Atlas Systems Requirements Document

November 26th, 2025

Project Sponsor: Morgan W. Boatman - Missions and Madness, Winter Communication LLC
Faculty Mentor: Ogonna Eli
Team Members:
John Zeledon (Team Lead)
Tristen Calder (Release Manager)
Hunter Beach (Architect)
Mitchell Morris (Recorder)

Accepted as baseline requirements for the project (Date: November 25th, 2026):

For the Client: [illegible] *For the Team: Zeledon, Morris, Calder, Beach*

TABLE OF CONTENTS

1. Introduction

Our project, *Missions and Madness,* lies at the intersection of mobile, learning-based, and location-based gaming. All of these domains have been growing over the past few decades. Now, improving mobile technology can blend real-world exploration with the excitement of games. Games like *Pokémon GO* and activities like *Geocaching* have demonstrated the global potential of this industry, which attracts hundreds of millions of players worldwide and generates billions in revenue each year. Beyond entertainment, this genre has been increasingly used for education, wellness, and community engagement, showing that playful, technology-supported exploration can meaningfully connect people to their environment.

Our sponsor, Morgan Boatman, developed *Missions and Madness* as a local initiative in Flagstaff, Arizona. His goal was to encourage participants to explore their surroundings, collaborate with others, and think critically about the world around them. It began as a pen and paper project in the form of a handbook that guides players through real-world missions that blend adventure, problem-solving, and reflection. Players visit local landmarks or community spaces and answer situational awareness questions designed to promote observation and mindfulness.

The first version of *Missions and Madness* successfully provided its goal of community engagement and self-improvement. The project remains limited in scale and functionality. For starters, each game had to be specially designed by Morgan himself, which limited him to Flagstaff. Furthermore, since the game couldn't exist without Morgan, it was quite hard to spread it quickly.

The second prototype was an application worked on by last year's capstone team, *The TaskMasters*, and it provided its goal of creating an app that can follow the specified flow for the game, see **Chart 1** in our **Problem Statement**. However, it could only reach what it was designed for, a small user base, relying on local data entry and minimal interface design. To fully realize Morgan's vision of a nationwide game.

Now our team, *Atlas Systems,* will improve upon this second prototype. The redesigned system will incorporate real-time GPS tracking, dynamically generated maps and points of interest (POIs), a unified user interface, and a cloud-hosted data infrastructure. These improvements will enable the game to expand beyond Flagstaff, allowing anyone to experience adventure in their own city.

Ultimately, *Missions and Madness* seeks to foster a sense of connection, collaboration, and self-improvement by combining technology and adventure. The redesigned application will help communities rediscover their local environments, turning exploration into a social and reflective experience.

2. Problem Statement

Although *Missions and Madness* presents an innovative concept, as mentioned in the **Introduction**, the existing system faces significant barriers that prevent it from achieving its goals, including but not limited to:

- **Manual Setup:** The sponsor or facilitator manually creates mission locations, exercises, and events.
- **Limited User Experience:** There is also an unpolished and inconsistent UI, which takes away from the User Experience. There are also no tutorials to help new players.
- **Local-Only Data:** The local SQLite database restricts the scaling of the game. It is currently limited to only points in Flagstaff.
- **Lack of Mapping:** Players receive text-based instructions with no integrated map or GPS tracking. This can make it difficult to start to learn about their local environment.

While last year's prototype from Task Master was able to provide the necessary game flow seen in **Fig. 1** on the next page, its flaws made it greatly limit both player engagement and system sustainability.

In its current form, Missions and Madness cannot deliver the immersive and nationwide experience envisioned by the sponsor. A redesigned solution is necessary to improve the infrastructure, automate data retrieval, and provide a seamless user experience.

