PI Face Recognition System

**Team:**

Joseph Dementri

Thien Do

Patrick Jacaszek (Scrum Master)

Peter Lieb

Samuel Liebe-randall

Dimitrios Topalis

**GitHub:** https://github.com/ThienDo95/SeniorProjectMaster

**Slack:** surveillanceworkspace.slack.com

# **1) Purpose**

This document presents the design information for the Pi Face Recognition System (PFRS).

# **2) Scope**

This document covers design elements for the Pi Face Recognition System. Included within are descriptions of the Technical Architecture, Solution Design Elements, and Roles and Responsibilities related to the Pi Face Recognition System.

## ***2.1 Requirements Summary***

**Raspberry Pi**

* Utilize Amazon Rekognition with Raspberry Pi Zero W to be able to take a picture of a person standing in front of the system on a button press and send JSON requests back to the PI from AWS to validate users above a certain facial match percentage. On success, the owner of the Raspberry PI can program their device to take any action they would like.
* Ability to locally store images of all users who attempted to use system on SD card

**Website**

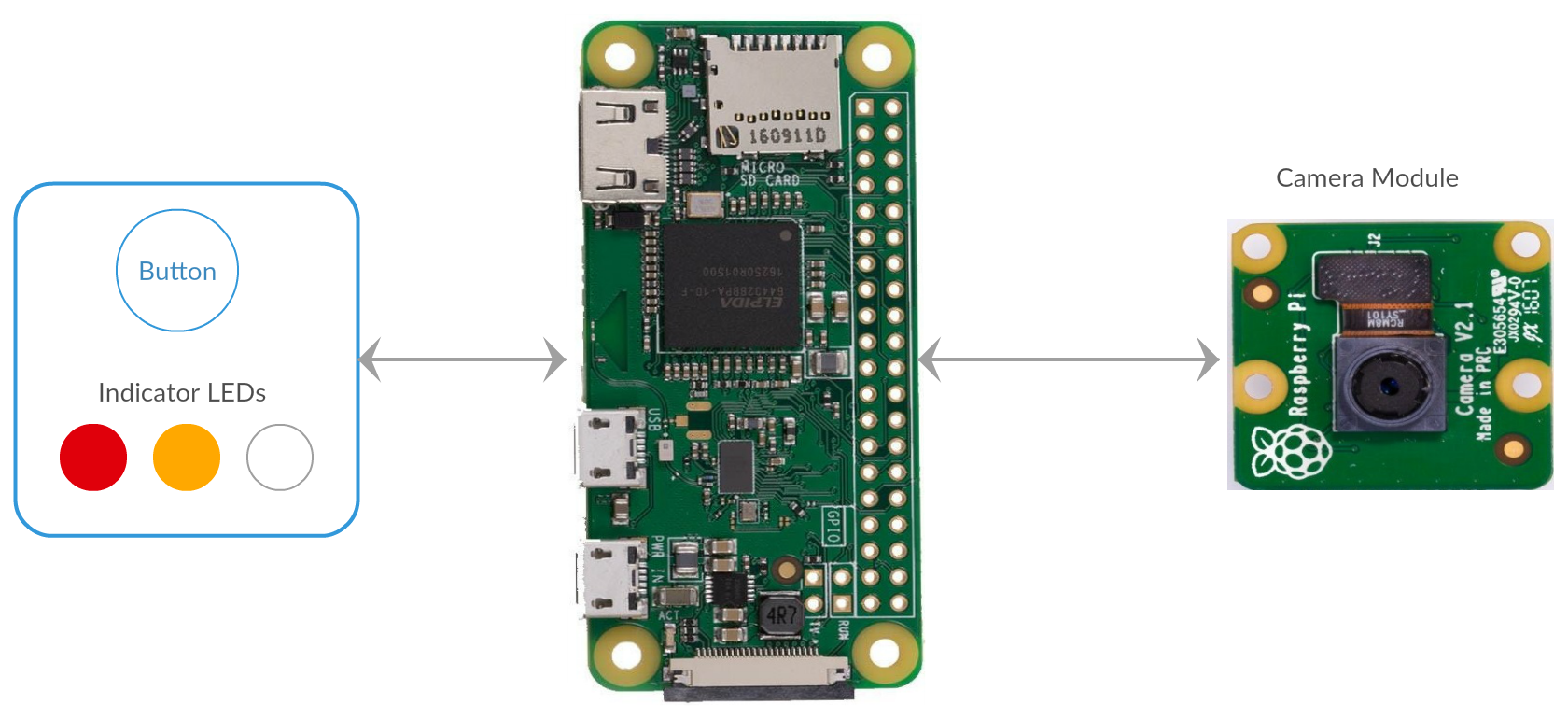
* Website will allow users to sign-up and log in to their accounts.
* User will be assigned a unique token to their account for easy identification
* Users have the capability to upload the pictures as well as deleting pictures for recognition service
  + Uploaded picture will be use as references to compare whoever rings the doorbell and getting scanned
    - If the person is recognizable, he or she will have the access to get in the property
    - If the person is not, he or she will not have the access to get in property
  + Deleted picture will be removed from the database. He or She will no longer has access to the property

