Facial Motion Recognition System

by

Ruiwen Li
Songjie Cai
Tor Saxberg

Submitted in partial fulfillment of the requirements
for the degree of
Bachelor of Computer Science and Engineering
School of Engineering
Santa Clara University

Santa Clara, California
December 3, 2017

Facial Motion Recognition System

Ruiwen Li
Songjie Cai
Tor Saxberg

Department of Computer Engineering
Santa Clara University
December 3, 2017

ABSTRACT

Facial recognition has proven to be a very useful and versatile, from Facebook photo tagging and snaptchat filters to modeling fluid dynamics and designing for augmented reality. However, facial recognition has only been used user login servicies when expensive and restricted hardware technologies, such as in smartphone devices like the iphone x. This project aims to apply machine learning teqchniques to reliably distinguish user accounts from common cameras to make facal recognition logins more accesible to website and software developers. To show the feasability of this idea, we will create a web API that recognizes a users face to log them in to their account, and we will create a simple website to test the reliability of our system. In this paper, we discuss our database-centeric architecture model, use cases and activity diagrams, technologies we plan to use for the website, api, and machine learning algorithms, and provide some samples of what the system will look like.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Face recognition technology has been used for a variety of applications from automatic Facebook photo tagging and Snapchat lenses to phone security and surveillance. Facial recognition systems rely on unique facial features as an additional layer of security to identify and distinguish people whether theyre new faces or old ones in a database.

Although facial recognition technology offers another layer of security for applications, its accuracy is highly dependent on lighting and uniqueness, and its vulnerable to simple hacking techniques like impersonating users with pictures. For instance, the face recognition system of Samsung Galaxy S8 is very convenient for easy unlocking, but it can be easily fooled by a photo of the user. Apple has provided a solution to solve this photo trick by relying on dual cameras and an array of projected infrared dots to detect depth in its new facial recognition system. However, such a solution is limited to devices with expensive hardware upgrades and cant be applied to lower cost applications. Higher costs often limit other improvements, complicate manufacturing, and raise the price for consumers.

1.2 Solution

Our solution will focus on protecting facial recognition solutions from impersonation attempts. Our system will detect a randomized gesture with depth perception to prevent images from being used to impersonate a user, with a strong enough training set, the system will also prevent prepared videos from circumventing its security. Our solution is pure software-based, requiring no additional hardware expenses, and can be applied to a wide range of applications including building security, unlocking cellphones, and website logins.

We will test out solution on the latter, creating a website that implements our facial recognition system to login a user to the site. We will test it against impersonation attempts, and ensure it correctly identifies the real user.

Chapter 2

Requirements

2.1 Functional Requirements

- Critical:
  - Account creation and login system
  - Certification with camera
  - Distinguish between real person and images
  - Recognize faces and gestures

2.2 Non-functional Requirements

- Critical:
  - Secure API
- Recommended:
  - Fast authentication
- Suggested:
  - Continuous improvement

2.3 Design Constraints

- Uses computer camera
- Uses video
- Web based platform
- Camera frame speed

Chapter 3

Use Cases

The use case represents the list of actions and event steps which define the interactions among users, websites, and the API.

