
Savage
Security

1. Project Idea Proposal

Group #2

Project Overview

Project Idea Title:
SEP Project - Tag sensing system for security

Proposed By (Team Lead and Members' ID and Names):
Daniel Ebrahimian - 12961660 (Leader)
Miriam Tym - 12577692
Cemal Ulas Kundakci - 12479248
Jingyi Yang - 99181859
Phone Myat Naing - 12415562
Senghuot Lay - 12632776

Date:
Spring 2018

Team/ Organisation Name:
FEIT/ UTS

Approved by (Customer/ Program Manager):
Dr. Asif Q. Gill

1. Project Description
Group #2 will be developing a security tag sensing system, the system will log who has entered, how long they have been there for and when they leave. During our initial meeting with the group, the project brief was documented. This brief encompasses the technologies being used, the objectives and the end product being worked towards.

2. Planned Duration and Budget
To deliver this project the group will be implementing the Agile approach. The Agile project will have Two short releases. Each release will have Three standard Sprints. The estimated cost for this project is around $200, this includes the price of the Arduino if the group is unable to borrow the necessary equipment.

3. Project Objectives
Implement a maximum security door lock. Provide a security access control system with NFC sensoring and electronic doors. Also provide logging for entries, admin view and control features.

4. Project Scope (Target System)

In Scope:

- Automatically open the door when a key is tapped on and the key card is authorized
- Statical information for logging of entries and exits with timestamps and IDs
- Log in and log out of the web page, Admin views for visualising database logs
- Sound identification when the correct or incorrect key is logged
- Exact Key ID Authorization of the door
- Report generation for weeks or for each person
- Lock and unlock the door from admin view

Out of Scope:

- Facial recognition
- Camera sensor
- In the event of fire or emergency, lock release
- Asking Alexa for the current status of the door or how many people have entered/exited for the day
- Mobile Application development

5. Risks, Constraints and Dependencies

Risk:

- The User can log in as an admin without having the correct password
- The Door will open even without the correct key ID
- The Door would not open without the correct key
- No sound provision that would allow the user to recognize if it is open or incorrect password
- The door would not automatically open
- Arduino short-circuiting or breaking
- No internet access, therefore no access to the door
- RF interference
- Equipment malfunction
- Malicious signal jammers
- Arduino battery exploding

Constraints

- The project must be developed, tested and implemented prior
- Limited full stack web development/IoT/Arduino and MQTT knowledge
- Limited access to the device as it could require additional cost
- Compatibility of full stack development with IOT
- All current developers are new to the environment

Dependencies

- Arduino is dependent on internet and electricity
- Database access depends on Node JS and Express JS along with MQTT
- MQTT is used to communicate with IoT devices
- The environment is dependant on correctly configured docker container

2. Introduction

In this project, we will be developing a security system that utilizes IoT (Internet of Things) and a web application. This hybrid platform will be used for authenticating users when they try to enter a room, such as one in a corporate building and unlock the door for them. Users will interact with the security system via an NFC card. The aim of this documentation is to demonstrate our design and management process towards creating this application.

2.1 Members

Name	Student ID	Role
Daniel Ebrahimian	12961660	Team leader & Technical Lead
Miriam Tym	12577692	Scribe
Cemal Ulas Kundakci	12479248	Scrum Master
Jingyi Yang	99181859	Quality Assurance
Phone Myat Naing	12415562	Scribe
Senghuot Lay	12632776	Tester

3. Agile Requirements

3.1 Team Charter

3.1.1 Team Purpose

The purpose of this assignment is to create a functioning maximum security door lock. Provide a security access control system with NFC sensoring, and electronic doors. Provide history management for entries, administrative view and control features. We will also create comprehensive paperwork along with the security system.

3.1.2 Team Member Knowledge, and Skills

- **Daniel Ebrahimian:**

Daniel is very experienced in C#, Python, C++, Objective-C, Java and Swift. Has a background in teaching and tutoring, this would make him a suitable Project Leader or Technical Lead.

- **Senghuot Lay:**

Senghuot has C# and ASP.NET knowledge, past experience as a project leader/Scrum Master in Software Engineering Studio, well versed with front-end design using bootstrap and interaction design principle.

- **Jingyi Yang:**

Professional Skill Set: C#, JavaScript, W3C Validated HTML5, BEM/CSS3 via Sass
Also, I have achieved to gain experience in SQL. I have also a knowledge of database programming, Data Structures, and Algorithms. I have the understanding of creating software applications for Enterprise environment and design principles. Proficient in human computing interaction and usability testing.

- **Cemal Ulas Kundakci:**

Cemal has many years of experience as a software developer with Java and C# knowledge, has good theoretical understanding based on computer science. Good at academic writing.

- **Phone Myat Naing:**

Phone has a thorough understanding of C# and Java. He has also decent knowledge and experience in academic writing.

- **Miriam Tym:**

Miriam has experience coding in Java, Swift, Angular. She is also competent coding in HTML, although not very experienced working with databases and Microsoft Azure she will be able to pick it up quickly. Miriam is proficient at academic writing and has spent many years developing this skill.

3.1.3 Member Roles

- **Daniel Ebrahimian: Project Leader & Technical Lead**

Daniel has taken the role of the project leader and the technical lead. As project leader he is tasked with planning, executing and promoting activities that a project undertakes. He is also the Technical Lead as such he leads the development team. He coordinates and schedules meetings and focuses on any development team issues.

- **Cemal Ulas Kundakci: Scrum Master**

Cemal is undertaking the role of the Scrum Master, as such, he is the coordinator for the agile software development approach. He must communicate with the team to reach a consensus as to what needs to be completed during each sprint. Similarly, he must keep the team focused on work and protect the team from any outside distractions.

- **Jingyi Yang: Quality Assurance**

Jingyi has been tasked with the role of Quality Assurance, he must ensure the project meets or exceeds the scope of the project. He needs to thoroughly analyse the project to ensure it meets

the group's standards. He must also identify if it does not meet these standards and will be tasked with bringing it to the group's attention for it to be fixed.

- **Senghuot Lay: Tester**

Senghout's role for the duration of this project is that of the Tester his job is to create tests for the system to ensure there are no issues or faults within the system. He will create the test based on the system created then implement them at each release and input the data into a defect log.

- **Phone Myat Naing and Miriam Tym: Scribe**

Phone and Miriam have been assigned the role of scribe for this assignment. The scribe is tasked with taking notes throughout every meeting and creating agendas for future meetings to ensure there are set goals we can complete before and during each meeting. They were given this role as they good listeners and can take thorough notes throughout meetings.

- **All members: Scrum Developer**

All members of the group are considered Scrum Developers, they will be coding and creating the functional project. Each member will have some input into the project whether it be large or small.

3.1.4 Project Rules and Consequences

The project consequences describe ramifications on project members should they fail to adhere to the following rules:

- Complete work assigned to you.
- Show up to all group meetings unless stated previously that you can't attend.
- Do not slack off during group meetings.
- Do not submit work that hasn't been approved by other group members.

The possible consequences if these rules are broken are shown below in the order of severeness:

- A group discussion with the culprit discussing the issues and explaining the need to change their actions.
- A bad peer review.
- Possible expulsion from the group.

Should any of the above become necessary the Team member's name and actions will be noted here, as well as the group's necessary response.

3.1.5 Desired End Result

The desired end result of this assignment is to allow a user to scan their NFC and have the have the system either open the door or reject their request. The door should then close behind the user to ensure no one else is admitted after them. History information of door entries should be accessible from a web interface that supports filtering. The documentation created alongside this application should be in-depth and thoroughly cover all the designing and planning that went into creating the system.

3.2 Requirements

3.2.1 Functional Requirements

This project has the following functional requirements:

User Story #	As a/an	I want to	So that	Estimates	Priority	Staus	Release	Iteration	Process/ Service
101	Admin	will view the system logs from a web interface	to view the history of who unlocked the door.		H		R1	I1	User Access
102	Admin	should be able to control who can access which door	be able to view all the system history for entries.		H		R1	I1	Management
103	Admin	block a security card	prevent a person from entering.		H		R1	I1	Management / User Access
104	Admin	add new sensor	to allow for building expansion or added security.		M		R1	I2	Management / User Access
104	Admin	be able to add a new security card	to authorize a new employee.		H		R1	I2	Management / User Access
105	Admin	should be able to filter history data	to a particular user, date of entry and frequency of access.		M		R1	I2	Management / User Access
106	Admin	allocate access permissions	so people can have different access levels.		M		R1	I2	Management / User Access
107	Admin	replace a lost card	to let an authorized person that has lost their card in.		H		R1	I2	Management
108	Admin	remove a sensor	to remove rooms from the system due to the old doors and building changes.		H		R1	I2	Management / User Access
109	Admin	will have a master key card	has access to every room and physical access to this card will be strictly controlled.		L		R1	I2	Management / User Access
110	Admin	will have the ability to grant access or revoke access	by using the card via the summary interface on the website.		M		R1	I3	Management / User Access
111	System Admin	should be able to retrieve report data	the total number of unique people entering a room for today via an API		H		R1	I3	Management
112	Vistor	will use their NFC cards	to enter the room.		M		R1	I3	User Access

~~Func ID~~	~~Description~~
~~1~~	~~Admin will view the system logs from a web interface that requires a login~~
~~2~~	~~Visitors will use their NFC cards to enter the doors.~~
~~3~~	~~Admin should be able to control who can access which door and be able to view all the system history for entries~~
~~4~~	~~Admin should have the ability to block a security card.~~
~~5~~	~~Admin will be able to add a new door sensor to allow for building expansion or added security.~~
~~6~~	~~Admin should also be able to add a new security card for a new worker~~
~~7~~	~~Admin should be able to filter history data according to a particular user, date of entry and frequency of access~~
~~8~~	~~Admin shall be able to allocate certain permissions to different users depending on their security access.~~
~~9~~	~~Admin shall be able to replace workers lost or stolen security card.~~
~~10~~	~~Admin should have the ability to remove existing doors sensors in the case of old doors and building changes.~~
~~11~~	~~The system will have a master key card that has access to every room and physical access to this card will be strictly controlled.~~
~~12~~	~~External entities should be able to retrieve report data such as the total number of unique people entering a room for today via an API~~
~~13~~	~~Admin will have the ability to grant access or revoke access for a card via the summary interface on the website.~~

3.2.2 Non-Functional requirements

Func ID	Description
1	A printing company will be sent the worker's photo to create a new permanent security access card.
2	When there is a reliable internet connection, the website should reply within 2 seconds.

3	When there is reliable internet, the door should open within 2 seconds.
4	In the event of a networking outage, the Arduino should be able to recover its connection once the network becomes available again (without needing to restart the device)
5	The system should be able to handle at least 25 doors.
6	Online system cannot be accessed by anyone who is not an admin.

