GameBook Design Document

CRC Cards

User SubClasses : GameDeveloper, Admin	
Has Email Has Password Has a status(“verified,admin”) Knows Friends Knows Favorited Games Knows Posts Has a picture Can login Can make posts on forums Can add friends Can write reviews Can like games	Review Post Game

GameDeveloper SuperClass : User	
Can request to add game to Listings Can edit game info for Games which they are apart of	

Admin SuperClass: User	
Can perform CRUD operations On DB Can moderate posts Can ban/restrict users Can verify User to Game Developer	

AuthDAO	
Creates a new user, given username password and email Gets a user, given correct username and password	User

DataBaseDAO	
Adds a new game given Information Update an existing game info With fields Delete a Game from database	Game

GameNewsDAO	
Dynamically scrape Websites to find game news Parse data and return A JSON response	Game

Game	
Has a picture Has information Has a description Has reviews Has a title Has a rating Has a release date Has a publisher Has a genre Has List of Users Who Play it	Review User

Review	
Has a user Has a date (TimeStamp) Has a description Has a rating Has number of Likes	User

Post	
Knows the user who posted it Has a post number Has number of Likes Has a TimeStamp	User

Forum	
Has Topics	Topics

Topics	
Has Threads Has a Title	Thread

Thread	
Has a Headline Has a TimeStamp Has a User Number of Posts	Posts

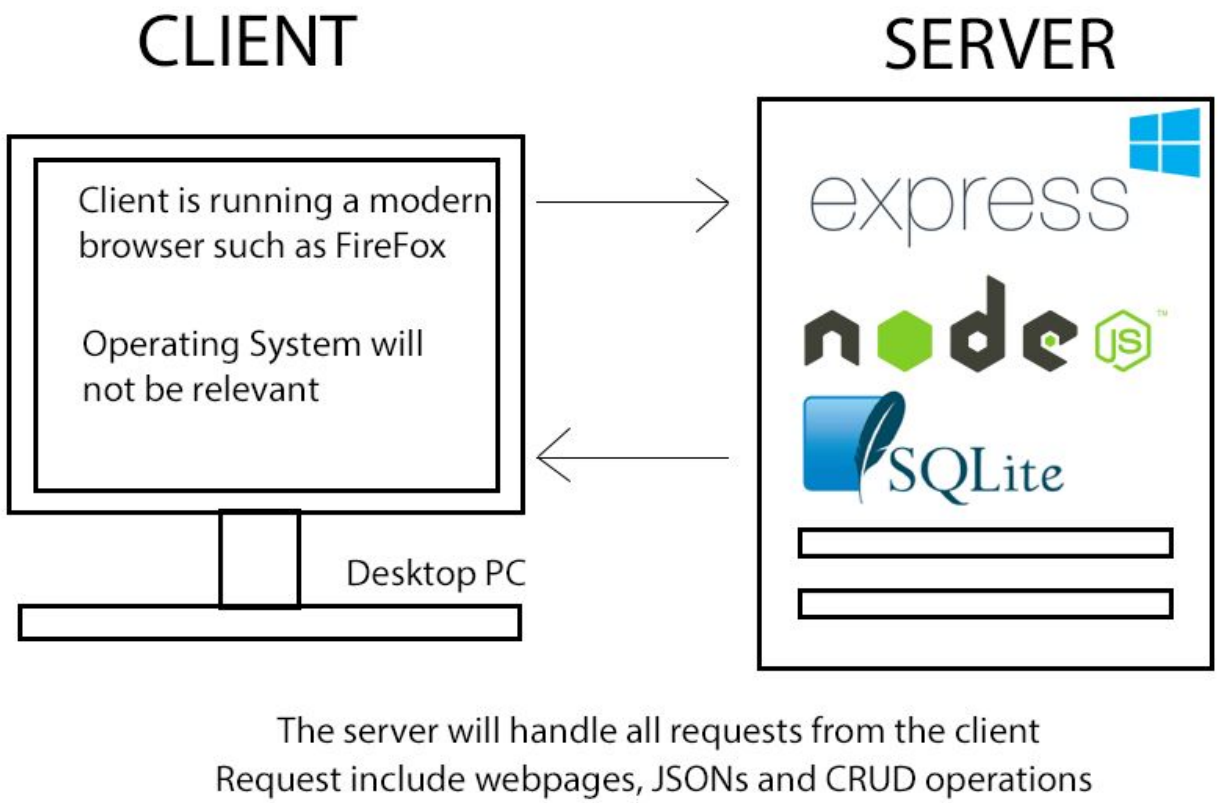
System interaction and Environment

The technologies that we will be using are the following

- Server: Node.js, Express
- Database: SQLite
- Front-End: HTML, JavaScript, CSS, VueJS

The server will be running Node.js 6.11.0 with Express 4.15.3 on a windows machine.
Most clients are running very popular and standard web browsers (Firefox, Chrome, Edge, Safari) and are capable of running JavaScript and HTML5 and therefore should be able to support our front-end technology.
We are currently not designing based around mobile browsers but support for it is on the table for future releases.

System Architecture



The server will handle all requests from the client
Request include webpages, JSONs and CRUD operations

Component Decomposition

The three parts of our software architecture are the client, server and database. The client is Composed of all the different web pages on our website, the server handles request from the client by making queries and performs CRUD operations on our database. Also, our server interacts with external API to get information for the client. The database contains information regarding users,games and the discussion board.

HomePage:

The homepage will contain information about popular games, latest games and latest gaming news. To get all this information, the client will make several ajax requests to the server on load.

To find latest gaming news the request will be processed through the gameNewsDAO and then a JSON containing {headlines,links} will be returned.

Both popular games and latest games also make an AJAX request which the server processes by making a query into the server and then returning a list of rows.

UserProfile:

To go onto the user profile, first the user must use the AuthDAO to log in. After logging in, the user can view and update information from the *user* card and can Perform any of the actions specified.

Friends:

Users will be able to view/add/remove friends which will be kept track of in the user profile and the database. Both users must give consent for a friendship to be formed.

Games:

Games are updated, created and edited through our gamesDAO and are viewed by users through the game information sheet. On the game information sheet Users can add *reviews* which we add to our database.

Forums

The forums view is composed of all the the cards *forum,topic,thread,post* all of Which are kept in our database.

Errors

In case of http/https request errors, for post requests we will redirect The user to an error page and ask them to try again. In case of get errors We will attempt to re- request for a certain period of time then, we will Alert the user if it fails after a certain amount of time. For some get content such as Images we will hold a default state which will be shown if an error has occurred.

To avoid browser incompatibility we will handle by testing our website with a variety of browsers then refactoring the portions which may cause problems.