Matthew Menten
Eli Jacobshagen
CSCI 5/4448 - Montgomery
Fall 2019

Project 4

Project Summary

Project Title: Facial Recognition with Raspberry Pi

Team Members:

- Matthew Menten
- Eli Jacobshagen

Overview:

The goal of this project is to create an application which can learn new faces, detect known faces, and keep track of unknown faces. We're thinking of something that could serve as a basic security system, almost like a NEST doorbell. The system will make use of a Raspberry Pi equipped with a camera. The Pi will record video with its camera and use a messaging system over a TCP socket to send this video stream to a computer running our application. If a face is detected in an image (within the video stream) received from the Pi, the application will determine whether that face is known or unknown. We plan to implement a feature which attempts to learn unknown faces and create a unique ID for them (stretch goal). For example, a user could determine if a specific unknown person keeps knocking on their door at 2:00 am. If time permits, we would like to include a publish/subscribe system so that users of the application can receive SMS alerts when certain faces are detected.

Project Requirements

1. The application shall consist of two main parts: the Raspberry Pi and the user software
    a. The Raspberry Pi shall record and send a live video stream to the user software over the internet
    b. The user software shall perform facial recognition on this video stream and display the results.
        i. The user software shall display the video stream with colored boxes and names around detected faces.
    c. The user software shall allow users to subscribe to event notifications. (stretch goal)

        i. Subscribed users shall be sent SMS alerts when faces are detected.
2. The user software shall provide interaction through a GUI
    a. The software and GUI shall be desktop based (not mobile or web)
    b. The GUI shall allow users to connect to the Pi upon application startup
        i. NOTE: This responsibility may be removed if we can successfully use a static IP and port for the Pi
    c. The GUI shall display the video stream from the Pi and the face-box overlay described in 1.b.i
3. The user software shall store learned faces and their associated names in a database
    a. The GUI shall allow users to add a new labeled face to the application's collection of known faces
        i. The user software shall communicate with the Pi to obtain pictures of the new face, which will then be processed to create a "face fingerprint"
        ii. The "face fingerprint" and associated name will be saved in a database managed by the user software
    b. The GUI shall allow users to delete labeled faces from the application's collection of known faces
        i. Users may delete a face by name
        ii. Users may delete all face data stored in the database
4. The GUI shall expose an interface to allow users to receive SMS alerts for faces detected by the application as described in 1.c.i (stretch goal)
    a. The GUI shall allow users to enter their phone number
    b. The user software shall send SMS alerts to the given phone number when a new face is detected
    c. The GUI shall allow users to unsubscribe from SMS alerts

Responsibilities and High-Level Architecture

- App
    - Holds and initializes the main graphics window (MainWindow object)
    - Executed on startup to kick off the application
- MainWindow
    - Holds menu bar (File, Edit, etc...)
    - Handles interaction with the menu bar (What happens when dropdown options are clicked)
    - Holds the CentralWidget object (Displaying video)
- CentralWidget

    - Defines main graphics layout (holds widgets to display video stream from Pi, etc.)
    - Holds various other widgets and switching application context b/t them
    - Holds the AppEngine
- AppEngine
    - Orchestrates communication between the different components
    - Receives video stream from VideoStreamer object
    - Uses FaceRecognizer object to determine faces in the stream
    - Holds the Database object and uses it to perform queries for known faces
    - Sends face-recognized video to the Central Widget for display
    - Notifies AlertObserver when faces are detected (stretch goal)
- VideoStreamer
    - Object to represent the Raspberry Pi sending a video stream to the application
    - Connects to the Pi using a TCP socket
    - Handles data (video stream) coming from the Pi and sends it to the AppEngine for processing
- Database
    - Abstracts interaction with the database (type of database, running queries, returning results, error handling)
- AlertObserver (stretch goal)
    - Observes the AppEngine and is notified when faces are detected
    - Sends SMS alert to subscribers (text message using Twilio API)

Use Cases

1. Name: Connect to Raspberry Pi
   Prerequisites: None
   Actors: User
   Tasks: The user runs the desktop software from the command line (user software). Upon running the software, the GUI pops up in a new window. Within the GUI, the user navigates to an to an option for establishing a connection with the Raspberry Pi. The user inputs an IP address to a prompt.
   Success Case: In a successful outcome of this use case, the user software will establish a connection with the Raspberry Pi and begin displaying the video stream within the GUI.
   Exceptional Cases: The only exceptional condition for this use case occurs if the user software cannot establish a connection with the Raspberry Pi. This may be due to an incorrect IP address, or an issue with the Pi itself (e.g. the Pi is turned

off, the application software on the Pi is not running, etc). If a connection with the Pi cannot be established, the user software will display a message to the user.

