Team 1 - Design Document

Trey Rosenfeldt, Jonah Irons, Helen Fivecoate, Austin Rogers, Jennifer Sheng, Mohana Barve

Index

Purpose

Our goal is to create a unique gaming experience where users will be able to explore parts of the Purdue University campus and learn about different Professors from all colleges in a fun and immersive experience. Additionally, students and new teachers will be able to learn fun trivia from the classes those professors teach to prepare them for the tougher and more experienced teachers. In our 2D turn-based battle game, it is the year 3025 at Purdue University… In an attempt to keep tuition frozen, the professors from 1000 years prior have been brought back to life, but with a twist - they're now evil cyborgs (most of them). A few ways this game will differ from other games (such as Pokemon) are: a unique character selection with differing abilities/storylines, quest and achievement implementation, and mini-map functionality. Other additions to differentiate our game from the others will be implemented as we assess our timeline.

Functional Requirements:

Game Setup and Core Functionality

1. As a Player, I would like a login and account creation so I can see the leaderboards and (if time allows) go into multiplayer areas.
2. As a player, I would like to save my game and be able to log into the same location, and point of the story I was at when I exited.
3. As a player, I would like to delete my save from the game.
4. As a player, I would like to interact with a pause screen to adjust the volume of sound effects and music.
5. As a player, I would like to interact with a pause screen to adjust the brightness of the screen and contrast.
6. As a player, I would like to be able to set custom key bindings for controls in the game.

Exploration and World Interaction

7. As a player, I would like to be able to move through the map using keyboard controls (up, down, left, right).
8. As a player, I would like to be able to view a mini-map of Purdue University for easier navigation during exploration and quests.
9. As a player and Purdue student, I would like to explore where buildings are on the main campus and their names.
10. As a player and Purdue, I would like to go into classrooms and into other buildings.
11. As a player and Purdue student, I would like to be able to learn about Purdue trivia by interacting with the environment and NPCs.
12. As a player, I would like to see teachers in their office hours and the ability to see classes full during their class time.
13. As a player, I would like to go onto a computer UI to see what classrooms are full during what time of day.
14. As a player, I would like to be able to experience different biomes/regions within the Purdue campus.
15. As a player, I would like to experience different music /audio depending on where I am on the map.
16. As a player, I would like to experience different music/audio depending on who I am battling / whether I am in combat or not.
17. (If time allows) As a Player, I would like to be able to explore/unlock fast-travel locations.
18. (If time allows) As a Player, I would like to have buses around Purdue that I can fast-travel with.
19. (If time allows) As a Player, I would like to see major events on campus like the grand pre and industrial round

table, as well as things like clubs and Greek life.

20. (If time allows) As a Player, I would like to have daytime and night time and time when I can go to my classes.
21. (If time allows) As a Player, I would like to have seasons along with events happening at the correct time of year.
22. (If time allows) As a Player, I would like a calendar to see the events and what day it is.
23. (If time allows) As a Player, I would like to have to go to sleep to skip the night and if I don't sleep I get fatigued.

Character and Customization Features

24. As a player, I would like to be able to choose between different characters to fight with.
25. As a player, I would like to be able to customize the head (this includes hairstyles, facial hair, mouth structure, and eyes) and body (this includes clothes, ties, bowties, and colors of clothes) of my character.
26. As a player, I would like to be able to play different characters from a database of different pre-made characters that I meet throughout the game.
27. As a player, I would like to have a class system for my characters (e.g. general fighter, hard hitter, healer, etc.) during battle.

Combat and Items

28. As a player, I would like to battle random enemies as I travel through the map
29. As a player, I would like to be able to use different attacks in battle.
30. As a player, I would like to be able to switch between character items during battle.
31. As a player, I would like to be able to use items that enhance my battling power and provide healing.
32. As a player, I would like to be able to view my character abilities and item inventory outside of battle.
33. As a player, I’d like to be able to collect items I encounter while roaming around the game.
34. As a player, I would like to see all of the items I am able to collect and which ones I have collected.
35. As a player, I would like to be able to battle boss professor cyborgs that grant greater rewards than NPCs do.
36. As a player, I would like to be able to develop different battle strategies based on which college my enemy is from.
37. As a Player, I would like to have a HUD with things like being able to see how many attacks I have left and being able to see how much more experience I need to level up and a health bar.
38. As a player, I would like to be able to have items that can refresh the amount I can use an item or have an item that can increase the damage of an item.
39. As a player, I would like to level up and be able to use that to make my items stronger.
40. As a player, I would like to have level-ups of my items (similar to evolutions in Pokemon)

Quest and Achievement Progression

41. As a player, I would like to be able to unlock and complete quests from NPCs to unlock new content.
42. As a player, I would like to be able to see cutscenes for key moments during the main quest.
43. As a player, I would like to be able to see cutscenes for the introduction as well as a final graduation scene where you can traverse walking on the stage.
44. As a player, I would like to be able to unlock achievements as I progress through and make choices throughout

the game.

45. As a player, I would like to have a quest menu so I can see what I need to complete, as well as potential rewards for completing these quests.
46. As a player, I would like to change majors (this means having the teachers and buildings for full majors).
47. As a player, I would like to add minors (this means having the teachers and buildings for full majors)
48. As a player, I would like to go onto a computer UI to see my graduation status and when I need to do my classes.
49. As a player, I would like to talk to my academic advisor when I need new classes.

Tutorial and Learning Features

50. As a player, I would like to be able to have a tutorial experience to ease me into the controls of the game.
51. As a player and Purdue student, I would like to see what classes are required for me to graduate/finish the game.
52. As a player and Purdue student, I would like to see the classes I will take each semester and which classes I need for prerequisites.
53. As a player and Purdue student, I would like to learn which classes in specific majors are more complex and in what order Purdue requires you to take them
54. As a player and Purdue student, I would like to learn where the classes are whether that is large lecture halls or small classes.
55. As a player and Purdue student, I would like to learn which teachers in specific majors are more complex, what classes they teach, and where their office is.

Multiplayer and Social Features

56. (If time allows) As a Player, I would like to have a multiplayer area.
57. (If time allows) As a Player, I would like to communicate with online players.
58. (If time allows) As a Player, I would like to have a leaderboard about who had the quickest completion time.
59. (If time allows) As a Player, I would like to have a leaderboard about who had the highest percentage of questions answered correctly.
60. (If time allows) As a Player, I would like to see a stats page about how I did and things like how long I walked or how many items I collected, etc.

Non-Functional Requirements:

Architecture and Performance

We plan to develop the game entirely through Godot, which is an open-source game engine for creating 2D and 3D games. Godot handles all aspects of game development and design, such as character modeling, scene creation and manipulation, physics, movement, and audio, and contains support for multiplayer games, if necessary. Godot's custom language "GDScript" will be utilized for development, and it will all be done within a Godot environment.

Godot offers a lightweight development experience, allowing us to have small download sizes and efficient performance for our game - this is crucial, as reducing input delay and latency is vital to a smooth and effective experience. We aim to limit this latency to under 50ms, but striving to keep it as close to a 15-30 ms range. Godot ties seamlessly with Github, which will allow us to control the flow of our development process and monitor new changes and implementations as we work through the different aspects of the development process.

If we deploy the game on the web and/or mobile, we will utilize the built-in Godot services that are in place for a smooth deployment, which will allow for consistency across all game versions, and offer great portability for the product. Save files will be stored locally, on the user's system, as there is no need for a database to store individual player information as the game is single-player. Should we choose to implement Multiplayer, the necessary information and the game itself will be hosted through a remote server such as AWS or DigitalOcean.

Security

Security will be a strong point in our project when we are working on the login sequence and page. We will have to make sure that the user names and passwords get saved, hashed, and salted to secure valuable user information while requiring passwords to be at least 8 characters. The game will also securely save the points they were in the game. Saving the point they were in the game includes achievements, location, and bosses defeated. Along with making sure the data is safe we also have to make sure if a flood of requests comes in or if someone is trying to do specific attacks our program is prepared and can handle/timeout the user account where 5 failed attempts will trigger a temporary lock.

