ECE 528- Application Software Design

Alan Palayil

Due Date: April 20th, 2023

Question 1:

Consider our members application discussed in Lecture 20-24:

i. If the server backend takes a while to respond and the user impatiently clicks the same checkbox multiple times, it may cause multiple requests to be sent, potentially making the browser even busier. This could be an issue if the browser or server is already experiencing a delay. When the response finally arrives, it might not accurately reflect the user's intended action due to the multiple clicks, leading to an incorrect update of the checkbox state and the view.

ii. To implement the proposed solution, you would need to handle the following events:
   - User clicks the checkbox: Upon this action, disable the checkbox to prevent multiple clicks while waiting for the server response.
   - RESTful response to get all groups arrives: Once the response is received, enable the checkbox again, allowing the user to interact with it as needed.

iii. To implement the solution within the MVC pattern, you would need to make the following changes to the model, view, and controller:
   - Model: Introduce a variable to track the state of a pending request. If the variable is set to 'false', it indicates that a request is pending, whereas if it's set to 'true', it means the response has been received.
   - View: Display an appropriate visual indicator (e.g., a loading spinner or disabled checkbox) to the user based on the state of the variable. If the variable is set to 'false', show that the user's request is still being processed.
   - Controller: Update the variable's state to 'false' when the user clicks the checkbox, and then set it to 'true' once the server response is received. This ensures that the view correctly reflects the request's progress and avoids potential issues from multiple clicks.

Question 2:

A possible alternative is:

An alternative approach to password authentication that avoids sending the account name and password directly to the server backend is to use a hashed version of the password. The web application can hash the password on the client-side and then send the hash to the backend for validation. This method reduces the risk of plaintext passwords being accidentally logged by a careless backend developer.