FaceAlbum Design Decision

2017WS Praktikum Informatik 1

Zheng Li

0906101

Table of Contents

1 Project Overview

The main assignment of this project is to develop a photo gallery application with face recognition and emotion detection function using Python and Machine Learning. The aim is to find out a suitable approach for users to sort their photo gallery by face of people and the emotion of them.

2 Scenarios

2.1 Use case diagram

Figure 1 shows the use case diagram

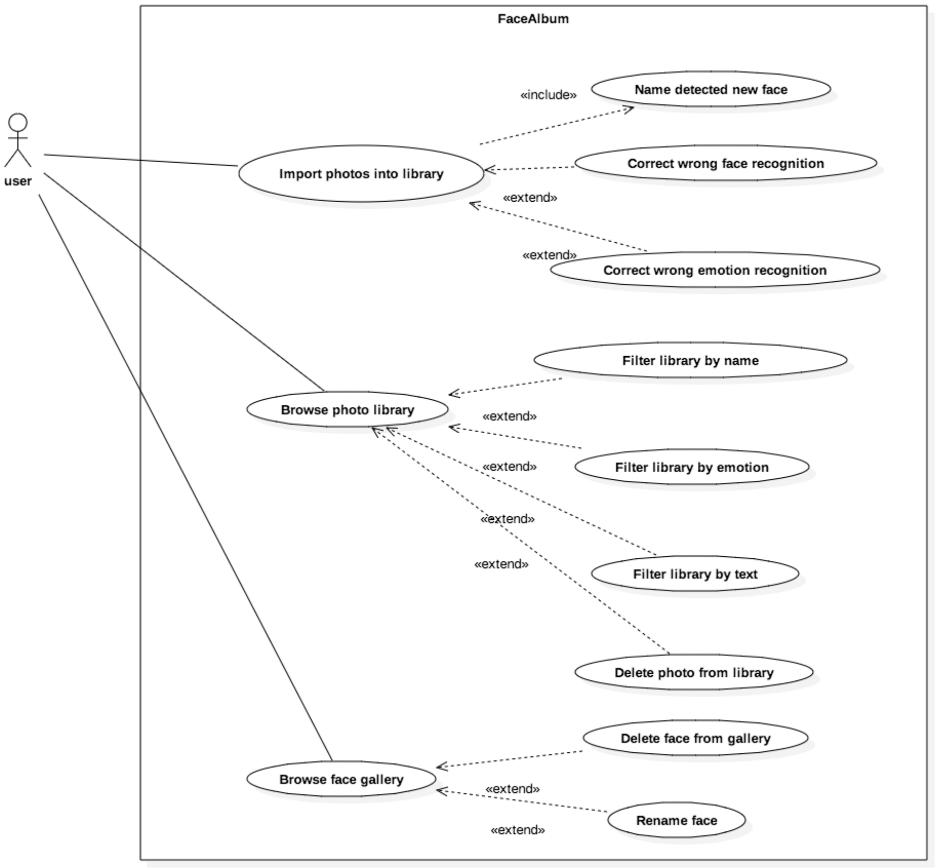


Figure 1 FaceAlbum Scenarios Use Case Diagram

2.2 Use Case List

2.2.1 Primary Actor	***2.2.2 Use Cases***
User	Import photos into library
User	Name detected faces
User	Correct wrong face recognition
User	Correct wrong emotion recognition

User	Browse photo library
User	Filter library by name
User	Filter library by emotion
User	Filter library by text
User	Delete photo from library
User	Browse face gallery
User	Delete face from gallery
User	Rename face

2.3 Use case: Import photos into library

Actors:	User
Description:	User can import new photos into the library.
Trigger:	Import button is clicked.
Preconditions:	• User provides a list of pictures files with compatible .format like JPG and PNG. • Picture can not be already included in the library.
Postconditions:	1. Pictures are imported into the library database. 2. Tumbnails of pictures are generated.
Normal Flow:	1. User clicks on „Import new photos“ button. 2. User selects a list of picture file. 3. User clicks import.
Alternative Flows:	None
Exceptions:	• File format is not compatible. • Picture already exists in library.
Includes:	• Name detected faces
Priority:	High
Frequency of Use:	Middle
Business Rules:	None
Special Requirements:	None
Assumptions:	None
Notes and Issues:	None

2.4 User case: Name detected faces

Actors:	User
Description:	User can give a name tag to a detected face during importing.
Trigger:	User clicks on the marked face in picture.
Preconditions:	• The detection of the face is correct.
Postconditions:	• The name tag of the face is stored in database. • The cropped out face picture is stored separately for later face recognition model training.
Normal Flow:	1. User click on a detected face in the picture. 2. User give it a name.

