# Java Web Development Basics

# Exam – Casebook

**Casebook** is a Social Media Application. You have been tasked to implement this application for an unusually low price, by an unusually rich client. There are several requirements you must follow in the implementation.

## Technological Requirements

* Use the Javache Web Server
* Use the Broccolina and Toyote request handlers
* Use the Summer Framework
* Use Hibernate native (no Spring Data)

The Technological Requirements are **ABSOLUTE**. If you **do not follow** them, you will **NOT** be scored for other Requirements.

Now that you know the Technological Requirements, let us see what are the Functional Requirements.

## Database Requirements

The **Database** of the application needs to support **1 entities**:

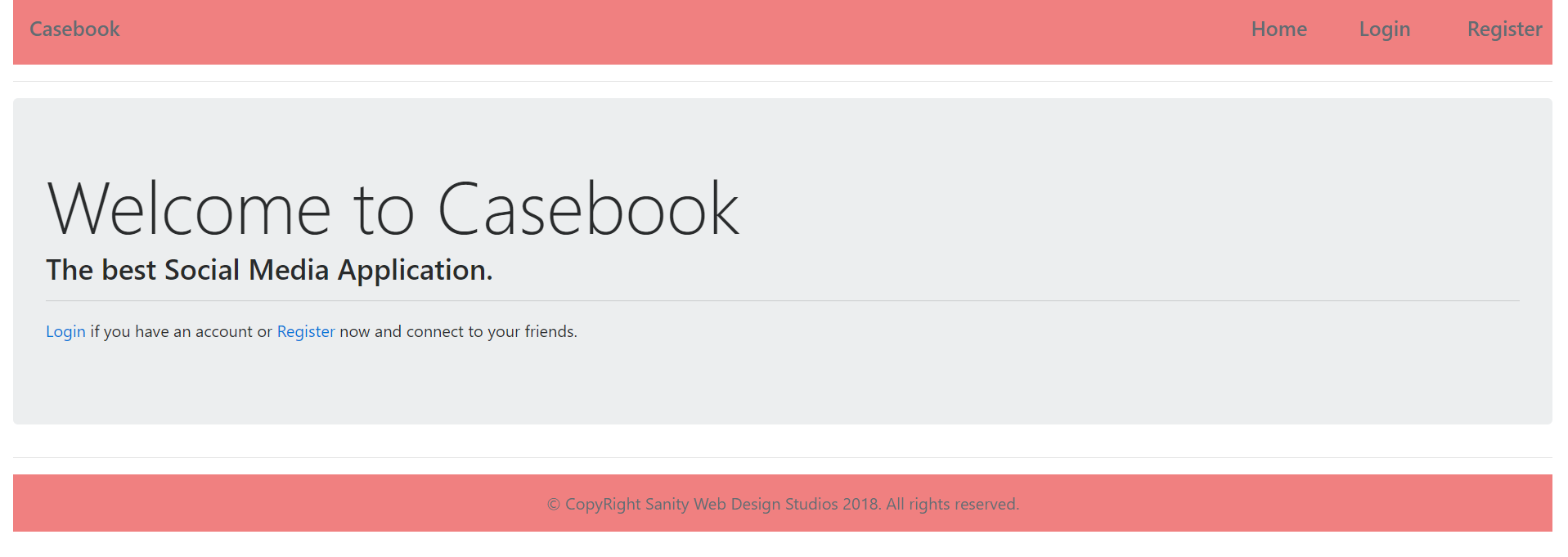
### User

* Has an Username
* Has a Password
* Has an Gender
* Has Friends (other users)