Usability

The interaction with the game's UI/UX is very important, as a smooth and comfortable experience is necessary for immersion to be preserved, as well as to ensure the learning experience is not interrupted. The response time of each button should be minimal, aiming to be under 40ms, and reducing lag and stuttering is a priority to ensure a proper gaming experience. Similarly, ensuring that load times stay within 10 seconds (on the high end) will maintain proper immersion. The log-in system will be necessary for saving progress so that a user does not have to restart their learning experience if they run out of time in that session. Implementing this feature and maintaining a smooth experience is vital to the performance of the product, and will be a priority during production.

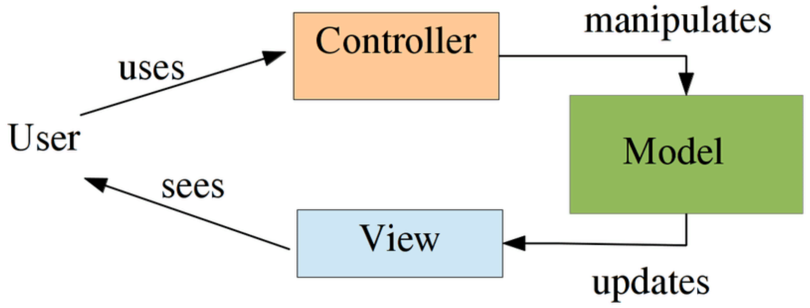
Hosting/Deployment

The game will be contained in a complete game package, which will be downloadable by interested players. The game will be set up so that it can be deployed to the web and/or to mobile, by utilizing Godot's built-in services. Similarly, if the project develops further, we will utilize Godot's provided MultiplayerAPI to support multiplayer play, and will host the website through a DigitalOcean or AWS server.

Design Outline

High Level Overview

Our project will be a video game that will allow users to explore Purdue’s campus and learn about different courses, majors, and professors. This game will work as a standalone, local execution application - this means that it will not need to communicate with a remote server of any kind, and will instead handle all game logic, rendering, and data storage locally. As our game is a single player game that is run much like Pokemon, all necessary data whether it is story related or character information, will be stored using a sort of embedded database or file system on the user’s device. This will follow the Model-View-Controller pattern, where the User uses the controller to interact and play the game, which will tell the model what to change in the game state, which is then sent to change the view of the game - it will continue in this loop until termination.



1. **User**
    a. User will utilize the controller to play the game
    b. User receives an updated game state from the View and responds accordingly

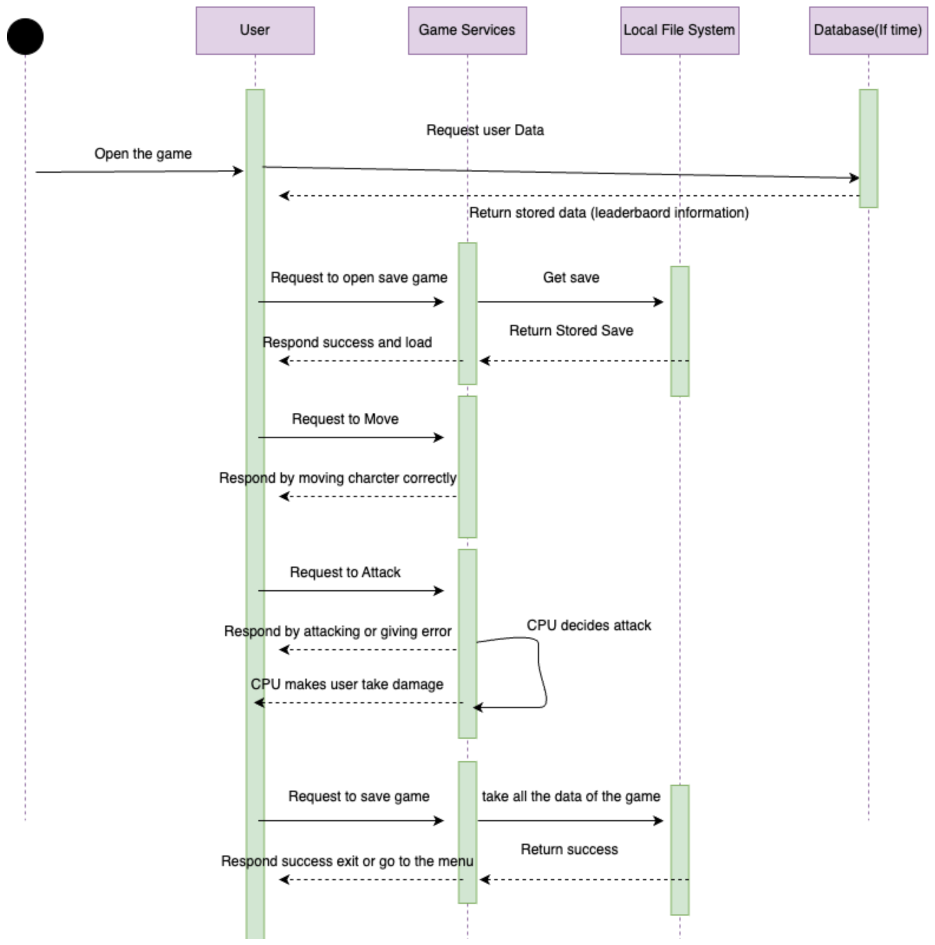
2. **Controller**
    a. Controller processes player inputs
    b. Controller sends the necessary game state updates to the Model

3. **Model**
    a. Model manages the game state
        i. Updates and retrieves user data
        ii. Updates user inventory
        iii. Updates user quest and world progression

4. **View**
    a. View handles rendering of game
    b. View handles and updates UI to reflect game state changes