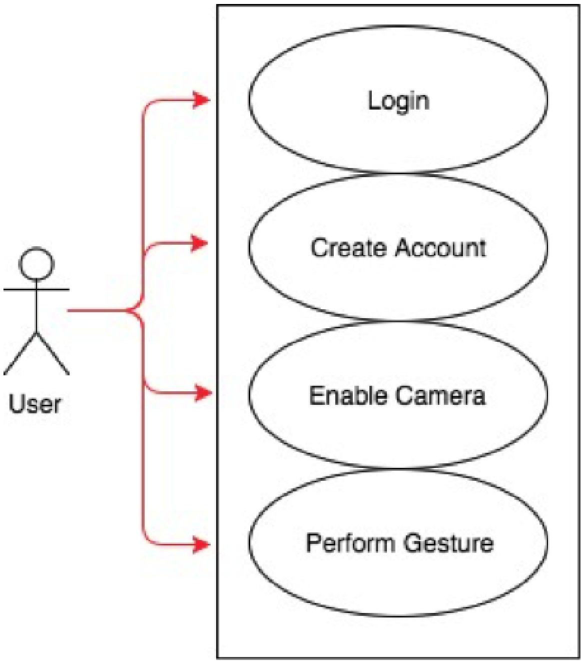


Figure 3.1: Use case diagram of the user

3.1 User

3.1.1 Use Case 1: Login

- Name: Login
- Goal: Provide user access to the system
- Actor: User
- Pre-conditions:
  - The user has an active connection to website

  - The user has previously signed up
  - The user knows his or her username

- Steps:

  - The user types in the username
  - The user clicks ”verify”
  - The system recognizes the user’s face and verifies that he or she is a real person

- Post-conditions:

  - The user’s username has been verified in the system

- Exception: The user enters invalid username

3.1.2 Use Case 2: Create account

- Name: Create account

- Goal: Provide user access to the system

- Actor: User

- Pre-conditions:

  - The user has an active connection to website

- Steps:

  - The user types in the username
  - The system takes some photos of the user
  - The system stores the pictures and username in the database

- Post-conditions:

  - The user’s account is created

- Exception: N/A

3.1.3 Use Case 3: Enable camera

- Name: Enable camera

- Goal: Provide the camera in user’s computer access to record user’s face

- Actor: User

- Pre-conditions:

  - User’s username has been verified

- Steps:

  - The system pops out a request to enable camera
  - The user clicks ”yes”

- Post-conditions:

  - The camera is enabled

- Exception: The computer does not have a camera

3.1.4 Use Case 4: Perform gesture

- Name: Perform gesture
- Goal: Verify whether the user is a real person or a photo
- Actor: User
- Pre-conditions:
  - The user enters valid username
  - The user's camera is enabled
  - The user's face has been detected by the system
- Steps:
  - The system displays a certain gesture for the user to perform
  - The user perform the gesture
  - The system analyze the user's gesture
- Post-conditions:
  - User successfully login
- Exception: The user performs wrong gesture

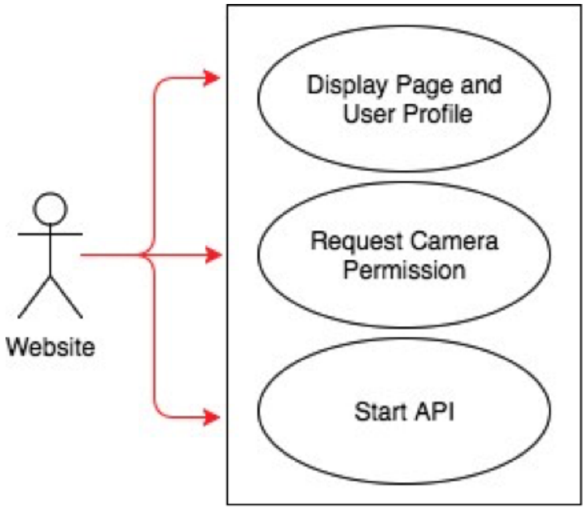


Figure 3.2: Use case diagram of the website

3.2 Website

3.2.1 Use Case 1: Display page

- Name: Display page
- Goal: Provide user access to the page in browser
- Actor: Website

- Pre-conditions:
  - The user has an active connection to website
  - The browser is compatible to the user's computer
- Steps:
  - Read HTML, CSS, and JavaScript files
  - Execute python files to make operations with server
- Post-conditions:
  - The page is successfully displayed
- Exception: 404 Not Found

3.2.2 Use Case 2: Start API

- Name: Start API
- Goal: Retrieve information using API technologies
- Actor: Website
- Pre-conditions:
  - The page is successfully display
  - User is interacting with the system
- Steps:
  - The system send a request to other websites using API to request information
  - The system receives responses from other websites
- Post-conditions:
  - The result is successfully return by the API
- Exception: The API does not work