Alternative Flows:	1. User click on a detected face in the picture. 2. User informs that it is a wrong detection by clicking not a face button.
Exceptions:	None
Includes:	None
Priority:	High
Frequency of Use:	High
Business Rules:	None
Special Requirements:	None
Assumptions:	None
Notes and Issues:	None

2.5 Use case: Correct wrong face recognition

Actors:	User
Description:	User can correct a wrong recognized face during importing.
Trigger:	Correct name button is clicked.
Preconditions:	1. User decides that the recognized face is wrong.
Postconditions:	1. The right name tag is assigned, and stored in the database. 2. Cropped face with the right name tag is ready for re-training of the face recognition model.
Normal Flow:	1. User clicks on „wrong face“ button. 2. User gives the right name. 3. User clicks “correct” button.
Alternative Flows:	None
Exceptions:	Non
Includes:	None
Priority:	High
Frequency of Use:	Middle
Business Rules:	None
Special Requirements:	None
Assumptions:	None
Notes and Issues:	None

2.6 Use case: Correct wrong emotion recognition

Actors:	User
Description:	User can correct a wrong recognized emotion during importing.
Trigger:	Correct emotion button is clicked.
Preconditions:	User decides that the recognized emotion is wrong.
Postconditions:	1. The right emotion tag is assigned, and stored in the database.

	2. Cropped face with the right emotion tag is ready for re-training of the emotion recognition model.
Normal Flow:	1. User clicks on „wrong emotion“ button. 2. User gives the right emotion. 3. User clicks “correct” button.
Alternative Flows:	None
Exceptions:	Non
Includes:	None
Priority:	High
Frequency of Use:	Middle
Business Rules:	None
Special Requirements:	None
Assumptions:	None
Notes and Issues:	None

2.7 Use case: Browse photo library

Actors:	User
Description:	User can browse the photos in the library.
Trigger:	„Browse library“ button is clicked.
Preconditions:	Library database is created.
Postconditions:	Photos in the library are shown in the gallery browser.
Normal Flow:	User click on “Browse library” botton. Photos in the library are shown in the gallery browser.
Alternative Flows:	User click on “Browse library” botton. Empty library is shown, application suggests user to import some photos.
Exceptions:	None
Includes:	None
Priority:	High
Frequency of Use:	Very High
Business Rules:	None
Special Requirements:	None
Assumptions:	None
Notes and Issues:	None

2.8 Use case: Filter library by name

Actors:	User
Description:	User can filter the gallery by name tag.
Trigger:	User select a name tag from name tag filter.
Preconditions:	1. Library database is created and has some photos imported.

	2. Photo is already classified by the face recognition model.
Postconditions:	Photos with the right name tag in the library are shown in the gallery browser.
Normal Flow:	1. User choose one name tag from name filter. 2. Photos with the right name tag in the library are shown in the gallery browser.
Alternative Flows:	None
Exceptions:	None
Includes:	None
Priority:	High
Frequency of Use:	Very High
Business Rules:	None
Special Requirements:	None
Assumptions:	None
Notes and Issues:	None

2.9 Use case: Filter library by Emotion

Actors:	User
Description:	User can filter the gallery by emotion tag.
Trigger:	User select a emotion tag from emotion tag filter.
Preconditions:	1. Library database is created and has some photos imported. 2. Photo is already classified by the emotion recognition model.
Postconditions:	Photos with the right emotion tag in the library are shown in the gallery browser.
Normal Flow:	1. User choose one emotion tag from emotion filter. 2. Photos with the right emotion tag in the library are shown in the gallery browser.
Alternative Flows:	None
Exceptions:	None
Includes:	None
Priority:	High
Frequency of Use:	Very High
Business Rules:	None
Special Requirements:	None
Assumptions:	None
Notes and Issues:	None

2.10 Use case: Filter library by text

Actors:	User
Description:	User can filter the gallery by a free text input.