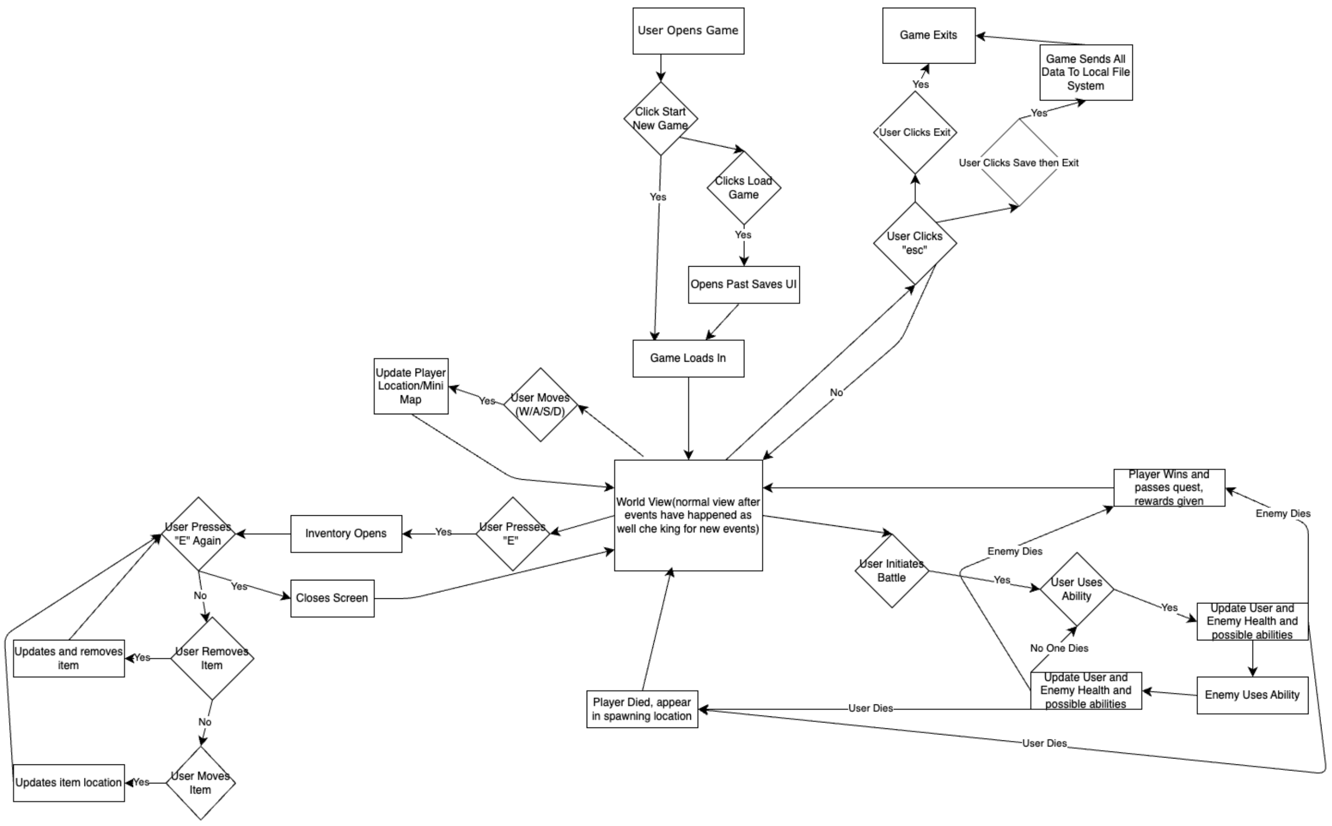
Sequence of Events Overview

The Sequence diagram below shows an example of the interaction with the User, Game Services(Player input and game logic), Local Storage (save system), Database (if time for leader boards and online content). The sequence is initiated by the game starting or being opened. As the user logs in or opens the game it will send a request to the game services to then send a request to the database to see if the user has valid information saved to show. If logged in the user then can save possible things like achievements and leaderboard stats with a database. Once the user opens the game if there is a save in the local files they can load up the save. When they are in the game they can move\explore, contribute to quests, add items to their inventory, answer trivia questions, as well as battle CPUs, this is all done by the user to the game services. From this, the user can save the game which will collect all the data that has happened and put it into the local storage.


User
Game Services
Local File System
Database(If time)
Request user Data
Open the game
Return stored data (leaderbaord information)
Request to open save game
Get save
Return Stored Save
Respond success and load
Request to Move
Respond by moving charcter correctly
Request to Attack
CPU decides attack
Respond by attacking or giving error
CPU makes user take damage
Request to save game
take all the data of the game
Return success
Respond success exit or go to the menu

User Flow Diagram Overview

The user flow diagram below shows some examples of the interactions between the user and the Game Services(this includes but is not limited to models inside the code for fighting, opening inventory, movement, sound, visuals, healing, and UI). this model demonstrates how user input transitions between different game states. Each action the player takes has to talk to game services and UI components with this flow chart trying to give a seamless look at what will happen.



Design Issues

Functional Issues:

1. What information do we need for signing up for an account?
    - Option 1: Username only
    - Option 2: Username and Password
    - Option 3: Username, Password and Email Address

   Choice: Option 2

   Justification: A username and password can give the user an ample level of security when combined with other requirements, such as a system that ensures a strong password is selected. This system makes sure the password is 8 characters or more so that brute force attacks are not a viable strategy for trying to gain unauthorized access to user's accounts. The username will also give the user an in-game way to recognize other players in designated multiplayer areas.

2. What level of customization should users expect when playing the game?
    - Option 1: Preset database of playable characters
    - Option 2: Customizable character appearance menu
    - Option 3: Customizable appearance and playstyle menu

   Choice: Option 3

   Justification: To increase game replayability and user satisfaction, a menu will be available to users where they are able to customize their character's appearance to their liking and choose between different archetypes of characters that suit their preferred playstyle. This will enable users to play the game multiple times in different ways so that the game does not become obsolete after a user is done with their first playthrough.

3. How do users gain strength while progressing through the game?
    - Option 1: Collectible items through battling
    - Option 2: By a factor of time spent within the game

   Choice: Option 1

   Justification: We want to reward players who play optimally within the game and not make it so that players have to spend hours within the game before they are able to progress to further stages. Therefore, players will be able to gain strength by using the in-game battle mechanics to collect items that will let them take on tougher challenges as the game progresses. Excellent preparation and performance within the battles will allow players to progress faster than average.

4. What options for gameplay will be available to users?

- Option 1: Main Quest
- Option 2: Main Quest and Side Quests

Choice: Option 2

Justification: The game will offer side quests that are not required for completion of the game, but can help player's abilities to complete the main storyline. These quests will also count towards completion achievements that dedicated players will be able to progress towards as they take on the main quest. This will increase interaction with the game due to players having multiple possible actions to take instead of a strictly linear progression.

5. How should we allow the users to communicate with each other?

- Option 1: In game proximity chat
- Option 2: Leaderboards
- Option 3: Chat Room

Choice: Option 2 & 3

Justification: The game will be competitive in nature so it is natural for users to want to see how they are doing compared to other players. A leaderboard allows those who play the game to check their progress against others and puts an emphasis on trying to perform well within the game. The game is also online, so a chat room will enable users to discuss strategy and in-game mechanics while not being together in real-life.

Non Functional Issues:

1. What game builder should we use?

- Option 1: Godot
- Option 2: Unity
- Option 3: BabylonJS

Choice: GoDot

Justification: Since we are creating the app with a 2D user experience Godot seemed to be the optimal choice due to its native programming language being focused towards game design and its built-in features for common problems that game designers incur during their work. It also features a relatively uncomplex user experience compared to other frameworks that have been built for games that are mainly in 3D. Godot also has built in multiplayer support so that we will be able to connect players within the game with relative ease. Finally, Godot offers a lightweight development experience, allowing us to have small download sizes and efficient performance for our game.

2. What language should we use for game building?

- Option 1: C++
- Option 2: C#
- Option 3: GDScript

Choice: GDScript

Justification: GDScript is the native language of the Godot framework so it was built with compatibility in mind. This will allow for a seamless interaction between the code and framework so that we are able to build without fear of incompatibility. GDScript shares a similar syntax to Python and is a high-level, object-oriented language. The syntax makes the language intuitive and easy to read while the creation of objects will reduce the amount of code that has to be repeated during game design. GDScript is also compatible amongst all major platforms such as Windows, macOS, Android and iOS so it will be portable for users who choose to play on different platforms.

3. Which cloud hosting platform should we use for the website?

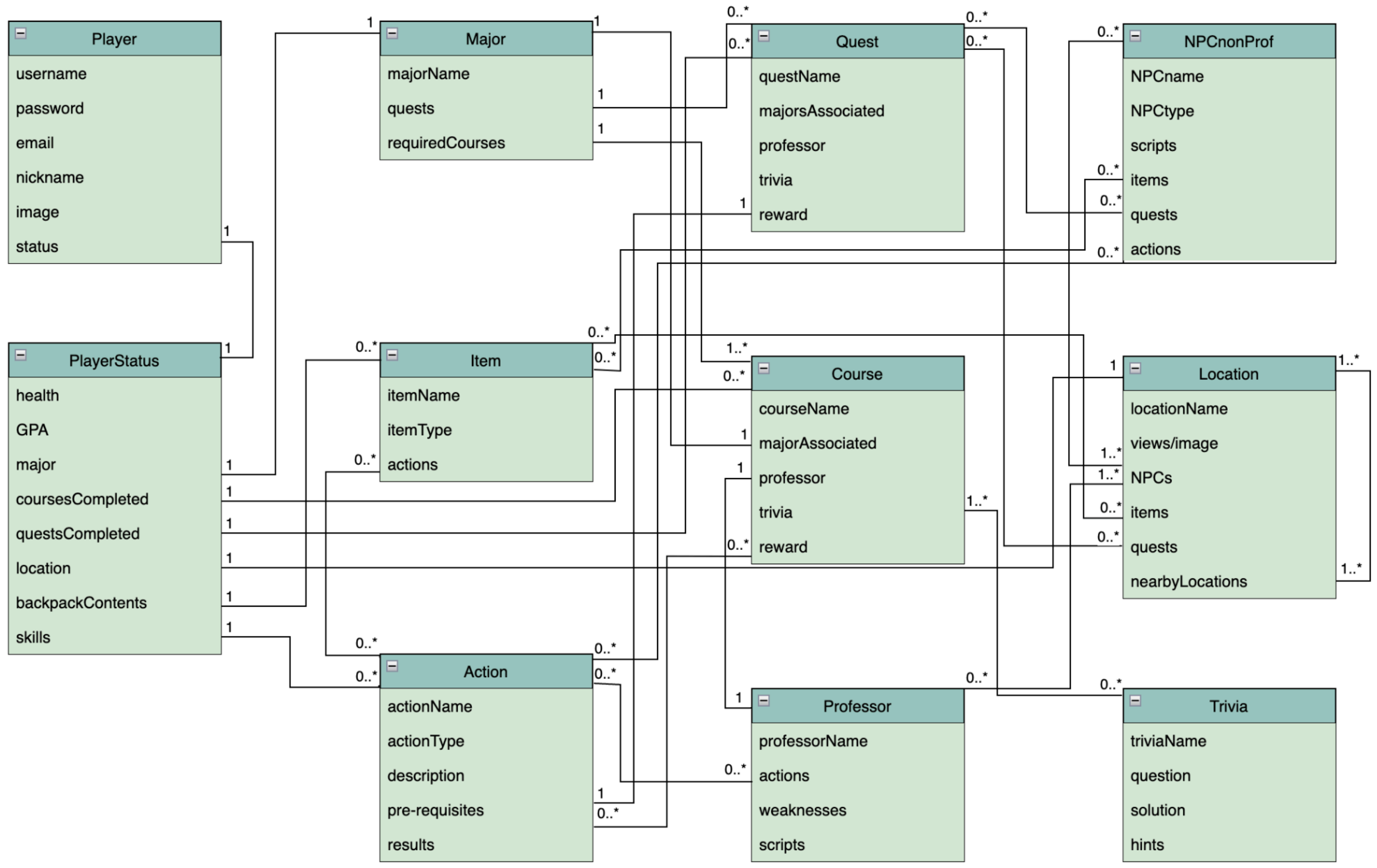
- Option 1: AWS
- Option 2: DigitalOcean
- Option 3: Microsoft Azure

