GONZAGA UNIVERSITY

School of Engineering and Applied Science
Center for Engineering Design and Entrepreneurship

PROJECT PLAN
October 23 2017

Credential Security API with Facial Recognition

Prepared by:

Anton Sebastian Vargas

Brian Mackessy

Elijah Michaelson

Reviewed by:

Professor Nadra Guizani
Faculty Project Advisor

Chris Sharman
Design Advisory Board Member

1 Project Overview

1.1 Project Summary

Over the past decade, many major companies of have been hit with massive hacks that have resulted in hundreds of millions of people's private information to be stolen. The old method of username and password are far too susceptible to hacks, and just simply aren't good enough to keep people's private data secure. Vulnerabilities, such as the reuse of passwords or the use of weak passwords, are regularly exploited by malicious agents and it is hard to enforce policies on users that protect them. Another problem with passwords is that if you are smart and have many different passwords, it can be almost impossible to keep track of all of them. Because of these problems with usernames and passwords we think that the only way forward is biometric security. What we are proposing to use your face as your password. Advances in machine learning have made it possible for computers to recognize faces with a high degree of accuracy, opening the door for a secure and easy to use credential policy system.

We plan to build an API for software developers to integrate into their login pages to allow their users to save their usernames and passwords in our service, and automatically access these usernames and passwords when they pass a facial recognition test. Our software will be easy to use and free, with the goal of attracting as many everyday users as possible. We also want to make sure that our software can work on any computer that has a camera on it, something that existing software doesn't do.

1.2 Project Objectives

The primary business objectives of this project are

- To improve web browser security by at least 30% which is measured by reducing the number of unwanted people using websites logged on as someone else by 30%.
- Have at least one small business use our product by the end of the school year
- Have at least 100 downloads of our security extension by the end of the school year

1.3 Project Stakeholders

Developers:

We as developers are in charge of the development and implementation of this project as well as communicating our progress to our sponsor/DAB member Chris Sharman.

Sponsor/DAB member:

Our sponsor/DAB member Chris Sharman will help keep the developers in check by making sure we are making steady progress and will fit the DAB requirements as well as inform us if our product will be something that a business will find useful.

Faculty Advisor:

Our faculty advisor Professor Nadra Guizani, like our DAB member, will help keep us on track but a bit more on the side of documentation as well as implementation. She will also keep us on a good pace to finishing and completing our project.

Target User Communities:

Our target user communities are businesses and cautious web users. Businesses often face issues with computer security and it is common protocol for employees to have to lock their computers or close their web browsers upon leaving their computers. Using our product would implement that extra level of security that prevents unwanted users from using an exposed web browser. Our product would also be easy to install on commercial machines for users that highly value their privacy.

1.4 Project Deliverables

Our primary deliverable for this project is an API that a developer could plug into their website to add facial recognition access to their website. The API needs to be simple enough for a programmer to use. The interface with the actual users needs to be simple enough from someone who doesn't know how to program to use. We want the users to be able to manage passwords though the already existing Google Credential software. Another deliverable is that the API needs to have minimal false positives in order to protect the user's information. The program also needs to automatically input the credentials when the facial recognition has been passed.

