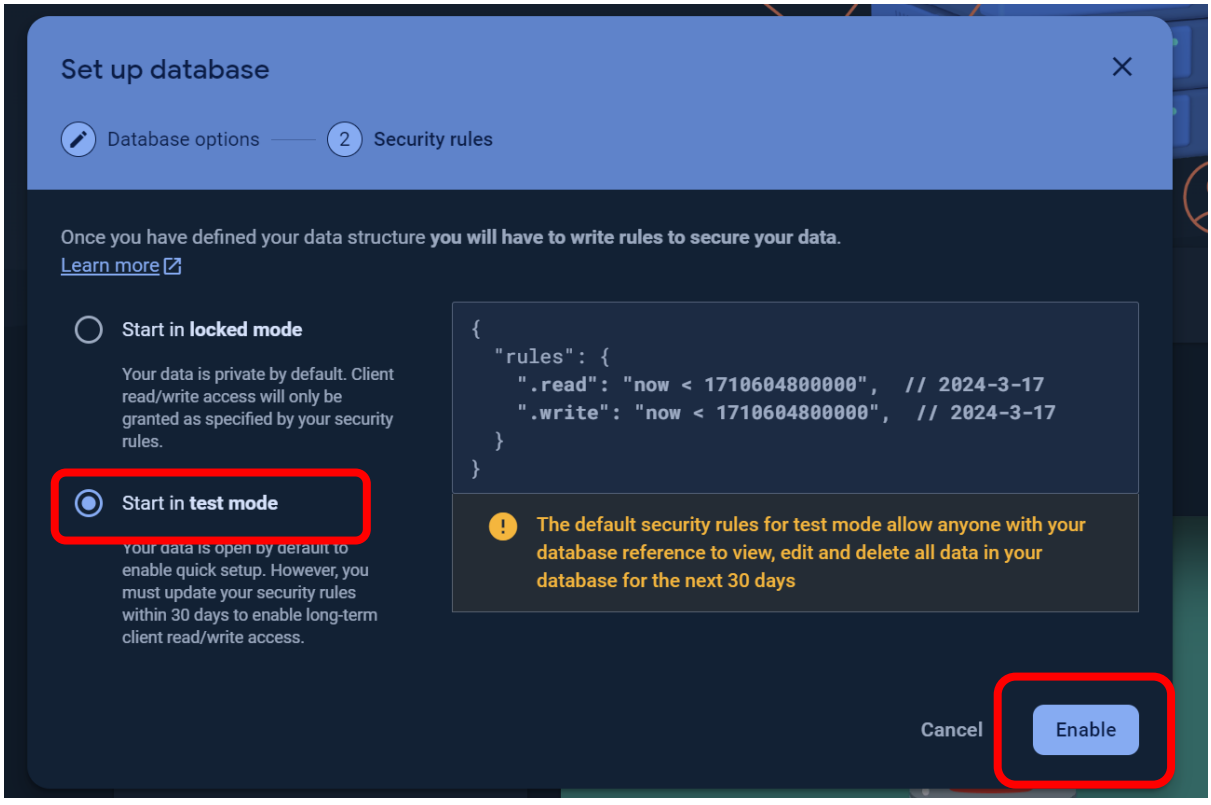
15. Click ‘Create Database”



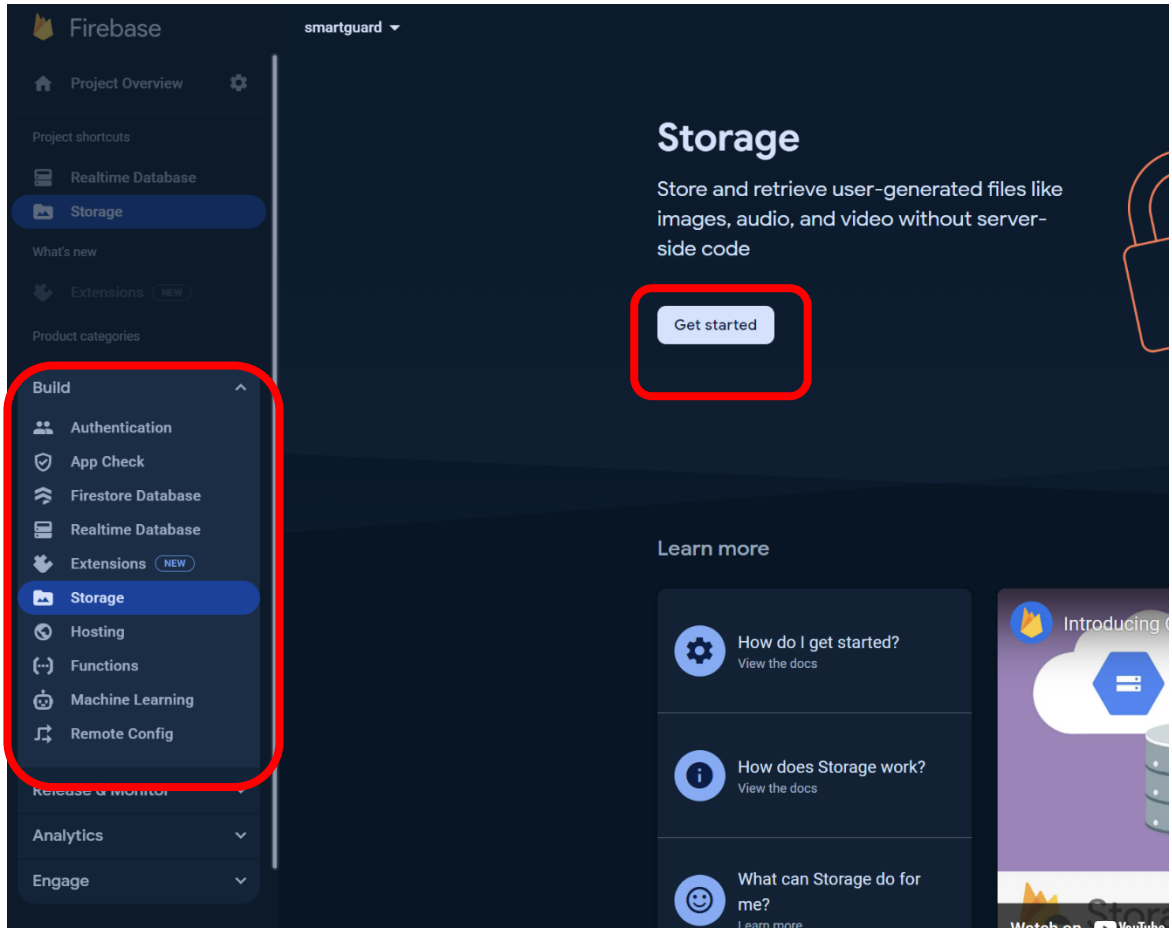
16. Use Default Selection for the Realtime Database location and Click “Next”.



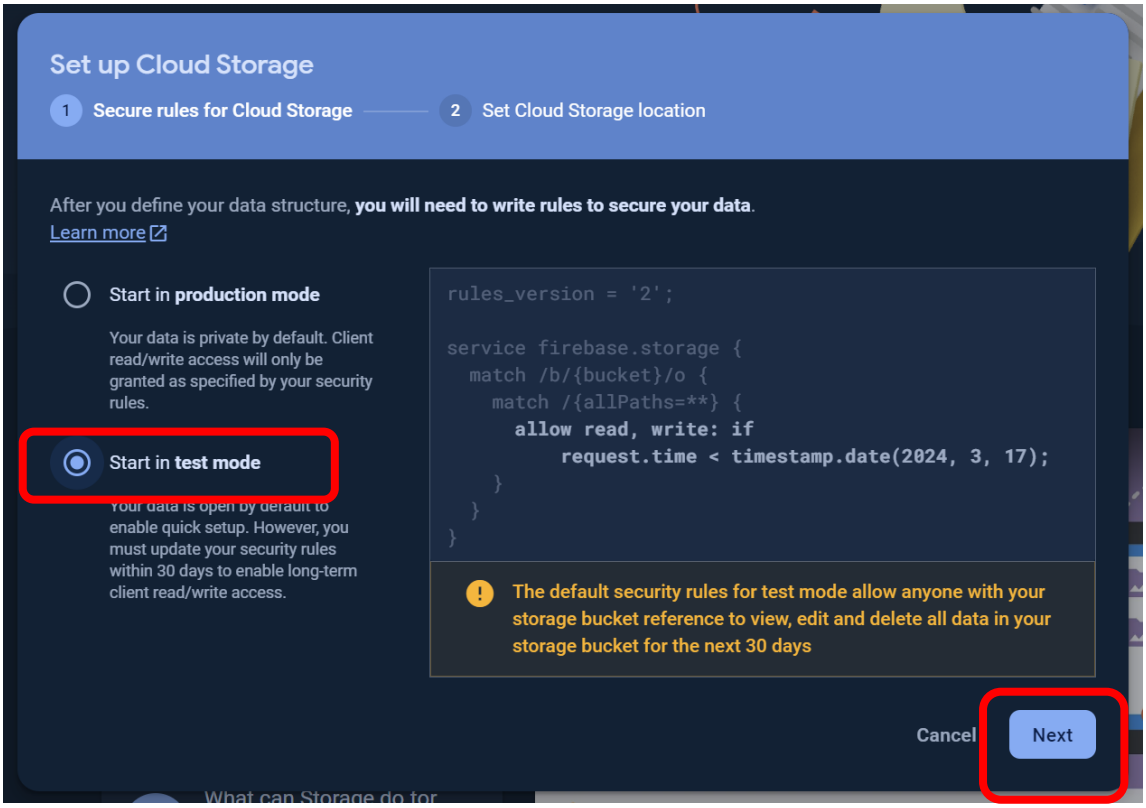
17. Under the security rules, Check “Start in test mode”.
18. Click Enable.