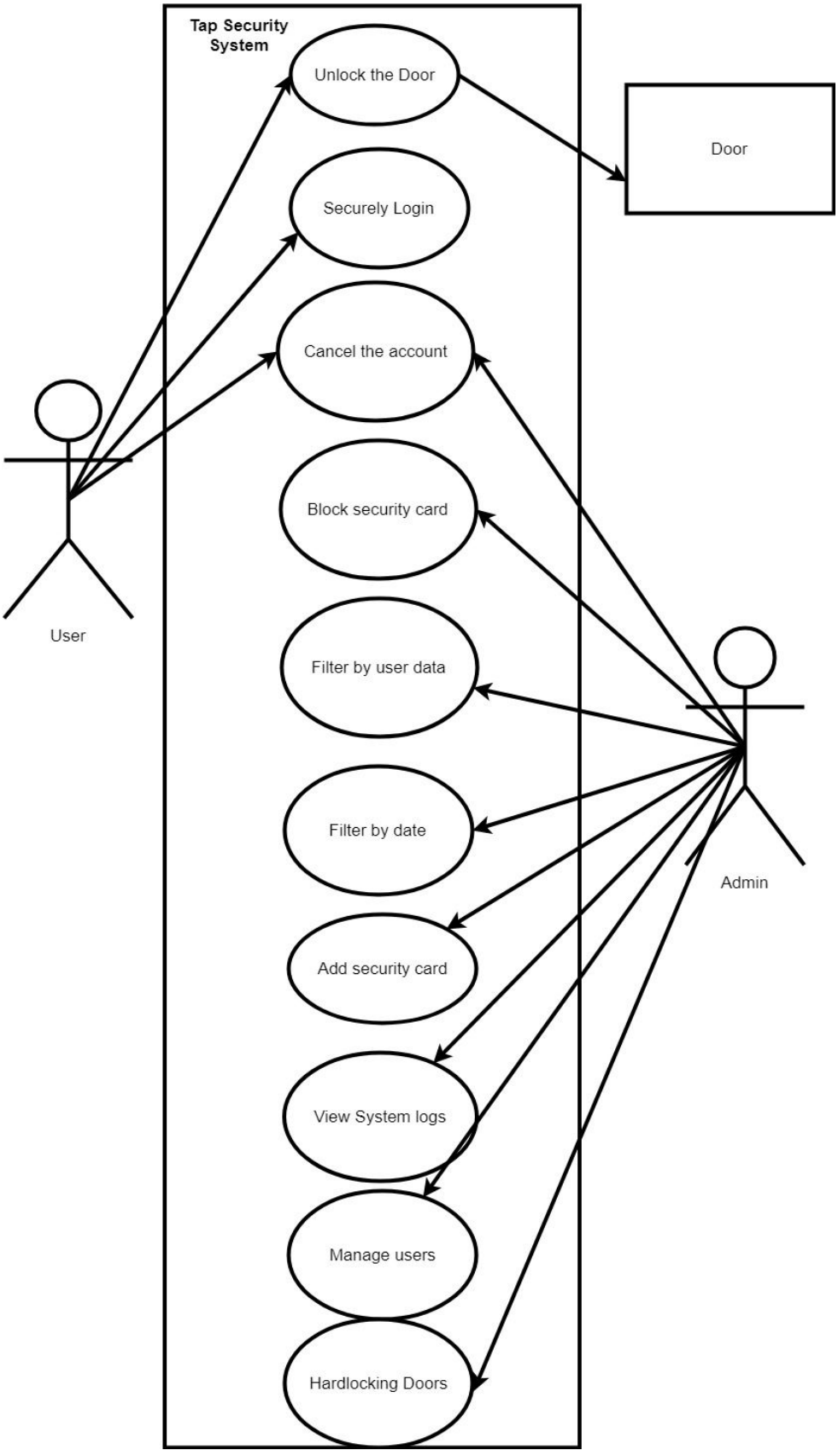
3.3 Data Requirements

3.3.1 Data Dictionary

Ref.	Data Element	Format	Length	Description
01 User				
DE-1001	UserID	ObjectId	12	Uniquely Identifies each user, generated by MongoDB once new user is added to the system. e.g. _id": { "$oid": "5b87a16f061fdf058699bfb9"
DE-1002	FirstName	String	50	First Name for each User
DE-1003	LastName	String	50	Surname for each User
DE-1004	PhoneNumber	String	50	Phone Number for each User
DE-1005	EmailAddress	String	50	Email Address for each User
DE-1006	Role	String	20	Role for the user such as Admin or Non-Admin
02 Card				
DE-2001	CardID	ObjectId	12	Uniquely Identifies each Card, generated by MongoDB once a new card is added to the system. e.g. _id": { "$oid": "5b87a16f061fdf058699bfb9"
DE-2001	CardNumber	Integer	20	Identify which card it is for accessing the room
DE-2001	Status	String	10	Status of the card whether it is active or inactive
03 Access_Request				
DE-3001	RoomEntryID	ObjectId	12	Uniquely Identifies each RoomEntry, generated by MongoDB once any request is made by the user in accessing the room.
DE-3001	Outcome	String	20	The outcome of the request whether the card is rejected or accepted in accessing the room

DE-3001	TimeStamp	TimeStamp	64	Stamp the time when a request is made. E.g 11:05pm 02/August/2018
04 AccessManager				
DE-4001	_id	ObjectId	12	Uniquely Identifies each the Access_Manager, generated by MongoDB to uniquely differentiate between every access manager.
DE-4001	AllowedCard	Array	> 1	An array that holds the ID of accessible cards
05 Room				
DE-5001	RoomID	ObjectId	12	Uniquely Identifies each the Room, generated by MongoDB once a new room is made and registered to the system.
DE-5001	Name	String	20	The name to identify which room is it. E.g. Room 05.505
DE-5001	Location	String	50	The location of the room. E.g. UTS Building 11
DE-5001	isHardLocked	Boolean	10	Determine whether the room is locked and allow no access

3.4 User Narratives



The image above is use case diagram for our project. The system will involve two actors: the user and the admin. The user will interact with the system in terms of logging in, unlocking the door, and cancelling their account. The admin will interact with the system via advanced features such as management of users, viewing of system logs and filtering user data, adding new cards and hard-locking doors.

Use Case ID	**UC101: Open security door using**
User Story	As a user I want to tap my card on the security sensor so that I can enter the room
Goal	Open the door
Priority	High
Actors	User
Pre-conditions	Have a security card
Post-conditions	User has entered the room successfully.
Trigger	Card tapped on sensor.
Main Flow	1. Card is tapped on the sensor 2. Card access is checked 3. If they have to correct permissions the door will open and the user can enter.
Exceptions	E1. Step 3. The User has incorrect permissions so the door will remain closed.
Non-Functional Requirements	Performance: Door will open within 5 Seconds

Use Case ID	**UC102: View all entries by certain users**
User Story	As an admin I want view all entries made by a certain person so I can track discrepancies
Goal	View access history for a person
Priority	High
Actors	Admin

Pre-conditions	Admin is logged in. The person exists in the system. The person has entries in the system.
Post-conditions	Admin has seen the entries made by a certain person
Trigger	Admin searches a user in the web interface
Main Flow	1. Admin selects user search 2. Admin enters person's name into filter 3. Admin selects show only entries
Exceptions	User doesn't exist
Non-Functional Requirements	Fast searching speed

Use Case ID	**UC103: Block a chosen security card**
User Story	As an admin I want to block a security card so that I can prevent unauthorized entries
Goal	Block the use of a certain security card.
Priority	High
Actors	Main - Admin
Pre-conditions	Admin is logged in. The person exists in the system.
Post-conditions	Admin successfully blocks the users card.
Trigger	Admin
Main Flow	1. Admin logs into the system 2. Find the security card to block 3. Block the chosen card.
Exceptions	E1. Step 2. Security card cannot be found.
Non-Functional Requirements	

Use Case ID	**UC104: View entries during a certain time range**
User Story	As an admin, I want to view the number of entries made in specific

	time range
Goal	View the entries made in a specific time block
Priority	Low
Actors	Admin
Pre-conditions	Admin is logged in. The person exists in the system
Post-conditions	The system successfully provides the admin with correct door entries that have been made for that time frame
Trigger	Admin
Main Flow	1. Admin logged in to the system 2. Admin views the history of the entries that have been made 3. Admin filters the history for a particular time frame from a particular date 4. The System shows the admin a correct query result
Exceptions	4. The system does not provide any result
Non-Functional Requirements	The System should respond within 2 seconds.

Use Case ID	**UC105: View Door history**
User Story	As an admin, I want to view the history of a certain door, so that I can generate reports.
Goal	Viewing database records between certain time range
Priority	High
Actors	Admin
Pre-conditions	Admin is in the system and logged in.
Post-conditions	Admin has seen the records for a certain room
Trigger	Admin
Main Flow	5. Admin logged in to the system 6. Admin views the history of the entries that have been made 7. Admin filters the history for a particular room 8. The System shows the admin a correct query result

Exceptions	4. The system does not provide any result
Non-Functional Requirements	System should respond within 2 seconds.

Use Case ID	**UC106: Find a certain User**
User Story	Admin wants to view the last door a user opened and how long ago so that security is able to locate a person in the case of an emergency
Goal	Locate a user
Priority	High
Actors	Admin
Pre-conditions	Admin is logged into the system.
Post-conditions	Admin successfully locates the user for emergency services.
Trigger	Admin
Main Flow	1. Admin logs into the system 2. Admin searches for the last door the users card opened 3. Finds the door and notifies emergency services
Exceptions	
Non-Functional Requirements	Admin should be able to locate a user within a minute.

4. Agile Planning

We began planning the progression of our product by starting with an initial sprint that would be used to measure the feasibility of the product. This sprint focused on creating a working physical prototype of the Door Security system. Including, an Arduino with NFC, WiFi, motor and the software to control them at a basic level. As well as a lightweight server to process the authentication requests. Planning the sprint in this way was crucial to identifying if creating this system was feasible and the technologies we chose would work.

The outcome of Sprint 0 determined the trajectory of our agile planning. We identified how the team is able to deliver certain functionality and how diverse our skill sets are in order to achieve deadlines. With this information, we are able to plan our development cycles with an appropriate amount of features per term. The technologies we planned to use changed due to the findings of Sprint 0, this was crucial for identifying problems earlier rather than later.

Our sprints are managed on a Trello board. Each task comes from a user story and may potentially be broken down if the sprint planner believes it will be more efficient to allocate smaller parts to multiple people. Features that block the progression of other functionality or have deliverable deadlines such as an assignment due date are allocated a Due Date field to its Trello card. Labels are used to colour code tasks to make them it easier to visually identify similar-type cards, such as ‘Implementation’ cards. The sprint planner and scrum master work together to identify who should be allocated to which task, we also allow team members to self-allocate a particular task if they desire. Team members are expected to utilize the organized Trello board to transition their tasks between three stages of progression: “To Do”, “Doing’ and “Done”.

Each sprint is reflected on with a retrospective. The insight discovered from the reflection is used to identify better ways of managing sprints. We will prioritize identifying methods of improving time efficiency and understanding of team member strengths. This is important for allocating tasks for the next sprint so we can be confident that we will complete our product by the final deliverable deadline.

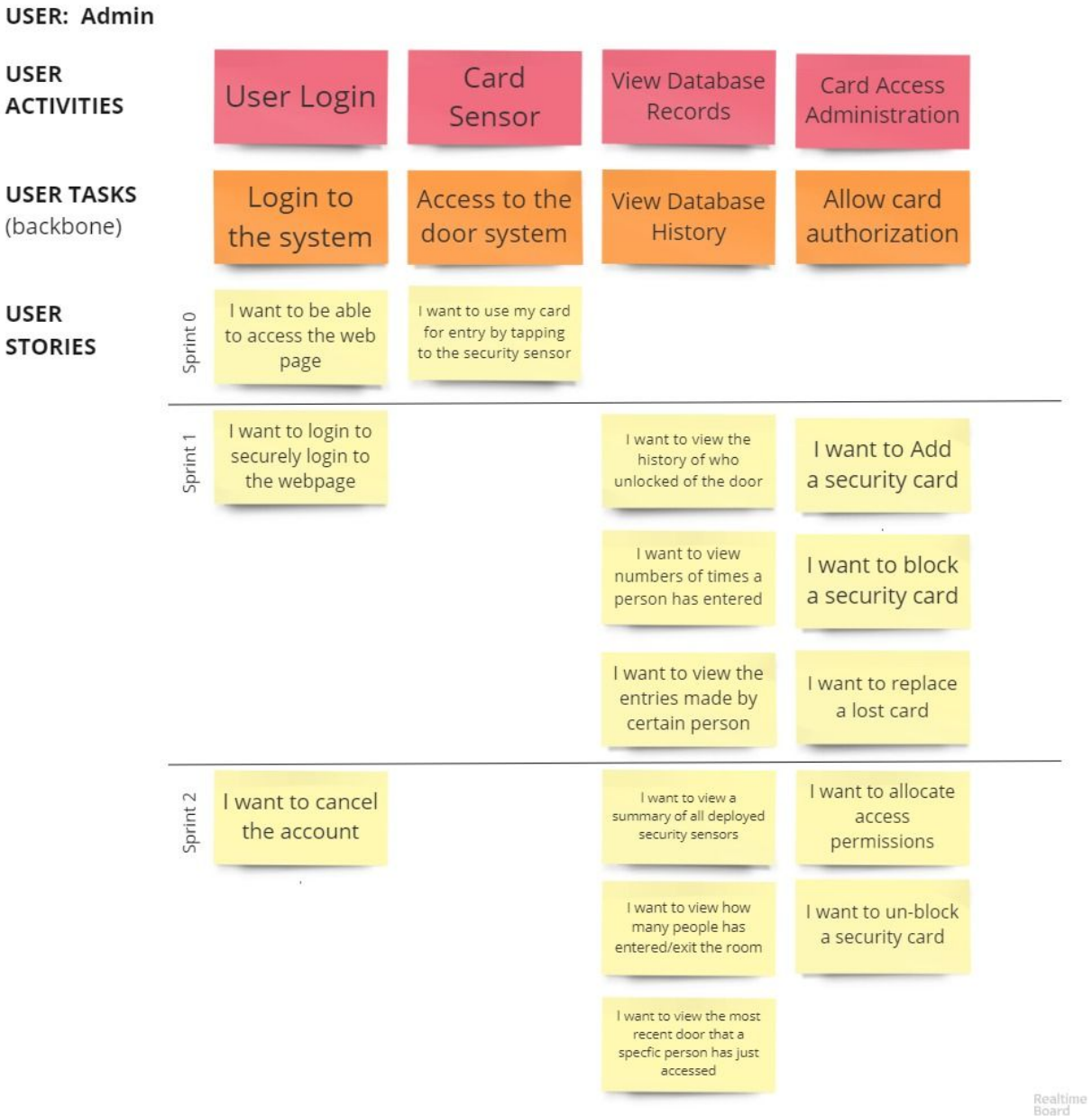
4.1 Project Scheduling

4.1.1 Overarching Schedule

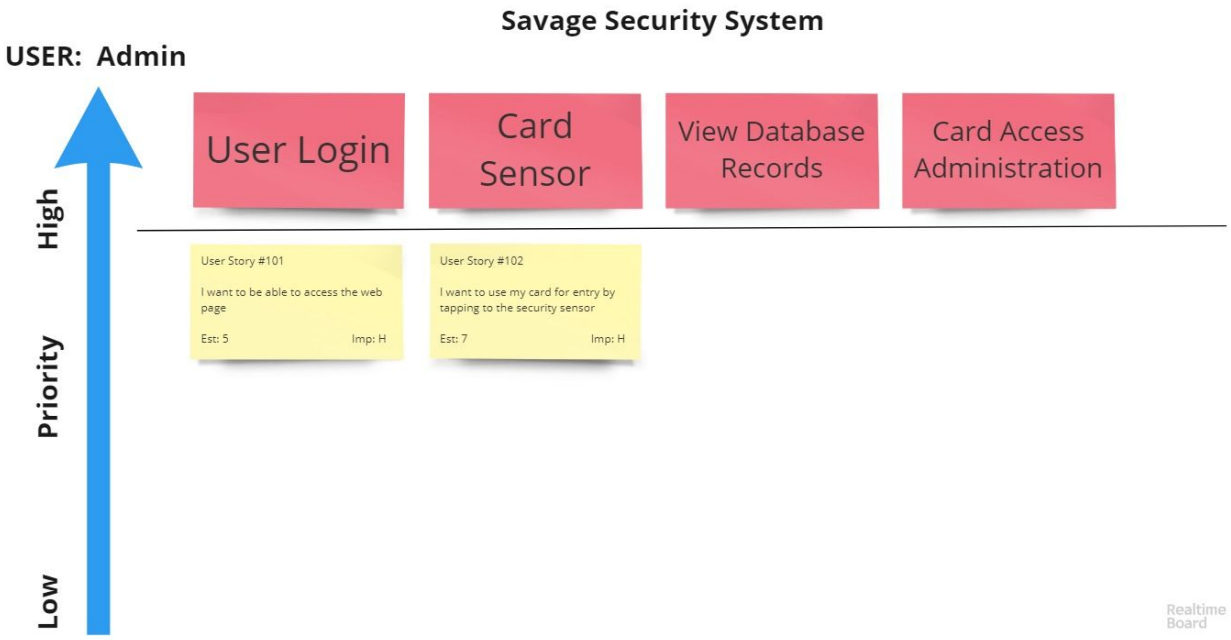
Sprint	Start Date	End Date
Release 1 : 7/8/18 - 3/9/18		
Sprint 0	7/8/18	14/8/18

