BoilerJourney

Design Document

Team 27

Colin Flynn
Zander Carpenter
Shubhaang Agarwal
Edward Kelley
Kashvi Sahjwani
Charlie Zhang

Index

Purpose

Overview

Just in this last year Purdue received 72,800 applications and admitted 50.3 % of those applicants. These potential students, and those who eventually accept, want information on what Purdue's campus offers. However, finding this information is difficult and not very entertaining. Websites like Niche and U.S. News give plenty of information but fail to provide an insight into the culture of Purdue. Even Purdue's own website struggles to give a full picture of the campus and its use can become stale or repetitive.

This is why we plan to make BoilerJourney. BoilerJourney will be a mobile 2D multiplayer role-playing game to address the issues of the other information sites used by potential Purdue students. In this game, users will be able to experience life on campus through tasks and missions. Using the experience they gain, students will be able to climb the class ranks until they can graduate and be well-informed about everything our school has to offer. Our users will also have the opportunity to interact and network with other students/players to make new connections and friends before they even arrive in West Lafayette.

Functional Requirements

1. As a user, I would like to be able to create a new game.
2. As a user, I would like to be able to load a previous game.
3. As a user, I would like to be able to save and view my progress.
4. As a user, I would like to be able to choose a virtual avatar.
5. As a user, I would like to customize my in-game appearance.
6. As a user, I would like to be able to travel around Purdue's Campus.
7. As a user, I would like to be able to view information about important locations.
8. As a user, I would like to be able to view my current class rank.
9. As a user, I would like to be able to rank up from freshman to senior.

10. As a user, I would like to be able to view my current experience and how much is needed to rank up.
11. As a user, I would like to be able to view a leaderboard that ranks all users by their experience points (xp).
12. As a user, I would like to be able to complete various quests and sidequests that reward xp.
13. As a user, I would like to be able to interact with non-playable characters (NPCs) around campus.
14. As a user, I would like to be able to complete tasks that teach me more about Purdue so that I can better understand the university.
15. As a user, I would like to be able to view ratings on important buildings and locations around campus.
16. As a user, I would like to be able to see Purdue during the day and night.
17. As a user, I would like to be able to see the map during the Summer.
18. As a user, I would like to be able to see the map during the Fall.
19. As a user, I would like to be able to see the map during the Winter.
20. As a user, I would like to be able to see the map during the Spring.
21. As a user, I would like to be able to enter Purdue's Memorial Union as the social hub of campus.
22. As a user, I would like to be able to view a global chat with everyone on the current server.
23. As a user, I would like to be able to send messages to other users.
24. As a user, I would like to be able to create chat groups with multiple users.
25. As a user, I would like to be able to join groups centered around various topics.
26. As a user, I would like to be able to find people with similar interests.
27. As a user, I would like to be able to get suggestions about things to do around campus.
28. As a user, I would like to be able to leave public reviews for different places.
29. As a user, I would like to be able to view reviews of restaurants around campus.
30. As a user, I would like to be able to create and/or join a study room with multiple people.
31. As a user, I would like to be able to personalize my in-game dorm room.
32. As a user, I would like to be able to find collectibles around campus.

33. As a user, I would like to be able to display my collectibles.
34. As a user, I would like to be able to view information on Clubs.
35. As a user, I would like to be able to view information on upcoming games.
36. As a user, I would like to be able to use transportation services at Purdue, so that I can travel to different locations faster.
37. As a user, I would like to be able to view a mini-map, so that I can navigate around the Purdue campus.
38. As a user, I would like to be able to have a quest log, so that I can keep track of pending and completed quests/sidequests.
39. As a user, I would like to be able to have an inventory, so that I can store and view all my collectibles.
40. As a user, I would like to be able to view item descriptions of the collectibles in my inventory, so that I can know specific details about an item.
41. As a user, I would like to be able to have a togglable UI.
42. As a user, I would like to be able to experience a storyline, so that I can understand how the game flows.
43. As a user, I would like to be able to submit bug reports, so that the developers are aware of game-breaking bugs/glitches in the game.
44. As a user, I would like to be able to have multiple save slots, so that I can play as different characters in the game.
45. As a user, I would like to be able to have a friend system, so that I can be online friends with players that I enjoy gaming with.
46. As a user, I would like to be able to travel off-campus to nearby towns like Lafayette, so that I can find new collectibles in these locations and expand my gameplay (if time allows).
47. As a user, I would like to be able to trade collectibles with other players.
48. As a user, I would like to be able to view a stats menu, so that I can understand what the stats are, what each stat does, and how many points I have in each stat.
49. As a user, I would like to be able to input and check my class schedule.
50. As a user, I would like to receive reminders for my classes (if time allows).
51. As a user, I would like to be able to visually see on the map where my classes are located.
52. As a user, I would like to be able to view news events occurring around Purdue.

53. As a user, I would like to be able to start a voice chat with another user.
54. As a user, I would like to be able to start a video call with another user (if time allows).
55. As a user, I would like to receive notifications if someone sends me a message (if time allows).
56. As a user, I would like to receive a notification if someone creates a study room (if time allows).
57. As a user, I would like to complete a daily Purdue trivia question to gain XP.
58. As a user, I would like to receive a notification when any of my friends joins the server (if time allows).
59. As a user, I would like to complete an in-game tutorial to understand the game (if time allows).
60. As a user, I would like to customize my controls (if time allows).

Non-Functional Requirements

Architecture:
We plan to develop a mobile application for iOS using the Godot Engine. Using Xcode and the iOS 15 SDK will allow us to test and debug the app as well as distribute it to all devices that run on iOS. We will also implement a separate frontend and backend that will allow us to easily and quickly modify the application i.e., execute continuous delivery.

Scalability:
The game will be split into multiple servers to maximize the quality of the gameplay for each player. We plan to implement this by making the maximum server size to be at most 40 users. We also plan to include server chat cooldowns so that global and individual chats are safe from abuse by high message traffic. Finally, the trading and marketplace system will be player-driven allowing it to scale with a growing player base.

Performance:
The game would run at a minimum of 30 frames per second and respond to touch inputs within 200ms so that players can experience gameplay that is smooth and responsive. We will also ensure that the initial loading screen takes less than 7 seconds so that users can start gameplay shortly after opening the game. We also plan to allow users to switch to a “Low Graphics” mode for faster game performance. We also plan to improve the game performance by implementing Occlusion Culling so that surfaces and objects that are not within the view of the camera are not rendered. Finally, we will allow users to view and switch between servers so that they can join a server with less lag if their device cannot handle the load of the current server.

Usability:
Our game will feature a simple and effective UI that maximizes the options for users while minimizing its difficulty of use. To accomplish this we will place the different UI components in locations so that they do not block important parts of the screen and are reachable for users while holding their phone. Our UI will use simple language that is easy to understand making it suitable for users who may not have played a game of a similar style before. Also, the quests in our game will have a clear objective so that the players can easily understand how to complete the quest. Finally, we will ensure that the map and music do not use any unpleasant or offensive colors or music.

