

Facial Recognition Access Control System Design Manual

by

Hoda Ahmed;
Student ID: C00214991

April 20, 2020

Abstract

Companies and organizations often overlook the importance of physical security to equipment rooms due to the increase of cybercrime and the emphasis that cybersecurity industries put on cybercrime. This report discusses the importance of physical security to equipment rooms and proposes a solution to fix this flaw: The Facial Recognition Access Control System. Facial recognition is the future, and it has already started being implemented in most places for security, criminal identification, law enforcements and even social media where users can upload photos of other people and tag them.

Contents

1	Introduction	1
2	Project Plan	2
3	System Architecture	3
3.1	Hardware Components	4
4	Class Diagram	5
5	User Interface	6
5.1	Raspberry Pi Desktop Application	6
5.1.1	Main Menu	6
5.1.2	Login Screen	7
5.1.3	Registration Screens & Flow	7
5.2	FRACS Web Application	8
5.2.1	Dashboard	9
5.2.2	User List	9
5.2.3	Blacklisted Users List	10
5.2.4	Account Settings	10
6	Database Design	11
6.1	Users Table	11
6.2	Logs Table	12
7	Detailed Use Cases	13
7.1	Login	14
7.2	Register User	15
7.3	Capture Face	16
8	System Sequence Diagrams	17
8.1	RPi Login Sequence Diagram	18
8.2	RPi User Registration Sequence Diagram*	19
8.3	Capture Image Sequence Diagram	21
8.4	Log Access Event Sequence Diagram	22
8.5	Web Application Login Sequence Diagram	23
8.6	Web Application Logout Sequence Diagram	24

1 Introduction

This Design Manual Document is a document that provides detailed information on how the facial recognition access control system will function and what the expected behavior of it will be. This document will provide an architecture overview, database design, detailed use cases and system sequence diagrams. The purpose of this document is to give the reader a deep understanding of how the entire system provides its functionality specified in the functional specification document.

2 Project Plan

A project plan, also known as a project management plan, is a document that contains a project scope and objective.[1]

For the plan for FRACS, an estimated gantt chart is provided to outline the dates for the deliverables of the project.

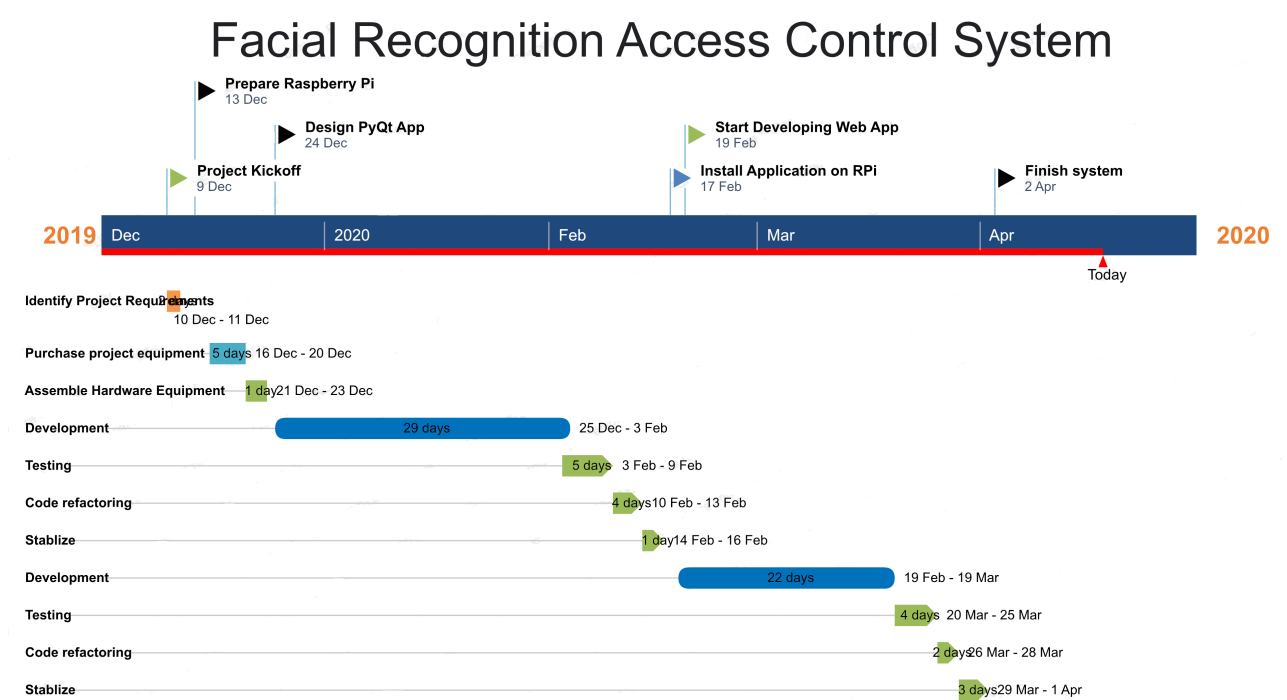


Figure 1: FRACS System Architecture

3 System Architecture

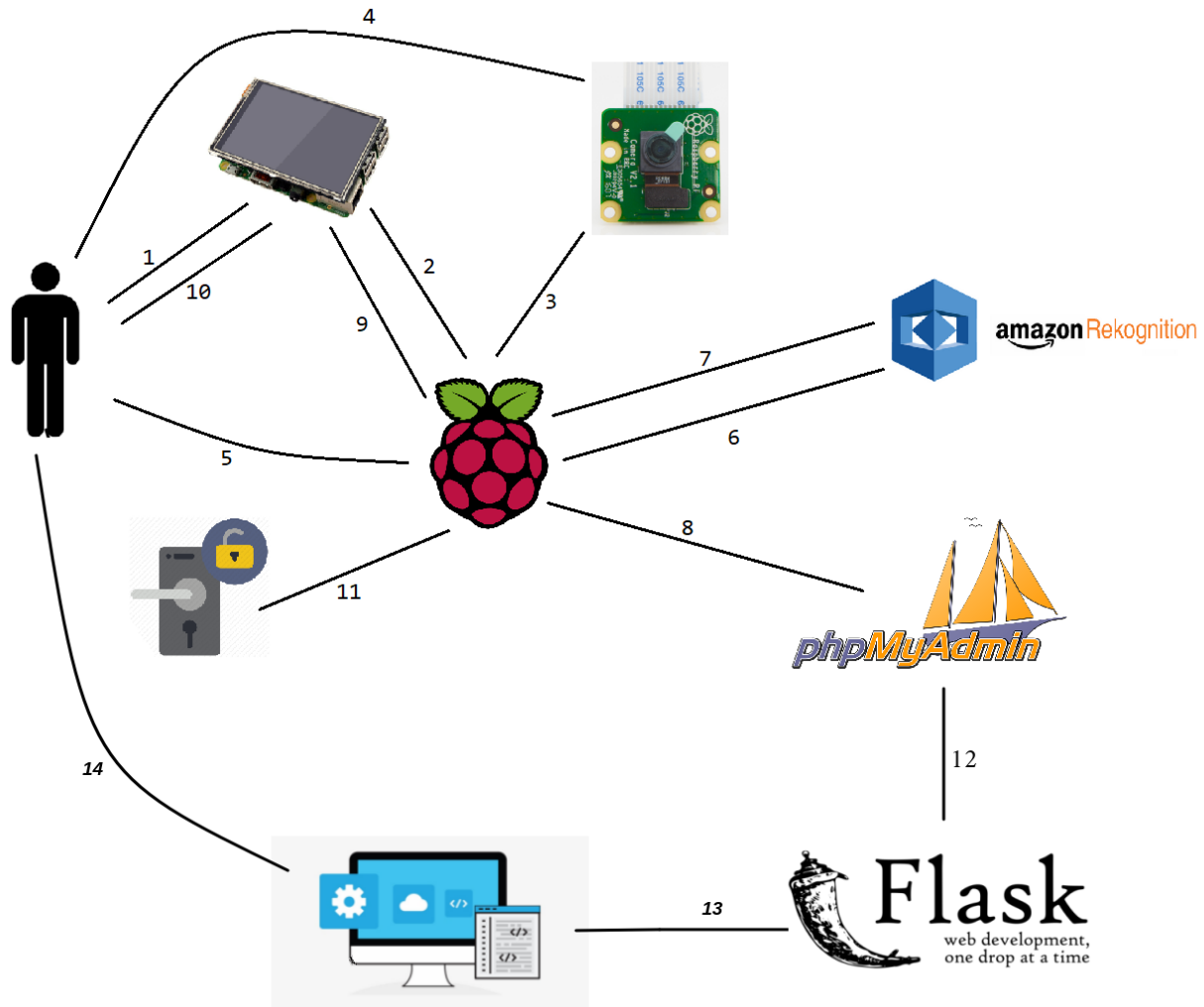


Figure 2: FRACS System Architecture

The facial recognition access control system is centered around the Raspberry Pi which is situated in a strategic position in the system. Most of the interactions of the system are carried out with/on/using the Raspberry Pi.