Sprint 1	14/8/18	21/8/18
Sprint 2	21/8/18	28/8/18
Release 2 : 3/9/18 - 15/10/18		
Sprint 0	3/9/18	10/9/18
Sprint 1	10/9/18	17/9/18
Sprint 2	17/9/18	24/9/18

4.1.2 User Story Card Wall



4.1.3 User Story Card Wall for Sprint 0



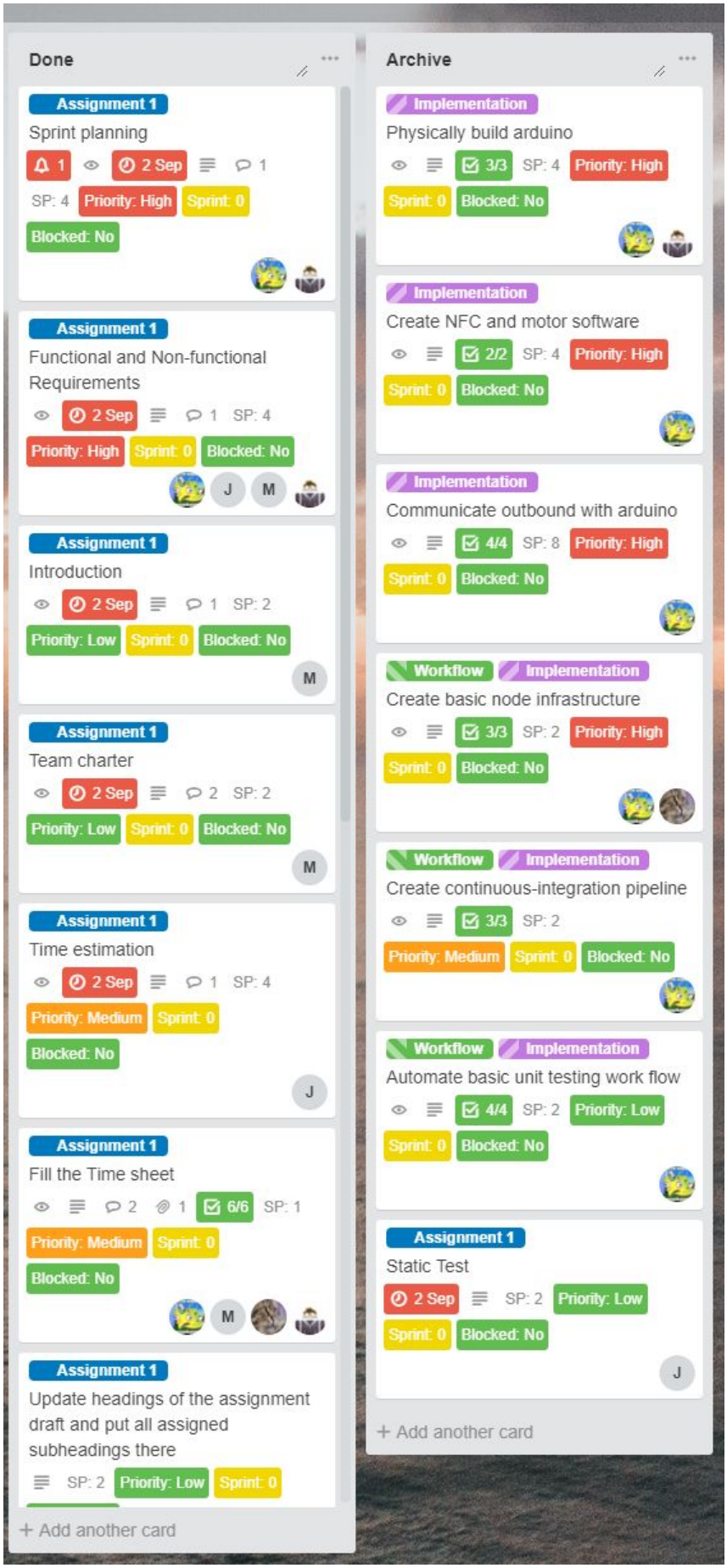
4.1.4 Sprint 0 Scheduling

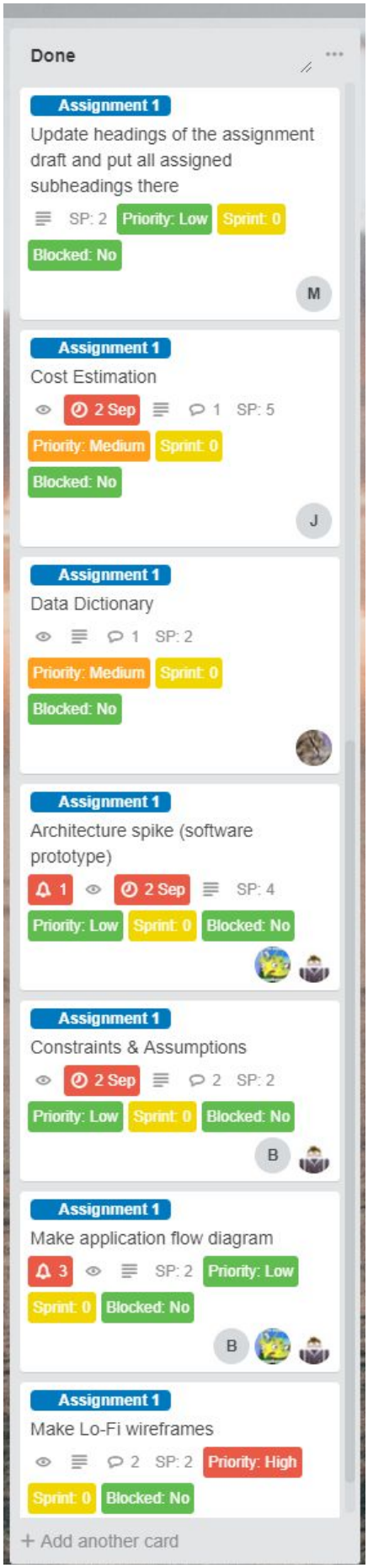
Item	Duration (H)	Start Date	End Date	Member
Application infrastructure	4	27/8/18	3/9/18	Cemal
Data Architecture	4	20/8/18	27/8/18	Senghuot
Project Schedule	6	27/8/18	3/9/18	Jingyi, Phone Myat
Create Database	8	27/8/18	3/9/18	Cemal, Senghuot
Cost Estimation	5	27/8/18	3/9/18	Jingyi
Time Estimation	4	27/8/18	3/9/18	Jingyi
Sprint Backlog	8	20/8/18	27/8/18	Daniel, Miriam, Phone Myat
Change control	2	27/8/18	3/9/18	Cemal
Risk (FMEA RISK)	4	20/8/18	27/8/18	Jingyi, Senghuot
Project Scope	2	20/8/18	27/8/18	Cemal, Miriam, Daniel
User Narratives	2	20/8/18	27/8/18	Phone Myat
Sprint planning	4	20/8/18	27/8/18	Cemal, Daniel

Functional & Non-Functional requirements	4	20/8/18	27/8/18	Jingyi, Cemal, Daniel, Miriam
Introduction	2	20/8/18	27/8/18	Miriam
Team Charter	2	20/8/18	27/8/18	Miriam
Data Dictionary	2	27/8/18	3/8/18	Senghuot
Architecture Spike (Software Prototype)	4	27/8/18	3/8/18	Daniel,Cemal
Constraints & Assumptions	2	27/8/18	3/8/18	Phone Myat, Cemal
Make application Flow Diagram	2	27/8/18	3/8/18	Cemal, Senghuot
Lo-Fi Wireframes	2	27/8/18	3/8/18	Phone Myat, Cemal, Senghuot, Miriam

4.2 Sprint Backlog

During Sprint 0, we have completed many foundation tasks. Particularly to web infrastructure, continuous-integration and Arduino hardware and software. The following are the completed tasks from our Sprint board.


Done
Assignment 1
Sprint planning
1
2 Sep
1
SP: 4
Priority: High
Sprint: 0
Blocked: No
Assignment 1
Functional and Non-functional Requirements
2 Sep
1
SP: 4
Priority: High
Sprint: 0
Blocked: No
J
M
Assignment 1
Introduction
2 Sep
1
SP: 2
Priority: Low
Sprint: 0
Blocked: No
M
Assignment 1
Team charter
2 Sep
2
SP: 2
Priority: Low
Sprint: 0
Blocked: No
M
Assignment 1
Time estimation
2 Sep
1
SP: 4
Priority: Medium
Sprint: 0
Blocked: No
J
Assignment 1
Fill the Time sheet
2
1
6/6
SP: 1
Priority: Medium
Sprint: 0
Blocked: No
M
Assignment 1
Update headings of the assignment draft and put all assigned subheadings there
SP: 2
Priority: Low
Sprint: 0
+ Add another card
Archive
Implementation
Physically build arduino
3/3
SP: 4
Priority: High
Sprint: 0
Blocked: No
Implementation
Create NFC and motor software
2/2
SP: 4
Priority: High
Sprint: 0
Blocked: No
Implementation
Communicate outbound with arduino
4/4
SP: 8
Priority: High
Sprint: 0
Blocked: No
Workflow
Implementation
Create basic node infrastructure
3/3
SP: 2
Priority: High
Sprint: 0
Blocked: No
Workflow
Implementation
Create continuous-integration pipeline
3/3
SP: 2
Priority: Medium
Sprint: 0
Blocked: No
Workflow
Implementation
Automate basic unit testing work flow
4/4
SP: 2
Priority: Low
Sprint: 0
Blocked: No
Assignment 1
Static Test
2 Sep
SP: 2
Priority: Low
Sprint: 0
Blocked: No
J
+ Add another card

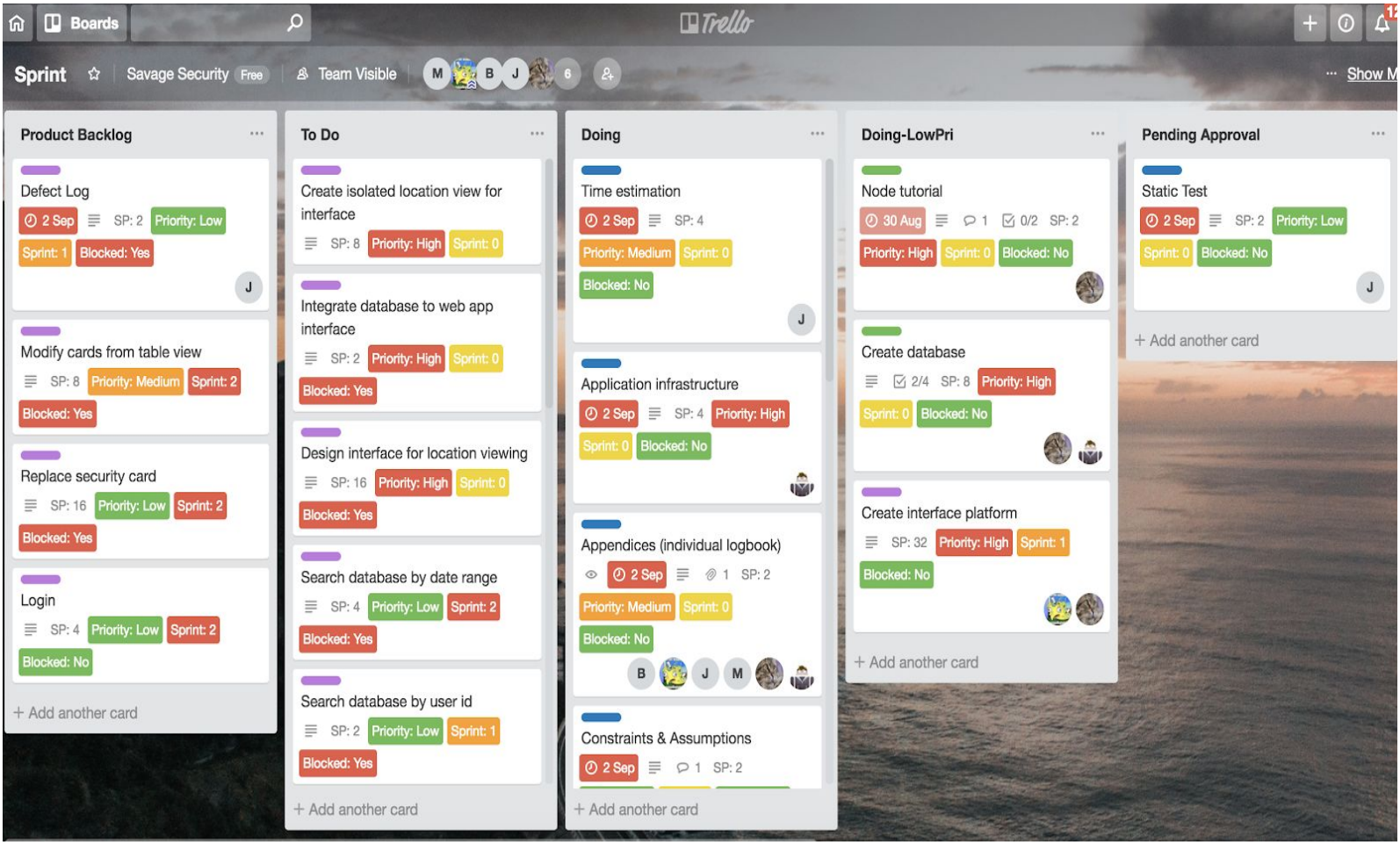

Done
Assignment 1
Update headings of the assignment draft and put all assigned subheadings there
SP: 2
Priority: Low
Sprint: 0
Blocked: No
M
Assignment 1
Cost Estimation
2 Sep
1
SP: 5
Priority: Medium
Sprint: 0
Blocked: No
J
Assignment 1
Data Dictionary
1
SP: 2
Priority: Medium
Sprint: 0
Blocked: No
Assignment 1
Architecture spike (software prototype)
1
2 Sep
SP: 4
Priority: Low
Sprint: 0
Blocked: No
Assignment 1
Constraints & Assumptions
2 Sep
2
SP: 2
Priority: Low
Sprint: 0
Blocked: No
B
Assignment 1
Make application flow diagram
3
SP: 2
Priority: Low
Sprint: 0
Blocked: No
B
Assignment 1
Make Lo-Fi wireframes
2
SP: 2
Priority: High
Sprint: 0
Blocked: No
+ Add another card