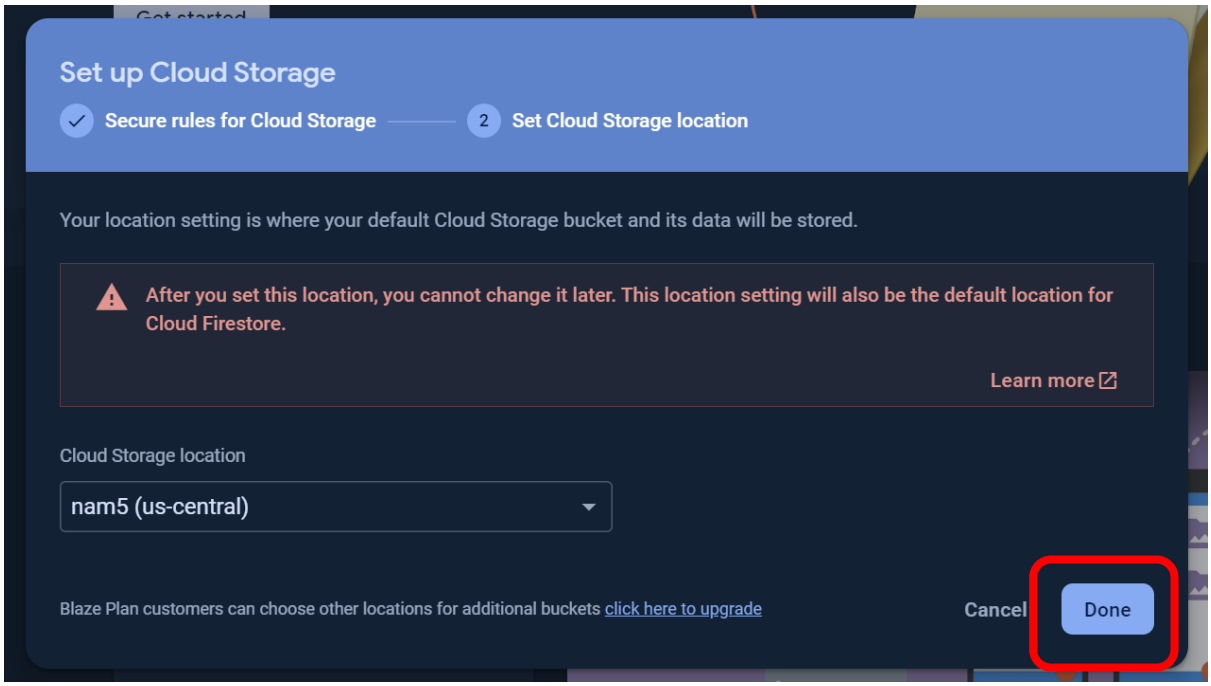
19. After Realtime Database is created, Navigate back to the menu located at the left side of the webpage
20. Click “Build”
21. A dropdown will appear after clicking “Build” and Click “Storage”
22. Click “Get Started”



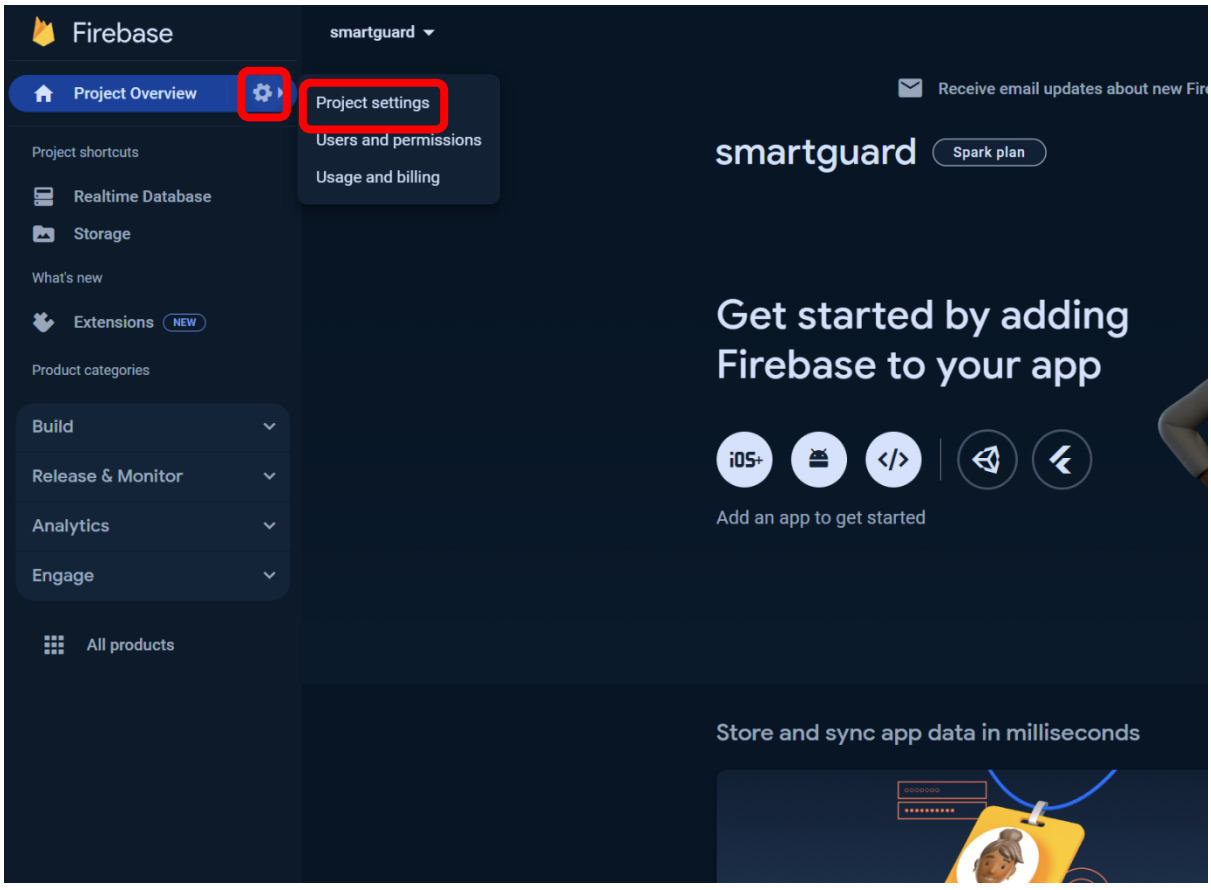
23. A popup appears after clicking “Get Started”, Check “Start in test mode”
24. Click “Next”



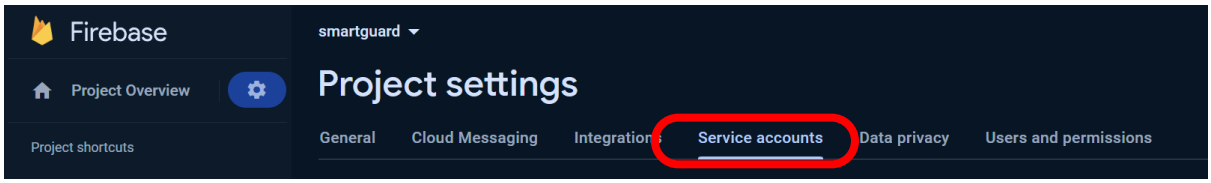
25. Use Default Selection for the Cloud Storage location and Click “Done”.



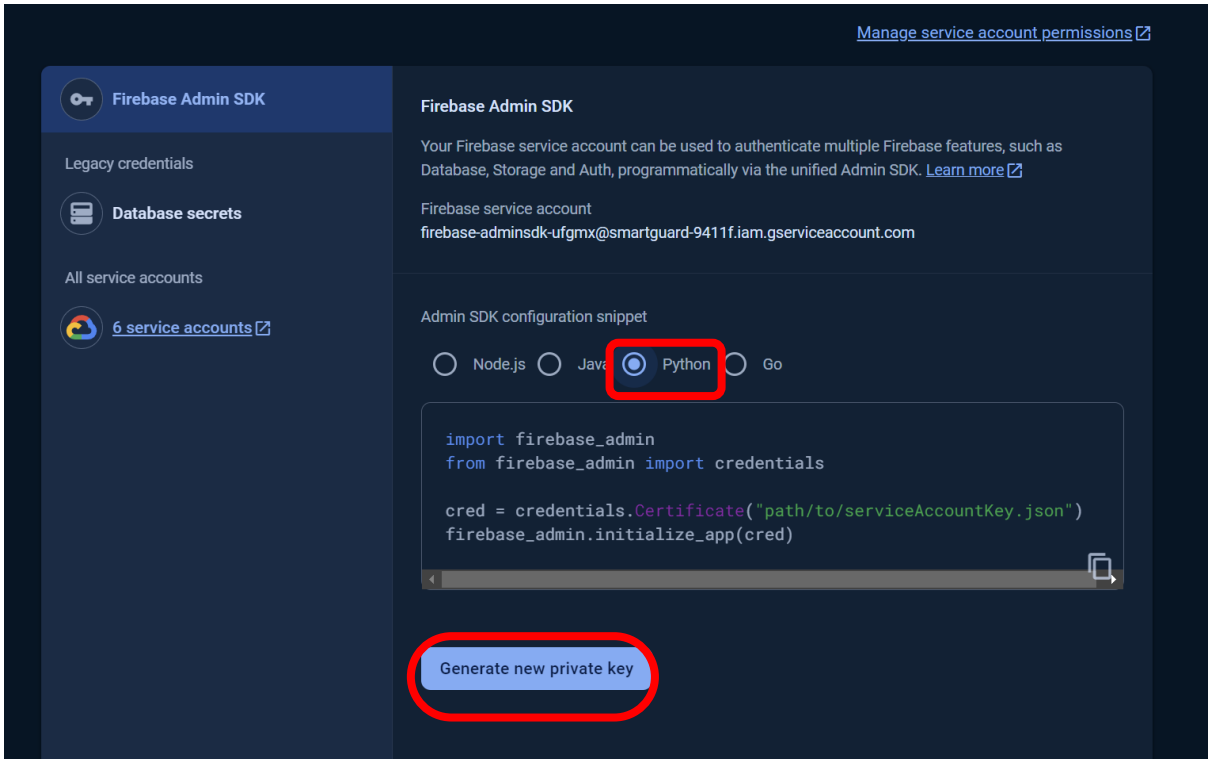
26. Once you have created the Cloud Storage, navigate back to the left side menu of the webpage and at the top of the menu --- “Project Overview”, Click the Settings Icon which looks like a Gear.
27. Click “Project Settings”.



