SDDEC24-23

Website | https://sddec24-23.sd.ece.iastate.edu/
Email | sddec24-23@iastate.edu

Wargaming Simulator

Client | Reid Coates and Major Stephanie Jones (on behalf of ISU AFROTC)

Advisor | Ahmed Shakil

Team Members

Reid Coates | Client Coordination and Backend Development

Jack Kelley | Organization and Frontend Development

Alexander Hassan | Testing and Frontend Development

Luke Muilenburg | Frontend Development

Version 3 - Revised 12/5/24

Executive Summary

Development Standards & Practices Used

- Layered architectural structure (Practice)
- HTTP communication standards (Practice)
- [5] GameMaker Studio documentation standards (Practice)
- IEEE/ISO/IEC 26514-2021 - Design and Development of Information for Users (Standard)
- IEEE/ISO/IEC 14754-2021 - Software Life Cycle Processes - Maintenance (Standard)
- IEEE/ISO/IEC 41062-2019 - Software Life Cycle Processes - Software Acquisition (Standard)

Summary of Requirements

- Intuitive user interface
- MacOS and Windows Compatible
- Familiarizes user with Agile Combat Employment (ACE) strategy

Applicable Courses from Iowa State University Curriculum

- COM S 309 - Software Development Practices
- COM S 317 - Introduction to Software Testing
- COM S 319 - Construction of User Interfaces
- COM S 339 - Software Architecture and Design
- COM S 363 - Introduction to Database Management Systems

New Skills / Knowledge Acquired From Project

- GameMaker Studio (Game Maker Language)
- Configuring remote access to Raspberry Pi

Figures, Tables, and Definitions

Definitions

ACE - Agile Combat Employment (war strategy)

AFROTC - Air Force Reserve Officer Training Corps

GameMaker Studio - Video game development platform

GML - Game Maker Language (propriety GameMaker Studio programming language)

Wargaming - Action of playing a war game; Action of engaging in a campaign using military strategy

Table of Contents

1. Introduction

1.1. Problem Statement

Wargaming is an important part of the ISU AFROTC curriculum. Its purpose is twofold: to teach cadets to understand how the Air Force employs warfighting assets and to teach the concept of Agile Combat Employment (ACE). ACE is a proactive and reactive operational scheme of maneuver executed within threat timelines to increase survivability while generating combat power. Currently, these principles are taught practically through the use of a board game system called Global War 2030. This is where our task comes in.

Since the wargame is currently only represented physically it is difficult to scale to many users with different schedules and geographic locations. Our goal is to solve this access and scalability bottleneck by translating the existing board game into a digital format. This would allow us to reach a broader user base and deploy the software to more AFROTC groups than solely ISU AFROTC. This would allow more cadets to practice ACE through a more hands-on wargaming environment. By solving this format bottleneck, our goal is to increase the amount of practical wargaming experience available to AFROTC cadets and facilitate collaborative training through a 2 player multiplayer system.

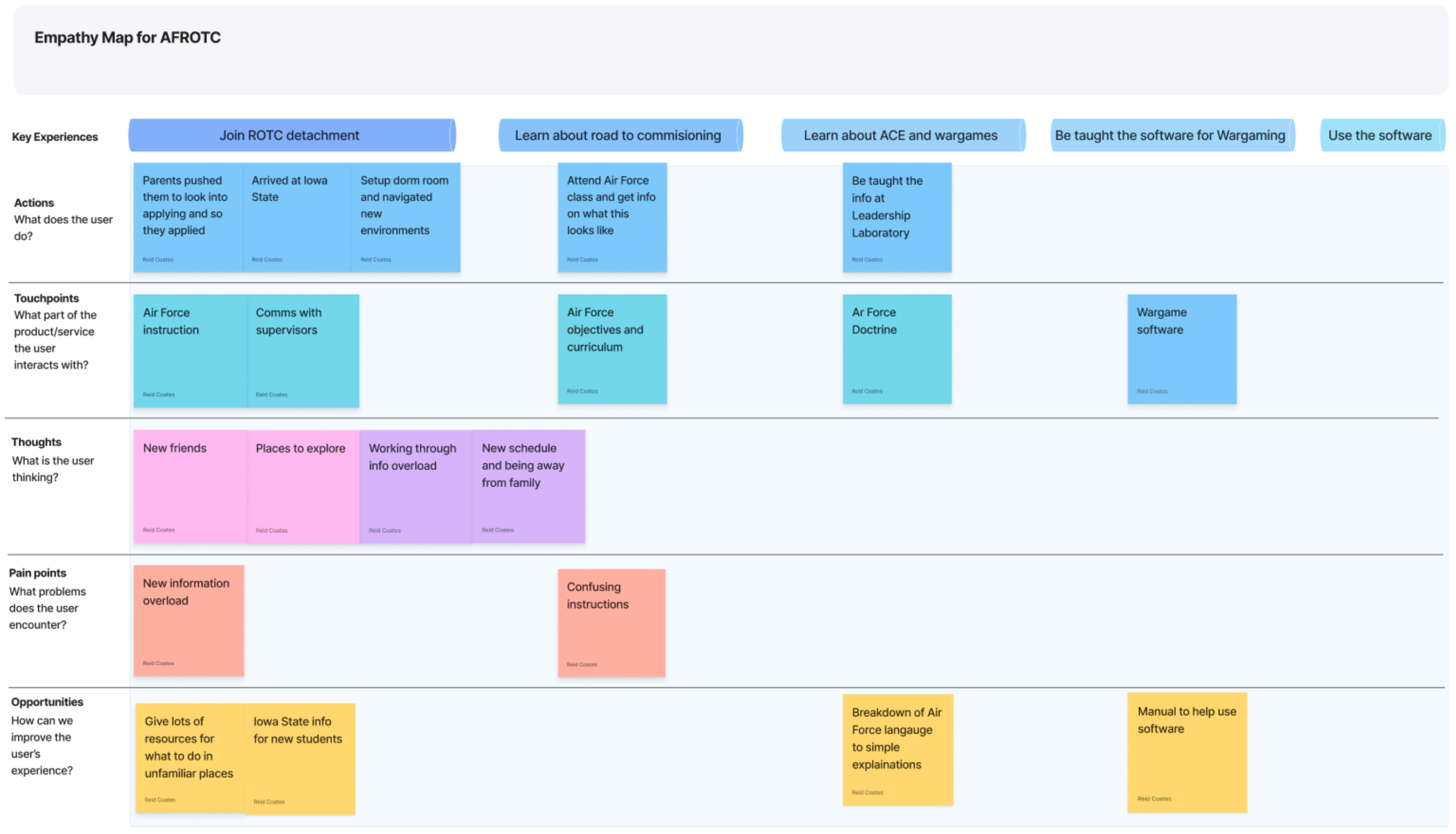
1.2. Intended Users

The product we create will be used by all detachments of the Air Force ROTC at Iowa State University. This product is intended to be used by both Cadre/instructors and Cadets at all levels in AFROTC. All AFROTC cadets will benefit from the results of the project because every cadet will be able to access the current wargaming sessions they are involved in remotely via the internet and will not be forced to update a physical game board. There are multiple groups of specific users who are relevant to our project, but there are three primary groups.

First, we have AFROTC Cadets in leadership positions. This group of cadets is generally very stressed as they need methods to teach, as well as learn, Wargaming to and from their subordinate cadets. Leader cadets are full-time students, as well as in a high-ranking student leadership position within the Air Force ROTC at Iowa State University. Cadet leaders of a detachment need to familiarize cadets with the Agile Combat Employment strategy. With their full schedule, the leaders do not have time to spend hours learning and explaining the complex rules for the various strategy games that currently exist and need a user-friendly application that can guide cadets through the rules and help them focus on learning strategy. Our game will provide direction for the legal moves and decisions that can be made to reduce the amount of time Cadet leaders need to spend teaching the technicalities of the game, and focus on the overall learning outcome of familiarization with the ACE Strategy.

Next, we have Cadets not in leadership positions. These students are very overwhelmed with adapting to the responsibilities of Cadet life and managing their schedules. Time is very valuable, and so is the ability to quickly and easily learn the new material being thrown at them on a weekly basis. With these needs, it is important that whatever system is used to teach the ACE Strategy is user-friendly (easy to pick up) and provides proper tools and calculation functionality to easily play through the turns (quick). Our virtual adaptation of the board game currently used for instruction Cadets on the ACE Strategy will provide resources to make the game run more quickly, as well as guide Cadets through the possible moves and decisions they can make to reduce the amount of time it takes to learn to play the game. At the end of this document, you can find the empath map for our Cadets not in leadership positions.

Finally, we have the Instructors / Cadre. This group of users is also very busy, and they are required by profession to teach AFROTC cadets how to employ Air Force assets, as well as Agile Combat Employment. The instructors need to be able to consistently and efficiently teach these objectives, and the current methods of doing so are both physical and overly complicated, thus leading to a degradation of the teaching of ACE. They will benefit from the Wargaming simulator because each instructor will be able to teach ACE without having to create and employ a physical alternative to Wargaming, and instructors will be able to use our simulator with only a simple internet connection.



Empathy Map for AFROTC

Key Experiences	Join ROTC detachment	Learn about road to commisioning	Learn about ACE and wargames	Be taught the software for Wargaming	Use the software
Actions What does the user do?	Parents pushed them to look into applying and so they applied (Reid Coates) Arrived at Iowa State (Reid Coates) Setup dorm room and navigated new environments (Reid Coates)	Attend Air Force class and get info on what this looks like (Reid Coates)	Be taught the info at Leadership Laboratory (Reid Coates)		
Touchpoints What part of the product/service the user interacts with?	Air Force instruction (Reid Coates) Comms with supervisors (Reid Coates)	Air Force objectives and curriculum (Reid Coates)	Ar Force Doctrine (Reid Coates)	Wargame software (Reid Coates)	
Thoughts What is the user thinking?	New friends (Reid Coates) Places to explore (Reid Coates) Working through info overload (Reid Coates) New schedule and being away from family (Reid Coates)				
Pain points What problems does the user encounter?	New information overload (Reid Coates)	Confusing instructions (Reid Coates)			
Opportunities How can we improve the user's experience?	Give lots of resources for what to do in unfamiliar places (Reid Coates) Iowa State info for new students (Reid Coates)		Breakdown of Air Force langauge to simple explainations (Reid Coates)	Manual to help use software (Reid Coates)	

Figure 1 - AFROTC Cadet Empathy Map

2. Requirements, Constraints, and Standards

2.1. Requirements and Constraints

2.1.1. Functional Requirements

A primary functional requirement of our project is deployability. The frontend will be developed using GameMaker Studio which is an IDE that is specifically intended for video game development. It is compatible with both MacOS and Windows, which will add more support for our user base, as both operating systems are prominent. The backend will be developed as a Java Spring Boot application and initially deployed to an ISU managed virtual machine. To meet our deployability requirement, we will produce documentation outline the steps for deploying the Java artifact on an Ubuntu Linux distribution. This will give our client the information necessary to change the deployment location to a more suitable host if needed in the future.

2.1.2. Resource / Physical Requirements

In regards to resource and physical requirements, the system shall be deployable as a Java Spring Boot application. There must be documentation outlining the deployment of the Java artifact on a 2024 or later Ubuntu Linux distribution, which is where the initial backend Spring Boot system is hosted.

The system must have a modular backend in which SQL (SQL 4 standard) databases can be exchanged with no development time except the generation of the query statements and Java methods to execute the statements (**constraint**). This requirement is created in an effort to allow the backend system to be adapted by other teams in the future, possibly other ROTC Detachments. The system must store data on a server that can be queried by up to 2 users to facilitate a game between 2 players who are not on the same network and not actively connected to each other (**constraint**).

2.1.3. Aesthetic / UX / UI Requirements

Usability is a key requirement for our Wargaming Simulator. The system shall teach users from any college degree background to independently operate software, in order to learn the concepts of wargaming (Objective 10.1-3 AFROTCI 36-2011 Vol 1) to the proficiency level outlined by HQ AFROTC. Understanding the workload of our user base (AFROTC Cadets) and the time frame in which the game is to be played (over the course of a semester), it is important that the interfaces of our game are intuitive and guide the user in not only quickly playing their turn, but also explaining the rules through gameplay. The board game predecessor to our project has a heavy learning curve, and part of the objective of our digital recreation is to lessen that learning curve.

The system shall support the creation of multiple game sessions, each between 2 different users, and store multiple games per account for a time period of no less than six months (**constraint**). The application shall be accessible on a standard 16:9 display at a minimum (**constraint**). The game shall be playable with a mouse and keyboard as I/O devices (**constraint**).

2.2. Engineering Standards

2.2.1. Code Conventions for the Java Programming Language

Our backend application will be a Java Spring Boot Application; thus, good practices and coding conventions shall be followed whenever applicable. We want this software to be maintainable by someone other than the authors if need be. Additionally, 80% of the lifetime cost of a software application is due to maintenance, so we shall follow the Code Conventions for the Java Programming Language via Oracle in order to make our systems maintainable and scalable.

2.2.2. Rule Book

Since our game will be a child of a physical board game, there is an existing rulebook for said board game. While this is not an official engineering standard by any means, it does serve as a set of guidelines to help us not only shape our UI, but also the mechanics and logic of our system.