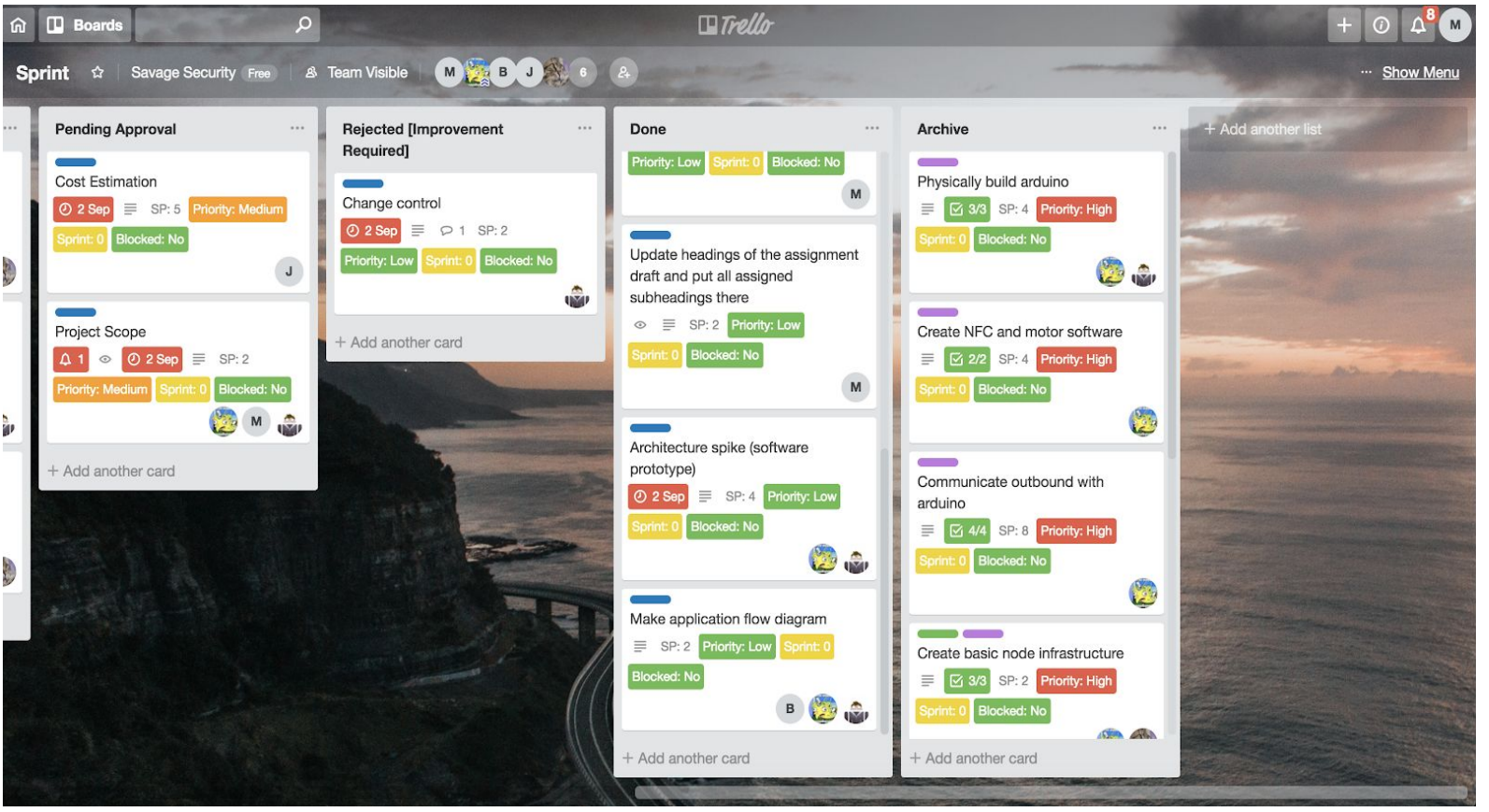
4.2.1 Trello Sprint Backlog

The Trello Sprint Backlog will comprise of a couple of columns such as Product Backlog, To Do List, Doing List, Doing-LowPri, Pending Approval, Rejected, Done and Archive. Each sprint will contain product backlogs and each of them will have its own specific priority, story points (in hours), due date and dependency. Sprint 0 will be the start of our sprints and Sprint 2 will be the end of our sprints. Currently, we are in Sprint 0 and will be proceeding to Sprint 1 at mid-semester.

To further explain how our sprints would work in Trello, each user story will be assigned to an individual or a group of team members. The tasks will be given accordingly to the current sprints and priority. The user story will have their specific due dates. Once a user story has been achieved, it will then be proceeded through “Pending Approval”. This is where every team member discusses and analyzes the completed feature and decides on whether or not it should be approved or changed. The tasks also contains a series of checklists so that it meets the requirements of completion. After it’s approved, it will then be sent to “Done”.

In Sprint 0, we have accomplished a number of high priority cards that were vital to initiating this project. Our highest priority was to obtain an Arduino as it is the main hardware that is required for a door security system. After that, our software team has created NFC and motor software and have performed communication checks with the Arduino and proved to be successful. Basic node infrastructures and continuous-integration pipelines have been created as well. These are just one of the few high priorities that have been accomplished in the past few weeks and more will be added as the project nears completion.


Boards
Trello
Sprint
Savage Security Free
Team Visible
Show M
Product Backlog
Defect Log
2 Sep SP: 2 Priority: Low Sprint: 1 Blocked: Yes
Modify cards from table view
SP: 8 Priority: Medium Sprint: 2 Blocked: Yes
Replace security card
SP: 16 Priority: Low Sprint: 2 Blocked: Yes
Login
SP: 4 Priority: Low Sprint: 2 Blocked: No
+ Add another card
To Do
Create isolated location view for interface
SP: 8 Priority: High Sprint: 0
Integrate database to web app interface
SP: 2 Priority: High Sprint: 0 Blocked: Yes
Design interface for location viewing
SP: 16 Priority: High Sprint: 0 Blocked: Yes
Search database by date range
SP: 4 Priority: Low Sprint: 2 Blocked: Yes
Search database by user id
SP: 2 Priority: Low Sprint: 1 Blocked: Yes
+ Add another card
Doing
Time estimation
2 Sep SP: 4 Priority: Medium Sprint: 0 Blocked: No
Application infrastructure
2 Sep SP: 4 Priority: High Sprint: 0 Blocked: No
Appendices (individual logbook)
2 Sep 1 SP: 2 Priority: Medium Sprint: 0 Blocked: No
Constraints & Assumptions
2 Sep 1 SP: 2
+ Add another card
Doing-LowPri
Node tutorial
30 Aug 1 0/2 SP: 2 Priority: High Sprint: 0 Blocked: No
Create database
2/4 SP: 8 Priority: High Sprint: 0 Blocked: No
Create interface platform
SP: 32 Priority: High Sprint: 1 Blocked: No
+ Add another card
Pending Approval
Static Test
2 Sep SP: 2 Priority: Low Sprint: 0 Blocked: No
+ Add another card


Boards
Trello
Sprint
Savage Security Free
Team Visible
Show Menu
Pending Approval
Cost Estimation
2 Sep SP: 5 Priority: Medium Sprint: 0 Blocked: No
Project Scope
1 2 Sep SP: 2 Priority: Medium Sprint: 0 Blocked: No
+ Add another card
Rejected [Improvement Required]
Change control
2 Sep 1 SP: 2 Priority: Low Sprint: 0 Blocked: No
+ Add another card
Done
Priority: Low Sprint: 0 Blocked: No
Update headings of the assignment draft and put all assigned subheadings there
SP: 2 Priority: Low Sprint: 0 Blocked: No
Architecture spike (software prototype)
2 Sep SP: 4 Priority: Low Sprint: 0 Blocked: No
Make application flow diagram
SP: 2 Priority: Low Sprint: 0 Blocked: No
+ Add another card
Archive
Physically build arduino
3/3 SP: 4 Priority: High Sprint: 0 Blocked: No
Create NFC and motor software
2/2 SP: 4 Priority: High Sprint: 0 Blocked: No
Communicate outbound with arduino
4/4 SP: 8 Priority: High Sprint: 0 Blocked: No
Create basic node infrastructure
3/3 SP: 2 Priority: High Sprint: 0 Blocked: No
+ Add another card
+ Add another list

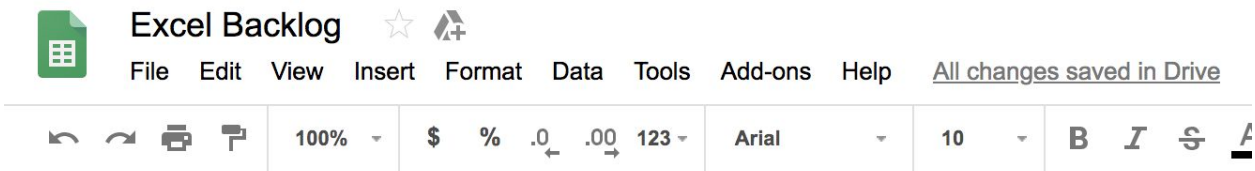
There are many available options to create our sprint backlog, software such as Trello, Excel spreadsheets and GitLab Boards. Each has many positive and useful features but we can only use one. In the end, we decided to use Trello to create our sprint backlog. Our reasoning is described below.

4.2.2 Trello

Trello was chosen as it has a very intuitive and easy to use UI, you have the ability to drag and drop cards into any of the lists you have created. Similarly, you can alter each card with a coloured tag, this can represent the sprint, the priority, and if it is blocked or not. It's very easy to make changes and move cards around. Furthermore, users are emailed whenever there is a change made to a card or if it is moved into a new list. For example, if I moved time estimation into pending approval everyone would be emailed. This would notify them to check the time estimation and approve it or move to rejected. This would tell me that it needs more work before it will be approved. With all these added features it's very obvious as to why we chose Trello instead of Microsoft Excel or GitLab Boards.

4.2.3 Microsoft Excel

Microsoft Excel was not used due to the difficulty of moving cards into different lists. You are unable to drag and drop as with Trello, instead, you are forced to copy and paste, or delete the old one and write it again in the new sprint. Similarly, it's very hard to create tags in excel. To add the priority to each card you have to copy and paste into every one. There is also no easy way to obviously differentiate these tags. Excel also lacks an easy way for members to see what's been changed. Unlike with Trello team members are not emailed when changes are made. As such unless you knew where exactly everything on the spreadsheet before you may not notice what has been changed. Due to these reasons Trello was the obvious choice.



	A	B	C	D	E	F
1	Done	Doing	Pending Approval	To do	Product Backlog	
2	Sprint planning Priority: High	Sprint backlog Priority: High	Cost Estimation Priority: Medium	Create isolated location view for interface Priority: High	Replace security card Priority: Low	
3	Functional and Non-Functional requirements Priority: High	Data architecture Priority: Medium		Integrate database to web app interface Priority: High	Defect Log Priority: Low	
4	Introduction Priority: Low	Make Lo-Fi wireframes Priority: High		Design interface for location viewing Priority: High	Modify cards from table view Priority: Medium	
5	Team Charter Priority: Low			Search database by date range Priority: Low	Login Priority: Low	
6	Update headings of the assignment draft and put all assigned subheadings there Priority: Low			Search database by user id Priority: Low		
7	Architecture spike (software prototype) Priority: Low					

4.2.4 GitLab Boards

We investigated the possibility of using Gitlab boards as a feature management board. We had a terrible initial experience as we encountered many bugs and breakages with the interface. Simple tasks such as creating and moving cards and when adding a weighting would freeze the interface. We decided this was a rather new feature for a platform that doesn't specialize in this functionality and chose to use the more mature platform, Trello.