2. Name: Learn a New Face
   Prerequisites: Connect to Raspberry Pi
   Actors: User, Person whose face will be learned
   Tasks: Within the GUI, the user navigates to an option for adding a new face to be recognized by the application. The GUI displays a message informing the person whose face will be learned to stand in front of the Raspberry Pi camera. The GUI exposes controls which allow the user to take multiple pictures of the person whose face will be learned. After a threshold number of photos have been taken of the new face, the GUI will display a message informing the user that it has learned and saved information about the new face.
   Success Case: In a successful outcome of this use case, the application will be able to detect and display a box/name around the newly added face.
   Exceptional Cases: The only exceptional condition for this use case occurs if the application cannot detect the new face after the user adds it. Users will know this case has occurred if the person whose face was learned still shows up as "Unknown" in the GUI video stream display. Users may delete this face from the database (by name) and readd it using the above tasks to fix this exception.

3. Name: Delete a Known Face
   Prerequisites: Learn a New Face
   Actors: User
   Tasks: This use case assumes that the user has started running the user software on their computer. Establishing a connection to the Raspberry Pi is not necessary for this use case. Within the GUI, the user navigates to an option for deleting a specific face from the application. The GUI prompts the user to enter the unique name corresponding to the face which will be deleted. The user enters a name into the prompt.
   Success Case: In a successful outcome of this use case, the GUI will display a message informing the user that the specified face was successfully deleted.
   Exceptional Cases: The only exceptional condition for this use case occurs if the application cannot find saved information for the specified name. In this case, the GUI will display a message informing the user that no data is associated with the given name.

4. Name: Delete all Faces
   Prerequisites: None.

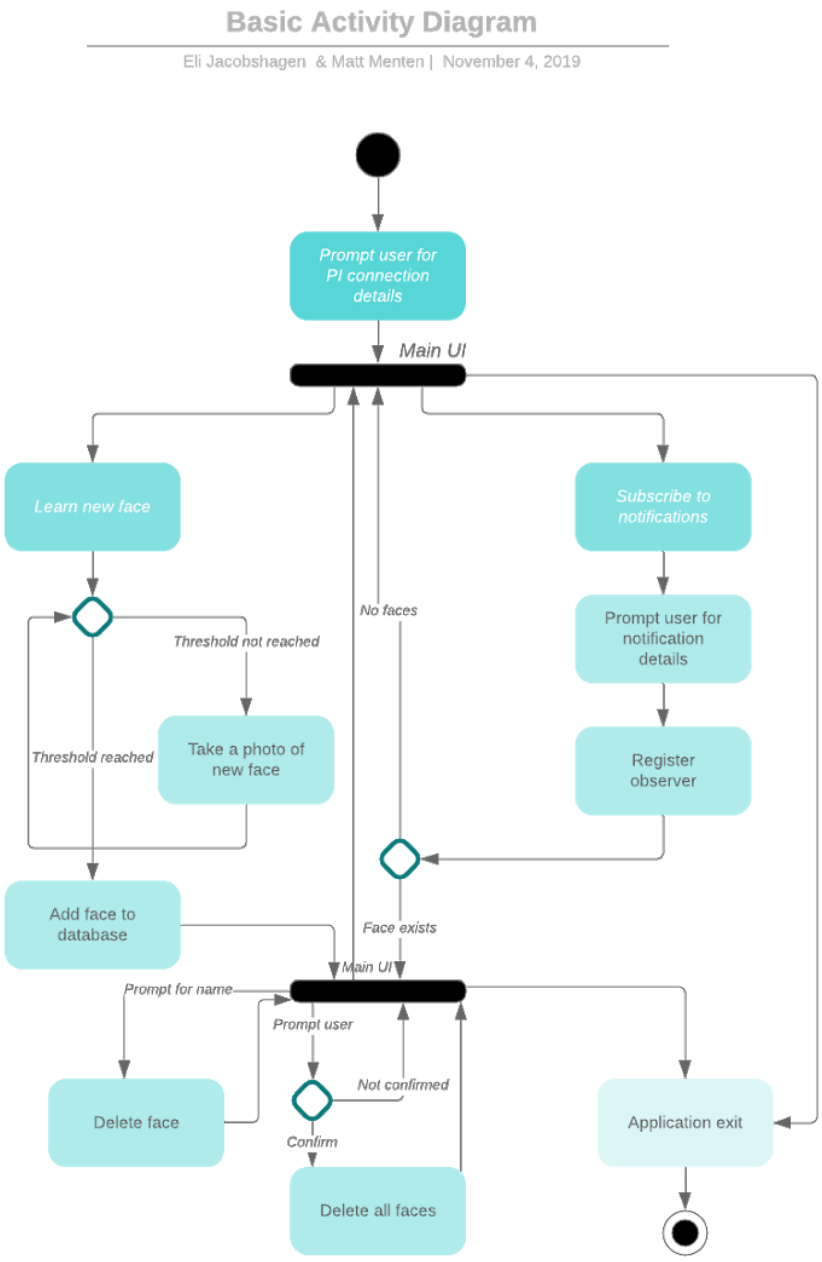
Actors: User
Tasks: This use case assumes that the user has started running the user software on their computer. Establishing a connection to the Raspberry Pi is not necessary for this use case. Within the GUI, the user navigates to an option for deleting all faces from the application. The GUI prompts the user asking for confirmation. The user either confirms or rejects the prompt.
Success Case: If the user accepts the prompt, the application will display a message informing the user that all saved faces have been deleted. If the user rejects the prompt, the application will do nothing.
Exceptional Cases: There are no exceptional conditions for this use case.