As outlined in the diagram above, the user interacts with the LCD touchscreen to trigger the starting of the desktop application on the Raspberry Pi and logs in by entering their credentials. Following this, the camera is activated, and it takes a picture of the user. That picture is then stored on the Raspberry Pi which carries out the API calls to AWS Rekognition, sending the picture as a HTTP request. Once the API received the HTTP request, it detects the face in the image, extracts the facial features and stores it in a vector. That vector is compared against the existing vectors in the database to check whether a user exists (and is authorized to enter) or not. The result is returned to Raspberry Pi as a HTTP response, and it contains a face ID of the person. Once it receives the face ID, the Raspberry Pi checks for it in the local database along with the credentials entered at the start by the user. If the face ID/credentials combination exists, the Raspberry Pi

sends a signal to the lock, triggering it, and unlocks the room for the user. The Raspberry Pi also records a new access entry in the logs database. However, if the face ID/credentials combination doesn't exist, then the user is denied access and a failed access attempt is also logged.

3.1 Hardware Components

The following diagram is a simple depiction of how the circuit will be for the Raspberry Pi to control the lock and trigger it when a user is authenticated and authorized to gain access.

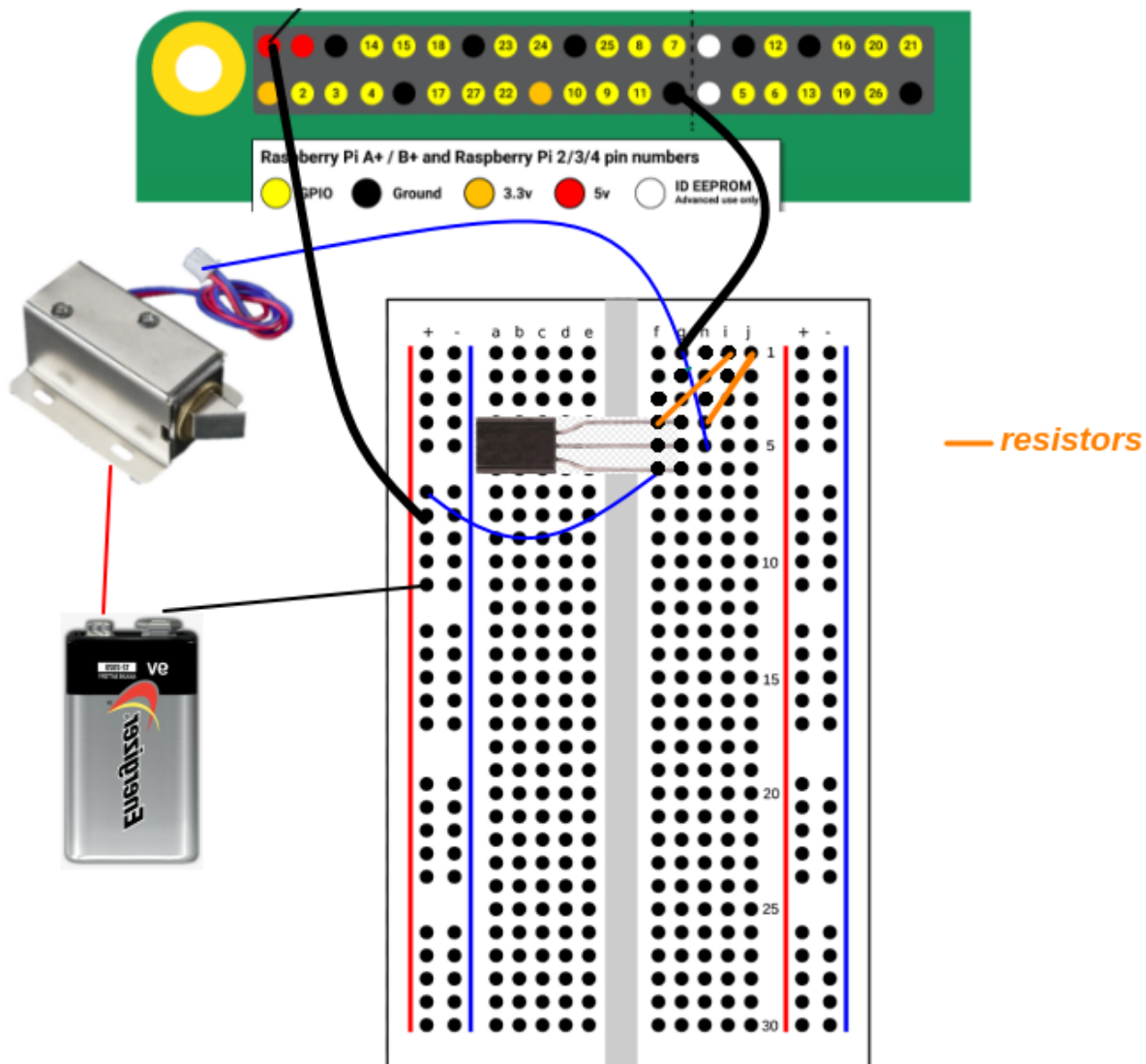


Figure 3: FRACS System Circuit

The solenoid lock will need 6 volts to operate, but the maximum voltage that the Raspberry Pi can provide as output is only 5V. Therefore, the 9V battery is used to give power to the lock to pull back or release its tongue. The breadboard is used to easily connect each electrical component and wires together, and to make it easier to visualize the flow of current.

4 Class Diagram

"A UML Class Diagram gives an overview of a software system by displaying classes, attributes, operations, and their relationships. This Diagram includes the class name, attributes, and operation in separate designated compartments." [2]

A class diagram is essential, as it can be a very informative diagram before the development of any system. It not only delivers the idea to system owners on how the system will operate and how each component will interact with the other, but it also gives the developer a few good hints as to what they will be developing.

The following class diagram is designed for the FRACS system.

FRACS Class Diagram

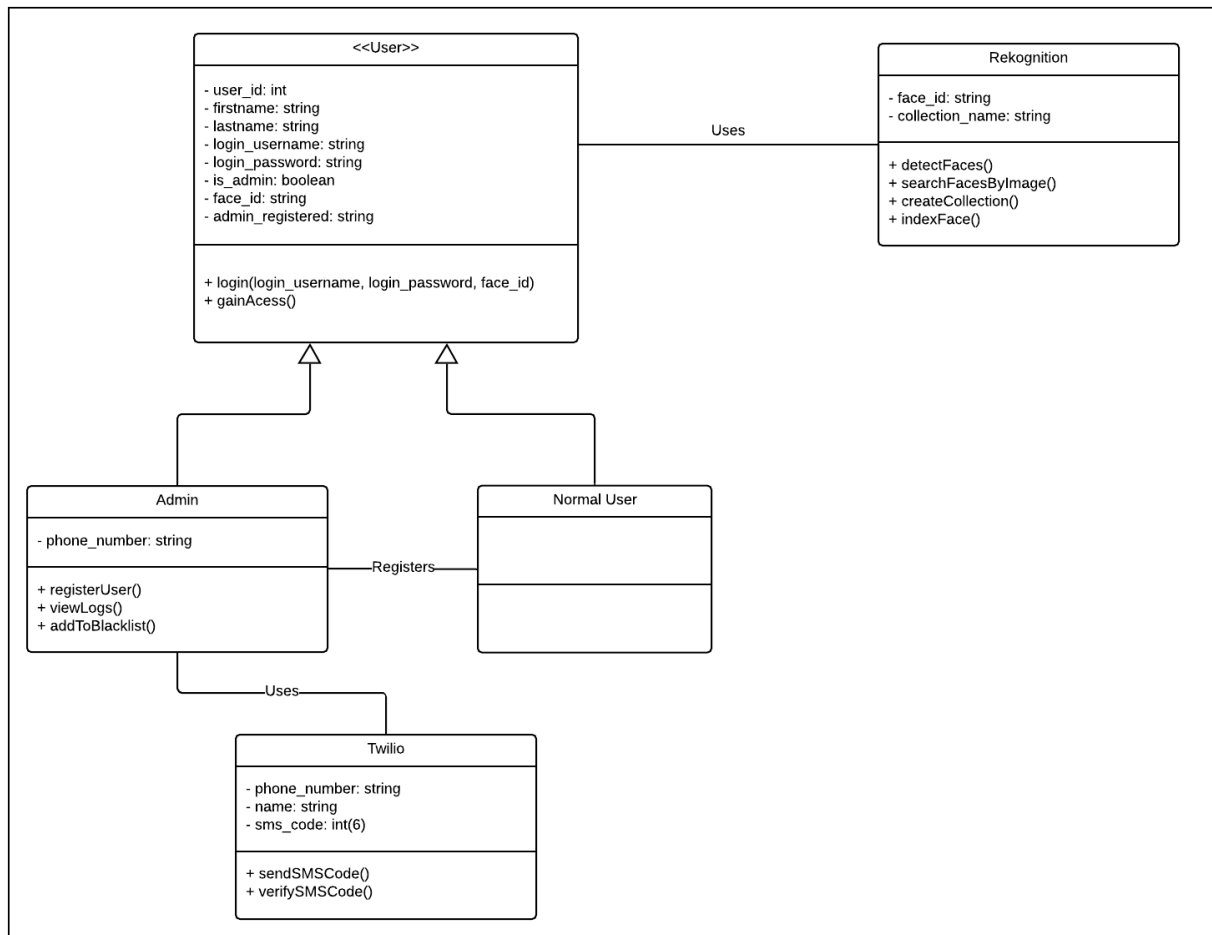


Figure 4: FRACS Class Diagram

5 User Interface

"The ability to pick up a gadget and do things without overthinking how it works is not only a good thing, it's what buyers have come to expect." [3]