Safety and Privacy:
Users will be protected by Godot’s built-in security functions for networking to protect their personal information. This will result in secure and private messaging for those using the chat and trading systems. We will also use Godot’s provided encryption system to protect the passwords and data for players. Users will be able to specify what data they want to be hidden from the public to protect their privacy from other players. If time allows we will also implement a system to retrieve accounts in cases of forgotten usernames/passwords and implement anti-cheating measures to ensure a fair gameplay experience for all users.

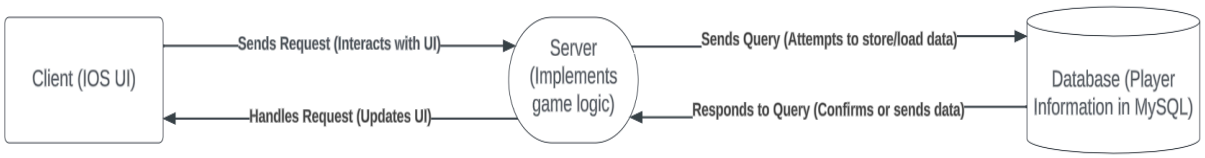
Design Outline

High-Level Overview

Our project will be a 2D RPG game available on IOS to teach users about Purdue University. This game will be built using the Godot game engine as it is powerful and has its own 2D-specific engine. To implement our app we will follow the client-server model. The client-server model was chosen as this gives our app more reliability and flexibility while keeping control in a centralized place. Our reliability will be stronger as we can add more servers to handle larger user bases and give our group more flexibility allowing us to implement networking capabilities and keep our map in a fixed state across all users. This architecture also allows us to keep our game logic and implementation in one central location ensuring a consistent gameplay environment for all players.

The client side of our model will be the IOS app that is seen and interacted with by the user. We will use Xcode to take our game from the Godot engine and make it compatible with IOS. The server side will be the backend or host of our game which is responsible for managing the game state, processing player actions, and handling communication between clients. To store and access needed user data our game needs to run our servers will use a database. The database that will work best for the game's needs is MySQL.

High-Level Diagram



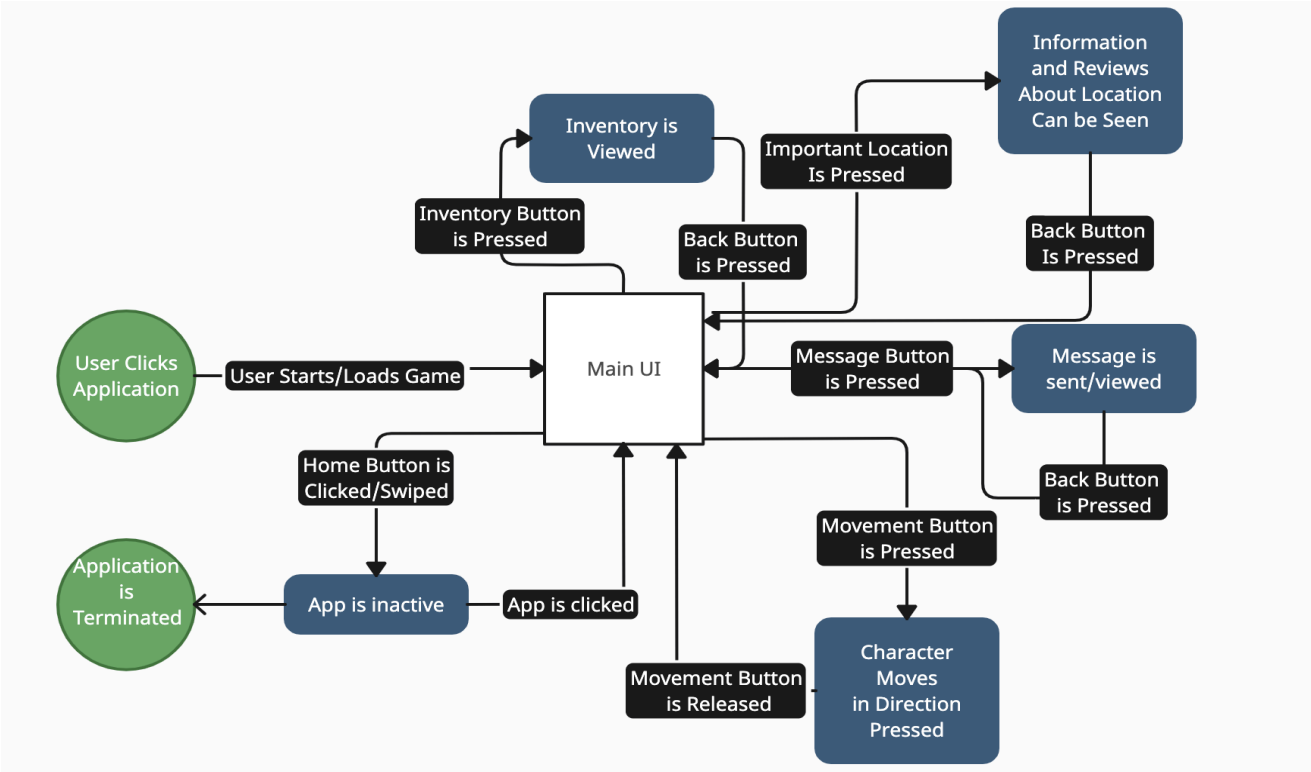
1. **Client**
    - The IOS client will serve as the user interface for the game.
        - Will be built using Godot’s game engine.

- The client will make requests to a singular server to complete actions within the game.
    - Messaging, completing tasks, viewing information on campus, movement, inventory, etc…
- The client will receive information back from the singular server to complete the request and update the UI.

2. **Server**
    - A server will represent an instance of the game running on a network with a limited capacity.
    - The server will receive requests from clients and determine their type.
        - Database information request vs. UI update
    - The server will then send back the requested information and/or update the UI accordingly.
3. **Database**
    - The database will store all information needed to allow users to continuously save their state.
        - Username/Passwords, game data (exp, collectibles, …), chat logs, etc…
    - The database will respond to valid requests from the servers and respond with the needed data.
    - The chosen database for our game is MySQL.

User-State Diagram

The diagram below shows the main states that our users will be in while playing BoilerJourney. The central state in which the user will spend most of their time is the main UI which will feature the 2D map of Purdue's campus along with needed controls and options. From this UI, users will be able to access movement controls, view their inventory, send/receive messages, view information, and complete important activities. Then, once a user is done they will swipe out of the app leaving it inactive or terminated.



Design Issues

Functional Issues

1. What information do we need to sign up for an account?
    a. Username and password only
    b. Username, password, and email address
    c. Username, password, email address, and phone number

    Choice: b

Justification: For account registration, opting for a username, password, and email address is imperative to establish user identity and ensure account security. Email verification serves as an additional layer of authentication and aids in password recovery/resetting when required. While including phone numbers enhances security, it might not be necessary for our application's scope. Utilizing SMS verification could also incur additional costs. Hence, username, password, and email suffice for sign-up, balancing security needs with user convenience and minimizing potential financial burdens associated with additional verification methods.