4.3 Risks

Risk Analysis Form							
Facilitators: Senghuot Lay / Jingyi Yang Team members: Daniel Ebrahimian, Cemal Model/Product: Savage Security			Prepared By: Senghuot Lay Document review: Joseph Assistance provided by: All team members				
Risk ID	**Quality Risk Category**	**Failure Model/Quality Risk/Effect**	**Severity**	**Priority**	**Risk Priority Number**	**Recommended Action**	**Requirement Tracing**
Risk Category 1	Software Development						
R001	Software Development	Actual progress of group not aligning with the planned progress	4	3	12	Identify the problem with the team member and try to find a solution.	
R002	Software Development	Developer's vision does not align with the client's	1	2	2	Having a meeting between the client and the team to ensure both are aligned with the functionality where possible, what is good and what can be implemented	
R003	Software Development	The difficulty of a requirement implementation is underestimated	2	2	4	Consult with team members first if anyone can pick up on or assigned together. However, if the problem still occurs, consult with	

						the client to discuss the feasibility of this feature.	
Risk Category 2	Team Management						
R004	Team Management	The team member was sick or does not have enough abilities to work on their assigned task	3	2	6	Ask another team member who is willing to and has the capabilities to complete the task	
R005	Team Management	Conflicting group ideas	3	1	3	Be democratic where everyone is allowed to have opinions and votes on the idea whether they are good or bad	
R006	Team Management	Lack of team direction	3	3	9	Refer back to the goals listed on the team charter and consult with Project Leader	
R007	Team Management	Incompletion of allocated tasks by team members	2	2	4	Find the reason why it is incomplete. Reallocate the task to one or more team members. Consult with Scrum master and Project Leader to break the task down into smaller portions.	
Risk Category 3	Security						
R008	Security	A card that is authorized for a room but the room still rejects its entry	1	1	1	File a bug report	Func ID 2 // 3.2 Functional Requirement//Savage Security Assessment 1.doc
R009	Security	The user is unable to access the door system without having their valid access card.	3	2	6	File a bug report	Func ID 2 // 3.2 Functional Requirement//Savage Security Assessment 1.doc
R010	Security	Login system fails to prevent an unauthorized user from accessing the platform	2	1	2	File a bug report	Func ID 1 // 3.2 Functional Requirement//Savage

							Security Assessment 1.doc
Risk Category 3	System Functionality						
R011	System Functionality	The server that is used for hosting the website would not load	2	3	6	Refers to Developers	Func ID 1 // 3.2 Functional Requirement//Savage Security Assessment 1.doc
R012	System Functionality	Admin is unable to add/block or replace a card	3	3	9	Refers to Developers	Func ID 6 // 3.2 Functional Requirement//Savage Security Assessment 1.doc
R013	System Operations	The system did not provide any notification to the user such as a beep or click that would allow the user to know that the door is open	2	3	6	Refers to Developers	Func ID 2 // 3.2 Functional Requirement//Savage Security Assessment 1.doc
R014	System functionality	The filter system did not provide robust information regarding the filtration of data, eg. filter or how many accesses for today, while it provides information about yesterday	4	4	16	Refers to Developers	Func ID 7 // 3.2 Functional Requirement//Savage Security Assessment 1.doc
Risk Category 4	Data Integrity						
R015	Data Integrity	An unauthorized entity modifies the authorized card collection	1	1	1	Refers to Developers	Funct ID 6 // 3.2 Functional Requirement//Savage Security Assessment 1.doc
R016	Data Integrity	Leakages in admin credential.	2	2	4	Refers to Developers	Funct ID 1 // 3.2 Functional Requirement//Savage Security Assessment 1.doc

4.3.1 Risk Measurement Matrix

Severity: is the impact of the defect on the application (out of 5); **Note: The lower the number, the more severe the risk is.**

1. Blocker: system crash, corruption of data, critical functionality missing.
2. Critical: failure to save data; important functions missing, security issue.
3. Major: website hangs; script error, functions don't work correctly
4. Important: UI issue; small function issue.
5. Minor: cosmetic problem.

Priority: reflects the impact of the defect on the business and consequently how quickly developers need to find and fix it (out of 5); **Note: The lower the number, the more prioritised the risk should be.**

1. Urgent: to be fixed immediately.
2. High: to be resolved in the next release.
3. Medium: defects the must be fixed before shipment
4. Important: defects that can be fixed after shipment
5. Low: defects are not as strong as desirable, can be deferred to next release.

4.4 Project Scope

4.4.1 In-Scope

ID	Item	Description
IS01	View full and filtered history of the accesses.	Savage Security can view the full history and filter the history by date, user and room
IS02	The ability to locate where certain users are.	Savage Security can search for the latest door entry for every user.
IS03	Add and delete security cards.	Savage security can add new security cards, either for a new user or a new card for a user that lost their old one. They can also delete a security card for a user who no longer works at the company or for a

		lost card
IS04	View how many people people have entered a room.	Savage Security keeps track of how many users have entered a room.
IS05	View how long a user has been in a certain room.	Savage Security has the ability to see how long users spends in each room during a day.
IS06	Modify certain security card permissions.	Savage Security can also change the permissions on certain security cards depending on the level of access allowed to that user.

In order to implement the scope items, components given below are needed.

- Arduino hardware
    - Uno board
    - Male-to-female wires
    - Female-to-female wires
    - 5v motor
    - Motor controller
    - NFC/RFID reader module
    - Wifi Module
    - USB uplink cable
    - Battery
- NFC Keycards
    - UTS student/staff id cards
- Node.js Runtime(back-end)
- Express Application Framework(back-end)
- React Web interface(front-end)
- Docker container
    - Reproducible development environment
- GitLab Continuous Integration
    - Gitlab runner
    - Automated unit testing
    - Regression testing

4.4.2 Out of Scope

ID	Item	Description
OS01	Creation of new physical security cards	Savage security outsources the creation of the permanent security cards.
OS02	Physical hardware to	Savage security provides the software for the security

	implement the system.	system but does not provide the hardware required for the network setup
OS03	2- factor authentication for the admin	Since our system runs in an isolated network, this level of security for authenticating admin is deemed unnecessary considering time and resource constraints.
OS04	Master physical keys for recoverability	Savage security outsources the physical keys and their management related to system.

4.5 Change Control

As this is a software project, change is anticipated. This will affect the project as it may add or remove work. Since we are using an Agile methodology we are more flexible and adaptive towards changes. However, this does not mean that changes shouldn't be maintained in a controlled manner.

Unmaintained changes may lead to scope creep which will cause delays and delay project milestones and maybe even deadline. Therefore, a formal change control workflow is designed for the project. The main rationale behind the change control procedure is that the changes cost more when they are incorporated at later stages of the project development process.
In this change control workflow;

- All but simple cosmetic changes will be made as a change request and follow the normal workflow
- Simple cosmetic changes will be integrated directly, they will be assigned to a person that was responsible for the relevant part of the project during the sprint.
- Change requests will be analyzed by their effect on design, architecture and implementation. Such an analysis will take place during group meetings.
- Simpler changes that do not affect high-level design or architecture will be incorporated to project during meetings by changes required user stories or use cases or adding new ones. According to their re-estimated priorities and weights, they will be placed on relevant sprint and will be processed as usual stories.
- More complex changes that require architecture and/or high-level design of the system will be further analyzed and re-estimated. Change impact points will be located and all the relevant parts of the system and required work to reach the desired state will be determined.

- Complex changes will be given priority on their implementation as further reworking the whole project might cause a significant delay in the schedule and demotivate the team.
- There is an option of deferring complex changes and provide them as a new project proposal for further direction, however, this option is only available if the change request does not impact the main functional requirements in a major way.

4.6 Constraints and Assumptions

Assumptions

- All users will be assumed to have only one card.
- All Arduinos will be assumed to run on an isolated network.
- All user will be assumed that they will be connected to the internet so that they can have access to the webpage
- Arduinos will be assumed that it can perform basic functions such as opening and closing a door.
- Reliable network conditions are assumed
- Only one person will enter per door entry request (this includes the door opening and closing)
- Only one person will leave the room at a time
- No power surges are assumed

Constraints

- Time constraints as we do not have unlimited time to work on the project.
- Personal and other commitments adding time constraints for each people also limited opportunities for meeting
- A limited number of people present for the project.
- The limited expertise of the team members regarding chosen software architecture and technology stack
- Arduino tasks may break other tasks.
- Implementation of a web interface for a drill-down style for displaying rooms in a physical site location is constrained because it is dependent on the infrastructure of navigating this implementation style to be created.
- Limited budget
- Constrained by physical hardware to read NFC cards. This has shaped our approach to designing a security door system. Furthermore, the Arduino that we have has limited capabilities for IoT libraries such as Azure IoT, this has caused us to take different approaches towards our communication layer in the software.

- When no network is present, the door will remain closed.

4.7 Cost Estimates

4.7.1 Overall Estimate

The project ‘Savage’ consists of two releases (R1 & R2). Each release contains three Sprints (S0 - S2). There are 12 weeks allocated to this project. For this project as well. Assuming that our rate of pay would be $80 per hour.

Also, $300 has been set to purchase units necessary for this project. The list includes but is not limited to:

- Arduino UNO
- Buzzer
- WiFi shield
- Motor
- Motor controller
- Red & Green LEDs
- Male-to-Female wires
- Female-to-Female wires
- Resistors
- Breadboard
- 9v Battery Connector
- Adafruit NFC/RFID reader
- USB Type-A to Type-B connector

Total cost of project (R1 - R2)

= Team members (6) * Hours per week (12) * Number of weeks (12) * Rate per member ($80) + Arduino board and other items ($300)

= $69,420

4.7.2 Release 1 Estimate

Release 1 Cost

= Team members (6) * Hours per week (12) * Number of weeks (6) * Rate per

member ($80)

= $34,560

4.7.3 Sprint 1 Estimate

Sprint 1 Cost

= Team members (6) * Hours per week (12) * Number of weeks (2) * Rate per member ($80)

= $11,520

4.8 Time Estimation

4.8.1 Time Sheet | Jingyi Yang

Day	Week 1 (06/08)	Week 2 (13/08)	Week 3 (20/08)	Week 4 (27/08)	Week 5 (03/09)		..	....	Total
Monday	0	0	0	0	4		0	0	4.00
Tuesday	3	4.5	3.5	3.5	3.5				18.00
Wednesday	0	1	0	1	0				2.00
Thursday	0	1	3	5	0				9.00
Friday	0	0	2	1	0				3.00
Saturday	0	0	0	5	0				5.00
Sunday	0	0	0	8	0				8.00
Total	3.00	6.50	8.50	23.50	7.50				**49.00**
Rate	$80	$ 80.00	$80	$ 80.00	$80	$ 80.00			
Total	$ 240.00	$ 520.00	$ 680.00	$ 1,880.00	$ 600.00				**$ 3,920.00**

4.8.2 Time Sheet | Phone Myat Naing

Day	Week 1 (06/08)	Week 2 (13/08)	Week 3 (20/08)	Week 4 (27/08)	Week 5 (03/09)	…	..	....	Total
Monday	0	4	4	2	4	0	0	0	10.00
Tuesday	3	3	3	3	0				12.00
Wednesday									
Thursday	0	2	2	4	0				8.00
Friday									
Saturday	0	3	3	3	0				9.00
Sunday									
Total	3.00	12.00	12.00	12.00					**39.00**

Rate	$80	$ 80.00	$80	$ 80.00	$80	$ 80.00			
Total	$ 240.00	$ 960.00	$ 960.00	$ 960.00					**$ 3,120.00**

4.8.3 Time Sheet | Senghuot Lay

Day	Week 1 (06/08)	Week 2 (13/08)	Week 3 (20/08)	Week 4 (27/08)	Week 5 (03/09)		..	….	Total
Monday								0	
Tuesday	3	3	3	3	0				12.00
Wednesday				8	0				8.00
Thursday			2	3	0				5.00
Friday	3		3		0				6.00
Saturday			5		0				5.00
Sunday			3	12					15.00
Total	6.00	3.00	16.00	26.00					**51.00**
Rate	$80	$ 80.00	$80	$ 80.00	$80	$ 80.00			
Total	$ 480.00	$ 240.00	$ 1,280.00	$ 2,080.00					**$ 4,080.00**

4.8.4 Time Sheet | Miriam Tym

Day	Week 1 (06/08)	Week 2 (13/08)	Week 3 (20/08)	Week 4 (27/08)	Week 5 (03/09)	…	..	….	Total
Monday	0	0	0	0					
Tuesday	3	3	3	3					12.00
Wednesday	0	0	0	0					
Thursday	0	0	2	2					4.00
Friday	0	0	0	0					
Saturday	0	0	3	4					7.00
Sunday	0	0	4	7					11.00
Total	3.00	3.00	12.00	16.00					**34.00**
Rate	$80	$ 80.00	$80	$ 80.00	$80	$ 80.00			

Total	$ 240.00	$ 240.00	$ 960.00	$ 1,280.00					**$ 2,720.00**

4.8.5 Time Sheet | Daniel Ebrahimian

Day	Week 1 (Date)	Week 2 (13/8)	Week 3 (20/8)	Week 4 (27/8)	Week 5 (3/8)	…	..	….	Total
Monday	0	0	0	3	2	0	0	0	5.00
Tuesday	0	3	5	7	7				22.00
Wednesday	0	0	6	8					14.00
Thursday	0	4	5	4					13.00
Friday	3	0	0	0					3.00
Saturday	0	5	0	3					8.00
Sunday	0	0	2	10					12.00
Total	3.00	12.00	18.00	35.00	9.00				**77.00**
Rate	$80	$ 80.00	$80	$ 80.00	$80	$ 80.00			
Total	$ 240.00	$ 960.00	$ 1,440.00	$ 2,800.00	$ 720.00				**$ 6,160.00**

4.8.6 Time Sheet | Cemal Ulas Kundakci

Day	Week 1 (06/08)	Week 2 (13/08)	Week 3 (20/08)	Week 4 (27/08)	Week 5 (03/09)		..	….	Total
Monday	0	0	0	3	0	0	0	0	3.00
Tuesday	3	3	4	5	0				15.00
Wednesday	0	0	2	4	0				6.00
Thursday	0	2	4	5					11.00
Friday	0	2	0	2					4.00
Saturday	0	5	0	3					8.00
Sunday	0	0	2	4					6.00
Total	3.00	12.00	12.00	26.00					**53**
Rate	$80	$ 80.00	$80	$ 80.00	$80	$ 80.00			
Total	$ 240.00	$ 960.00	$ 960.00	$ 2,080.00					**$ 4240.00**

5. Agile Architecture

5.1 Data architecture

Our system is required to unlock a door based on an approved card-authorization request. Based on this requirement we have created several entities. Our primary desire was to support a lookup that was based on information of knowing the card and room. We created an intermediary for this lookup named Access_Request that communicates with another entity, Access_Manager. The role of the later entity is to handle deciding whether a card is authorized to a room.