2.2.3. Lack of Relevant Standards

Outside of the conventions for Java, our team was unable to find many relevant standards for our project. Taking away the educational nature of our project, it is simply a video game. Gaming is an industry that doesn’t have many official development standards. This is because it is an industry with so many vastly different products, so the requirements and necessary standards are unique from project to project. With this, we plan to develop our own data structure, procedural, and naming standards before getting into development of the core project.

A relevant aspect of the development of our project is that our client is the AFROTC, which is an organization that has many of their own protocols and standards that need to be followed. With that in mind, there are not any technical standards that would dictate constraints or procedures for us to follow in the development of our project. The standard delivered to us (Objective 10.1-3 AFROTCI 36-2011 Vol 1) only states the learning outcome of the project. That is, the game should help familiarize Cadets with the ACE strategy. Outside of that requirement, our team has been given near free reign to go whatever direction we think will best accomplish that learning outcome, while also remaining true to the original board game.

3. Project Plan

3.1. Project Management / Tracking Procedures

Our team is adopting a blend of waterfall and agile development. This includes Weekly SCRUM Sprints, Monday team meetings, Thursday check-in/advisor meetings, and Flowchart tasks with color codes for sprints. This is because our project requirements are largely not fixed, and thus, our team must be able to adapt to ever-changing requirements. We have planned multiple sprints out from week one, so we did not adopt a pure agile management style, but we still have standups and client-involvement, thus, a blend of management styles was achieved. Our Team tracks progress via GitLab and Sprint plans provided in section 3.2.

3.2. Task Decomposition

Since our project requires a robust front and back-end codebase, task decomposition is very important to us to ensure that developers working on both sides of the problem are able to successfully integrate code during merges. The agile approach we are making use of helps us in this regard. The next page contains our task decomposition chart (Figure 2), which is broken up by design sprint.

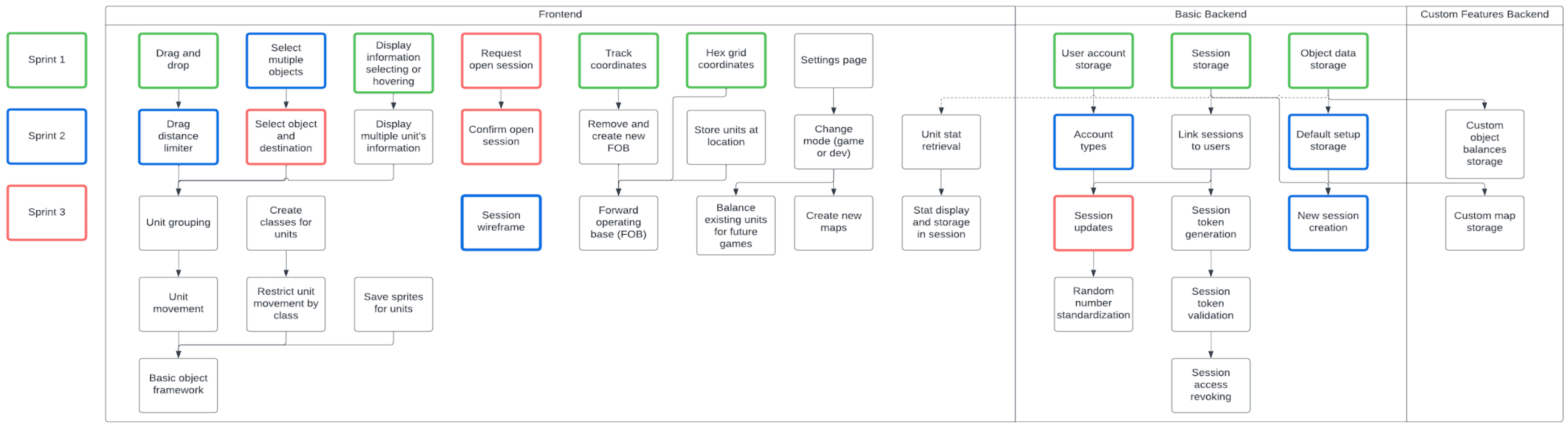


Figure 2 - Task Decomposition Flowchart

3.3. Project Proposed Milestones, Metrics, and Evaluation Criteria

3.3.1. Evaluation Criteria and Metrics

Progress for our project will be determined by accomplishing a task outlined in a project milestone. As we are developing a game, we have split up the tasks into reasonable chunks that flow into each other effectively for development. Many of the milestones are feature implementations as opposed to something that can be quantified.

Our first metric values will come when we reach our user testing period mid-Fall. These values will be qualitative, dictated by the experience conveyed to us by a group of AFROTC cadets who will be using our simulator. We intend to develop an experience form to receive feedback on the user experience, ease of learning the game rules, and a place for comments and additional feedback as to how we can better suit our game to fulfill the task of effectively familiarizing cadets with the ACE strategy.

The project milestone and respective subtasks are listed below, along with a brief description of each subtask. A Gantt chart with all milestones and tasks can be found below the list, outlining the overall timeline for the milestone, as well as the timeline for each task within the milestone.

3.3.2. Project Milestones and Subtasks

1. Game Component Proofs of Concept

    a. Hexagonal Game Grid
        - Develop frontend logic for using a hexagonal grid as opposed to squares

    b. Drag and Drop Asset Movement
        - Develop frontend logic for clicking on an object, and dragging it around the screen, then dropping it in another grid

    c. Mouse Hover Pop-Up Information Window

- Develop frontend logic for creating a pop-up window that displays game information to the user when hover the mouse over a tile

d. Coordinate Pairs API
- Develop backend logic for receiving and storing grid location coordinate pairs from frontend
- Develop backend logic for sending stored grid location coordinate pairs to frontend

e. Match Making API
- Develop backend logic for creating game lobbies
- Develop backend logic for connecting two users to the same specified lobby

2. User-to-User Communication

a. GameMaker HTTP Requests
- Develop frontend logic for making HTTP requests to interface with the backend

b. Database Implementation
- Implement a database for the backend to store and retrieve game data

3. Core Game Logic (Turns, Movement, Combat, Win Conditions)

a. Asset Movement (Planes)
- Implement framework for moving game assets (namely planes) around the game board, updated locations should be sent to the backend for storage and update on the opponent's board

b. User Turn Iterations
- Implement framework for guiding the user through all decisions and possible actions to be made on a turn, and then execute the actions and save data to the backend

c. Combat Losses Calculations
   - Implement framework for calculating the losses after combat (in accordance with the rules supplied by our client)

d. Win Condition Determination
   - Implement framework for determining the winner of a game (in accordance with the rules supplied by our client)

e. Match Making API
   - Implement framework for users generating lobbies
   - Implement framework for users inputting and connecting to specific lobbies

f. Session Variable API
   - Implement framework for sending and receiving game variables (board states)

4. Game Assets (Art, Effects, Points, Purchasing)

   a. Resource Art Assets
      - Create and upload sprite art assets for planes, armaments, and other game assets

5. Application Deployment

   a. Generate Executable Application through GameMaker Studio
      - Generate an executable using GameMaker Studio for both MacOS and Windows that we can host and distribute to users through our web server

   b. Host Backend as Remote Server on Raspberry Pi
      - Configure a Raspberry Pi to operate as a server, running our backend code and our database

6. Play Testing (with Cadets)

   a. Performance Assessment (Finding and Fixing Bugs)

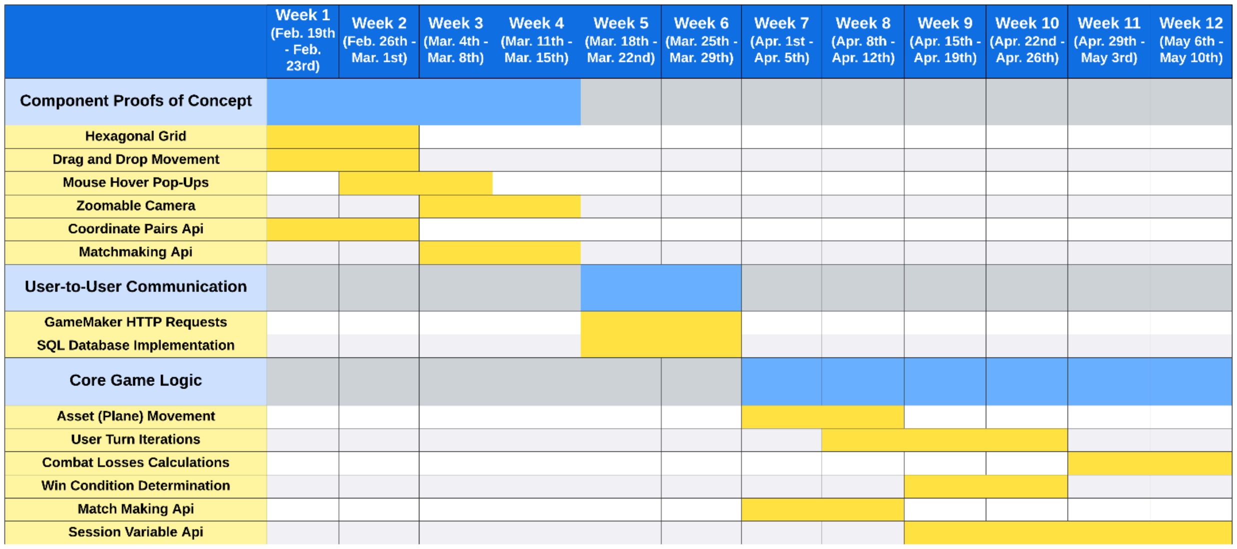
  - Be more intensely involved in early user testing to find and fix bugs that appear once the game is released to a wider user-base

b. User Interface Assessment (Usability)
  - Later in testing, provide opportunity to receive feedback from cadets about the user interfaces and usability of the simulator

c. Game Rule Clarity Assessment (Usability / Learnability)
  - Later in testing, provide opportunity to receive feedback from cadets about the game rule clarity; How easy is it to learn to play the game?

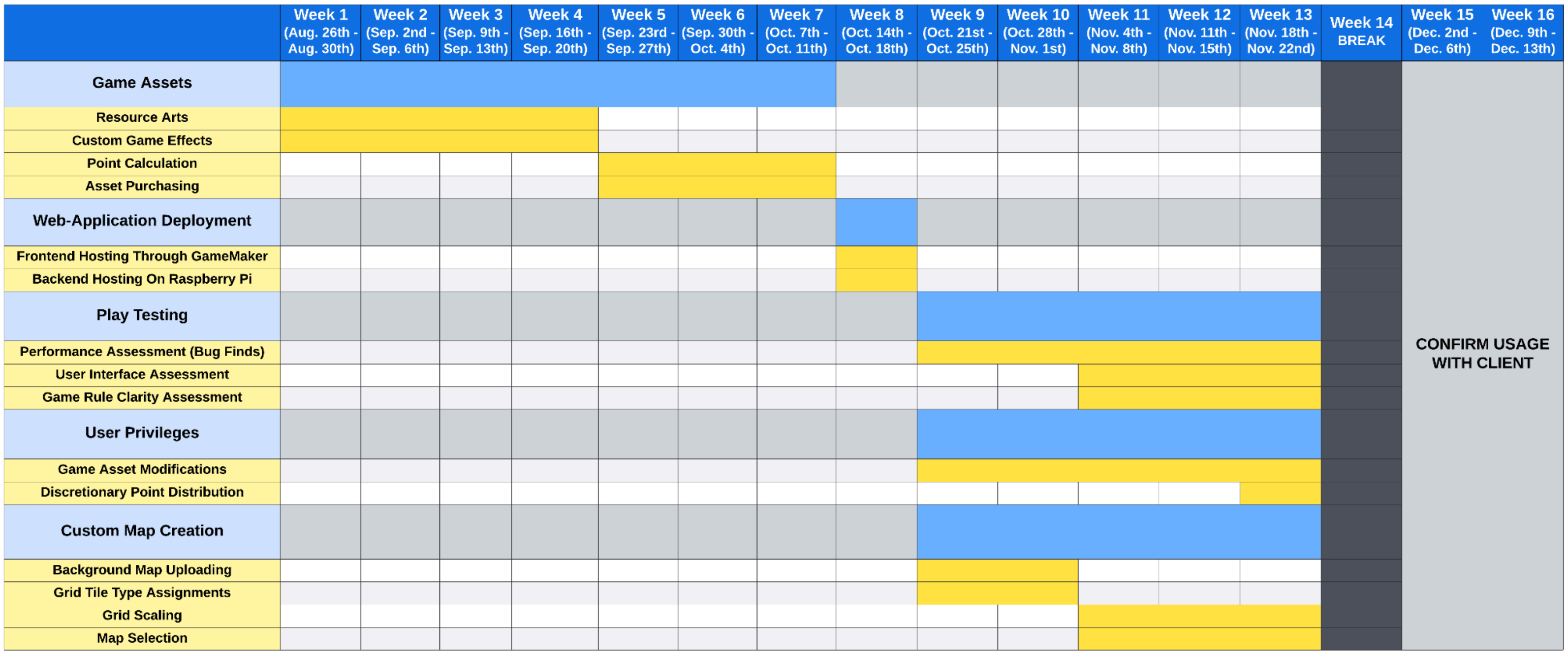
3.4. Project Timeline / Schedule

Spring 2024 Milestones



	Week 1 (Feb. 19th - Feb. 23rd)	Week 2 (Feb. 26th - Mar. 1st)	Week 3 (Mar. 4th - Mar. 8th)	Week 4 (Mar. 11th - Mar. 15th)	Week 5 (Mar. 18th - Mar. 22nd)	Week 6 (Mar. 25th - Mar. 29th)	Week 7 (Apr. 1st - Apr. 5th)	Week 8 (Apr. 8th - Apr. 12th)	Week 9 (Apr. 15th - Apr. 19th)	Week 10 (Apr. 22nd - Apr. 26th)	Week 11 (Apr. 29th - May 3rd)	Week 12 (May 6th - May 10th)
Component Proofs of Concept	🟦	🟦	🟦	🟦								
Hexagonal Grid	🟨	🟨										
Drag and Drop Movement	🟨	🟨										
Mouse Hover Pop-Ups		🟨	🟨									
Zoomable Camera			🟨	🟨								
Coordinate Pairs Api	🟨	🟨										
Matchmaking Api			🟨	🟨								
User-to-User Communication					🟦	🟦						
GameMaker HTTP Requests					🟨	🟨						
SQL Database Implementation					🟨	🟨						
Core Game Logic							🟦	🟦	🟦	🟦	🟦	🟦
Asset (Plane) Movement							🟨	🟨				
User Turn Iterations								🟨	🟨	🟨		
Combat Losses Calculations											🟨	🟨
Win Condition Determination									🟨	🟨		
Match Making Api							🟨	🟨				
Session Variable Api									🟨	🟨	🟨	🟨