3.2.3 Use Case 3: Display user profile

- Name: Display user profile
- Goal: Provide user permission to see his or her information
- Actor: Website
- Pre-conditions:
  - The user is successfully logged in
  - The user's browser is compatible with his computer
- Steps:
  - The system user's information from the database
  - The system lists the information on the web page
- Post-conditions:
  - User's information is successfully displayed
- Exception: Database connection error

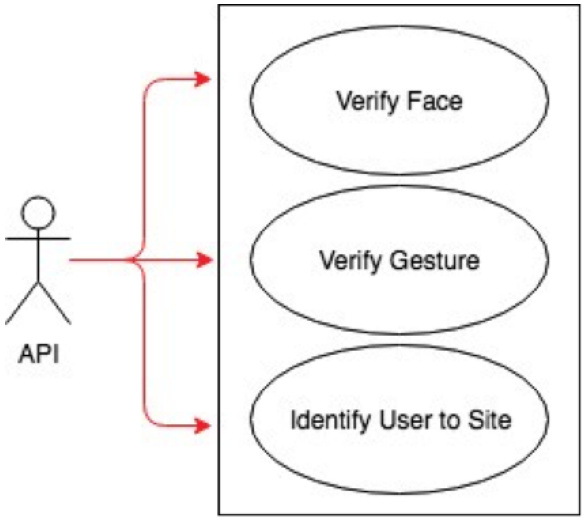


Figure 3.3: Use case diagram of the API

3.3 API

3.3.1 Use Case 1: Verify face

- Name: Verify face
- Goal: Recognize user's face in the camera
- Actor: API
- Pre-conditions:
  - The user's camera is enabled
- Steps:
  - Separate the camera video into multiple frames
  - Use pre-trained machine learning model to check if there is a face in the frame
  - Draw bounding box around the face detected
- Post-conditions:
  - User's face is successfully detected
- Exception: User's face is not found

3.3.2 Use Case 2: Verify gesture

- Name: Verify gesture
- Goal: Test if the user is real
- Actor: API
- Pre-conditions:
  - User's face is detected

- Steps:
  - The system displays a specific gesture on the screen
  - The user perform the gesture accordingly
  - The system verify is the gesture is correct
- Post-conditions:
  - The gesture is verified and user is logged in
- Exception: User performs wrong gesture

3.3.3 Use Case 3: Identify user to site

- Name: Identify user to site
- Goal: Map user's face with those in the database
- Actor: API
- Pre-conditions:
  - The user's face is detected
- Steps:
  - Search the database
  - Pick the user in the database with the most similar picture as the login user
  - return the user's information
- Post-conditions:
  - User is found in the database
- Exception: Database connection error

