TEAM NAME: Ominous Corp.
TEAM NUMBER: 201-4
MEMBERS:
Jake Martin
Ryan Marizza
Angelo Vacca
Mauro Nannini
Chirag Telang

PROJECT MILESTONE 4

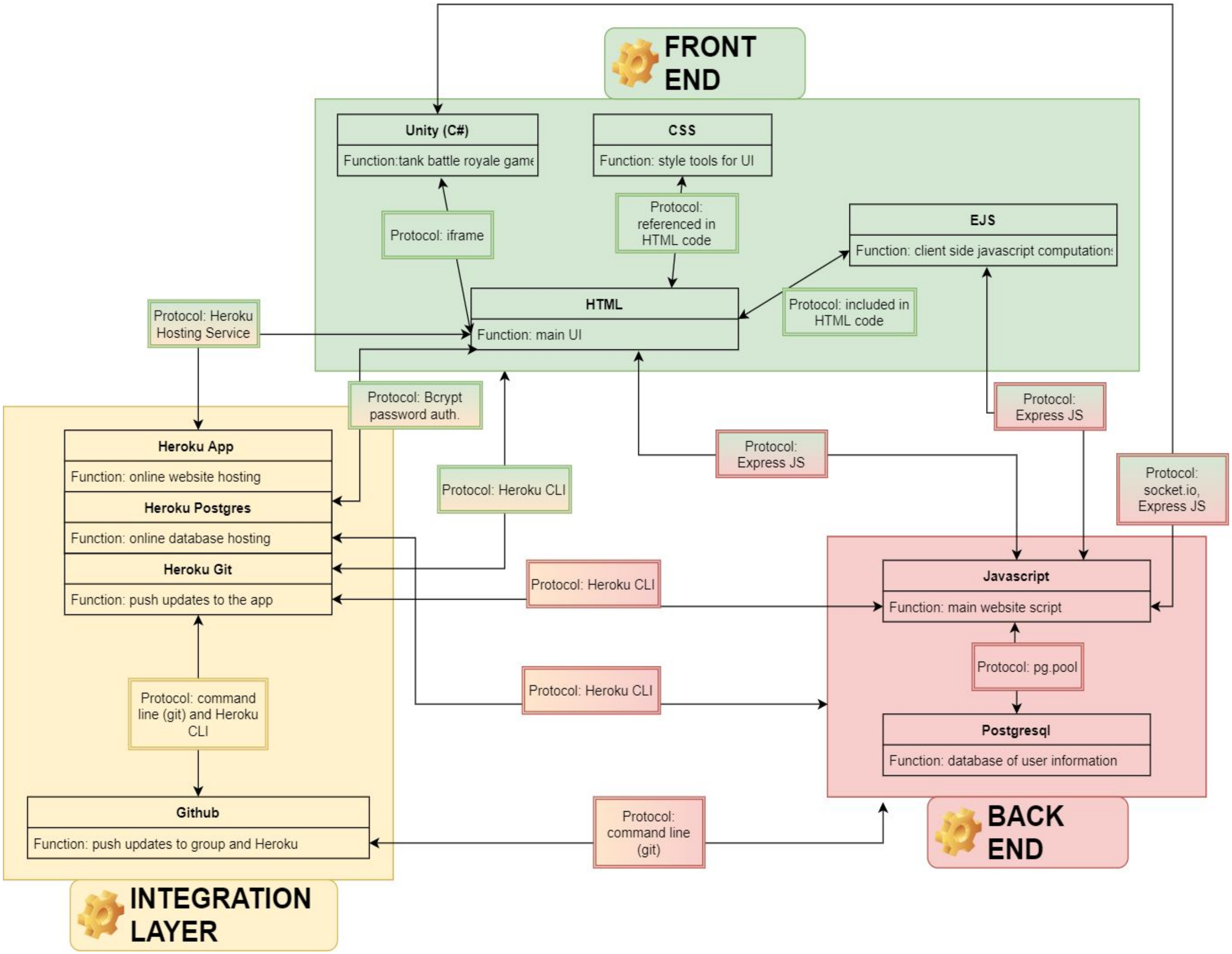
++

Revised List of Features:

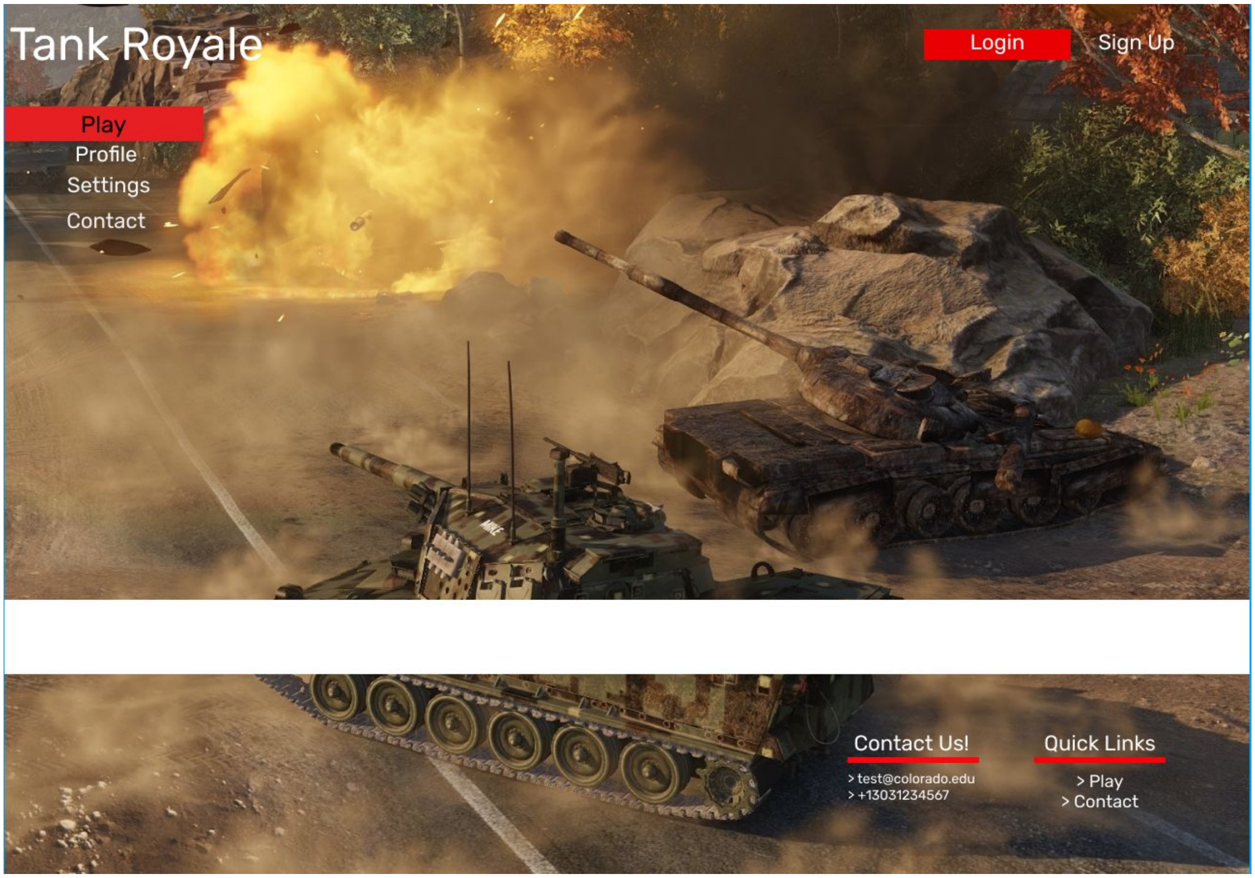
- Projecting the game on the website created for it (high)
- Changes to game features
  - Want to add features for powerups and multiple weapons (low)
  - Want to add features for looting other players (low)
  - Main menu navigation for players (low)
- Reduce the amount of tabs embedded in the home page (medium)