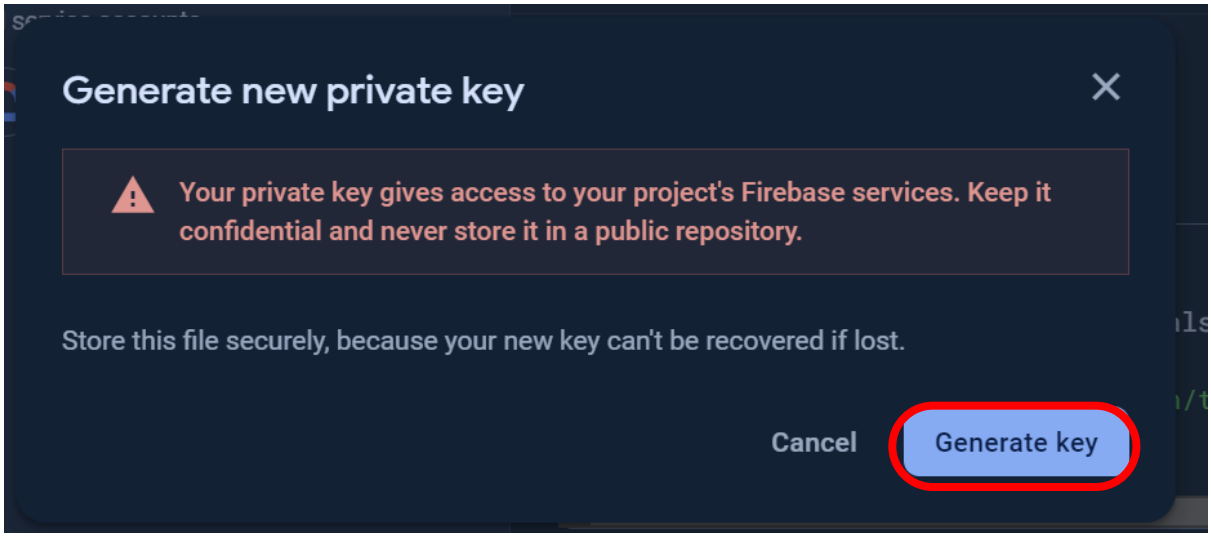
28. Click “Service Accounts”.



29. A popup will appear and Check “Python”.
30. Click “Generate new private key”.



31. A popup will appear, ignore the warning and Click ‘Generate Key’.



32. Once a new key is generated, copy the following highlighted code.

```
cred = credentials.Certificate("path/to/serviceAccountKey.json")
firebase_admin.initialize_app(cred)
```

33. The following copied code will be pasted in all 3 Python files in Pycharm (main.py, EncodeGenerator.py, AddDataToDatabase.py).

For main.py
In line 30, 31, replace “cred = credentials.Certificate("serviceAccountKey.json")” with the copied code.

```
# Firebase Error Check
try:
    cred = credentials.Certificate("serviceAccountKey.json")
    firebase_admin.initialize_app(cred, {
```

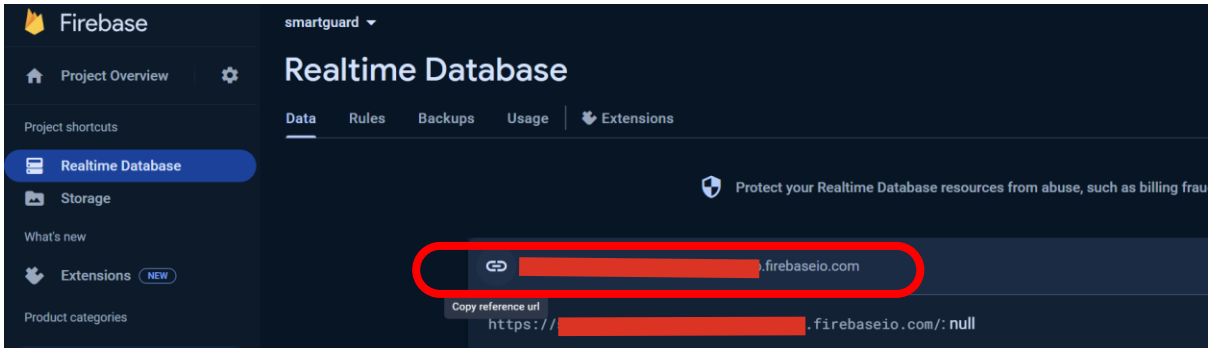
For AddDataToDatabase.py
In line 6, 7, replace “cred = credentials.Certificate("serviceAccountKey.json")” with the copied code.

```
def initialize_firebase():
    """Initializes Firebase application with given credentials and database URL."""
    cred = credentials.Certificate("serviceAccountKey.json")
    firebase_admin.initialize_app(cred, {
        'databaseURL': "https://[redacted].firebaseio.com/"
    })
```

For EncodeGenerator.py
In line 12, 13, replace “cred = credentials.Certificate("serviceAccountKey.json")” with the copied code.

```
def initialize_firebase():
    cred = credentials.Certificate("serviceAccountKey.json")
    firebase_admin.initialize_app(cred, {
        'databaseURL': "https://[redacted].firebaseio.com/",
        'storageBucket': "[redacted].appspot.com"
    })
```

34. Navigate back to Firebase.
35. Navigate back to the created Realtime Database.
36. Copy the database reference URL.



37. Once copied, use the copied link to replace the database URL link in the 3 Python files in Pycharm (main.py, EncodeGenerator.py, AddDataToDatabase.py).

For main.py
In line 32, replace https://attendancesystem-9a29a-default-rtdb.firebaseio.com/ with the copied reference link (your own firebase database link)

```
# Firebase Error Check
try:
    cred = credentials.Certificate("serviceAccountKey.json")
    firebase_admin.initialize_app(cred, {
        'databaseURL': "https://[redacted].firebaseio.com/",
        'storageBucket': "[redacted].appspot.com"
    })
    print("Firebase Admin SDK initialized successfully.")
except Exception as e:
    print(f"An error occurred during Firebase Admin SDK initialization: {e}")
```

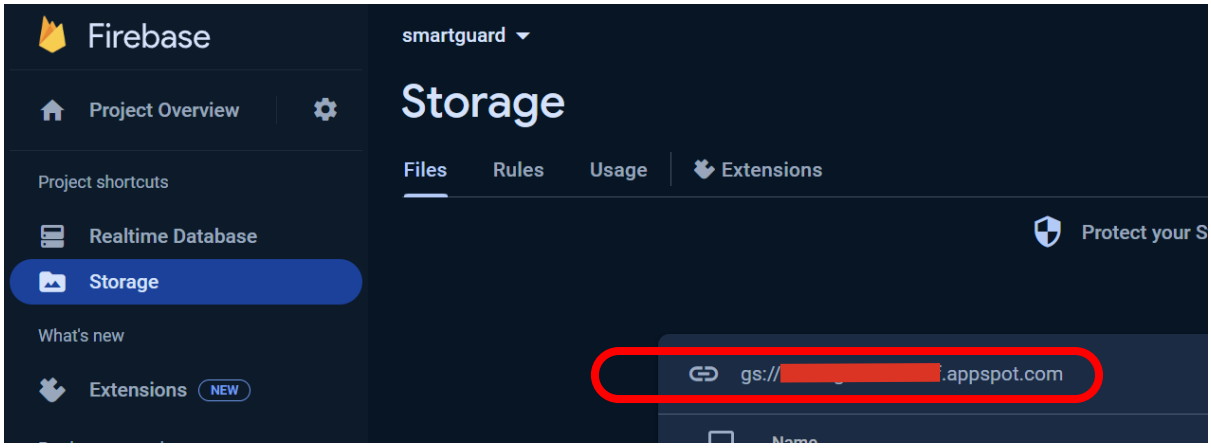
For AddDataToDatabase.py
In line 8, replace https://attendancesystem-9a29a-default-rtdb.firebaseio.com/ with the copied reference link (your own firebase database link)

```
def initialize_firebase():
    """Initializes Firebase application with given credentials and database URL."""
    cred = credentials.Certificate("serviceAccountKey.json")
    firebase_admin.initialize_app(cred, {
        'databaseURL': "https://[redacted]firebaseio.com/"
    })
```

For EncodeGenerator.py
In line 14, replace https://attendancesystem-9a29a-default-rtdb.firebaseio.com/ with the copied reference link (your own firebase database link)

```
def initialize_firebase():
    cred = credentials.Certificate("serviceAccountKey.json")
    firebase_admin.initialize_app(cred, {
        'databaseURL': "https://[redacted].firebaseio.com/",
        'storageBucket': "[redacted].appspot.com"
    })
```

38. Navigate back to Firebase.
39. Navigate back to the created Storage.
40. Copy the storage reference URL.



41. Once copied, use the copied link to replace the storage URL link in the 2 Python files in Pycharm (main.py, EncodeGenerator.py).

For main.py
In line 33, replace “attendancesystem-9a29a.appspot.com” with the copied reference link (your own storage database link)

```
# Firebase Error Check
try:
    cred = credentials.Certificate("serviceAccountKey.json")
    firebase_admin.initialize_app(cred, {
        'databaseURL': "https://[redacted].firebaseio.com/",
        'storageBucket': "[redacted].appspot.com"
    })
    print("Firebase Admin SDK initialized successfully.")
except Exception as e:
    print(f"An error occurred during Firebase Admin SDK initialization: {e}")
```

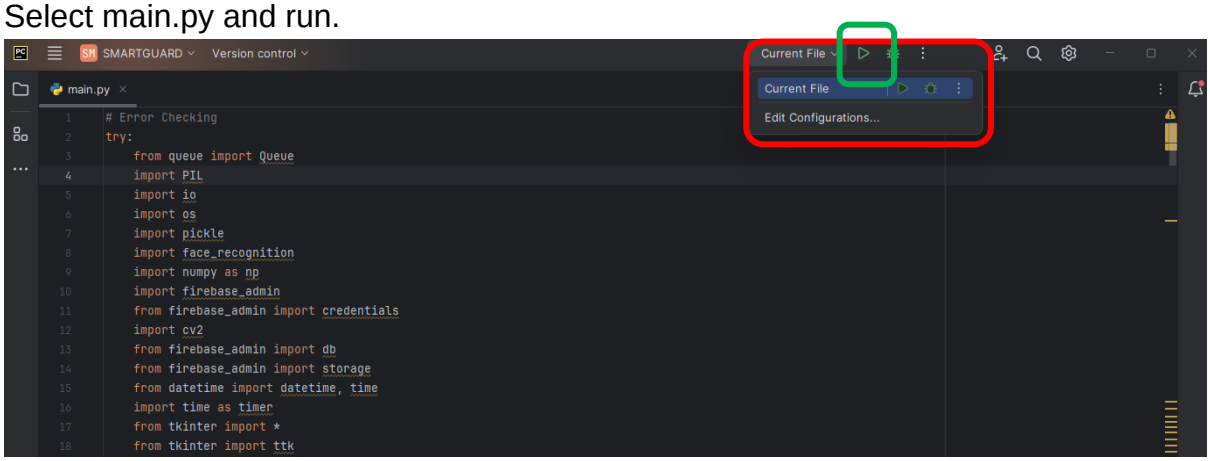
For EncodeGenerator.py
In line 15, replace https://attendancesystem-9a29a.appspot.com with the copied reference link (your own storage database link)

```
def initialize_firebase():
    cred = credentials.Certificate("serviceAccountKey.json")
    firebase_admin.initialize_app(cred, {
        'databaseURL': "https://[redacted].firebaseio.com/",
        'storageBucket': "[redacted].appspot.com"
    })
```

