Team Information:

Connor Yoshimoto – g4yoshi (GitHub: Connor-Y)

**Overview:**

Our application is a game sharing site. At its most fundamental level it provides a platform for users to sell or rent games to those interested in them. A common scenario for people playing video games, known as gamers hereafter, is that they accumulate a collection of games that they no longer wish to play. Typically, this is due to the fact that the gamer has completed said game but it may also be due to simple dislike. Regardless of the reason, a gamer typically builds a significant collection of these games, with the most prolific gamers having collections of upwards of a hundred games. At this point in time, these games hold little value to the gamer herself but may represent a significant value for someone who has not played the game. In this fashion, our site hopes to capitalize on the unused collections of games that people possess by allowing said people to sell or rent these games to people who have an interest in them. Our site provides a marketplace for games, allowing users to quickly and easily sell, rent or purchase games.

**Design Architecture:**

Our application uses a three-tier architecture dividing between ‘Presentation Logic’, ‘Application Logic’, and ‘Database Logic’. Presentation logic contains what the user directly interacts with, specially the web pages, and then sends submitted information or requests for data to the Application Tier. The Application Logic contains all of the server side logic. Its job can be divided into six major steps.

1.) To receive requests from the Presentation tier,

2.) To process those requests

3.) Send necessary requests to the Database tier

4.) Receive a response from the Database Tier

5.) Apply logic to the received data

6.) Send a response back the Presentation tier.

The final tier is the Database tier, which is only used to interface with the database itself. It contains various queries for adding or retrieving information to a separate database.

For this project, everything found in the ‘public’ and ‘views’ folders are the ‘Presentation Logic’.

These folders only contain html, it’s styling (.css, .hbs) and a small amount of associated javascript (.js). These are the pages that the user sees and directly interacts with and are responsible for displaying information and receiving user input. The presentation tier then passes on any requests to the ‘Application Logic’.

The ‘Application Logic’ is contained with the ‘server.js’ file. The server is responsible for serving static files, front end validation, password encryption/decryption, logic for features (search, recommendation etc.) and deciding information from the Database Tier to get.

The ‘Database Logic’ is concerned solely with interfacing with the database. After receiving a request from the Application tier, it uses a relevant query to obtain the correct information from the database. It then sends this information back to the application tier to be used.

The reason we chose this design is to allow flexibility in changing part or all of our code. Separating the Database logic from the Application logic allows us to switch between various databases without the need to rewrite any of the application logic. Likewise, we can implement new features for the application without having them interfere with our database logic. This same pattern can be applied to the presentation tier and the server tier.

**Security:**

**Optimization:**

**Video Link:**