Facial Recognition System : Tools and development phases

Submitted by:
Roaya Neffeti
Yessine Barkia

Programming language :

This project was mainly focused on Python Programming and its libraries

Libraries and frameworks :

The face detection module will use **OpenCV** library for implementation by use of the frontal Haar Cascade face detector in either.

Tools that will be used in the solution:

- **Python** programming language
- **Tkinter** a standard GUI toolkit for Python programming language. We'll use this to build the user interface.
- **OpenCV** library for computer vision tasks, it is an open-source machine learning and computer vision library. it's a cross-platform library and is free to use
- **face_recognition library** for facial recognition ; a Python package for facial recognition tasks. It provides an easy-to-use interface for detecting and recognizing faces in images and videos
  (github link: ageitgey/face_recognition)

- **Silent-Face-Anti-Spoofing** silent face anti-spoofing detection technology is to judge whether the face in front of the machine is real or fake.
  (github link: https://github.com/computervisioneng/Silent-Face-Anti-Spoofing.git)

- **pickle library** for object serialization
- **datetime and time libraries** for handling time and date data
- **os library** for interacting with the operating system

→ The code we will be proposing uses a pre-trained face recognition model to perform facial recognition, and saves the detected faces in a database using pickle serialization. Additionally, the code provides an API to log attendance by recognizing faces in uploaded images.

Pre-trained face recognition model: The code uses a pre-trained face recognition model to perform facial recognition. This means that the model has been trained on a large dataset of images to be able to recognize faces with a high degree of accuracy. In the provided code, the face recognition model is from the **face_recognition library**.

Saving detected faces in a database using **pickle serialization**: After detecting faces, the code saves them in a database using the pickle serialization technique.
→pickle serialization : This means that the faces are converted into a stream of bytes and then written to a file using the pickle library. This allows the faces to be easily stored and retrieved later without having to recreate the face recognition model each time
→Pickle is a Python module that allows objects to be serialized and deserialized, which means that the objects can be converted into a format that can be stored in a file and retrieved later.

Face detection tools : (Haar cascade classifier)

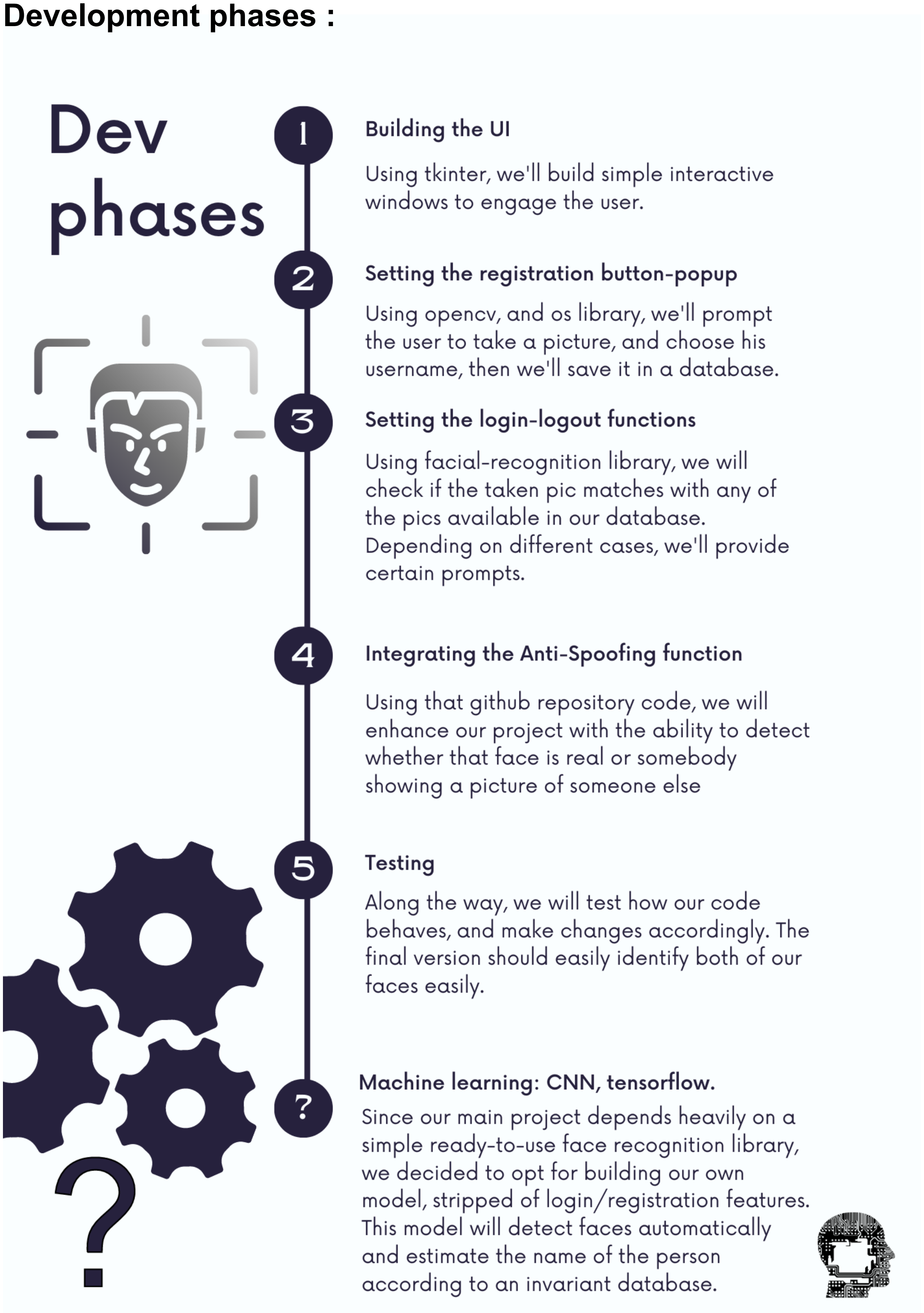
Haar Cascade Classifier: a machine learning-based approach for object detection used in OpenCV. It uses a set of positive and negative images to train a classifier that can detect the presence of an object, such as a face, in an image