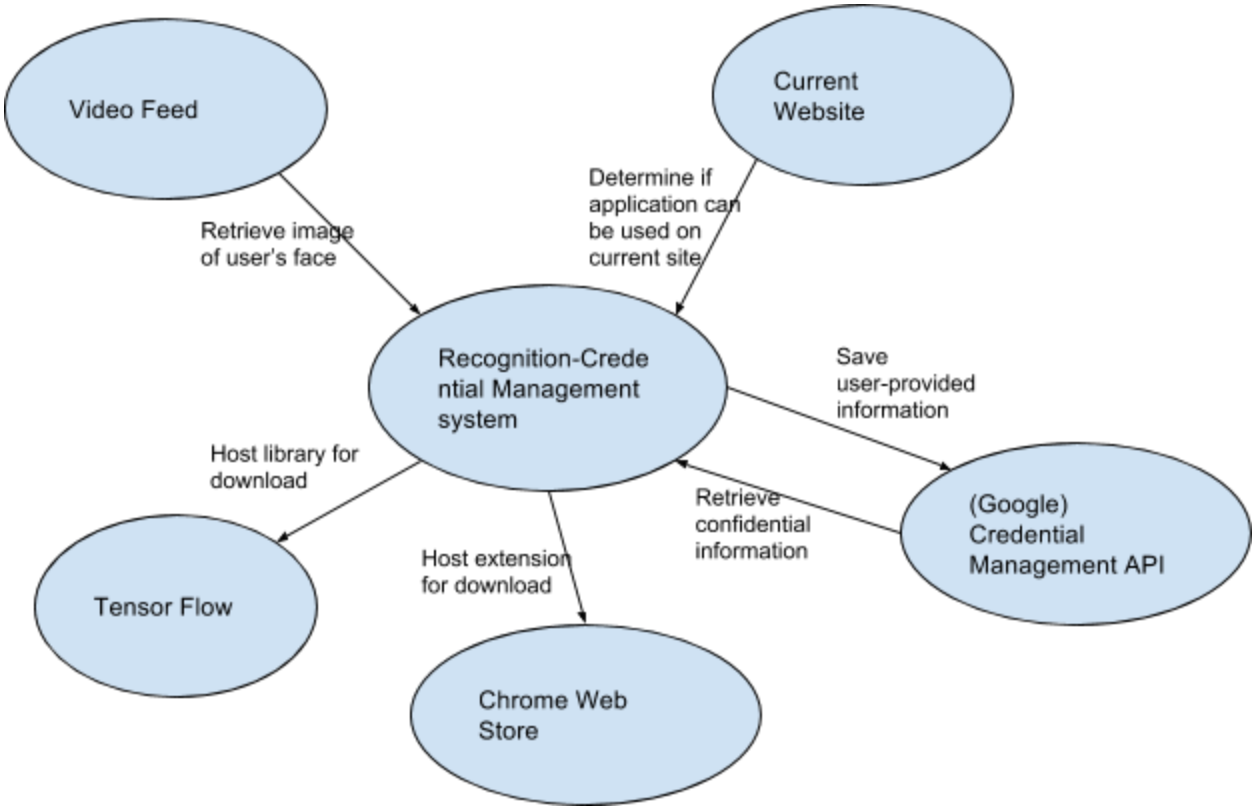
1.5 Project Scope

One out of scope component of our project is Google's Credentials Management API. We will be using (but not implementing) this API to securely store our front-end user's usernames and passwords. This will greatly simplify our task because we won't have to implement complicated security measures to keep our user's confidential information safe. Another is the video feed and the current website, both of which will provide information as to what state our system should be in. The former will inform our application whether or not to autofill the credentials based on the user's face, while the latter will determine whether the website the user is on has credential fields that our application can fill out. Additionally, we will need some external services to provide our product to the public. Since we are developing a software library as well as an extension that uses that software library, we will need to use two different services. We will place our API on GitHub.

The main component that is in-scope to our project is the neural network. We will need to develop a specific architecture that will suite our needs, namely a small training set size and the need for accurate classification. We will be able to use some external API functionality to give us some direction on how our recognition system should be developed, but the majority of this component of our project will be developed in-house. Additionally, we will need an in-scope component that can identify password fields on the current website and fill in the credential information appropriately. This component should be relatively easy to implement. Also, we will need to develop a user interface for the extension. It is our aim

to keep this interface relatively simple, with the ability to add/remove passwords as its chief functional role. Figure 1.5 shows the scope of our project.

Figure 1.5



1.6 Related Work

In the field of facial recognition there is already an unbelievable amount of work. There is software that is designed to recognize distinct faces and software that can recognize emotions. We are going to focus on recognizing faces. The fact that there is thousands of projects already on facial recognition gives us a great base to build off of. In the field of facial recognition to replace passwords there is a lot of work already done as well. Most notably the new iPhone will have facial recognition software on it, to allow people to log into their phones. Other companies such as TruKey have created downloadable software to allow you log into websites using your face or fingerprint. How we plan to differentiate ourselves is to create a Google Chrome extension to do the work, which hasn't been done. The reason we believe this is the best way to do this is because of its simplicity. Everything can be done

within your browser, which is ideal for the average person who has no desire to mess around with software.