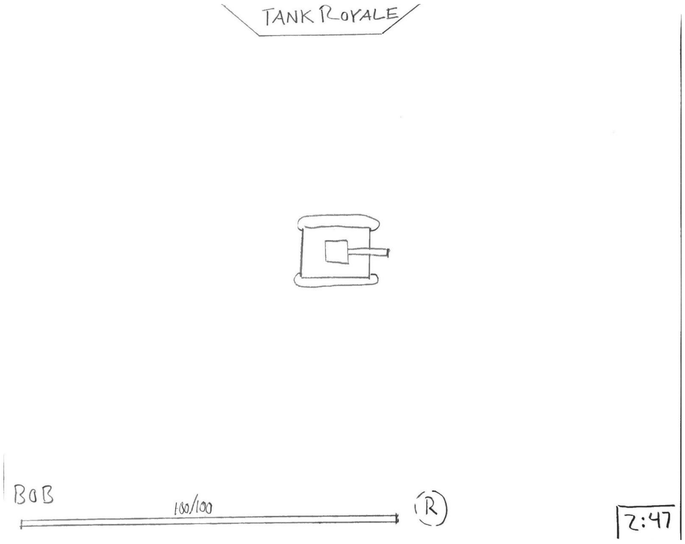
Architecture Diagram

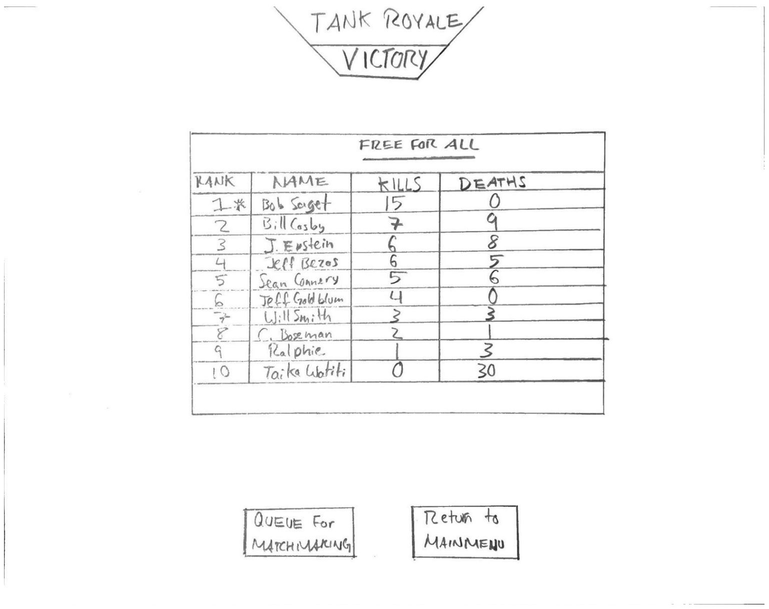
Each major component of the planned software architecture for Tank Royale is included in the following diagram. Also, the communication between software components is described via communication protocols each labeled "Protocol" in the diagram. Each protocol names the main tool(s) that allow the software components to communicate.


FRONT END
Unity (C#)
Function:tank battle royale game
CSS
Function: style tools for UI
Protocol: iframe
Protocol: referenced in HTML code
EJS
Function: client side javascript computations
HTML
Function: main UI
Protocol: included in HTML code
Protocol: Heroku Hosting Service
Protocol: Bcrypt password auth.
Protocol: Heroku CLI
Protocol: Express JS
Protocol: Express JS
Protocol: socket.io, Express JS
Heroku App
Function: online website hosting
Heroku Postgres
Function: online database hosting
Heroku Git
Function: push updates to the app
Protocol: Heroku CLI
Protocol: Heroku CLI
Javascript
Function: main website script
Protocol: pg.pool
Postgresql
Function: database of user information
Protocol: command line (git) and Heroku CLI
Github
Function: push updates to group and Heroku
Protocol: command line (git)
BACK END
INTEGRATION LAYER

Front End Design:







FREE FOR ALL

RANK	NAME	KILLS	DEATHS
1 *	Bob Saget	15	0
2	Bill Cosby	7	9
3	J. Epstein	6	8
4	Jeff Bezos	6	5
5	Sean Connery	5	6
6	Jeff Goldblum	4	0
7	Will Smith	3	3
8	C. Boseman	2	1
9	Ralphie	1	3
10	Taika Watiti	0	30