Fall 2024 Milestones



	Week 1 (Aug. 26th - Aug. 30th)	Week 2 (Sep. 2nd - Sep. 6th)	Week 3 (Sep. 9th - Sep. 13th)	Week 4 (Sep. 16th - Sep. 20th)	Week 5 (Sep. 23rd - Sep. 27th)	Week 6 (Sep. 30th - Oct. 4th)	Week 7 (Oct. 7th - Oct. 11th)	Week 8 (Oct. 14th - Oct. 18th)	Week 9 (Oct. 21st - Oct. 25th)	Week 10 (Oct. 28th - Nov. 1st)	Week 11 (Nov. 4th - Nov. 8th)	Week 12 (Nov. 11th - Nov. 15th)	Week 13 (Nov. 18th - Nov. 22nd)	Week 14 BREAK	Week 15 (Dec. 2nd - Dec. 6th)	Week 16 (Dec. 9th - Dec. 13th)
Game Assets	🟦	🟦	🟦	🟦	🟦	🟦	🟦							⬛		
Resource Arts	🟨	🟨	🟨	🟨										⬛		
Custom Game Effects	🟨	🟨	🟨	🟨										⬛		
Point Calculation					🟨	🟨	🟨							⬛		
Asset Purchasing					🟨	🟨	🟨							⬛		
Web-Application Deployment								🟦						⬛		
Frontend Hosting Through GameMaker								🟨						⬛		
Backend Hosting On Raspberry Pi								🟨						⬛		
Play Testing									🟦	🟦	🟦	🟦	🟦	⬛		
Performance Assessment (Bug Finds)									🟨	🟨	🟨	🟨	🟨	⬛	CONFIRM USAGE WITH CLIENT	
User Interface Assessment											🟨	🟨	🟨	⬛		
Game Rule Clarity Assessment											🟨	🟨	🟨	⬛		
User Privileges									🟦	🟦	🟦	🟦	🟦	⬛		
Game Asset Modifications									🟨	🟨	🟨	🟨	🟨	⬛		
Discretionary Point Distribution													🟨	⬛		
Custom Map Creation									🟦	🟦	🟦	🟦	🟦	⬛		
Background Map Uploading									🟨	🟨				⬛		
Grid Tile Type Assignments									🟨	🟨				⬛		
Grid Scaling											🟨	🟨	🟨	⬛		
Map Selection											🟨	🟨	🟨	⬛		

Figure 3 - Spring and Fall 2024 Milestone Gantt Charts

3.5. Risks and Risk Management / Mitigation

Note: All tasks listed below are described further in section 3.4.

1. Game Component Proof of Concepts
   a. Risk of underestimating the complexity of certain game components, creating delays in implementation.
      - Risk factor: 0.4 (medium)
   b. Merging separate proofs of concepts into one common codebase may require significant overhead and reworking.
      - Risk factor: 0.3 (low)
2. User-to-User Communication
   a. Performance issues if the communication between frontend and backend systems is not optimized for scalability.
      - Risk factor: 0.5 (medium)
   b. Difficulty in keeping track of game states between users, leading to inconsistencies or opportunities for cheats.
      - Risk factor: 0.5 (medium)
3. Core Game Logic
   a. Balancing game difficulty and fairness will require a lot of testing as a large number of game scenarios are possible.
      - Risk factor: 0.3 (low)
   b. Complexity in the implementation of game mechanics may lead to unintended functionality and bugs.
      - Risk factor: 0.4 (medium)
4. Game Assets
   a. Resource Arts may have compatibility issues with Game Maker Studio as asset resources are derived from a physical board game.

- Risk factor: 0.4 (medium)

5. Executable Application Development

   a. Compatibility issues with non-desktop or laptop devices accessing the application.
      - Risk factor: 0.3 (low)

   b. Software Security issues if we do not properly authenticate server traffic and encrypt/protect user data, databases, and server files/settings.
      - Risk factor: 0.5 (medium)

6. Play Testing

   a. Cadets may not provide timely and constructive feedback.
      - Risk factor: 0.3 (low)

   b. Identifying and reworking critical issues discovered by the cadets may require a lot of backtracking work.
      - Risk factor: 0.5 (medium)

3.6. Personnel Effort Requirements

TASK	**Estimated Work Time (Hours)**
Game Component Proofs of Concept	
Develop frontend logic for using a hexagonal grid as opposed to squares	6
Develop frontend logic for clicking on an object, and dragging it around the screen, then dropping it in another grid	6
Develop frontend logic for creating a pop-up window that displays game information to the user when hover the mouse over a tile	8
Develop frontend logic for a camera that can zoom in and out, allowing for scalable map sizes when developing maps	6

Develop backend logic for receiving and storing grid location coordinate pairs from frontend	8
Develop backend logic for sending stored grid location coordinate pairs to frontend	6
Develop backend logic for creating game lobbies	12
Develop backend logic for connecting two users to the same specified lobby	8
User-to-User Communication	
Develop frontend logic for making HTTP requests to interface with the backend	20
Implement a database for the backend to store and retrieve game data	20
Core Game Logic (Turns, Movement, Combat, Win Conditions)	
Implement framework for moving game assets (namely planes) around the game board, updated locations should be sent to the backend for storage and update on the opponent's board	10
Implement framework for guiding the user through all decisions and possible actions to be made on a turn, and then execute the actions and save data to the backend	12
Implement framework for calculating the losses after combat (in accordance with the rules supplied by our client)	10
Implement framework for determining the winner of a game (in accordance with the rules supplied by our client)	6
Implement framework for users generating lobbies	10
Implement framework for users inputting and connecting to specific lobbies	12
Implement framework for sending and receiving game variables (board states)	20
Game Assets (Art, Effects)	
Create and upload sprite art assets for planes, armaments, and other game assets	10
Web-Based Application Deployment	
Configure our GameMaker frontend application to be hosted as a web-based application using the GameMaker Studio provided hosting services	10

Configure a Raspberry Pi to operate as a server, running our backend code and our database	20
Play Testing (with Cadets)	
Be more intensely involved in early user testing to find and fix bugs that appear once the game is released to a wider user-base	4
Later in testing, provide opportunity to receive feedback from cadets about the user interfaces and usability of the simulator	3
Later in testing, provide an opportunity to receive feedback from cadets about the game rule clarity; How easy is it to learn to play the game?	5
TOTAL ESTIMATED HOURS:	**232**

Table 1 - Estimated Personnel Hours

Each task in the above table has an estimated number of hours to complete accompanying it to the right. The rows that are highlighted denote the 6 major task categories, with each category's subtasks below.

3.7. Other Resource Requirements

The deployment of this project requires a computer or virtual machine with an Ubuntu Linux distribution installed. For now, this project is deployed on an ISU virtual machine created explicitly for this project. In the future, we plan to host the backend on a Raspberry Pi that will be in the Armory, so we will need our client to obtain a Raspberry Pi for us to implement that.

Outside of physical resources, we also need personnel. We intend to get a group of cadets to volunteer to playtest our simulator to help find bugs and other small fixes initially. We will have our volunteer group continue to play the game over multiple weeks, so they can then give us feedback in regards to the user experience. Our game needs to quickly and easily convey the rules of the game, so our testing group will be tasked with analyzing the ability of our project to do so. They will also provide us with any additional feedback they see necessary in regards to the ability of our software to effectively teach the ACE strategy.

4. Design

4.1. Design Context

4.1.1. Broader Context

Our design is made to the standards of the AFROTC customer. This includes cadets and other personnel.

Area	Description	Examples
Public health, safety, and welfare	Our design increases the ability of AFROTC to conduct wargaming exercises to practice their understanding of strategic situations and expands their capability to safeguard.	By making wargaming easier to access, our design increases AFROTC readiness.
Global, cultural, and social	Groups may include but are not limited to specific communities, nations, professions, workplaces, and ethnic cultures.	Our game focuses on teaching the ACE combat strategy and aims to reflect the values of the Air Force and influence a culture that emphasizes ACE.
Environmental	A large part of our software deals with refueling aircraft. Efficient fuel use is a large part of whether or not a player will succeed.	Refueling aircraft need to be placed to maximize the efficiency and coverage of a player's fighter aircraft. Additionally, our plans are to host our free software on a single Raspberry Pi, and this will negligibly affect our environment.
Economic	This can include the financial viability of your product within your team or company, cost to consumers, or broader economic effects on communities, markets, nations, and other groups.	Product needs to remain affordable for target users, product creates or diminishes opportunities for economic advancement, high development cost creates risk for organization

Table 2 - Broader Context Analysis

4.1.2. Prior Work / Solutions

This project is new to Senior Design and has not been previously attempted by another student team. As a result, review of literature related to our task and tools is important. Since our goal is to translate an existing board game into a digital format and make improvements on it, the rulebook [1] is key to our success. We spent significant time this semester reading the rulebook and playing truncated practice games to make sure we have a full understanding of the existing game and what our customer wants us to replicate. In addition to the rules for the game, we also read a lot of literature on our primary development tool, that being Game Maker Studio 2. The GameMaker Manual website was very helpful in getting the team acquainted with the software [2]. The last major document for us to review is related to the competitive advantage of our project. While many pieces of wargaming software exist, none of the publicly available softwares are designed to teach ACE, or Agile Combat Employment [3]. This concept is mentioned in other locations of our design doc, but it is really the keystone or the ‘why’ for the existence of our wargame. Teaching AFROTC students to practically employ its tenants is our customer’s main goal and therefore a primary metric for the team.

As mentioned above, there are many competitors in the digital wargaming market space. These include franchises we have researched such as Total War, Paradox, and Eugen. While these pieces of software would be very difficult for us to compete with, we have a significant advantage over existing software when it comes to our AFROTC customer. Our software is designed to cover ACE principles, while these other wargames are not designed with this goal in mind. Additionally, our software is made to work with very low system requirements so that it can be run on a wide variety of machines instead of just computers with a dedicated graphics card, for example. Due to these factors, we believe the solution we are developing better serves the needs of ISU AFROTC than existing wargaming solutions do.

4.1.3. Technical Complexity

The following evidence seeks to justify that our project is of significant technical complexity:

1. **Subsystems**

    a. Hex Map and Coordinates System

    - The project includes a hex map with hex coordinates This subsystem involves intermediate principles of geometry, axial coordinate systems, and data structures to adjust and display the map data efficiently.

    b. Sprite Movement

    - Implementing sprites that can be moved around the map involves animation, user movement constraints, and sending all movements via HTTP requests to the backend system.

    c. User Authentication and Server Communication

    - Our project incorporates a server-client architecture with user authentication features. This subsystem encompasses network protocols, encryption and Hashing algorithms, and database management principles.

    d. Core Game Logic

    - Implementing core game logic based on the rule book of the physical board game is complex. This subsystem will likely involve algorithms for game mechanics, decision-making processes, and player constraints.

2. **Scientific, Mathematical, or Engineering Principles**

    a. Geometry

- Principles of geometry are utilized in showing and adjusting the hex map and coordinates system, and performing game calculations based on a hexagonal coordinate system.

b. Data Structures

- Various data structures, such as lists, maps, and GML structs, are used to efficiently store and adjust game-related information, such as map tiles, sprites, and user attributes.

c. Networking

- Understanding network protocols, latencies, and security principles is necessary for implementing server-client communication and user authentication.

d. Game Design Principles

- Incorporating game design principles such as balance, strategy, and usability requires an understanding of human-computer interaction and user experience design.

3. **Challenging Requirements**

a. Implementing User Authentication

- Integrating secure user authentication and game creation between players requires adherence to industry standards for encryption, password hashing, and secure communication protocols to ensure data confidentiality and user privacy. We currently use SHA256 Hashing for authentication, a cryptographically secure industry standard.

b. Core Game Logic Implementation

- Translating the rules and mechanics of a physical board game into a digital format while maintaining gameplay balance, usability, and complexity poses significant challenges in design and implementation. We also need the game to be realistic in order to best teach ACE to ROTC Cadets.