Additionally, much of the recognition software appears to simply use static facial features to determine the identity of the user. We are hoping to make our system more robust and secure by incorporating time-series analysis into our network. Some potential time-series data we have considered training our network on include: emotions, reactions to images, skin reflectivity with light differences. Incorporating this more sophisticated analysis into our software will not only make our system more secure, but also differentiate us from competitors. On the front end side, all of the related work looks very elegant and pleasing. This can be attributed to such companies having teams of designers and artists guiding the development of their user interface. As such, we will be taking inspiration from the related work we identified (such as the Iphone X, TruKey, and FaceVault among others) to guide our user interface development.

2 Project Requirements

2.1 Major Features

Table 1: Major Features of API

Feature	*Description*
An API that developers can use	An API that web developers can implement in their websites
Access webcam	Request and obtain access to webcam to get in video/image feed.
Ask user for name and password	Fields for username and password input
Populate username and password fields	Given a known user, fill in the element with the password and username.
Identify user with high accuracy	We would like to have greater than 95% accuracy in recognizing an individual. Because of the confidential nature of the application, false positives are worse than being unable to identify a known user.
Allows users to customize passwords	The user shall be able to add/remove/modify their list of passwords. This can give the user more control over which websites have facial recognition functionality.
Visual indications of software in operation/success	The application gives some visual cue to the user that it is attempting to autofill/recognize, and then indicate whether this attempt is successful or not.
Storing webcam images for neural network usage	Be able to store images for training the neural network. Keep the files locally.

Additional help for user to capture their image (pop up a box that displays the camera)	The recognition task can sometimes be faulty, especially if the input image is noisy. Therefore, there will be a visual tool to help obtain clear images from the user.
Add additional users per site	Be able to have more than one user for gaining access to a site.
Efficient and Easy to use	The product must not be too slow or obstruct the user's website experience

2.2 Initial Project Backlog

Table 2: Initial Product Backlog

Requirement	*Description*	*Major Feature*	*Priority*	*Estimate*
API that developers can embed in their website	The product must be usable by HTML5/JSS web developers	An API that developers can use	high	10 hours
User can access webcam through the API	users must be able to access the webcam	Access webcam	high	6 hours
User can take pictures of themselves to increase security/train the net	The webcam must store images taken to train the neural network	Storing webcam images for neural network usage	middle	4 hours
User can open a website, and have the API recognize them	The user can have the API secure their websites	Populate username and password fields	high	15 hours
The program can't be tricked by a picture of a person.	The security model should be able to detect whether what is in front of them is a picture or an actual person	Identify users with high accuracy	high	20 hours
The recognition process must be faster than simply typing in the password	The process must be time efficient.	Efficient and Easy to use	middle	20 hours

Must be able to recognize faces despite day to day changes (haircuts, glasses)	The neural network must be able to recognize based solely on facial features and not be changed by temporary aesthetics like haircuts or glasses	Identify user with high accuracy	middle	20 hours
Less than 5% rate for false positives	Must never give access to unauthorized individuals	Identify user with high accuracy	high	20 hours
API autofills their password	The API must autofill	Populate username and password fields	high	8 hours
There needs to be a place where the user can manage their passwords/usernames	A management interface for the users to decide what passwords they want	Ask user for name and password	low	8 hours
Minimally intrusive for the users web browsing experience	Must not disturb the user	Efficient and Easy to use	low	5 hours

2.3 Additional Features

Some stretch goals we have discussed is the ability to add new users to your personal facial recognition. This means that a husband and wife could both use facial recognition to access the same bank account.

Another stretch feature is using time series data of the user moving their head so that the software could detect whether it is looking at a 2-D or 3-D object. This would be an excellent feature because it would stop someone simply showing an image of someone's face to access their credentials. Another way to verify whether the image in 3-D or 2-D is to use artificial flash from the laptop screen to detect whether light bounces off the face naturally or unnaturally.