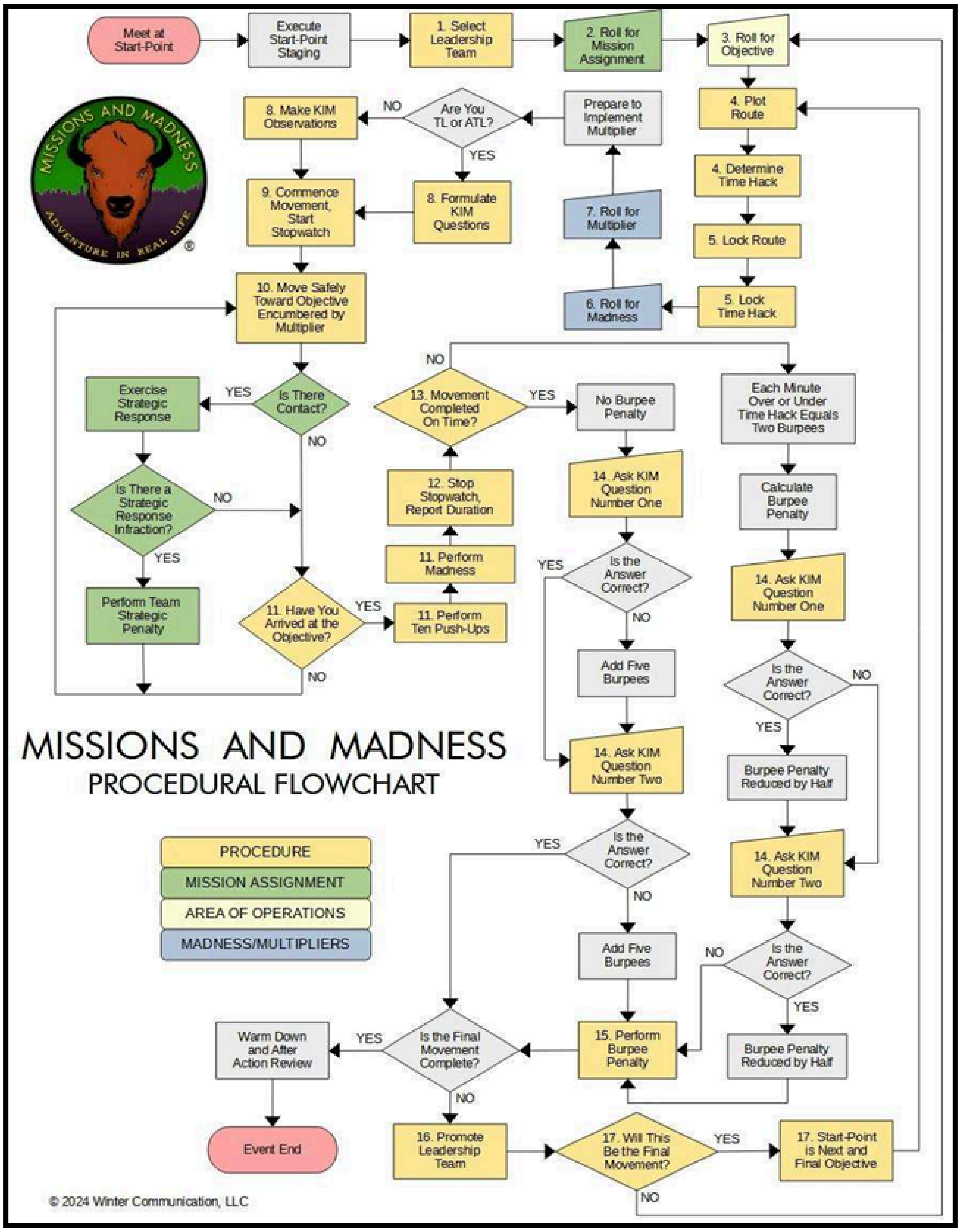

Meet at Start-Point
Execute Start-Point Staging
1. Select Leadership Team
2. Roll for Mission Assignment
3. Roll for Objective
4. Plot Route
4. Determine Time Hack
5. Lock Route
5. Lock Time Hack
6. Roll for Madness
7. Roll for Multiplier
Prepare to Implement Multiplier
Are You TL or ATL?
NO
YES
8. Make KIM Observations
8. Formulate KIM Questions
9. Commence Movement, Start Stopwatch
10. Move Safely Toward Objective Encumbered by Multiplier
Is There Contact?
Exercise Strategic Response
Is There a Strategic Response Infraction?
Perform Team Strategic Penalty
11. Have You Arrived at the Objective?
11. Perform Ten Push-Ups
11. Perform Madness
12. Stop Stopwatch, Report Duration
13. Movement Completed On Time?
No Burpee Penalty
Each Minute Over or Under Time Hack Equals Two Burpees
Calculate Burpee Penalty
14. Ask KIM Question Number One
Is the Answer Correct?
Add Five Burpees
14. Ask KIM Question Number Two
Burpee Penalty Reduced by Half
15. Perform Burpee Penalty
Is the Final Movement Complete?
Warm Down and After Action Review
Event End
16. Promote Leadership Team
17. Will This Be the Final Movement?
17. Start-Point is Next and Final Objective
MISSIONS AND MADNESS
PROCEDURAL FLOWCHART
PROCEDURE
MISSION ASSIGNMENT
AREA OF OPERATIONS
MADNESS/MULTIPLIERS
MISSIONS AND MADNESS
ADVENTURE IN REAL LIFE
© 2024 Winter Communication, LLC

Fig. 1 Missions and Madness Procedural Flowchart

Seen above is what the game is currently capable of, though we plan to improve how smoothly the game flows between phases in this year's work.

3. Solution Vision

To address the limitations of the current prototype, *Atlas Systems* will improve upon *Missions and Madness* to help it become a cross-platform mobile game that supports real-time location tracking and nationwide point-of-interest data with a unified, accessible interface. The redesigned system will expand the app's reach and usability while preserving its core purpose of encouraging exploration. Our key features are as follows:

- **Cross-Platform Deployment:** The system will be built in **Unity**, allowing one shared codebase to deploy to both Android and iOS devices.
- **Real-Time Map and GPS Integration:** Players will see their live position, destination, and route displayed on an interactive map that updates dynamically as they move through the world, using **Unity's** built-in tools and **Geoapify**.
- **Nationwide Points of Interest:** The **OverpassTurbo** and **WikiData** APIs will supply real-world POIs around the user's location, enabling the game to generate missions anywhere with sufficient local data.
- **Unified and Scalable Interface:** A consistent and polished UI using **Unity Prefabs** will ensure intuitive navigation across all devices, creating a cohesive visual experience for both new and returning players.
- **Cloud Data Synchronization:** Game progress, missions, and player data will be securely stored and synchronized through an online database. This cloud will also help support multiplayer gameplay.
- **Interactive Tutorials**: Short, guided tutorials will help onboard new players, teaching gameplay mechanics and safety reminders in a clear, non-intrusive way.
- **Expandable Architecture:** We will rework and clean up the codebase, and aim to follow modular design principles, making it easier for future teams to maintain and extend the system without rewriting core components.