3.4 Swim Line Diagram

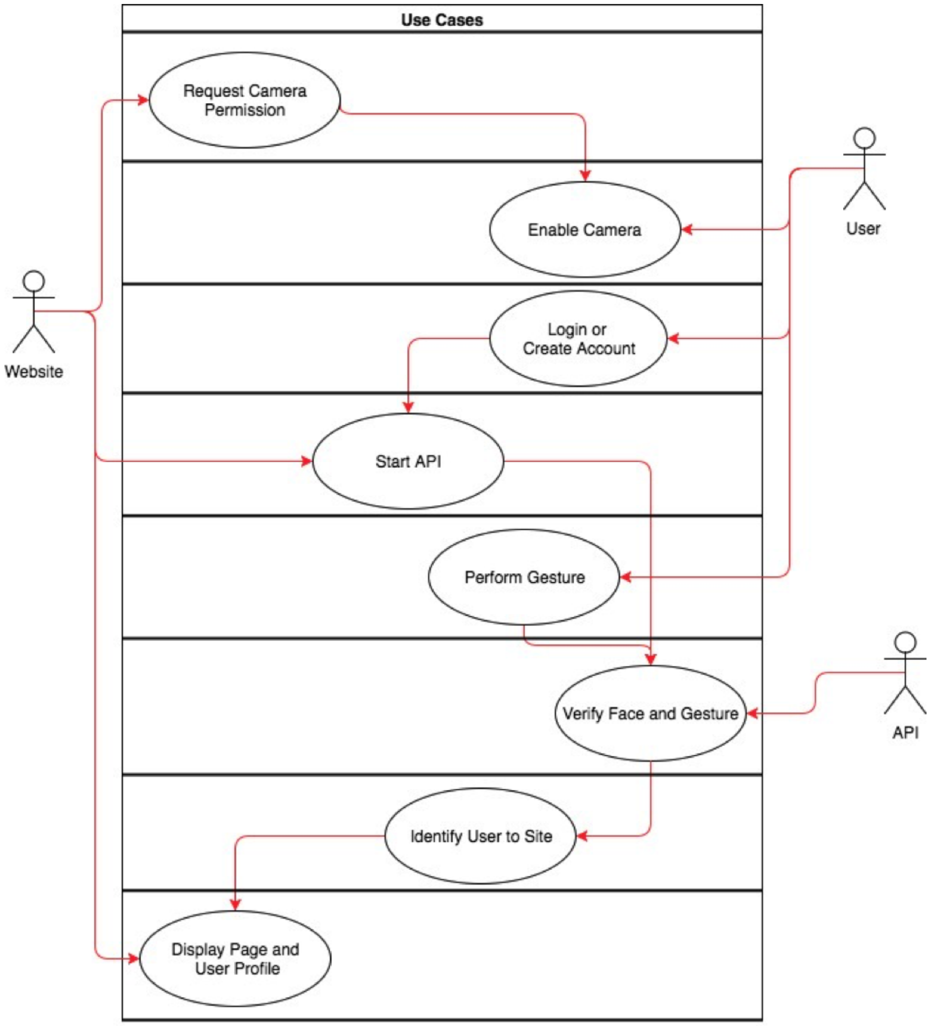


Figure 3.4: Use case diagram of the whole system in a timely basis

Chapter 4

Activity Diagrams

4.1 User

As a user, after he/she enters the index page of the website, he/she may attempt to log into by enabling the camera or create an account. The user needs to wait for the system response after the system finishes recording. After that, the user should perform given gestures from the server

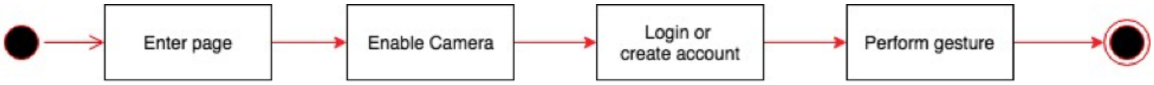


Figure 4.1: High level view of user operations

4.2 Website

The website will always wait for camera permission to start the next step. Depending on different user operations, the website will call different APIs. After the user successfully logs in, the username will be display on the page

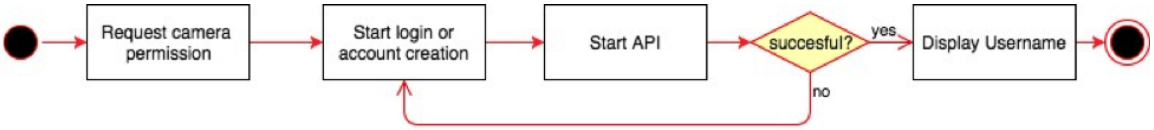


Figure 4.2: High level view of website operations

4.3 API

If the server sends a "login" request, the API will be called to camera recording, which will be sliced into pictures later. If the face is recognized, the API analyzes the gesture and return the username if both two tests are passed

For account creation, the camera data will be saved if the face is detected. After that, the gesture will be analyzed and the user's data will be saved in database if the gesture is correct