Web Service Design:

- We aren't really using any web services APIs
- Our website hosting is on Heroku
- Our game is built with the Unity game engine

Database Design:

The database is in Postgresql hosted online using the free service, Heroku Postgres. In the database there is a "users" table which contains the information about users. Specifically the users table is of the form:

Id (serial) (PK)	Email (unique varchar)	name (varchar)	Encrypted Password (varchar)
1	test@gmail.com	testo	e5e9fa1ba31ecd1ae84f75caaa474f3a663f05f4
2	john@gmail.com	johntank	d1e8a70b5ccab1dc2f56bbf7e99f064a660c08e3
...	...	...	...

When a user registers an account, their password is encrypted using a Java library called Bcrypt.

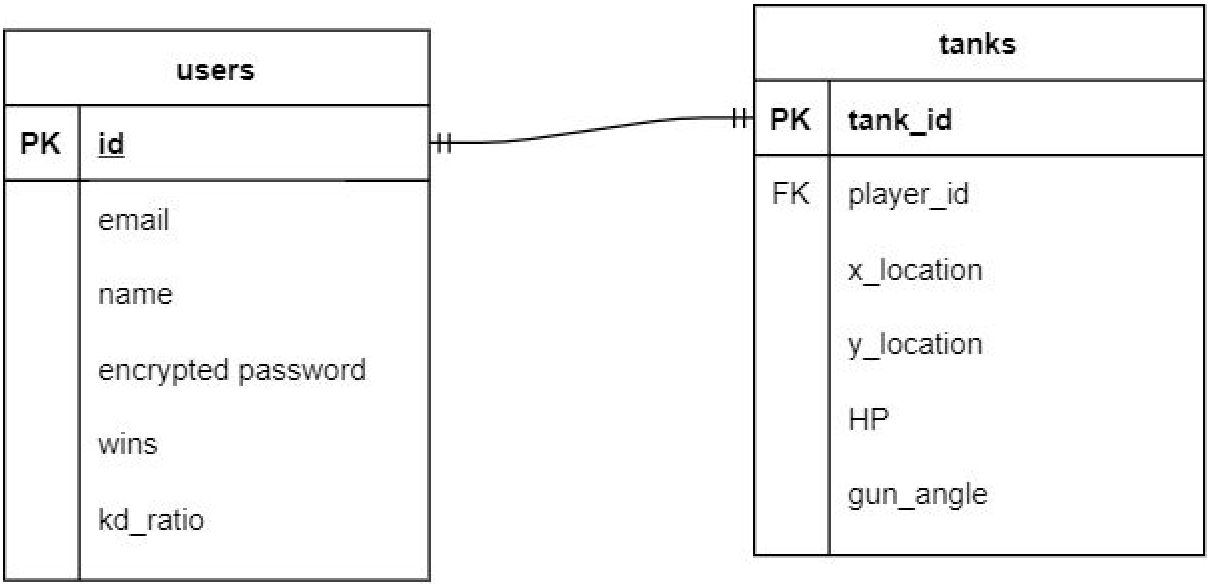
For now this is the only table in the database. We haven't started implementing the multiplayer server just yet, so there may indeed be another database storing things like player location, HP remaining, the angle of the tanks' guns, etc…
Suppose we do indeed need such a database to implement multiplayer functionality. Here is a rough outline of how it would look:

Tank_id (serial) (PK)	**Player_id (int) (FK)**	**X_location (int)**	**Y_location (int)**	**HP (int)**	**Gun_angle (int(360))**
1	5	132	8	95	275
2	1	145	23	65	33
...	...	...	...	...	...

