**SAMPLE USE CASE DOCUMENT**

**SIGN UP/REGISTER**

|  |  |
| --- | --- |
| Summary | This use case will allow a user to set up an account with the application, setting the login factors and other key attributes of the user account. |
| Preconditions | 1 – User must be on the welcome activity of the application  2 – User cannot have already logged in previously |
| Triggers | User will click on/push the ‘Sign Up/Register’ button |
| Main Flow | 1 – A new activity is started, bringing the user to the Sign Up Information page |
|  | 2 – The user will fill out all required items on the form, including setting username, user e-mail, and password, as well as other non-required fields |
|  | 3 – The user will click on/push the ‘Sign Up’ button |
|  | 4 – A dialog box repeating the user input information will pop up prompting the user to review the information to be submitted |
|  | 5 – The user will select ‘OK’, the password will be hashed, and the information will be sent to the authenticator database |
|  | 6 – The database will be checked to make sure no other user with the same username already exists |
|  | 7 – The user information will be stored in the database |
| Alternative Flow | 2a – Upon arrival at the Sign Up Information page, the user selects the ‘Cancel’ button and ends the process |
|  | 5a – If the password does not meet the requirements, the user is informed and then process will jump to 6b |
|  | 6a – If a duplicate user name is found the user name is informed via dialogue box |
|  | 6b – The user is returned to the Sign-Up Information form with all fields except for the user name and password retaining the original values entered by the user |
| Postconditions | The user information and hashed password are stored in the database |
| Business Rules | 1 – The user password must be hashed and not transmitted in plain text  2 – Usernames must be unique  3 – Passwords must be a minimum of 14 characters |

**LOG IN**

|  |  |
| --- | --- |
| Summary | This use case will allow a user to enter their credentials and authenticate with the authentication server to log into their account |
| Preconditions | 1 – User must be on the welcome activity of the application  2 – User cannot have already logged in previously |
| Triggers | User will click on/push the ‘Log In’ button |
| Main Flow | 1 – A new activity is started, bringing the user to the Log In page |
|  | 2 – The user will enter their credentials |
|  | 3 – The user will click on/push the ‘Log In’ button |
|  | 4 – The password will be hashed, and the information sent to the authentication database |
|  | 5 – The authentication database will validate the provided username and hash |
|  | 6 – The authentication database will return a toke for the user to use on future requests |
|  | 7 – The user is redirected to the Landing Page activity to further access the program |
| Alternative Flow | 2a – Upon arrival at the Log In page, the user selects the ‘Cancel’ button and ends the process |
|  | 5a – If the credentials do not validate, the user will be returned to the Log In page, with a message displayed that the user was unable to log in |
|  | 5b – An entry is made into the authentication database noting the time of the failed login attempt for the user |
| Postconditions | A token is returned to the user for use with future requests |
| Business Rules | 1 – The user password must be hashed and not transmitted in plain text  2 – Tokens expire after 24 hours and must then be renewed |