PRIVASEER

FEEL SAFER

A home surveillance system with face recognition.

by Antal Calina

1. Repository

Schematics, diagrams and codebase are contained under the following git repository: https://github.com/KaliAnt/Privaseer

2. User Requirements

The system meets the following requirements:

1. The system must save information about user activity
2. The system alerts the user when unknown activity has taken place
3. Every information is available in the application or through web
4. The system detects the face of the subject
5. The system has to process the data and send it to the server

3. System overview

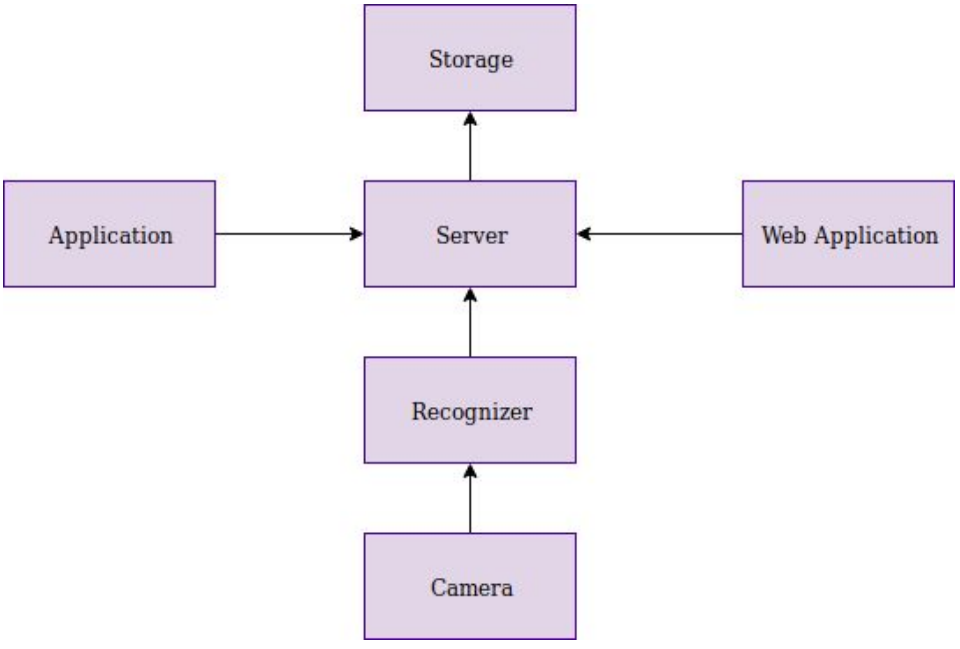

Fig 1: System diagram

Through the camera, the recognizer module collects the information and at motion, it triggers the face detection script which also recognizes the face.

The server gets data from the recognizer through REST services and saves the data in the database. It responds to POST and GET requests from the applications.

The storage is a SQLite database and stores the data that was pushed by the server.

The applications provide a visual interpretation of the data and keep the user up to date with the events that occurred.

4. Circuit overview

Fig 2: Circuit overview

The camera is connected at the Raspberry Pi and through OpenCV library, the information is collected and interpreted

5. Software design

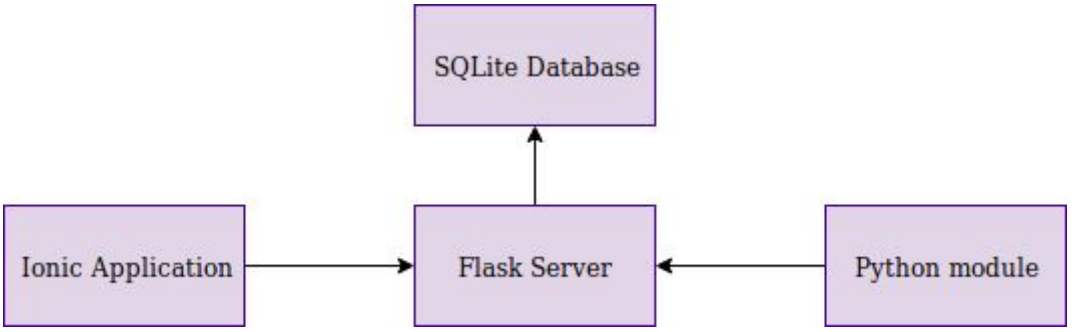

Fig 3: Software entities

5.1. Python module

The python module has 2 components: capture.py and recognizer.py. ***capture.py*** component captures the video and sends data to recognizer when it detects motion. The script takes no arguments:

```
python ./capture.py
```

recognizer.py component takes data as input. It can either learn a face from the data or test if the input image has a face and if the face matches with any of the the known subjects faces.

```
python ./recognizer.py <path_to_file> <test/train>
```

5.2. Flask Server

Flask is a microframework for Python based on Werkzeug and Jinja 2.
The Flask Server provides RESTful request dispatching.
It takes an argument, which initializes the database or runs the server:

```
flask run
flask init
```

To configure the server, export the path to the app:

```
export FLASK_APP=<path_to_server>
```

5.3. SQLite Database

SQLite is a self-contained, high-reliability, embedded, full-featured, public-domain, SQL database engine. The database holds informations about the events that occured.

5.4. Ionic Application

Ionic is a complete open-source SDK for hybrid mobile app development. It provides consistent data access, control, and protection to the structured and unstructured data. The application provides a simple user friendly interface for viewing events and profiles.

6. Results and further work

The current project supports the following functionalities:

- Reading data and saving information about user profiles
- Image processing , face detection and face recognition
- Sending data to server and saving it in a database
- Displaying information through the web and mobile applications

The following list of functionalities are to be supported:

- Loading images through web/mobile application and input them to the recognizer
- Location of the user (inside the house, outside)
- Alerting the user and the local authorities when an unfortunate event has occured