GateKeeper

by

Isaac Depue, Aaron Klaus, Hunter Coffland,
Brent Roberts, Thomas Lyons

Submitted to
the Faculty of the School of Information Technology
in Partial Fulfillment of the Requirements for
the Degree of Bachelor of Science
in Information Technology/Cybersecurity

Isaac Depue	4/24/2022
Issac Depue	Date
Aaron Klaus	4/24/2022
Aaron Klaus	Date
Hunter Coffland	4/24/2022
Hunter Coffland	Date
Brent Roberts	4/24/2022
Brent Roberts	Date
Thomas Lyons	4/24/2022
Thomas Lyons	Date
Tyler Hopperton	4/24/2022
Tyler Hopperton	Date

University of Cincinnati
College of
Education, Criminal Justice, and Human Services

April 2022

Table of Contents

Table of Figures

Abstract:

Everyday all around the world physical access control mechanisms are used to control who can go where and when. These systems often utilize a badge system that is flawed and vulnerable to abuse, or they incorporate biometric hardware that costs companies' tens of thousands of dollars. With GateKeeper we can lock down physical access points and require the user to authenticate using various methods with their local mobile device. Rather than having unfettered access to buildings, a smooth and fast authentication middleman can be added in to improve security, without greatly reducing costs.

Introduction:

Project Summary: GateKeeper will provide a cost effective three factor authentication that includes badge scanning, facial recognition, and location verification through a mobile application that uses onboard facial recognition software, an onsite badge reader, and GPS location through onboard systems of the user's cell phone. GateKeeper will be able minimize the personnel needed to secure a building through offloading responsibilities such as verifying badges to the application.

Problem Statement: Businesses and organizations are under constant threat of security breaches based off poor security relating to current security card IDs. Theft of credentials through attacks such as Man-In-The-Middle that can steal identity credentials combined with lost/misplaced security card IDs pose a dynamic threat to a company's security. "So it is possible to say that if a user loses his or her smart card, all information in the smart card may be revealed to the attacker" (Choi, pg. 1222). Additionally, companies will spend tens of thousands of dollars on various protocols and biometric infrastructure to handle their secure locations. For example, Idemia, a security company offers a biometric fingerprint scanning device that costs "$12,402.82" (neobits, 2022) and that does not include retinal scanning. Current dual factor authentication methods do not provide adequate security measures to prevent unauthorized access. GateKeeper is designed to utilize an easy to use, multi-factor authentication method to enhance security aspects of current networks with minimal expense and maximum user efficiency.

Solution: The solution in eliminating the security threats, infrastructure costs, and old-school methods is an application that everyone can conveniently use from their mobile device while maintaining the same standards of security by implementing multi-factor authentication. The self-service mobile biometric aspect removes the immense cost of on-site biometric hardware, while providing improved security with an application that is easy to use and extremely difficult for threat-actors to overcome. By adding biometric security measures that people can utilize without having to remember complex passwords, this increases effectiveness and efficiency for the end user while simultaneously securing access to the company.

Contact Information:

Team Member	Degree + Track Track N/A for BSCyber	Email	Phone Number
Aaron Klaus	BSIT- Game Design	Klausaj@mail.uc.edu	937-545-3724
Isaac	Networking/Systems	Depueic@mail.uc.edu	513-356-3905
Brent	BSIT - Cybersecurity	Robertbt@mail.uc.edu	513-439-4489
Thomas	Cybersecurity	Lyonsta@mail.uc.edu	513-965-1645
Hunter	Networking	cofflahj@mail.uc.edu	740-253-5925

Project Source:

All of us within our group have had prior experience with dealing with access management with regards to smart card/badge readers. With security becoming a more severe issue to focus on, our group was brainstorming methods and techniques that could be implemented to create a more secure process. We started considering options regarding biometrics and how they could be implemented to develop a more secure access process. Our team decided that creating a more cost-effective, secure access process made the user feel like they held the keys to the kingdom, and from that we produced the title of the project, GateKeeper.