2. How would we determine the leaderboard ranking?
    a. Xp
    b. Class Ranking
    c. Number of quests completed
    d. Both Xp and Class Ranking

    Choice: d

Justification: In determining the leaderboard ranking, we chose to integrate both XP and class ranking criteria. By categorizing players into distinct groups based on class rankings (Freshman, Sophomore, Junior, Senior), we foster a competitive environment within each cohort. Players within the same class rank compete for the top position using their accumulated XP. At the end of each season, successful

players advance to the next class rank, while those at the bottom are demoted. This approach incentivizes continuous engagement and skill development while ensuring fairness and relevance across diverse player demographics.

3. How would we determine map settings (season and time of day)?
    a. By allowing users to switch between different maps
    b. By using real-time location data to determine the settings

    Choice: a

Justification: Considering the complexity and resource-intensive nature of obtaining real-time user location data, we opted for a user-controlled approach to map settings. Allowing users to switch between different maps based on their preferences provides flexibility and control, enhancing their overall gaming experience. This approach ensures that users can explore various environments without waiting for real-time data processing, thereby minimizing potential delays. By empowering players to choose their desired maps, we prioritize user convenience and accessibility while maintaining a streamlined development process within the project's time constraints.

4. How can users communicate with each other?
    a. Real-time server chat system
    b. Real-time global chat system
    c. Discussion Board

    Choice: a

Justification: The main purpose of allowing users to communicate with each other is to share info about the game and build friendships with other players. Whenever a player has a question or comment about the game, they should be able to immediately post it to the chat where other players can respond promptly. Hence, we advocate a real-time server chat system as the most effective means to facilitate this interaction. Implementing a global chat system could lead to performance issues due to potential high message traffic when numerous players are using the

chat simultaneously. A discussion board wouldn't suit our needs as we expect chat messages to be short and frequent.

5. How should the game handle player progression?
    a. Linear progression through levels with increasing difficulty.
    b. Branching paths based on players' choices.
    c. Dynamic scaling of challenges based on player skill.

    Choice: a

Justification: Implementing a linear progression system with increasing difficulty levels ensures a balanced learning curve, keeping players engaged with achievable goals. This approach provides a clear sense of advancement, unlocking new challenges and rewards as players progress. By gradually introducing more complex gameplay mechanics and obstacles, it maintains player motivation and prevents frustration. Additionally, linear progression allows for a structured gameplay experience, guiding players through the game's content in a coherent manner. This systematic approach not only enhances the overall gaming experience but also helps players develop their skills gradually, ensuring they are adequately prepared for more challenging tasks as they advance further in the game.

Non-functional Issues

1. What software should we use?
    a. Unity
    b. Godot Engine
    c. Unreal Engine
    d. GameMaker

    Choice: b

Justification: We decided to use the Godot Engine as the main software to develop our 2D RPG mainly because of our limited prior experience in game development. Its interactive interface, features, and documentation are very beginner-friendly as

compared to other software that are targeted toward professionals. Additionally, Godot is also a free and open-source software that will allow us to access advanced game development tools at no cost. Lastly, Its compatibility with Apple's development tools such as Swift and Xcode also played a big role in us choosing Godot since we will be using those tools to make our game available on iOS devices.

2. What SDK (Software Development Kit) should we use?
    a. iOS SDK
    b. Android SDK

    Choice: a

Justification: The iOS SDK, coupled with Swift, offers developers an efficient development experience. Swift's concise syntax and advanced features accelerate coding processes, leading to faster development cycles. Furthermore, the uniformity of iOS devices minimizes fragmentation, ensuring a consistent user experience across various devices. These attributes collectively facilitate streamlined app development and maintain high standards of quality and consistency within the iOS ecosystem.

3. What database should we use?
    a. MongoDB
    b. GeoJSON
    c. DynamoDB
    d. MySQL

    Choice: d

Justification: MySQL is a reliable source choice for its robustness and stability. It offers widespread features for managing relational data efficiently with full-text indexes and stored procedures. MySQL is compatible with most popular programming languages and frameworks, which simplifies the integration process. On top of that, it is open-source which provides additional access to resources and

expertise. It will provide us with the foundation for scalable and secure data storage and retrieval.

4. What security measures should we implement to protect user data?
    a. Using Godot secure chat
    b. Using privacy encryption features
    c. Allowing users to control what others can see

    Choice: a & c

Justification: Using the secure chat feature available in the Godot engine will allow users to be protected against malicious content and create a safe space for people to share their thoughts and connect with each other. Using the secure chat feature will also provide users with a sense of security by preventing any personal chats from being leaked or accessed by others, thereby allowing them to remain carefree and assured of their privacy. Additionally, allowing users to be able to control what others can see from their profile, like their name, date of birth, and courses they are in, will make it so that the users don't have to share any information they don't want to and don't feel like their privacy is being violated by strangers online. On the other hand, while a privacy encryption feature is very effective, it is not feasible since Godot does not have the capability to fully support that feature

5. How will we handle too many players?
    a. Allow limited users to access the game at once and create waiting rooms till the servers open
    b. Allow priority access to regular users or those with higher ranking
    c. Create multiple servers each with a limit