4.2. Design Exploration

4.2.1. Design Decisions

Backend Portability for Various Hosting Scenarios

Currently, the backend server for our project is being hosted on a virtual machine provided by Iowa State. We chose to use this method of hosting for development because it is a system we are familiar with and was the lowest barrier to entry option we had, meaning we could focus more on logic instead of framework.

Moving forward, however, we intend to host our backend server on a Raspberry Pi that will be stored in the Armory. This is because the virtual machine provided to us by Iowa State is much more computing power than we need to run our project, and eliminates the possibility of Iowa State shutting down the server after our team has graduated.

Despite hosting on a Raspberry Pi being a sound enough solution for our project implementation, an objective of our development is to make the backend as portable as possible to allow the AFROTC to change server hosting as desired. This could be requesting permanent VM space through Iowa State to host the server similar to how we are hosting it now, or renting server space through some other party. This could reduce the potential of the physical server (Raspberry Pi) being unplugged or disrupted in some other way due to miscommunication or lack of security and physical organization.

Descriptive and Intuitive User Interface to Simplify Game Rules

At a high level, our project is simply converting a war strategy board game used by the AFROTC into a web-based equivalent. With this, we very well could have taken the approach of simply digitizing the game assets. Meaning we replicate the board, game pieces, add functionality for dragging those assets around, and leave our contributions at that. This would resolve the issue of limited resources and having to acquire physical game pieces to play the game and learn about the ACE strategy.

To manage the issue of rule complication and reduce the amount of time spent learning the game, allowing the users to focus on learning outcomes instead of game rules, we decided to implement useful interfacing concepts to make the game much easier to learn and play, allowing cadets to focus on strategy. Some of these concepts include displaying possible move locations for game pieces and preventing players from making any illegal moves.

Other concepts include displaying more information to users about the game assets they have available to them and handling all of the combat calculations and losses for the users. In our experience with the physical board game, we found these to be the most prohibitive aspects of the game to the learning experience. Those aspects are also the most time consuming, as keeping track of moves and submitting them for calculation takes a long time with the current board game, and our digital version of the game will be able to handle those calculations much more easily.

Determining Whether to Use HTTP or HTTPS for Server Calls

Our team was faced with the decision of whether to use HTTP or HTTPS requests for our backend. We have started our development with HTTP because it is the framework our team was most familiar with, as well as it is easier to implement for us to get started with testing concepts and getting a base framework in place. However, we have reached the point of user account creation and managing usernames and passwords for our users. This led our team to have a discussion on how we wanted to manage security.

The initial thought our team had in regard to security was to hash the usernames and passwords. This then led to a discussion as to what hashing algorithm we should use, or if we should develop our own algorithm. We decided for the moment that we will use one of the provided hashing algorithms in Game Maker Studio, as it provides a level of security without taking too much implementation time.

Moving forward, we plan to do one of three things: implement our own custom hashing algorithm, update the backend to use HTTPS protocols, or do both of the prior two options.

4.2.2. Ideation

An example of a design decision our team was faced with was the gameboard grid for our game. The Lotus Blossom diagram below (Figure 4) displays some of the possible solutions we considered, as well as some pros and cons around those solutions to help make our decision. In the diagram, the light gray boxes are pros and the dark gray boxes are cons. The five possible solutions we came up with were a square-tiled grid, a square-tiled grid that has the ability to be scaled up or down, a hexagonal-tiled grid, a hexagonal-tiled grid that has the ability to be scaled up or down, and a hexagonal-tiled grid with a camera that can be zoomed in or out.

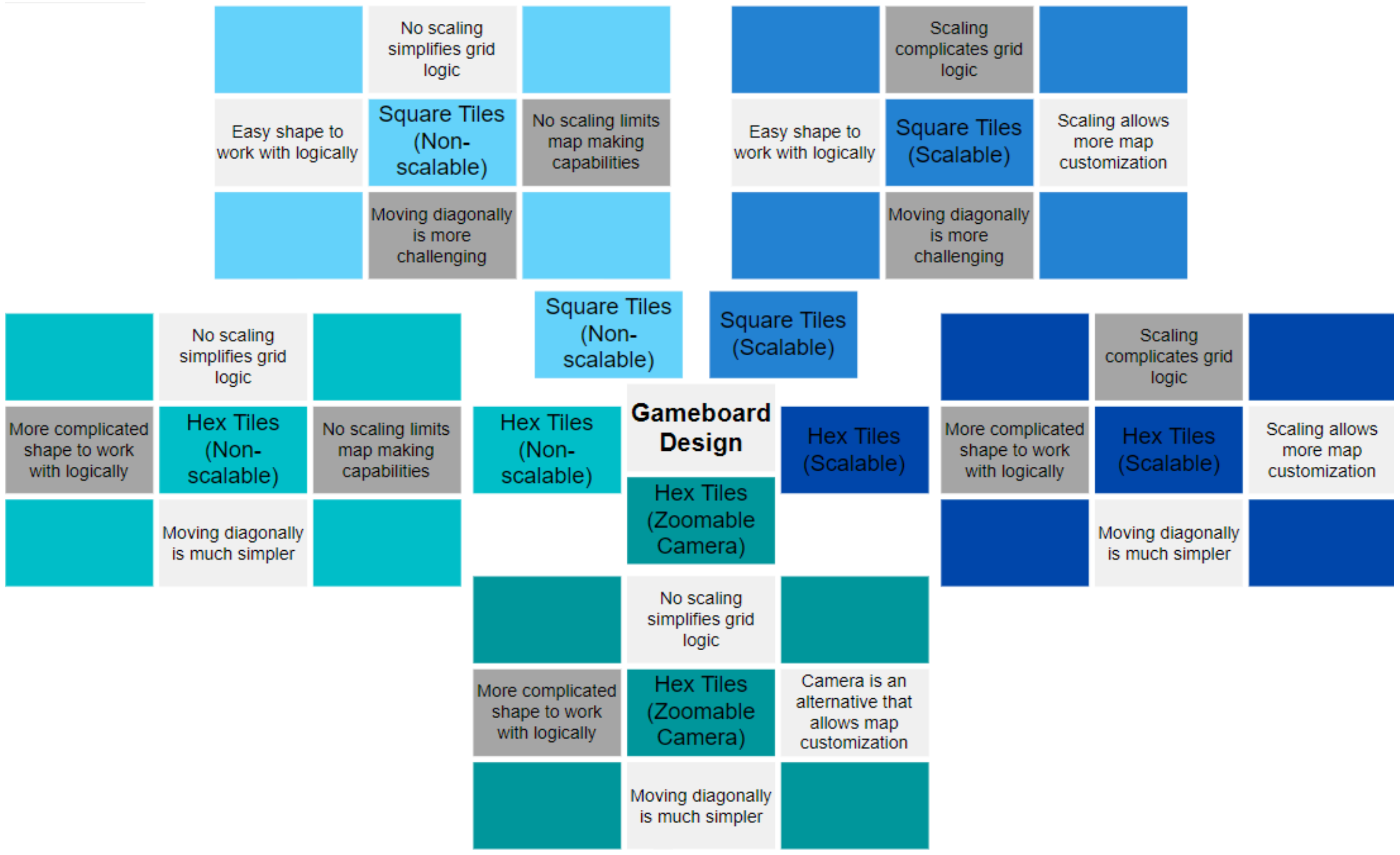

No scaling simplifies grid logic
Easy shape to work with logically
Square Tiles (Non-scalable)
No scaling limits map making capabilities
Moving diagonally is more challenging
Scaling complicates grid logic
Easy shape to work with logically
Square Tiles (Scalable)
Scaling allows more map customization
Moving diagonally is more challenging
Square Tiles (Non-scalable)
Square Tiles (Scalable)
No scaling simplifies grid logic
More complicated shape to work with logically
Hex Tiles (Non-scalable)
No scaling limits map making capabilities
Moving diagonally is much simpler
Hex Tiles (Non-scalable)
Gameboard Design
Hex Tiles (Scalable)
Hex Tiles (Zoomable Camera)
Scaling complicates grid logic
More complicated shape to work with logically
Hex Tiles (Scalable)
Scaling allows more map customization
Moving diagonally is much simpler
No scaling simplifies grid logic
More complicated shape to work with logically
Hex Tiles (Zoomable Camera)
Camera is an alternative that allows map customization
Moving diagonally is much simpler

Figure 4 - Gameboard Design Lotus Blossom

4.2.3. Decision-Making and Trade-Off

We found with each of the square-tiled options, the shape would be easy to work with for visuals and most other logic, but diagonal movement would be difficult to manage. The hexagonal-tiled options would all involve more logic, but would yield a better gameplay experience with ability to move diagonally effectively. With this, we decided we would be using a hexagonal-tiled grid, as it is also more true to the original grid used for the board game we are recreating.

We then found that all of the non-scalable grid options would limit the ability to create custom maps, which is functionality that our client has expressed interest in being able to do. On the contrary, we found that all of the scalable grid options would be difficult to implement logically, particularly with the visuals and scaling the assets. With this, we developed a third scaling solution which is creating a camera that can zoom in or out. This would allow us to restrict the visible board area and programmatically disable the tiles that are out of view, allowing us to effectively scale the board without changing any asset sizes.

With all of the findings from analyzing our Lotus Blossom diagram, we determined the best shape to use for our grid would be hexagonal tiles and the best scaling option would be to use a zoomable camera object. Thus, we have decided to use the hexagonal-tile with zoomable camera grid option from the Lotus Blossom diagram (seen at the bottom of the diagram above).

4.3. Proposed Design

4.3.1. Overview

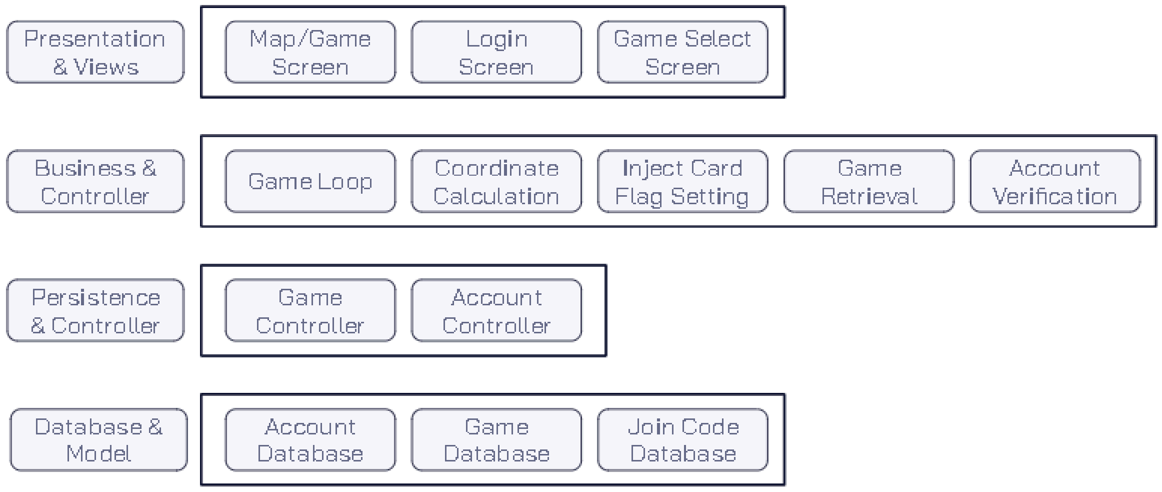


Figure 5 - Wargaming Simulator Architectural Layer Diagram

The system we are developing is an Air Force Wargaming Simulator where players, or users, are able to log into the game and play a simulated wargame, or battle, between another logged-in player. This is done via one player requesting the creation of a game through a menu and then sending the requested code to another player they want to battle. Then, the system will allow the users to submit their game moves on a hex tile game board. Each game lasts roughly 10 turns, and winners are decided on predetermined winning conditions, such as holding certain tiles at the end of the game or destroying all enemy units. All in all, the purpose of this game is to teach the ACE combat strategy to Air Force ROTC Cadets in accordance with USAF requirements.