Project Objectives/Goals:

Upon completion of the project, our goals are to develop an access management application that produces a secure method of entry utilizing biometric data along with location verification and card access. Our end goal is to have the user scan their badge at a door, then have a secure access token request sent to the camera of your cell phone, then upon approval, have the secure access token request then verified for your location and if that is approved the user is granted a secure access token and is approved for entry. Obtaining an app as such would achieve our goal of providing low-cost, enhanced security for small to large businesses whose aim is to protect their property and assets.

Team Members and Responsibilities:

As a team with some specialties, but no real expertise in any area, we have decided to all consume the same all-around role so we can constantly work together, provide feedback, understand each arm of the project, and hold each other accountable while being able to assist one another. This will also ensure that every team member is up to date on the details and status of the project, and should problems arise, then the group is aware and able to address the situation and provide feedback and/or solutions.

Software Dev, Researcher, Hardware integration, Database management

- Responsible for researching the best solution and helping provide constant feedback
- Responsible for planning out and carrying out application development
- Responsible for experimenting with hardware to develop an infrastructure to demonstrate our solution
- Responsible for following best security practices and looking for vulnerabilities
- Responsible for helping put together and managing any required database
- Responsible for testing the solution and always thinking of alternative methods

Project Scope:

Our team will develop a comprehensive application that ties in a multitude of factors to enable a simple, safe, and secure authentication method that can save costs while at the same time increase security measures. This solution will include code triggered after a badge is scanned that sends a request to our app on the associated users mobile device. This app provides a user interface where the user can quickly open the app, which will scan their face, and check their physical location, before sending a success or failure response back to the door controller. Initial hardware will include raspberry pi's, breadboards with a badge scanner, lights, and other required pieces that we will put together to act as our door and controller. After having a solution that uses this hardware to scan the badge and successfully authenticate with the developed app, we will investigate further integration of the solution into existing open-source door controller software, or a way of integrating a hardware controller with this code functionality as a physical sub-component stemming from the main control panel.

Quick Project Timeline:

Task #	Task Name	Duration	Start Date	End Date
1	Researching authentication	1 week	Sept 6	Sept 13
2	Researching open-source software for door controllers and badge scanners	1 week	Sept 13	Sept 20
3	Creating an application wireframe for our mobile app	1 week	Sep 20	Sept 27
4	Building hardware infrastructure with raspberry pi and badge scanner	2 weeks	Sep 27	Oct 11
5	Developing the mobile app's UI	2 weeks	Oct 11	Oct 25
6	Developing code and small user database for hardware solution	2 weeks	Oct 25	Nov 1
7	Developing code to perform authentication between the hardware solution and the mobile application	2 weeks	Nov 1	Nov 15
8	Integrating face ID and location data as well as back up authentication alternatives into the mobile app	2 weeks	Nov 15	Nov 29
9	Establishing communication between the hardware and the app, perhaps by calling our apps API to trigger authentication	1 week	Nov 29	Dec 6

