USE CASES DOCUMENT for USER API [version 0.1]

[UC-000]: User Creates new account

High Level Description: Web client creates a new Proof Buddy account.

Basic Flow: User “signs up” for a new student account by interacting with the “sign up” feature. New users are given the option to create a student, teacher, or administrator account. The user will be presented with the following fields: “Username”, “Password”, and “Email Address”. The user will fill this information out then press the “Submit” button. From there the system authenticates and authorizes user’s account.

Preconditions: The user must not presently have an account under the same email. The user must have pre-authorization for teacher and or administrator accounts. The user must provide a valid email address for account creation and verification. The user must not leave any of the following field blank: “Username”, “Password”, and or “Email Address”

Postconditions: User account is created, and account information communicated from the front end to the backend and stored in the database.

Extensions/Alternates: As of now, no system for pre-authorizing a teacher or administrator account. A robust system will need to be created to authenticate and authorize these account types.

[UC-001]: User Logs in

High Level Description: User Logs into Proof Buddy

Basic Flow: User “logs in” to proof buddy by providing an email and password combination by interacting with the log in button. User email and password will be entered into the “email” and “password” fields in the front-end web page. Users will click the login button once both fields have been satisfied.

Preconditions: The user must have a valid account that they have created. The user must provide an email address that corresponds to an account they previously created. The user must provide the correct corresponding password that was used when creating the account that they are logging into.

Postconditions: The user is successfully logged into Proof Buddy with their account authenticated and authorized.

Extensions/Alternates: Presently there is no system for hashing a user's password and this presents a security risk. In the future the system will encode/hash user passwords with JWT

tokens. This will offer a URL-safe way to communicate sensitive information between the front-end and back-end of Proof Buddy.

[UC-002]: User Resets password

High Level Description: User will reset their password to their account.

Basic Flow: The user presses the “reset password” button on the front-end application. The user is presented a field to enter their email address associated with their account. The user will receive an email from Proof Buddy containing a hyper link for password recovery. The user will click the link and a new web page will appear with two fields: “New Password” and “Confirm New Password”. On this webpage, the user will enter a “New Password” in the first field, and the same password in the “Confirm Password” field and then subsequently press the submit button. The submit button will run a script, that will replace the user’s password with the newly supplied password.

Preconditions: The user must provide an email address that is currently associated with an existing Proof Buddy account. Both the “New Password” and the “Old Password” must be identical in character and case. Furthermore, the user must have access to the email address they are supplying for the password reset.

Postconditions: The user successfully resets their password, and the corresponding user data information is updated in the database.

Extensions/Alternates: Presently, there is no front-end option for a password reset, nor a back-end feature to allow for a password reset internally. As the project progresses, this feature will need to be a high priority, and must use some form of email service to send password rest links.

[UC-003]: User Logs Out

High Level Description: User will log out of their account from the current session.

Basic Flow: The user presses the “Log Out” option/button on the front-end application.

Preconditions: The user must be logged into their account to use the log out function.

Postconditions: The user has successfully logged out of their account.

Extensions/Alternates: Presently, there is no user log out feature. A high priority is to couple this feature development with the “log in” feature.