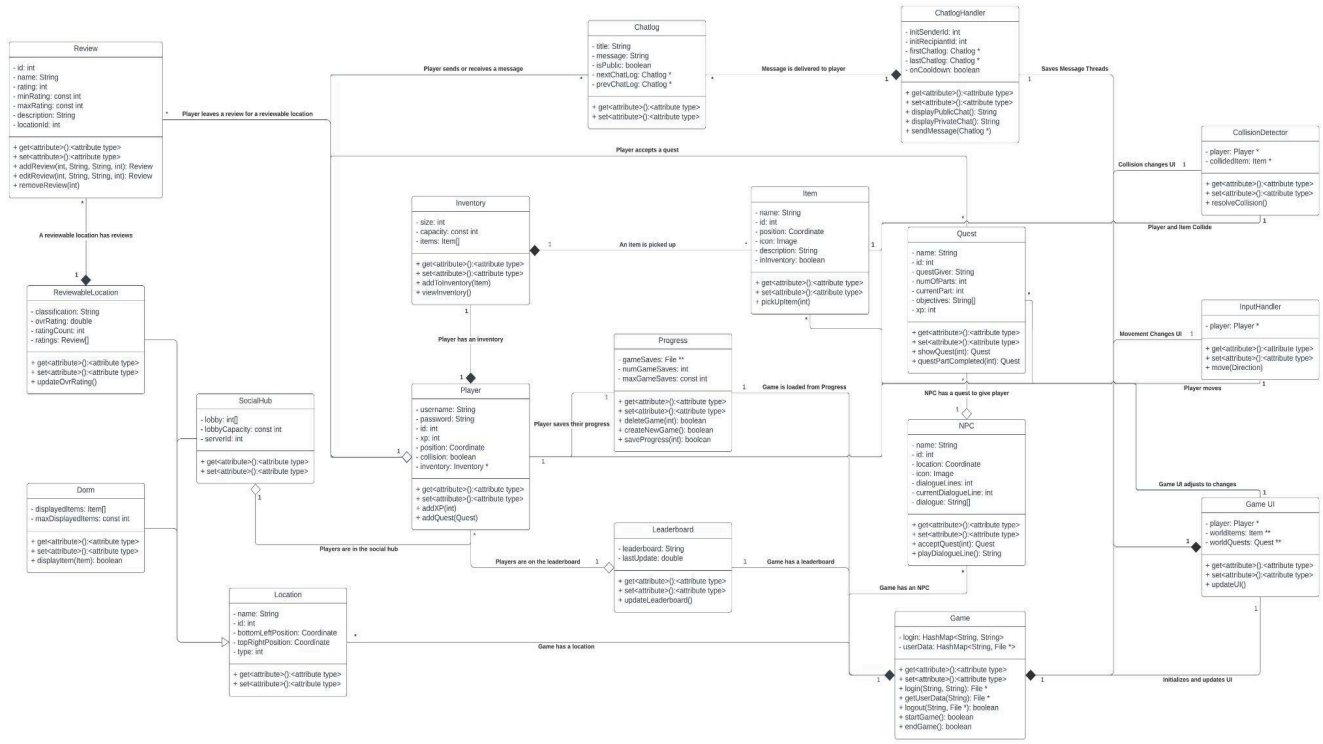
    Choice: c

Justification: We decided to split the game into multiple servers to maximize the quality of the gameplay for each player. We plan to implement this by making the maximum server size to be at most 40 users. This would allow all players to get the chance to play the game at all times without having long wait times and would prevent overcrowding while maintaining a high-quality gameplay environment for

the users. Waiting rooms are annoying to most users and would lead to them losing interest in the game. On the other hand, not allowing someone new to access the game would deteriorate the gaming experience which would result in a negative perception of the application and may result in less engagement.

Design Details

Class Design



Descriptions of Classes and Interaction between Classes

- **Game UI**
  - We will implement a game UI class that manages all the UI components and their changes
    - This will include methods to call the other classes that are needed to control our game logic
- **Game**
  - The Game class will manage users signing into and logging out of the game, will retrieve data from their selected save file to launch the game in the state that it was saved in, will manage player-to-player communication, and will manage the entities that are a part of the game (player, NPCs, Locations)

        - Includes data structure managing user credentials and a data structure managing users' save data
- **Review**
    - A Review object is created every time a user wants to leave a review for a Location
        - This review will feature a 1-5 star rating
        - A message about why this review was left
        - Possibly a count value of how many people viewed the review as helpful
    - Instance Variables: Name, Text, Reviewer, LocationReviewed, Rating, Date, HelpfulCount
- **Location**
    - Location objects represent real-life places around Purdue that you can interact with
    - Location is a superclass for the ReviewableLocation, SocialHub, and Dorm objects.
        - The ways that players can interact with the Location objects are different depending on their type
    - Instance Variables: Name, Location, Type
- **ReviewableLocation**
    - ReviewableLocation objects are created at the start of the game representing real-life locations at Purdue
    - The objects will feature information about the location that is necessary for understanding the campus
        - Name, Address, Description
        - Classification (Residence Hall, Dining Court, Academic Building, Study Area)
    - This will also contain a list of Reviews on the specific location which will be used to calculate an average rating
    - Instance Variables: Name, Location, Description, Classification, Reviews (array of review objects)

- **Quest**
    - A Quest object is created for each quest available in the game.
    - This object will represent important information about each of the quests that will be available to complete
        - Name of the quest, the objectives, number of XP given upon completion, number of parts
    - This will be used to keep track of players progress
    - Instance Variables: Name, Objectives, XP, NumOfParts, QuestGiver
- **NPC**
    - NPC objects are created for each NPC in the game.
    - These objects will include information about the NPC including various messages
        - When an NPC is interacted with there will be a set message given to inform players more about the campus
    - Instance Variables: Name, Type, Message
- **Inventory**
    - This class will represent a user's specific inventory of collectibles and consumables
        - specific for each user
    - This will be achieved by using a list of Item objects representing the items the user has found while playing the game
    - Instance Variables: items (array of Item objects)
- **Item**
    - An Item object is created for each item in the game. These items will typically represent collectibles or consumables
    - The items will be acquired through various tasks or can be randomly found around campus
        - Will have specific names and descriptions to represent where they were found or how they were obtained
    - Instance Variables: Name, Type, ID, Icon, Description

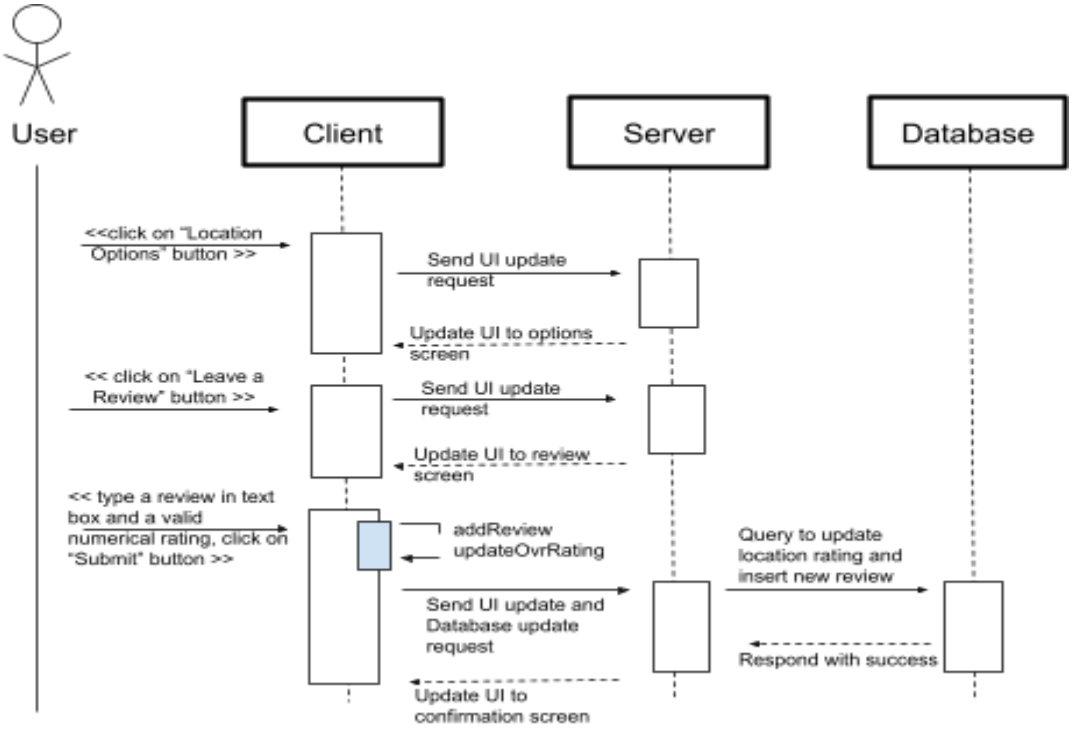
- **Dorm**
    - This object will represent the dorms for each specific user in the game
    - We plan on creating a customizable dorm that players can enter and exit
    - This object will contain the information needed to update the appearance of the dorm
        - Name of the player, Color scheme of the dorm, Style
    - Instance Variables: Name, color, style
- **Progress**
    - Progress is a class that handles save progresses in the game.
        - Whenever a player creates a new game, the Progress class instantiates a Game object.
        - Whenever the player deletes a save slot, the Progress class deletes the Game object.
        - Whenever a player saves a game, the Progress class will also update and save progress for the appropriate Game object.
    - Instance Variables: Games (an array of all save slots)
- **InputHandler**
    - This class handles all the user input from the main app.
    - These inputs can come in many different forms
        - Keyboard, Tapping the movement controls, Tapping the buttons, etc.
    - This class will then call the appropriate class based on which button or action was pressed so the UI can be updated and, if necessary, make changes.
    - Instance Variables: Player
- **PlayerList**
    - This class is responsible for maintaining a list of Player objects representing players in a server.
        - Will be updated when new users enter or exit the game