4.3.2. Detailed Design and Visuals

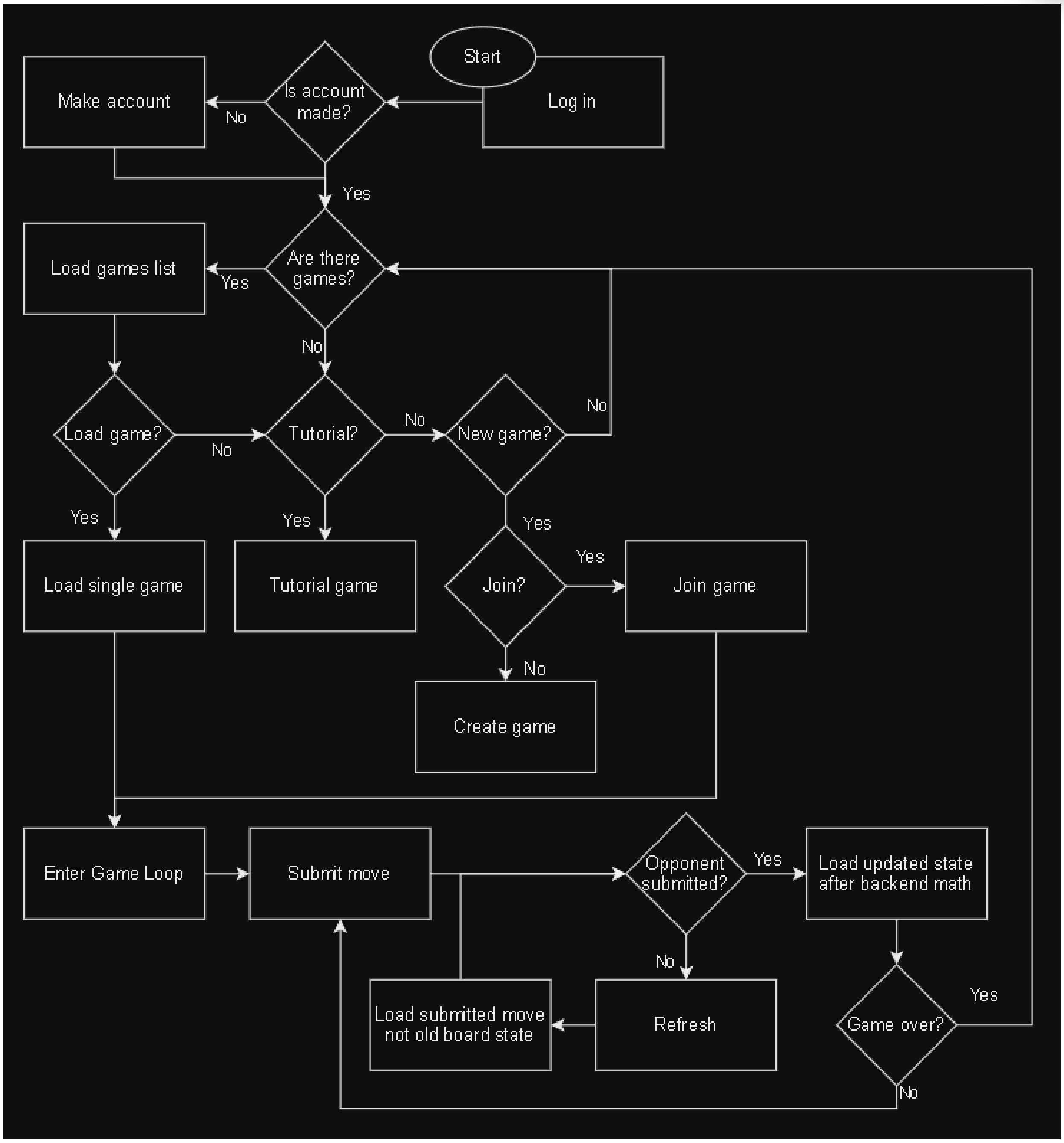


Figure 6 - Game Logic Flowchart

Our design for the Wargaming Simulator's visuals includes both Front and backend visuals. For our frontend visuals, we have 2 notable layers: the Presentation layer, or what the user sees, and the Persistence layer, which handles communication from the frontend to the backend.

Our presentation layer has 3 main core features vital to the project's design. One is the hex grid, which is how sprites move around the map via the user's control. This control is limited via the sprite's stats. We also have Resource Lists, which is the pool of sprites the user will be able to control, select, and otherwise view and manage. Then we have Menus, which are the GUIs that allow the users to connect with each other and the database and battle each other.

Our persistence layer has account creation, which allows the user to save and query account information. We also have the Matchmaking requests module, which connects our users to other users via the creation of game sessions. We also have the session update handler, which handles the session updates between two users.

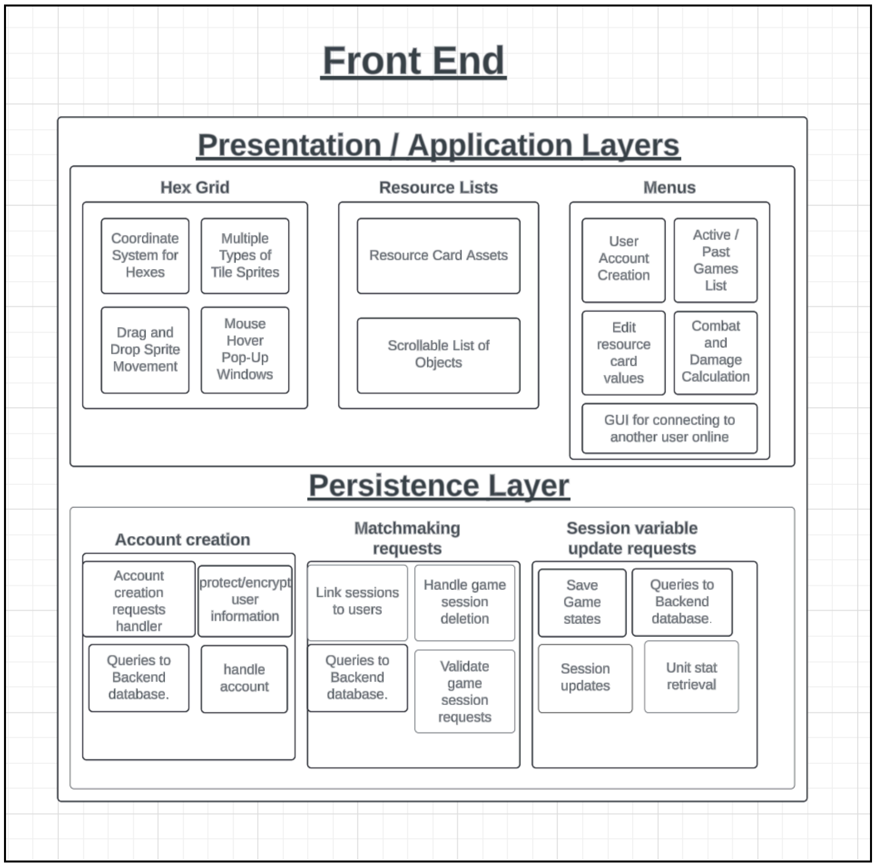

Front End
Presentation / Application Layers
Hex Grid
Coordinate System for Hexes
Multiple Types of Tile Sprites
Drag and Drop Sprite Movement
Mouse Hover Pop-Up Windows
Resource Lists
Resource Card Assets
Scrollable List of Objects
Menus
User Account Creation
Active / Past Games List
Edit resource card values
Combat and Damage Calculation
GUI for connecting to another user online
Persistence Layer
Account creation
Account creation requests handler
protect/encrypt user information
Queries to Backend database.
handle account
Matchmaking requests
Link sessions to users
Handle game session deletion
Queries to Backend database.
Validate game session requests
Session variable update requests
Save Game states
Queries to Backend database.
Session updates
Unit stat retrieval

Figure 7 - Frontend Architectural Layers

Next, we have the backend design, as seen below, which is a model, view, and controller architecture. The models, Gamefile and Account, represent all data of the users and sessions that the users have. The Views represent the ability of the system to not only create game sessions but also find games, reset games, and serve as helper functions and features to the system. The Controllers handle session updates and other similar day-to-day or turn-to-turn functionality.

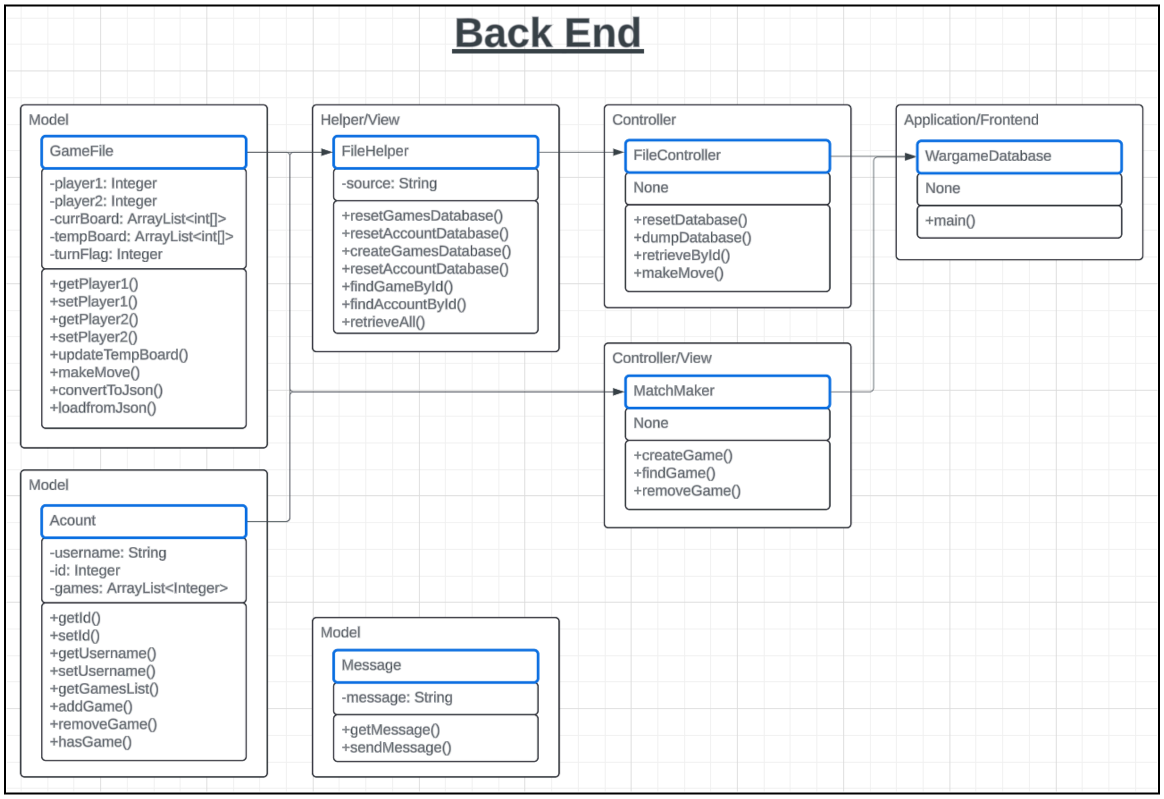


Figure 8 - Backend Layers Diagram

4.3.3. Functionality

An AFROTC cadet will be able to connect to the web-hosted game via any device that supports the minimum screen size for the game (i.e., a desktop, laptop, or tablet). A player will be able to send HTTP requests via button presses and other user input while playing the game. These HTTP requests will be sent to the backend server, currently hosted on a VM on the University's campus. This server will parse JSON data sent from the front-end and then, if required, send a request to a database containing information about users, game sessions, and other game data. The Java server will then send the requested information to the front-end Game Maker system to display, where the cycle may repeat. Below is a diagram that displays the possible navigations a user can take through our application.

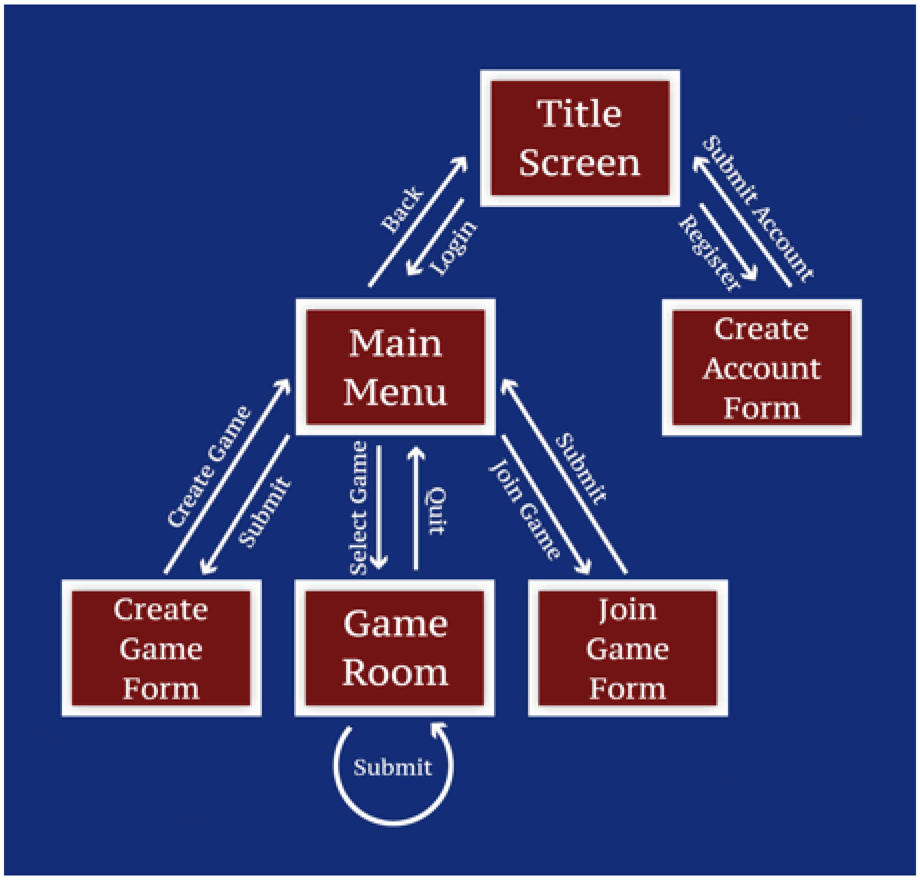
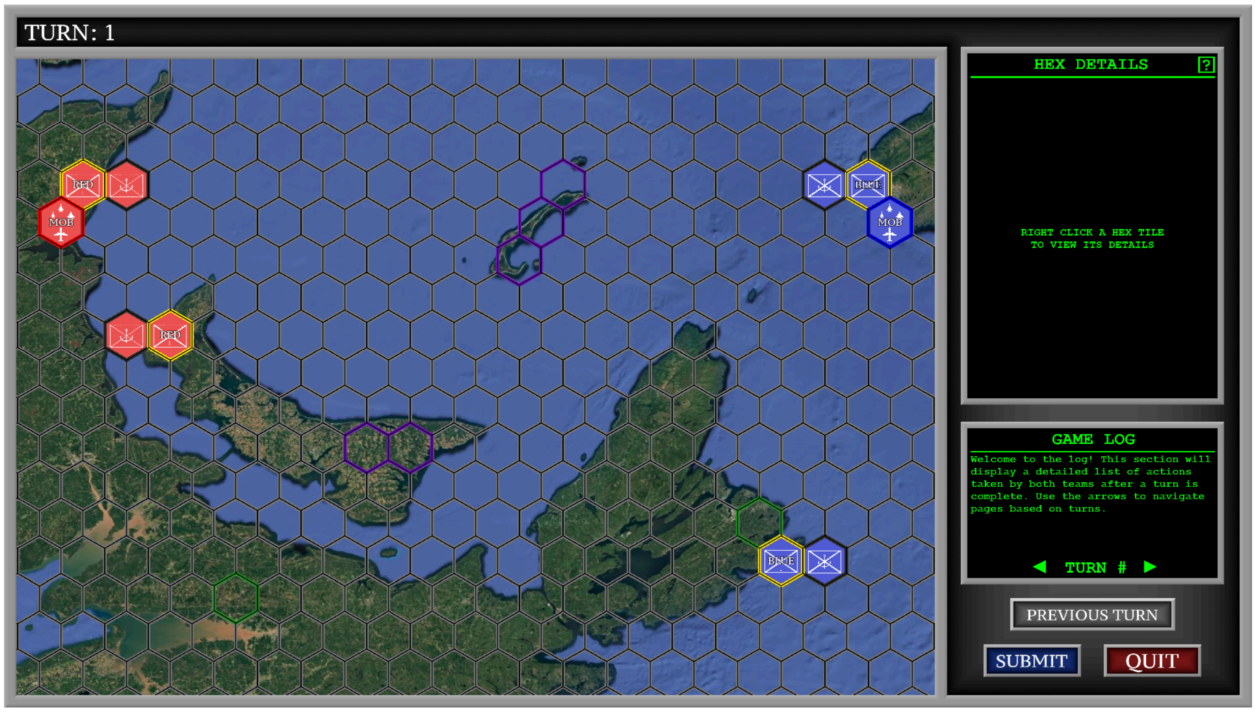