Choice: DigitalOcean

Justification: DigitalOcean is a developer-friendly platform with a focus on simplicity, scalability and cost-effectiveness. It offers a simple UI with a dashboard that makes it easy to manage servers from one place. It is also cheaper than AWS and Azure with their basic servers being $4 dollars/month. The service also provides automated backups for a quick recovery if necessary and DDoS protection via Cloudflare integration incase of an attack.

Design Details

Class Design



Descriptions of Classes and Interaction between Classes

The classes are designed based on the objects and categories in our game. Each class has a list of attributes which is the characteristics owned by each object. Most of the classes will be created before a player begins.

Broad view of Classes:

- Player (username, password, email, nickname, image, status)
    - PlayerStatus (health, GPA, major, coursesCompeleted, questsCompleted, location, backpackContents, skills)
- Location (locationName, views/image, NPCs, items, quests, nearbyLocations)
- Major (majorname, quests, requiredCourses)
    - Quest (questname, description, goal, reward)
    - Course (courseName, majorsAssociated, professor, trivia, reward)

        - Trivia (triviaName, question, solution, hints)
- Item (itemName, item type, actions)
- Action (actionName, actionType, description, pre-requisites, results)
- NPCnonProf (NPCname, NPCtype, scripts, items, quests, actions)
- Professor (professorName, actions, preferredActions, weaknesses, scripts)

Detailed View:

Highlighted words indicate class interactions

- **Player** (username, password, email, nickname, avatar, status, items, location)
    - Each player will need to create an account with a username, password, nickname, and email.
    - Their username and password will be required to login next time.
    - On first login each player will need to choose their avatar as well
    - Throughout game play players will have their current status (see PlayerStatus) saved so that they can go back to their current place in the game upon reloading the game.
- **PlayerStatus** (health, GPA, major, coursesCompeleted, questsCompleted, location, backpackContents, skills)
    - In order to retrieve a players current place in the game and to display to the player where they are at in the game, we need to keep track of several factors
    - Health, GPA: As they go through the game players will need to balance both their health and GPA and keeping track of these for further game play is critical.
    - Major, coursesCompleted, questsCompleted: these show what progression the player has already gone through. Each player at the beginning will choose a major and will need to complete courses and quests related to that major.
    - Location: players are able to move around the Purdue campus and go inside of several buildings. This will save the players current location
    - backpackContents, skills: as they go through the game, different quests, courses, and NPCnonProfs (see NPCnonProfs below) will give players new items and skills
- **Location** (locationName, views/image, NPCs, items, quests, nearbyLocations)
    - In our game players will be able to move around a 2D map and enter different locations where a variety of different courses, quests, and/or NPC interactions will take place
    - Each location will have a name and will have an image that will be displayed on the screen.
    - Some locations will trigger quests or NPC interactions
    - Locations may be connected to other locations
- **Major** (majorname, quests, requiredCourses)
    - Upon starting each player will choose what major they would like to enroll in.
    - Each major has its own unique name and will have specific quests that go along with it.
    - Each major also has a series of required courses that students will need to complete in order to graduate

- **Quest** (questname, description, goal, reward)
    - Quests will be shorter goals for players to complete.
    - Quests will each have a name and a short description that will tell the player what to do.
    - They will also need to have a goal for the player to meet and a reward whether that be an item, a skill/new action, or something other.
- **Course** (courseName, majorAssociated, professor, trivia, reward)
    - Courses will be the main form of advancement or "levels".
    - Courses will each have a name and will be associated with a major.
    - Each course will have one professor who is an evil cyborg. The professor will be the main villain that needs to be beat to complete each course.
- **Trivia** (triviaName, question, solution, hints)
    - Each course will have a series of trivia questions players will need to complete to reach the final battle.
    - Each trivia question will need an identifier and the question being asked.
    - When solving these questions players' answers will need to be compared to a solution.
    - If players are struggling several hints should be available to help players along.
- **Item** (itemName, item type, actions)
    - Items will be available for players to collect from completion of quests, courses, and possibly interactions with NPCs.
    - Items will need a name and there will be several types of items such as weapons, healing potions, powerups, and more.
    - Many items will have different actions players will be able to do using the item.
    - Items will be stored in an inventory (aka their backpack) so players can view and use them during gameplay.
- **Action** (actionName, actionType, description, pre-requisites, results)
    - When battling villains players will have a variety of options for battle moves.
    - These actions will have a name and types similar to the items, so fighting, healing, powerups, and more.
    - Actions will need a description to tell players what they do and how to use them.
    - Most will be associated with completing a course, quest, or gaining an item and those will be the pre-requisites to using the action.
    - Each action will accomplish some form of task and/or change something which is signified by the result.
- **NPCnonProf** (NPCname, NPCtype, scripts, items, quests, actions)
    - Not all characters in the game have to be professors, there is potential for there to be other NPCs like other students and TAs.
    - These NPCs will have a name and type like student or TA.
    - They will need to have scripts to talk with players.
    - Some NPCs will be able to give players items and/or actions or trigger quests.

- **Professor** (professorName, actions, weaknesses, scripts)
    - Professors are the main villains.
    - They will each have a name.
    - Each professor will have actions that they can perform during battles.
    - Each professor will also have weaknesses that students will be able to learn and exploit.
    - Professors will have scripts to communicate with players as well.