## ***2.2 Limitations***

* High cost to scale and maintain with AWS
* Raspberry Pi Zero W specs may not be enough for mainstream use
* User requires some technical skill to be able to add various modules to their PI after a successful face validation

# **3) Solution Design**

## ***3.1 Raspberry Pi***



This section describes the basic functionality of the Raspberry Pi (Pi). There are three parts to this setup: The Pi, the camera module and the Input/Output (IO) device.

The Pi is the driving force of this project. It controls both the camera module and the IO device using a programming language called Python. Python is a scripting language. The Python script used modules to both control the camera module and IO device as well as communicate with the Amazon server.

The camera module is used to take the target photo that will be compared to the stored photos on the Amazon server.

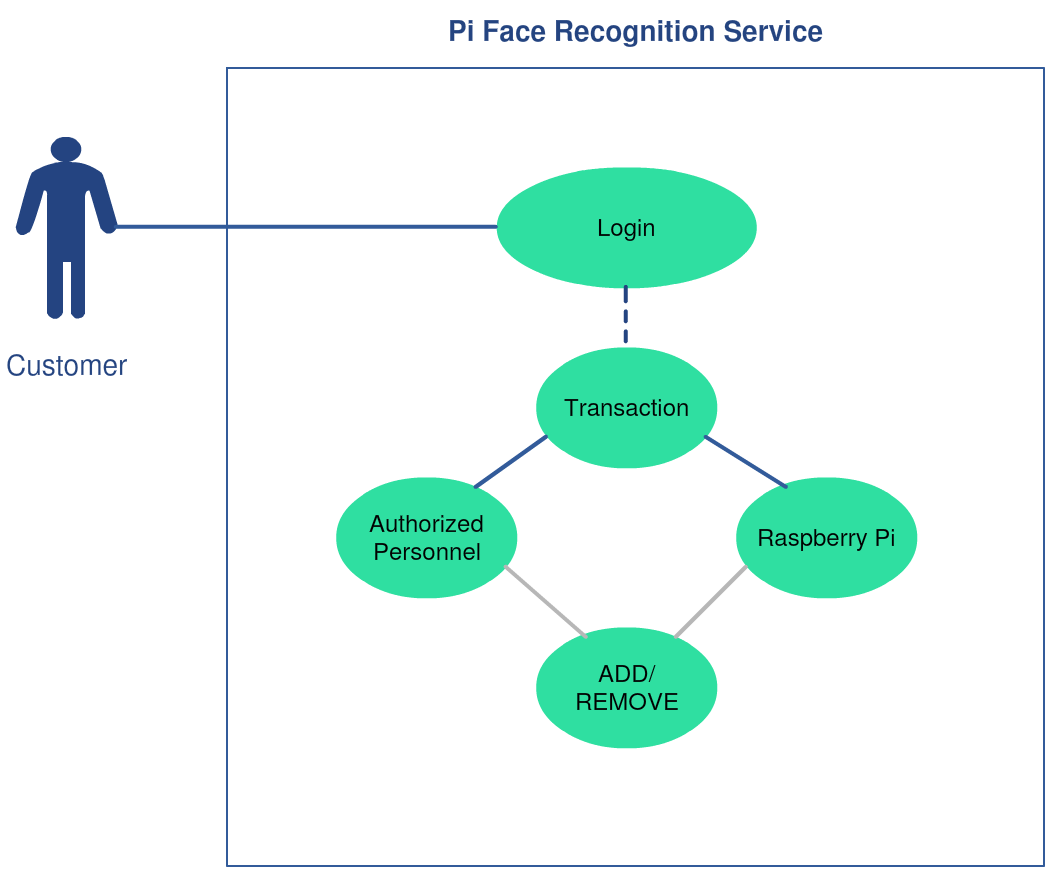
The IO device is connected to the Pi through the General Purpose Input Output (GPIO) pins. The GPIO pins allow a Pi to physically control electronic components. Our instance consists of two types of components: a button and three indicator Light Emitting Diodes (LED). Each color of LED indicates a different state of the process:

* Solid Red = Standby
* Flashing Amber = Processing
* Solid White = Success
* Flashing Red = Failure

A button press allows the unit to move from the standby state to the processing state. From that point on, a state change is made depending on the success or failure of the image processing.