Figure 9 - Usage Flow Diagram

Once in the Game Room (see diagram above), the user is presented with many possible options to take. Entering any game room will present the same window and map for the user to play on, as seen in the figure below.

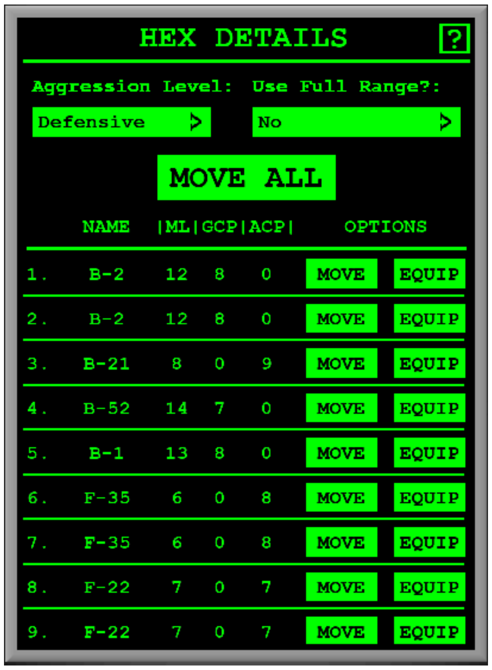


RIGHT CLICK A HEX TILE
TO VIEW ITS DETAILS

Welcome to the log! This section will display a detailed list of actions taken by both teams after a turn is complete. Use the arrows to navigate pages based on turns.

Figure 10 - Main Game Screen

One of the key functions of the game is the ability to right-click on a tile to view its details. When a tile is right-clicked, the Hex Details window on the right side of the screen becomes populated with some options: an Aggression Level dropdown, a Use Full Range option, a Move All button, and a list of the assets located within that tile, as shown in the diagram below.



Aggression Level: Defensive

Use Full Range?: No

	NAME	ML	GCP	ACP	OPTIONS	
1.	B-2	12	8	0	MOVE	EQUIP
2.	B-2	12	8	0	MOVE	EQUIP
3.	B-21	8	0	9	MOVE	EQUIP
4.	B-52	14	7	0	MOVE	EQUIP
5.	B-1	13	8	0	MOVE	EQUIP
6.	F-35	6	0	8	MOVE	EQUIP
7.	F-35	6	0	8	MOVE	EQUIP
8.	F-22	7	0	7	MOVE	EQUIP
9.	F-22	7	0	7	MOVE	EQUIP

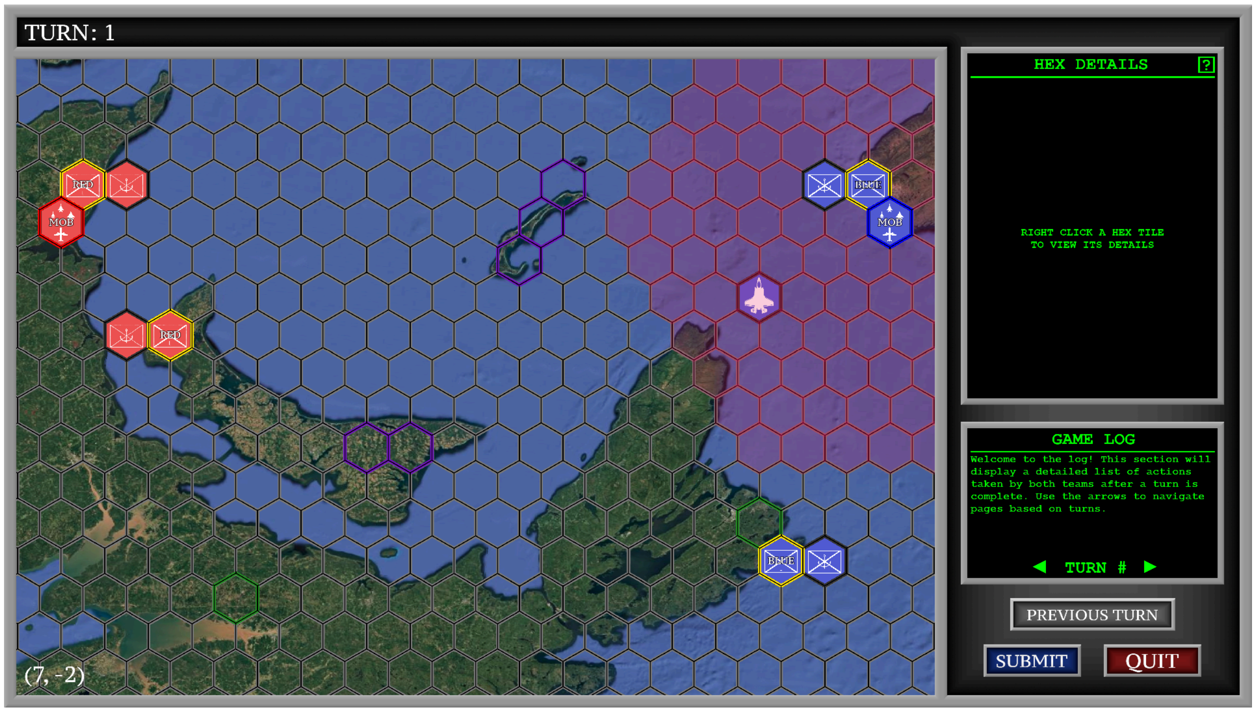
Figure 11 - Hex Details Window

For the aggression level dropdown menu, there are 3 options: Defensive, Conservative, and Aggressive. The option selected will modify how combat is assigned. If a group of assets in a single tile (a squadron) is assigned the defensive aggression level, those assets will only engage in combat if they are attacked. If the group is assigned the conservative aggression level, they will attack the smallest and/or weakest opposing squadron. Lastly, if the group is assigned the aggressive aggression level, they will attack the largest and/or strongest opposing squadron.

For the “use full range” dropdown, the user can select yes or no. If yes is selected, the assets in the tile will try to fire their missiles at the furthest opposing assets still within range of the missile. If no is selected, the assets in the tile will fire their missiles at whatever opposing assets they would fight according to the selected aggression level.

The move all button allows the user to move all of the assets in the tile as a squadron, excluding any assets with a movement limit of 0 (meaning they are immobile). The movement range of the squadron is limited by whatever the lowest movement limit score is among assets in the squadron.

For each asset listed in the tile, there are two buttons present: Move and Equip. If the move button is selected, the user is prompted to move that individual asset anywhere within its movement limit score. A red highlight is applied to all possible tiles for the asset to be moved to, and the asset currently being moved is semi-transparently displayed under the cursor (see figure below). The user is prevented from dragging the selected tile outside of the range of the highlight.

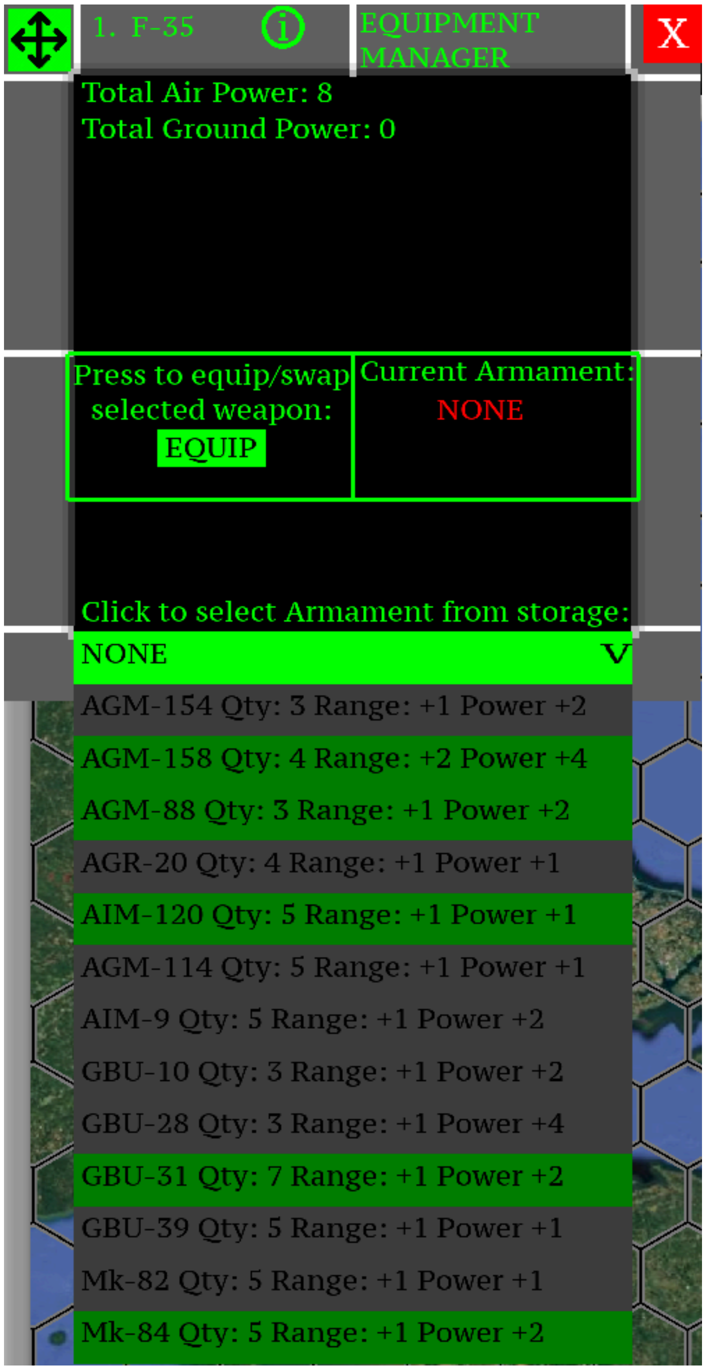


RIGHT CLICK A HEX TILE TO VIEW ITS DETAILS

Welcome to the log! This section will display a detailed list of actions taken by both teams after a turn is complete. Use the arrows to navigate pages based on turns.

Figure 12 - Movement Highlight

If the user selects the equip button, an equipment manager window is then displayed. This window gives the user current stats for the selected asset in regard to combat power. It also has a dropdown with a list of possible armaments to be equipped to the selected asset. The user will be warned if the asset is incompatible, otherwise the armament will be equipped and ready for use in combat (see diagram below).