    - A PlayerList object is created for each server in the game.
    - Instance variables: players (an array of Player objects)
- **Players**
    - This object represents the individual users of our game
    - Contains information about the user needed for the game to function
        - Username, PlayerID, Display Name
    - Will be used to keep track of players on server
- **LeaderBoard**
    - This class is responsible for keeping a live leaderboard for our game
        - Will contain a list of players in the current server
    - The leaderboard is based on the players current XP level, with the higher XP players at the top
    - Will be using Player and PlayerList to get information on servers current players and their XP
    - Instance Variables: leaderboard, lastUpdate
- **CollisionDetector**
    - A class that detects collisions and updates the game state accordingly. It can be used to check if the player is colliding with a solid entity. It can also be used to check if NPCs are colliding with solid entities.
- **ChatlogHandler**
    - This class is responsible for handling public and private chatlogs in the game.
    - This class works by creating two arrays of Chatlog objects, public and private, and then using them to display sent messages.
    - Instance Variables: publicChatlogs (arrays of Chatlog objects), privateChatlogs (arrays of Chatlog objects)
- **Chatlog**
    - This class represents the objects that contain sent and received messages within our messaging system.

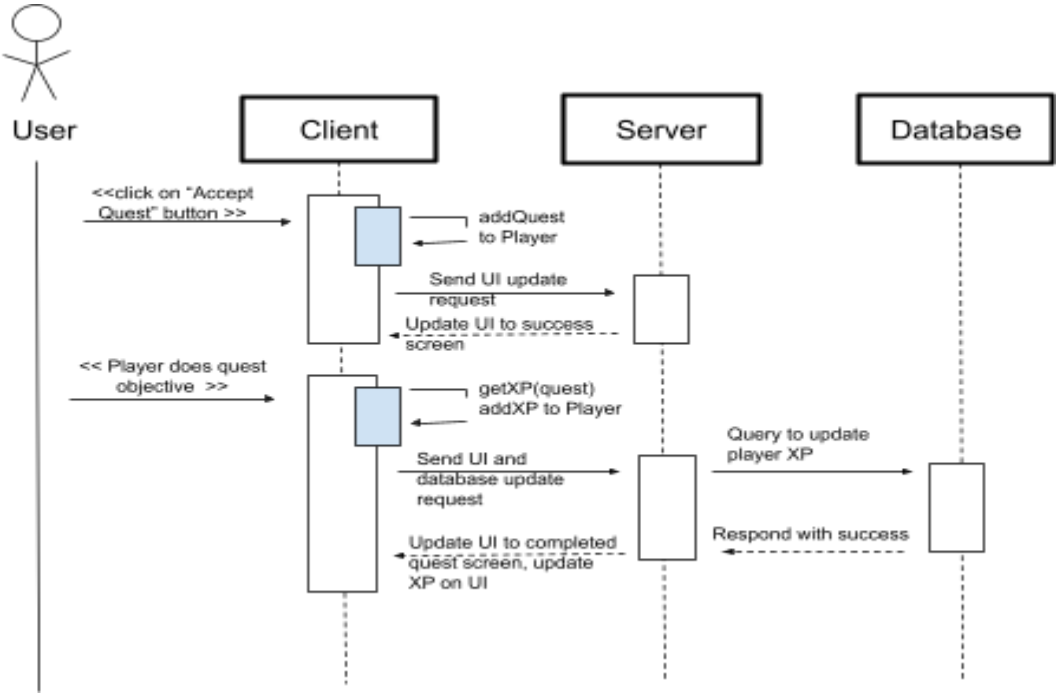
    - A chatlog object is created for each chat in the game.
        - Server chats are stored in the publicChatlogs array
        - Private chats are stored in the privateChatlogs array
    - This object follows a linked-list structure with links to the next and previous chatlogs.
    - The created Chatlogs are then stored within the corresponding ChatlogHandler arrays
- **SocialHub**
    - This class is responsible for handling the information and methods needed to implement the social hub for our game
    - This hub will be located at Purdue's Memorial Union and will be a way for students to learn more about events happening on campus
    - Will be a specific social hub for each server
    - Instance Variables: lobby, lobbyCapacity, and serverID