In order to communicate with the Web Application’s endpoint the Python script utilizes the requests module which allows the sending and receiving of information to and from an endpoint.

## ***3.2 Web Application***



Login Screen

* Prompt user for email and password information to log into system
* Pass entered information to Database for verification
* Sign user in upon successful verification, reject input on unsuccessful verification

Registration Screen

* Prompt user for name, email, and password
* Validate name, email and password against regular expressions
* Pass to Database to store as customer and hash password for security

My Profile

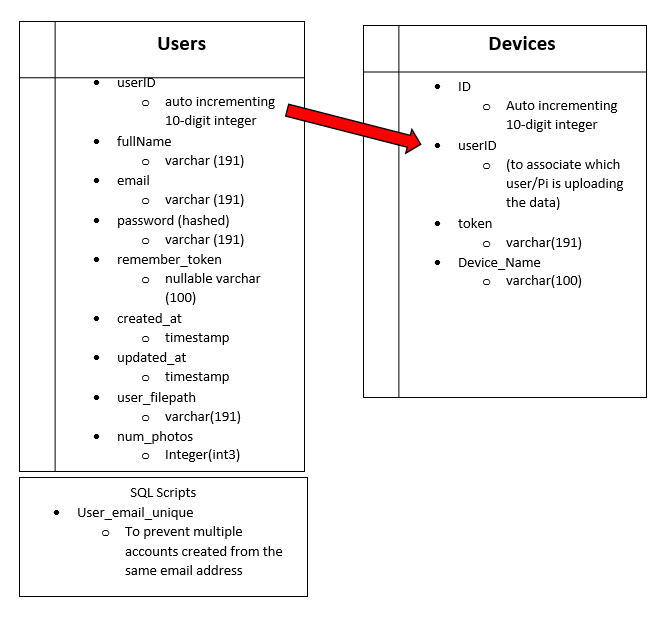
* Display connected devices to account
* Prompt user to upload images into account and stores it on the server
* Prompt user to add device to account which then generates a token for user to use

## ***3.3 Backend***

### ***3.3.1 AWS Rekognition***

AWS Rekognition is a image and video analysis service based on deep learning learning technologies and is focused on object, facial, and emotion detection. PFRS utilizes Rekognition’s facial comparison feature through the provided API to determine the similarity between faces in different images.