Keeping this quote in mind, interface design should be as simple as possible, informative and easy to use.

The Raspberry Pi desktop application will have simple colours to reflect positively and give a more user-friendly atmosphere.

The web application will be attempted to be responsive, meaning it can be used on any device capable of running a browser. A few design possibilities are provided below.

5.1 Raspberry Pi Desktop Application

The following diagrams are simple prototypes of how the Raspberry Pi desktop application should look like and what the flow will be.

5.1.1 Main Menu

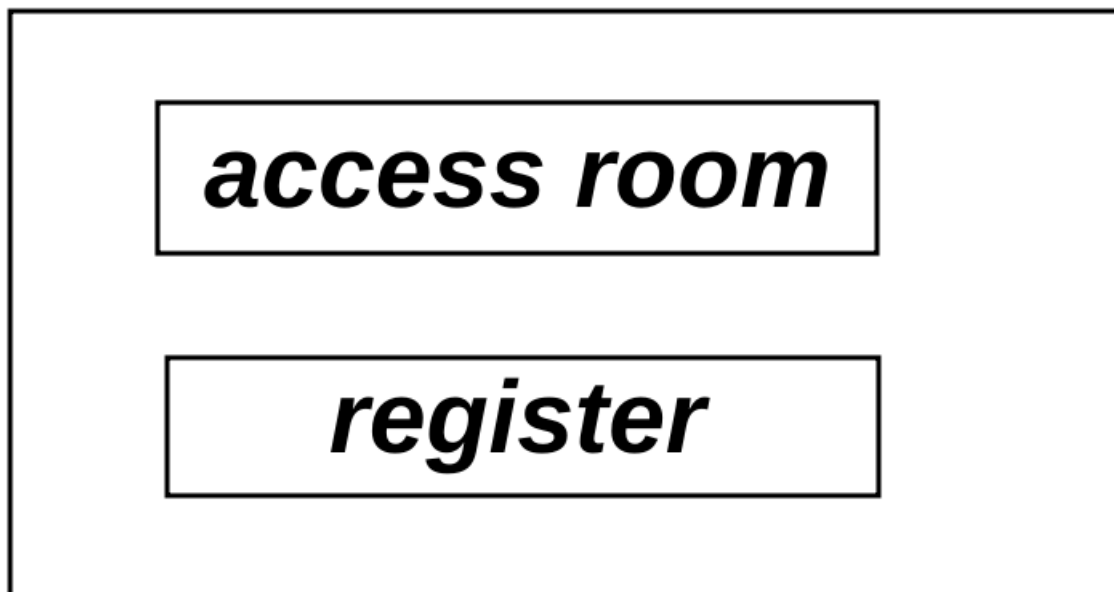


Figure 5: Raspberry Pi Main Menu

5.1.2 Login Screen

access room

username

password

login

Figure 6: Raspberry Pi Login Screen

5.1.3 Registration Screens & Flow

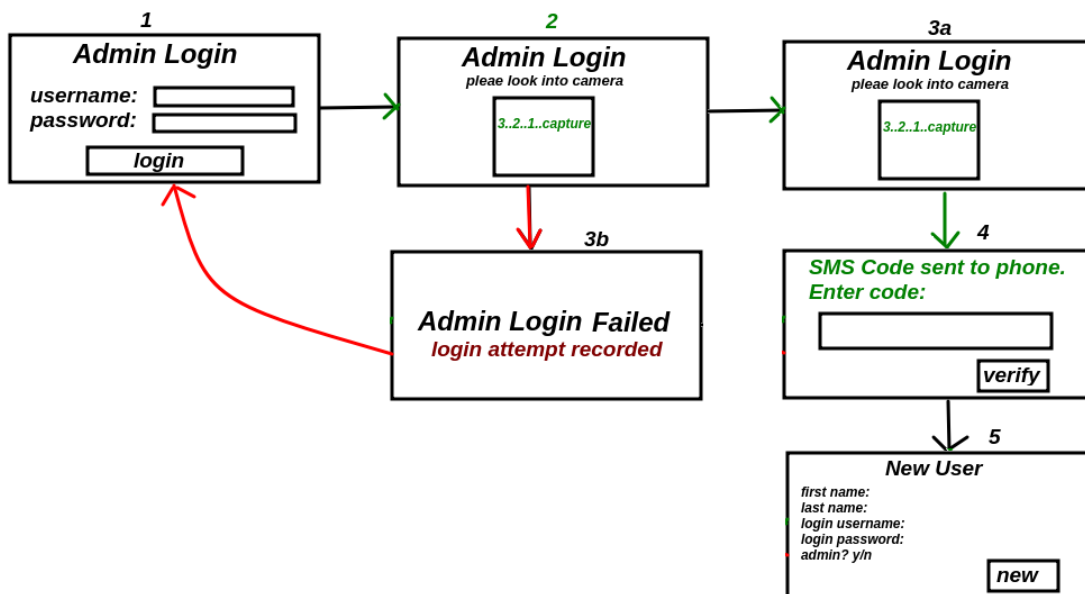


Figure 7: Raspberry Pi Registration

5.2 FRACS Web Application

The following diagrams are simple prototypes of each page of the facial recognition access control system web application. Like the Raspberry Pi desktop application, the design will be simple to avoid over-complexity and achieve good user experience.