Sequence Diagrams

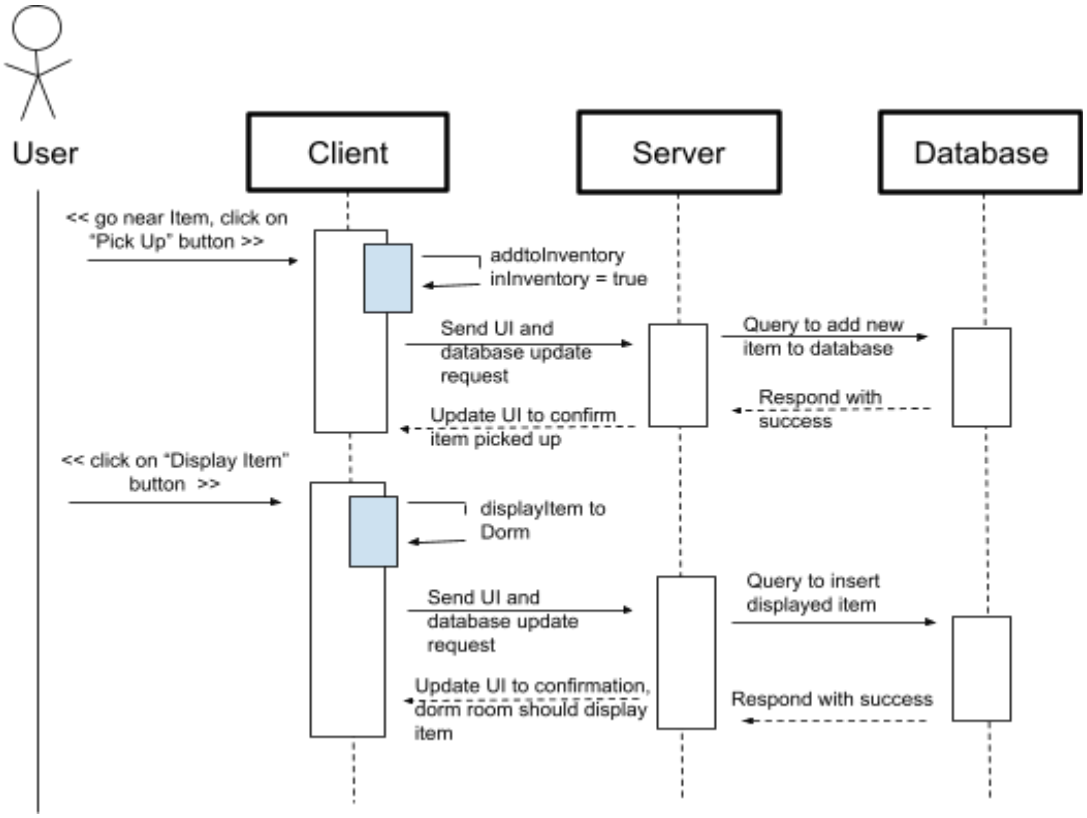
Adding a review to a location



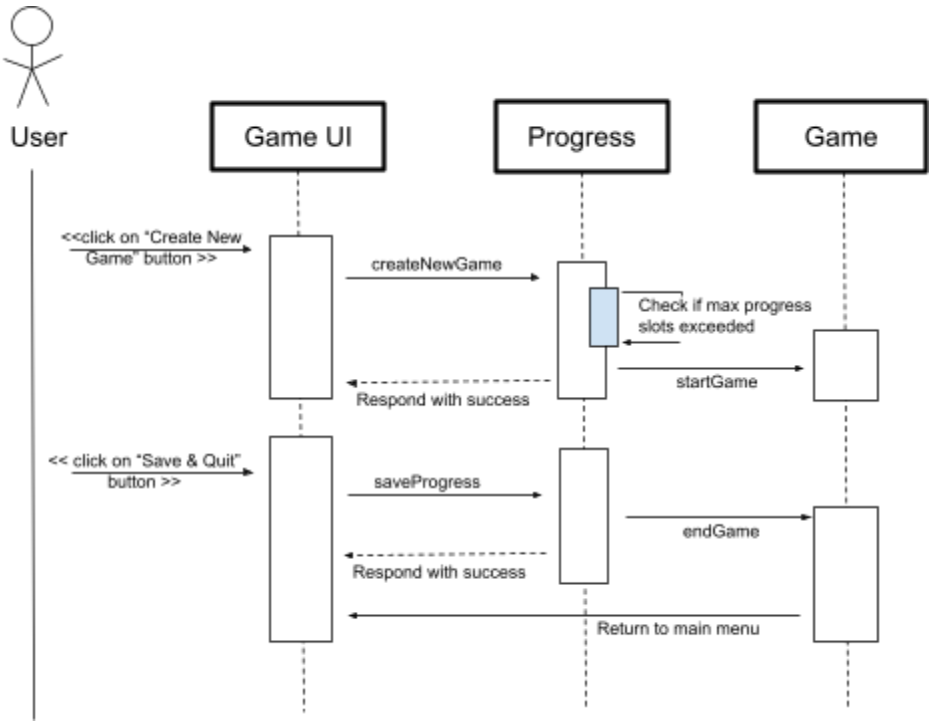
Starting and completing a quest



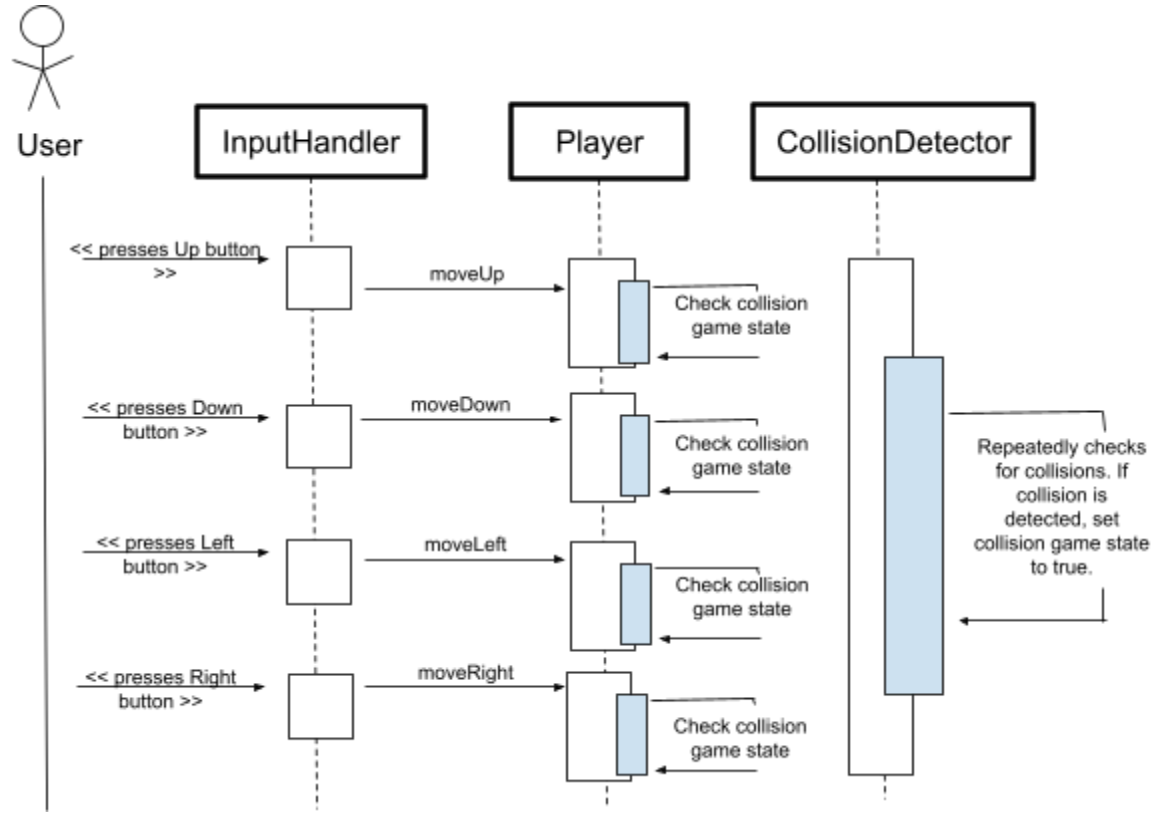
Picking up an item/collectible



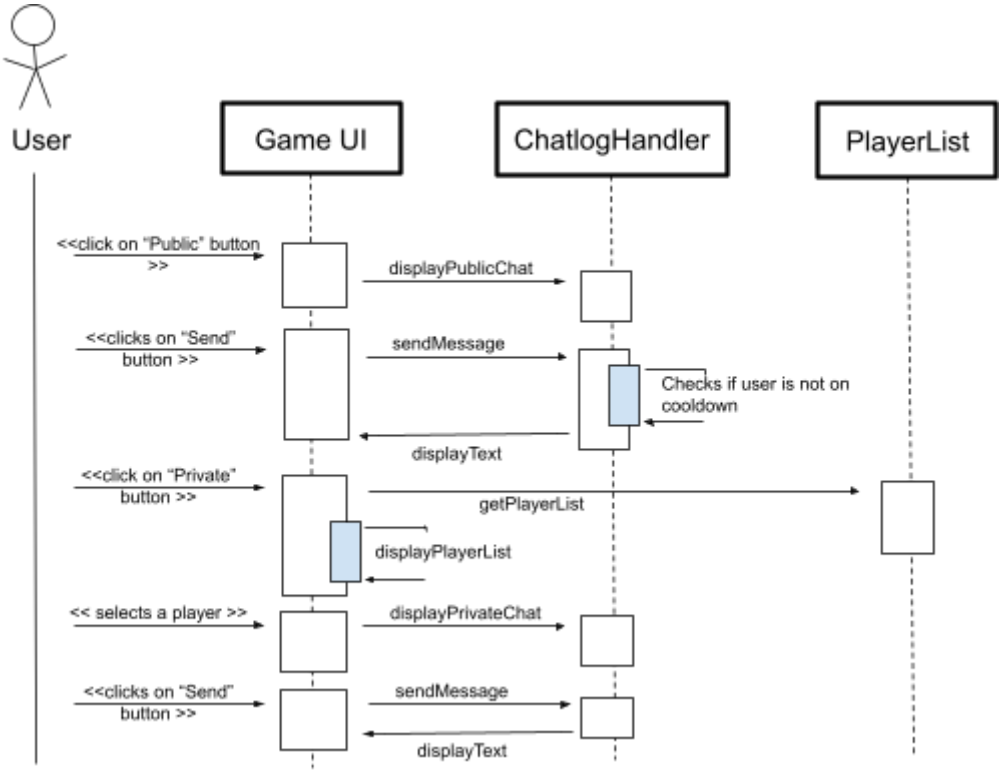
Creating new game/saving and quitting game



Moving the character

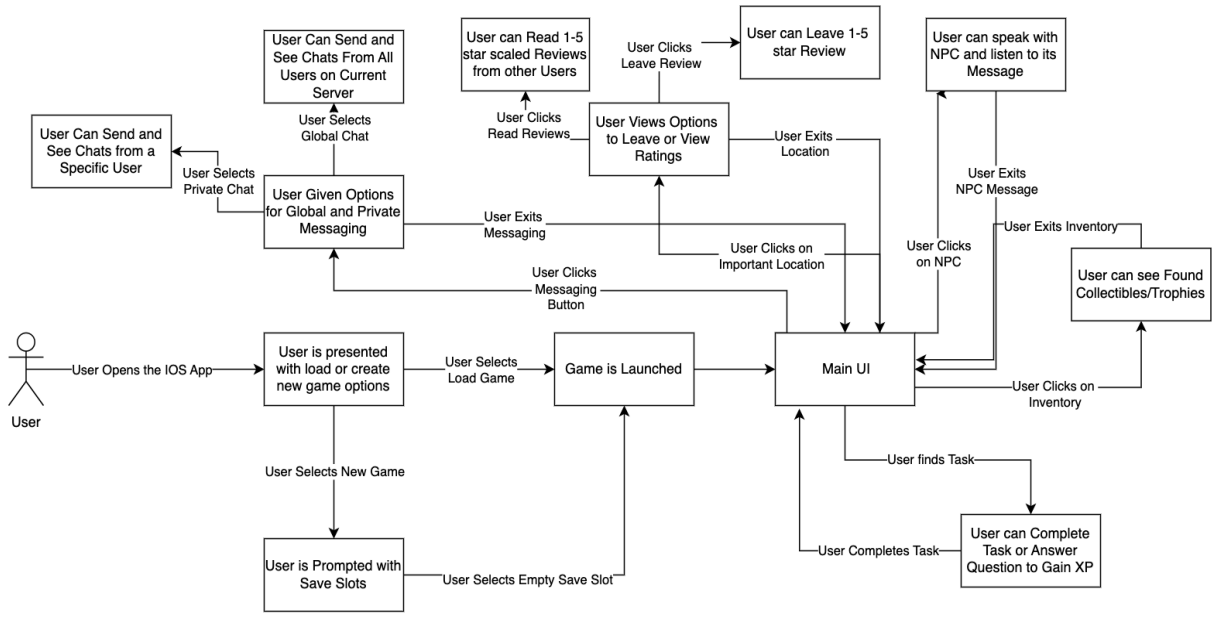


Public and private messaging



Activity Diagram

The diagram below shows the main activities that our users should be able to complete. The user will begin by loading into the app and seeing the option of being able to create a new game, or if