3 Design Considerations

3.1 Initial User Interface Design

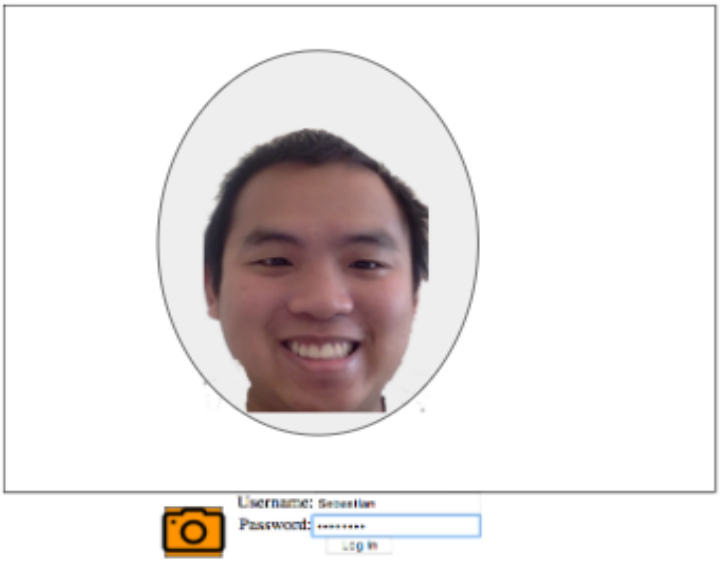
The main goal we had for the user interface for our API is that it had to be as minimally intrusive as possible. So with that in mind, we developed a UI that does just that while also providing security. The feedback we got from our advisor was implementing another feature to have the lock screen be unlockable with a password as well as not having bright colors while the extension is idle. Below is a mockup of how our proof of concept HTML page showcasing our API in action would work with the minimum amount of functionality we will provide.

First time user

Figure 3.10

There would be a camera icon next to the username and password fields. This image is a placeholder that the developer can modify. A new user would be instructed to type in their username and password and upon entering them, the user would click on the camera and the user would see Figure 3.11

Figure 3.11



The user's camera would then be opened up and they would have to position their face into the oval.
The camera icon would then change color indicating that it is processing. Again, the developer will have control over how this would be handled on their own website. Then the user would be directed with Figure 3.12

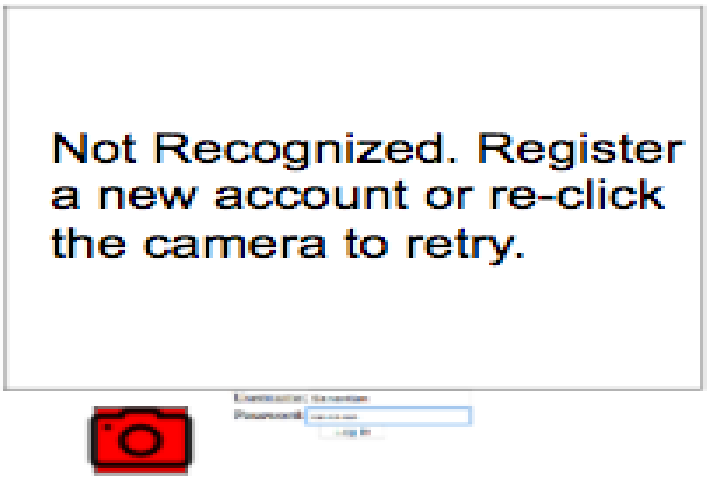
Figure 3.12

The user would then be shown a screen that the training was successful.