QuickStart Guide

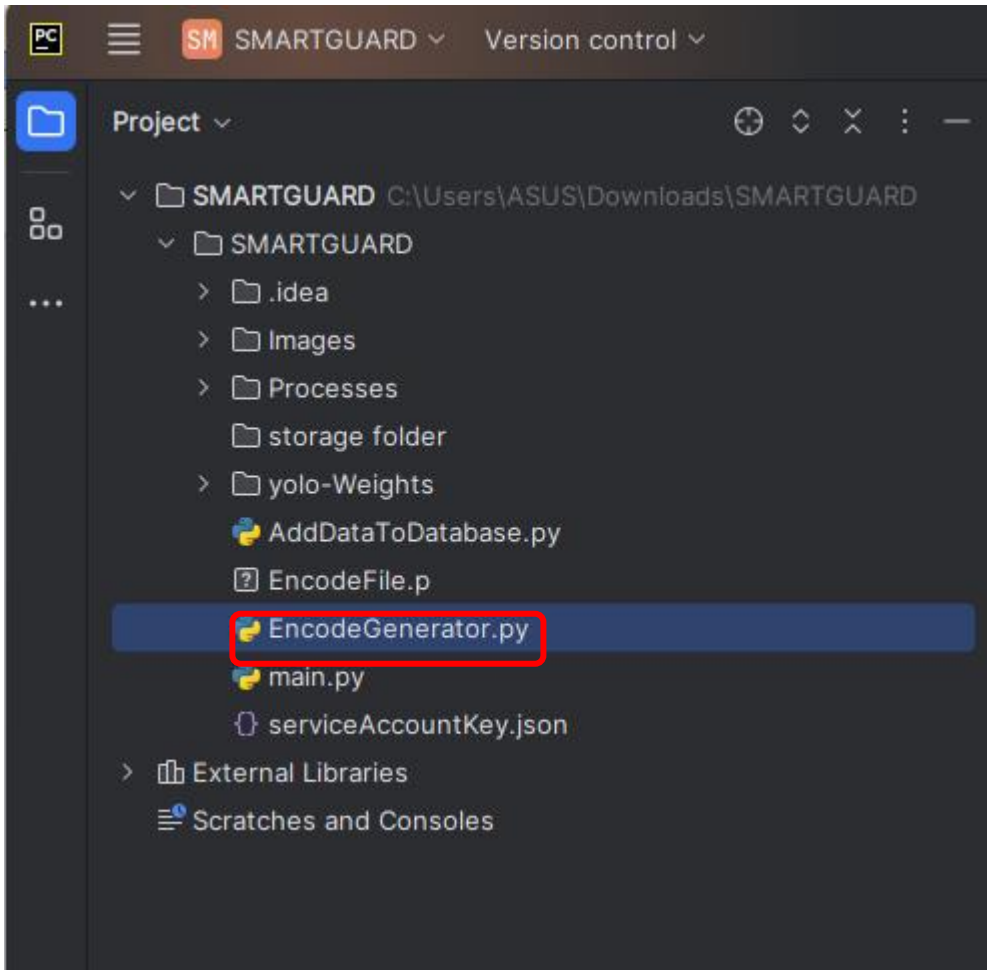
Run SMARTGUARD.py

Select main.py and run.



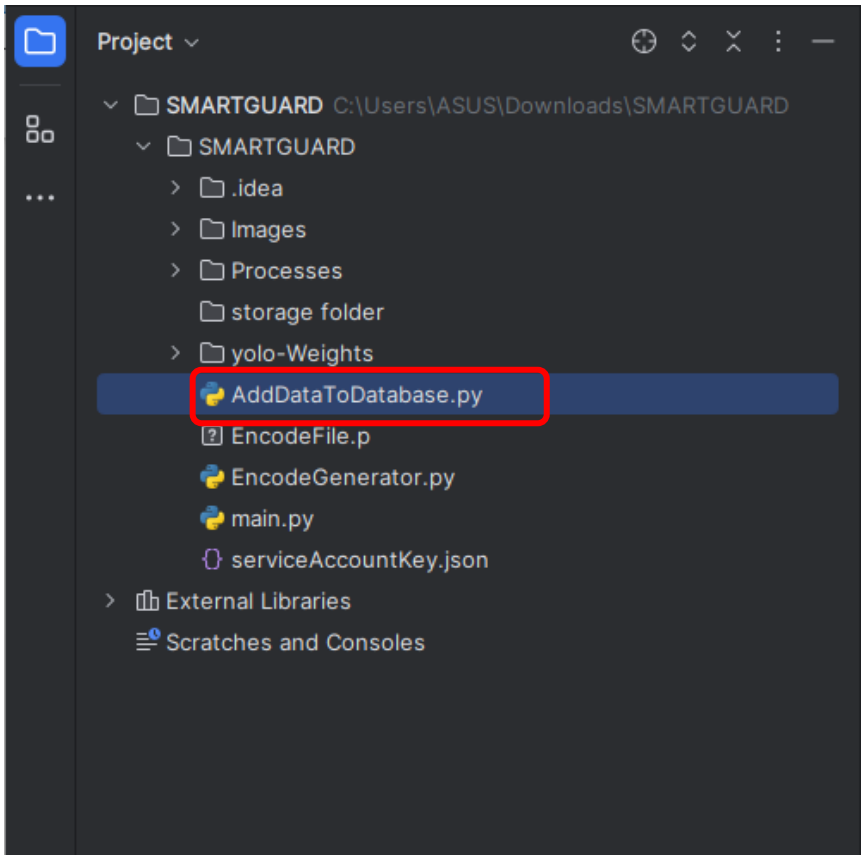
Run EncoderGenerator.py

Double Click to access the EncoderGenerator File, then follow the same method of running as [Run SMARTGUARD]



Run AddDataToDatabase.py

Double Click to access the AddDataToDatabase file, then follow the same method of running as Run SMARTGUARD



Configure automatic reset

1) Open main.py
2) Locate the following code:

```
56  # SETUP #
57
58  # set reset time
59  reset_time_of_day = time(18, 5)  # Reset at 2:00 AM
60  last_reset_date = None  # Track the last reset date
61
```

3) Modify the “time()” code to a time of your choice
(time(hours,minutes) in the 24 hour clock format, for example time(18,5) = 6.05pm as the reset time)

Training manual

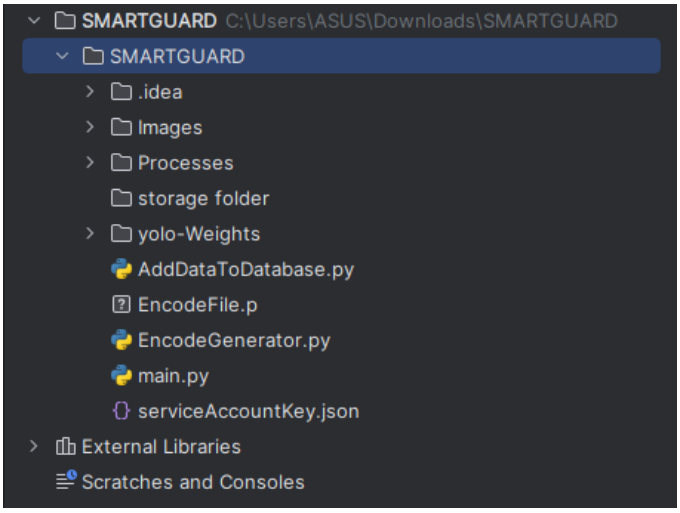
This training manual will include a step by step procedure on how to run the application from scratch/set it up yourself.

It will cover the following details:

4) Setup of supporting files
5) Setup of main file
6) GUI Interface explanation

Supporting Files Setup

You will need to set up and configure 2 files and process and save images to a folder before running the main application file. The reason is to upload image data, add the image data to the database, and process the face recognition encoder. It applies both to startup and in future updates.



1. Upload image to “Images” Folder
2. AddDataToDatabase.py
3. Encodegenerator.py

Upload images to “Images” Folder

Step 1: Procure a suitable front-facing picture from your employee. Then save it at a location. Remember where you saved the image as it will be used. Refer to the images below on what is suitable and unsuitable.

Suitable Images are images that have the following properties:

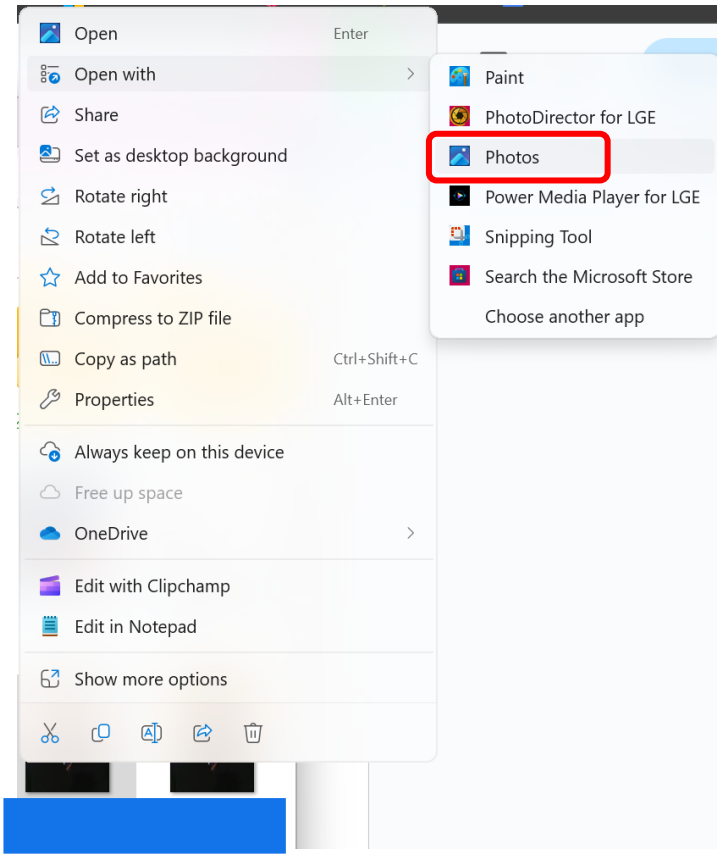
7) Front Facing Self Portraits
8) Face at centre of image
9) Clean Images
10) No image modifications like bitmojis or stickers

Unsuitable Images are images that have the following properties:

1) Images that have been passed through filters or warped (top left)
2) Images with an object blocking any facial features (top right)
3) Super zoomed/cropped images (bottom left)
4) Images with stickers or bitmojis (bottom right)

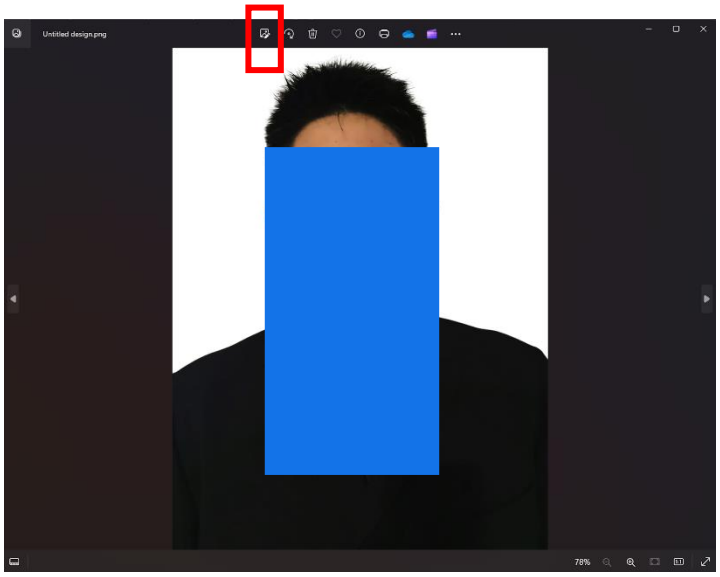
Step 2: Open Image Editing Software

Once the image is acquired, find where the image is saved on your local device and right click on the file. Then, open with Photos.