5.2.1 Dashboard

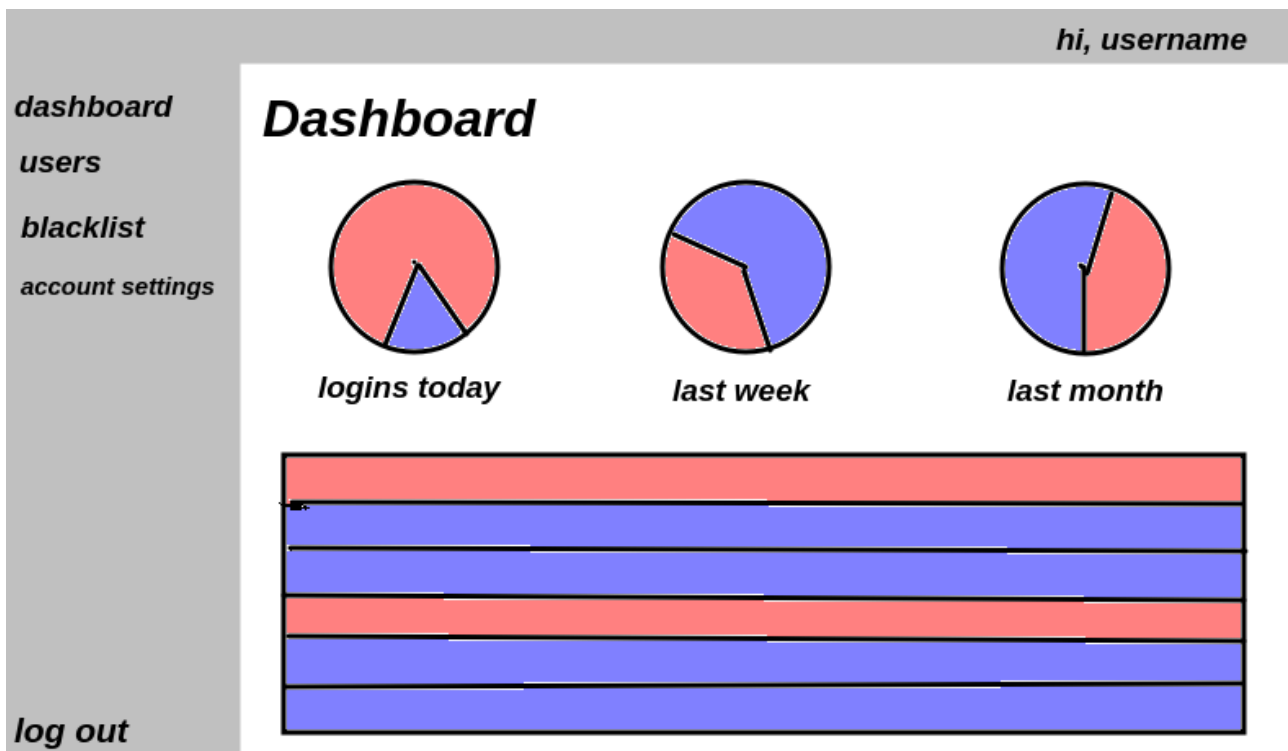


Figure 8: Web Application Dashboard

5.2.2 User List

The Users list interface features a grey sidebar on the left with navigation links: **dashboard**, **users**, **blacklist**, **account settings**, and **log out**. The main content area has a grey header with the text **hi, username**. Below the header, the title **Users** is displayed. A search bar with the placeholder text **search** is located in the top right corner. Below the search bar is a table with 10 rows and 5 columns. Each row has a green triangle icon in the first column. The table is scrollable, as indicated by the scrollbar on the right side.

▲				
▲				
▲				
▲				
▲				
▲				
▲				
▲				
▲				
▲				

Figure 9: Web Application Users List

5.2.3 Blacklisted Users List

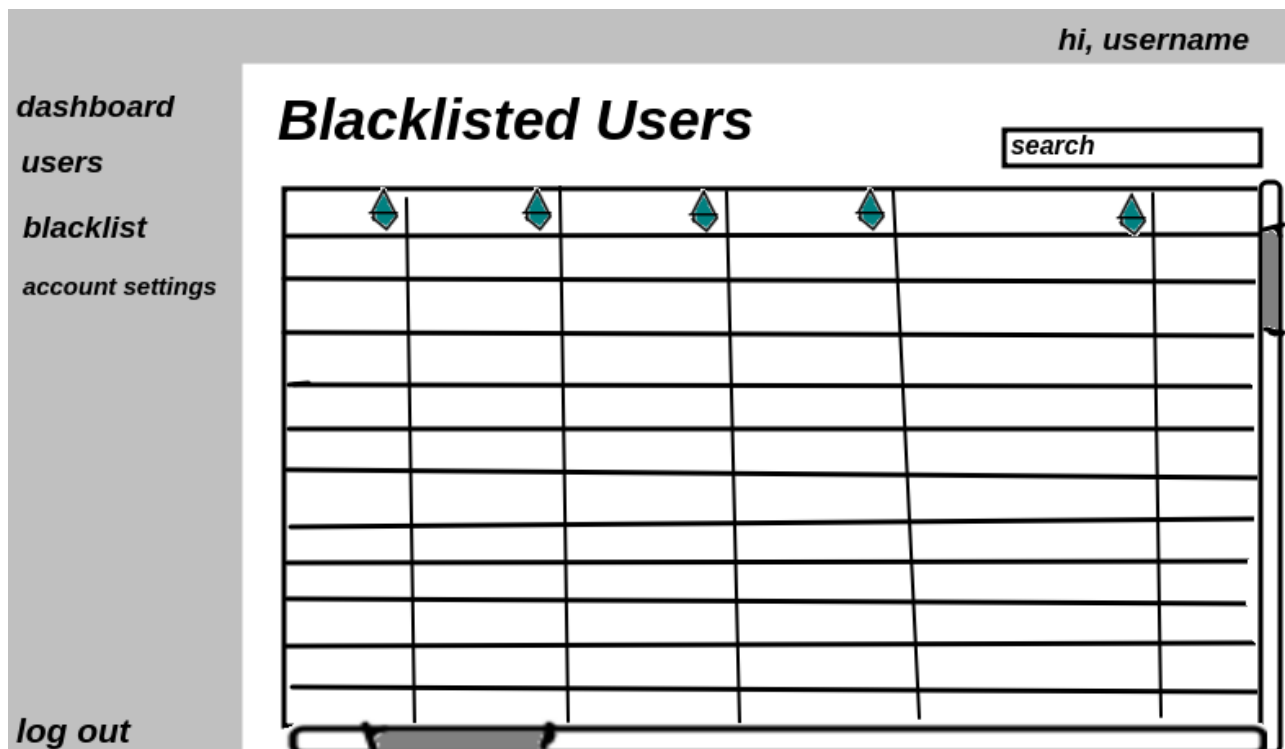


Figure 10: Web Application Blacklisted Users List

5.2.4 Account Settings

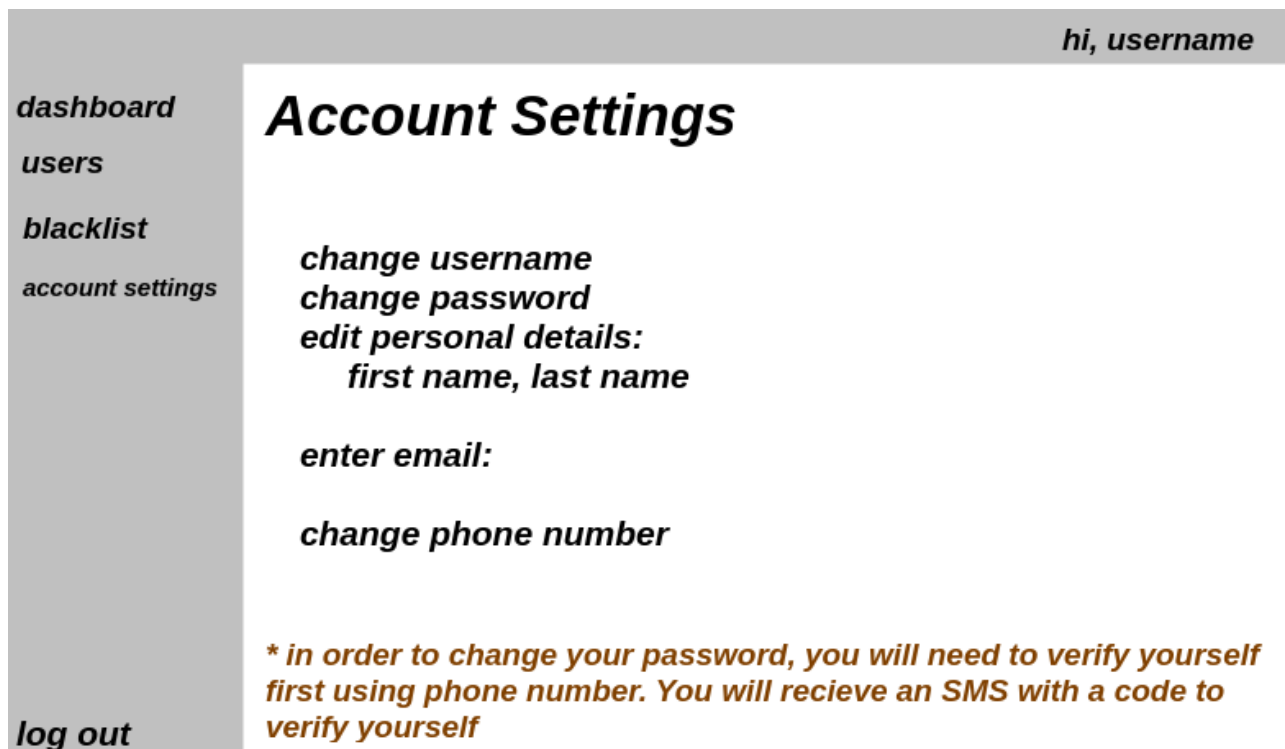


Figure 11: Web Application Account Settings

6 Database Design

The database created for this system is relatively simple. There will only be one database for both the desktop application and the web application.

