Web and Database

Awesome Spel uses a Web Server for some of its online features. The features mainly enable the player to keep track of their progress online, although offline play is possible. The following features come together to create our online save game platform:

* A secure user management system
* Robust Server Side scripting with error reporting
* MySQL database technology
* Item and Location tracking

These features are implemented through the use of three individual modules:

1. A MySQL Database Server
2. Client Side Saved Game Management and Update calls
3. Server Side JavaScript to connect the two together.

**The MySQL Database**

Our database uses modern, secure techniques of storing data. When an user is created, they are assigned an unique ID used to identify their actions and entries in the database. They receive an e-mail to Welcome them to the game and to give some general information. For secure storage of their passwords a pseudorandom 16-byte salt is generated that is then added to the password before hashing it with the Secure SHA256 hashing technique. Furthermore, every time the user logs in a new salt is generated and the password is rehashed to prevent the usage of rainbow tables even more. We use prepared statements to communicate with the database, to reduce the risk of SQL injection attacks. The database uses Indexes to make sure certain combinations of factors are always unique, for instance allowing the same player to only have any given item once. This further prevents any mishaps down the line.

**Client Side**

The client keeps track of a Saved Game file for storing the current player location and the items that are picked up. This Saved Game is also relayed to the server. When a player logs in the most recent data is acquired from the server. This is then written to the saved game, to allow local access and reduce the amount of requests the server has to process. Whenever the player enters a landmark or quits the game, their in-game location is send to the server. The same thing happens when a piece of evidence is picked up. The client side also allows players to create an account. The entered data are checked for formatting, such as e-mail addresses being valid and passwords matching client side. After that initial check the data, if valid, is relayed to the server where additional checks are run.

**Server Side**

The server uses JavaScript running on the Node.JS platform. We use the Express plugin to communicate with the client, although no static or dynamic content is actually served. We only transfer raw game data to and from the client. The server is built to keep running, which means we check our input and return errors is something is wrong, to both the client and the server logs. This can range anywhere from usernames already existing or passwords being incorrect, to incomplete or incorrect data being entered. Our system is the only system that provides a degree of security, as the permissions were set to enable ‘anyone’ to read data in the folders on the server by default, by the system administrator. We changed this to deny read access and protect our source code and the passwords inside it.

**Room for improvement**

There is certainly room for improvement in this area. As far as security is concerned, the username and password are currently being send over using the plain HTTP protocol, however using HTTPS will greatly reduce the risk of data being compromised. It would also be better to use the hashing function several times, to make it even harder to pre-compute hash values. That being said, our system is secure enough for the purpose of a small player base and a server that will not be up indefinitely. We don’t expect to attract malicious individuals.