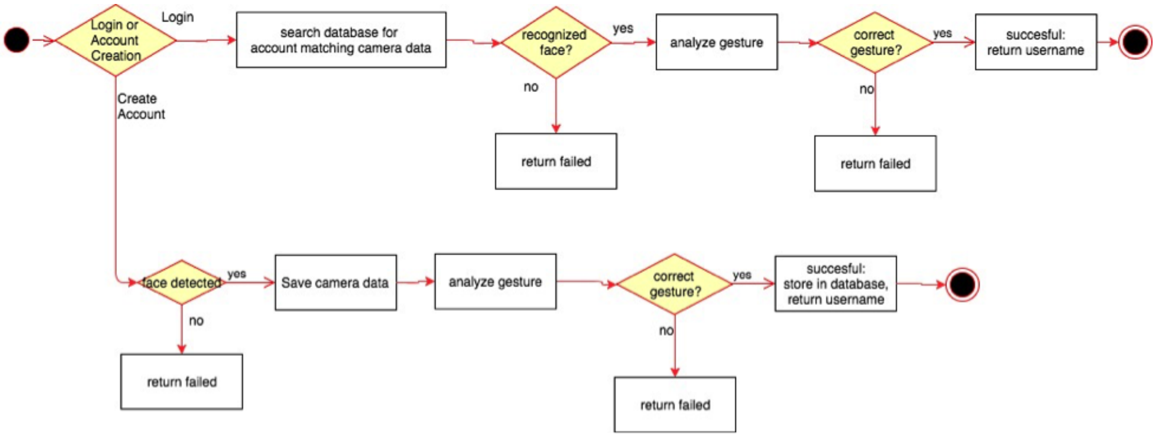


Figure 4.3: High level view of API operations

Chapter 5

Conceptual Model

5.1 Web template

The web template (See figure 5.1) shows the main components of the login and account creation page. When the user first opens the page, a window will pop up to ask permission of the camera access. There are two buttons on the page: login and create account. If the user clicks ”Login”, a camera will open to detect user’s face and recognize his or her identity, then leads to the user profile page. On the other hand, if the user clicks ”Create account”, the camera will take a few photos of the user and store them in the database, and create an account for the user.

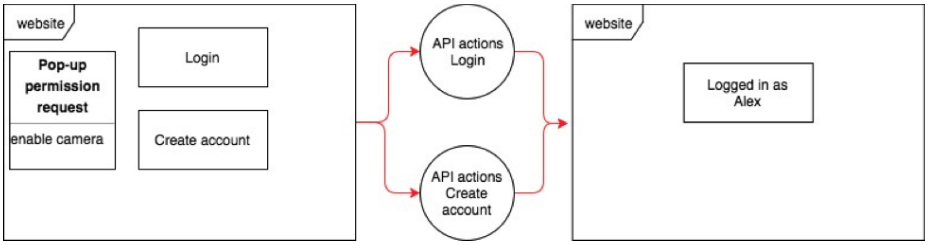


Figure 5.1: Web template conceptual model

5.2 Account Creation

The account creation page (See figure 5.2) explains the process of creating an account in detail. After user chooses to create an account, the system will require the user to enter his or her name, and open the camera to take three photos of the user. The system will then verify if the user in front of the camera is a real person by assign a gesture for him or her to perform. The system will then create an account and login for the user.

5.3 Login

The login page (See figure 5.3) is similar to the account creation page. The system will first detect the user’s face, and map it to the most similar user in the database. The system will then let the user confirm that the identification is correct. After verification, the system will then determine if the user is real by assigning a gesture for him to perform. When everything is confirmed, the user can login to the system.