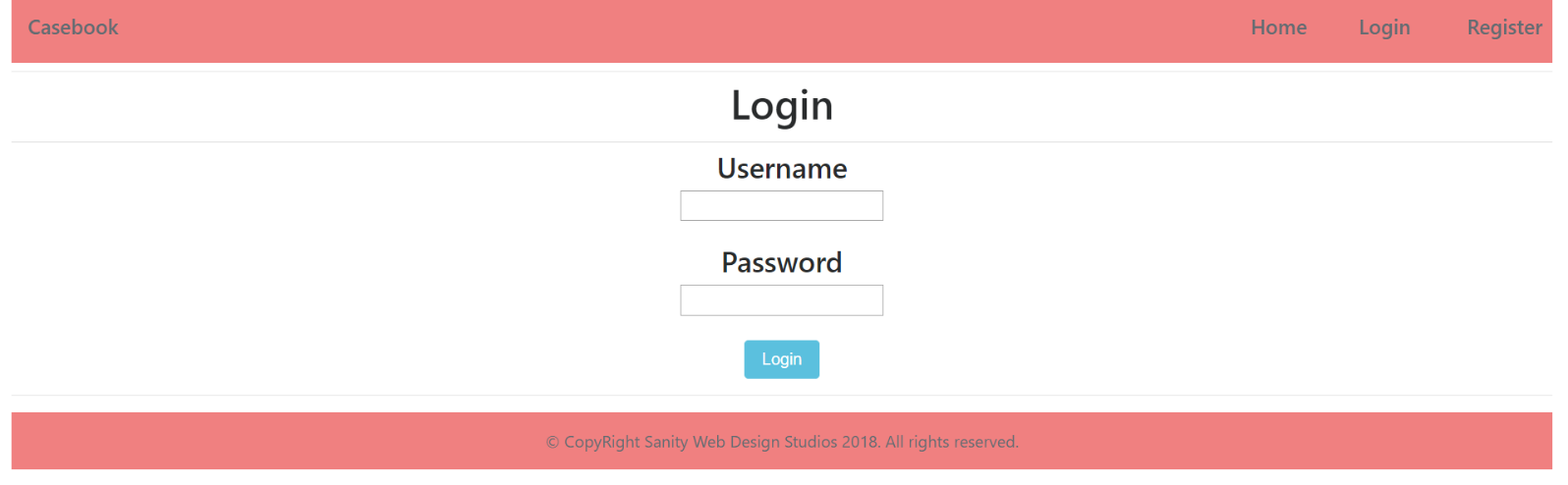
Implement the entities with the **correct datatypes**, and implement **repositories** for them.

## Template Requirements

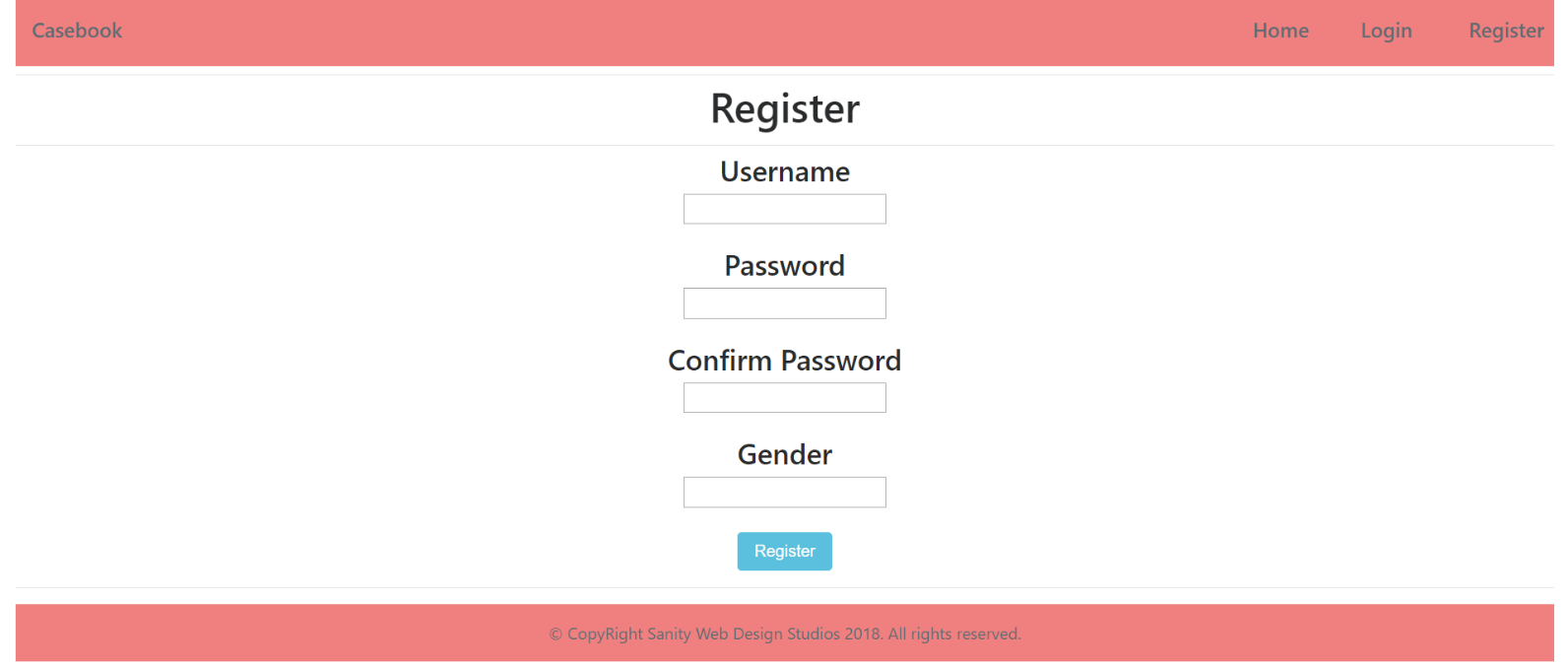
### Index Template (route = “/”) (logged out user)