6.1 Users Table

Database Name: *FracsDB*

Tables Names: *fracs_users*

Description: *Used to store user information to access equipment rooms and web application.*

Table Structure & Sample Data:

```
mysql> describe fracs_users db_help
-> ;
```

Field	Type	Null	Key	Default	Extra
user_id	int	NO	PRI	NULL	auto_increment
firstname	varchar(255)	NO		NULL	
lastname	varchar(255)	NO		NULL	
login_username	varchar(255)	NO		NULL	
login_password	varchar(255)	NO		NULL	
is_admin	tinyint(1)	NO		0	
face_id	varchar(255)	NO		NULL	

```
7 rows in set (0.00 sec)

mysql> 
```

Figure 12: Users table design

```
mysql> select * from fracs_users;
```

user_id	firstname	lastname	login_username	login_password	is_admin	face_id
1	hoda	ahmed	hahmed	password123	1	abcdefghijklmnop
2	emma	watson	ewatson	passw0rd123	0	qazwsxedcrfvtgby
3	mary	oconnor	moconner	PaSsw0rd123	0	plmokiujnhybujmo
4	john	david	jdavid	qwerty146	1	jdhevuagwlovfhqb
5	john	doe	jdoe	oiehjriof78	1	uy838vksjdhiuq3D

```
5 rows in set (0.00 sec)

mysql> 
```

Figure 13: Users table sample data

6.2 Logs Table

Database Name: *FracsDB*

Tables Names: *fracs_logs*

Description: *Used to store login information. Both successful and unsuccessful login attempts are recorded and stored here.* **Table Structure & sample data**

```
mysql> describe fracs_logs
-> ;
+-----+-----+-----+-----+-----+-----+
| Field          | Type          | Null | Key | Default | Extra          |
+-----+-----+-----+-----+-----+-----+
| log_id         | int           | NO   | PRI | NULL    | auto_increment |
| username_entered | varchar(255)  | NO   |     | NULL    |                |
| face_id        | varchar(255)  | NO   |     | NULL    |                |
| attempted_user_id | int          | YES  |     | NULL    |                |
| login_datetime | datetime      | NO   |     | NULL    |                |
| is_successful   | varchar(10)   | NO   |     | NULL    |                |
+-----+-----+-----+-----+-----+-----+
6 rows in set (0.00 sec)

mysql> 
```

Figure 14: Logs table design

```
mysql>
mysql> select * from fracs_logs;
+-----+-----+-----+-----+-----+-----+
| log_id | username_entered | face_id          | attempted_user_id | login_datetime      | is_successful |
+-----+-----+-----+-----+-----+-----+
| 1      | jdavid           | jdhevuagwlovfhqb | 4                 | 2020-04-11 18:41:12 | SUCCESS       |
| 2      | hahmed           | abcdefghijklmnop | 1                 | 2020-04-11 18:42:14 | SUCCESS       |
| 3      | ewatson          | qazwsxedcrfvtgby | 2                 | 2020-04-11 18:43:09 | FAIL          |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)

mysql> 
```

Figure 15: Logs table sample data

7 Detailed Use Cases

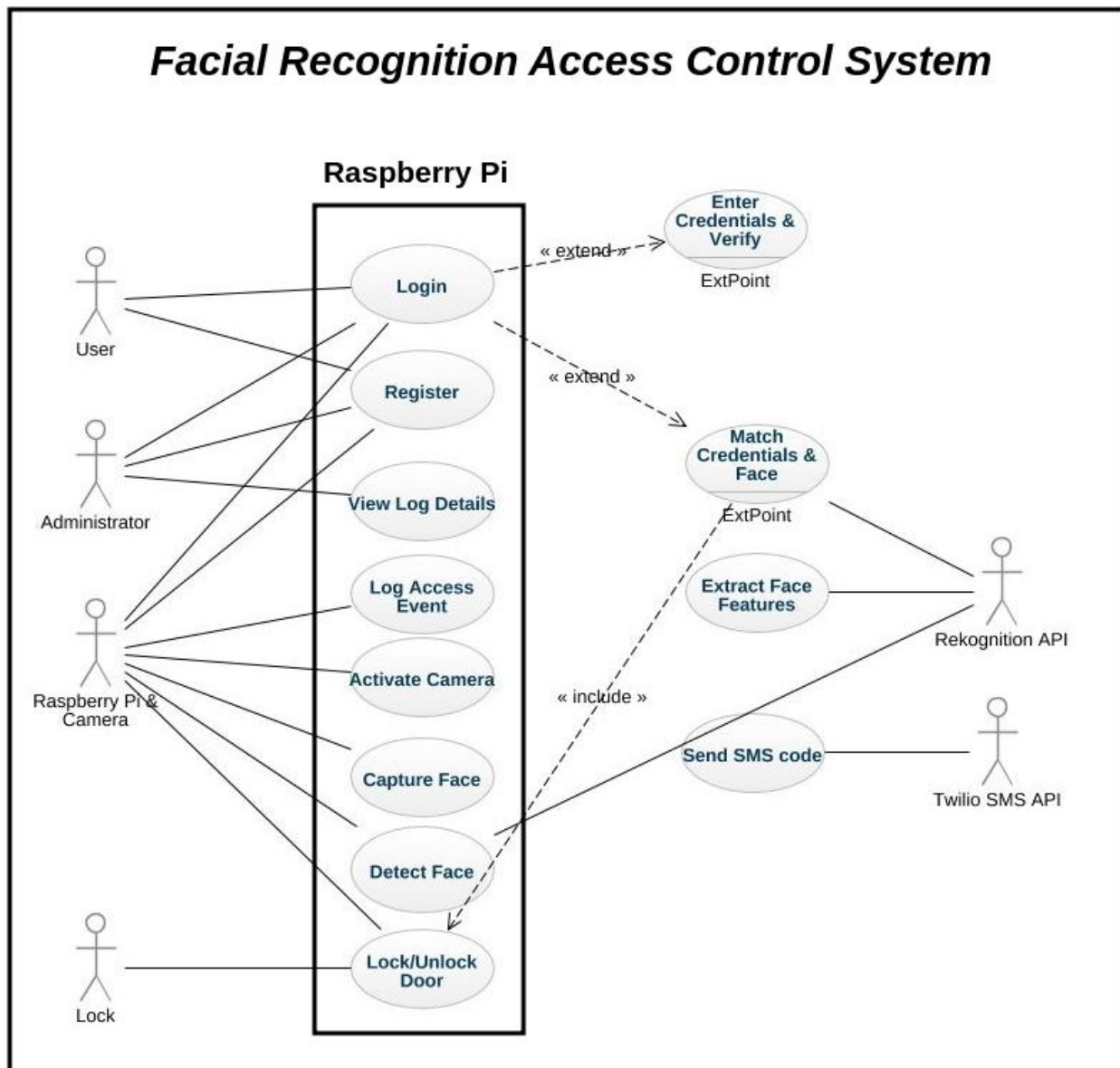


Figure 16: FRACS Use Case Diagram

7.1 Login

Actors Involved: Admin/User, Raspberry Pi (& camera), Rekognition API.

Description: This use case begins when a user wishes to login to the system to access the room. The user will enter their username and password, and then get their face scanned by the camera, and analysed by Rekognition API and then granted access into the room if they are authorized. The use case ends when the user is authorized to access the room and logs in successfully. Whether the login attempt was successful or not, it is logged for security purposes.

Main Success Scenario:

1. User clicks on Login button on desktop app.
2. User enters their username/password and clicks login.
3. Camera is enabled, countdown of 3 starts.
4. User's face is captured, image is processed by Rekognition API.
5. Credentials match user's face, door is unlocked.
6. Access log recorded to database.

Alternatives:

5a. Entered credentials don't match with user's face.

1. The system informs the user that the credentials they entered don't match with stored facial information and displays the Login Screen again.
2. The user re-enters their username and password, and then gets their face scanned.
3. Steps repeated until positive confirmation or if maximum attempts have been reached(5), freeze access for a period of time and log attempts.

7.2 Register User

Actors Involved: Admin, User, Raspberry Pi, Rekognition API.

Description: use case begins when the admin wishes to register a user to the system.

Main Success Scenario:

1. Admin clicks on Register button on desktop app.
2. Admin enters their username/password and clicks login.
3. Camera is enabled, countdown of 3 starts and then admin's face is captured.
4. Admin face and credentials are validated.
5. SMS code is sent to Admin's phone.
6. Admin enters code into Raspberry Pi. Raspberry Pi validates code.
7. Code is correct, user now enters their data in form on desktop application.
8. User clicks next. Raspberry Pi checks username availability.
9. Camera is enabled and user gets their face scanned, analysed and stored along with form data.
10. Registration is successful and user can now login and access room.

Alternatives:

4a. Admin entered incorrect credentials OR credentials don't match with face.

1. The web app informs the user that the credentials they entered don't match with stored facial information and displays the Login Screen again.
2. The admin re-enters their username and password, and then gets their face scanned.
3. Steps repeated until positive confirmation or if maximum attempts have been reached(3), freeze access for a period of time and log attempts.

6a. Admin entered incorrect SMS code.

1. The Raspberry Pi informs the admin that the SMS code they entered is not correct.
2. The system offers a chance to send new code (max. 3 times).
3. If all attempts used up, admin is blocked from attempting to register new user to system.
4. Raspberry Pi informs admin that in order to be unblocked, they need to login to web application once.

8a. User entered an unavailable username

1. The Raspberry Pi informs the user that the username they entered is not available. It also tells the user to check that they don't have an account already created.
2. The Raspberry Pi returns back the user information form for the user to modify

9a. User's face is already registered with the system.