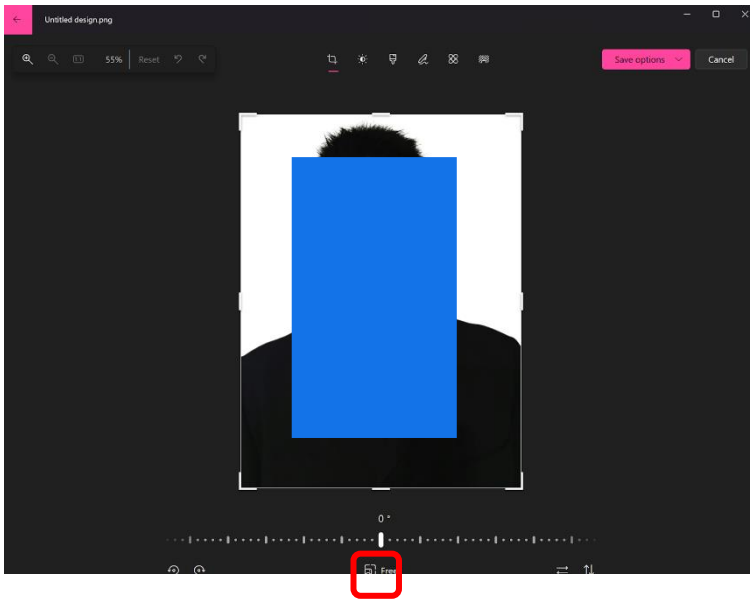


Step 3: Image Resizing

Once the image has been opened in Photos, click the edit image button. Refer to the image below to find the location of the image editing tool.

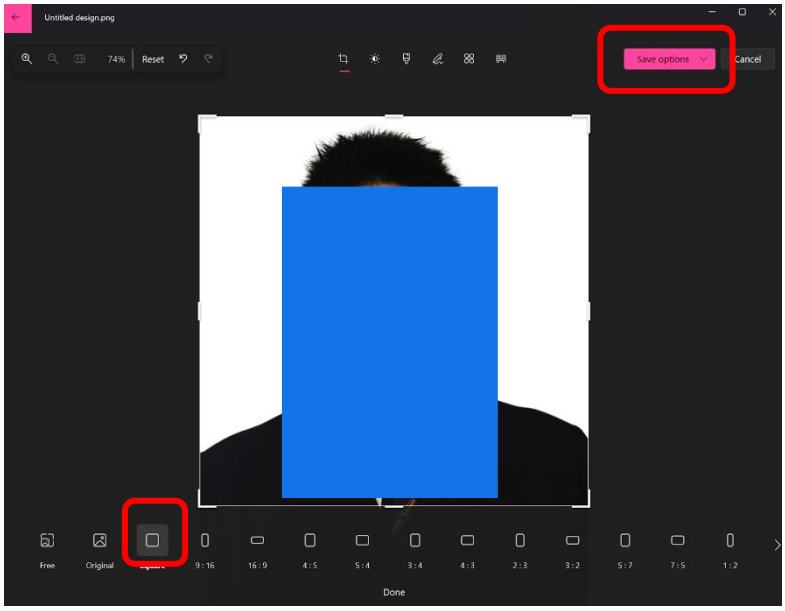


Next, click the resizing option at the bottom middle of the image. Refer to the image belowto find the location of the image editing tool.

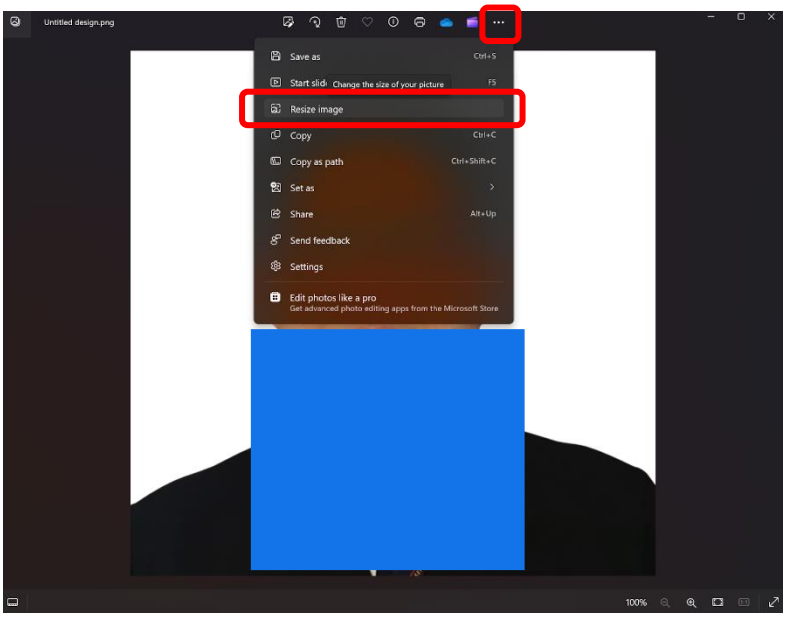


Then click the square option and align the face to the centre of the square as much as possible.

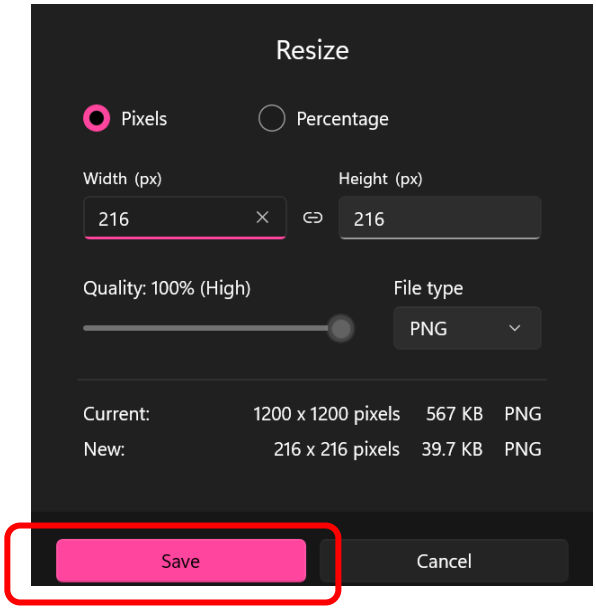
After that, click “Save Options” and select the option “Save”. Refer to the image below



Now, click the 3 dots and select the option “Resize image”.



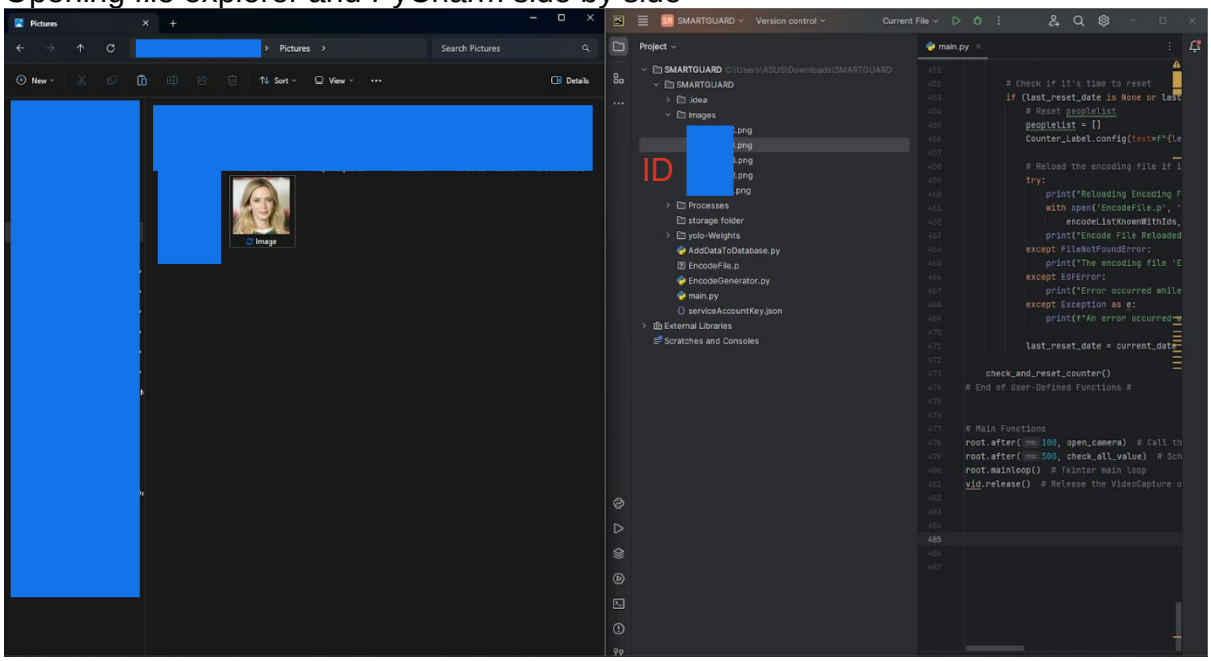
In the pop-up, select the pixels option, and change the width and height to 216px as shown, then click save.



Step 4: Relocating & renaming the image.