This redevelopment aims to transform *Missions and Madness* into a nationwide, adaptable, beginner-friendly, and polished product ready for the app store. We aim to address all of Morgan's issues with the second prototype and remedy them with these solutions.

4. Project Requirements

This section defines the requirements for our redesigns and extensions to Morgan's *Missions and Madness.* We will start with **4.1 Domain-Level Requirements,** where we will broadly and briefly lay out the features that the user needs from a domain perspective. Then, we will offer more details in **4.2 Functional Requirements**, which specify the concrete features and behaviors the system must provide. We will express these with user stories, prototypes, and the MoSCoW method (Must Have, Should Have, Could Have, Won't Have). Finally, we will wrap up our requirements with **4.3 Performance** (Non-Functional) Requirements, which describe how efficiently and reliably the system should perform, and **4.4 Environmental Requirements,** which define the technical platforms/constraints under which the system should work.

4.1 Domain-Level Requirements

In the following table, we list out our domain requirements, which satisfy our goals from **3. Solution Vision,** along with a more thorough description of what each exactly means. They also come with an ID we will reference in later sections to show how we plan to tie them to our user stories.

Domain Requirement #	Domain Requirement	Thorough Description
DR1	Multi Platform	The system must reliably function across Android and IOS to allow the highest number of users to play. It should also not need two separate code bases, just one, which can build to both.
DR2	Real-Time Location Awareness / Tracking / Display	The system must display a live map of where the user is, an updating point of where the user is, what POI they are heading to, and a GPS between those points.
DR3	Integrated POI Database / Nationwide Coverage	The system must be able to reach a database of POIs around the user to show possible places the user can run their next mission. Furthermore, this system must work in any city at least as populated as Flagstaff to help make the game more playable nationwide.
DR4	Consistent / Polished User Interface / Usability	The system's UI must be unified across all screens and have more design than just basic Unity assets. It must dynamically scale and be intuitive to users. The main stakeholder to decide how well this section is fulfilled is Morgan. The system must be inherently understandable

DR5	Data Synchronization / Control	The system should provide a way to store user and mission data online. This should also provide a framework for connecting multiplayer games
DR6	Tutorials and Beginner-Friendly	The system should provide interactive tutorials and help features that teach players the mechanics of the game and the specificities of the exercises without overwhelming them.
DR7	Expandable / Maintainable	The system architecture must allow for easy expansion of later teams and have understandable source code so it can be maintained once we are gone.

Table 1. Our Domain Requirements are listed and explained

4.2 Functional Requirements

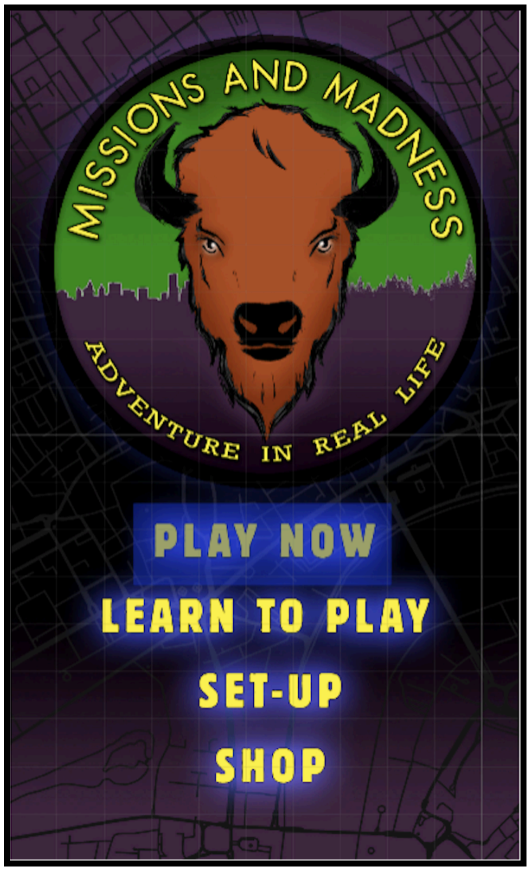
In this section, we list our functional requirements, which describe the specific features and behaviors that the system must implement to fulfill our main goals. We will express these with user stories, prototypes, and MoSCoW priorities. User stories related to a certain theme (i.e., User stories related to Unified and Responsive UIs) will be grouped. Then, if the User story is in the **MUST HAVE** section, a prototype will be provided and elaborated on. Once all user stories have been listed, it will be discussed how they all help satisfy our **Domain-Level Requirements**

4.2.1 MUST HAVE

4.2.1.1 US1 - Unified and Responsive UI

User Story / Stories:
"As a picky player, I want the interface to look consistent and scale correctly so the game feels polished and quality."
"As an uncertain player, I want the interface to clearly display when I click on buttons so that I know the game is working."
"As an impatient player, I want the interface to load fast on button presses so that I know the game is working."