5. Name: Subscribe to SMS Notifications
   Prerequisites: None.
   Actors: User
   Tasks: This use case assumes that the user has started running the user software on their computer. Establishing a connection to the Raspberry Pi is not necessary for this use case. Within the GUI, the user navigates to an option for subscribing to SMS notifications. The GUI prompts the user to enter their phone number and the maximum frequency of notifications.
   Success Case: In a successful outcome of this use case, the user software will send an SMS message to the user when the application detects a face. An SMS will not be sent unless the time since the last notification is greater than or equal to the user-specified maximum frequency. Success is determined by the user being able to receive notifications on their phone.
   Exceptional Cases: The only exceptional conditions for this use case occur if the user-specified phone number is not valid, or if the user software encounters an error while attempting to send an alert. In both of these cases, the user software displays a warning message which informs the user of the error.

User Types:

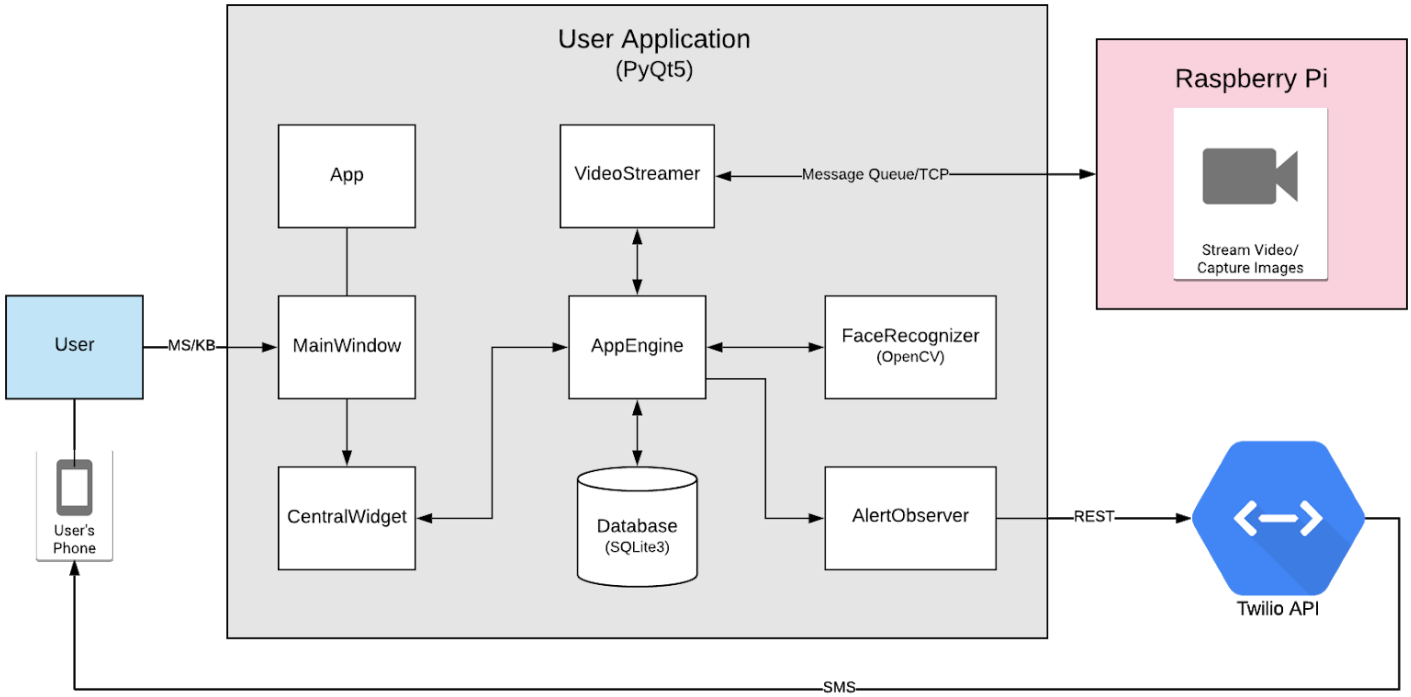
- Main App User (connect to Pi, recognize faces, add/delete faces, get alerts)
- Person whose face will be learned by the app (have pictures taken by Pi)