To upload the image to the “Images” folder easily, locate the image in File Explorer and drag the image to the folder in PyCharm. In the pop-up in PyCharm, click refactor. After the image has been successfully moved, right click the image, and go to refactor, and click rename. Then, rename the image to the worker ID of your choice. Refer to the next 4 images for the process.

Opening file explorer and PyCharm side by side



Click refactor to move the image to the folder

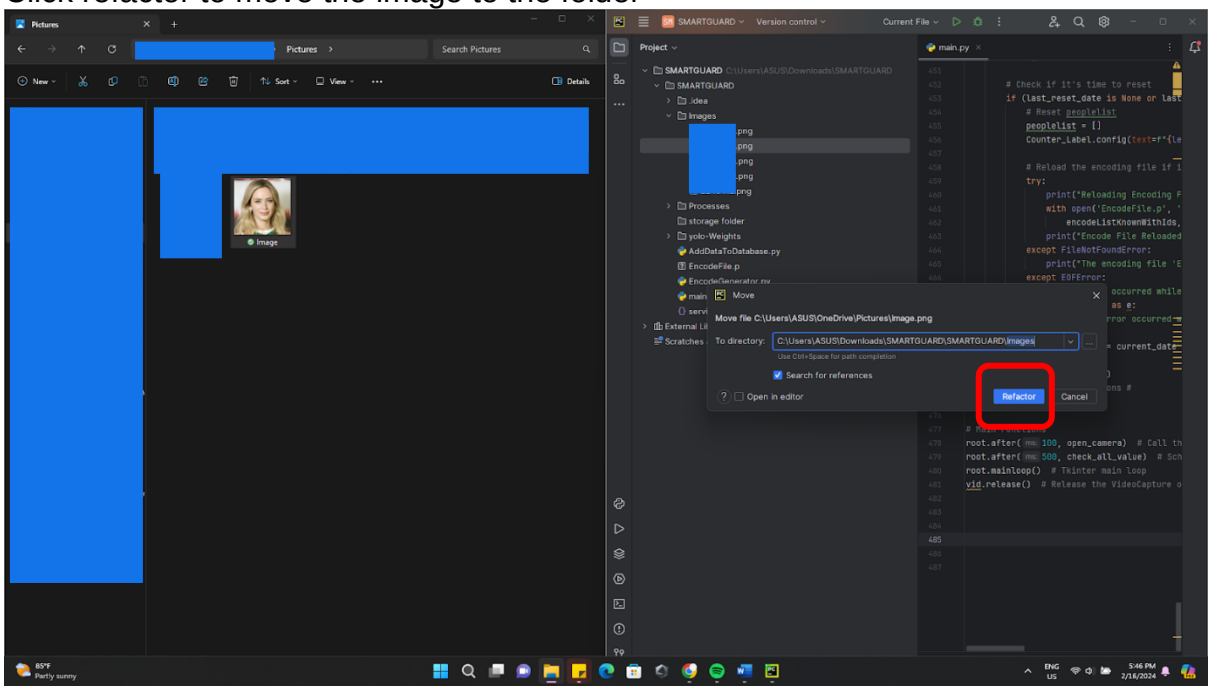
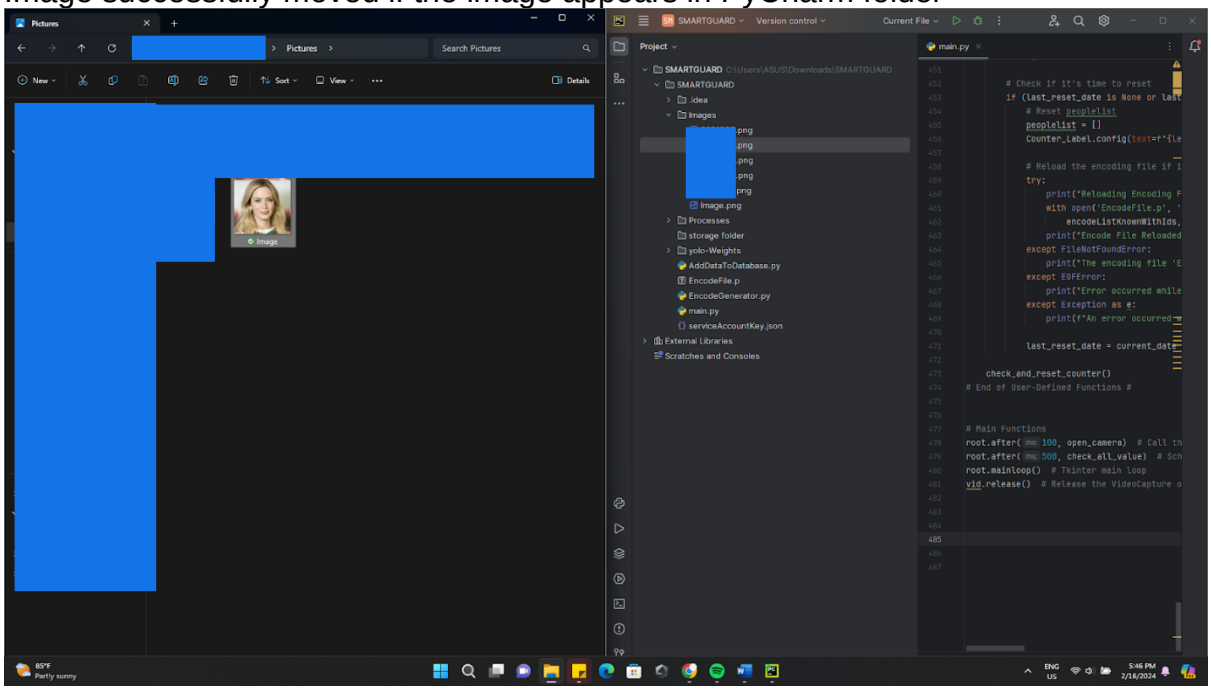
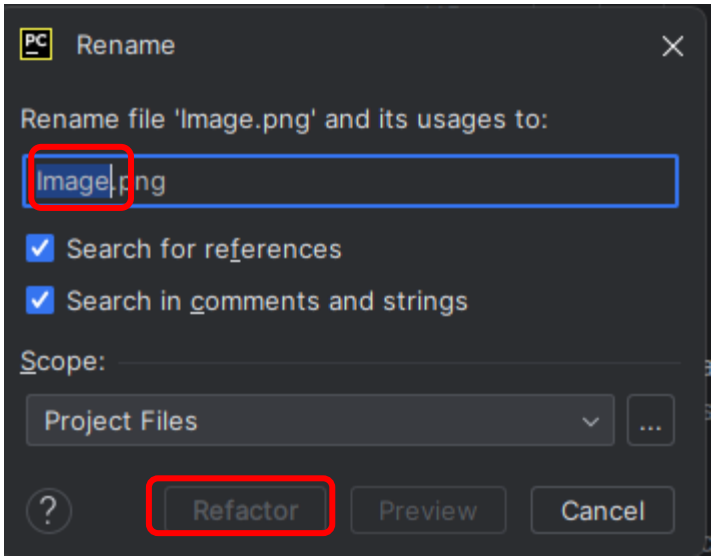


Image successfully moved if the image appears in PyCharm folder



After right clicking the image in python, refactor > rename and change it to a worker id of your choice.



Replace “Image” to the worker ID of your choice (example: 222983R). Then click refactor.

Using AddDataToDatabase.py to add worker information:

Step 1: Open the file in PyCharm

Step 2: Locate the following Code in the file:

```python
def initialize_firebase():
    """Initializes Firebase application with given credentials and database URL."""
    cred = credentials.Certificate("serviceAccountKey.json")
    firebase_admin.initialize_app(cred, {
        'databaseURL': "insert your DB link"
    })

def update_employee_data(employee_data):
    """Updates employee data in the Firebase Realtime Database."""
    ref = db.reference('Employee')
    for key, value in employee_data.items():
        ref.child(key).set(value)
        print(f"Updated data for {value['Name']}")

if __name__ == "__main__":
    # To add data, follow the format below
    data = {
        "Worker 1 ID": {
            "Name": "Worker 1 Name",
            "Employee ID": "Worker 1 ID",
            "Position": "Worker 1 Position"
        },
        "Worker 2 ID": {
                "Name": "Worker 2 Name",
                "Employee ID": "Worker 2 ID",
                "Position": "Worker 2 Position"
        },
    }

    # Initialize Firebase and update employee data
    initialize_firebase()
    update_employee_data(data)
```

Step 3: Replace the database link with your own that you created.

Step 4: To update, replace the following fields in data with your own workers names and ID and position.

Step 5: Run the python file by clicking in the code space and use the keyboard shortcut Shift+F10

Create the encoding file

Step 1: Open EncodeGenerator.py

Step 2: Check if the database URL and storage bucket are your keys and endpoints

```
def initialize_firebase():
    cred = credentials.Certificate("serviceAccountKey.json")
    firebase_admin.initialize_app(cred, {
        'databaseURL': "https://[redacted].firebaseio.com/",
        'storageBucket': "[redacted].appspot.com"
    })
```

Step 3: Run the encoding file

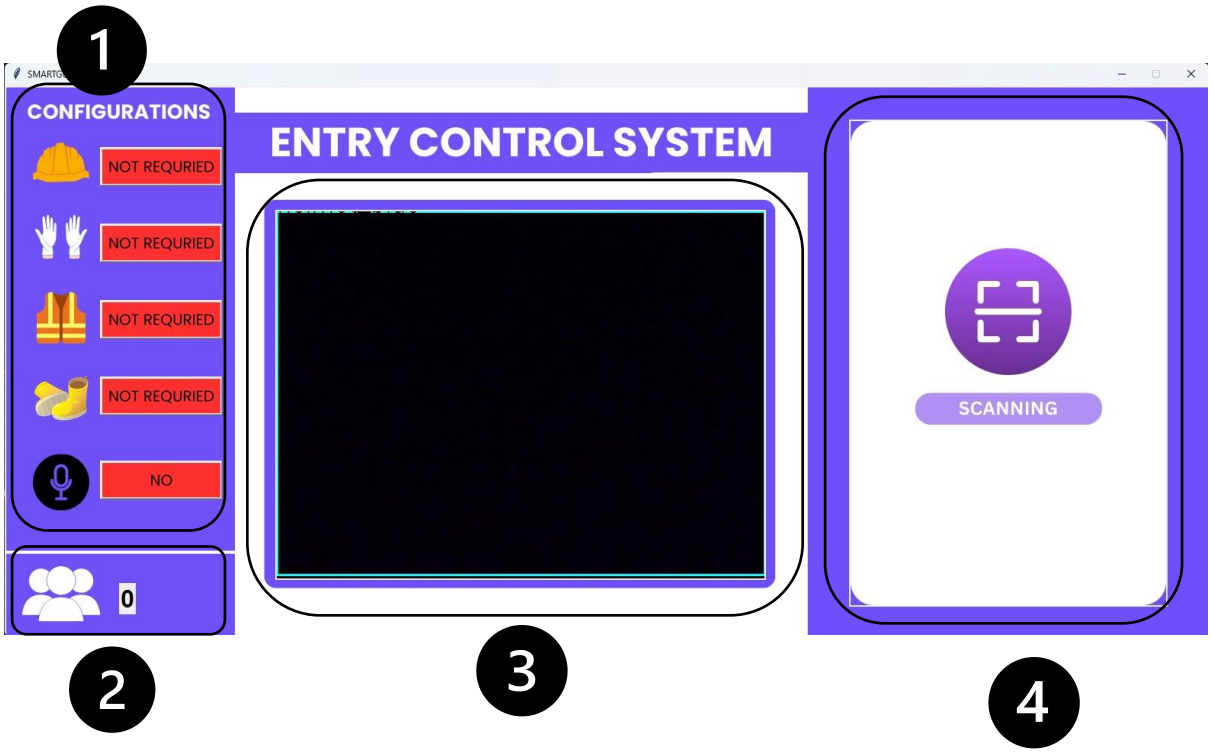
Step 4: File labelled “EncodeFile.p” will be created. **Do not touch this file**

