**READ ME**

Summary:

I have created a server that based on the following assumption:

* It’s a server designed for MOBA game which is high-paced, synchronous and high interaction with players. Player controls a single character with real-time strategy(Ex. League of legends)
* A database would store user information such as unique username, email, hashed password, user Id, create data and cetera.
* Achievement and statistics schemas are used for user records. After each game round, the server would update the game statistics and achievements for each user automatically.
* Users and developers have access to the server with different authority. Users are limited to get user public information, game statistics and achievements, while developers are allowed to use the server to create, post update and validate under certain circumstances, such as if a user is cheating in the game or being reported for several times, the developer is allowed to delete his achievements after validation

Architecture:

Macintosh HD:Users:wenluying:Downloads:classical game deployment architecture (3).png

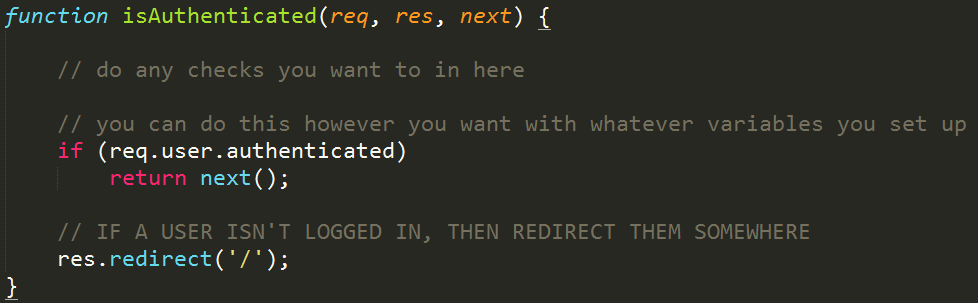
I plan to use classical game deployment architecture to handle the MOBA game. Several servers are involved to handle different tasks and all data is stored in one database. I also plan to use cache in each server to store data for a certain period then post it to database. Check details Pseudo part 3.

Pseudo:

1). MOBA game is based on role play and user can pick any character in each game, so there should be a character table(Champion table in my case) for each user. Due to time limit I haven’t implemented the API for this table, but user and developer should both be allowed to grab information from the server. This is of great importance in MOBA game because users would check their game status from time to time and they should be permitted to check how their teammates performed in the past games. As well as developers and third parties can use the server to generate data and do post-game data analysis.



2). Since this server is exposed to all users, it should add a middleware authentication to prevent players post and update any game statistics and to avoid cheating to some extent. We only allow admin or game server to update the data when they passed the authentication.



3).Classical game deployment architecture should be deployed for this MOBA game. we should actually build several servers to handle based on different functionality. In the real world the web server is consisted of several servers, like payment server, game statistics server and achievements server to handle huge data interaction. For instance, a matching server is used to match different players and create a suitable virtual game server then it would handle the game data. We certainly should cache(write-back cache) internal data for each game so that we can reduce the number of DB transaction to several magnitude and improve performance.

4) Data validation. We need to make sure each data post to the server follow the certain rules, such as it’s not allowed to post both win and lose to certain user. We can create a rule-based model to verify data.

5) Add Friends. This is an important aspect in MOBA game since the platform provides more chance for players to know each other and become friends. A friendship is really a two-way relationship; each entity is linked to another. I would like to store as an array of user\_id and username in the User schema.

A few notes:

1). In order to protect user’s information, I only return public information and hide the password.

2) Referring to password, I salt hash passwords using Node.js Crypto. Salt hashing is a technique in which we take user password and a random string of characters, called salt, hash the combined string with a hashing algorithm and store salt and hashed password in the database. This is a one-way process, which is quite safe to some extent. And this technology can avoid storing the same-hashed password in the database when different users have the same password.

3) Regarding to database, I chose non-relational database for several reasons. Firstly, tradition DBs can’t efficiently store heavily structured hierarchical data. For each character, we could build a custom object database and update the schema easily. What’s more, in the real game word it’s more likely that each character is document stored. Secondly, MongoDB provides more operations than SQL. Last but not least, since MOBA is real-time strategy game and allows millions of users to play simultaneously, NoSQL is designed for high concurrency.

Tech Stack

Framework: Express

Database: MongoDB

Language: Javascrip

QA: postman, mocha

To run the server:

* Install mocha: npm install mocha –save-dev
* Run database:

mongod –dbpath /path-to-folder/myapp/node\_modules/mongodb –port 27017

* Run application: npm start
* Run test: npm test