Trigger:	User give a text description of the filter.
Preconditions:	1. Text are simple, consist of the names and emotions of the search. 2. Text contains the name and the emotion that the user want to search for, for example like “Steve Marry happy”. 3. Text does not contain complicate logical words, for example like “Steve and Marry are both not very happy”.
Postconditions:	Photos with the right name and emotion tag in the library are shown in the gallery browser.
Normal Flow:	1. User types in the text and click filter. 2. Photos with the right name and emotion tag in the library are shown in the gallery browser.
Alternative Flows:	None
Exceptions:	None
Includes:	None
Priority:	High
Frequency of Use:	Very High
Business Rules:	None
Special Requirements:	None
Assumptions:	None
Notes and Issues:	None

2.11 Use case: Delete photo from library

Actors:	User
Description:	User can delete photos in the library.
Trigger:	„Delete“ button is clicked.
Preconditions:	None
Postconditions:	Picture deleted from library
Normal Flow:	1. User click on “Delete” botton. 2. Picture entries of the photo is deleted in pictures table in database. 3. Faces in that picture is also deleted in faces table in database.
Alternative Flows:	None
Exceptions:	None
Includes:	None
Priority:	Middle
Frequency of Use:	Middle
Business Rules:	None
Special Requirements:	None
Assumptions:	None
Notes and Issues:	None

2.12 Use case: Browse face gallery

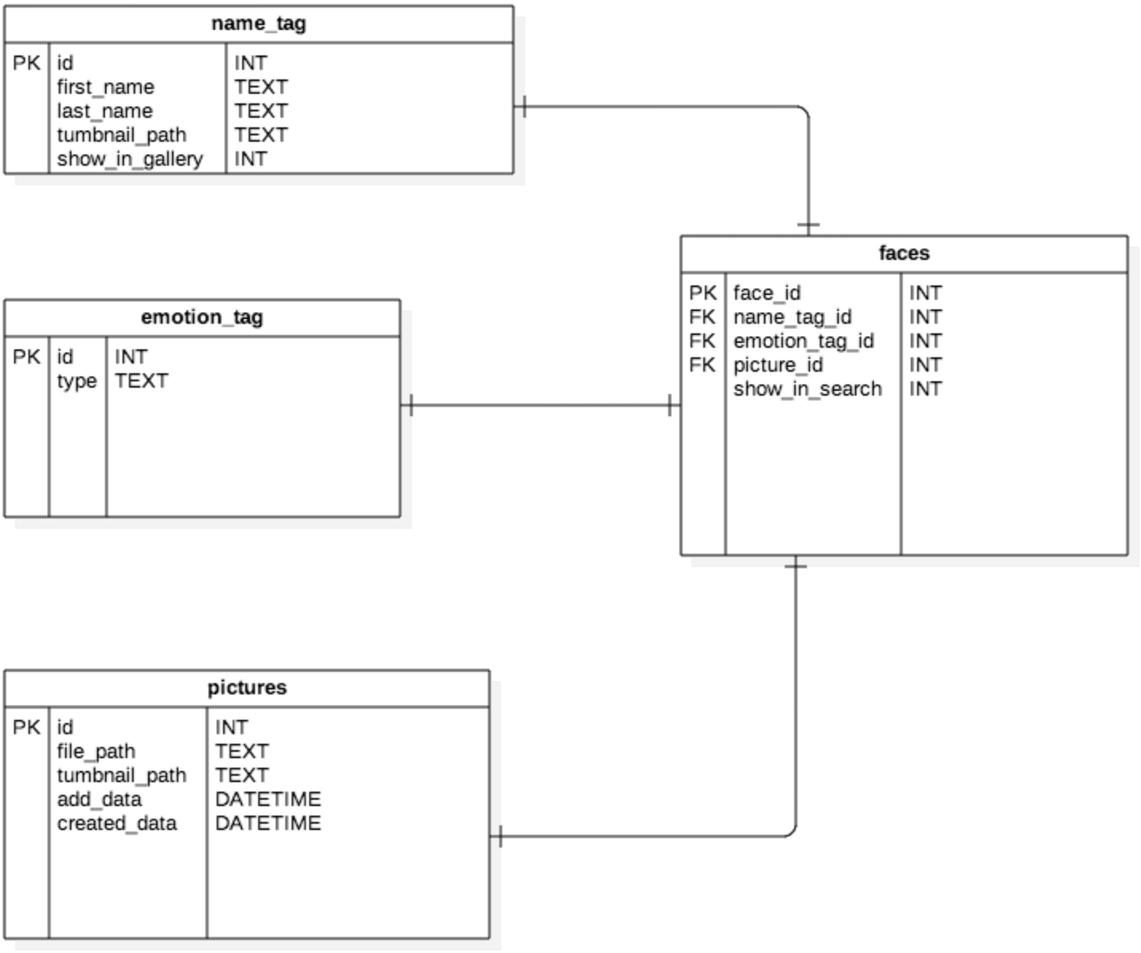
Actors:	User
Description:	User can browse the faces already identified in the library.
Trigger:	„Browse faces“ button is clicked.
Preconditions:	At least one face can be identified by the model .
Postconditions:	Faces in the library are shown in the gallery browser.
Normal Flow:	4. User click on “Browse faces” botton. 5. Faces in the library are shown in the gallery browser.
Alternative Flows:	1. User click on “Browse faces” botton. 2. Empty library is shown, application suggests that there are still not enough data collected.
Exceptions:	None
Includes:	None
Priority:	High
Frequency of Use:	Very High
Business Rules:	None
Special Requirements:	None
Assumptions:	None
Notes and Issues:	None