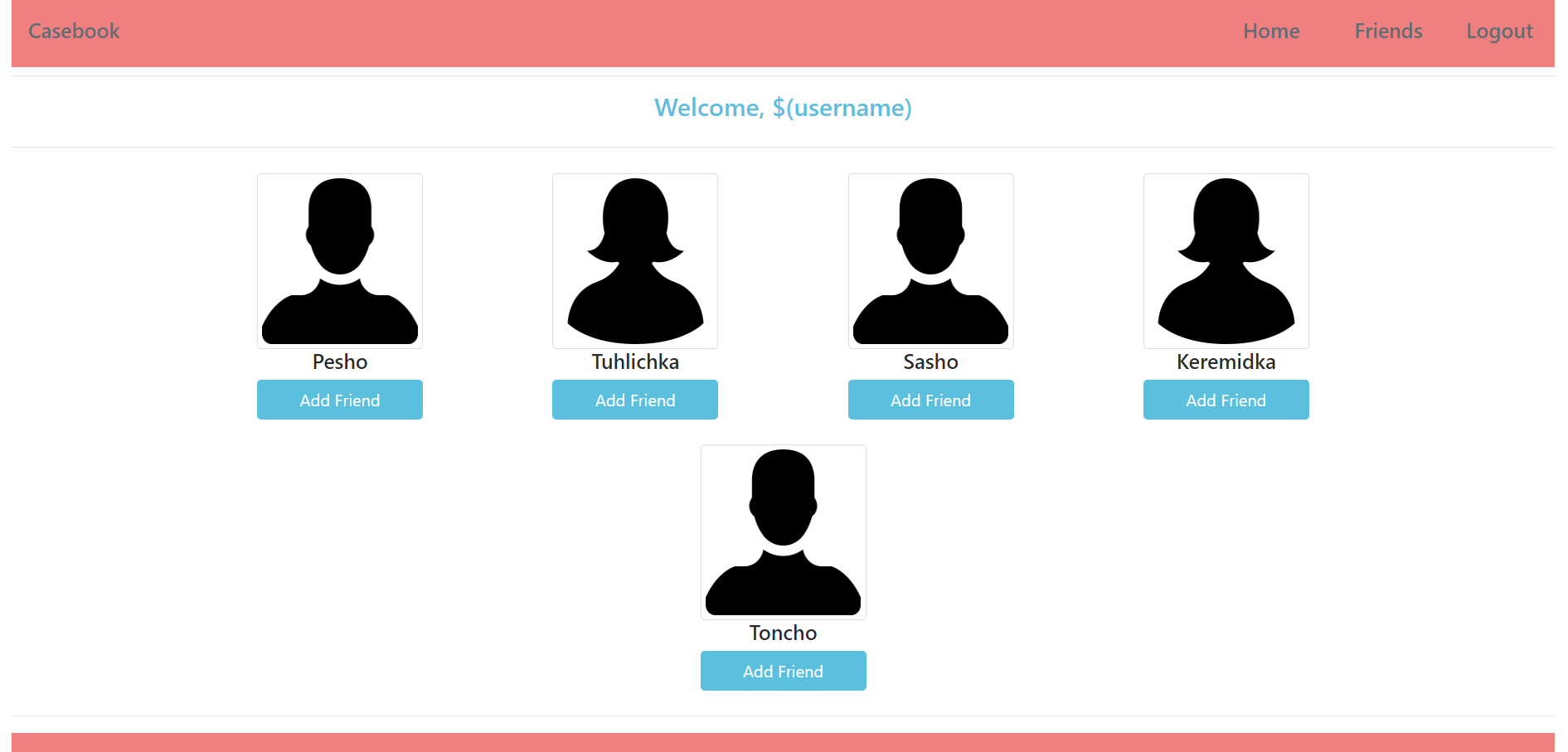
### Login Template (route = “/login”) (logged out user)