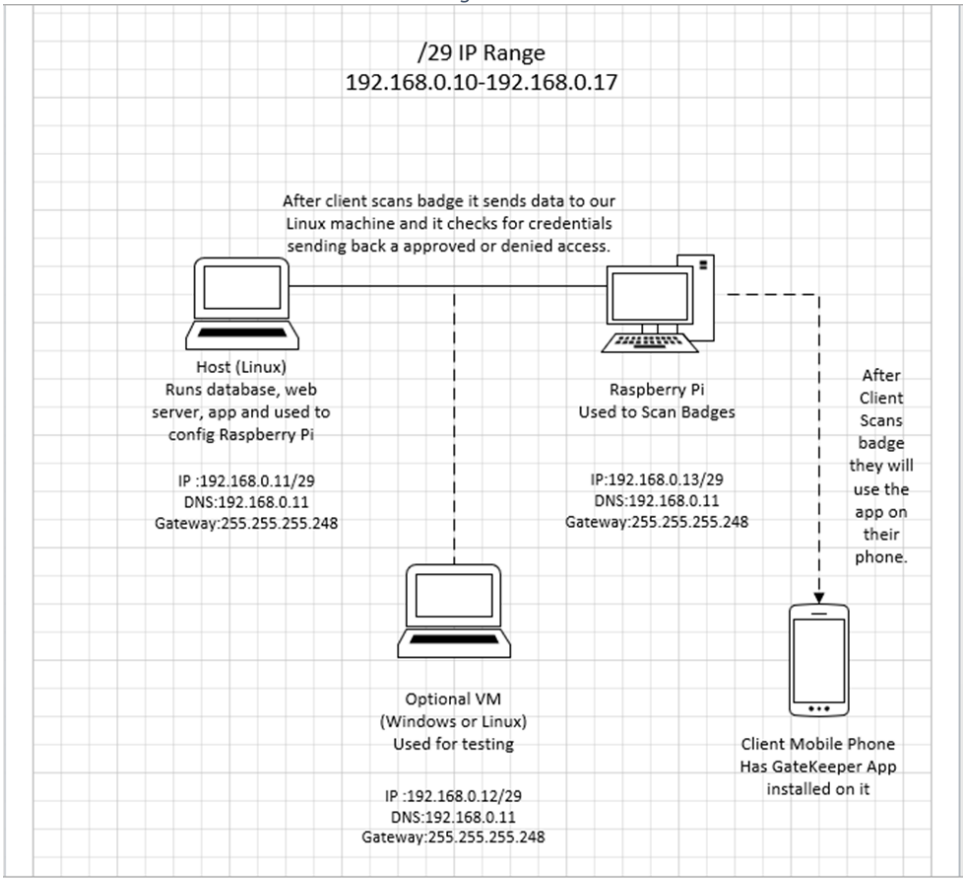
10	Testing different scenarios to confirm reliability and successfully showing use of application	1 week	Dec 6	Dec 13
11	Write proper documentation	1 week	Dec 13	Jan 3
12	Officialize application, integrate logos, etc.	3 weeks	Jan 3	Jan 24
13	Write paper on its use	3 weeks	Jan 24	
14	IF more work is required and time prevails: Try to further integrate API with existing door controllers or add our hardware component to the main control panel of existing physical access control solutions			
15	Present our solution			

Technologies Used:

Technologies we intend to use include a physical badge scanner running on a laptop or raspberry pi, A server running on a raspberry pi to use for IP based network-authentication and communication with the badge scanner, a mobile application integrating multiple services including facial recognition and location data, potentially a breadboard connected to the raspberry pi with physical components to act as our door and show success or failure of authentication, potentially a local database on our controller to manage badge associations and that users access level, etc.

Technical Architecture Diagram:

Figure 1



We host mainly everything on our Linux machine, there we are running the app, web server and database. It is hard wired up to our raspberry pie device for configuration purposes. We also used a VM machine to test connectivity and access. Basically, how they all interact is the client will have our Gatekeeper App installed on their personal device. When they scan their badge the raspberry pie device will send the credentials to our Linux machine and if they have the mobile device that is registered for the person whose badge it is, the app will prompt them for a facial scan. The Linux device will then either approve or deny the request.

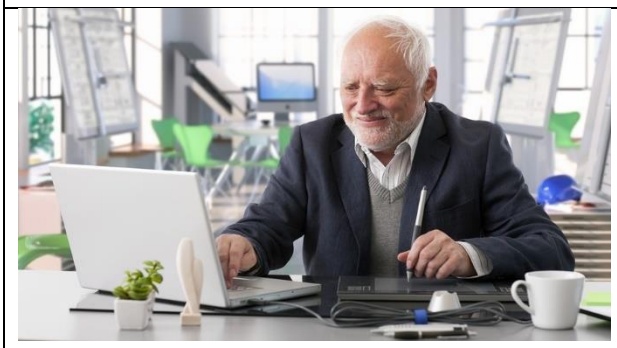
User Personas:

Below are various User Personas of people who would likely use our application. They all have different behaviors, frustrations, and needs.