Prototype(s):
(On next Page)



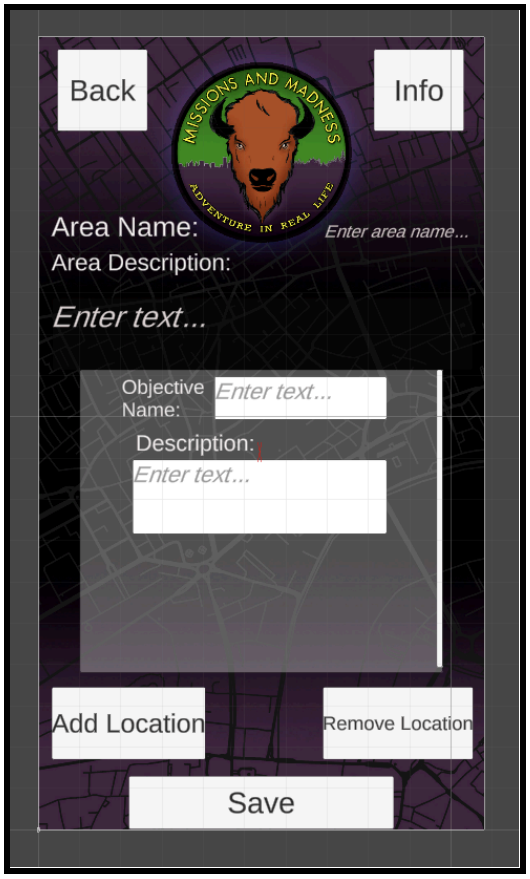


Fig. 2 Prototype of glow effect **Fig. 3 Prototype of Transparent / Multi-Sized UI**

Together, these prototypes show two ways we are actively working to improve the UI. In **Fig. 2,** we have all the buttons matching Morgan's preferred font style and a blue glow upon clicking buttons, to better achieve the UI goal of visibility, meaning players know when an action they take does something (i.e., they know when they pressed the button). Seen in **Fig. 3,** we reworked another page to have transparent UI on certain sections and different sizes of text to make it clearer where the player has to interact.

4.2.1.2 US2 - GPS and Player Tracking

User Story / Stories:
"As a directionally challenged user, I want my location and target on a live map so I do not get lost".
"As someone who has recently moved to a new town, I want to play with a map of my town so I can begin to make a mental model".

Prototype(s):

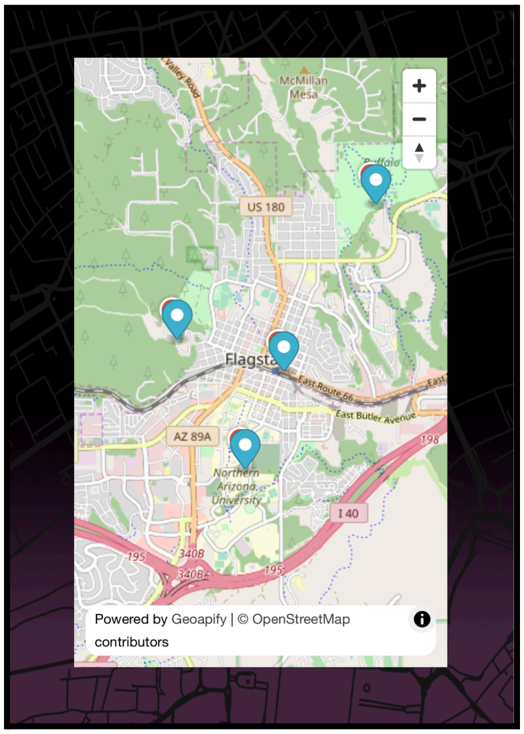
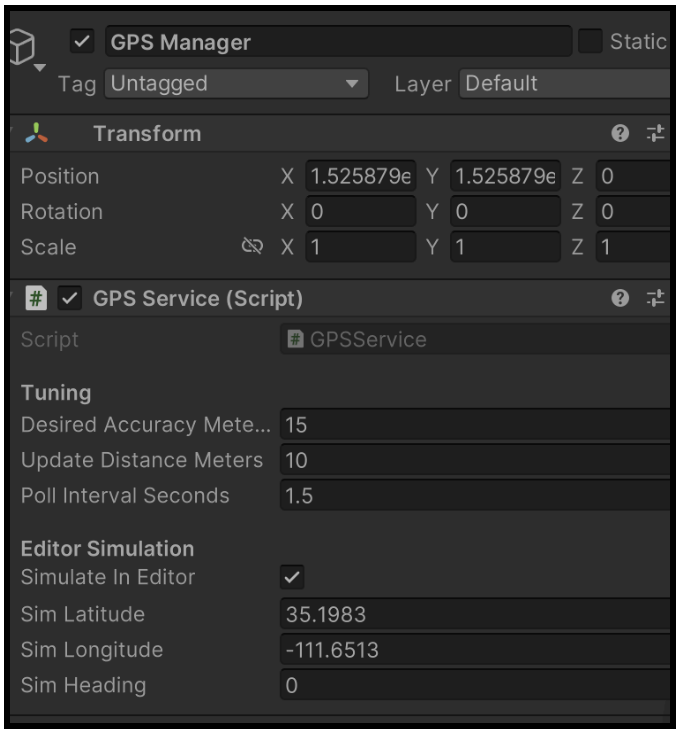