Activity Diagram



Basic flowchart of all use cases. Some application messages omitted for space.

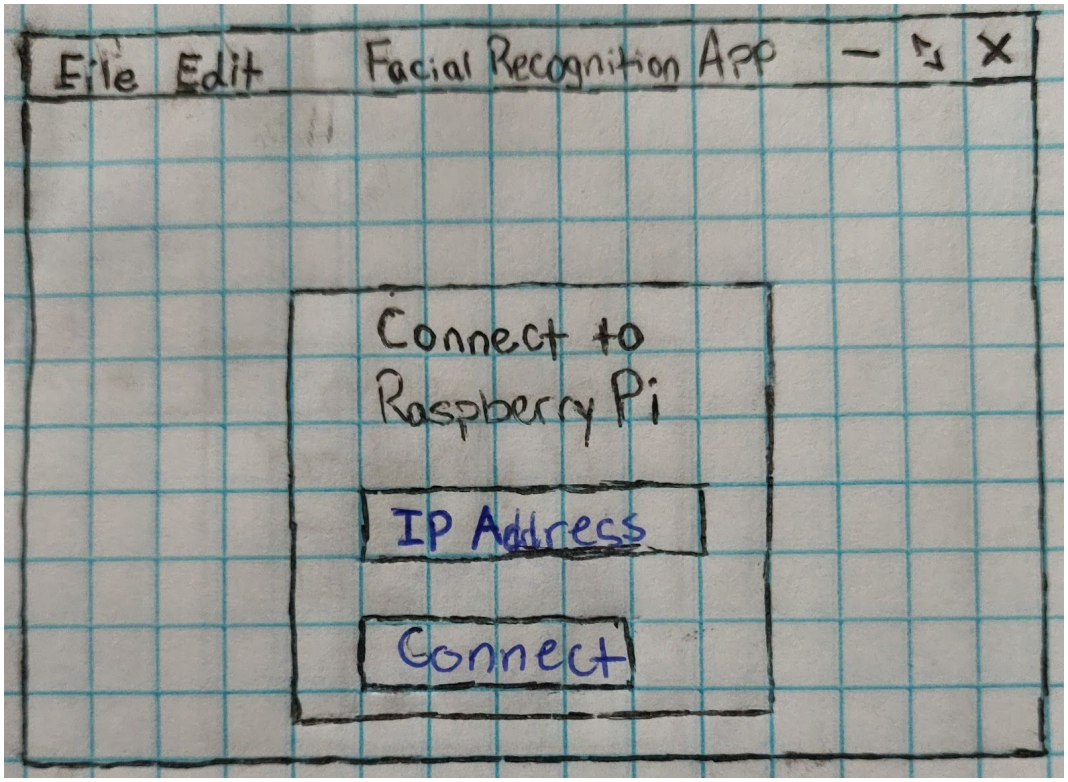
Architecture Diagram

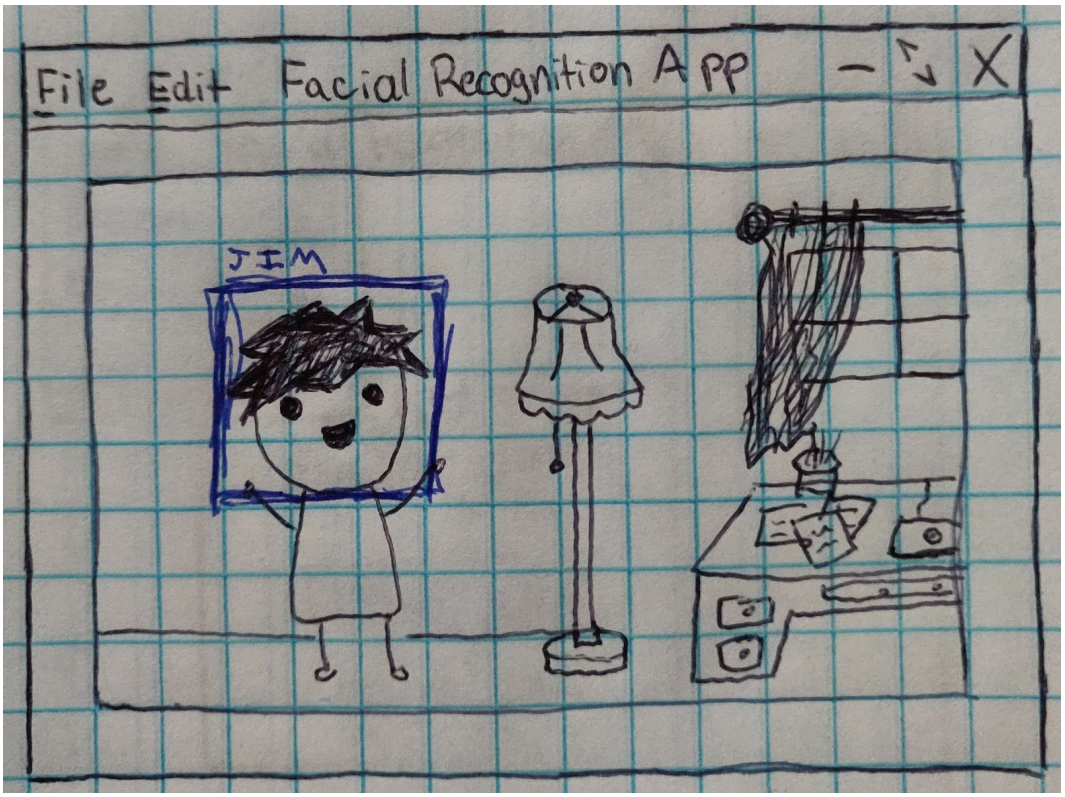


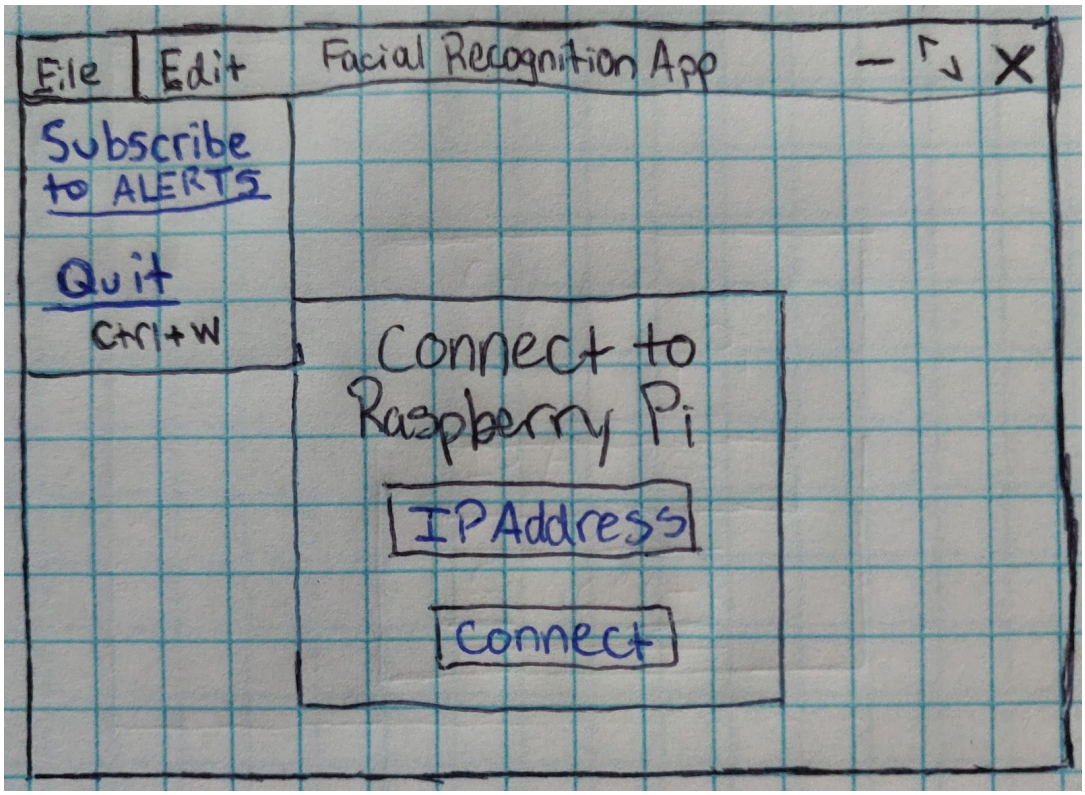
Data Storage

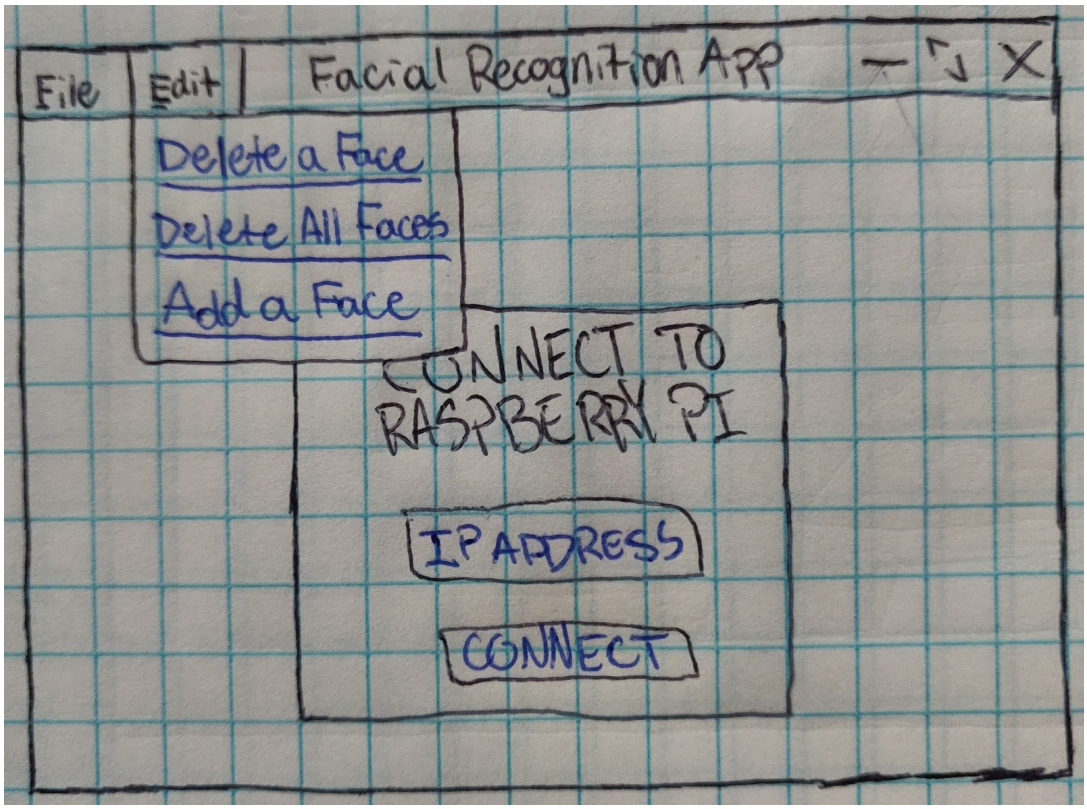
Our application will store and persist data using a Sqlite database. Sqlite databases are connected to and interacted with similarly to a MySQL or Oracle database, except they exist in a single file on the filesystem, instead of being managed by a server or separate process. To interact with our database, we will create an implement a database class. This class will abstract connecting to and performing queries/inserts/deletes on the database. This abstraction will ensure that the underlying database implementation (Sqlite) will be decoupled from operations our application performs on its database. The database class will implement private methods to execute both queries and non-queries (inserts and deletes). The database class will expose public methods to create the database, add a new named face, delete a named face, and wipe the entire database.