1. The Raspberry Pi informs the user that they are already registered with the system as the face that was scanned exists in the database.
2. The Raspberry Pi returns back the user information form for the user to modify or exit to the main menu themselves.

7.3 Capture Face

Actors Involved: User/Admin, Raspberry Pi, Camera, Rekognition API

Description: Use case begins when a user has logged in and had their face scanned. It covers the period of time after they have entered their credentials and are getting their face scanned. This use case also takes place when a user is being registering to the system.

Main Success Scenario:

1. Camera is enabled
2. Face is captured and image is stored in Raspberry Pi.
3. Raspberry Pi sends HTTPS request to Rekognition API, including the stored image.
4. Image of face is examined and analysed using Rekognition API.
5. Response is received from Rekognition API.
6. Image is over-written twice with random bits (or with 0's) and is permanently deleted from Raspberry Pi. Over-writing makes it harder for an image (or any file generally) to be restored.

Alternatives:

2a. Problem occurred with camera trying to take picture.

1. The Raspberry Pi will notify the user of a system error and will redirect back to the login screen.
2. User will enter their username and password and try to get their face scanned again.

8 System Sequence Diagrams

Sequence diagrams are an essential part of every project plan that's ever put in place. They are interaction diagrams that show in minute detail how each operation in a system is carried out. They are time focused and they show the order of the interaction visually by using the vertical axis of the diagram to represent what messages are sent and when^[4].

8.1 RPi Login Sequence Diagram

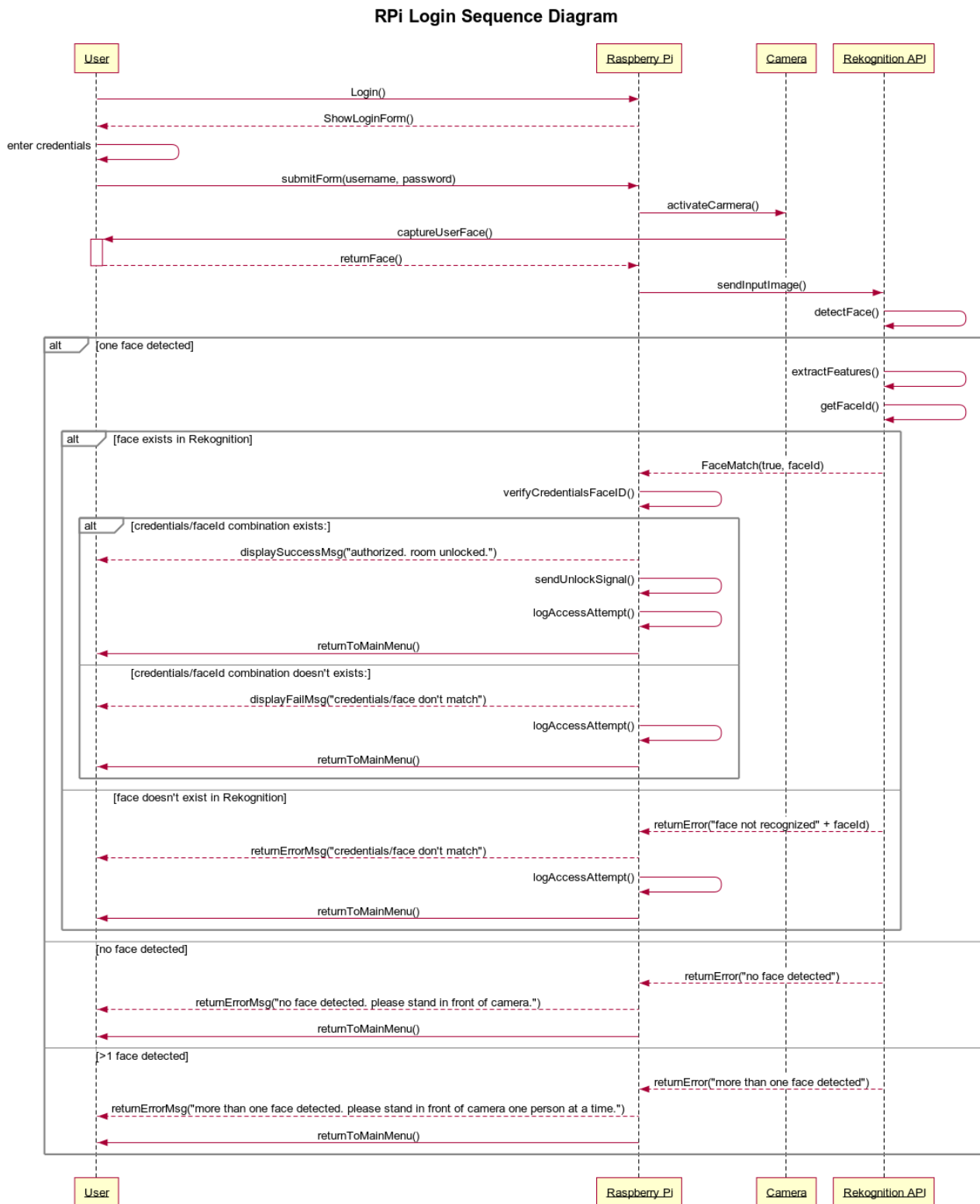


Figure 17: Facial Recognition Access Control System Login Sequence Diagram

8.2 RPi User Registration Sequence Diagram*

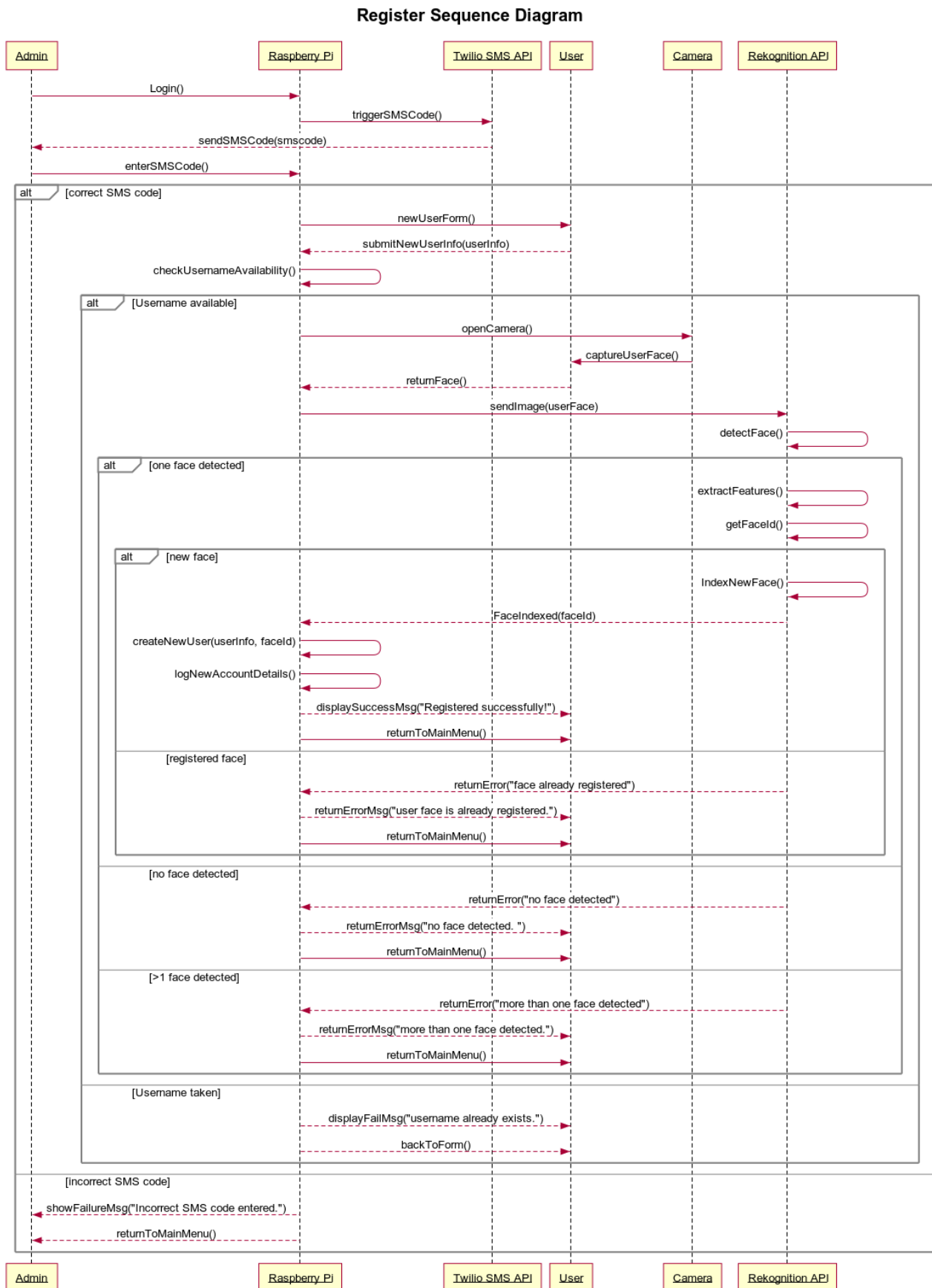


Figure 18: Facial Recognition Access Control System Registration Sequence Diagram

