# Outline of the Software Being Tested

The software I am testing is a desktop GUI application developed in Python as part of a project for my home university library that I was working on with a colleague. The project is about face recognition, and it aims for three main user goals: to tag a person and save its data in a metadata file, recognize the known faces that were tagged before, and to cascade tags to all other images in the collection in which the same face appears.

## Languages and libraries used

* Python
* face\_recognition: Python library that implements face recognition with a deep learning model
* tkinter: Python library for implementing the GUI
* openCV: Python library for computer vision

## GitHub Repo

<https://github.com/Sherif-Sakran/face_recognition>

# Learning Outcomes

## Analyze requirements to determine appropriate testing strategies

### Range of requirements, functional requirements, measurable quality attributes, …

Requirements were categorized into functional and non-functional requirements where the former included what the software does and the latter explains how the application performs. For the purpose of this project, I will be working on both requirements 1) the software would recognize the known faces in a given set of photos; 2) The software stores the data entered in a general metadata file and in a separate file. The full list of requirements can be seen in *1.1-requirements.txt* in the repository.

### Level of requirements, system, integration, unit.

There are three units in this software that could be thought of. Firstly, fitting the faces into the ML model, along with other information about the photo for the metadata. Secondly, recognizing the faces such that the model takes a photo that contains face(s) then compare them to the database to predict the person(s) in the photo. Both fitting and recognizing are to be integrated. Lastly, the GUI that integrates with them to form the system.

### Identifying test approach for chosen attributes.

Unit testing will be used for entering the metadata information withing the given restrictions. Unit testing will also be used for recognizing faces in photos that were tagged manually. Followingly, function testing will be applied to compare the sequential recognition of faces to a parallel version, which will be developed. A system test is to be done after the GUI is designed and integrated with the two previously tested units.

### Assess the appropriateness of your chosen testing approach.

The modularity of the software matches well with the chosen unit tests, and in addition to the system tests done, the previously done component-based testing enabled an acceptable choice of a ML model. Please refer to *1.4-appropriateness.txt* in the repository for more details.

## 2. Design and implement comprehensive test plans with instrumented code

### 2.1. Construction of the test plan

Test is an integral part of the lifecycle of software projects, and for this project, a test plan was constructed to show how the testing will be done and fitted into the lifecycle of the development of the project. The document can be seen in *Test\_plan.docx* in the repository.

### 2.2. Evaluation of the quality of the test plan

The test planning document discusses two main vulnerabilities: detailed requirements and schedule, and it shows how I planned to manage them by extending some parts of the development and choosing an agile methodology to allow for iterative feedback.

### 2.3. Instrumentation of the code

Drivers are to be used for the mentioned requirements. As discussed in the test planning document, a trained module driver will be used for the recognition module and a metadata driver will be used for the information storing module. Moreover, a model was trained on faces to be used in the recognition module.

### 2.4. Evaluation of the instrumentation

The document shows how the mentioned drivers would help in testing the two main modules being tested. The first driver would provide a trained module and allow the module under test to go through the unit testing required for checking the recognition accuracy. This specific test in itself implied component-based testing as well. The second driver provides the information as if it were generated from the GUI entries. Once the other modules are ready to replace the drivers, sub-systems are ready for testing.

## 3. Apply a wide variety of testing techniques and compute test coverage…

### 3.1. Range of techniques

For R1, recognizing faces, it was tested through functional testing in addition to structural testing, which has a coverage of 100% For better performance, some modifications was applied to that unit, and it has undergone regression testing accordingly. On the other hand, R2 was divided into two ITFs, which were both tested through functional testing using the unittest module in Python. A system testing was done at the end on the GUI of the system. Please refer to *Testing.docx* for detailed documentation.

### 3.2. Evaluation criteria for the adequacy of the testing

Since it cannot be concluded that testing was adequate, the notion of *in*adequate criteria was adopted. Consider R2 where the test suite was chosen to follow the obligations asserted by the specifications. Both the date and name(s) entries were tested to match the design rules. For the date specifically, tests were designed to exercise all the branches. Statement and branch coverage of 100% still cannot guarantee a rigorous testing.

### 3.3. Results of testing

Some of the tests failed due to either a missing logic or mistaken condition that leads to unintended control flow. For R1, a missing logic for raising exceptions with non-photo or unpermitted files in addition to some uncontrollable factors, such as model accuracy that was part of the component-based testing done to choose the ML model. For both ITFs of R2, the results showed a missing logic an a logical error. Please refer to Testing.docx for detailed documentation.

### 3.4. Evaluation of the results

Results were considerably satisfying where some of the test cases that were chosen as boundary values failed and detected errors in the software. They were especially designed to be specific and distinct from each other such that a failed test case directs to one confirmed cause. That was possible for this specific software, and solving those errors was enough to pass the tests when done again as shown in *Testing.docx* in details.

## 4. Evaluate the limitations of a given testing process

### 4.1. Identifying gaps and omission in the testing process.

While testing R2 was straightforward without deficiencies because it systematically tests some conditions within the date/name(s), R1 was challenging. R1 implementation uses a ML model from *face\_recognition* library, and tests 5 and 6, which failed, suggest low accuracy of the model used. Therefore, a better component-based testing and/or more training to the model with more data would be considered for remedy.

## 4.2. Identifying target coverage/performance levels for the different testing procedures.

For R2\_date (YYYY-MM-DD), the testing procedures aimed at meeting the design rules obliged by the specifications through a complete coverage of branches. For R1, the target performance was dependent on the model used, which is subject to component-based testing, and it targeted a complete statement coverage.

## 4.3. Discussing how the testing carried out compares with the target levels.

R2 Testing targets met the targets as per the specification and the test suite results. On the other hand, the target levels of R1 showed some deficiency. Although it was meant to keep tests *modular*, some uncontrollable dependency on the model used led to unsatisfactory, less confident results for the targets.

## 4.4. Discussion of what would be necessary to achieve the target levels.

For R1, more rigorous criteria should be applied to the component-based testing in the sense that the chosen model would be the best option, and the model itself should be trained on suitable photos in terms of quality and quantity. As such, resulting conflictions will be, confidently, due to the recognition unit being under test.