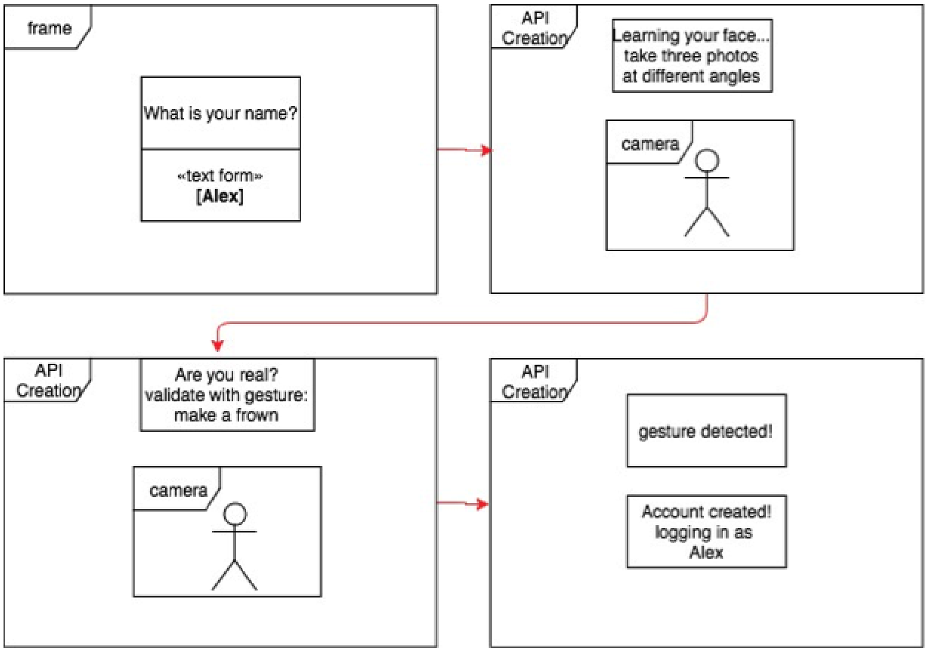


Figure 5.2: Account creation conceptual model

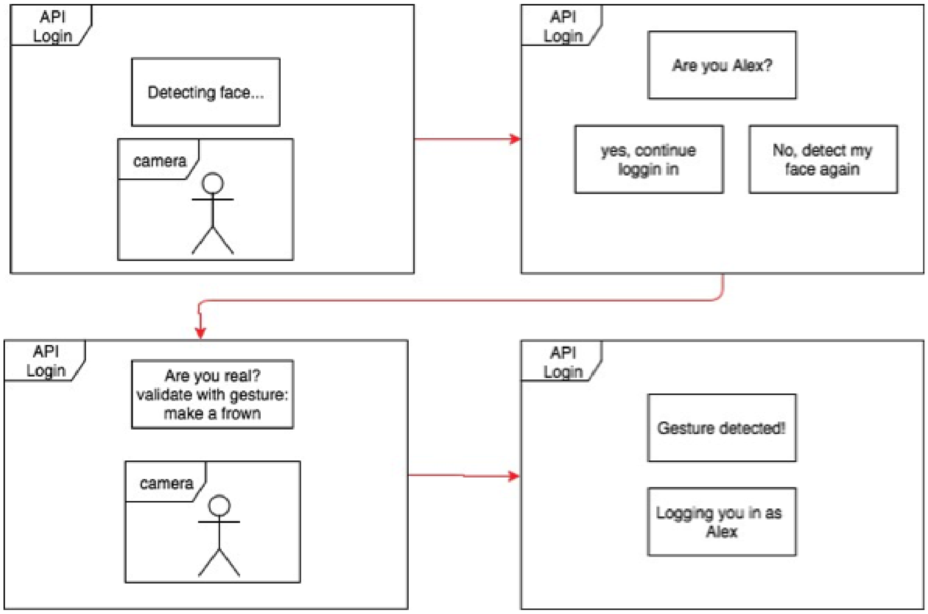


Figure 5.3: Login conceptual model

Chapter 6

Architecture

Since all data will be stored in our database, We decide to build an advanced data-centric architecture for our design to make the data more accessible and manageable for users. In addition, considering security and management of APIs, we choose to have a three-tier architecture because it helps protect data security and improve manageability of different APIs. In our design, users post requests to the web server and the server will call different APIs depending on the requests from users.

