# Java Web Development Basics – Retake Exam

# Java EE Block – 30 pts

The next several tasks, will test your knowledge on the Java EE components you’ve seen throughout the course. For your solutions, just submit your source code. It will be configured and deployed on Tomcat, while being checked.

## Simple JSPs – 10 pts

Using your knowledge on Java EE, implement 2 simple JSPs which will be used to render data dynamically. But to render data dynamically, we will need some sort of data object. So let’s create our Channel. The Channel is a data object which stores data about an abstract **media** **channel**. You will see later what it will be used for.

First implement a class Channel – which has these properties:

* Name – a **String**.
* Description – a **String**.
* Type – can be one of the following values – (“Game”, “Motivation”, “Lessons”, “Radio”, “Other”)
* Followers – a **Integer**.

**Initialize** a collection which holds **3 Channels** with the **following data**:

|  |  |  |  |
| --- | --- | --- | --- |
| Channels | | | |
| Field | 1 | 2 | 3 |
| Name | Markiplier | Gary’s Channel | SoftUni |
| Description | MarkipleirGAME ch … | Look at me! I’m cool! | Best online courses ever!!! |
| Type | Game | Motivation | Lessons |
| Followers | 10000 | 253 | 12515 |

Now that you’ve got a data collection, we can start implementing our **JSPs**.

### All Channels

Implement a JSP – channels/all.jsp, which renders only the names of the Channles. Upon clicking a name of a Channel, you should be **redirected** to a **details page**, with **query parameter** – the name of the Channel.

### Channel Details

Implement a JSP – channels/details.jsp, which **renders full data** about the **selected** Channel. The selected Channel should be extracted from the collection, using the name from the **query parameters**.

**Note**: The design of the tasks stated above is by your choice, if it fulfills the requirements.

## Simple Servlets – 20 pts

Using your knowledge on Java EE, implement 3 simple Servlets, which work with a shared data.

### Channel Create

Implement a **Servlet** – ChannelCreateServlet, which listens on route “/channels/create”.

Upon a **GET** request, it should return a form which accepts **3 inputs** – a name, a description and a type.  
**NOTE**: You should only be able to submit “Game”, “Motivation”, “Lessons”, “Radio”, “Other” as values for the type.

By **default,** the Channel’s followers are **0**.

The **form** should send a POST request to the same route.

Upon a **POST** request, you should **redirect** to “/channels/all”.

### Channels All

Implement a **Servlet** – ChannelAllServlet, which listens on route “/channels/all”.

Upon a **GET** request, the **Servlet** should render a page with the names of all created Channels. Upon clicking a name of a Channel, you should be **redirected** to a “/channels/details”, with **query parameter** – the name of the Channel.

### Channel Details

Implement a **Servlet** – ChannelDetailsServlet, which listens on route “/channels/details”.

Upon a **GET** request, the **Servlet** should extract the Channel with the given name in the **query parameters** and render a page with **full information** about it.

There should also be a **button** [Follow]. Upon **clicking** it, you should **increment** the Channel’s followers with 1 and redirect to “/channels/all”.

**Note**: The design of the tasks stated above, is by your choice, if it fulfills the requirements.

**Note**: The algorithm for transfering data, is by your choice, if it fulfills the requirements.

# Application Block – 70 pts

# Mish-Mash

**Mish-Mash** is a web application for media channels, like YouTube, with the difference that everything is mashed up like Mish-Mash. You have been tasked to implement this application by the Indiaver organization. There are several requirements you must follow in the implementation.

The application has been developed to somewhere. The previous developer was not able to implement a lot of the main functionality, but he implemented the guest functionality – **Login**, **Register**, **Logout** etc. The other functionalities are up to you. He also broke some of the Framework’s main modules: Javache, Broccolina, Summer, for unknown reasons…

Below, you will see, how the application should behave, if implemented correctly. You should also fix It, so that it is implemented correctly.