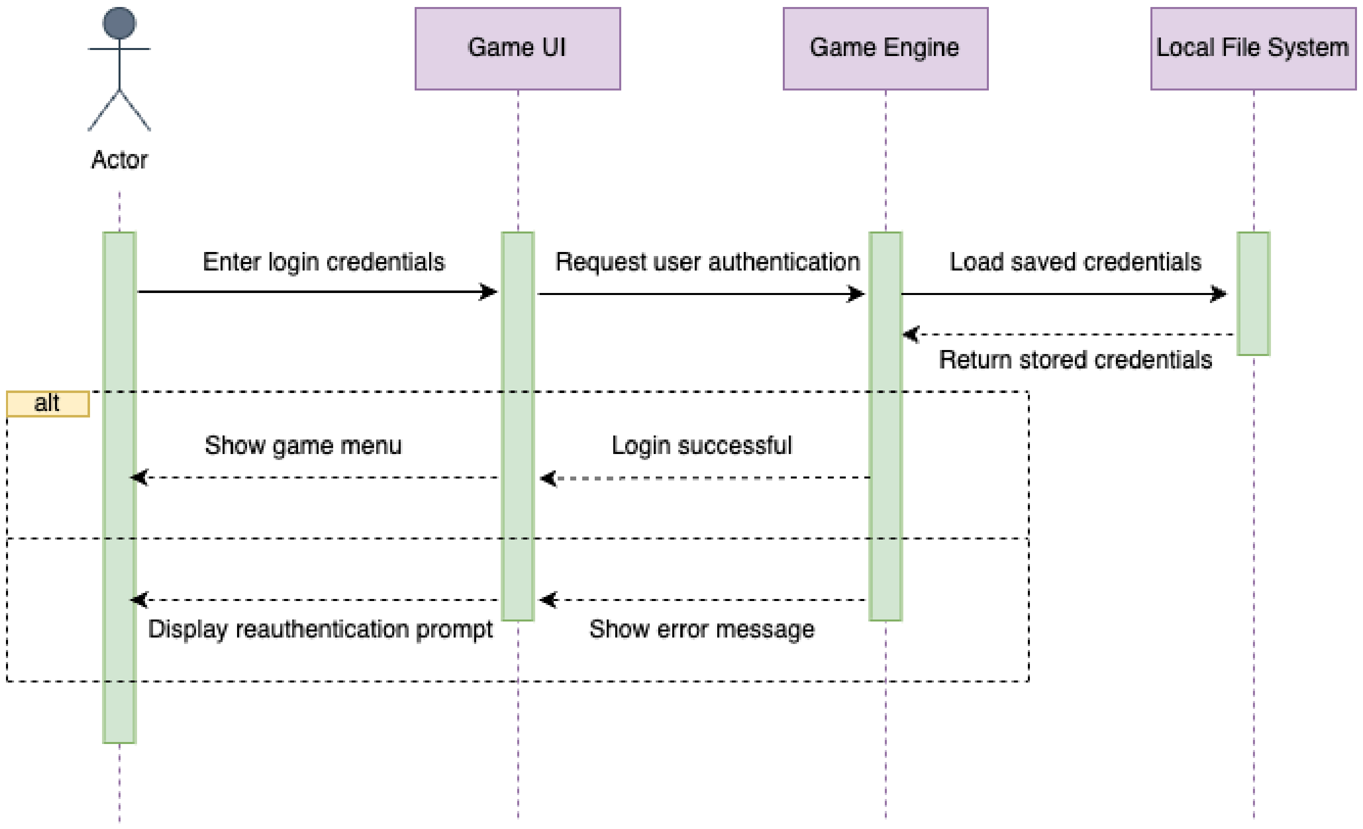
Sequence Diagram

The following sequence diagrams illustrate three major events in this game: user login, exploring Purdue's campus, and unlocking and completing quests. The sequence demonstrates how different components of the game interact within a local client-engine model.

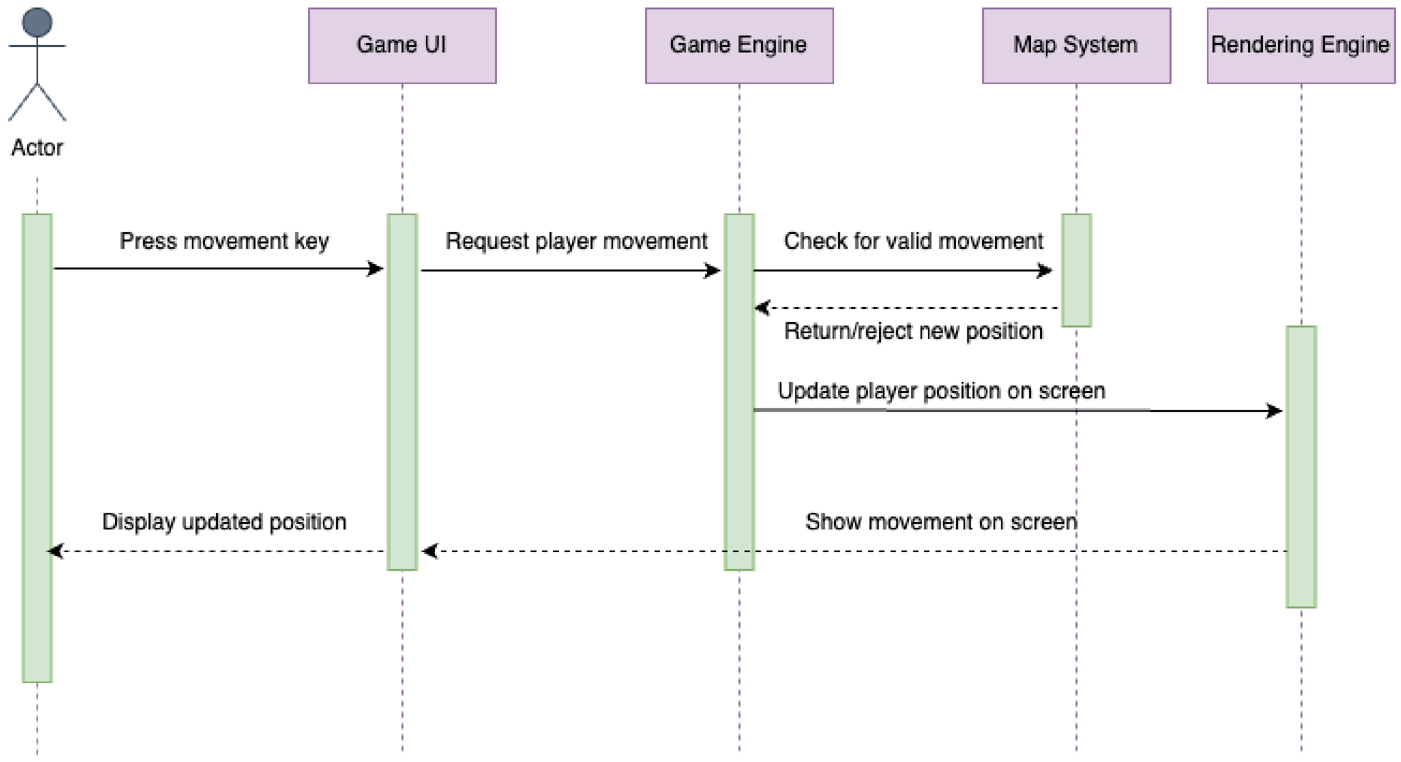
When a player performs an action (such as logging in, moving through the map, or interacting with an NPC), the Game UI (Client) sends a request to the Game Engine, which processes the request, updates the game state, and interacts with the local file system when necessary (e.g., loading or saving game progress). The Game Engine may also update the Rendering System to reflect changes in the UI. Finally, the Game UI displays the updated game state to the player.

Unlike traditional client-server models, this game runs entirely on the player's local machine, with the Game Engine acting as the core system handling all logic, interactions, and file storage.