### ***3.3.2 Database***

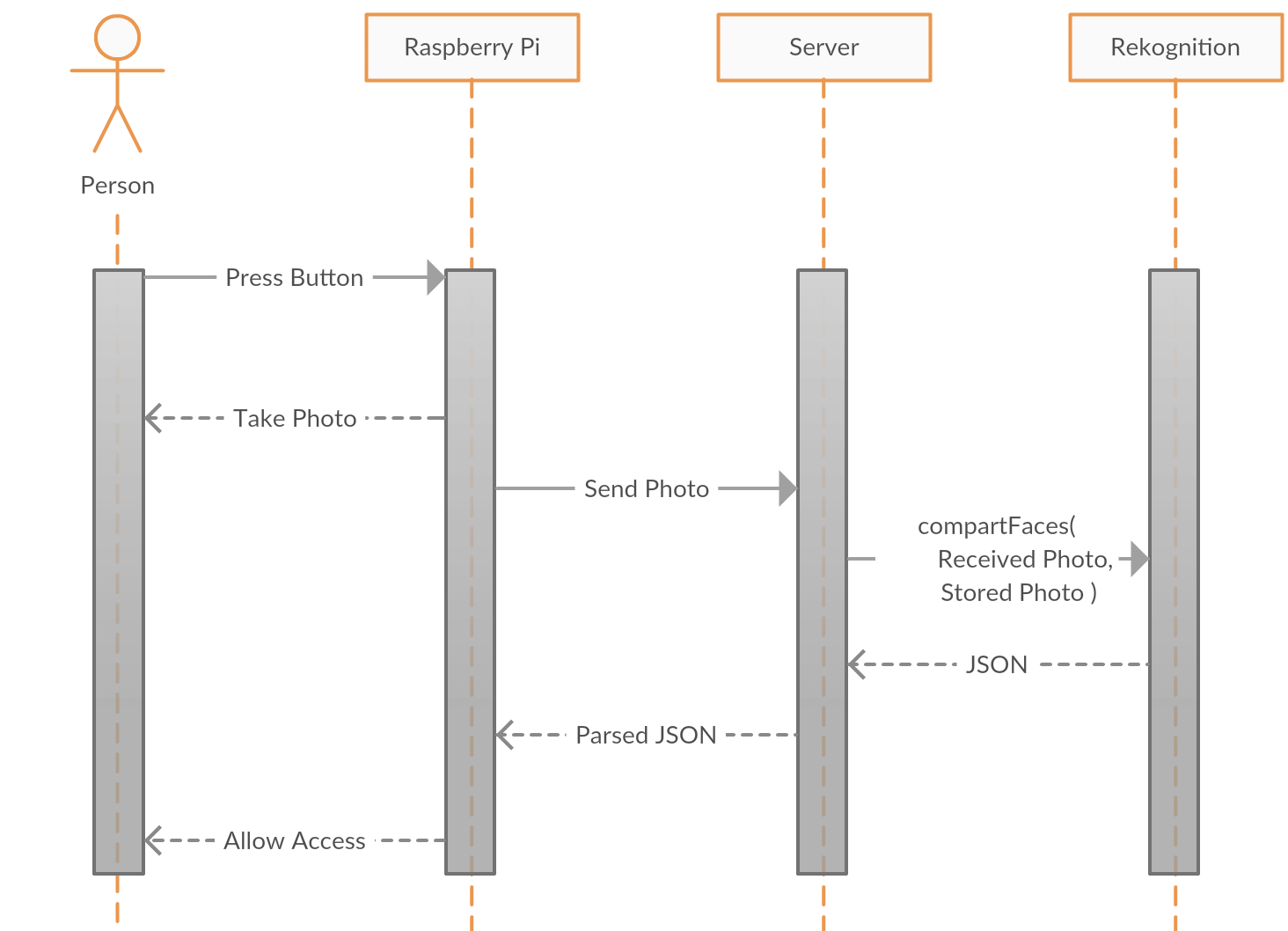


The project will contain two tables in our database. The first table, Users, contains all the user info when someone creates an account on our website. They are given a unique userID (which will be used to associate whose data is stored in our other table, UserData), their name, a unique email (which the script “user\_email\_unique” checks for), a password hashed (we don’t store passwords in plaintext), a remember user token, and the time of account creation and account updated. In addition, each user is given a folder on the server where their photos will be stored, which the full file path is saved to this table under user\_filepath. Finally, each user has a limit of their number of reference photos, which is kept as an integer also in this table. In the second table, devices, contains information about each device (Pi) registered to the PFRS. Each device contains an ID in the table, and references the same user\_ID number of the user account. A token is also generated when the device is added, to confirm a valid Pi is sending data to the server. Finally, the device can be given a name when it is registered.

### ***3.3.3 RESTful API***

* ~~~/register
  + User reaches this endpoint when they click the register button on the homepage. This directs them to a registration form which when the register button is clicked, creates a new user in the table with the information provided, as long as the email provided is unique.
* ~~~/login
  + User reaches this endpoint when they click the login button on the homepage. This directs them to a login page which when the login button is pressed, validates that the email address is already in the users table, and that the password matches what was the password for that user.
* ~~~/password/reset
  + User reaches this endpoint when they click the forgot password button on the login page. This directs them to a page where they can enter their email associated with their account. When they click send password reset link, this sends an email to their mailbox with instructions to reset the password.
* ~~~/face\_validation
  + This will be the key talking point between the Raspberry Pi and the server. When a user specifies that a new device will be added to their account, a token will be generated and visibly displayed on the user’s profile page. This token will be used on the script that will run on the PI which will then access this page when sending photo data between the server. This token will be validated before use and on success, will proceed to allow the user to use the Pi as intended. With a valid token and links to 2 images, the user can make direct calls using their browser to receive a JSON response from the server.