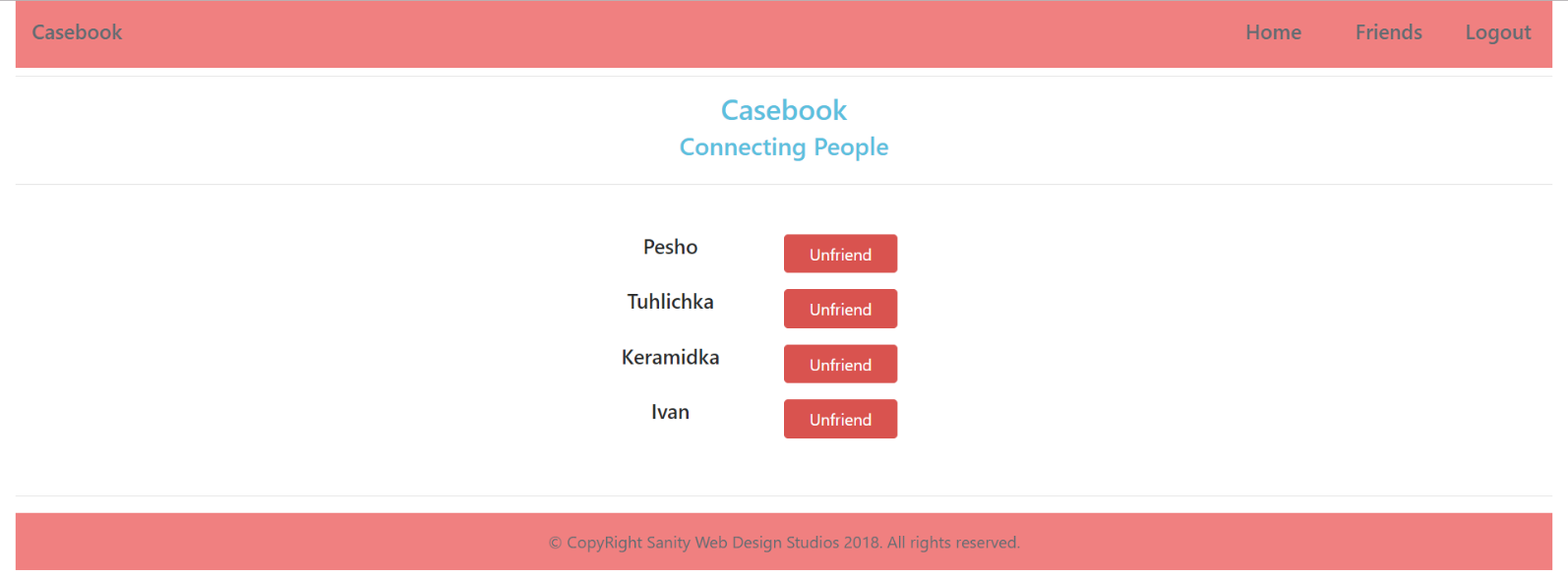
### Register Template (route = “/register”) (logged out user)