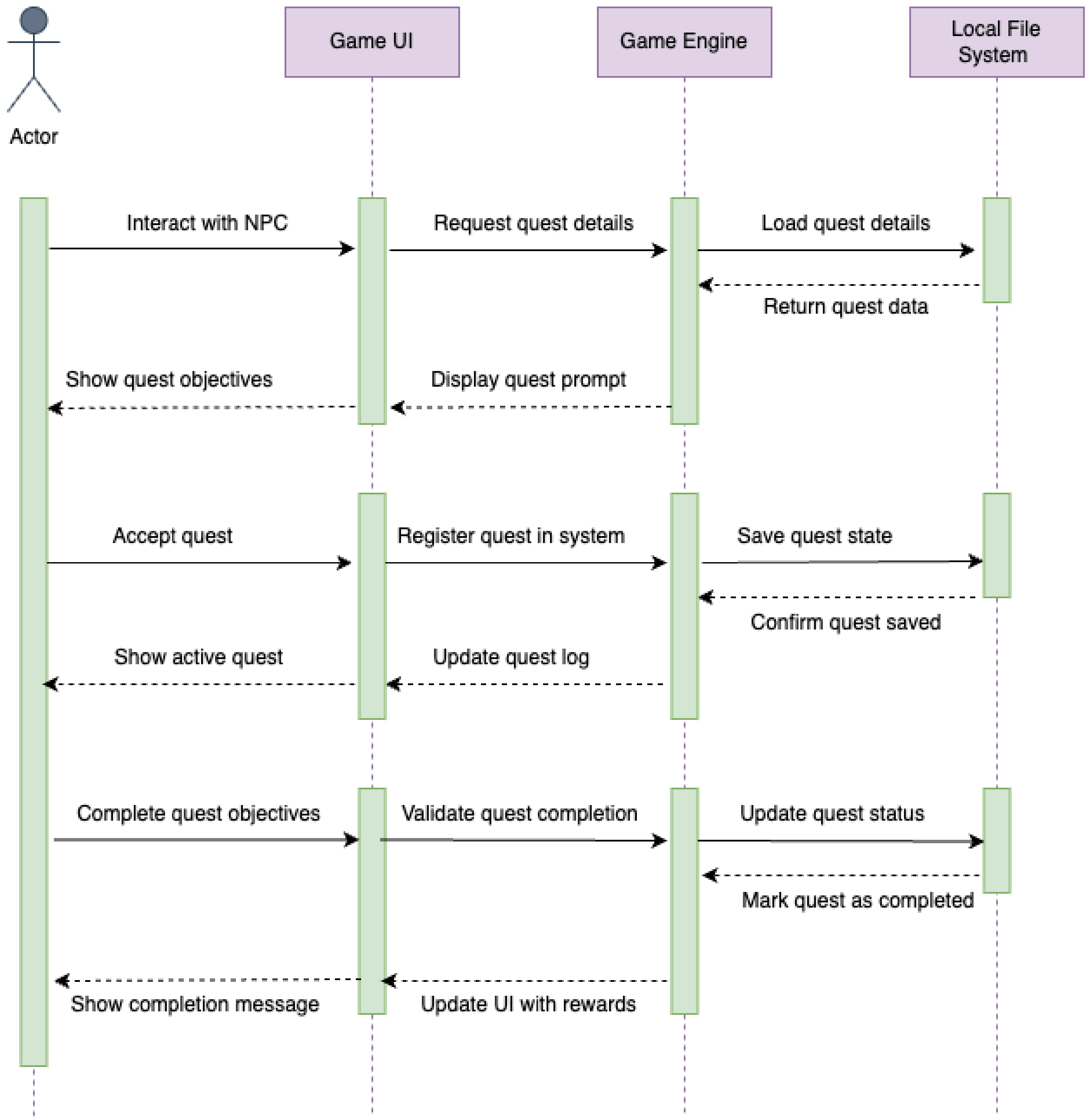
1. Sequence of events when users login



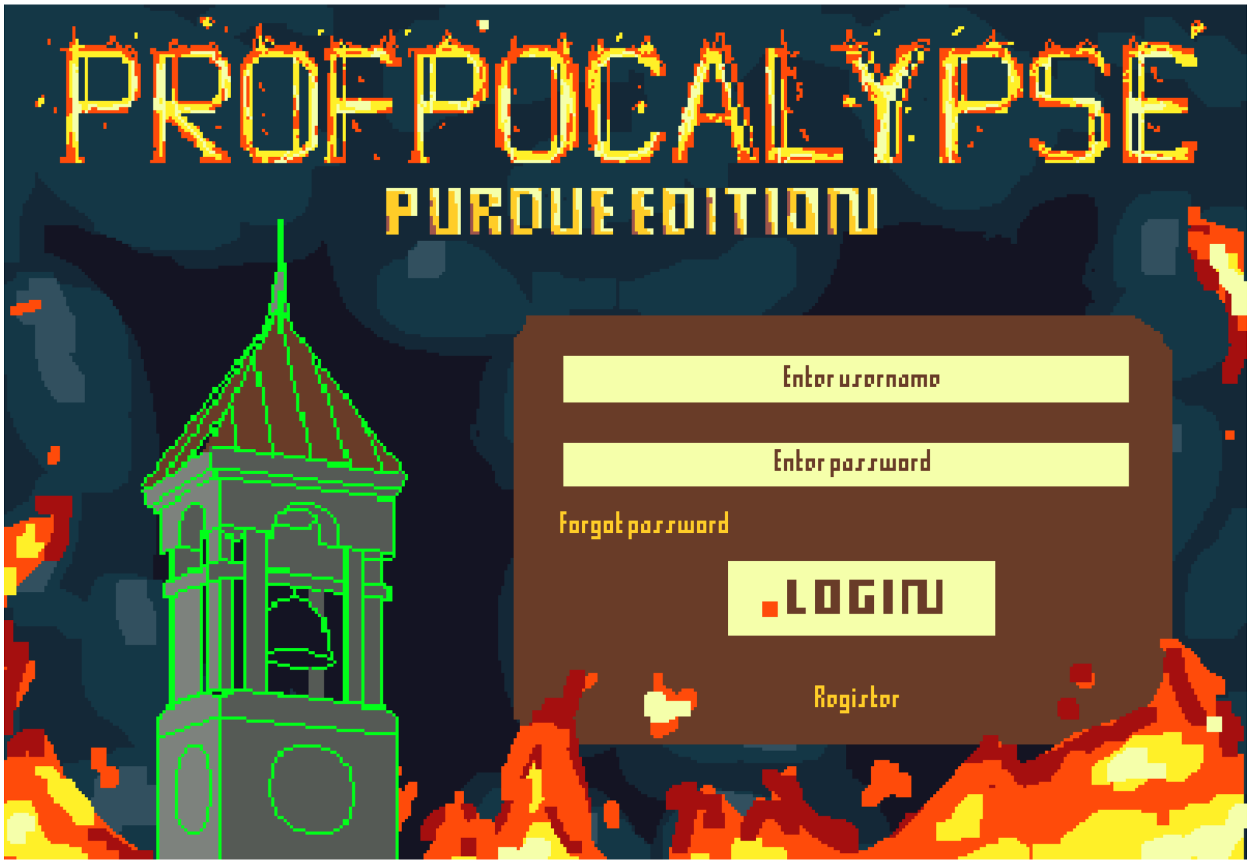
2. Sequence of events when user explores the Purdue campus by moving through the map using keyboard controls



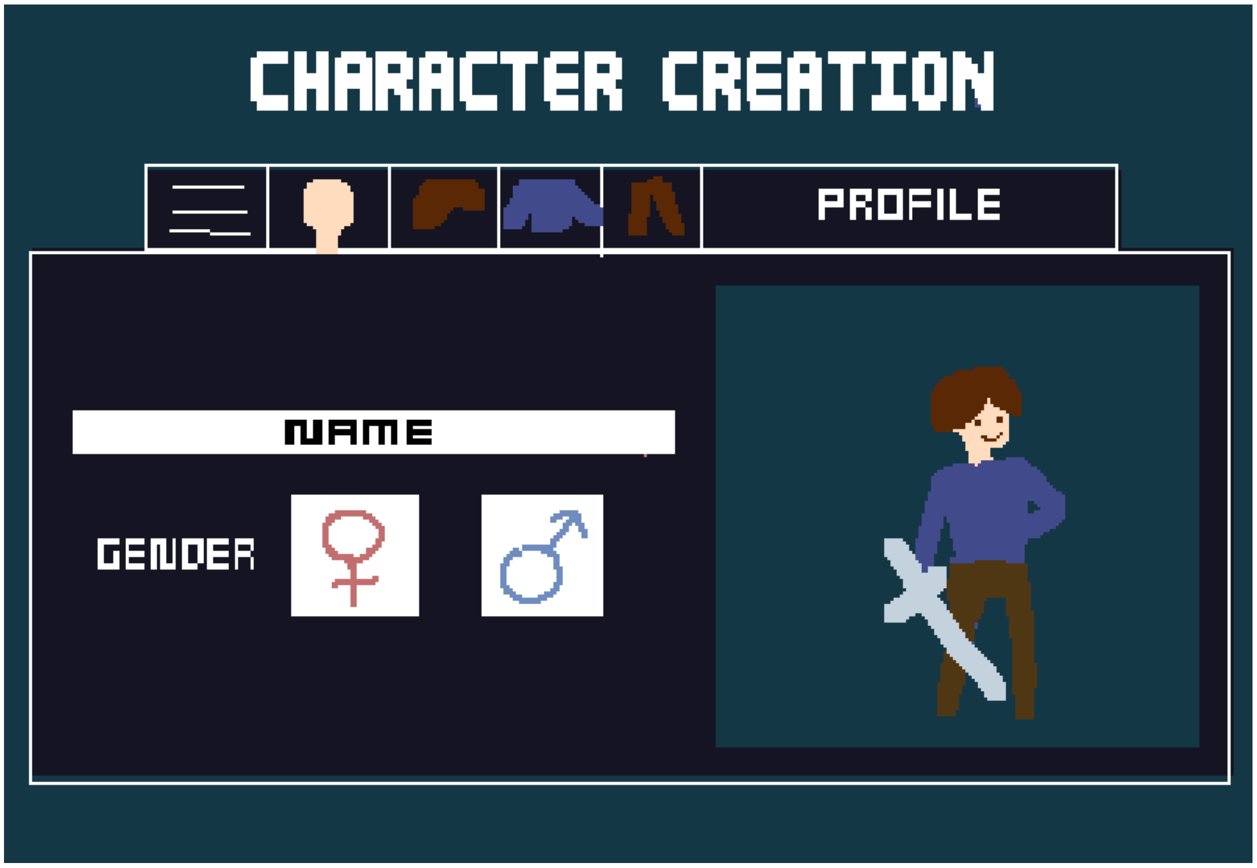
3. Sequence of events when a user unlocks and completes quests from NPC's



