The software being tested implements face recognition. Given a set of photos, it should be able to tag faces along with metadata and to recognize the known faces that were fitted into the ML model. Detailed requirements can be found in in *1.1-requirements.txt* in the repository, but for the purpose of the test plan, the following two requirements will be considered:

R1: The software would recognize the known faces in a given set of photos. Considering figure 1 that shows a photo of princess Diana at the pyramids of Giza, the software should recognize the face(s) in the photo.

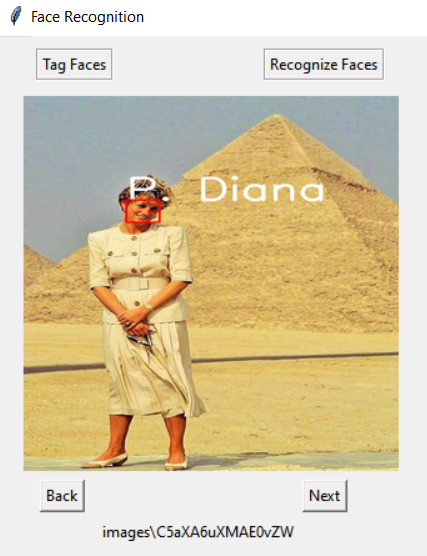


Figure 1

R2: The software stores the data entered in a general metadata file and in a separate file. For *simplicity,* the requirement would focus on only storing the metadata in a separate file, and the overall metadata file would not be necessary to be included at this stage as seen in the following figure.



Figure 2

# Priority and Pre-requisites

* R1: This is a functional requirements and is considered a major module in the system. As such:
  + It will be given a high priority for testing and resources, and early approaches are to be considered for validation and verification to ensure the program meets the requirement and specifications, respectively.
  + It should be analyzed and tested in a way to keep the software safe without optimistic inaccuracy. That is, it must accept only the correct predictions of recognizing faces, and correct here is to be determined based on the confidence of the model about the prediction.
  + The inputs and outputs are as follows:
    - Inputs:
      * A set of photos
      * A trained deep learning model
    - Outputs:
      * Prediction of the face(s) in the given set of photos
  + To verify the software, tests should consider variety of test cases to cover major input space, suggesting the following:
    - For the aim of an accurate model, *redundant* checks are to be done using
      * low quality photos
      * photos with multiple persons
      * photos with faces are close to each other
      * photos without faces
    - testing with different photo extensions
    - testing with different set sizes and checking if there is a maximum limit
* R2: This is requirements is a module but with lower weight of the program. As such:
  + Medium priority will be given and low resources are to be allocated
  + The *simplicity* principle suggests testing a restricted version of the requirement, however, it should not cause any issues to our objective in this case of storing the metadata.
  + The inputs and outputs are as follows:
    - Inputs will be the metadata information
      * Name(s)
      * Photographer
      * Date (YYYY-MM-DD)
      * Place
      * Event
      * Description
      * Source
    - Output will be metadata that follows JSON format and saved in a .csv file
  + To verify the software reads and saves the metadata as expected, checks should cover variety of combinations for the inputs with more focus on should-be-entered entries, which is only the names in this case.
    - The following checks are suggested for the name(s) entry:
    - More names than detected faces
    - Less names than detected faces
    - Different separation character
    - Entering no names
    - Assuming the date should be constrained to the ISO format and would not allow for human errors, which was not required, the following checks are to be checked:
      * Entering an invalid date
        + Days more than the month
        + Month > 12 or < 01
        + Future dates
        + Characters
        + …

# Scaffolding and Instrumentation

The project follows the agile development methodology and is built on a *bottom-up* testing approach which requires having the low-level modules to be implemented and tested using drivers whenever needed that are to be replaced afterwards. The abstracted version of the subsystems withing the system can be seen in figure 3.

Figure 3

For the chosen requirements:

* R1: in order for the software to recognize faces, we need to have a trained model that can recognize the faces. Assuming we have a trained model, our module under test will be recognizing faces in one or more photos. As such, we need a driver that works as the model that recognizes the faces as seen in figure 4, and for that, I will be using hard-coded encodings that represent face features, and the module being tested would use those encodings to recognize faces in the input set of photos. It is worth mentioning that this test itself will imply a test for the recognition component adopted from the face recognition library that implements a model with deep learning.

Figure 4

* R2: for the software to store the metadata file, it will get it through entries in the GUI. However, to have this module done and tested independently before having the GUI ready to be integrated, we can use a driver that directly provides those information, such as the name and source, as seen in figure 5. This driver can be replaced by the module that reads those entries from the GUI once ready.

Figure 5

# Process and Risk

Timeline

Description automatically generated with medium confidenceThe estimated time for the project throughout its lifecycle is five weeks. Some resources were provided before the start of the project, which were found to be useful, asserted this estimation. The resources included references to libraries that implement ML models and photos that could be used in testing. The schedule shown in figure 6 is designed for the project development, and it indicates every step in the lifecycle and its estimated time.

Figure 6

Risks are mainly about requirements and schedule. The documented requirements given to us for the project were limited without a detailed explanation of features being required, and we were supposed to come up with a larger, clearer set of requirements on our own from the “user goals” to reach the specifications properly. The project is supposed to be iterative which enables us to reconsider or modify any of the requirements, but this may cause a delay in the software delivery in case of any major change of requirements. Although that was with very low possibility, that was considered for extending the code and integration phase by two days, testing subsystems by one day, and testing the system by one day. That may seem unnecessary, but it could be helpful, especially because no contingency plans were considered.