Table 1

User Persona: 1	
 Src: https://exactsports.com/blog/expectations-of-a-college-athlete-an-everyday-battle/2011/11/04/	Student Athlete Mikey Wallace 20 Male
Behavior	- Often accessing buildings late at night to study and work on projects. Uses multiple buildings during hours they should be inaccessible to public - Must access locker room and sport facilities that are inaccessible from most campus-goers for practice and workouts.
Pain	- Having to get into school buildings late to work on classwork, without being restricted to certain locations or lacking security. - Having to access the locker room during necessary hours without having to contact a coach or somebody with a key
Needs & Goals	- A system that understands where he should be able to access locations and when and can provide him smooth yet secure access that does not hinder his performance with his busy schedule.

User Persona: 2	
 Src: https://www.vecteezy.com/vector-art/5273269-female-doctor-cartoon-character-on-white-background-young-or-teenager-lady-doctor-in-uniform-and-casual-wear-medical-workers-or-hospital-staff-vector	Hospital Staff Lead Sara Craft 42 Female
Behavior	- Manages new staff in the hospital - Responsible for maintaining hospital protocols and being conscious of employee access to restricted areas - Regularly moving throughout different areas of the hospital.
Pain	- Keeping track of the employees and if they have directed access to areas - Keeping security protocols in line and doing their part to keep the hospital secure - Needing to have quick access to a different area in the hospital at any time
Needs & Goals	- A system that can handle new employees, track their access, and access history for her. It should be easy to change this access level or provide a temporary pass to some locations - A secure system that will keep the hospital staff accountable yet allow authorized personnel to access areas they need to with little resistance.

User Persona: 3	
 Src: https://www.marketwatch.com/story/what-older-workers-can-and-cant-do-with-their-401k-plans-2014-10-24	Non-technical employee Tom Olenick 64 Male
Behavior	- General employee who does not keep up to date with all the latest technology - Sticks mostly to paper and pencil work when possible - Shows up to work and often forgets to bring their badge and needs to have security personnel or another employee let them in.
Pain	- Not knowing how to use complicated mobile apps - Needing a backup resource to let them in when badge is forgotten
Needs & Goals	- A fallback plan when the user forgets their badge or phone and needs to access the building - A simple app that can easily be figured out and used without needing assistance

User Persona: 4	
 Src: https://edition.cnn.com/2019/09/18/business/new-ups-uniforms/index.html	Package Delivery Person Suzy Quell 31 Female
Behavior	- Travels around and delivers packages to numerous buildings, often needing to get access from front desk workers. - Goes in and out of buildings that they are not an employee of, but make regular visits to
Pain	- When delivering during strange hours sometimes you can't get into the mail drop-off of a building if access is not provided. - Goes to the same buildings on a regular basis, but loses a lot of time requiring front desk employees to get the door for them every time - Has no universal way of accessing a building.
Needs & Goals	- A regulated way to access buildings that are consistently delivered to. - A way to maximize efficiency when taking packages to buildings that have locked entrances.

Use Cases:

Below are several Use Cases for the GateKeeper application to help present a blueprint of the functionality found within the app.

Table 2

Use Case ID	1
Use Case Name	Sign In
End Objective	Gain access to the system with users associated account
User/Actor	Existing user/admin
Trigger	From the sign-in page, the user enters the email and password in the sign-in fields
Frequency of Use	Every app opening – unless permitted to remember sign in
Preconditions	Users must have an account
Basic Flow	1. The user opens the app and is prompted for their email and password 2. The user enters their emails and password in the sign-in fields 3. The system validates the data and checks that the email and password are associated 4. If validation passes, the user is logged into the dashboard 5. If validation fails, the alternate flow is triggered
Alternate Flow	1. Invalid Data input: - System notifies user that the provided information did not match the stored data - System prompts the user to enter the required fields - System validates the data and checks that the email and password match - If validation passes, the user is logged into the dashboard - If validation fails, the flow is repeated 2. User forgot password:

	- User is prompted to enter their recovery information or security question - User enters information - Recovery email is sent to the provided email account if it exists
Postconditions	1. Login - Users log into their account and can see the dashboard. 2. Login Failure - Invalid data was entered, failed to log into account

Use Case ID	2
Use Case Name	Sign Up
End Objective	Become a registered user
User/Actor	New app user
Trigger	From the sign-in page, the user selects “sign up”, displaying the sign-up view and prompting the user to enter personal information
Frequency of Use	Once per user
Preconditions	None
Basic Flow	6. The user opens the app and selects sign-up at the initial sign-in page 7. The user enters Email, password, and other required fields for the system to establish their account settings 8. The inputted information is validated 9. If the data passes validation it is stored by the system 10. If the data fails validation the alternate flow is triggered.
Alternate Flow	3. Invalid data input: - System highlights to the user which data is invalid - System prompt’s the user to enter this field again - User enters required fields - System validates information - If data passes validation, it is stored - If data fails validation this flow is repeated
Postconditions	3. Success - Data stored, account created, confirmation is provided to the user 4. Failure - Invalid data was entered, account failed to be created

Use Case ID	3
Use Case Name	Managing account preferences
End Objective	Update user or account settings in the system
User/Actor	Existing user
Trigger	From the dashboard, user goes to the settings pane
Frequency of Use	Infrequent – varies upon user preference
Preconditions	- User has an account - User can sign into their account
Basic Flow	1. The user signs into the app reaching the dashboard 2. From the dashboard the user selects 'settings' 3. The system displays the stored user data 4. The user enters new data for any field they wish to update 5. The user tells the system to save and store the new data 6. The system validates the input 7. If validation is successful, the new values are stored, and the user is notified of success 8. If validation fails, trigger the alternate flow
Alternate Flow	4. Invalid data input: - System highlights to the user which data is invalid - System prompts the user to enter this field again - User enters required fields - System validates information - If data passes validation, it is stored - If data fails validation this flow is repeated
Postconditions	5. Success - Data stored, account updated, confirmation is provided to the user 6. Failure - Invalid data was entered, account failed to be updated

Use Case ID	4
Use Case Name	Perform Authentication
End Objective	Users' identity is verified, authentication response sent to badge server, door opens
User/Actor	Existing user
Trigger	The user put their badge up to a scanner at a location they are trying to access
Frequency of Use	Every time physical access is requested
Preconditions	- User has a GateKeeper account - User can login to their account - User has a badge registered to their account
Basic Flow	1. User presses badge against scanner 2. Scanner reads the badge and pulls the associated user data 3. If the user has access rights to this location, an authentication request is sent to their registered mobile device 4. If the user does not have access rights granted to them for this location, flow fails and alternate flow (1.) is triggered. 5. System prompts the user to open applications on their registered device 6. User opens application and signs in 7. System prompts user to begin authentication 8. User clicks the button to begin 9. System runs the function scanning their face ID 10. If face scan is a match, system checks location match 11. If face scan is not a match, trigger alternate flow (2.) 12. If the system finds that location data matches the location in the server, respond to the authentication request with success 13. If the system finds that location data does not match the location in the server, trigger alternate flow (2.)

	14. Badge server receives authentication success 15. Badge server allows user access to this location
	1. Access not granted - Badge System ends door process - System sends notification to user informing them that they don't have access to this location and what to do if they believe they should have access. 2. Authentication failed - Due to incorrect location data or facial ID, the authentication failed. - User is notified of which authentication method failed, and to begin the scan again - Fail response returned to request from badge server, ending the connection and preventing the door from being unlocked
Postconditions	7. Success - User is authenticated, door is unlocked for the user 8. Failure - The user failed the authentication or is not allowed access to this location. The door is not unlocked.

