Library Front End Development Outline

Intended Result:

* Create/Implement several user interface scripts that will perform certain functions for users.
  1. Create and Manage a user auth list
  2. Provide a lore timeline of the world and user events.
  3. Access a list of best user tools/scripts for specific uses.
  4. Serve up a list of services publicly provided by other users.
  5. Store important memes created by users.
  6. Provide a puzzle to solve.

**What is Front End?**  
 The Front End of the ARP Project is the abstract area where all direct connections by users and the handling of their requests take place. It contains the users and scripts that provide page generating and serving, and communicates to the Back End regarding DB Queries and authentication.  
  
 It does NOT provide direct access to databases / sensitive information, nor does it store GC, high-level upgrades, or sensitive information inside databases (such as authentication information, user info, authentication tokens, etc).

**How Does It Work?**

All users that are part of the front end are either T1 or T2 users, and have solely character and script upgrades loaded. Their gc / activity logs are to be air-gaped from Back End servers by a single user that can easily be replaced if breach attempts occur.

All services should be able to be routed through one script if possible. However, some services should also be able to be accessed directly for easier use (such as library lookup/search, secret trading, etc). This allows new and experienced ARP users alike to find using ARP easy and intuitive.

**More Info Regarding the Front End**

To prevent frustration and confusion, the Front End should be as seamless as possible. To that end, queries to the Back End and their Databases should be as user-friendly as possible, and error messages as readable and understandable as possible too. Simple and clear language over complex mumbo jumbo is a must. (Eschew Obfuscation)

The interface pages generated by Front End are to follow the Hackmud page conventions as closely as possible, while striving to remain unique within those constraints. (Note to Self: Make sure to understand what those conventions are!)

Finally, pages should be able to be generated between 2 seconds (non-db queries) and 4 seconds (containing db-queries). Page crashes should be avoided at all cost, and systems should be in place to make sure that quick page generation protects users from issues resulting from server lag and fluctuation in run time.