Returning User

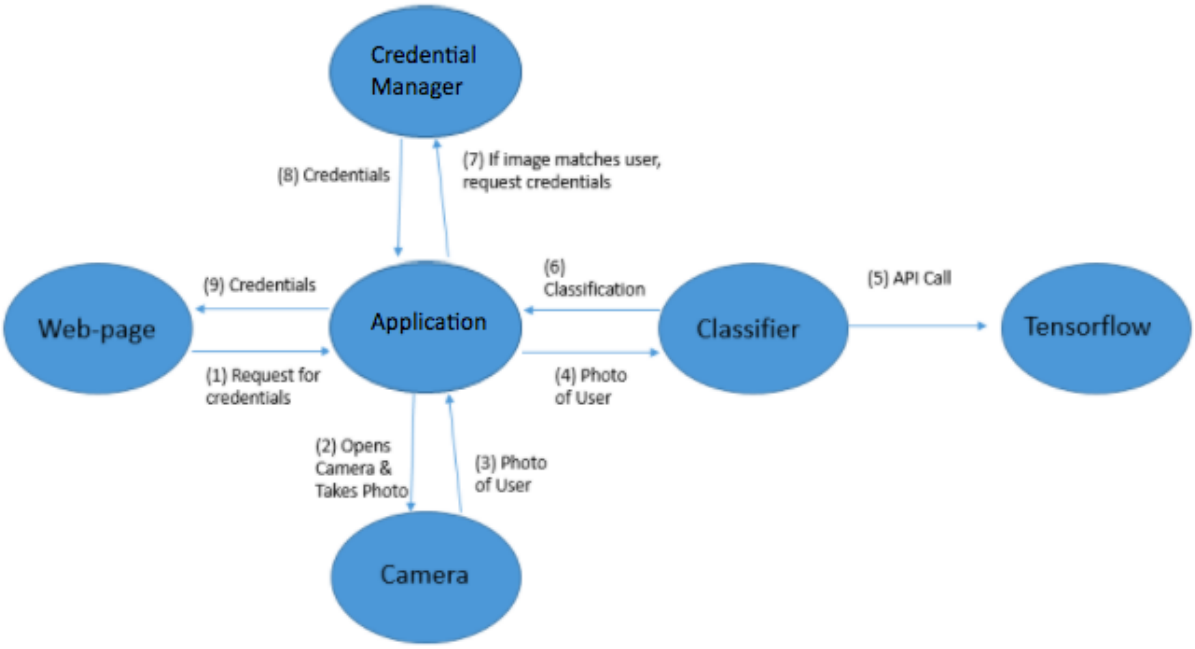
A returning user would be greeted with the screen in Figure 3.10. The next time they want to log in, they would just have to click on the camera icon. If they pass the facial recognition screening, they would be logged in. If not, they would be greeted with Figure 3.13

Figure 3.13



3.2 Initial Software Architecture

The first part of our architecture is an application that can detect when a user has visited a page, whose credentials are stored within our system. Once the request for credentials is received by the API, the API gets access to the user's camera and captures an image. This image is then sent to our classification component, which uses a Convolutional/DNC/Echo State Neural Network to reference the newly captured image to other photos of the user to determine whether the person should have access to the credentials or not. The classification component will be a python script using TensorFlow, and will notify the API with its classification. If the user is successfully verified, the API will fetch the credentials from our credential manager, which is an external component of our application. The extension will then auto-fill the username and password, and log-on the user.



4 Project Risks

One major risk associated with our project has to do with the image classifier not being effective enough. If the classifier produces too many false positives or negatives, it will not be useful to the users. Obviously error needs to be minimized, but the real question that needs to be addressed is how much error can there be for the program to still be useful to users. Users will not use our tool if the accuracy is not high enough. In order to prevent this risk, our project uses methods that have been proven to classify images at a level acceptable to most users. If we follow the path of successful image recognition project before us, is decreases the likelihood of us failing. Throughout the project, we will consistently perform accuracy tests on our classifiers to ensure that there will be no surprises late in the project. One situation where we may need to mitigate risk is if the users need an especially high level of security. If the user is

using software to secure important information, there can be no false positives, which will require the classifier to be exceptionally accurate. If this risk becomes a reality and our classification code is not working to high enough level, we can't be afraid to abandon our classifier. Ideally, we would like to write a lot of the classification code ourselves, but if we run into problems, we may have to use open source code that is proven to work for accurate classification.
Another major risk associated with this project is that the classifier is not fast enough at classification to justify use. If the software is significantly slower than just typing in the password, people won't want to use our software. Luckily there are lot of different ways to solve image recognition project that work at various speeds and accuracy. Tensorflow tutorials we have done indicate that a convolutional neural networks should be fast enough to where this won't be an issue. Despite this, we can monitor this risk by continually testing the speed at which our classifiers run. If they are not consistently faster than typing in your password, we will have to pivot our plan. We have been researching a neural network known as a liquid state machine, which is designed for speed. It works by hard coding parts of the network to reduce the amount of computation with minimal effect on accuracy.

