Score : 95/100

Pac-Rats Development Report Summary

Group 8 - Jonathon Repta, Amal Syed, Edward Liang, Jennifer Alonso

Design Goals:

Core to *Pac-Rats'* design philosophy is to create a game that is accessible, fast, and fun. In regards to accessibility, this philosophy is extended to several different concerns. Firstly, the game is designed to run on as many devices as possible. This means supporting a range of devices, whether it be mobile phones, chromebooks, or powerful gaming computers. Secondly, Pac-Rats is designed to be a browser based game. We choose to implement it using web technologies because it makes it easy for players to get into a game -- without needing to install any external software.

System Design:

We will use the model-view-controller pattern for the broader "site" module, and then use the "client-server" architecture for the game module. We choose the model-view-controller pattern for the site since it is a battle tested choice for building full-stack applications like the one that we're building. Furthermore, using the MVC architecture will allow us to separate concerns and reduce coupling between components. For the game module, we once again opted for a battle tested solution to the problem we're trying to solve.

To store persistent data we will be using a relational database, specifically a PostgreSQL database. In the database, we will store user information such as usernames, passwords, emails as well as information regarding map data and other persistent game information. Furthermore, aspects of this database will be accessible and modifiable by game admins through an intuitive GUI. Other persistent information stored might include reports generated, and other possible user confidentials.

Additional Design Considerations/Object Design:

The application will be decomposed into primarily two subsystems. A "game" module, and a "site" module. The game module contains all of the site's architecture and will implement the model view controller pattern, as that is a standard pattern for full stack web applications. The "game" module will be a subset of the site module and contain all of the logic pertaining to the game itself. The "game" module will implement a client server architecture in order to host games. Furthermore, the game module will utilize the observer pattern in order to synchronize views/game state between players during gameplay.

Access to premium features of the site will have their access restricted to users who meet the user class requirements necessary to access them. For example, server hosting features should

only be accessible to P2W users. Furthermore, P2W users should not see any sort of ads within the game. Access to moderation tools should also be restricted to moderators.
In order to restrict access to certain features, we implemented a security proxy. We did this by utilizing a relatively standard "proxy" design pattern in our overall application design.

Project Issues:

As with any game, there is the possibility that the game does not catch player interest, as the indie game market is quite a crowded one. Furthermore, there is the possibility that the game is marketed improperly. There is also the possibility that our game servers could fall victim to cheating/hacking, which would worsen the player experience.

Since *Pac-Rats* does not replace any existing products, there is no risk when it comes to things like data migration and so on. In addition, there's no previous version of the product to compare to. However, that doesn't mean the product has no potential issues. Since *Pac-Rats* is a free-to-play game, it will rely on things like in-game purchases, premium memberships, etc to generate revenue -- things which many users may choose not to opt into. This is especially a problem since *Pac-Rats* is an online that requires not only the initial development costs, but additional costs to keep the servers running.

Risks/Costs:

There are several risks associated with developing *Pac-Rats*. The greatest risk, as with most projects, is that it will not be profitable. Specifically, for Pac-Rats, the issue lies with not attracting a large audience of users. Since the game requires customer support as well as server and development costs, the game needs to find a monetization scheme that is able to at least sustain these costs. Due to the rise of indie games in recent years, the market has gotten larger and more competitive. Meaning we need a product that does something unique to retain an audience of players.

Following the scrum agile methodology, the game will be ready for the general public in about a year. This includes the time it takes to create game assets as well as create the website frontend/backend. Typically the cost of a low to high end indie game is from ($25,000 - $250,000). Since our game is relatively simple, we estimate development costs ranging from $100,000-$150,000.

Waiting room/Project retrospective:

In retrospect, the guiding principles we used for this project were important to its development and aided in keeping the guiding vision of the project on point. Specifically, the aims of accessibility which we laid out in the initial design documents helped to guide our application design in developing ideas for the project and for influencing the final application design.