Run SMARTGUARD

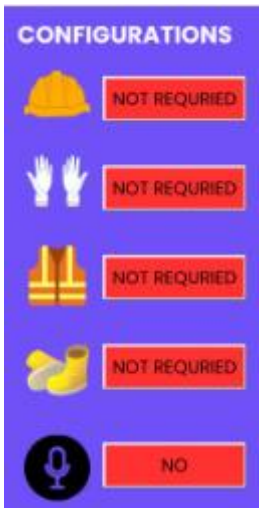
Step 1: Access the main.py file

Step 2: Run the main.py file (Refer to Quickstart Guide)

Understanding the GUI



Configuration:



Purpose:

To enhance flexibility and adaptability in different operational contexts, we have introduced a configuration section within our application. This feature allows users to tailor the required Personal Protective Equipment (PPE) needed for access authorization, where the user can go through each equipment and determine if it is a criteria that is needed for their environment. This customization capability ensures our object detection application can be effectively utilized across a broad range of scenarios, providing a versatile solution that ensures safety and compliance.

Components

Helmet Button : This feature allows for the activation or deactivation of the helmet requirement. When enabled the button would change in the color green while saying required and furthermore, it will be mandatory that the user must not only be registered in the database, but also be wearing a helmet to be granted access. If the user does meet the criteria and fails to wear a helmet when this button is enabled, the model will notify the user that access is denied due to the absence of helmet, despite their registration in the database.

Gloves Button : This feature allows for the activation or deactivation of the gloves requirement. When enabled the button would change in the color green while saying required and furthermore, it will be mandatory that the user must not only be registered in the database, but also be wearing gloves to be granted access. If the user does meet the criteria and fails to wear gloves when this button is enabled, the model will notify the user that access is denied due to the absence of gloves, despite their registration in the database.

Vest Button : This feature allows for the activation or deactivation of the safety vest requirement. When enabled the button would change in the color green while saying required and furthermore, it will be mandatory that the user must not only be registered in the database, but also be wearing a safety vest to be granted access. If the user does meet the criteria and fails to wear a safety vest when this button is enabled, the model will notify the user that access is denied due to the absence of safety vest, despite their registration in the database.

Safety Shoes Button: This feature allows for the activation or deactivation of the boots requirement. When enabled the button would change in the colour green while saying required and furthermore, it will be mandatory that the user must not only be registered in the database, but also be wearing boots to be granted access. If the user does meet the criteria and fails to wear boots when this button is enabled, the model will notify the user that access is denied due to the absence of boots, despite their registration in the database.

Voice Assistant: This button enables users to toggle the voice assistant feature. Upon activation, it initiates a text to speech function. For example, if button 1 is pressed to enforce the helmet requirement, it will announce “Helmet detection is enabled. Additionally, if the action is to remove the requirement, it will state “Helmet detection is disabled.

People count:

This section is designed to record and track the number of users that were granted access, maintaining a total count of all individuals that had entered. Additionally, there is a customizable feature that has been integrated to reset this count at intervals and the parameters for this reset function can be adjusted within the code, as detailed in the following picture.

Setting up automatic update timings

The automatic update affects 2 things: Worker Count Reset and reading the updated encoder file if any.

```
56  # SETUP #
57
58  # set reset time
59  reset_time_of_day = time(18, 5)  # Reset at 2:00 AM
60  last_reset_date = None  # Track the last reset date
61
```

Configure the timing here. (Refer to Quick Start guide on how to configure)

Camera

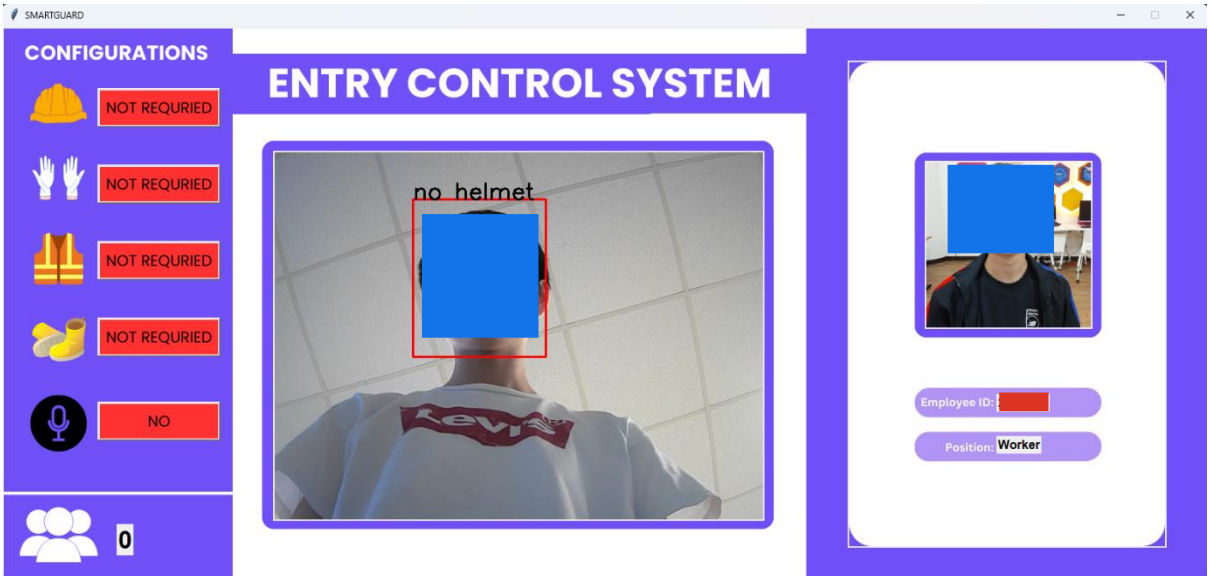
This section displays the feed from the camera whether it is a webcam or the user's device, the function of it is to perform the various functions that were mentioned above, and ultimately determining access for a person.

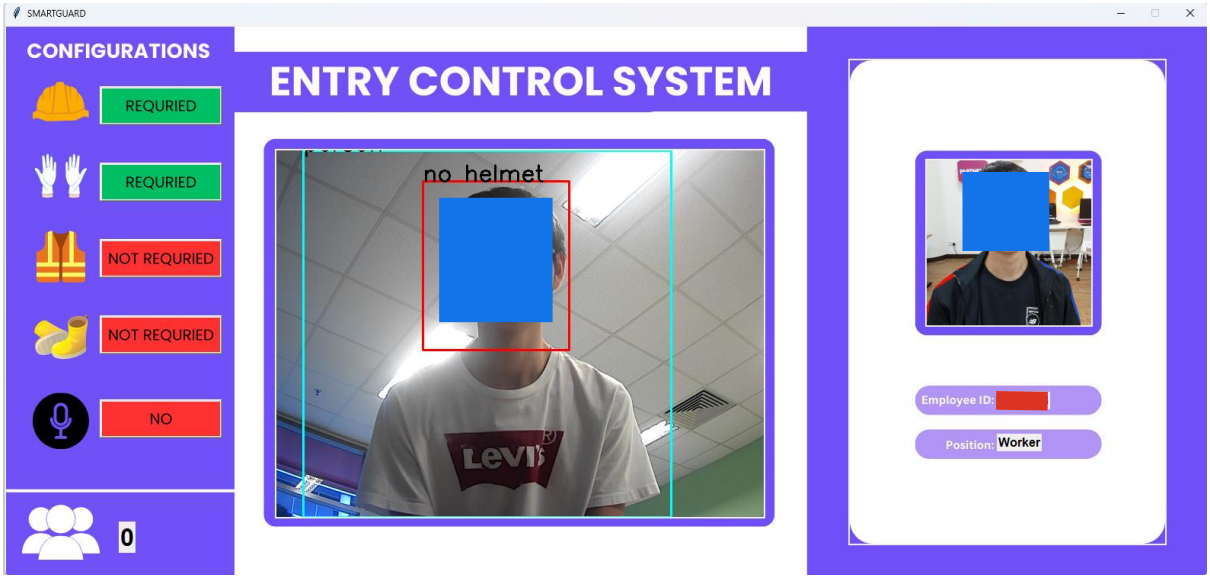
Scanning page

This section indicates whether a user is granted or denied access based on the predefined configuration and the user's registration in the firebase, there are the various scenarios and outcomes.

The user has been registered in the database.