**For the user registration part of the Facial Recognition Access Control system, the admin needs to sign in to complete the registration process. Therefore, to avoid over-complicating the sequence diagram for Registration, it will use a reference to the previously defined Login sequence diagram. However, this reference will exclude the door unlocking functionality, and will move straight to the next point in the Registration sequence diagram.*

8.3 Capture Image Sequence Diagram

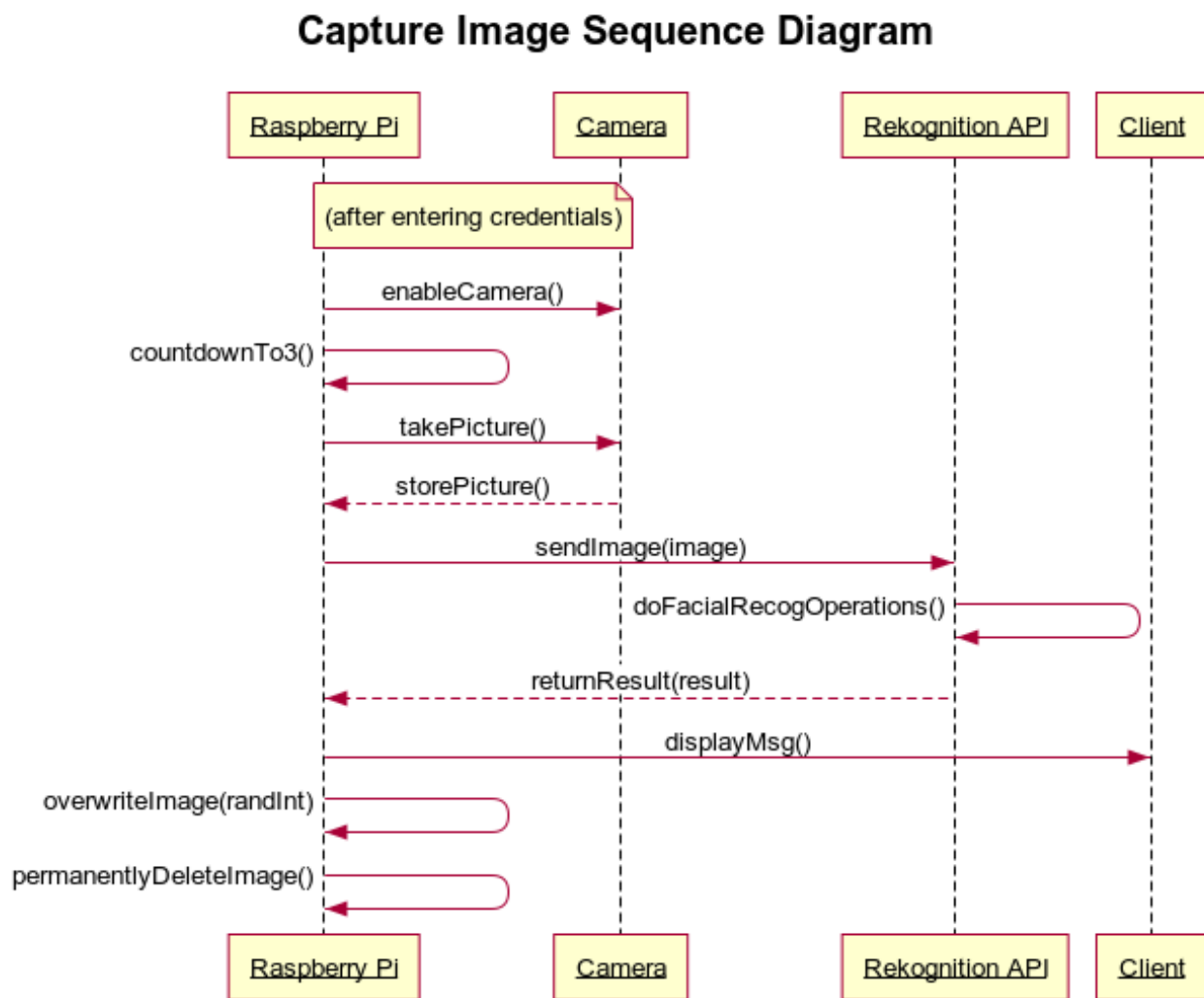


Figure 19: Facial Recognition Access Control System Capture Image Sequence Diagram

8.4 Log Access Event Sequence Diagram

Log Access Event Sequence Diagram

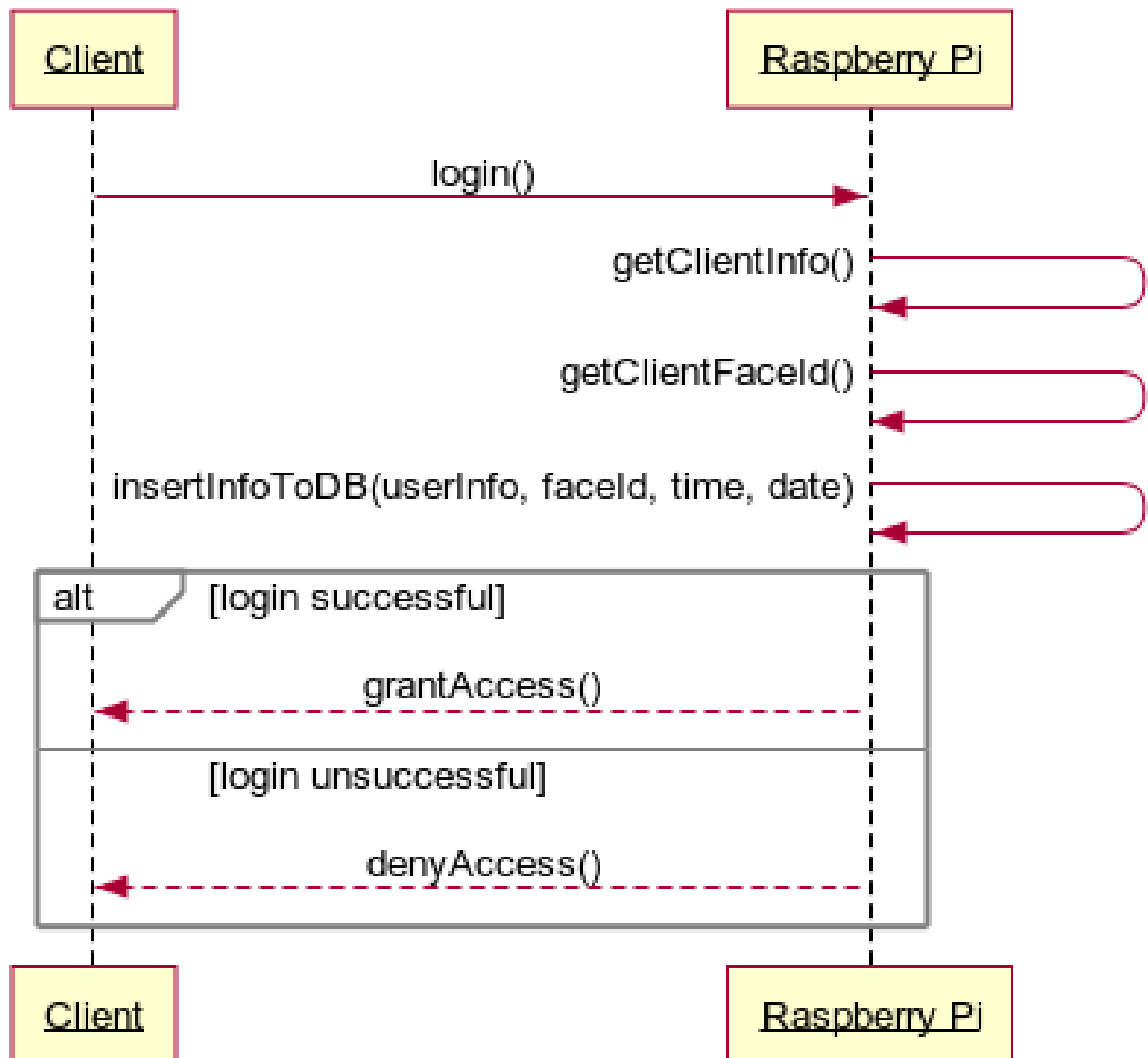


Figure 20: Facial Recognition Access Control System Log Access Event Sequence Diagram