## 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.

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

## Database Requirements

The **Database** of the Mish-Mash application needs to support **2 entities**:

### User

* Has an Id – a UUID String.
* Has an Username
* Has a Password
* Has an Email
* Has Followed Channels – a collection of Channels.
* Has an Role – can be one of the following values (“User”, “Admin”)

### Channel

* Has an Id – a UUID String.
* Has a Name
* Has a Description
* Has a Type – can be one of the following values (“Game”, “Motivation”, “Lessons”, “Radio”, “Other”).
* Has Tags – a collection of **Strings**.
* Has Followers – a collection of Users.

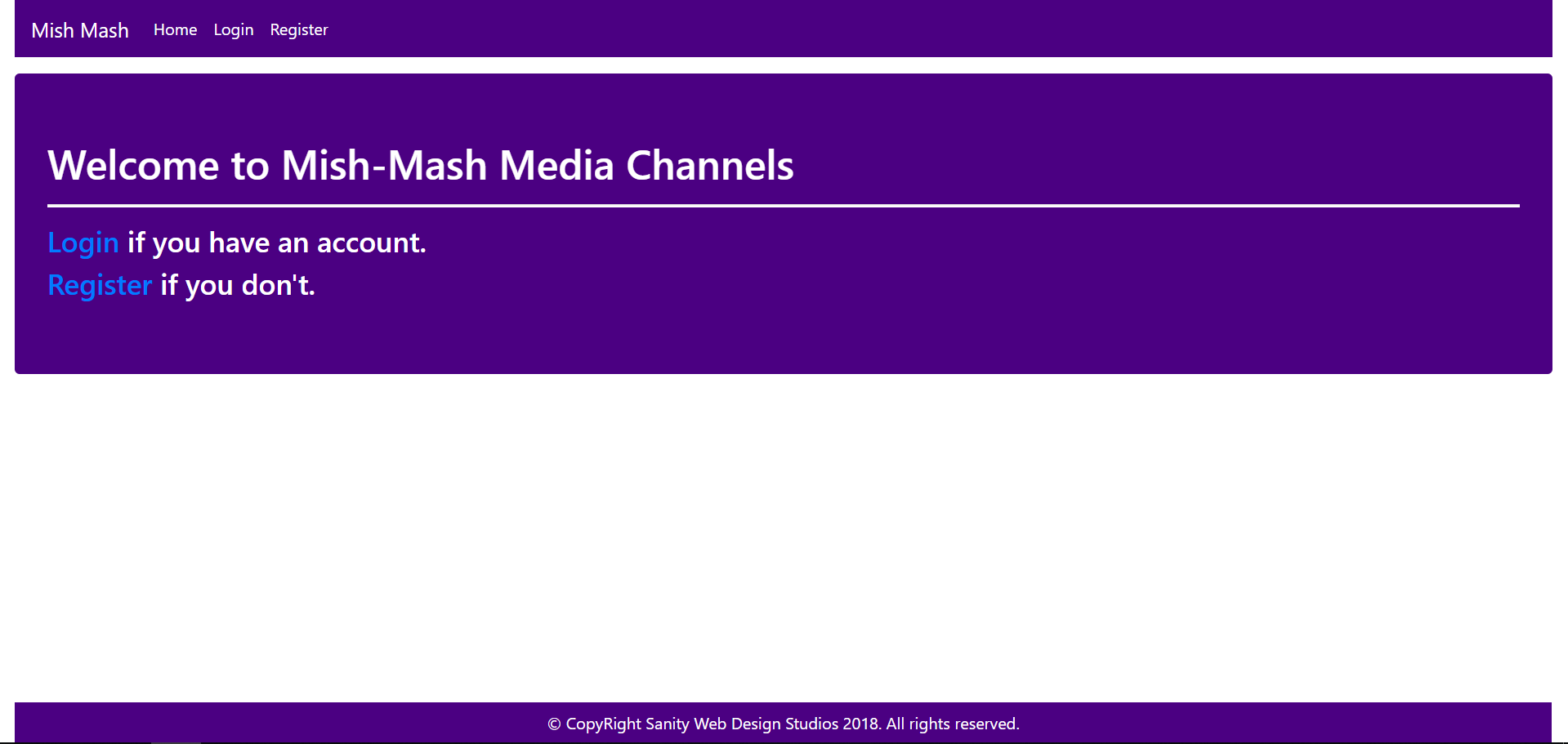
Implement the entities with the **correct datatypes**.

