Requirements Analysis

Analysis of critical requirements for *StarBeasts* by Team Russian Blue for CMSC447

1.1 Introduction

The purpose of this document is to provide an outline of the various requirements that must be followed for the Russian Blue team's web game project *StarBeasts*. These requirements fall into the following categories: Business, Functional, User/Usability, System, Performance, and Quality. *StarBeasts* is an open-world web-based video game that falls into the category of RPG – Role-Playing Game. The user will be playing as an astronaut who has to travel to various planets, collect items, improve skills, and win battles.

1.2 Overall Description

The key potential features of *StarBeasts* include intuitive controls that are standard across many web-based games, such as using a map to progress through the game, character customization, item collection, opportunities to increase different skill sets, and fighting gameplay. For user classes and characteristics, the primary type of user who will play StarBeasts is a single player who will want to win the game by "leveling up" and defeating enemies. They will also want an appealing storyline to follow and intriguing gameplay mechanics.

2.1 Business Requirements

StarBeasts aims to align with the business objectives for the CMSC 447 course outlined by Dr. Nick Allgood. It shall be a web-based video game that includes at least three game levels with increasing difficulty. It shall also heavily rely on databases to display high scores, save user information, and hold game states. It shall also include two additional features that differ from the core requirements mentioned previously.

2.2 System Features and Requirements

For the software system, the game shall use API calls to both a public API and an application-specific API that acts as a middle layer between the backend and front end in which it will handle application routing. It shall also integrate databases to store critical information such as game states and high scores.

2.3 Functional Requirements

The program shall present interactive menus, such as a start menu and player menu, that allow the user to interact with the game. The system shall, when prompted, store the current game state in a local database; this includes saving the current player skill sets, health statistics, inventory, location on the map, and level. Upon starting a new game, the program shall provide the user with character customization and name selection. The program shall also present the user with a tutorial on basic gameplay mechanics upon loading the first level of the game.

While playing the game, the program shall allow the user to use a game menu that will act as the main catalyst for progressing through the story. The program shall also provide the user with various enemies that can be fought, meaning that the user can choose from a predetermined set of actions, such as scanning targets, attacking targets, and using items in the inventory [1]. Upon defeating the enemy, the program shall unlock the next level to the user; if the user defeats the enemy on the last level, the program shall inform the user that it has won the game. While fighting enemies, the system shall save the highest score dealt to an enemy in a local database that can be displayed as a leaderboard, where the user can see other players' high scores.

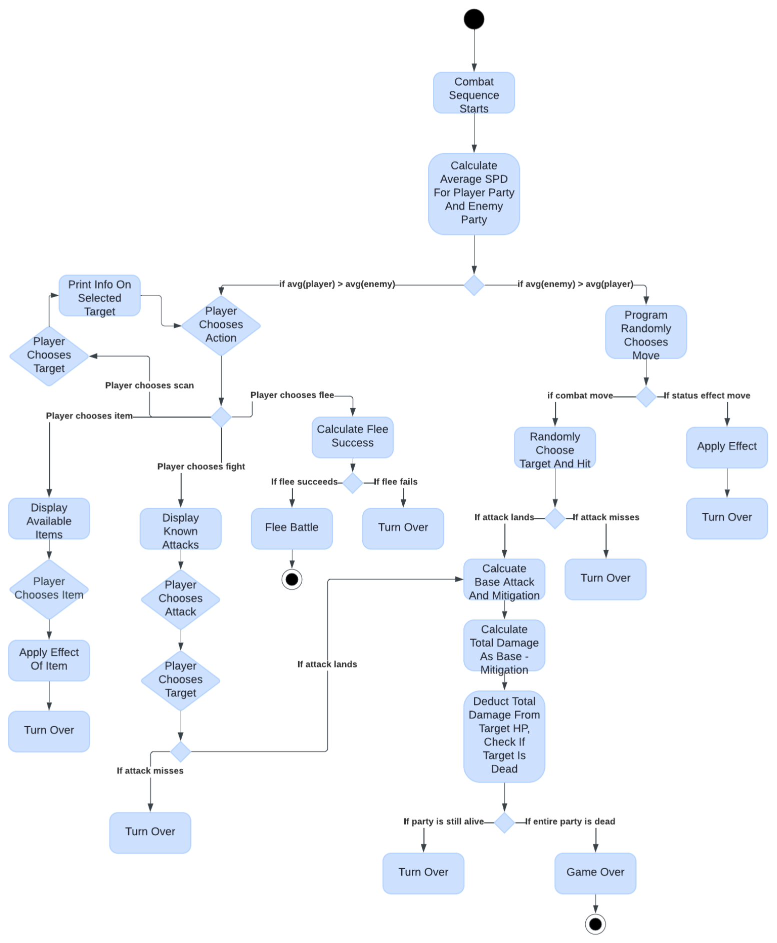

FIG. 1.
Combat Activity Diagram

2.4 User Requirements

Requirements for the user are split up depending on whether the user is starting a new game or is returning to a saved game state. For a new user, the game shall show them a start menu where they can choose to either start a new game or view the game's settings where they can edit sound, brightness, and other features. If they choose to start a new game, they shall be able to select a starter character and name. After those selections, they shall be able to load the first level and begin playing with the option of viewing a game tutorial [2].