Fig. 4 Prototype of the map



GPS Manager ✓ Static
Tag Untagged Layer Default

Transform

Position	X	1.525879e	Y	1.525879e	Z	0
Rotation	X	0	Y	0	Z	0
Scale	X	1	Y	1	Z	1

GPS Service (Script) ✓

Script: GPSService

Tuning

Desired Accuracy Mete...	15
Update Distance Meters	10
Poll Interval Seconds	1.5

Editor Simulation

Simulate In Editor	✓
Sim Latitude	35.1983
Sim Longitude	-111.6513
Sim Heading	0

Fig. 5 Inspector view of our GPS

In the above, in **Fig. 4**, the map from Geoapify has not only been implemented into the game in Unity but has also been uploaded to a physical device for real testing of the program. Further combined with Unity's GPS services in **Fig. 5,** we get the player's location as coordinates to load the map. Tested on an iPhone 16 Pro Max, the prototype works as expected in regards to zooming, scrolling, and selecting points on the map. The map loads in an acceptable amount of time over both WiFi and cellular networks.

4.2.1.3 US3 - POI Database

User Story / Stories:
"As an explorer, I want to see many POIs around me so I can learn more about my town."
"As a statue enthusiast, I want only POIs which are statues, so I can play this game with my desired interests."
"As a user who lives in a big city, I want to limit my POIs so I am not overwhelmed when opening up the app."
"As a user who lives in a rural town, I want to see all possible POIs so I can find something to go to, even if they are uneventful."

Prototype(s):

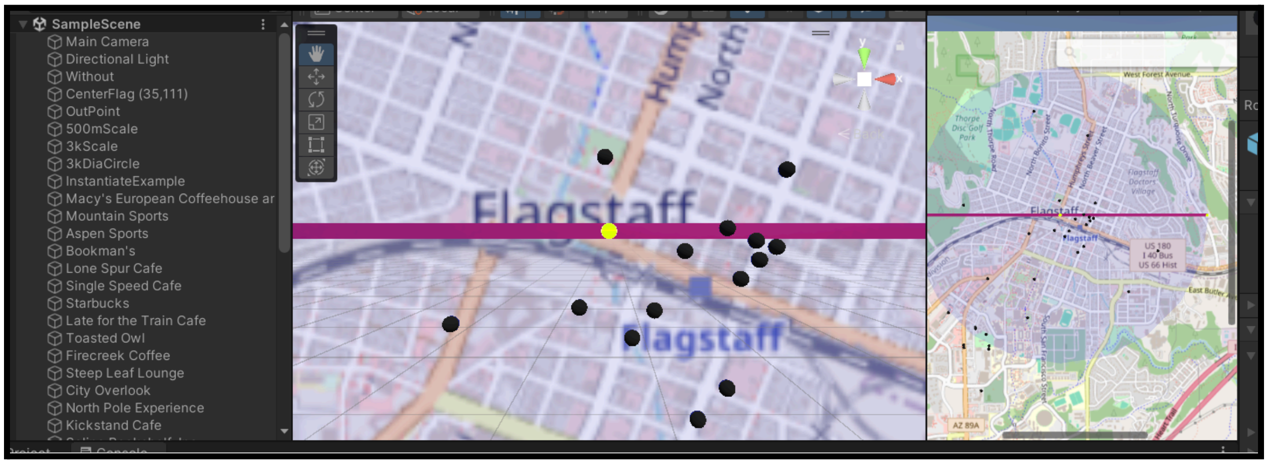


Fig. 6 A Prototype in Unity that queries for POIs and displays them on a map

As seen above in **Fig. 6,** we currently have working prototypes that can query databases. So far, we can query both the **OverPassTurbo** Database as well as the **WikiData** Database. In **Fig. 6,** we are querying Overpass Turbo for stores and landmarks in a 1500m radius (blue circle) around the center of Flagstaff (yellow circle). Then, using a scale (red line) which helps us map world co-ordinates to **Unity** co-ordinates (black circles). Our next steps will be working to implement this into the Map APIs of **4.2.1.2**

4.2.2 SHOULD HAVE

4.2.2.1 US4 - Tutorials

User Story / Stories:
"As a new player, I want a clear intro to how to play the game, so that I can start playing without confusion."
"As an impatient player, I want a quick intro to learn the game, so I don't waste time on unneeded time."
"As an older player, I want non-tech-heavy introductions to the game, so I can play despite being new to my phone."
"As someone new to exercise, I want tutorials on the exercises, so I can get the most out of my missions."