## ***3.4 Prototype***

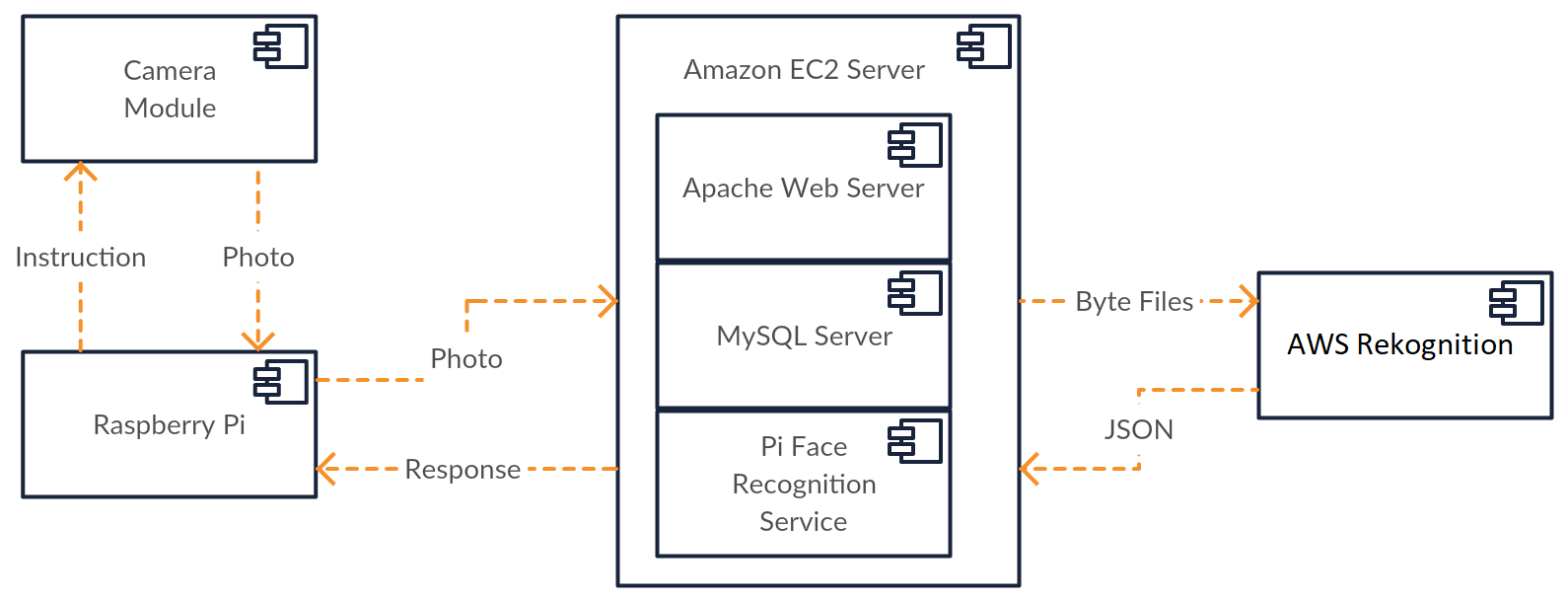


The prototype will demonstrate the basic functionality of PFRS. Although, there should be no difference between the Raspberry Pi setup at the prototype stage and the setup at the final stage of this project.

The Pi will take a photo of the user when a button is pressed and send it to the and endpoint on our EC2 server which will then send that image and a stored image to AWS Rekognition and either indicate that the person is allowed or denied access depending on the results.

The only difference between this stage and the final stage will be the Web Application and the ability that determines which account the Pi sending the photo belongs to.

# **4) Technical Architecture**



## ***4.1 Tech Stack***

### ***4.1.1 Raspberry Pi***

The Raspberry Pi used is the Zero W which has the following specifications:

* 1 GHz, Single-Core CPU
* 512MB RAM
* CSI Camera Connector
* HAT-compatible 40-pin Header
* 802.11n wireless LAN

The Operating System (OS) installed on the Zero W is Raspbian, which is Raspberry Foundation’s officially supported OS.

The camera module used is the second version of the official module developed by the Raspberry Foundation. It includes a Sony IMX219 8-megapixel sensor.

### ***4.1.2 Amazon EC2 Server***

The Amazon EC2 Server used to host the website runs a LAMP stack:

* Linux
* Apache2
* MySQL
* PHP

The server has the following specifications:

* Intel(R) Xeon(R) CPU E5-2676 v3 @ 2.40GHz, 2400 MHz CPU
* Cirrus Logic GD 5446 GPU
* 1 GB RAM
* Ubuntu 16.04.4 LTS OS

## **5) Roles and Responsibilities**

* *Raspberry Pi functionality/Rekognition functionality***:**
  + Dimitrios Topalis
* *AWS/Rekognition functionality/Linux Instance/Scrum Master***:**
  + Patrick Jacaszek
* *Website Design/PHP/SQL Coding***:**
  + Peter Lieb
  + Joseph Dementri
  + Samuel Liebe-Randall
  + Thein Do