Overview:

"Cypherspace"

Cypherspace is a secure message posting board, where users are able to create, edit, or delete posts. These posts are then publicly displayed for other users to read. CypherSpace also contains an additional VIP level, in which only select users can participate. VIP users may read all the posts including the normal ones, but normal users may not access VIP posts. There is an additional third permission level for this application: Administrator level. Administrators are capable of creating and deleting users, as well as creating and deleting groups, and assigning users to said groups. Besides these proposed permission levels, a ground level "guest" may read normal user's posts but may not participate in any way.

As mentioned before, Cypherspace has the capacity to host multiple groups. Users, including VIP level, are only capable of posting and reading in their assigned group. Should they wish to change groups, the administrator must be contacted.

This application has four main components: A React-based web client, where messages are viewed and all other requests are carried out. A 'bridge' server, tasked with handling user request tokens, as well as serving as the communication port between Resource and authentication server. The authentication server, responsible for login validation, user registration, and group manipulation, and the resource server, tasked with post storage, display, and categorization. The client will only directly contact this bridge server, not the authentication or resource servers. Both the servers and the client communicate via REST API, opening (and calling) certain endpoints with any given method depending on the function requested.

Going into more detail about the authentication server, it firstly receives a request from the bridge after the user has attempted to login in the client. This request, containing username and password, is compared against the users database. Once the comparison is complete, the authentication server answers with a token to the bridge. This token contains parameters to decide if the user is successfully authenticated, whether they are VIP or not, and their assigned groups. It takes the form of an object with key-value pairs formed by the parameters mentioned above. Should authentication prove successful, the bridge server will save the provided token for that specific user's session. All incoming requests from this point on are only carried out after checking the current session token in the bridge server, making sure the user has permission to carry out the action they intend to complete. Since this token is saved, the client is able to perform any and all actions allowed to them without needing to re-login. Should a logout be triggered, the session token in the bridge server is completely reset, and the session ends. All administrator actions are carried out through the authentication server as well. User and group creation, deletion, and manipulation are all possible, each reaching a determined endpoint in the server. Any changes that involve database manipulation are immediately saved after a successful request, so the data persists after shut down.

The resource server’s main task is to serve and store all posts made by users. When a post is sent to the resource server, it has the capacity to understand and classify it based on: the ID of the sender, whether he’s VIP or not, and the group the user belongs to. This metadata is attached to the post by the bridge server using the aforementioned session token, so it’s

not possible for users to pose as other users. Additionally, the server receives separate parameters for the title and content of the post. This is relevant, since this separation of post sectors allows the client to efficiently display and stylize the posts that the user is allowed to see. When a request to create a post reaches the resource server, it instantly saves it to another database, separate from the authentication database. Users can only delete their own posts, and this is again effectively checked using the bridge server session token supplied by the authentications server. As a final precaution, the user ID is sent along with the delete request and checked by the resource server before deleting the post for good. There is only 1 resource server.

The authentication and resource servers are launched separately, but with the same command: maven spring-boot:run. This command must be run in the server's directory. Please note that maven install must be run during first-time setup. Similarly, the bridge server launches with the command node bridge.js, but npm install must be run first. Finally, for the react client, npm install is run, then npm starts to trigger the react development server. This serves the html css and javascript to a localhost with a port of the launcher's choosing. Do note that there is a limited number of ports that are allowed by the bridge's access policy. These are: 3000, 3064, 3065, 4000. The bridge has a global variable to determine what acns server should be accessed for the other servers. This allows avoiding hard coding the addresses since a simple change to the global variable will suffice.

PHASE 3 UPDATE

Client usage of cypherspace remains largely the same, even after the underlying cryptographic security procedures have been added. Users may expect the application to work as they expect, save for some initiation actions.

Should the client never have connected from that specific browser, they will receive a popup prompting them to confirm the server identity (bridge). This is done via fingerprint comparison. Users are expected to receive the server's fingerprint from the administrator, then make the comparison with the server to make sure it's the same. Should they be different, the user will cancel the connection and report a potential cyber attack.
This process happens only on the first connection. Subsequent connections will be standard.

Similarly, the administrator performs a fingerprint check of the resource server. On successful first connection, the bridge will launch a prompt for the admin to compare the expected and received fingerprints. This ensures that both servers are not being impersonated.

IMPORTANT: The bridge sends a verification request to the resource server right after it's launched, and the client sends a verification request to the bridge in a similar manner. As such, servers and client MUST be launched in the following order:

Resource -> Authentication -> Bridge -> Client.