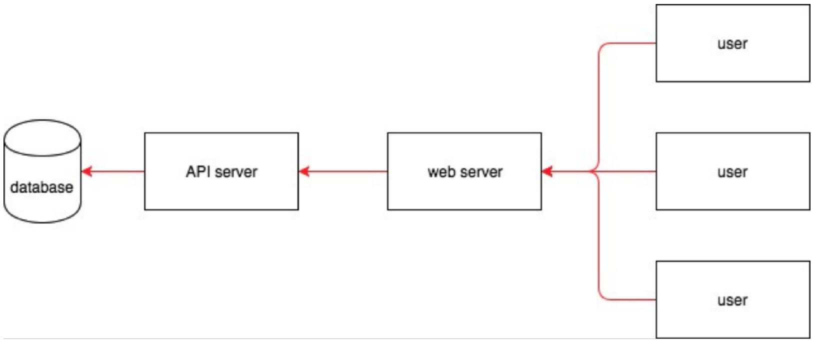


Figure 6.1: Architecture diagram of the system

Chapter 7

Technology Used

7.1 Programming Languages

- HTML5

HTML5 is used to create documents on the web page. It defines the structure and layout of a Web document by using a variety of tags and attributes.

- CSS3

CSS3 is used to describe the presentation of Web pages. It also makes the web page responsive to different devices.

- JavaScript

JavaScript is used as a client side scripting language. Its code is written into an HTML page. When a user requests an HTML page, the script is sent to the browser.

- Python

Python is used for back end operations such as training machine learning models and retrieve information using API provided by other websites.

- SQL

SQL is used to communicate with a database. As the standard language for relational database management systems, SQL statements are used to perform tasks such as update data on a database, or retrieve data from a database.

7.2 Python libraries

- Scikit Learn

Scikit Learn is a simple and efficient tool for data mining and data analysis.

- Tensorflow

Tensorflow is an open-source software library for Machine Intelligence. It can be used to implement different machine learning algorithms.

7.3 Applications

- SQLlite, API technologies

SQLlite is a self-contained, high-reliability, embedded, full-featured, public-domain, SQL database engine. It is used to store both account information and pictures of the users.

- API technologies

API technologies provide access of data stored in the database to the public.

- Web frameworks

Web frameworks support development of the web API.

- Github

Github is used for version control of the project. It allows programmers to update, revise, or restore a history version of the project. It also facilitates team collaboration on a project.

Chapter 8

Test Plan

8.1 Alpha

8.1.1 White Box

For the white box testing, we will test each portion of the code to ensure that they function as intended. We will design a series of varied test cases to use on the system to verify that all of the constraints are met in any situation. The test cases will assert the system's ability to maintain its functions in standard use situations, as well as potential edge cases.

8.1.2 Black Box

Black box testing will enlist the help of individuals not involved with the development of our application, which will likely include classmates and friends. They will understand what the system is designed to do, but not how it is done. The testers will then use the system to sign up and login, as if they are real users of the system.

8.2 Beta

Finally, our beta test will simulate real life scenario, apply the login system to a larger application. While the program should be mostly tested and confirmed to work as intended by this point, we will continue to monitor its performance as there is always the possibility of new issues appearing during live use over an extended period of time.