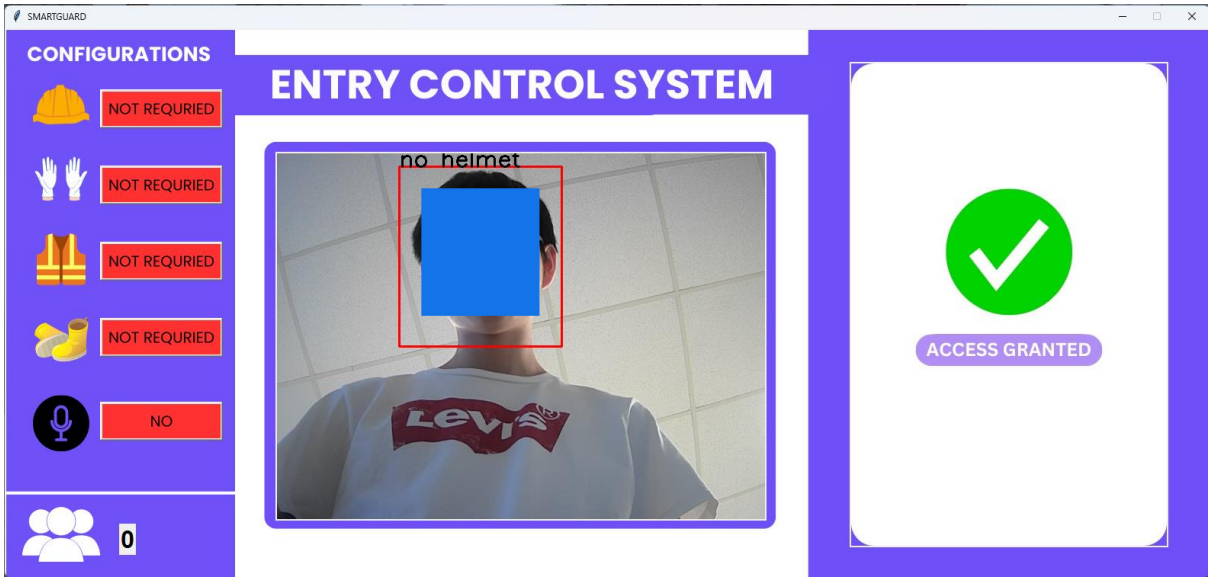
- The system extracts and displays the IDs, image, and role of the worker, alongside announcing the name of the detected employee. This action occurs even if the condition criteria that the user set is not met.





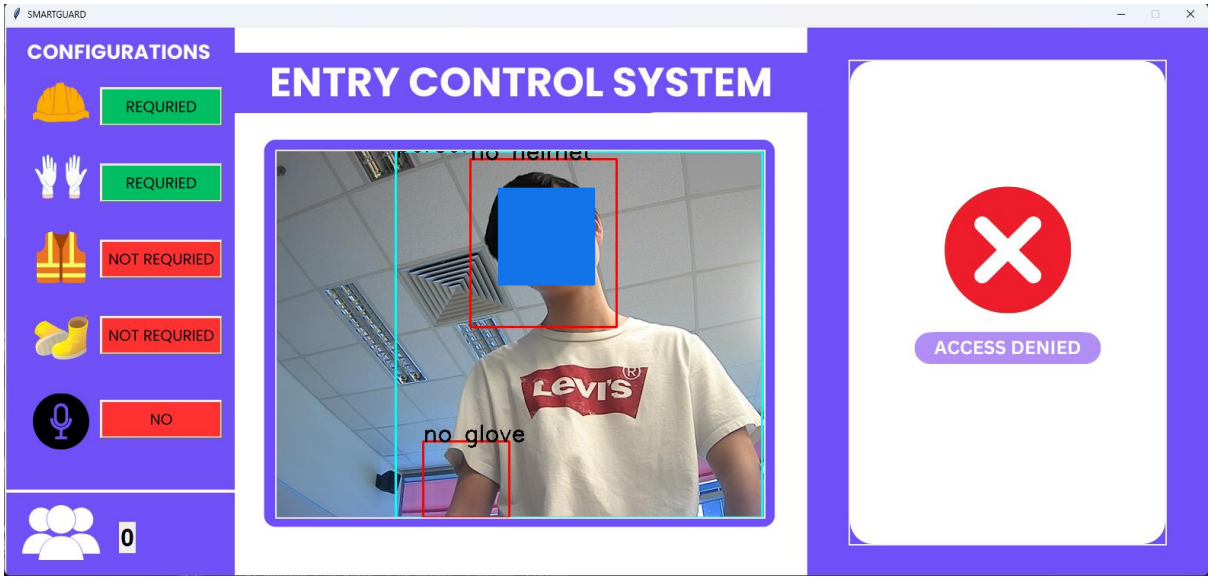
The registered user fulfils the criteria that is set by the configurations.

- Access granted will be displayed accompanied by an audio announcement stating that the user has been granted access.



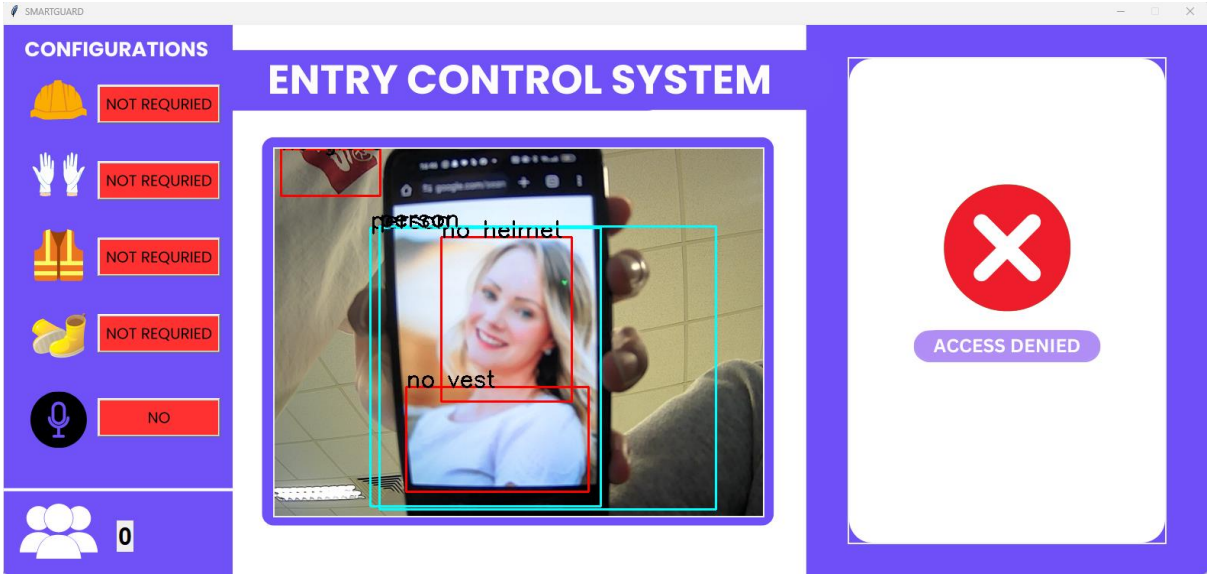
The registered user does not fulfil the criteria set by the configurations.

- Access denied will be displayed while announcing that the user has been denied access. Additionally, it will specify the personal protective equipment (PPE) that the user is missing that resulted in the denied access.



The user is not registered in the database.

- Access denied will be displayed while announcing that unauthorized personnel has been detected.



General FAQ:

1) Is there an image format requirement?

Yes there is, worker images should all be in .png format. Please use image converters to convert it into the proper image format. Else the program may return errors

2) What if I do not have an inbuilt webcam?

For Webcam, you can attach an external USB camera and the code will run as per normal

In the event you want to run an external webcam instead of the inbuilt webcam, replace the vid variable with the following code “cv2.VideoCapture(1)”

3) Do I have to worry about duplicate images in the Firebase/Database?

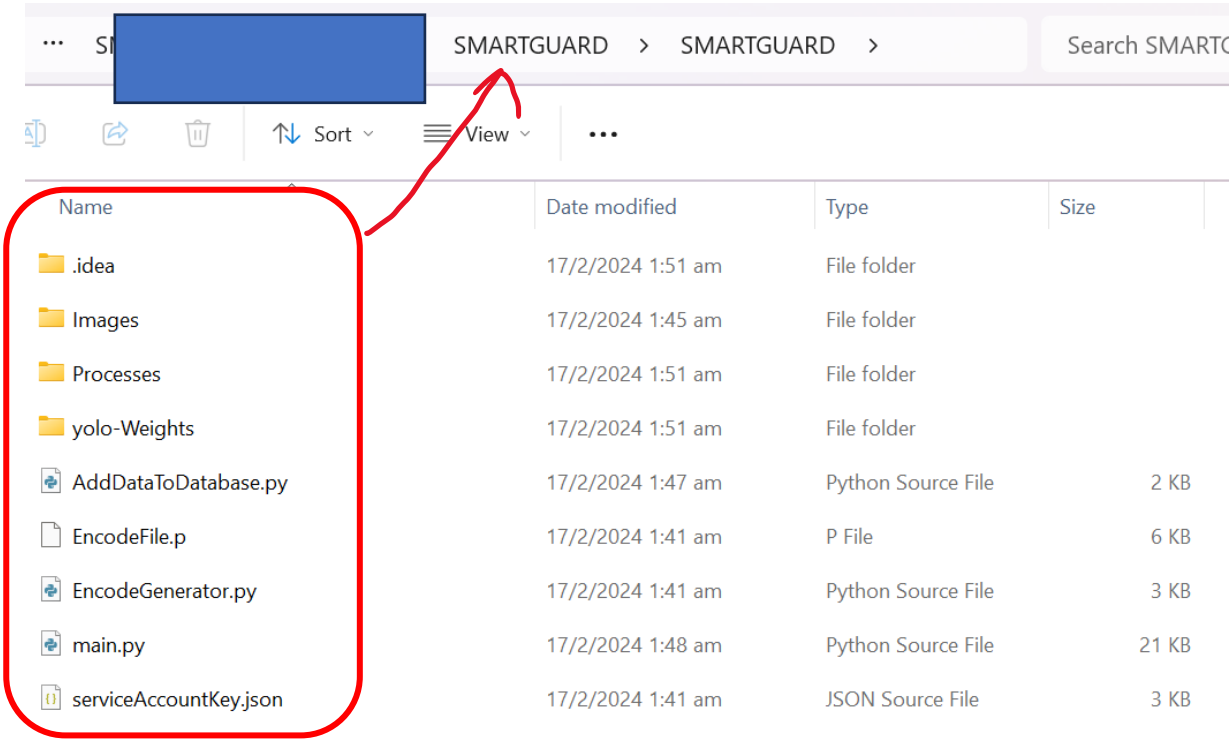
No you do not, images will not be duplicated in the databases even if you try to upload the same picture **ONLY** under the condition that you are running the “AddDataToDatabase.py” file

4) Will the environment affect the application?

Yes it will. Preferably the camera is set up at a place with sufficient lighting, however there must not be a light source shining directly into the camera, else it will result in an inability to accurately detect objects or people.

5) Why is it unable to retrieve the images/weights?

In the event this error occurs, please follow the following steps:



Select all the files, then use the keyboard shortcut Ctrl+X to cut all the files. Then navigate to the folder shown in the image above by the arrow and then Ctrl+V to paste all the files, before deleting the empty SMARTGUARD folder by clicking on it once and pressing “delete” on your keyboard(shown below)