Image classification tools (CNN)

The CNN-based model is going to be built to detect faces from different angles and under varying conditions to achieve more accurate results

- TensorFlow : deep learning library used to build convolutional neural networks (CNNs).
- Keras: a high-level deep learning API that can be used to build and train CNNs quickly and easily.
- NumPy: a fundamental library for scientific computing in Python and can be used to handle multi-dimensional arrays required for image processing tasks.
- OpenCV
- Image data: we will be collecting free images of celebrities + our own pictures to form the database that will be used to train the cnn model

Development phases :



Dev phases

1 **Building the UI**

Using tkinter, we'll build simple interactive windows to engage the user.

2 **Setting the registration button-popup**

Using opencv, and os library, we'll prompt the user to take a picture, and choose his username, then we'll save it in a database.

3 **Setting the login-logout functions**

Using facial-recognition library, we will check if the taken pic matches with any of the pics available in our database. Depending on different cases, we'll provide certain prompts.

4 **Integrating the Anti-Spoofing function**

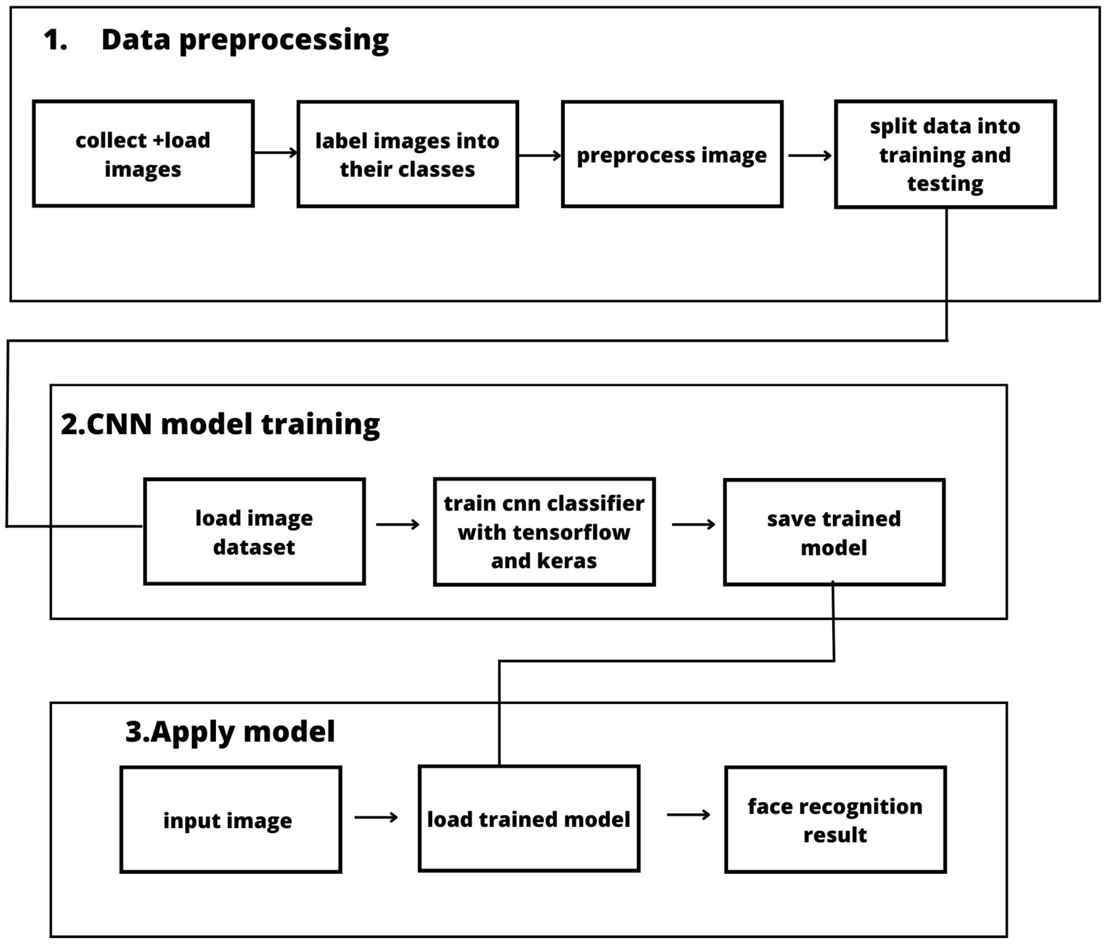
Using that github repository code, we will enhance our project with the ability to detect whether that face is real or somebody showing a picture of someone else

5 **Testing**

Along the way, we will test how our code behaves, and make changes accordingly. The final version should easily identify both of our faces easily.

? **Machine learning: CNN, tensorflow.**

Since our main project depends heavily on a simple ready-to-use face recognition library, we decided to opt for building our own model, stripped of login/registration features. This model will detect faces automatically and estimate the name of the person according to an invariant database.


1. Data preprocessing
collect +load images
label images into their classes
preprocess image
split data into training and testing
2.CNN model training
load image dataset
train cnn classifier with tensorflow and keras
save trained model
3.Apply model
input image
load trained model
face recognition result