No Prototypes have been implemented yet for this user story, as it currently only **SHOULD HAVE**, but it goes without saying why these user stories are important. By providing tutorials on how to play the games or complete the exercises, we lower the barrier of entry for players, making the game easier to play with others, whatever their knowledge of the game.

4.2.2.2 US5 - Cloud Data Access

User Story / Stories:
"As a developer, I want the system to store data online so that I can easily change information for the game for many users at once, without a tech update."
"As a social player, I want the system to let us pass the game between our phones, rather than passing around one phone, so that I can just hold my own phone."

Another fundamental goal, though it is currently regulated to **SHOULD HAVE**, as Morgan would like us to get the MVP working first, then branch out to multiplayer, as that is quite a feature to add to the program. That being said, allowing multiplayer between phones is still very valuable, as it would make it safer to play with players you do not know.

4.2.3 COULD HAVE

4.2.3.1 US6 - Difficulty Customization

"As a new player, I want a complete path to my location, so I can focus on learning the game."
"As a normal player, I want a small challenge of planning my own route between points on the map so that I can learn more routes."
"As a competitive player, I want little to no mapping information, so that I can challenge myself."

This is a feature that Morgan is rather interested in, as he wants to give the player more personalization of how one will play the game; however, we talked and noted that since it isn't a core feature, it will be held as something the game **COULD HAVE** for now.

4.2.3.2 US7 - Testing and Deployment

"As a Developer, I want to push my game with no bugs to build a better product."
"As a Developer, I want to push my game to as many platforms as possible to gain the most traction."
"As a picky player, I want to see no bugs while I play so that I can have a better experience."

These user stories made us think of the importance of testing our game to be the most polished product possible. Still, it isn't a core feature, so it will be held as something the game **COULD HAVE** for now.

4.2.4 WONT HAVE

Our team has decided that there are no goals to put in **WONT HAVE** for the time being.

4.2.5 Mapping User Stories to Domain Requirements

While this section was not specifically requested from our instructions, our team felt it was important to help map sections **4.1** and **4.2** together. The following chart elaborates on how our **User Stories** will help our team satisfy our **Domain Requirements**.

Domain Requirement (DR)	User Stories (US) that satisfy it
DR1 - Multi Platform	US7 - Testing and Deployment
DR2 - Real-Time Location Awareness / Tracking / Display	US2 - GPS and Player Tracking US6 - Difficulty Customization
DR3 - Integrated POI Database / Nationwide Coverage	US2 - GPS and Player Tracking US3 - POI Database
DR4 - Consistent / Polished User Interface / Usability	US1 - Unified and Responsive UI US4 - Tutorials
DR5 - Data Synchronization / Control	US5 - Cloud Data Access
DR6 - Tutorials and Beginner-Friendly	US4 - Tutorials
DR7 - Expandable / Maintainable	US7 - Testing and Deployment

Table 2. How we can map our User Stories to our Domain Requirements

With our functional requirements covered in **4.1** and **4.2**, we can now move to explaining our non-functional requirements in **4.3** and **4.4**.

4.3 Performance Requirements

Below are the listed performance requirements we will use to test our application and assess how well we can address **Domain** and **Functional Requirements**. All **Performance Requirements** assume these tests are run on standard mid-range mobile devices (Android Galaxy 10+, iOS 10+), with the GPS enabled and location permissions enabled, and access to a cellular data network (4G, or 5G). Not all **Domain Requirements** are listed out here, as things like **DR1** and **DR7** will come by virtue of how we construct our product, and will not be things we can test for.

4.3.1 - Supporting DR2 (Real-Time Location Awareness / Tracking / Display)

The GPS and Unity map should refresh at least once every 30 seconds to provide a smooth experience. The displayed location should be accurate within 50 feet. Continuous use of a GPS can be straining on a battery; ours should use < 10 % battery per 10 minutes of active gameplay.

4.3.2 - Supporting DR3 (Integrated POI Database / Nationwide Coverage)

The system should query for POIs and return within 25 seconds. If queries are too large to load (Beyond 100 points), they should limit to that number to avoid long wait times, and if they are too small (<10 points), they should try the other database, and less specific queries to find the necessary points. The system should be able to work in any city, from something small like Anthem, AZ, to a large one like Boston, MA.

4.3.3 - Supporting DR4 & DR6 (Consistent / Polished User Interface / Usability & Tutorials and Beginner-Friendly)

New users should be able to complete the tutorials and understand how to play a mission in the game and perform basic tasks within 15 minutes. During these tests, screens should load quickly within 1-2 seconds. During these tests, UI should be straightforward enough to have newcomers complete all tasks with <5 user errors. Players should be able to learn how to do exercises from game videos within 3 minutes per exercise.

4.3.4 - Supporting DR5 (Data Synchronization / Control)

The system should be able to send and retrieve cloud-hosted mission data within 20 seconds to ensure players can switch who is the group leader without phone handoffs for multiplayer.