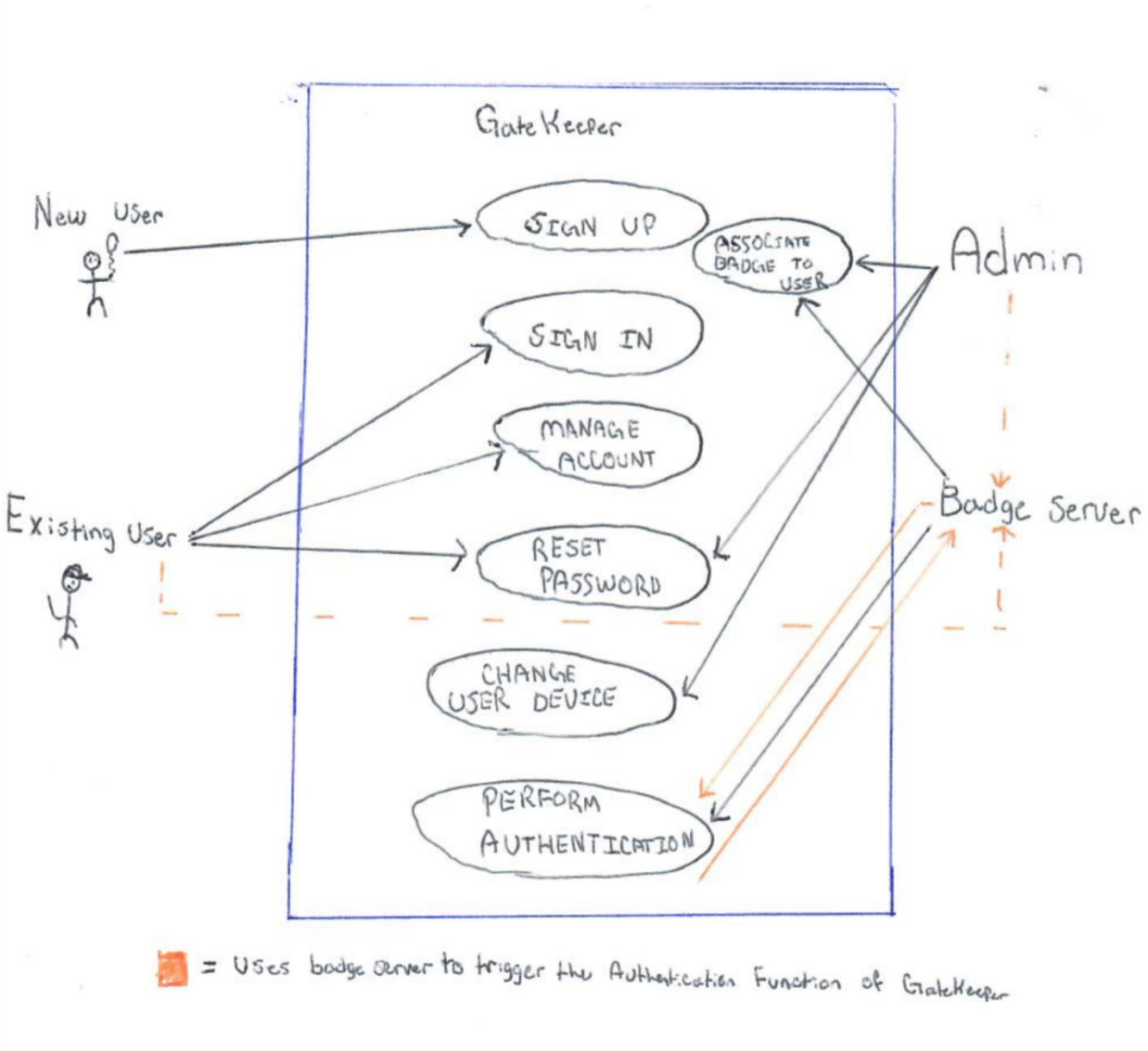
Use Case ID	5
Use Case Name	Register user to badge
End Objective	Have a badge ready for user
User/Actor	Admin
Trigger	From the admin console, begin user badge creation function
Frequency of Use	Once per user – unless a remake is required
Preconditions	- User has an account - User can log into their account - Admin can log into the admin console
Basic Flow	1. The admin opens the admin portal and goes to the badge creator 2. The admin selects the email/username of the user needing a badge 3. The admin selects the role of the user including the locations they have access to if it isn't already stored. 4. The admin selects the 'begin scan' button 5. The badge scanner begins scanning for an RFID card 6. The admin holds the user's card up to the scanner 7. If the badge is blank the system recognizes the card and associates the desired user's data to the card. 8. If the badge is already associated with a user, trigger alternate flow (1.) 9. The system notifies the admin that the badge has been written to
Alternate Flow	1. Badge has existing data: - System notifies admin that the badge that is being scanned to has an existing user associated to it - System prompts the admin to decide to choose a different badge or overwrite the existing data. - User decides to use a different badge or overwrite the data - The system prompts the user to place the badge again and scans for the card.