Furthermore, given some inspiration and enough time, it would be nice if there were some statistics for a user's game history. Supposing this is possible, we may want to extend the "users" table to resemble something more like:

Id (serial) (PK)	**Email (unique varchar)**	**name (varchar)**	**Encrypted Password (varchar)**	**Wins (int)**	**Kills/Deaths ratio (float(5,3))**
1	test@gmail.com	testo	`e5e9fa1ba31ecd1ae84f75caaa474f3a663f05f4`	`55`	`13.551`
2	john@gmail.com	johntank	`d1e8a70b5ccab1dc2f56bbf7e99f064a660c08e3`	`2`	`01.001`
...	...	...	`...`	`...`	`...`

Here is an entity-relation diagram for the proposed database structure:



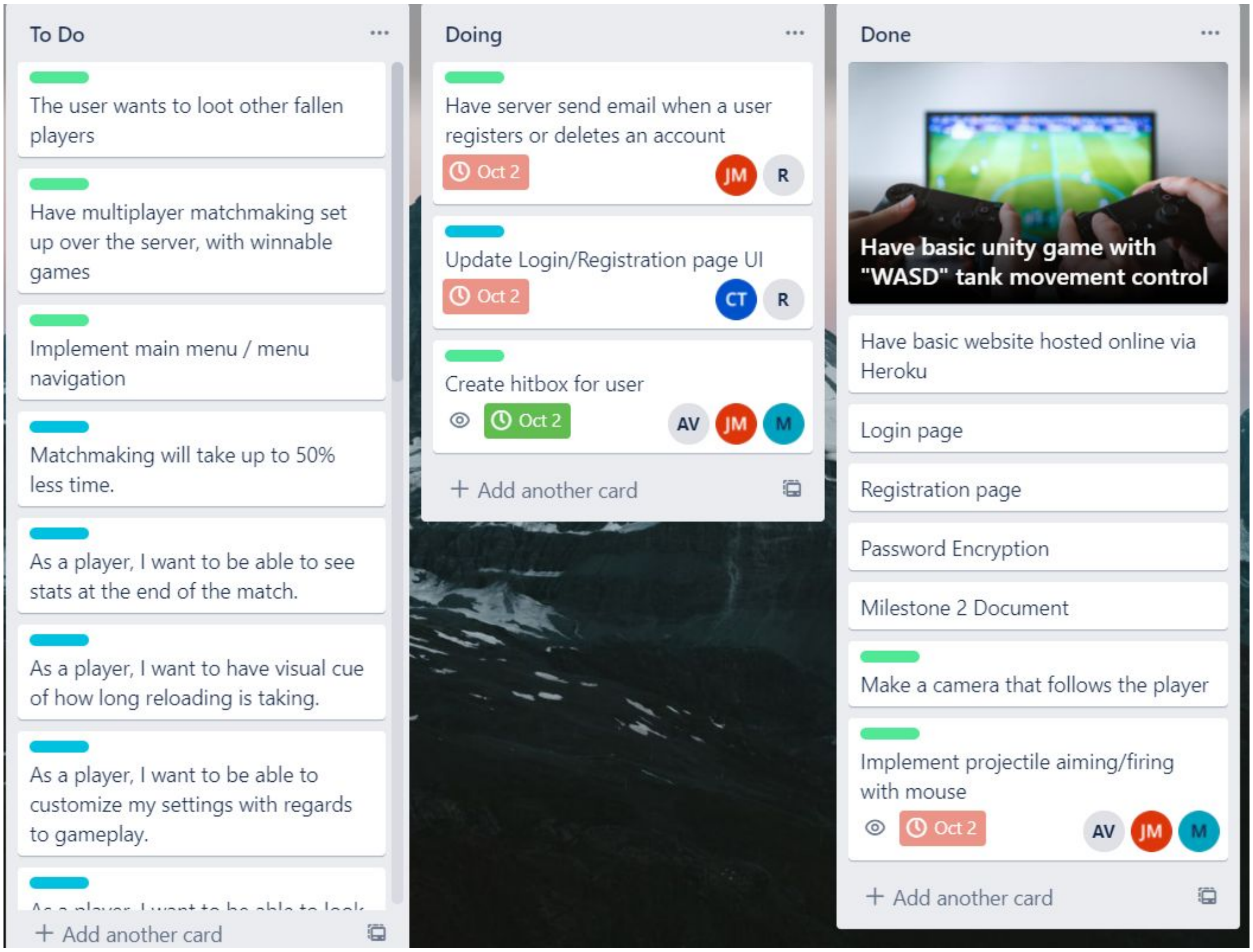
This is the (unfinished) idea for such a database structure: At the start of each match, a tank is created for a user with a new (unique) tank_id. The match goes on with Unity using the values in the tanks table to display interactive multiplayer opponents in real time. When the HP of any tank reaches 0 that tank is removed from the tanks table. When there is only one tank left, the game is over and that tank is removed from the tanks table. At any time there is only 1 tank per player and only 1 player per tank (because the tanks are created per game instance).