4.4 Environmental Requirements

The following section elaborates on what external constraints we must adhere to in our design. For our code editor, our team is constrained to Unity as it was the decision of last year's team and cannot be swapped without abandoning all prior code. Since our client wants a mobile game, we are constrained to design our app for that, and will be designing for both Android and iOS phones, as they are the most common types, about 98% of all global phones. Beyond that, we have no strict requirements, and simply chose which APIs and technologies fit our program the best.

5. Potential Risks

Several risks could affect the success of the project. The biggest technical challenge is GPS reliability across devices, since hardware differences can lead to inconsistent accuracy. Another potential issue is managing API rate limits or downtime, which could interrupt map rendering or location queries. Database security and backup configuration can also carry risk, as improper handling could lead to data loss or breaches. Finally, schedule constraints may pose challenges since the project timeline is short and requires coordination across multiple technologies.

The team plans to minimize these risks by testing GPS and API functionality early in development, automating database backups, and maintaining clear documentation for all setup procedures. We will also perform small-scale load tests to verify stability before deploying the final version.

We have planned to use APIs for both the points of interest data and the map itself. While there can be downtime with any system, we have chosen APIs that are well-known and widely used and should be resistant to failures. In the event of an API failure, the app will not function correctly, but this cannot be avoided in any way, so this is a risk we will need to accept. The app is in a usable but limited state currently, and this will be a good fallback if an API were to run into issues or become unavailable. All essential game functions like dice rolling and exercise randomization are done by simple code on the user's device, and this local-only part of the app will be a good fallback.

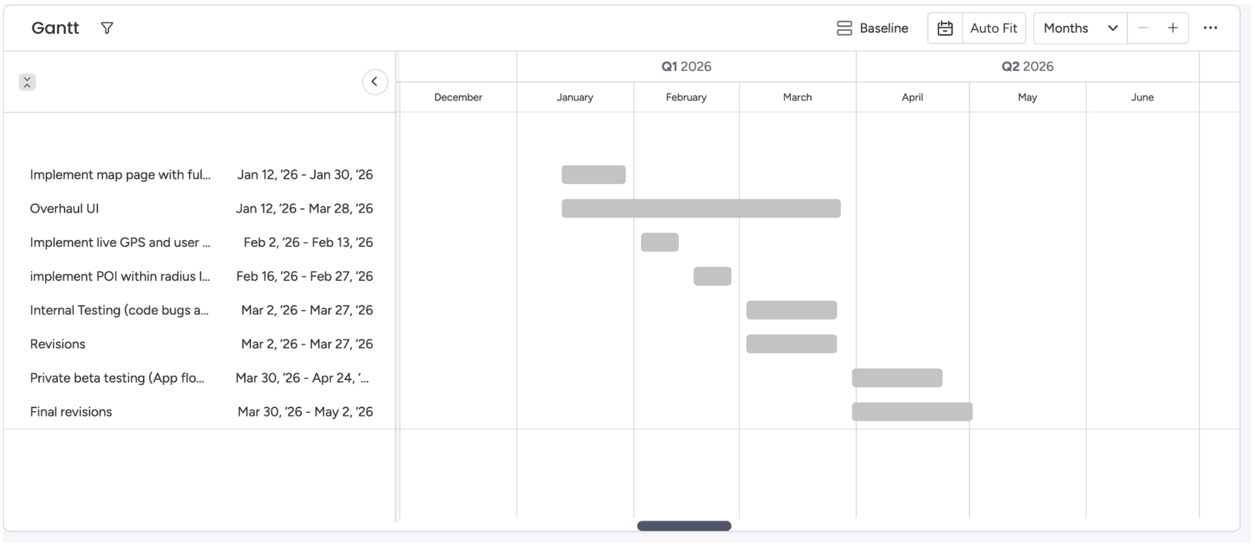
In addition to the risks of reliability issues and program performance hiccups, we need to consider the risks involved with using GPS data. Location data, even imprecise locations, can be used for a multitude of nefarious purposes. For this reason, we will not be storing any GPS data since any data stored could potentially be leaked. In addition to the lack of data storage, we will be implementing a GPS data access permission prompt when the app is opened for the first time. This is required by Apple and Google in order for an app to be deployed to their app stores or even to get access to location data from the system.

There are two main categories of sensitive data that are relevant to apps like the one we are making: GPS and Personally Identifiable Information. While we do need to use GPS data for the app to work, we do not need to collect or use personal data. There is a portion of the app that prompts users for their names, but users can use any name they would like, and there is no other personally identifiable information collected.

Another significant risk with an app that guides users to locations in the real world is that they could be led to dangerous areas or places where they are not allowed to be. To reduce this risk, we will likely add a warning pop-up before any GPS features are used, instructing users that they should be careful and that they are responsible for their own actions.

While our program will face some potential risks with regard to reliability, accuracy, API reliability, user behavior, and inherent risks with handling user data, we will reduce the risks by planning accordingly and following best practices.