UI Mockups

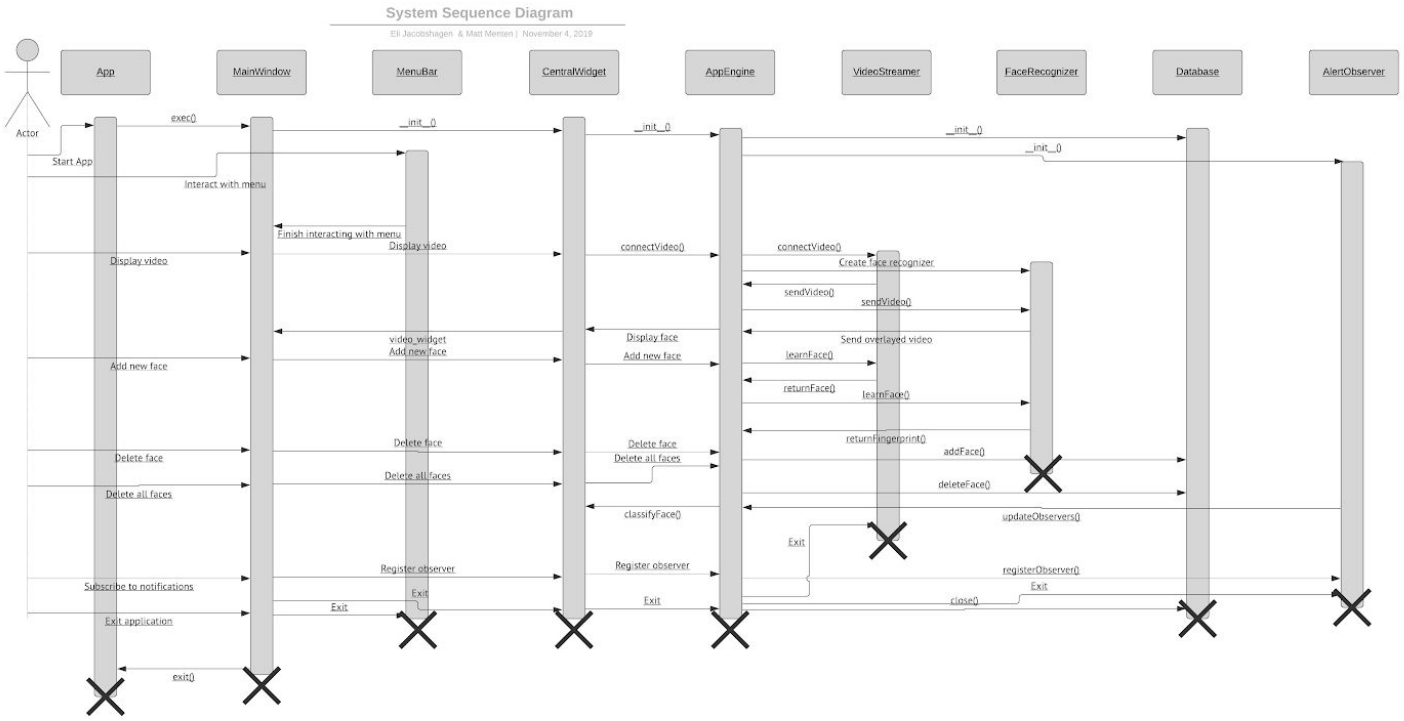

File Edit
Facial Recognition App
Connect to Raspberry Pi
IP Address
Connect


File Edit
Facial Recognition App
JIM


File
Edit
Facial Recognition App
Subscribe to ALERTS
Quit
Ctrl+W
Connect to Raspberry Pi
IPAddress
Connect


File
Edit
Facial Recognition App
Delete a Face
Delete All Faces
Add a Face
CONNECT TO RASPBERRY PI
IPADDRESS
CONNECT