5 Initial Product Release Plan

5.1 Major Milestones

Table 3: Major Milestones

Milestone	*Description*	*Target Completion Date*
API UI	The design of the API which includes the functions which a developer can access will be complete.	Third week of November
Neural Network back-end on an HTML page (proof of concept)	Our API will have a working neural network facial recognition system as well as a basic website with a log in functionality to use our API	Third week of January
Neural Network accuracy improvement to 90%	Improve our neural network to have a 90% accuracy rating to improve security	Second week of February
Have a deployable API	Have an API that is ready to be used by developers and we will test this by having another senior design group (Sparespace) implement our API. The API will provide the developer with camera access and autofill functionality upon an acceptable level of confidence from the classifier. The developer will have flexibility over the confidence level.	Third week of March

5.2 Initial Sprint Releases

Table 4: Sprint Release Plan

Sprint Date	*Sprint Goal*	*Backlog*	*What we will demo*
4th Week in Oct to 1st week in Nov	Have API functions set and discover if they are implementable	Access Webcam, Populate username and password fields	Have an HTML page that demonstrates functions that the API can accomplish without having the neural network part
2nd Week in Nov to 3rd Week in Nov	Test the API UI and modify it according to feedback from users.	Efficient and Easy to Use, An API that a developer can use	Demo our revised API to our faculty advisor and DAB member
4th Week in Nov to 1st Week in Dec	Develop and implement a neural network that can do facial recognition given an image	User can take pictures of themselves to increase security/train the net	Demo a python script that can train on an image set of a person and return an accuracy rating after being given another image
2nd Week in Dec to 3rd Week in Dec	Integrate our neural network scripts into the API	Identify user with high accuracy	Demo the code in our API and receive feedback on it to make sure it is easy to use for prospective users
4th Week in Dec to 1st Week in Jan	Implementing the Credentials API to our testing HTML page to use as a log in	An API that developers can use. API autofills their password	Demo basic log in features of the webpage
2nd Week in Jan to 3rd Week in Jan	Have a basic website that has a login using facial recognition to implement our API	User can take pictures of themselves to increase security/train the net. User can open a website, and have the API recognize them	Demo the site and the facial recognition (Expecting an average of 70-80% accuracy)
4th Week in Jan to 1st Week in Feb	Given feedback from our demo of the previous week, implement changes as well as further API UI refinement	Minimally Intrusive	Demo the improved API
2nd Week in Feb to 3rd Week in Feb	Improve our neural network to have a 90% accuracy rating to improve security by performing a fractional design of experiments to optimize our neural network	Identify user with high degree of accuracy	Present the results of our optimization
4th Week in Feb to 1st Week in Mar	Test the improved neural network with the HTML	Less than 5% rate for false positives	Present the results of our tests and demo the new neural network

	page we created earlier and record any changes in improvements		
2nd Week in Mar to 3rd Week in Mar	Have an API that is ready to be used by developers and we will test this by having another senior design group (Sparespace) implement our API	An API that developers can use	Implement our API onto Sparespace's react log in. Demo their log in page with our API.

6 Maintenance Considerations

Because our project will rely on using various open source libraries such as Tensor Flow and the Credential Manager, we will need to ensure that we have developer(s) tracking whether there are security problems with any of the API calls. This is especially important because our project is a security application, and our team needs to make sure that we are always using the most up-to-date and well supported security protocols. This will require a significant level of expertise in computer security as the developer(s) will need to know the strengths/weaknesses of emerging security protocols and will need to know when new-found vulnerabilities put our system at risk. In order to ensure that future maintenance goes smoothly, we plan to develop our system in a highly modular fashion. That way we will minimize impact on the system if we need to pull out and replace components of the code later on.

Additionally, we will have developer(s) available if there is some change in the format of websites that would cause us to be unable to identify and populate the credentials field. While this is unlikely in the short term, browsers/markup languages evolve and we need to ensure that our system is able evolve as well. Similar to the security considerations, we will need developer(s) keeping track of changes to the languages/technologies that we rely on and ensure that our system is able to adapt too.

7 Project Management Considerations

So far, we have been meeting two times a week to work on the project: a couple of hours after class on Thursday and 3-4 hours Sunday evenings. These two meetings give us a chance to discuss what we need to do for Software Engineering class and a chance to work together solving problems for our individual task. We meet with our advisor on Wednesdays at two to discuss what we have worked on this week, and what we will do for the following week. We have been keeping our DAB member updated via email.

We have been breaking up the work into two distinct parts. The first is the user side of the project including, but not limited to: accessing camera, designing GUI, and everything involved with detecting and inputting credentials. So far, we have been working on this as a team, but moving forward Sebastian is taking lead on this part of the project. The second part of the project is implementing the classifier. This part has and will be mostly worked on by Brian and Elijah.

We have been using Trello, Google Docs, and Github to keep our DAB member, advisor, and ourselves up to date on the happenings of the project.