2.13 Use case: Rename face

Actors:	User
Description:	User can rename faces in the gallery.
Trigger:	„Rename“ button is clicked.
Preconditions:	None
Postconditions:	Faces is renamed.
Normal Flow:	User click on “Rename” botton. Name_tag entry of that face is renamed.
Alternative Flows:	None
Exceptions:	None
Includes:	None
Priority:	Middle
Frequency of Use:	Middle
Business Rules:	None
Special Requirements:	None
Assumptions:	None
Notes and Issues:	None

3 Data structure of the Database

Figure 2 shows the ER-diagram of the data structure.



name_tag

PK	id	INT
	first_name	TEXT
	last_name	TEXT
	tumbnail_path	TEXT
	show_in_gallery	INT

emotion_tag

PK	id	INT
	type	TEXT

faces

PK	face_id	INT
FK	name_tag_id	INT
FK	emotion_tag_id	INT
FK	picture_id	INT
	show_in_search	INT

pictures

PK	id	INT
	file_path	TEXT
	tumbnail_path	TEXT
	add_data	DATETIME
	created_data	DATETIME

Figure 2 ER-diagram of the data structure

4 Logical View

Figure3 shows the class diagram

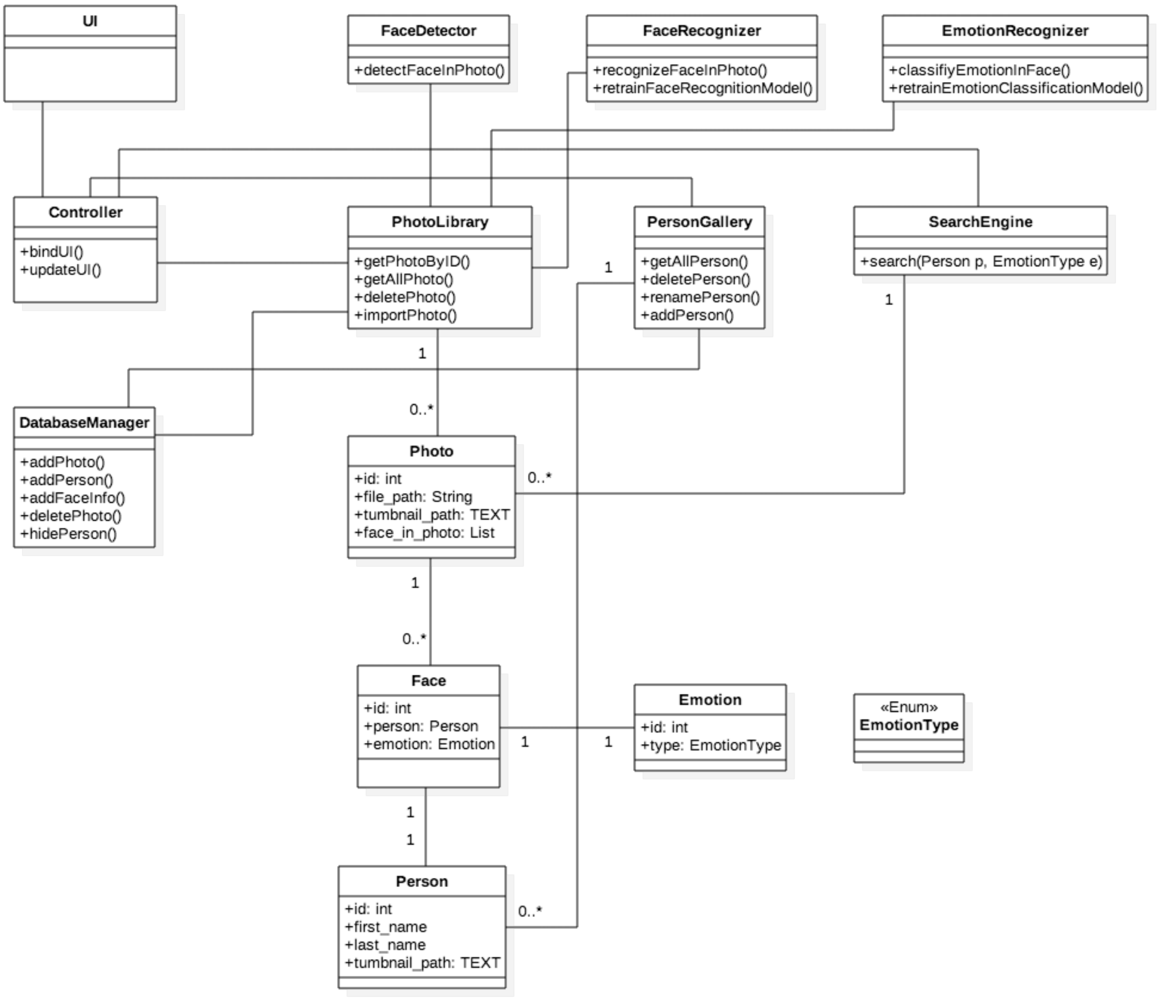


Figure 3 Class diagram

5 Development View

Figure 4 shows the component diagram.

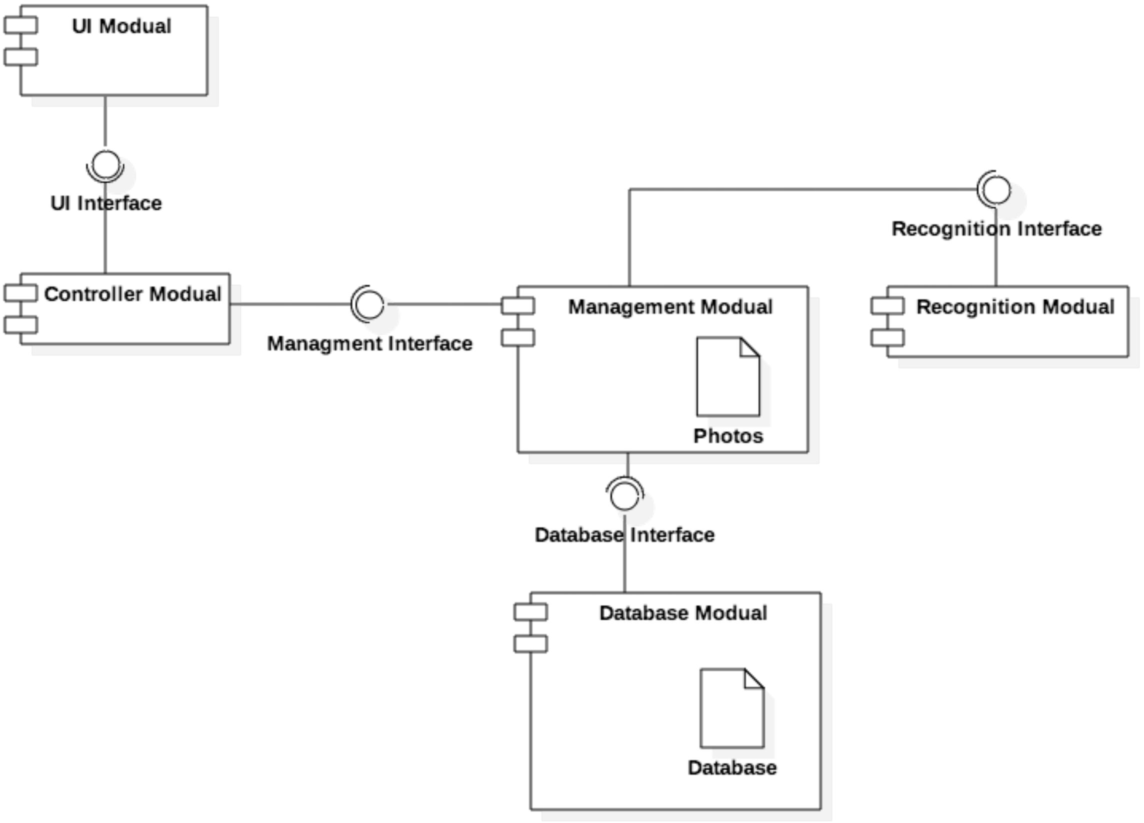


Figure 4 Component diagram

6 Process View

6.1 Import photos

Figure 5 shows the activity diagram of importing photos.