Sequence Diagram



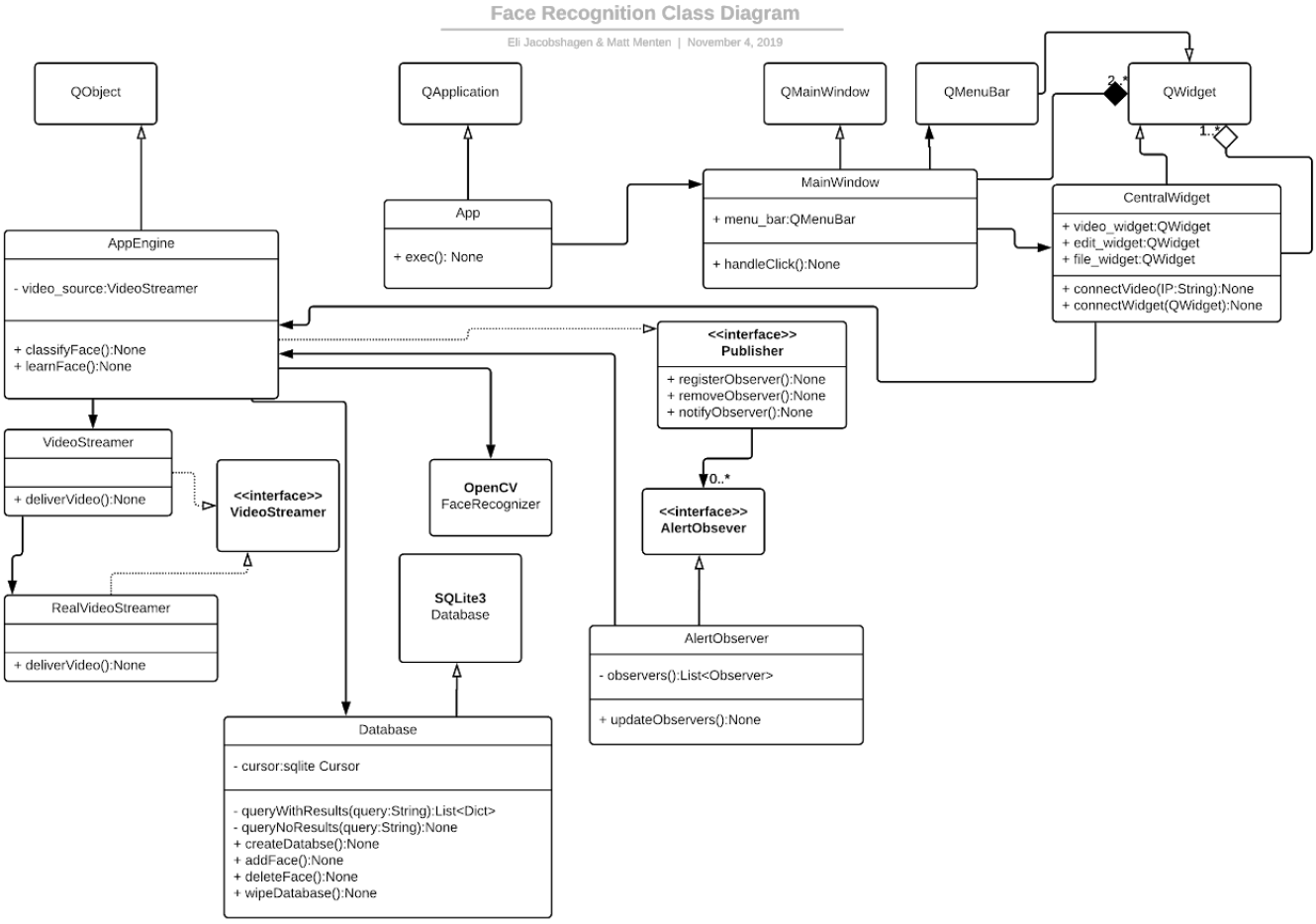
This sequence diagram models *all* user interaction with the system and its objects, as opposed to having three sequence diagrams that model one interaction each.

Interactions

1. Application establishes a connection to the Raspberry Pi, begins to receive a video stream, and displays the stream with the facial recognition overlay within the central widget.
    a. Application boots: App, MainWindow, MenuBar, CentralWidget, AppEngine, Database, AlertObserver created
    b. User interacts with CentralWidget and enters IP address: Application creates VideoStreamer and FaceRecognizer objects, begins receiving video stream
    c. VideoStreamer sends video to AppEngine, AppEngine sends images to FaceRecognizer, FaceRecognizer sends overlayed images to AppEngine, AppEngine sends overlayed video to CentralWidget for display
2. User adds a new face to be recognized by the application
    a. Application boots and user connects to the Raspberry Pi (shown above)
    b. User interacts with MenuBar and selects option to add a new face

    c. MenuBar sends a signal to the CentralWidget informing it that the user wishes to add a new face
    d. CentralWidget communicates with AppEngine, AppEngine communicates with VideoStreamer to collect and receive images of new face, AppEngine sends images to FaceRecognizer and collects results, AppEngine communicates with Database to save the new face
3. Users deletes a specific face profile from the application
    a. Application boots and user connects to the Raspberry Pi (shown above)
    b. User interacts with MenuBar and selects option to delete a face
    c. MenuBar sends a signal to the CentralWidget informing it that the user wishes to delete a face
    d. CentralWidget collects the name to be deleted via a text prompt
    e. CentralWidget sends name to AppEngine, AppEngine communicates with Database and the specified face is deleted

Class Diagram



Pattern Use

1. Proxy
    a. The application communicates with a remote object: the Raspberry Pi.
    b. Sending commands to the Pi and receiving data is a good example of the Proxy Pattern
    c. In our Class Diagram, the Proxy object is the VideoStreamer, which interacts with the Real Subject (the Raspberry Pi)
2. Observer (SMS notification system)
    a. The AlertObserver is notified by the AppEngine when a new face is detected.

b. Since the AlertObserver is external to the core functionality of the system, wrapping it with the observer pattern keeps our design more loosely coupled.
c. In our Class Diagram, the Subject/Publisher is the AppEngine and the Observer/Subscriber is the AlertObserver.