previously played, load a previous save. They will then be spawned into the 2D campus of Purdue with movement controls and simple options. From there users will be able to move around the map to find locations, leave reviews, and complete various tasks. Users will also be given the option to view their inventory of collectibles or message other players.



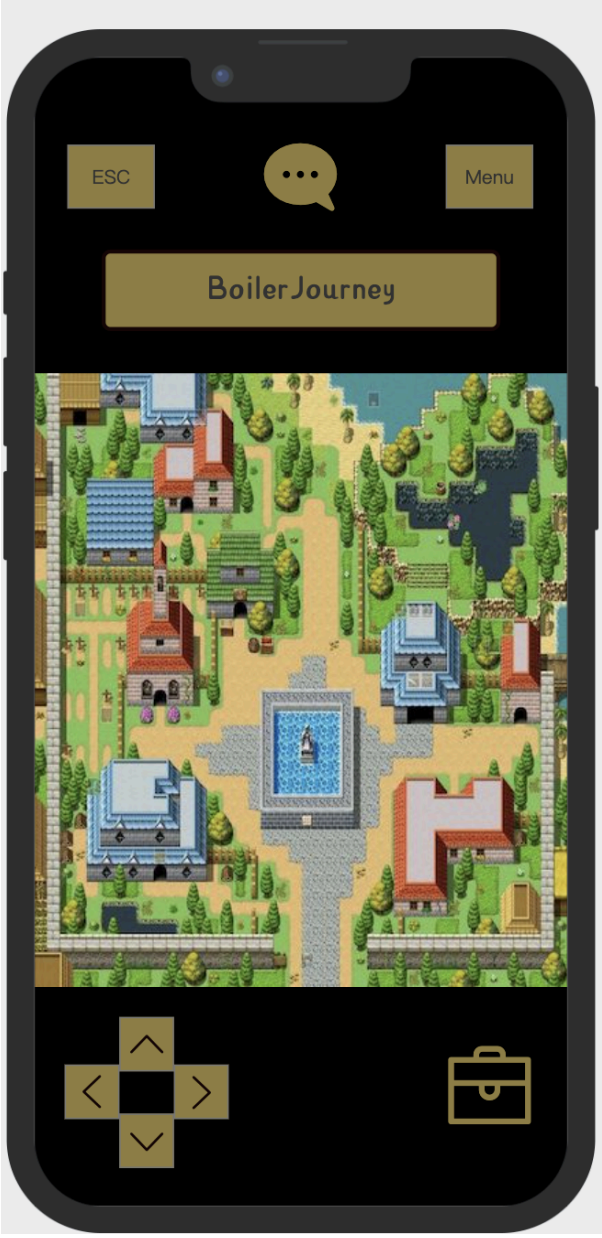
UI Mockups

Loading Screen

This mockup shows an example of a loading screen that will appear when a user launches our app. This screen will include the title of the app along with a compelling design that makes the user want to play our game.