6. Project Plan



Implement map page with ful...	Jan 12, '26 - Jan 30, '26
Overhaul UI	Jan 12, '26 - Mar 28, '26
Implement live GPS and user ...	Feb 2, '26 - Feb 13, '26
implement POI within radius l...	Feb 16, '26 - Feb 27, '26
Internal Testing (code bugs a...	Mar 2, '26 - Mar 27, '26
Revisions	Mar 2, '26 - Mar 27, '26
Private beta testing (App flo...	Mar 30, '26 - Apr 24, '...
Final revisions	Mar 30, '26 - May 2, '26

Fig. 7 A Gantt chart of our project plan

Our team has carefully considered the best path forward towards the goal and planned the next semester accordingly. Our plan can be seen in **Fig. 7** above. We will start working on implementation on day one, January 12, 2026. We will start by finalizing the map page and integrating it into the app fully with all navigation to and from it wired, and make sure it activates reliably and quickly. The next step would be to display the user's location on the map in real time. We may encounter some issues in this phase because we need to make sure the users are prompted for GPS access on a range of devices across both iOS and Android. We will test on as many phones as we can, but we will need to rely on the beta testing later in the semester to ensure that there are no edge cases. Once we have the map and live GPS readout working and integrated fully, we will build on it by wiring the logic for the points of interest. This stage will include modifying current pages of the app to intake data, such as desired walking distance or even what kinds of points the user would like to go to.

Implementing these core features would take us from mid-January, when the semester starts, all the way to the end of February. During this time and onwards, into late March, we will also be continuously working on the UI overhaul. This aspect of the project has been partitioned into an exceptionally wide window because it will be extremely comprehensive and wide-reaching in the project. Additionally, we will need to be working collaboratively with our client, Morgan, on each phase of the UI implementation since it is very subjective in nature. Due to the subjectivity of UI development goals, we are expecting to go through multiple iterations, which will take considerable time, and there may be stretches of a week or so where a meeting cannot be arranged with the client due to scheduling constraints.

We expect to have everything implemented to a reasonable extent by the end of February, except the UI. In March, we will stress test the application internally, trying to find any bugs in the logic, hiccups in the app flow, and edge cases we hadn't thought of in the planning phase. We will fix these by the end of March in preparation for one of the most exciting parts of the project, giving it to real users. The UI should be finalized by the end of March as well, which means that we would have a fully working app ready for testing and feedback by real users.

Since this is an end-user-facing product, testing in the real world with real users is absolutely critical. We need to know if the UI is too difficult for non-tech people to use, or if the app has issues loading over a weak cell network, or breaks on a phone we couldn't test on before. To prove the app works in the real world and can take several hours so we will likely run these tests on the weekend at various places around Flagstaff, such as downtown or the NAU campus. At the end of the game, we will collect feedback through a questionnaire.

The exact approach we take for the app distribution for testing will not be known until closer to when we need to conduct the testing, since we still don't know if we will be publishing this app to real app stores. If we do publish the app, we will invite people to private betas through the Apple App Store or Google Play Store. If we do not publish the app, we will bring our laptops to the event and load copies of the app onto the phones of the test run participants.

The final phase of the project would be to polish the app to be turned over to the client fully. During the testing phase, we will be implementing user feedback and working on this final polish. Ideally, we will host test runs frequently, trying to reach as many participants as possible and implement their feedback before we run the app again the next weekend. We have planned on finishing the project on Friday, May 1, 2026, which will allow us to focus on final exams and graduation during the very last week of the semester.

7. Conclusion

This capstone project incorporates many of the things we have learned during our time at Northern Arizona University and draws on the skillsets of many individuals in the team. We will use our core skills of writing clean, readable code that accomplishes the goal efficiently, and also draw on more niche experiences such as game design, app development, and UI expertise. As is often the case in computer science, we will stand on the shoulders of those before us; specifically, we will be building on top of a previous capstone group's year of hard work and planning. We will take the app they made that serves as an excellent MVP for the project as a whole, and truly make it excellent. The project will demand a careful balance of both frontend and backend development since the UI or functionality could each make the app unusable, but it will require both working in harmony to create a clean, usable, and fun app for our clients and their clients, the end users.

During the development phase, our team will add a map with live readouts of the player's location, a nationwide POI database, clean up the app UI, and introduce fun new animations. We may even add different game modes that make the app accessible for new players and make it challenging for the more experienced players. If implemented correctly, this app has the potential to bring people closer together and closer to their community. In this document, we have defined the features we plan to implement and even outlined some we may not get to implement, as well as planned an entire semester of programming and vigilant testing. While this project will require a year of hard work and commitment, we are excited to bring the client's vision into reality.

8. Glossary

Term	Definition
API	A plug-in point that lets our app talk to another service.
Authentication	Plug-in and verification
Backend	The server side that stores data and runs logic.
Deployment	Releasing the app to users (test or live).
Minimum Viable Product (MVP)	The smallest useful version to prove value.
Scalability	Ability to handle more users/data without slowing or breaking.
Latency	The delay between a request and the response is often due to networking, and even the speed of light
Geospacial	Anything tied to real-world locations and coordinates.
Point of Interest (POI)	A notable placereal-worldp
Unity	The game engine we use to build the mobile app.

Table 3. Our Glossary of Tech-related terms