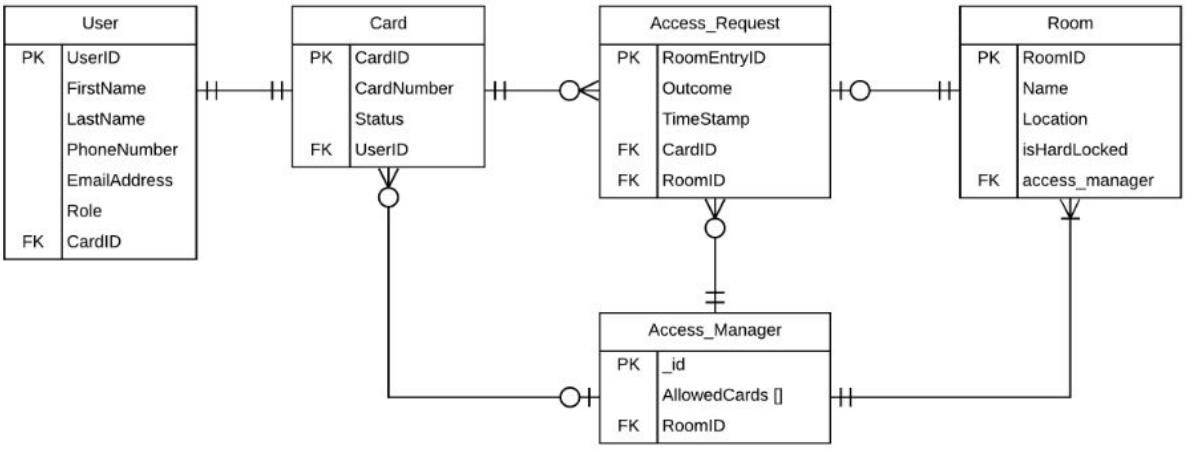


Figure 5.1.1 ERD Diagram

This ER diagram illustrates the five main entities:

- User
- Card
- Access_Request
- Access_Manager
- Room

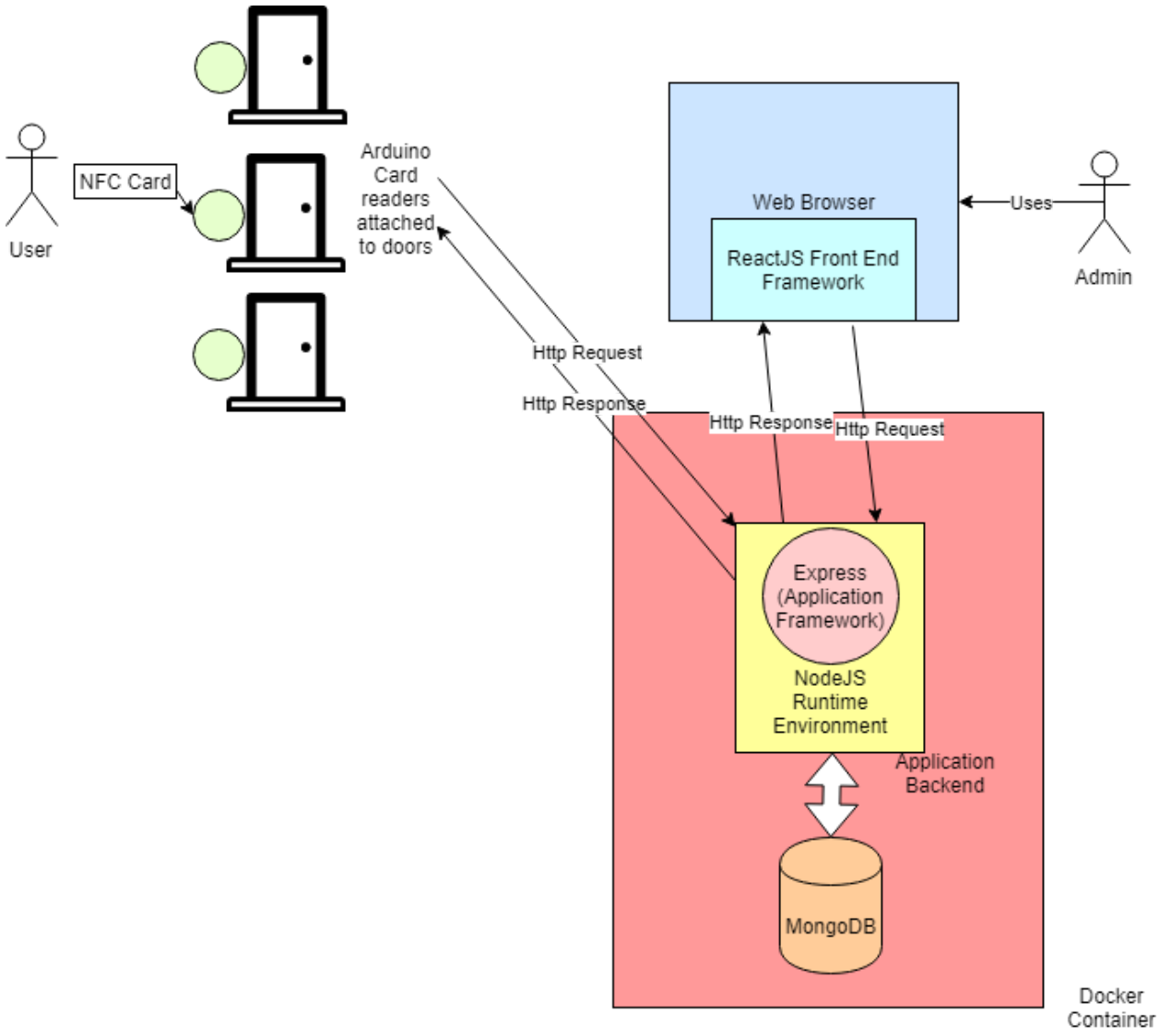
The entity-relationship diagram used in the Savage Security Application aims to reflect the real life process of creating a robust security system of access. This diagram above (Figure 5.1.1) illustrates the relationship between each entity in this platform. Firstly, the User entity includes attributes of UserID, FirstName, LastName, PhoneNumber, EmailAddress, Roles and a foreign key of CardID. In this entity, User ID is the unique identification of User entity as it allows query search to a unique user for its attributes.

Role attribute allows the system to determine whether the user is an Admin or Non-Admin User. Relationship between the User and Card Entity, demonstrates that the User will have one and only one card for Room Access Request. When a User uses the card for the door access, the Access_Request entities sends request for both Card ID and Room ID attributes. Subsequently, the Access_Request sends the Card ID to the Access_Manager entity to authenticates whether the Card ID have access to the Room ID. Therefore, it would provide a reject or allows access to the room.

In this diagram (Figure 5.1.1), we decided to input an Allowed Card as an array attribute because MongoDB is an no sequel database structure that uses a dynamic schema for unstructured data that allows us to store arrays or any database format.

5.2 Application and Infrastructure Architecture

Savage security application uses IoT and Responsive Web Application architecture. The system consists of an NFC reader, arduinos and a backend server with database and Admin web interface for viewing data and managing the system. Architecture diagram is given below



Arduino with NFC reader is used for checking user access to rooms. A user should tap their cards to access rooms and upon tapping, Arduino will read their card. Arduino then connects to our backend server with their card number and server replies with a message instructing arduino to whether unlock the door, activate a green LED and beep or activate a red LED and beep.

The server is running NodeJS runtime environment and Express Application Framework. NodeJS runtime provides an event-driven runtime to provide the server with the ability to run Javascript code for the backend. Express Application Framework provides a set of features for web applications such as routing, database integration and error handling.

Server stores all the arduino messages and their status in a database. MongoDB is used as a database. Express handles MongoDB integration and MongoDB is stored locally within the same server as NodeJS and Express.

Docker is used as a container for backend server. Containerization makes the server portable that can be ported to a different server provider easily. Furthermore, using a container helps with dependency management and automated deployment and Continuous Integration. GitLab is used for this project as version control and CI.

Admin view of the system is handled by React. React is a front-end Javascript framework that is used to render views for user interface. React allow us to make dynamic web pages for Admin views and it displays data dynamically retrieved from the backend server. Bootstrap is used to make web pages responsive. By using responsive web pages for the interface we make it platform independent as it can run on any device with web browsing capabilities.

As for front-end alternatives, we have considered VueJS and Angular for React. Due to React's current popularity, availability of documentation and manageable learning curve, Angular and Vue were not selected. For our backend, ASP.NET Core and Python Django was considered. However, in order to simplify the programming languages needed, NodeJS selected as a best candidate because it uses Javascript as well. Also, NodeJS provides JavaScript Runtime so our React code can be executed on the server which can be used for Server Side Rendering if needed. As for IoT, Raspberry Pi was considered as an alternative to Arduino. However, Raspberry Pi is not as power efficient as Arduino, so for implementation that would be running continuously, power savings is an important factor, therefore, Arduino is selected.

5.3 Architecture Spike (Software Prototype)

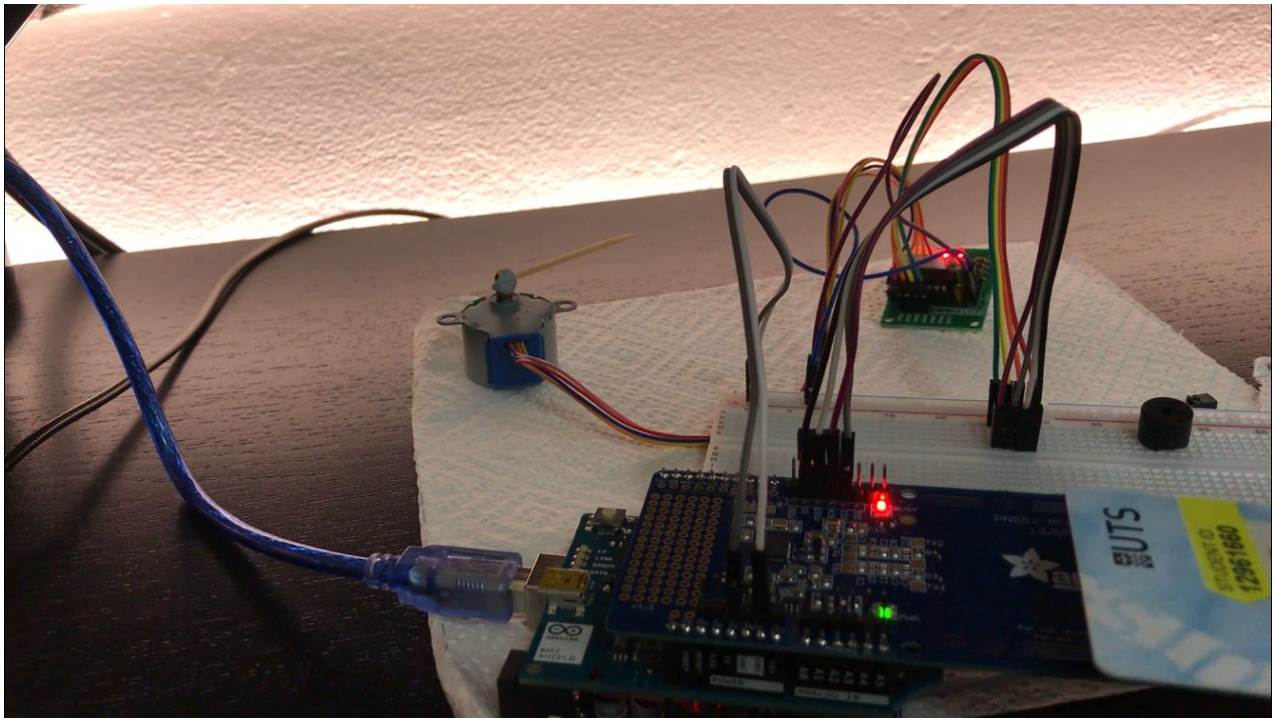
5.3.1 Arduino prototype

The system we are creating is using an Arduino to unlock a door. Our goal is to have the door-open request go through the Internet and communicate with a server for authentication. Many technical unknowns are existent in this goal and we chose to investigate the feasibility of this process with a prototype before we continue with this idea.

Creating the prototype began setting up a controlled environment. This was a network that ran on its own VLAN. The Arduino was connected to the wireless network with

WPA-2 on the 2.4ghz band via a WiFi Shield that was physically connected to the Arduino. A laptop was wirelessly connected to the same network that hosted a web server. This web server was created rapidly with the Flask web framework in Python. It simply served a webpage that would approve and reject incoming requests based on an ID communicated to the web server. The UID of the tapped card to the Arduino was extracted and summarised and then sent to a lightweight web server. The Arduino communicated with the server via a web request. All communications occurred locally on the isolated network.