1. F-35

EQUIPMENT MANAGER

X

Total Air Power: 8
Total Ground Power: 0

Press to equip/swap selected weapon:
EQUIP

Current Armament:
NONE

Click to select Armament from storage:

NONE

AGM-154 Qty: 3 Range: +1 Power +2
AGM-158 Qty: 4 Range: +2 Power +4
AGM-88 Qty: 3 Range: +1 Power +2
AGR-20 Qty: 4 Range: +1 Power +1
AIM-120 Qty: 5 Range: +1 Power +1
AGM-114 Qty: 5 Range: +1 Power +1
AIM-9 Qty: 5 Range: +1 Power +2
GBU-10 Qty: 3 Range: +1 Power +2
GBU-28 Qty: 3 Range: +1 Power +4
GBU-31 Qty: 7 Range: +1 Power +2
GBU-39 Qty: 5 Range: +1 Power +1
Mk-82 Qty: 5 Range: +1 Power +1
Mk-84 Qty: 5 Range: +1 Power +2

Figure 13 - Equipment Manager Window

Lastly, there is a ? button in the top-right of the Hex Details window. Clicking this button will provide the user with an explanation of the options and stats presented in the Hex Details window. Clicking the button again will close that dialog

In addition to the Hex Details window, there is a Game Log window immediately under it. A textual description is generated of what happened during each turn of the game and is presented in the game log after each player submits and the turn is passed. The user can use arrows provided to look back at previous turns and refresh on what has happened over the course of the game.

To conclude the overview of the game room, there is a button available that says Previous Turn. Selecting this button will play an animation of any combat that occurred the previous turn. This button does nothing if it is the first turn of the game or if no assets engaged in combat during the previous turn. Below you can see a still image of the combat animations playing.

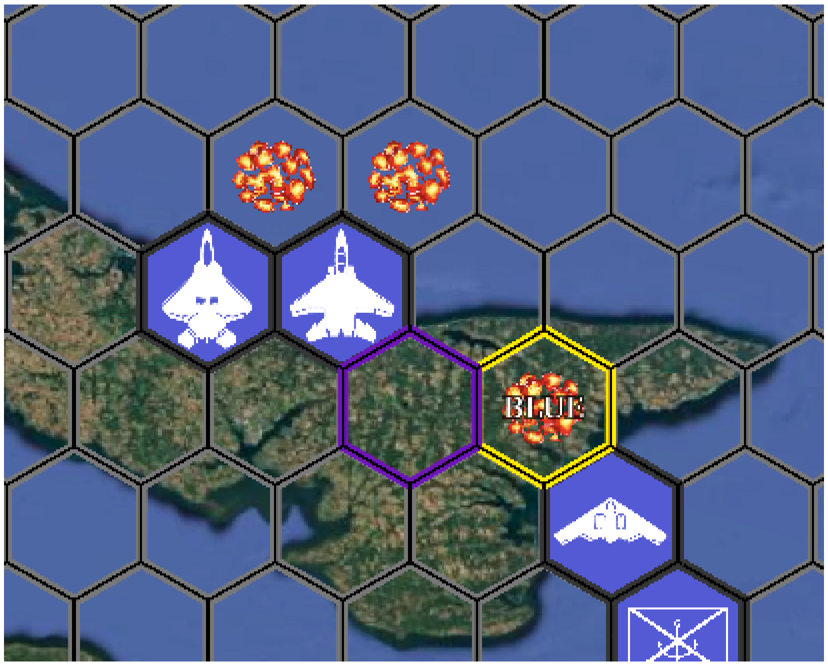


Figure 14 - Combat Animations

4.4. Technology Considerations

We are making use of proven and reliable software tools to build our design. As mentioned above, these tools fulfill the purpose of our 2 major modules; the client-side GUI and the server-side, which stores gameplay data and allows for multiplayer functionality.

On our front end, we are using a 2D game development engine called Game Maker Studio 2. This is a very powerful piece of software that gives us tools to quickly create 2D graphics and game pieces that a user can move around. This tool is also very popular in the game development space, which means that there is a wealth of documentation and tutorials available to teach us how to work with its unique systems. While GM Studio 2 is very powerful, it does have some drawbacks. First of all, it uses a special programming language called GML (Game Maker Language). While it is similar in structure to Java, the syntax and built-in functions are very different. GM Studio 2 also makes use of unique block-based programming conventions for creating objects and graphics. This means that we had to do a lot of learning about GM Studio exclusively, which is difficult to transfer to other programs. Despite this significant weakness, the power provided by the engine is worth the trade-off. This is mostly due to the ease with which we can create graphical assets, cameras, and all of the other necessary aspects of the GUI.

Alternatives for the front end include just using Java features in an IDE like Eclipse or using a separate game engine like Unity. We have discussed both of these alternatives, but they have far more weaknesses than GM Studio when applied to our project. Using straight Java deprives us of many of the graphics tools that make 2D game development in GM studio much easier. From the perspective of using Unity, it provides us with many necessary tools for game development such as a physics engine and graphical tools for things like shadows and light sources as well as textures. However, all of these systems are optimized for a 3 dimensional perspective, and many workarounds are necessary for clean 2D functionality. Because of this analysis, we believe that GM Studio 2 is the best tool for us to use on the front end.

4.5. Design Analysis

So far, we have built a basic Game Maker application that includes a hex map with hex coordinates, and sprites that are able to be moved around the map. Our game also implements a camera that will follow the user's mouse to adapt to different screen sizes of the game's players. We also have a start menu in which a user is able to send a login request to our server. We have set up a Ubuntu Red Hat Linux 2204 server in which a spring-boot application is currently serving our first set of HTTP requests. The server is also connected to a Maria database, which is also successfully setup. All of the coded implementation is currently on our team's Gitlab alongside issue boards.

Our plans for future design and implementation are shown as follows, also mentioned in previous sections of this Design Document: fully implement User-to-User Communication, implement Core Game Logic from the rule book of the physical board game, import and develop functionality for game assets, package and deploy the Web-Application, and introduce play testing with AFROTC Cadets

The current work done so far significantly implies the overall project's feasibility. The full implementation of the project is a function of time. The game's logic and player constraints complexity are the main factors in increasing the time for implementation.

5. Testing

5.1. Unit Testing

The vast majority of our unit testing occurs on the backend of our project. We have many API calls that we have created for various functions. A list of those calls includes:

- POST /game/create
- PUT /game/join
- PUT /game/update
- GET /game/retrieve
- DELETE /game/delete_game
- POST /account/create
- GET /account/login

To test each of these requests before connecting them to the frontend of our project, we have used Postman to ensure that the calls work and the bodies / headers are formatted properly. We have a shared Postman workspace that we use for storing our requests to test the backend server. We also have a mock server on Postman that can be used to store responses from the backend and test the frontend requests using those stored responses by making calls to the mock server.

5.2. Interface Testing

Our frontend is developed in GameMaker Studio using their proprietary GameMaker Language (GML). Our team hasn't implemented any frontend testing tools that are compatible with GML at the moment, and have performed all of the interface testing manually so far. This includes dragging assets and ensuring intended behavior. For frontend-to-backend testing, all of our implementation up until this point has been based on buttons, so that functionality has also been tested manually by pressing the buttons and ensuring intended behavior.

5.3. Integration Testing

The primary integration path in our project is connecting the GameMaker frontend with the Spring Boot backend (hosted on an Iowa State provided virtual machine). Much of this testing is also being performed manually at the moment. To ensure account creation and login is functional, we launch the app and create a new account, and sign in. To ensure game assets are being saved to the server properly, we launch the app, move the assets from their original location, and submit them to the server. To ensure game asset coordinates are being served properly from the server, we run the app, move the assets, and update the board from the server.

5.4. System Testing

Our system testing will look very similar to our integration testing. Given our project is a video game, play testing makes the most sense. With a video game, user experience and game flow is just as important as making sure logic behaves as expected. This means that it is important that there be human interaction with the application during testing, as automated testing can’t analyze user experience like a human could.

5.5. Regression Testing

Our team has opted to develop in such a way that we lay the framework for the entire project before moving forward in detailed development for any specific component of the game. With this, our additions inherently build upon previous functionality. This makes regression testing easy, as we are naturally performing regression testing as we perform integration and system testing, as new functionality would fail if old functionality fails. As long as our integration and system testing continue to yield successful tests and expected behavior, we can be confident that our previous implementations remain functional.

5.6. Acceptance Testing

To ensure our project meets the non-functional requirements in addition to the functional requirements, our team gathered a small group of AFROTC cadets to volunteer as play testers the first week of December. As mentioned in system testing, human interaction in testing a game is valuable in many ways that automated testing can't be. With this, we wanted to make sure our intended users were given a chance to use our software and provide us with feedback during the development cycle. The small group of play testers were quickly able to identify some new bugs that hadn't been discovered by our team and were able to provide us with feedback on the user experience and gave us some ideas for updates to make the game more user friendly. This led to our team creating a help menu to answer some of the common questions.

5.7. Results

Overall, the results of our testing show our application to be sound. We have a suite of unit and integration tests that run whenever the backend code is compiled, which currently displays no errors when building the final version of our server code. We also have a CI/CD pipeline for our Git repository that runs additional tests, as well as builds and deploys the code. The tests in the CI/CD pipeline are also fully passing for the backend code.

As for the frontend, we had to primarily rely upon the frontend development team to perform manual tests over the course of the semester, as new functionality was added to the game. We were unfortunately unable to find a means of automating unit tests for GameMaker, meaning we had to do all UI testing by hand. We did however manage to get a team of a few AFROTC cadets together to perform some play testing for us. They were able to quickly identify a few bugs that had gotten past our team and provide us with some valuable feedback of improvements we could make to our application that they would like to see. This feedback was very valuable to us, as they are our intended user group.

6. Implementation

With some minor concessions, we were able to fully implement the game this semester as we had hoped to. We were able to fully recreate the Global War 2030 board game as created by ISU AFROTC as a video game. In addition to recreating the visual pieces of the game that our client and the cadets will expect to see, we were able to make use of the advantages of a video game to greatly simplify how the game is played by handling all combat calculations, turn event tracking, and storage of board states to eliminate setup time.

We were also able to implement some new functionality that also makes the game easier to learn and play for our users. One example of this is the aggression level options that the user is presented with when making moves. In the physical game, users have to move each piece and determine whether or not that piece is going to engage in combat that turn. Below you will find the combat tracking sheet currently used by cadets to track asset locations, armaments, and combat. As you can see, it is quite tedious to fill out, and then perform the combat calculations, and repeat that process up to 10 times for each turn of the game.

A few of the concessions we had to make are not allowing users to create custom maps, not implementing a special armament store with instructor-granted points, and removing compatibility for mobile devices. Not moving forward with custom maps and the special armament store were dropped due to lack of time for implementation, whereas compatibility for mobile devices was dropped due to the infrastructure of our backend hosting and the way GameMaker manages web-based applications being incompatible with one another.

Despite those few concessions, none took away from the original integrity of the game and were mostly stretch goals that we would have liked for increasing the quality of life for our client and users. Overall, our user testers seemed immediately familiar with how to operate our game which speaks volumes to the level of accuracy our team had in recreating the original board game, while also making vast improvements for ease of use. During prep week of this year, our team will make the final steps of deploying the backend code to a Raspberry Pi, which we will then install in the Armory for our client to use to host our application. We will also host a single-page web-application from that Raspberry Pi for users to download the correct installer for their operating system, completing the deployment of our application and concluding our senior design project.

7. Professional Responsibility

Note: This discussion is with respect to the paper titled "Contextualizing Professionalism in Capstone Projects Using the IDEALS Professional Responsibility Assessment", International Journal of Engineering Education Vol. 28, No. 2, pp. 416–424, 2012

7.1. Areas of Responsibility

Area of Responsibility	IEEE Description	NSPE Differences
Work Competence	Both emphasize the importance of work competence and understanding in regards to technology.	The NSPE Code of Ethics applies to the engineering world as a whole, including many different areas of engineering work
Financial Responsibility	The IEEE emphasizes honesty and realism in all professional dealings, including financial matters.	While the NSPE Code of Ethics highlights the importance of technology advancements, it focuses more on broader engineering principles.
Communication Honesty	Both emphasize the importance of truthfulness and objectiveness in communications with others, especially customers.	Highlights honest communication in all professional interactions, while IEEE emphasizes public transparency.
Health, Safety, and Well-being	Both emphasize the importance of prioritizing safety, health, and welfare.	NSPE prioritizes public safety, health, and welfare in all engineering practices, while IEEE emphasizes innovative public welfare concerns.
Property Ownership	While the IEEE focuses on fairness and	NSPE requires engineers to safeguard confidential

	non-discrimination, it also highlights the importance of respecting other individuals.	information, while IEEE emphasizes ‘respect’ for intellectual rights.
Sustainability	Both emphasize the importance of environmental sustainability. The IEEE Code encourages realism in handling environmental issues.	The NSPE promotes sustainable engineering practices while the IEE code relies upon innovation to get things done sustainably.
Social Responsibility	Both emphasize the importance of avoiding harm to others. The IEEE principle states how important it is to conduct yourself honorably.	NSPE emphasizes honorable and ethical conduct in all domains while IEEE focuses on mainly Electrical and hardware responsibilities.

Table 3 - Areas of Responsibility

7.2. Project Specific Professional Responsibility Areas

1. Work Competence
   - Applies
   - Current Performance: High
   - Designing a war gaming simulator for Air Force ROTC is complicated, and it's important that team members have the necessary competence in software development, game design, and requirements engineering to complete the task.
2. Financial Responsibility
   - Applies
   - Current Performance: Medium

  - While our wargaming simulator may not involve financial transactions, there are still financial responsibilities related to the deployment of our project. We want the project to be free and able to run on minimal hardware in ISU.