8.5 Web Application Login Sequence Diagram

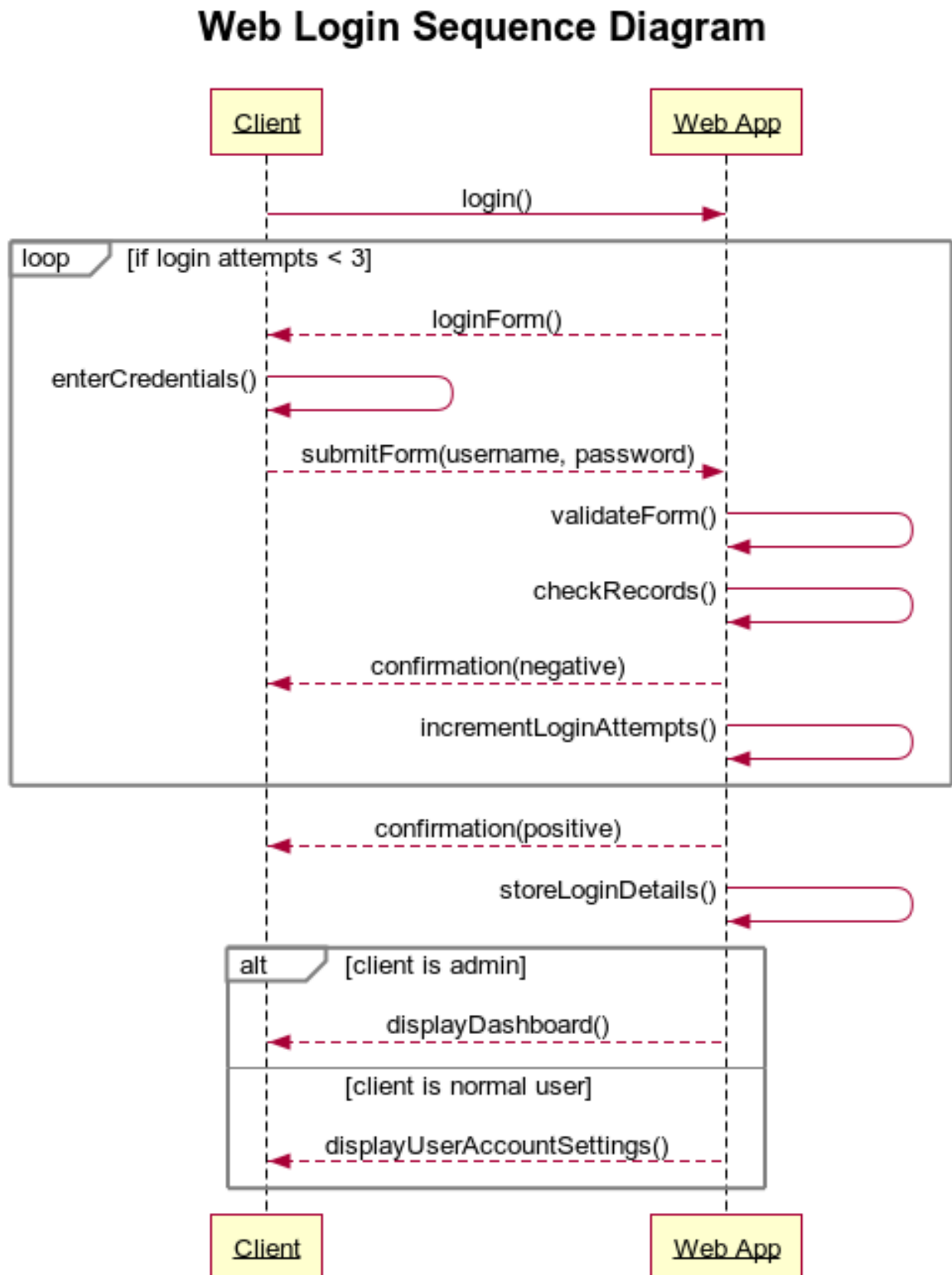


Figure 21: Facial Recognition Access Control System Web App Login Sequence Diagram

8.6 Web Application Logout Sequence Diagram

Web App Logout Sequence Diagram

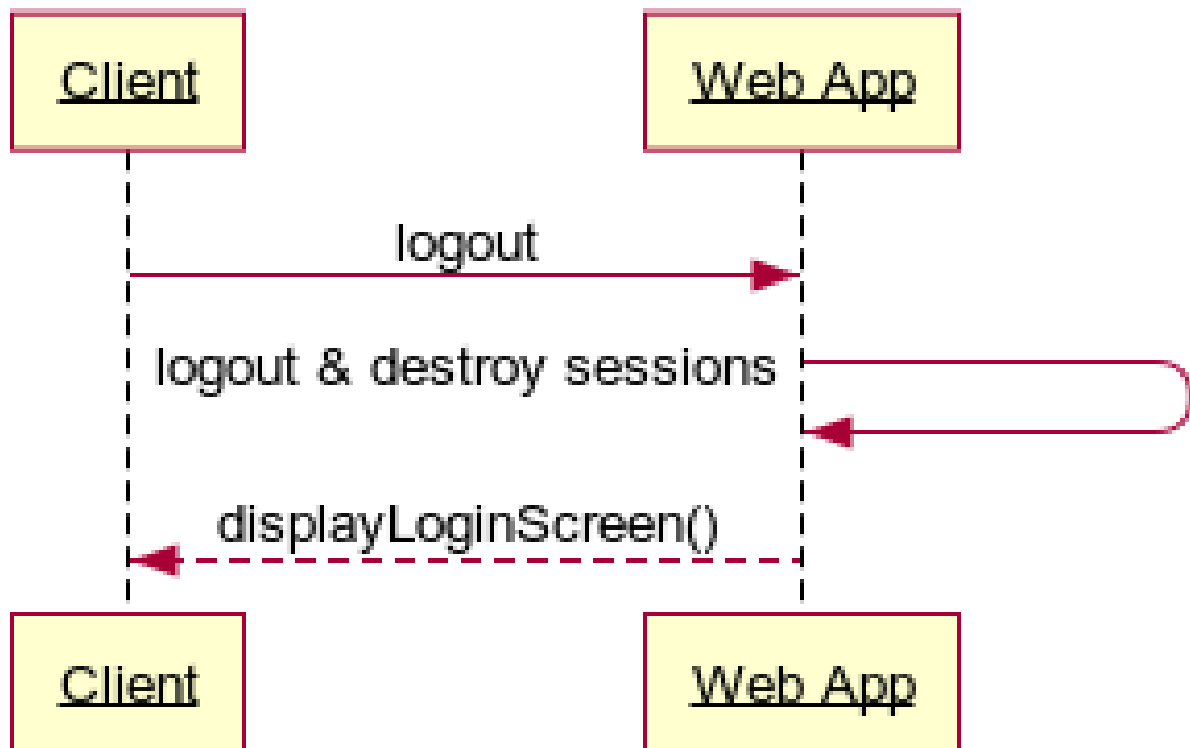


Figure 22: Facial Recognition Access Control System Web App Logout Sequence Diagram

References

- [1] Brett Harned. *How to Write a Good Project Plan in 10 Steps — Project Management Guide*. Teamgantt.com, July 2019. URL: <https://www.teamgantt.com/guide-to-project-management/how-to-plan-a-project>.
- [2] Guru99. *UML Class Diagram Tutorial with Examples*. Guru99.com, Dec. 2019. URL: <https://www.guru99.com/uml-class-diagram.html>.
- [3] James Kendrick. *3 rules for a good user experience (UX)*. ZDNet, July 2013. URL: <https://www.zdnet.com/article/3-rules-for-a-good-user-experience-ux/> (visited on 03/11/2020).
- [4] Visual Paradigm. *What is Sequence Diagram?* Visual-paradigm.com, 2019. URL: <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-sequence-diagram/>.

List of Figures

1	FRACS System Architecture	2
2	FRACS System Architecture	3
3	FRACS System Circuit	4
4	FRACS Class Diagram	5
5	Raspberry Pi Main Menu	6
6	Raspberry Pi Login Screen	7
7	Raspberry Pi Registration	7
8	Web Application Dashboard	9
9	Web Application Users List	9
10	Web Application Blacklisted Users List	10
11	Web Application Account Settings	10
12	Users table design	11
13	Users table sample data	11
14	Logs table design	12
15	Logs table sample data	12
16	FRACS Use Case Diagram	13
17	Login Sequence Diagram	18
18	Registration Sequence Diagram	19
19	Capture Image Sequence Diagram	21
20	Log Access Event Sequence Diagram	22
21	Web App Login Sequence Diagram	23
22	Web App Logout Sequence Diagram	24