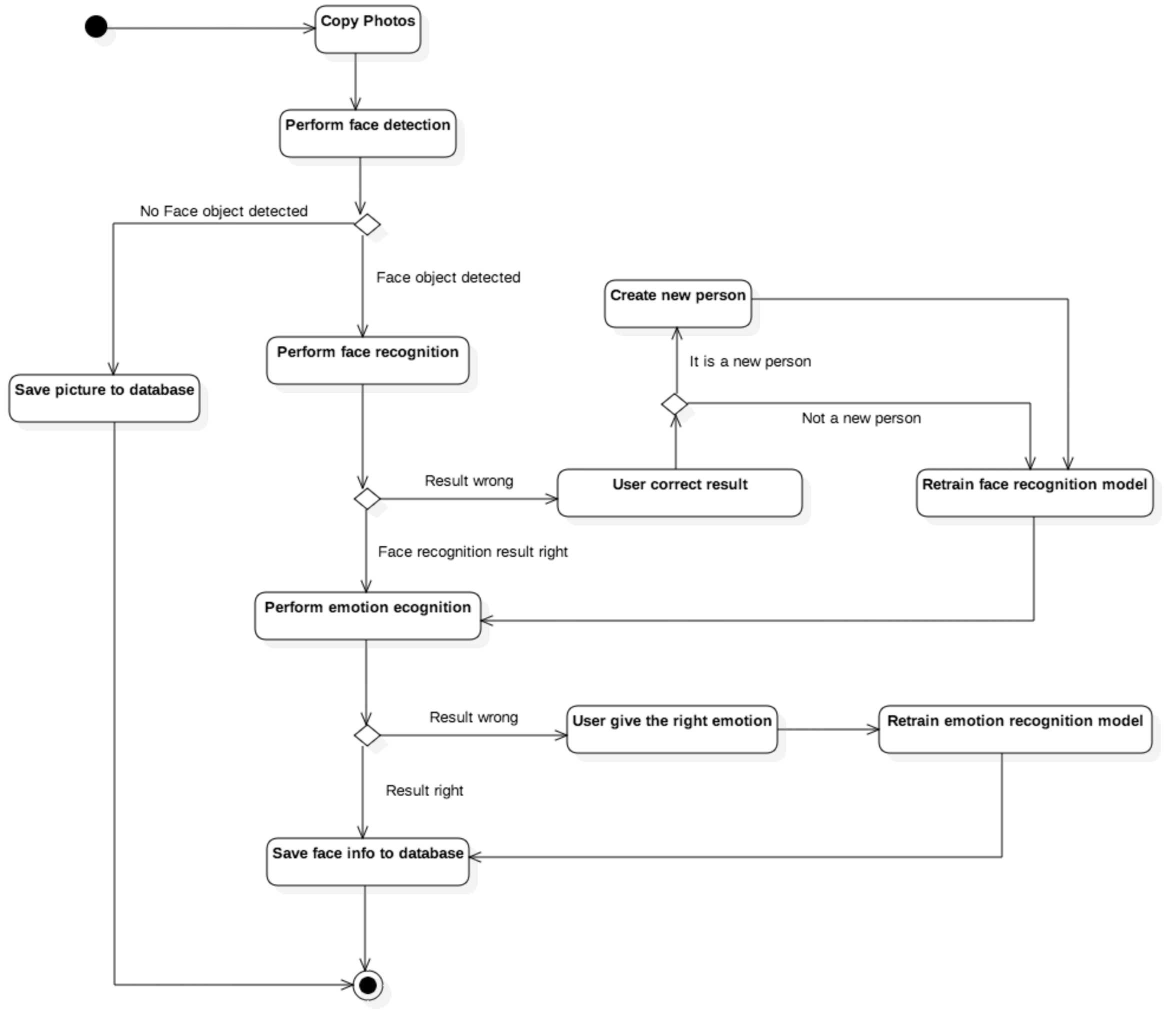


Figure 5 Activity diagram - import photos