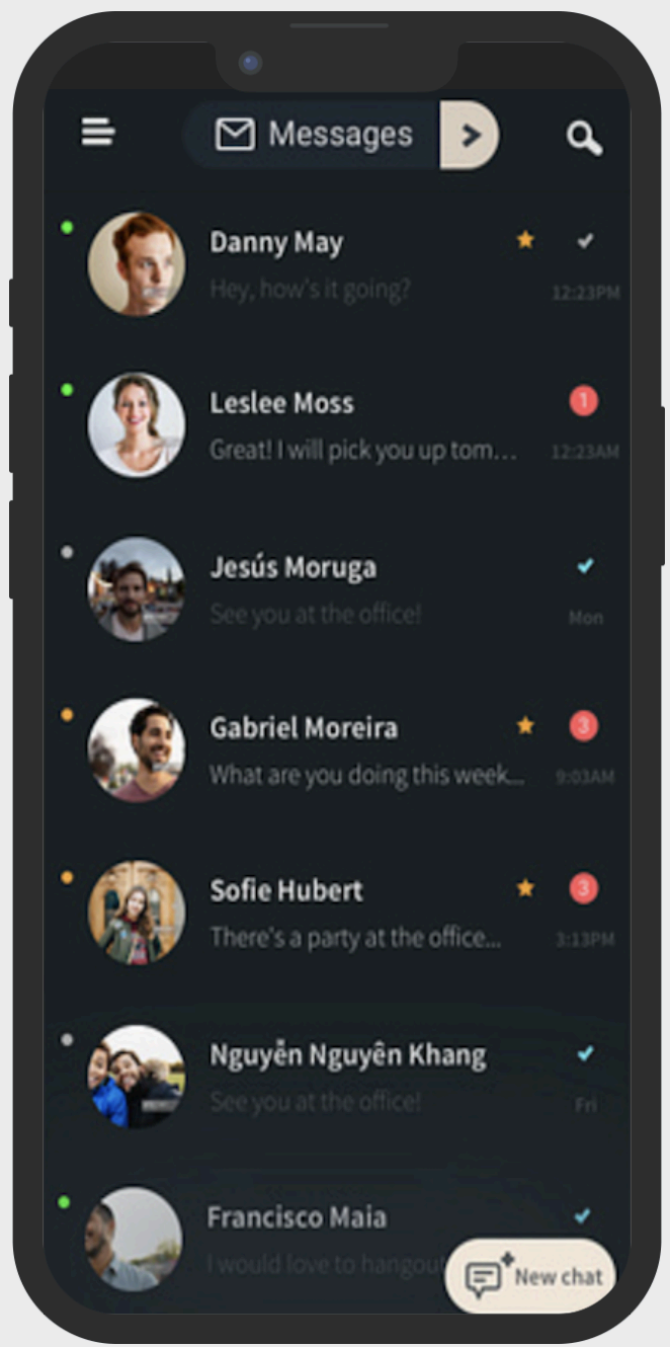
Main UI

This mockup is an example of the main interaction point of our game. This section of the UI will include the campus map that we designed, movement controls for the user as seen at the bottom of the screen, and various buttons that allow the user to access their inventory, campus information, the messaging system, and menus.



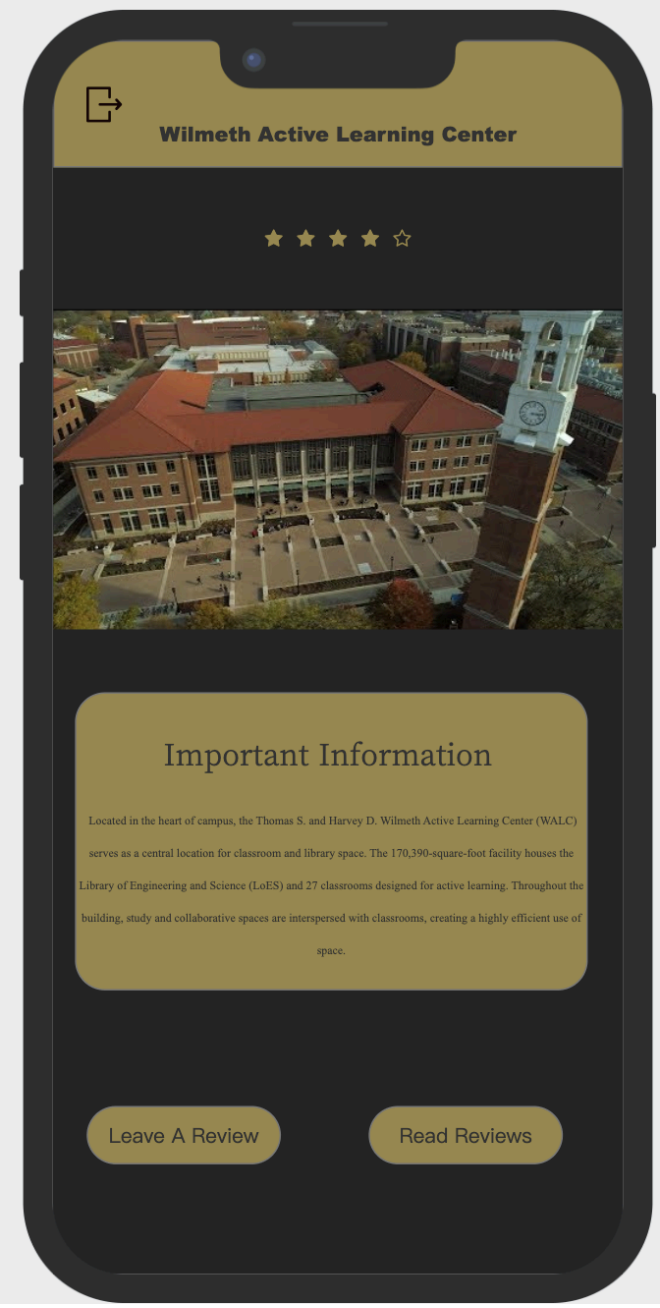
Messaging Section

This mockup is an example of the messaging section of our game. This section of the UI will include the options to send private and public messages to other players currently on their server.



Location Pop-Up

This mockup is an example of the location information section of our game. This section of the UI will include information about important locations and buildings around Purdue's campus along with reviews from other users.



Inventory

This mockup is an example of the inventory display of our game. While playing users will be able to gather various trophies and collectibles from around campus and store them. This page is an example of an inventory section where a user will be able to view what they have collected.