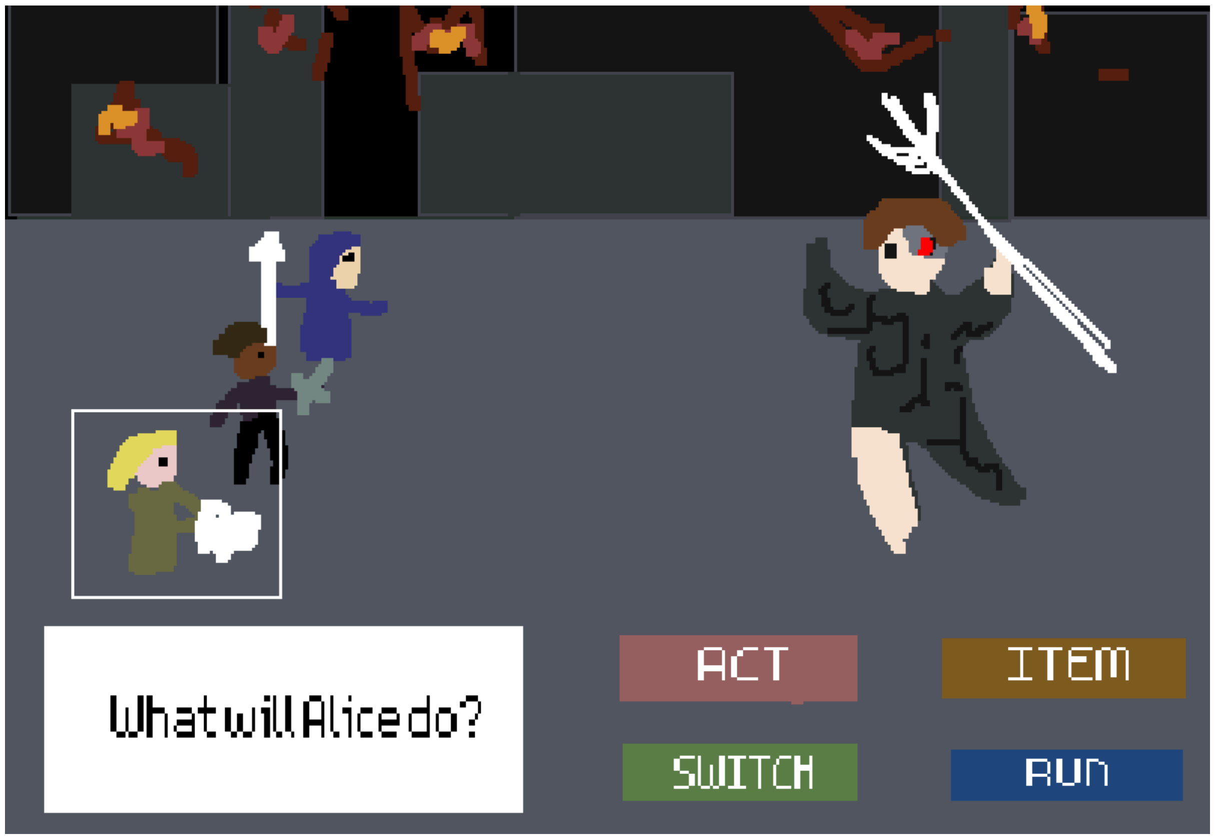
UI Mockup



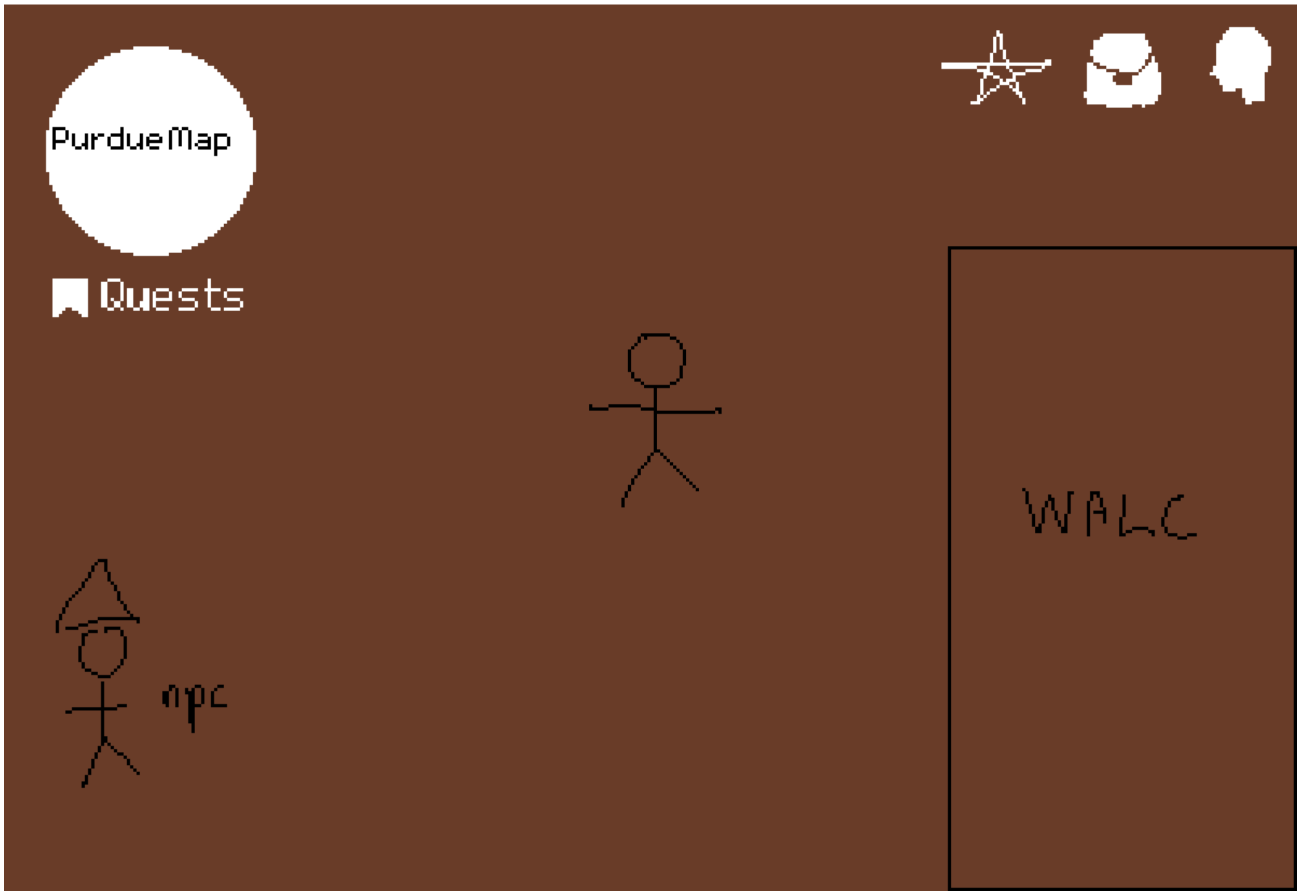
This is the login page. There is a button for the user in case they forget their password. There is a button for users to register and login.



