# CST8218 Assignment 2

Group size: 2-5, with the following requirements depending on the size of the group:

2 members: Authentication, Testing, and a React page

3 members: Authentication, Testing, React page, Java Swing Client or Mobile

4 members: Authentication, Testing, React page, Java Swing Client and Mobile

Due Date: See Brightspace for due dates of submission and demonstration

Objective: Add security constraints to protect both the JSF pages and the RESTful API of your Assignment 1 solution.

## Methodology (required) :

* Implement an account administration area with JSF pages to manage the appuser table, which contains the accounts for the application (re-use code from Lab 4).
* Localize (translate) the JSF page for creating a sprite (your choice of language)
* Practice applying Basic Authentication to the address-book application
  + Start by adding a user to the file realm and testing with that user
  + Then, add the @DatabaseIdentityStoreDefinition annotation to a class in your project (which one?) in order to test with users in your appuser table
* Apply Basic Authentication to your Sprite application JSF pages
  + Base the authentication on the accounts in your appuser table
  + Test both successful and unsuccessful authentication
  + Test both successful and unsuccessful authorization
* Apply Basic Authentication to your Sprite application RESTful API
  + Base the authentication on the accounts in your appuser table
  + Test both successful and unsuccessful authentication with Postman
  + Test both successful and unsuccessful authorization with Postman
* Add a login form (username/password) to your JSF pages to do Form Authentication
  + After you have tested Form Authentication, switch back to Basic Authentication (RESTful API will not work with Form Authentication enabled)
  + Both JSF pages and RESTful resources can be protected with Basic Authentication

## Feature Requirements:

* The RESTful interface works as specified in Assignment 1, except now it is secured. (NO RESTful API is required for the AppUser account instances.)
* Ensure your code has appropriate implementation comments.
* Langauge selector on index page, and Localize (translate) the JSF page for creating a bouncer (your choice of language)
* Implement an integrated User Administration area in your bouncer project to manage users for the project (re-use code from Lab4 and merge it into the Assignment 2 project folder).
* Lock down the application with authentication of users to the application, so that in order to use the JSF pages or the RESTful interface, the user account must have been added to the database with the right group:
  + RESTful resources accessible only to group ApiGroup
  + JSF Bouncer Pages (bouncer folder) accessible only to group PagesGroup
  + Admin group can access both of the above, as well as JSF admin area pages (appuser folder)
  + Other users cannot access any of the above, but they can access the JSF index page
* Implement some automated unit tests using Junit (Bouncer method tests are sufficient)
  + Tests must be meaningful – an example of a test that is not meaningful is a test that always passes by checking whether 1==1
* Implement some Selenium tests on the JSF pages
* Implement a React Page that displays the position and size information for the list of bouncers
* Implement a canvas with animation of the bouncers on the React page
* Implement a way to make changes to bouncers from the React page (to be dicussed in lecture)
* (Groups of 3) Implement a Java Swing Client that can make some sort of change to a bouncer; OR implement a Mobile Application that displays the position and size information for the list of bouncers
* (Groups of 4) Implement a Java Swing Client that can make some sort of change to a bouncer; AND implement a Mobile Application that displays the position and size information for the list of bouncers

Grading Criteria:

20% Deduction from the total grade for each missing group-size-related component

Examples:

* A group of three without Swing client and without Mobile client: -20%
* A group of four without Swing client or without Mobile client: -20%
* A group of four without Swing client and without Mobile client: -40%

**Grading Guide**

Peer review submitted individually, indicating percentage contribution of each group member including yourself. Example: John 45%, Jane 55%. (this will potentially be used to adjust grades)

AppUsers functionality merged into the Bouncer project: 2

Password Hashing: (included in 2 marks above)

RESTful resources properly done and accessible only to group RestGroup: 2

JSF Pages properly done and accessible only to group JsfGroup: 2

Internationalization: 1

Admin group can access both and AppUser table (included in 4 marks above for JSF and RESTful)

Other users cannot access either (included in 4 marks above for JSF and RESTful)

React Page:

Display: 2

Animate: 2

Make changes: 2

Testing: 3

Implementation comments:

Comments similar to Assignment 1, in all files added or changed: 4

**Demo (required, every group member present)**

**Be prepared to show source code, and answer questions about your work to get the full marks listed below**

Show that the source code is under control with git

Actually be ready to demonstrate: (app running, admin console running, Netbeans running, users added, and Postman ready for testing with correct URL)

Run app showing users with hash passwords:

Show Postman request successful:

JSF request fail:

Demo Testing:

Demo Group-related feature to avoid deduction

## Submission

With every group member present, demonstrate your final secured Bouncer program to your Lab Instructor, and submit a zipped archive of the Netbeans project folder(s) using the Brightspace link provided. Please use only zip, and do NOT use .7zip, or .rar formats for the submitted archive.