3. Communication Honesty
   - Applies
   - Current Performance: High
   - Clear and honest communication is essential for conveying project progress, challenges, and requirements to ISU AFROTC. Reporting work truthfully creates trust and helps collaboration among all people involved.
4. Property Ownership
   - Applies
   - Current Performance: High
   - We must respect the intellectual property, ideas, and information of clients and the United States Air Force. This includes getting permissions for third-party assets and in-game sprites, respecting intellectual property rights, safeguarding confidential information related to the project, and encrypting the data of our users.
5. Sustainability
   - Partially Applies
   - Current Performance: Medium
   - Our project may not have direct environmental impacts, but minimizing energy consumption in software development processes or considering the sustainability and environmental impact of hardware needed for the game is a good idea.

6. Social Responsibility
   - Applies
   - Current Performance: Medium
   - By teaching future generations of USAF Officers on how to better implement the ACE combat strategy, we directly contribute to the benefit of society and garner a community of Cadets around the Wargaming simulator while the Cadets use the Software.

7.3. Most Applicable Professional Responsibility Area

The most applicable area of responsibility in the context of our Wargaming Simulator is arguably Work Competence. This is because the success of our project relies on the competence and expertise of the four of us developing the project. Additionally, developing a software application, especially one with complex functionalities like a war gaming simulator, requires a high level of technical expertise in software development, and Domain knowledge. We as a team also need to ensure the quality of our project, since the reliability of our game is vital to the success of Air Force ROTC.

8. Closing Material

8.1. Discussion

While our team can't make any claims as to the results of our testing, as it hasn't been completed, we are confident in the effectiveness of our proposed solution. At the core of our client's problem is an ineffective means to properly educate cadets on an essential concept within the United States Air Force for combat engagement. What our client needs is a solution that will more efficiently and more effectively familiarize cadets with ACE strategy than the board game that is currently used. Our user interfaces will guide players through all decision making processes, perform the necessary calculations for them, store all game moves for them, and has minimal setup compared to a board game. The capability to perform calculations, store game moves, and minimal setup time all tackle the efficiency requirement of our client's problem. The user interfaces to guide players through all decision making processes tackle the effectiveness requirement of our client's problem, as less rules needing to be learned means more attention to be directed at learning strategy.

8.2. Conclusion

Our project has reached a completed and deployable state. While it has not received the attention to user testing that was initially planned, the functionality of the game is fully instantiated. Players can create and join games, make moves with complicated combat controls, receive both visual and text feedback for moves, and have games decided by win conditions. The completion of the project is due to the management style we employed where the framework was focused on before the functions.

8.3. References

[1] ISU AFROTC, *Global War 2030 AFROTC Educational Wargame Rules*. Ames, IA: Iowa State University, Feb 2024.

[2] ISU AFROTC, *AFROTCI 36-2011 Vol 3 (Cadet Operations)*. Ames, IA: Iowa State University, Jun 2023.

[3] ISU AFROTC, *AFROTCI 36-2011 Vol 1 (Excerpt Objective 10)*. Ames, IA: Iowa State University, Feb 2024.

[4] ISU AFROTC, *AFDN 1-21 ACE*. Ames, IA: Iowa State University, Aug 2022.

[5] GameMaker Manual, https://manual.gamemaker.io/ (accessed Apr 20, 2024).

[6] U. A. Force, “Quick links,” U.S. Air Force Doctrine > Home, https://www.doctrine.af.mil/ (accessed Apr 20, 2024).

9. Team

9.1. Team Members

- Reid Coates
- Jack Kelley
- Alexander Hassan
- Luke Muilenburg

9.2. Required Skills for the Project

- Experience with GameMaker Studio
- Experience with Java Spring Boot
- Experience working with Iowa State virtual machines
- Experience creating HTTP requests and custom API calls
- Experience with video game user interfaces
- Experience with strategy games

9.3. Skills Covered by the Team

- **Reid Coates**
    - Experience with GameMaker Studio
    - Experience with Java Spring Boot
    - Experience working with Iowa State virtual machines
    - Experience creating HTTP requests and custom API calls
    - Experience with video game user interfaces
    - Experience with strategy games
- **Jack Kelley**
    - Experience with GameMaker Studio
    - Experience working with Iowa State virtual machines
    - Experience creating HTTP requests and custom API calls
    - Experience with video game user interfaces
    - Experience with strategy games
- **Alexander Hassan**

    - Experience with Java Spring Boot
    - Experience working with Iowa State virtual machines
    - Experience creating HTTP requests and custom API calls
    - Experience with video game user interfaces
    - Experience with strategy games
- **Luke Muilenburg**
    - Experience with Java Spring Boot
    - Experience working with Iowa State virtual machines
    - Experience creating HTTP requests and custom API calls
    - Experience with video game user interfaces
    - Experience with strategy games

9.4. Project Management Style Adopted by the Team

Note: The following is directly quoted from Section 3.1 of this document.

Our team is adopting a blend of waterfall and agile development. This includes Weekly SCRUM Sprints, Monday team meetings, Thursday check-in/advisor meetings, and Flowchart tasks with color codes for sprints. This is because our project requirements are largely not fixed, and thus, our team must be able to adapt to ever-changing requirements. We have planned multiple sprints out from week one, so we did not adopt a pure agile management style, but we still have standups and client-involvement, thus, a blend of management styles was achieved. Our Team tracks progress via GitLab and Sprint plans provided in section 3.2.

9.5. Initial Project Management Roles

- **Reid Coates** - Client Coordination and Backend Development Lead
- **Jack Kelley** - Organization Lead and Frontend Development
- **Alexander Hassan** - Testing Lead and Frontend Development
- **Luke Muilenburg** - Frontend Development Lead

9.6. Team Contract

Note: The following is copied directly from our original team contract, retaining the original styling and formatting.

Team Members:
1) Reid Coates
2) Jack Kelley
3) Alexander Hassan
4) Luke Muilenburg

Team Procedures

1. Day, time, and location (face-to-face or virtual) for regular team meetings:

   Day: *Monday*
   Time: *1pm-2pm*
   Location: *MASC (5th floor Memorial Union)*

2. Preferred method of communication updates, reminders, issues, and scheduling (e.g., e-mail, phone, app, face-to-face):

   Our preferred method of communication within the team (Reid, Jack, Alex, and Luke) is our private Discord server. This will allow us to communicate to the entire group simultaneously, as well as keep a log of all conversations and decisions that were made.
   Our preferred method of communication with our advisor and client will be email.
   Reid will be our client and advisor contact point, making it easier for our team to keep track of what messages have been relayed to both our advisor and client (no duplicate communications), as well as make it easier for both our advisor and client to contact us. Instead of managing multiple conversations, our advisor and client will know specifically who to contact.

3. Decision-making policy (e.g., consensus, majority vote):

   All decisions will first be made by consensus, with the opposing views presenting their cases until a final plan is agreed upon. If an agreement cannot be made within our allotted meeting time, the decision will go to a majority vote. Given that we have 4 members, a split decision will be resolved by consulting our advisor or another mentor.

4. Procedures for record keeping (i.e., who will keep meeting minutes, how will minutes be shared/archived):

   Our meeting minutes will be tracked and recorded in a document that is in our shared Google Drive. Luke will be responsible for tracking and *recording the minutes for our weekly meetings, as well as documenting any major decisions or plans developed during that week.*

Participation Expectations

1. Expected individual attendance, punctuality, and participation at all team meetings:

   Each team member is expected to be present, on time, and an active participant for all meetings.
   Team members are expected to communicate any absences, conflicts, or tardiness via the team Discord prior to the scheduled meeting time, giving teammates ample notice of said absence, conflict, or tardiness.
   Team members are expected to actively participate during meetings: Providing feedback during discussions, updating teams on progress of assigned tasks, and asking questions or for help when necessary.

2. Expected level of responsibility for fulfilling team assignments, timelines, and deadlines:

   Each team member is expected to fulfill their assigned tasks on-time, with full effort and quality.
   If at any point an extension is needed or a timeline won't be met, the teammate in question should communicate the need for an extension with the group as soon as possible.
   Team members should ask for assistance whenever needed to avoid missing deadlines and disrupting the development process for the other members of the group.

3. Expected level of communication with other team members:

   At a minimum, each team member should provide the group with a weekly report of progress made, problems encountered, and an updated timeline on the completion of the tasks assigned to that team member.
   As mentioned previously, any problems or areas where a team member is in need of help should be communicated to the group as soon as possible to provide significant time to develop a plan for resolving the problem or readjusting the direction of the project if necessary.

4. Expected level of commitment to team decisions and tasks:

Team members are expected to fully commit to final decisions made by the team. Unless otherwise communicated or resolved, tasks are to be devised and assigned as a team and are to be completed as agreed upon by the group.

Team members are also expected to completed their tasks within the agreed upon amount of time, unless otherwise communicated and extended, and complete those tasks to the best of their ability. If a team member feels ill-equipped to successfully complete a task assigned to them, that team member should seek help from the group, and potentially reassign the task to a more suitable team member.

Leadership

1. Leadership roles for each team member (e.g., team organization, client interaction, individual component design, testing, etc.):

   Reid: Client interaction, Back-end design
   Jack: SCRUM master, Front-end design
   Luke: Team organization, Front-end design
   Alex: Testing, Back-end design

2. Strategies for supporting and guiding the work of all team members:

   Meeting notes will be taken, and any commitments will be followed. At the beginning of each meeting, we will provide scrum statuses to each other and make sure any obstacles are being addressed.

3. Strategies for recognizing the contributions of all team members:

   Our team Trello board will keep track of what tasks are currently being worked on, as well as what tasks have been completed. We can use this and the team member who was assigned to a given task to determine where the other members are in the development process and what all they have contributed to the project.

Collaboration and Inclusion

1. Describe the skills, expertise, and unique perspectives each team member brings to the team.

   Jack: Frontend development in Android Studio, game asset creation, Software Engineering internship and class experience with networking protocols and CRUD requests. Personal Java game development and minimal

GameMaker Studio experience for game development. Various leadership roles to aid in organization responsibility and use of Trello for SCRUM mastering.

Reid: Backend Java development using Springboot, understanding and teaching experience with Agile Combat Employment concepts, experience with interacting and operating within US Air Force environments, project management, and also wireframing and UI experience with user testing and interaction.

Alex: Backend Development in Spring Boot and Maven. Software Engineering Co-op with experience in Unit testing, traceability of requirements, and C.

Luke: Java backend development experience, software & computer engineering internship experience, familiarity with various wargaming systems, game development experience with Unity game engine

2. Strategies for encouraging and supporting contributions and ideas from all team members:

   Making our group decisions by consensus ensures all members will have their ideas heard and give everyone a chance to contribute to the various issues we will face throughout the development of our project in an open and fair setting. Also, assigning specific roles for each member ensures that all members feel as though they are making a significant contribution to the project because everyone has at least one area that they are solely responsible for.

3. Procedures for identifying and resolving collaboration or inclusion issues (e.g., how will a team member inform the team that the team environment is obstructing their opportunity or ability to contribute?)

   If at any point a team member feels as though they aren't being permitted to collaborate or are being excluded from the group, that team member is responsible for bringing the issue to the group by means of a respectful conversation to be had during our weekly meeting. If a team member doesn't feel as though their concerns are being heard or feel uncomfortable discussing the issue with the group, that team member should consult the advisor who can mediate conversation between the excluded member and the rest of the group until the problem is resolved.

Goal-Setting, Planning, and Execution

1. Team goals for this semester:
    - *Thoroughly understand and design a wargaming simulator.*
    - *Understand Agile Combat Employment.*
    - *Understand Air Force Air Assets and their functions.*
    - *Create a plan for the software architecture based on extensibility and usability.*
    - *Work with the Client to create an in-depth wire frame of our application.*
    - *Learn how to effectively program in Game Maker Language (GML).*

2. Strategies for planning and assigning individual and team work:
    - *Our team will use a Trello board as well as Github to visualize tasks that need to be completed or assigned.*
    - *Our Scrum Master will primarily be responsible for making sure every team member has an appropriate amount of work and will help each member identify and solve obstacles.*
    - *Any member can request the creation of a task that needs to be completed, and the team, especially the scrum master will approve/modify and assign the task.*
3. Strategies for keeping on task:
    - *Our meetings will follow a consistent structure to ensure all necessary discussions had and developments are made*
    - *Meetings will be kept short and to the point, reducing the amount of attention required*

Consequences for Not Adhering to Team Contract

1. How will you handle infractions of any of the obligations of this team contract?

    Due to the importance of getting off to a good start with our project this first semester, we need to make sure that all team members are meeting their obligations outlined in this document. It is reasonable for us to excuse occasional missed meetings due to extenuating circumstances. If a team member is struggling to meet their obligations, we will meet as a team to see what we can do to help him.

2. What will your team do if the infractions continue?

    If a pattern of missed meetings and incomplete work develops, we plan to bring it to the attention of our instructors as soon as possible to collaborate to find a solution to the problem. This is so that we can avoid a *backlog of incomplete work that others rely on developing, which would result in a lot of lost time down the road. If issues with a teammate develop, we should also be prepared to communicate transparently with the stakeholders in our project.*

****

a) *I participated in formulating the standards, roles, and procedures as stated in this contract.*
b) *I understand that I am obligated to abide by these terms and conditions.*
c) *I understand that if I do not abide by these terms and conditions, I will suffer the consequences as stated in this contract.*

1) Luke Muilenburg	DATE: 1/29/2024
2) Jack Kelley	DATE: 1/29/2024
3) Alexander Hassan	DATE: 1/29/2024
4) Reid Coates	DATE: 1/29/2024