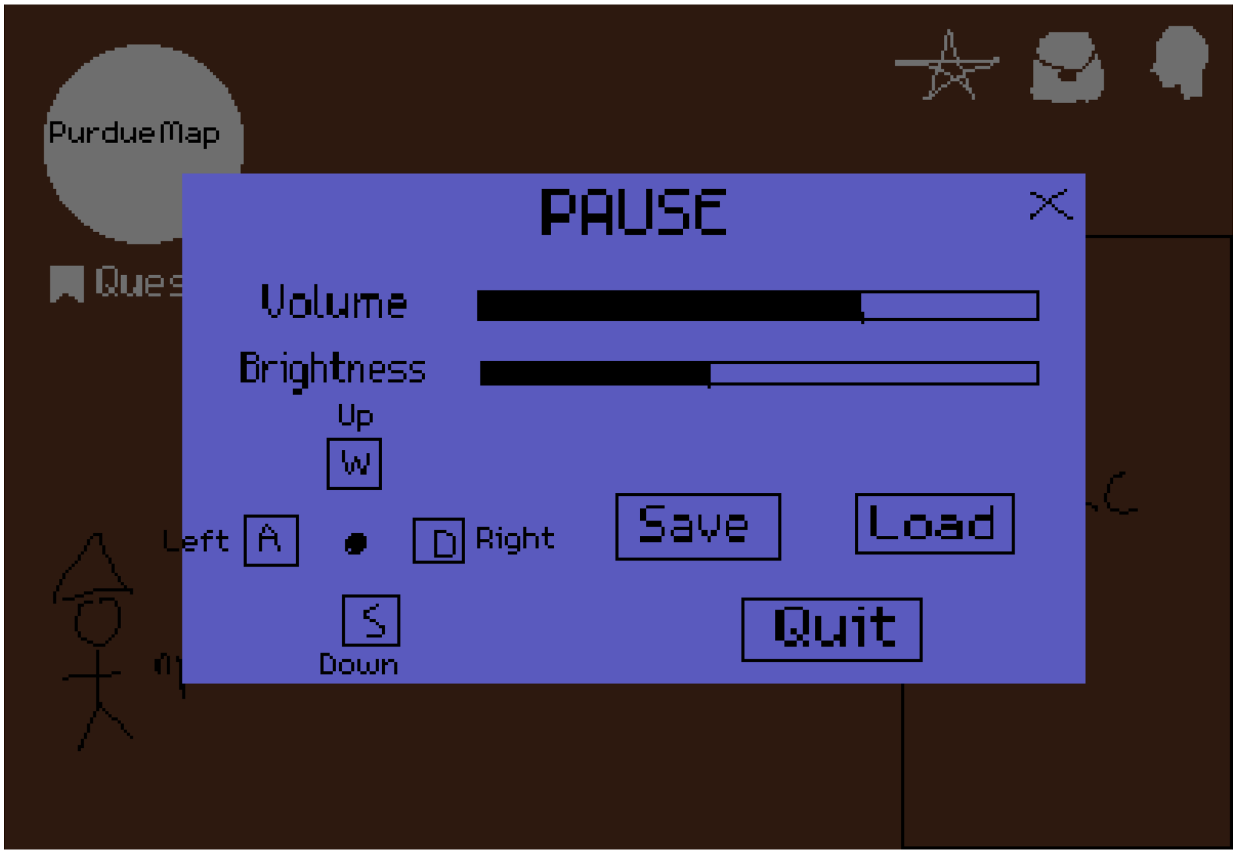
This is the character creation page. Users are able to give their character a name and gender, and customize skin tone, hair, shirts, pants, etc.



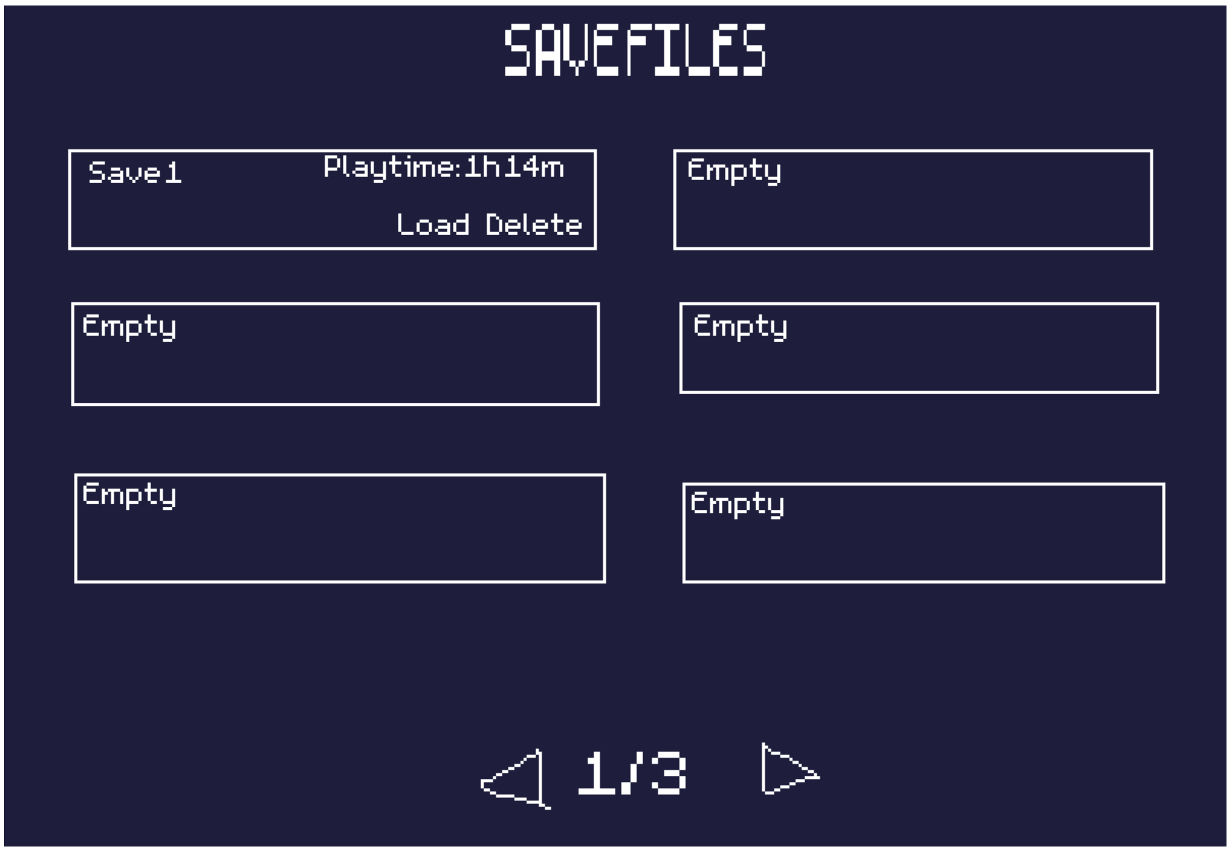
This is the battle screen. Users are able to pick between different characters during battle. Each character has different types of "acts," such as attack or heal. Users can open their inventory during battle to use consumable items. Users can also choose to flee a battle



This is the main playing map screen. The player is able to freely move up, down, left, and right. Users can enter buildings and interact with NPCs. On the upper right, users can view achievements, their inventory bag, and the different characters that they have obtained. In the character screen, the user can pick different move sets from ones that they have collected. There is a minimap on the upper left of the screen that shows a broader view of where the player is, and below is a button to enter the quests screen.



This is the pause screen. Here, users can adjust volume and brightness at any time during the game. Players can also change keybindings for actions like movement. Players can also choose to save their current game state or load a previous save file, and quit.



This is the save file page. Players can freely enter this file system any time during gameplay to save new files or load old ones. Players can also choose to delete past game files. Playtime is displayed for easier identification, and players can also name their own game files.