Chapter 9

Risk Analysis

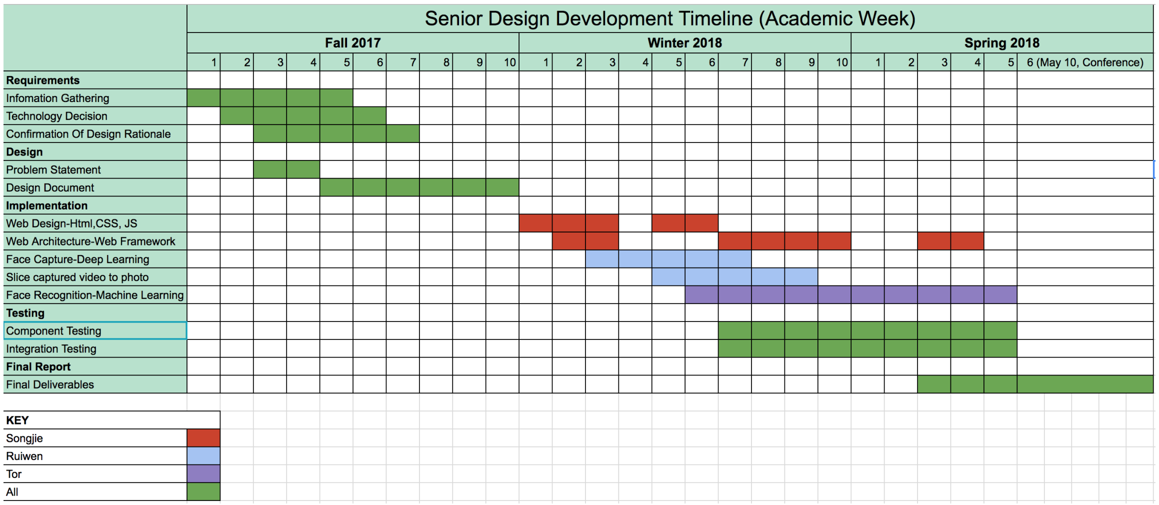
Table 9.1: Risk table

Risk	Consequences	Probability	Severity	Impact	Mitigation
Task scheduling issues for group members	Postpone the development period	0.8	0.9	4.8	Schedule weekly group meetings to check the progress tasks
Long response turnaround time	Long wait time for the user	0.7	0.6	3.6	Implement the system such that it is able to transfer faster framework
Techniques prove difficult or don't work	Can't implement design with preferred tools	0.4	0.8	3.2	Discover backup technologies, and start implementation early
Chosen Machine Learning Techniques don't work	Face detection fails to learn new faces or recognize old ones	0.5	0.7	3.5	Test algorithms early on, and consider a wide array of alterntives

Chapter 10

Development Timeline

The Development Timeline shows the project timeline in three quarter, each quarter in ten weeks. It includes tasks and estimated duration by weeks. The overall prograss of the project is represented in the timeline. We finished different tasks each week according to the timeline. It makes our project organized and well-panned. In the timeline, each color represents a team member's contribution, purple is Tor Saxberg's contribution, blue is Ruiwen Li's contribution, and red is Songjie Cai's contribution.



	Senior Design Development Timeline (Academic Week)																									
	Fall 2017										Winter 2018										Spring 2018					
	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6 (May 10, Conference)
Requirements																										
Infomation Gathering	🟩	🟩	🟩	🟩	🟩																					
Technology Decision		🟩	🟩	🟩	🟩	🟩																				
Confirmation Of Design Rationale			🟩	🟩	🟩	🟩	🟩																			
Design																										
Problem Statement			🟩	🟩																						
Design Document					🟩	🟩	🟩	🟩	🟩	🟩																
Implementation																										
Web Design-Html,CSS, JS											🟥	🟥	🟥		🟥	🟥										
Web Architecture-Web Framework												🟥	🟥				🟥	🟥	🟥	🟥			🟥	🟥		
Face Capture-Deep Learning													🟦	🟦	🟦	🟦	🟦									
Slice captured video to photo															🟦	🟦	🟦	🟦	🟦							
Face Recognition-Machine Learning																🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	
Testing																										
Component Testing																	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	
Integration Testing																	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	
Final Report																										
Final Deliverables																							🟩	🟩	🟩	🟩

KEY	
Songjie	🟥
Ruiwen	🟦
Tor	🟪
All	🟩

Figure 10.1: Development Timeline