	- The system writes to the card with the users' data
Postconditions	9. Success - Badge is associated with a user and their permissions. 10. Failure - User doesn’t have an account to assign to the card, no badge created for this user

Use Case Diagram:

Below is a use case diagram for GateKeeper. The elements interacting with the application from the front end are new and existing users. Admins and the Badge Server set up the badges and accounts, while also performing the authentication method. It is the badge server that calls for this authentication to be performed.

Figure 2



Testing Plan:

Overview

This section describes the plan for testing the code and architecture of the GateKeeper application. It details our methodology, the scope of our testing, the objectives we seek to achieve, test logs and a review.

Methodology

To assure a smooth development process while minimizing the number of bugs and vulnerabilities in our application. To ensure that we are developing a safe and working product, we will incorporate tests into change management and be sure not to push untested code into development. We will carry out unit testing on all views of the application and end to end system testing to make sure all dependencies are accounted for and functioning correctly. We chose this methodology because as an application, as the code is being updated, it is important to make sure nothing goes wrong, so we don't begin building on top of a broken product. In addition, the server needs to be able to handle sessions with different mobile devices, so performing end to end system testing will help us find errors in networking or connection management.

Scope:

Software:

- Create Account
- Login to Account
- Edit Account
- Delete Account
- Get Location Data
- Call faceID API
- Get Network Data
- Badge registration
- Communication between database and swift application via hosted JSON

UI:

- Manual unit testing of all app UI views

Hardware/Architecture:

- Testing the interfaces between the following subsystems:
    - Badge scanner and server
    - Server and external mobile devices

Objectives:
Testing will be carried out on a regular basis, with any bugs being documented and shared with the team at the time of discovery. Correction of discovered bugs will then be discussed and added into the weekly workflow. The end goal of our testing will be for:

a. All major features and use cases need to be accounted for
b. All use cases must account for all the user roles
c. All major bugs need to be resolved before the IT Expo

Test Logs and Procedures:

Table 3

Item #	Test Case #	User Role	Expected Output	Actual Output	Pass/ Fail	Pass/Fail Reason	Date
1	1 – Auth Pass	User	User auth successfully	User Failed auth	Fail	Pulled wrong long/lat Data	1/20/22
1	2 – Auth Pass	User	User auth successfully	User passed auth	Pass	Auth function passed	1/21/22
1	3 – Auth Pass	User	User auth successfully	User passed auth	Pass	Auth function passed	1/21/22
2	1 – Auth Fail	User	User fail auth	User failed auth	Pass	Auth function failed	1/21/22
2	2 – Auth Fail	User	User fail auth	User failed auth	Pass	Auth function failed	1/21/22
2	3 – Auth Fail	User	User fail auth	User failed auth	Pass	Auth function failed	1/22/22
3	1 – Register User	Admin	User created in system				
3	2 – Register User	Admin	User created in system				
4	1 – Update database	Admin	New database values populate				