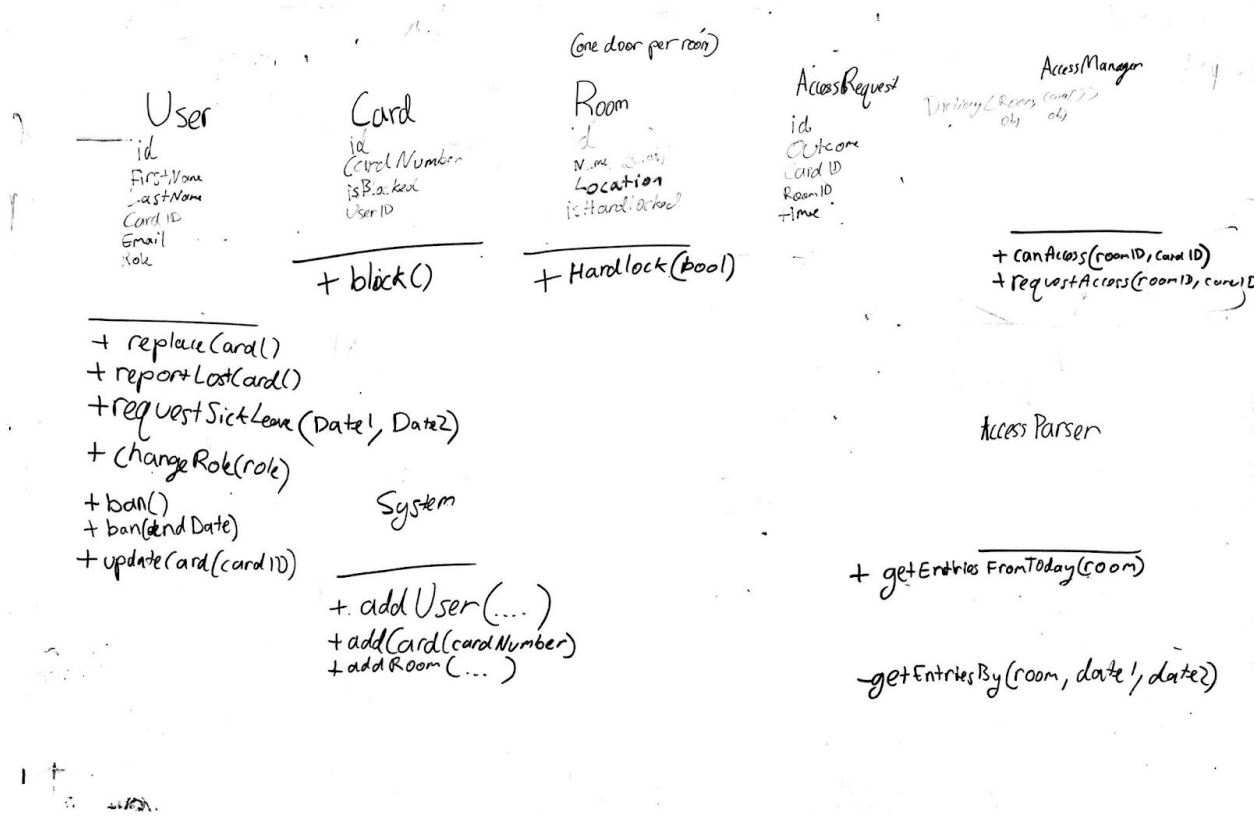
The Arduino would react to the response of the server. On success, a motor would rotate. This motor was physically wired to the Arduino. Below is an image of the prototyped setup.



The outcome of this architecture spike revealed many technical issues and system infrastructure that required modification and redesign. Originally, we planned to utilize the MQTT protocol to communicate with an IoT hub hosted on Microsoft Azure. However, by completing the prototyping of the platform, we found this to be incompatible with our Arduino. We therefore, changed this technology to a web server and web requests as the communication layer. We also gained insight and knowledge of Arduinos which was an unexplored technology for all our team members. We are now able to confidently move forward with our development knowing that our system can work end-to-end at a fundamental level.

5.3.2 Entity interaction

A separate meeting was held between the Technical Leader and Scrum Master to determine the basic data and software architecture that reflects the data requirements. While the intent of the meeting was to clarify data requirements and data architecture. It was decided that the software architecture and design needed to be explored further to refine the data architecture. In order to create a harmonious data and software architecture, we decided to create an informal class diagram related to database access in general. While discussing our basic entities such as room, card, user, door and access request we realized that the Access Management should be handled by a separate class which would be better represented as its own entity. We therefore altered the Data architecture further to reflect our Software Design. Generated diagram is given below.



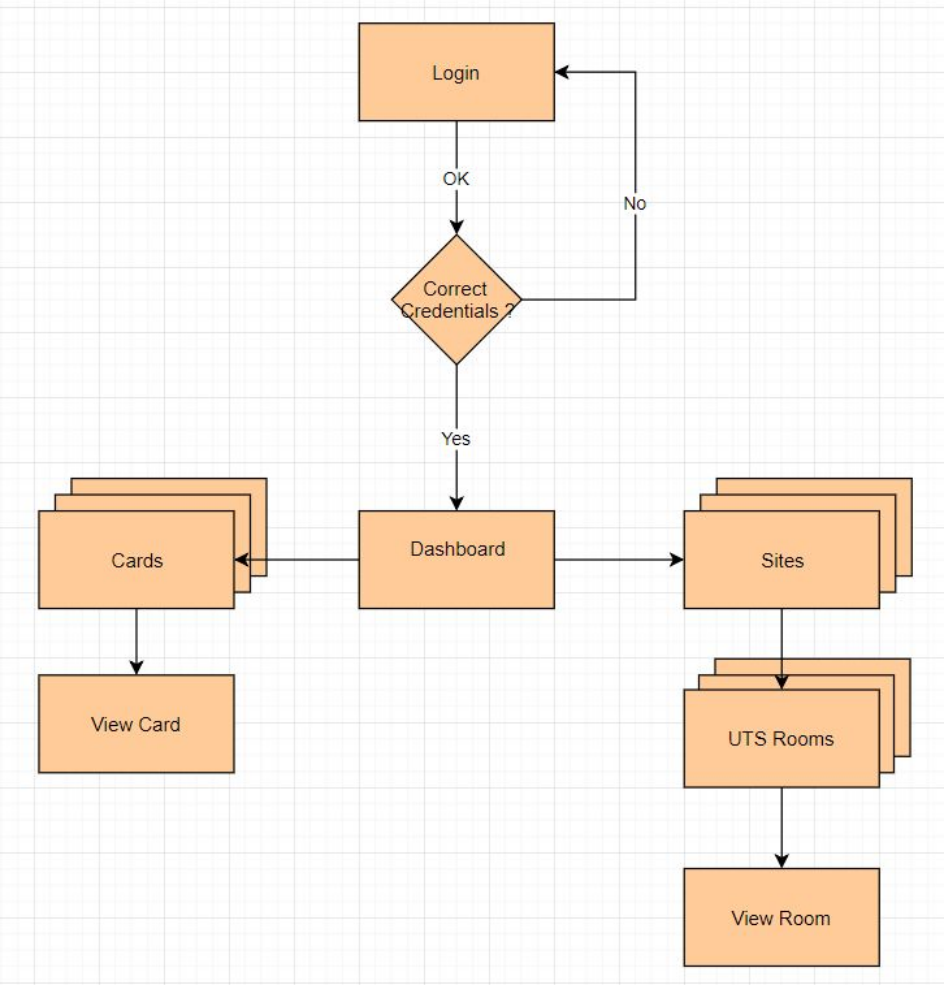
This is a diagram using a notation similar to UML Class Diagram. It reflects our model and functional entities. The diagram and discussions leading to its final design helped us gain insights into our data architecture.

The knowledge we gained from this meeting pathed the way of our ERD diagram and the way we approached solving the implementation hurdles of creating this system.

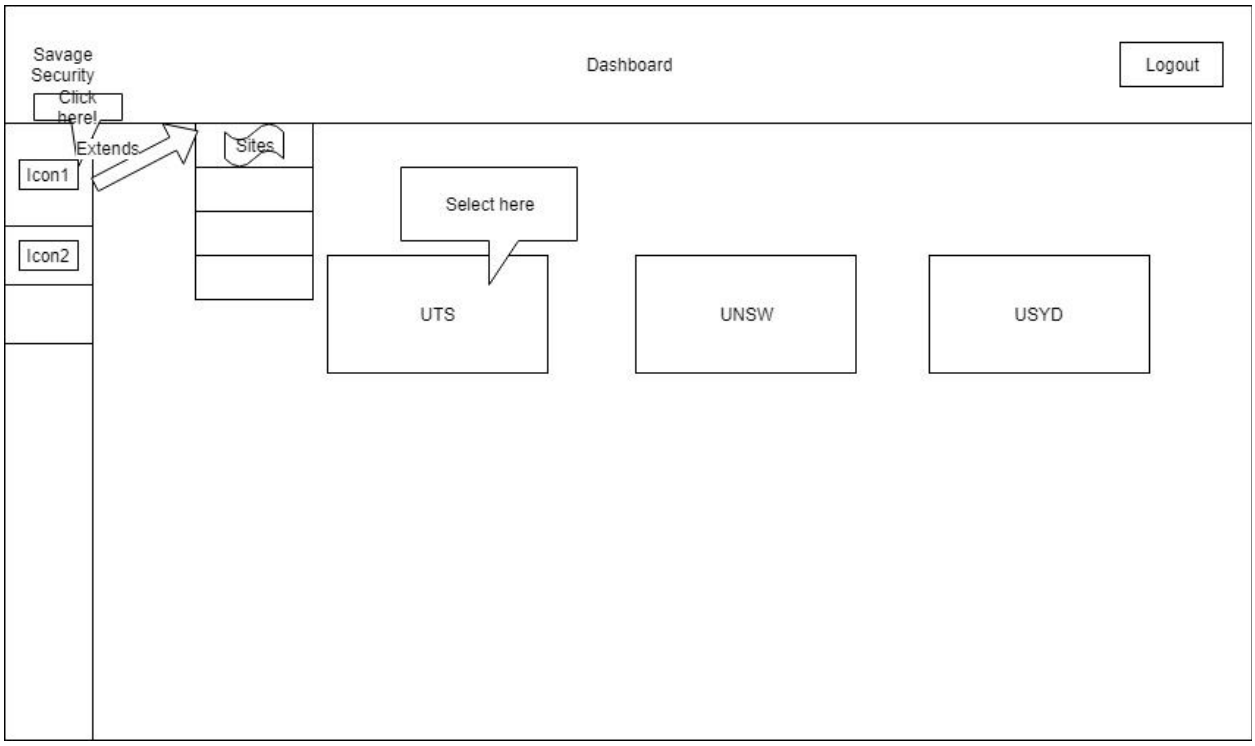
5.3.3 Web Interface

During our team meeting discussions, we brainstormed potential interface styles that supports the vision of our product. We were heavily in favour of creating a drill-down style interface that supports clicking on cards and into their contents. From this concept, we created a design for an interface to have a left-orientated navigation bar that had two primary buttons, one for dashboard and another for sites. In our context, sites refers to locations, for example UTS, UNSW or USYD, these are all physically different sites. By clicking on the sites button, the interafare shall show the current locations available for this application. Furthermore, clicking on a location would then should the available rooms in that location.

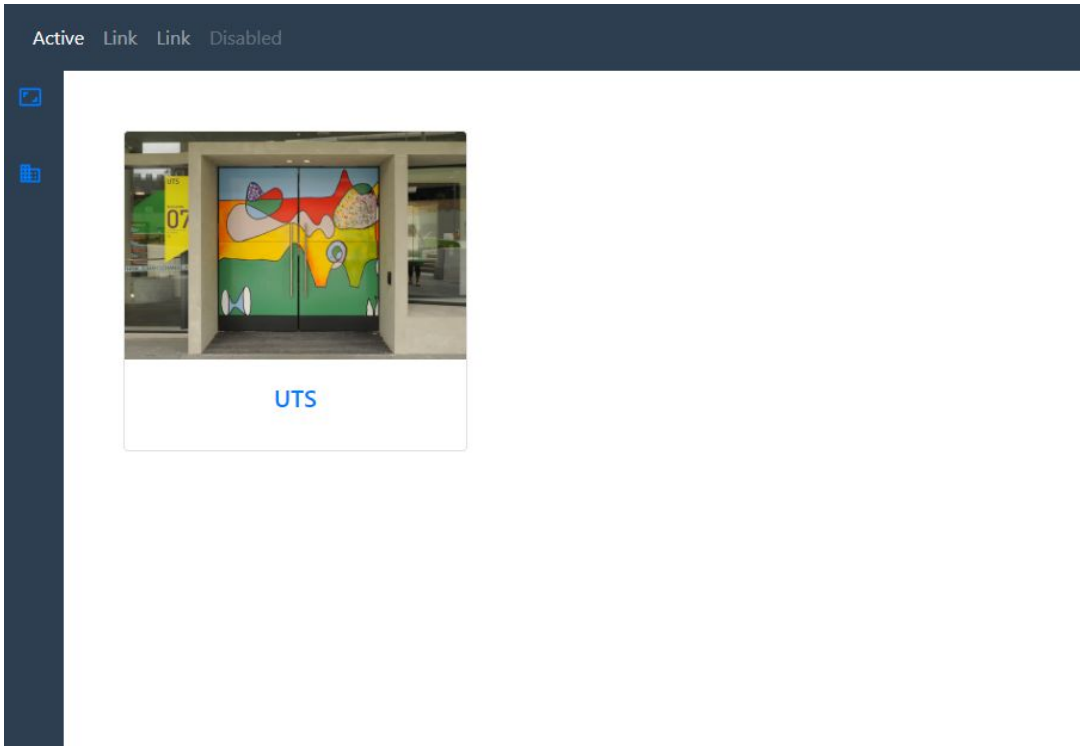
Flow diagram of the application for the first iteration is given below. It describes the flow of activities for the web interface.