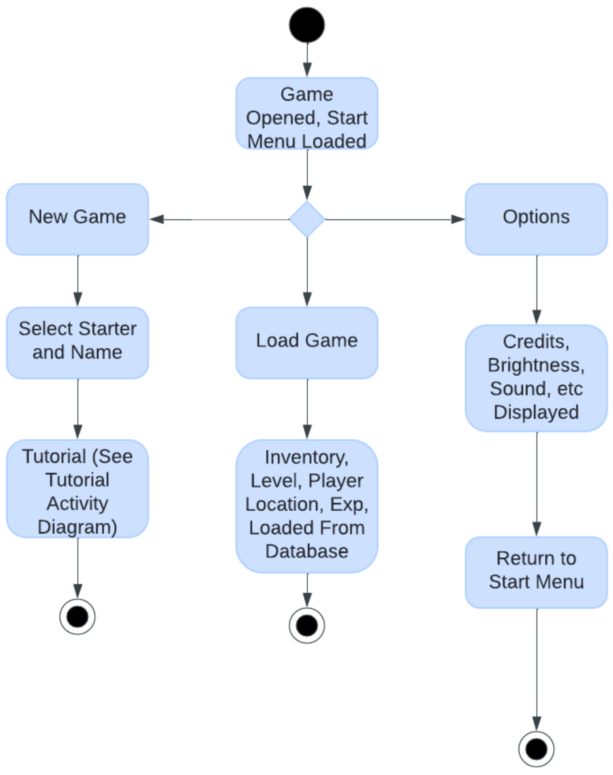
Game Opened, Start Menu Loaded
New Game
Options
Select Starter and Name
Load Game
Credits, Brightness, Sound, etc Displayed
Tutorial (See Tutorial Activity Diagram)
Inventory, Level, Player Location, Exp, Loaded From Database
Return to Start Menu

FIG. 2.
Start Menu Activity Diagram

After playing and saving a game, the user now has the option of loading a game during the start menu. While playing a game, the user shall have a player menu with the following options: "inventory," "map," "save," and "save and quit." If the user chooses "inventory," they shall be able to look through their acquired items, activate certain items, or check their player statistics. If the user chooses "map." they shall be able to view the game's map and see their player's current location. If the user chooses "save," the player's inventory, level, location, health percentage, and other player information shall be saved. Similarly, if the user chooses "save and quit," the same information shall be saved along with a redirect back to the start menu [3].

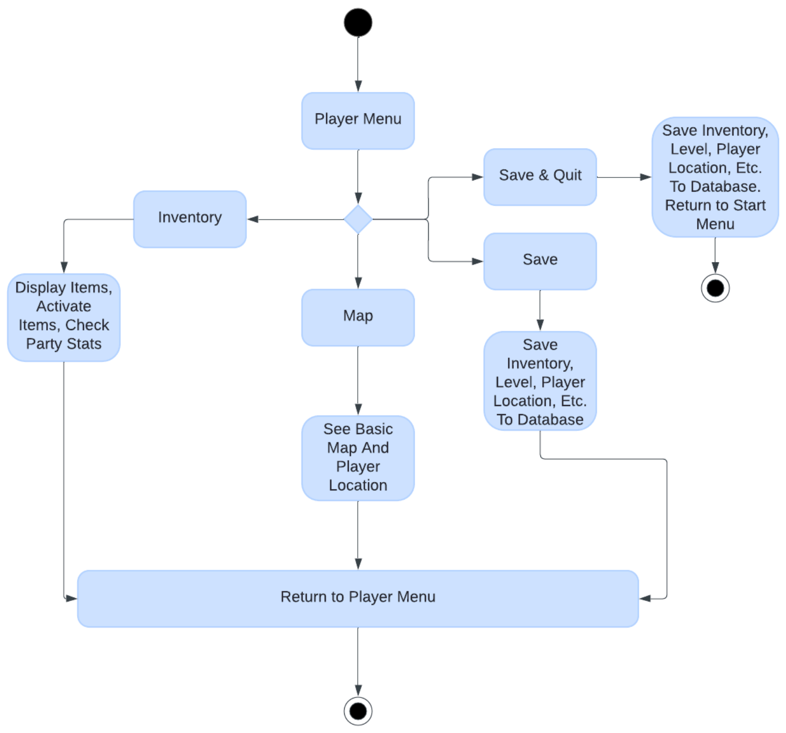
Player Menu
Inventory
Save & Quit
Save Inventory, Level, Player Location, Etc. To Database. Return to Start Menu
Save
Display Items, Activate Items, Check Party Stats
Map
Save Inventory, Level, Player Location, Etc. To Database
See Basic Map And Player Location
Return to Player Menu

FIG. 3.
Activity Diagram 3

While playing the game, the user shall be able to access all previously listed gameplay mechanics, including menu gameplay, fighting enemies, improving player skills, and advancing levels. The user shall be able to use various mechanics in fighting the enemies [1].

2.5 External Interfaces and Constraints

There are various external interfaces to be included in *StarBeasts*. The application shall use API calls to both a public API and an application-specific API that acts as a middle layer between the backend and front end in which it will handle application routing. APIs shall also be used for social features, such as sharing achievements on social media and updating a global database.

There are also project constraints to be noted. All written code shall be pushed to a main Github repository. Project updates shall be documented on the Jira collaborative platform. Certain updates shall be tied to certain sprints - the majority of the game's functionality shall be built during the project's second sprint, and the finalization of the game shall be completed during the project's third sprint. The game shall be ready for a live demo and presentation by the beginning of May.

2.6 Quality Requirements and Attributes

The program shall have simple mechanics and gameplay where new users can easily learn and navigate the game. The program's architecture shall be modular and well-documented for easier updates.