Individual Contributions:

- Jake Martin:
    - Working on the game prototype in Unity with Angelo. My main work has focused on the smooth movement of the tank, having it have momentum and “turn” so to speak.
    - Finding the free Unity-store assets that are the graphical backbone of the game.
    - Ryan and I have also been working to get the Unity game build and our heroku site to work together, which is still an in-progress issue.
    - For our Unity workflow, Angelo and I both use Unity collaborate, and then at the end of each session I push all of our changes to GitHub.
        - Latest commit (with Angelo): https://github.com/CSCI-3308-CU-Boulder/The-Ominous-Corp/commit/1f2618f2730c941feeb5f482fa876801f1006359
- Ryan Marizza

    - My main contributions have been to create the functional login/registration pages using Javascript and Postgresql .
    - Also, I hosted the app website on Heroku.
    - In doing this, I also had to host the Postgres database on Heroku using Heroku postgres.
    - I'm currently working with jake to get Unity game build to work on Heroku site.
    - For this milestone I made the architecture diagram and summarized the database structure.
    - Also, I'm working with Chirag to implement HTML stylization for the site.
        - Latest Comit to github: https://github.com/CSCI-3308-CU-Boulder/The-Ominous-Corp/commit/7c9763ae6b0fee50ad3435f4f8679c6a99e3b8bb
- Angelo Vacca:
    - Working on the prototype for the game and its features with Jake. Features created include: character firing projectiles, character hitboxes, character aiming, some work on character movement and overhead camera.
    - Assisting in finding resources and information about merging/integrating game with the website
    - Github commits are usually shared with Jake as we work together on the game and then Jake usually commits the changes made afterwards.
        - Latest commit with Jake: https://github.com/CSCI-3308-CU-Boulder/The-Ominous-Corp/commit/1f2618f2730c941feeb5f482fa876801f1006359
- Mauro Nannini
    - Has been unavailable in recent weeks due to illness, is just recently back and able to start helping. Looking into getting entire game file to appear on Node.js
- Chirag Telang
    - Working on creating the main home page in HTML, as well as the front end design of other webpages embedded within the home page
    - Assisting in finding resources and information about integrating my login page design with the actual database created by Ryan

Trello Board:



To Do

The user wants to loot other fallen players

Have multiplayer matchmaking set up over the server, with winnable games

Implement main menu / menu navigation

Matchmaking will take up to 50% less time.

As a player, I want to be able to see stats at the end of the match.

As a player, I want to have visual cue of how long reloading is taking.

As a player, I want to be able to customize my settings with regards to gameplay.

As a player, I want to be able to look

+ Add another card

Doing

Have server send email when a user registers or deletes an account
Oct 2 JM R

Update Login/Registration page UI
Oct 2 CT R

Create hitbox for user
Oct 2 AV JM M

+ Add another card

Done

Have basic unity game with "WASD" tank movement control

Have basic website hosted online via Heroku

Login page

Registration page

Password Encryption

Milestone 2 Document

Make a camera that follows the player

Implement projectile aiming/firing with mouse
Oct 2 AV JM M

+ Add another card

Challenges/Risks:

- Integrating the game with the website
    - Risk mitigation: Having the individual parts to be integrated fully planned out and formed.
- Not every feature planned for the game gets finished in time
    - Risk mitigation: Use the version of the game with all the parts that work instead of including everything.
- One of our group members contracted COVID-19 as well as a sinus infection, which kept him offline for a couple weeks
    - Risk mitigation: Communicating how to split/balance the workload between the group