Our drill-down design flows by navigating from Sites → Location → Room
The following Lo-Fi diagram is our prototype.



The following interface is our implemented prototype.



6. Appendices: Individual Contribution Logbooks

6.1 Daniel Ebrahimian - 12961660

1. 10/8
    a. Meet some team members are catch up with Software Engineering Practice subject (I joined the class late due to enrolment issues)
2. Date 14/8:
    a. Investigate potential hardware to be used for detecting and reading NFC cards
3. Date 16/8:
    a. Get arduino hardware and put them together
    b. Setup software to react to NFC cards and move a motor
4. Date 18/8:
    a. Investigate IoT hubs and brokers such as MQTT and Microsoft Azure
    b. Setup wifi module on arduino
5. Date 21/8: User stories, interface drafts, roles assigned, meetings organised, team charter started, organise trello usage. How we will plan our sprints and backlog.

6. Date 22/8:
    a. Conclude that MQTT will not work with the current hardware configuration. Investigate alternatives and end up utilizing web requests.
    b. Implement web requests as communication layer on the arduino.
    c. Setup test web server to handle authentication requests coming from the arduino
7. Date 23/8:
    a. Check all user stories and create functional and non-functional requirements.
    b. Work on project scope of the assignment. In and out of scope.
    c. Create continuous integration pipeline
    d. Automate unit testing in gitlab
    e. Have a technical meeting to discuss the pros and cons of no longer being able to use MQTT and the viability of the new solution. Agree to move forward with the proposed web request communication layer.
8. Date 26/8:
    a. Discuss entities for the ERD with group member
9. Date 27/8:
    a. Create trello sprint board and populate with tasks, deadlines, organized and colour coded
10. Date 28/8:
    a. Group meeting
    b. Investigate the architecture of the database
    c. Investigate class structure and communication between modules
    d. Theoretically prototype technicalities for processing data to the database
11. Date 29/8:
    a. Worked on Docker images
    b. Worked on the web platform with Node.JS, react, reactrouter
12. Date 30/8:
    a. Work on Trello board
    b. Allocate:
        i. Story points
        ii. Priorities
        iii. Create new columns for ‘Ready for testing'
    c. Write architecture spike
    d. Contribute to the flow control diagram
13. Date 1/9:
    a. Group meeting and assignment work
14. Date 2/9:

    a. Group meeting and assignment work
15. Date 3/9
    a. Group meeting and assignment work

6.2 Miriam Tym - 12577692

1. Date 7/8: This session was spent creating the group and watching videos showed to us by the tutor.
2. Date: 14/8: In class discussion regarding the assignment system choice. The end result was on a tag sensing security system. We created a team charter about our project idea and sent it to our tutor for the green light to start working on this idea.
3. Date 21/8: Roles were assigned during this meeting and my role is the scribe. As such I started taking notes. During this session, we started creating the use case stories. We were also given parts of the assignment to complete mine were as follows:
    a. Help with use case stories.
    b. Help do the Functional and Non-Functional requirements.
    c. The introduction.
    d. Team Charter.
    e. Help with the project scope.
    f. The sprint backlog.
    g. Read through the assignment and check for spelling errors.
    h. Create most of the In-Scope and Out-of-Scope items.
    i. I updated my timesheet and logbook throughout.
4. Date 23/8: During this session, the whole group worked on the Functional and Non-Functional requirements. This was done by grabbing the functions from the use case stories created last week. The goal was to have 45 constructed before the meeting, however, we only reached 30. This left us with 13 functional requirements and 5 non-functional.
5. Date 25/8: In this session, I started working on the team charter. I was able to partially complete, Team member knowledge and skills and,Member roles.
6. Date 28/8: During this meeting, I finished each of the member's roles and team member knowledge section. I also started working on the member rules and consequences.
7. Date 30/8: During this session, I started on the team purpose, desired end result and sprint backlog.
8. Date 1/9: Started and completed the In-Scope and Out-of-Scope items. I completed everything I was tasked with doing on this day. I then went and read

through them to check for spelling errors and put them in the sprint catalogue for approval.

9. Date 2/9: At this stage, I spent time polishing off all the areas I was meant to complete, then spent time going through the assignment and checking for spelling errors or misworded statements.

6.3 Cemal Ulas Kundakci - 12479248

16. Date 7/8: Class meeting
17. Date 14/8: Class meeting
    a. Class meeting to decide on application architecture, IoT and web app chosen.
    b. Investigate potential hardware to be used for detecting and reading NFC cards
18. Date 16/8:
    a. Acquire arduino hardware and put them together
    b. Setup software to react to NFC cards and move a motor to see feasibility
19. Date 17/8:
    a. Research on IoT technology, learned about hubs and brokers
20. Date 18/8:
    a. Join Daniel to investigate IoT hubs and brokers such as MQTT and Microsoft Azure
    b. Help setup wifi module on arduino
21. Date 21/8: Class meeting.
    a. Made user stories,
    b. Preliminary interface drafts,
    c. Team charter started,roles assigned,
    d. Trello is decided as our sprint backlog and board is organized
22. Date 22/8:
    a. Decided to use HTTP instead of MQTT due to our current hardware not supporting existing IoT hub libraries.
    b. Helped simple server implementation to handle requests and reply with an answer
23. Date 23/8: Group meeting
    a. Refine the user stories
    b. Create functional and non-functional requirements.
    c. Work on project scope of the assignment. In and out of scope.
    d. Setting up docker container and creating continuous integration pipeline by automating unit testing on github

    e. Final decision on using web requests instead of MQTT
24. Date 26/8:
    a. Help organize the trello board and take role in managing the board itself.
25. Date 27/8:
    a. Checking NodeJS and React tutorials.
26. Date 28/8: Group meeting
    a. Discuss the software architecture
    b. Discuss about data requirements and communication between system layers
    c. Create UML like diagram for describing functional and model classes
27. Date 29/8:
    a. Wrote a software architecture part
28. Date 30/8:
    a. Managing trello board
    b. Creating the flow control diagram
29. Date 1/9:
    a. Assignment work
    b. wrote change management
30. Date 2/9:
    a. Finalizing the report
    b. Helping out with User Narratives
    c. Helping out with Data architecture and requirements.

6.4 Jingyi Yang - 99181859

1. Week 1 Tue.

Participated in preparation for Assignment and completed workshop activities.
Discussed potential idea, infrastructure, programming languages to use for the project.

2. Week 2 Tue.

Based on the previous discussion, we are choosing to make an IoT project (Savage security) which is an application uses IoT and Responsive Web Application architecture. The system consists of NFC reader arduinos, backend server with database and Admin web interface for viewing data and managing the system. We decide on our software stack (C++, Node.JS, react, reactrouter and such)

3. Week 2 Wed.

Start to working on the Static test sheet

4. Week 2 Thur.

Work on the Trello board. Created, Allocated cards

5. Week 3 Tue.

User stories, interface drafts, roles assigned, meeting organised, team charter started. organise trello usage. How we will plan our sprints and backlog.

6. Week 3 Thu.

Check all user stories and create functional and non-functional requirements. The Team leader has confirmed that MQTT will not work with the current hardware configuration. To learn how to Implement web requests as communication layer on the arduino.

7. Week 3 Fir.

Working Manual testing sheet and Start to do a Static test.

8. Week 4 Tue.

Create a template for the project, it has font size, colour, style, layout and such.

   a. Project Logo with team
   b. Create timesheet for the project. Report the deflect form start test.
   c. Play around JS Func., JS & JSON.

9. Week 4 Wed.

Play around React js, ES6.

10. Week 4 Thu.

Play around NPM and Up & Running with NPM the Node Package Manager.

11. Week 4 Fir.

Play around Bootstrap 3

12. Week 4 Sat.

   a. Getting ready for Assignment 1 deliverables.
   b. Update Timesheet, create Trello card for and allocated tasks
   c. Removes cards from it due to some cards has been allocated in wrong places.
   d. Changes subheading, formatting the report.

13. Week 4 Sun.

   a. Making changes on Time sheet for team & cost estimates.
   b. Making changes on Card Wall, Overarching schedule, Sprint 0 Scheduling.
   c. Reformatting the report

14. Week 5 Tue.

a. Did SparkPlus and Self-Assignment sheet.

6.5 Phone Myat Naing - 12415562

1. Date 21/8: I was given role as scribe and start to take notes
2. Date 23/8: Worked with everyone during the meeting to finish off functional and non-functional requirements as well as completing use case stories.
3. Date 28/8: Completed use case diagram and worked on assumptions and constraints. Also worked on prototype drawing of lo-fi wireframes.
4. Date 30/8: Worked with everyone on setting up priorities and story points on sprint backlog on trello. As well as removing and adding more sprints.
5. Date 01/9: Worked on lo-fi wireframes and completed it.
6. Date 02/9: Finalized document for release
7. Date 04/9: Finished presentation smoothly. Proceed to make minor changes and discuss next step for assignment 2.

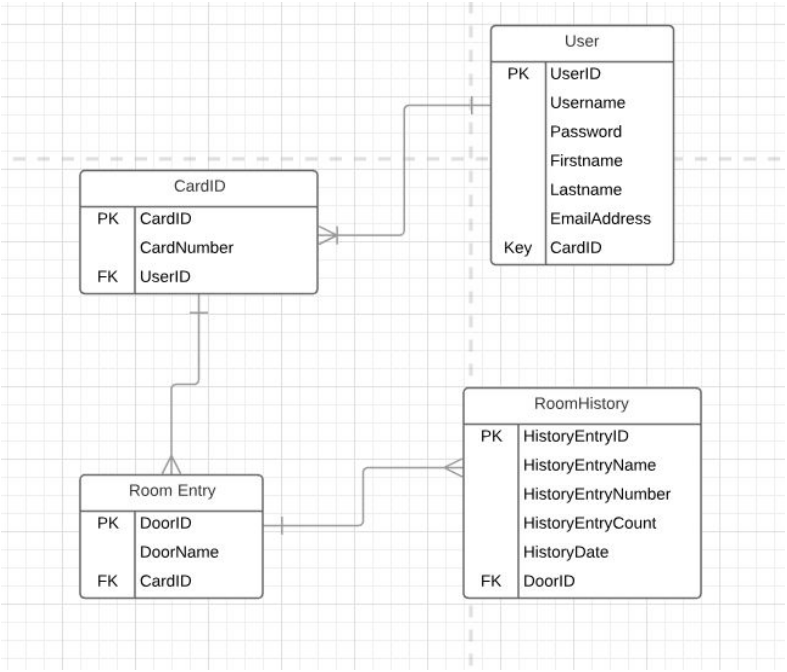
6.6 Senghuot Lay - 12632776

1. Date 07/08: Class Meeting
2. Date 09/08: Team meeting about the project in online booking system where the user is able to make an online appointment for hairdressing company
3. Date 14/08: Class Meeting
4. Date 21/08: Class Meeting
5. Date 23/08: Team meeting to in the confirmation of user story, and added basic node js router that allows the site to run using express js
6. Date 24/08: Complete FMEA risk analysis
7. Date 25/08: Complete basic ERD Diagram
8. Date 28/08: Class Meeting
9. Date 29/08: Work with Daniel in the implementation of the basic interface using React as the front-end and node js as the back end
10. Date 30/08: Team meeting and preparation for teaching the team about the actual code
11. Date 02/08: Drawn the complete version of ERD Diagram, Complete Data Dictionary, Make changes to the User Story Mapping, Make minor changes to the FMEA Risk.

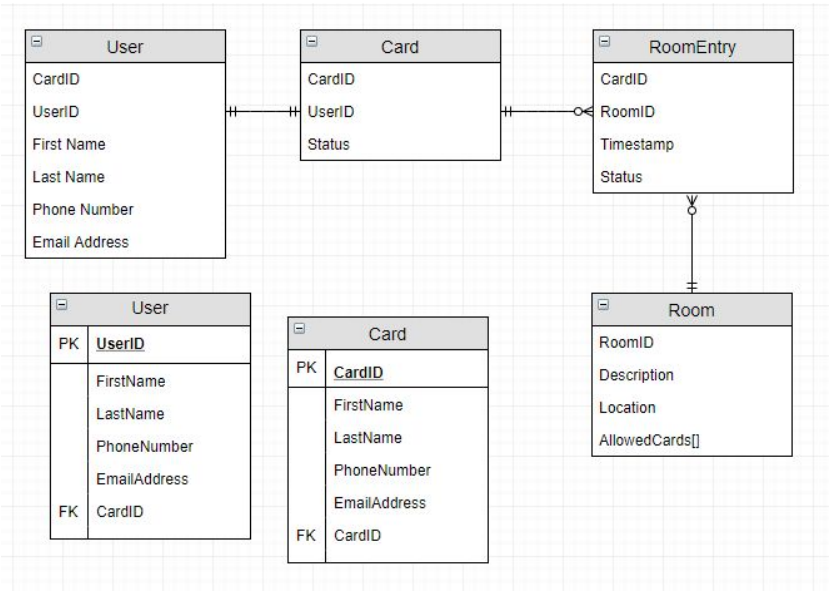
7. Appendix

7.1 ERD Diagram

7.1.1 ERD Diagram Version 1



7.1.2 ERD Diagram Version 2



7.1.3 ERD Diagram Version 3