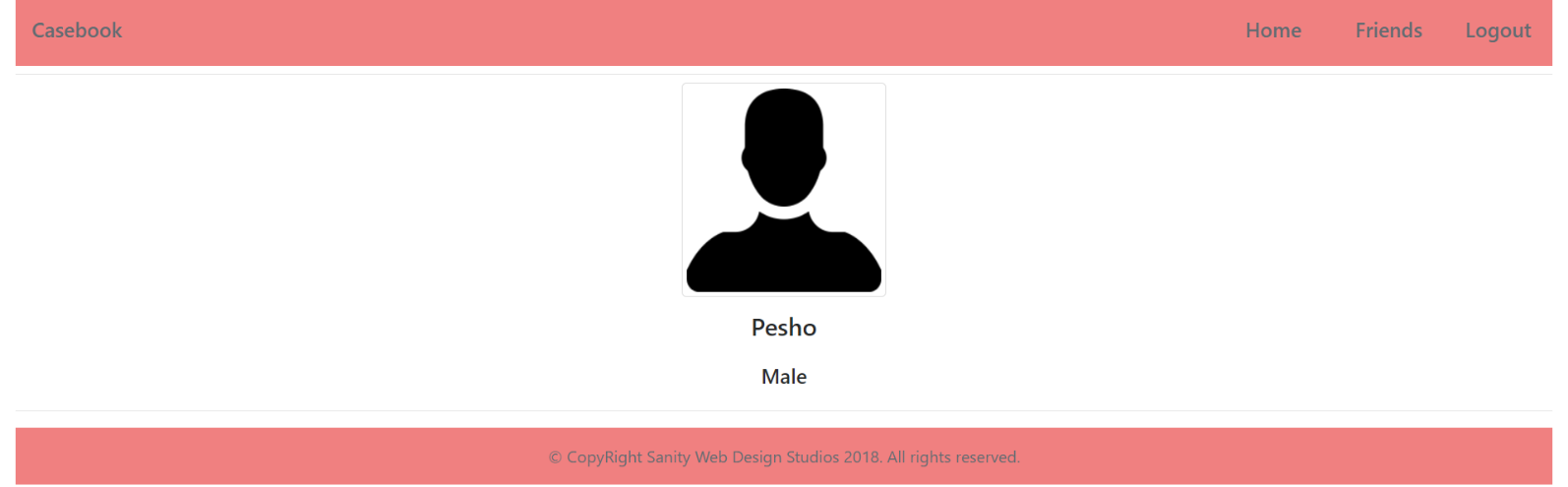
### Home Template (route=”/home”) (logged in user)



### Friends Template (route=”/friends”) (logged in user)



### Profile Template (route=”/profile/{id}”) (logged in user)



Some of the templates have been given to you in the application skeleton, but the others will be for you to implement, so make sure you implement them correctly. You can use the given ones as helpers.

**NOTE**: The templates should look **EXACTLY** as shown above.

**NOTE**: The templates do **NOT** **require** **additional** **CSS** for you to write. Only **bootstrap** and the **given css** are enough.

## Functional Requirements

The Functionality Requirements describe the functionality that the **Application** must support.

The **application** should provide Guest (not logged in) users with the functionality to:

* Login
* Register
* **View** the Index page.

The **application** should provide Users (logged in) with the functionality to:

* Logout
* **View** all Users (Home page)
* **Add** Friends (Clicking on [Add Friend] button on Home page)
* **View** self (**logged-in user**) Profile (Clicking on [Welcome, $(username)] message on Home page)
* **View** all Friends (Friends page)
* **Remove** Friends (Clicking on [Unfriend] button on Friends page)
* **View** friend Profile (Clicking on a friend’s name on Friends page)

The **application** should provide **functionality** registering a User with **2 possible genders** for the time being – “Female”, “Male”.

The Home page should view **ONLY** the **users** which **are NOT friends** of the **currently logged in user** and **are NOT** the **currently logged in user**.

The Friends page should view **ONLY** the **users** which **ARE friends** of the **currently logged in user**.

The **application** should **store** its **data** into a MySQL database, using Hibernate native.

## Security Requirements

The Security Requirements are mainly access requirements. Configurations about which users can access specific functionalities and pages.

* Guest (not logged in) users can access Index page.
* Guest (not logged in) users can access Login page.
* Guest (not logged in) users can access Register page.
* Users (logged in) can access Home page.
* Users (logged in) can access Friends page.
* Users (logged in) can access Add Friend functionality.
* Users (logged in) can access Remove Friend functionality.
* Users (logged in) can access Profile (self) page.
* Users (logged in) can access Profile (friend) page.
* Users (logged in) can access Logout functionality.

## Framework Requirements

Extend the given skeleton (Javache Web Server, Broccolina / Toyote Request Handlers, Summer Framework), by adding **static template** functionality.

To be more clear, make sure you **do NOT have** the footer and the navigation bar, in every possible template, as it is an **unnecessary repetition**.

Possible ways to do it:

* **Extract** the footer and navigation bar in separate templates. Create a functionality with which you **import** them in the **current template**.
* Create a base template, which holds the footer and the navigation bar, and transform all other templates into **partial templates**, which you then **render** in the base template.

## Scoring

This section describes how the scoring of the Exam Preparation will be made.

### Database Requirements – 10 points.

### Template Requirements – 20 points.

### Functional Requirements – 40 points.

### Security Requirements – 10 points.

### Framework Requirements – 20 points.