4	2 – Update database	Admin	New database values populate				
5	1 – Pull JSON from web server	System	JSON usable in the application	JSON not parsed	Fail	JSON not parsed	2/1/22
5	2 – Pull JSON from web server	System	JSON usable in the application	JSON parsed, not usable in function	Fail	JSON error, can't escape closure	2/4/22

Testing Review:

Various tests were carried out including testing of the User functionality and capability to perform an authentication successfully. Various bugs regarding how data was being passed from the database to JSON to Swift and to the function were found. Flaws prohibiting our communication with the badge server application were discovered as syntax errors in the swift code. Hours of testing and adjusting allowed us to figure out how to properly move data back and forth from our Swift mobile application, and our MySQL database.

Change Management Plan:

As part of our change management plan, anybody can make a change to their own local copy of the code and test it. Once they want to push it to production, they can send it to the rest of the team and have somebody else approve of it as well. Once tested and approved the change can be pushed into production. If it is a change that will greatly affect the way the app works, it will be looked over multiple times by multiple group members. We also want to make sure we are leaving comments, especially when we are changing or adding code.

For visual/UI changes the team will come to conclusions as a group on what changes to make. As far as code itself and different ways to perform a task, the people working on that task will decide between each other which changes should be made and why. There will be complete transparency about any change being put into place, and the level of criticality of that change can be elaborated on by the proposing team member. All changes, once approved, will be tested to verify all systems function as they should before moving this new change to the production environment.

Budget:

Our budget consists primarily of labor costs for our 5 developers. They will be paid a base developer salary and working full time. In addition to this we have Adobe XD for planning UI visuals and costs associated with a hosted database that we can access from anywhere.

Figure 3

	Rate Per/Hr.		Annual Salary	Hours Weekly		Ongoing Annual
Labor - IT	34	(5x) Software Dev	$ 70,000	40	-	$ 350,000 -
Labor - External	-	-	-	-	-	$ 70,000 -
Software - External	-	Adobe XD	$120	-		$ 70,120
Hardware - External	-	Hosted Database	$140	-	-	$ 70, 260
Misc.						
TOTAL			$ 70,260			$ 350,260

Problems Encountered and Analysis of Problems Solved:

Problem: Having a team with little development experience
Solution: Spending hours learning Swift, SwiftUI, Java, Angular, etc.

Problem: Not being physically close to teammates
Solution: Setting up times throughout the week to meet up in a live chat feed on discord and discuss where we are and what needs to be done.

Problem: Having to build a physical side for the app to scan badges from
Solution: Using a badge scanner and extra hardware like laptops and a raspberry pi to establish a server and build a network.

Conclusion:

There were several lessons learned so far. Below is a list of lessons learned, and skills developed:

- Developing with Swift and SwiftUI
- Using Java and Angular
- Hosting JSON on a webserver and parsing it to usable code in Swift
- Setting up databases
- Team and time management
- Wireframe and UI design
- Networking from badge scanner to server to mobile app to database
-

There are still missing implementations that could be added to GateKeeper if we were to pursue it further. However, we are satisfied with the current state of the app and feel it correctly demonstrates our initial goals for it. Gatekeeper is still lacking security countermeasures as we focused more on app development and functions.

References

"Idemia - 5202-000010-05 - MorphoWave Black Tower Frictionless Biometric, (Each)." n.d. Accessed April 24, 2022.

Choi, Younsung. 2017. "Security Weakness of Efficient and Secure Smart Card Based Password Authentication Scheme" 12 (7): 5.

Patil, Sonali, Komal Bhagat, Susmita Bhosale, and Madhura Deshmukh. 2015. "Intensification of Security in 2-Factor Biometric Authentication System." In 2015 International Conference on Pervasive Computing (ICPC), 1–4. https://doi.org/10.1109/PERVASIVE.2015.7087058.