## Template Requirements

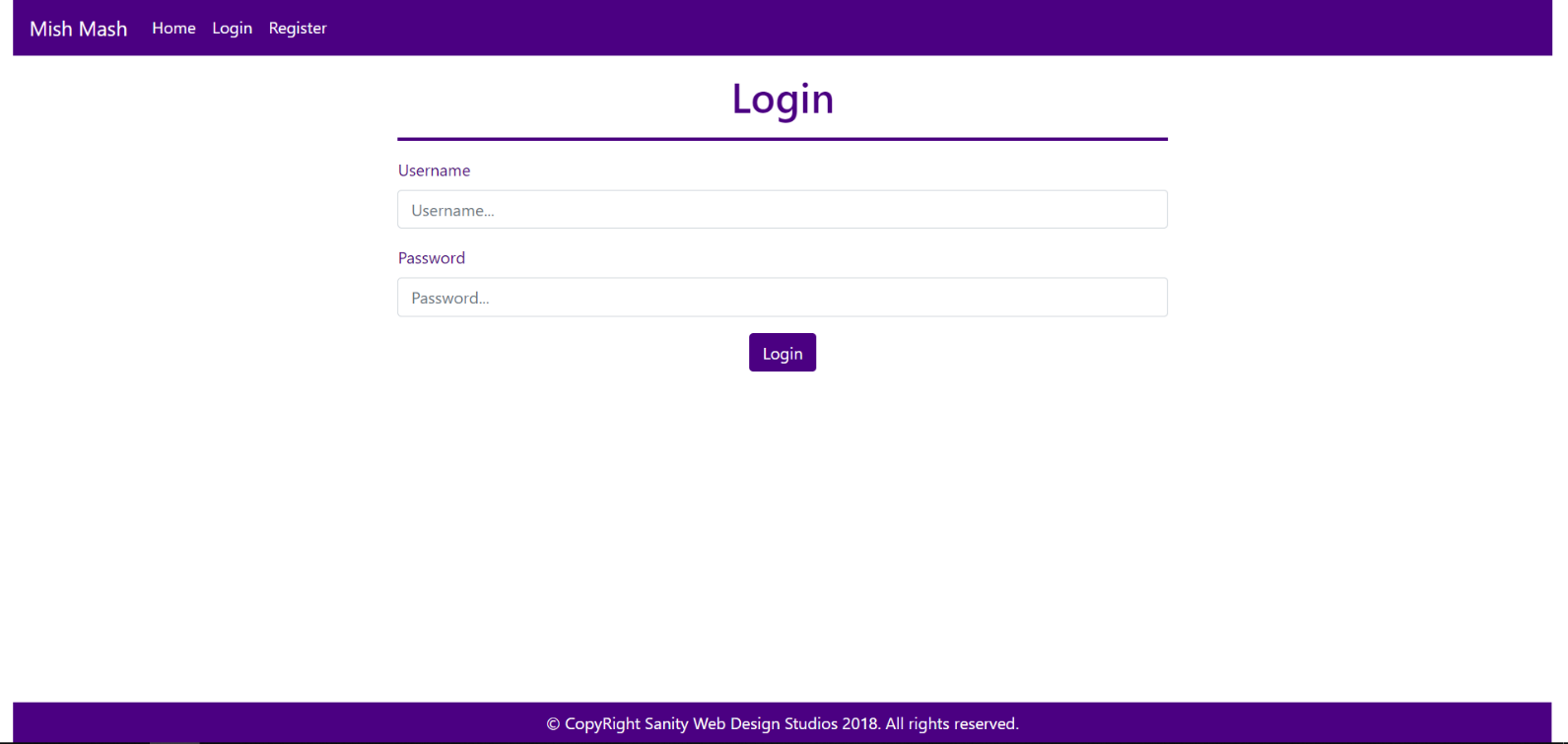
### Guest Templates

These are the **templates** and **functionalities**, accessible by Guests (**logged out** users).

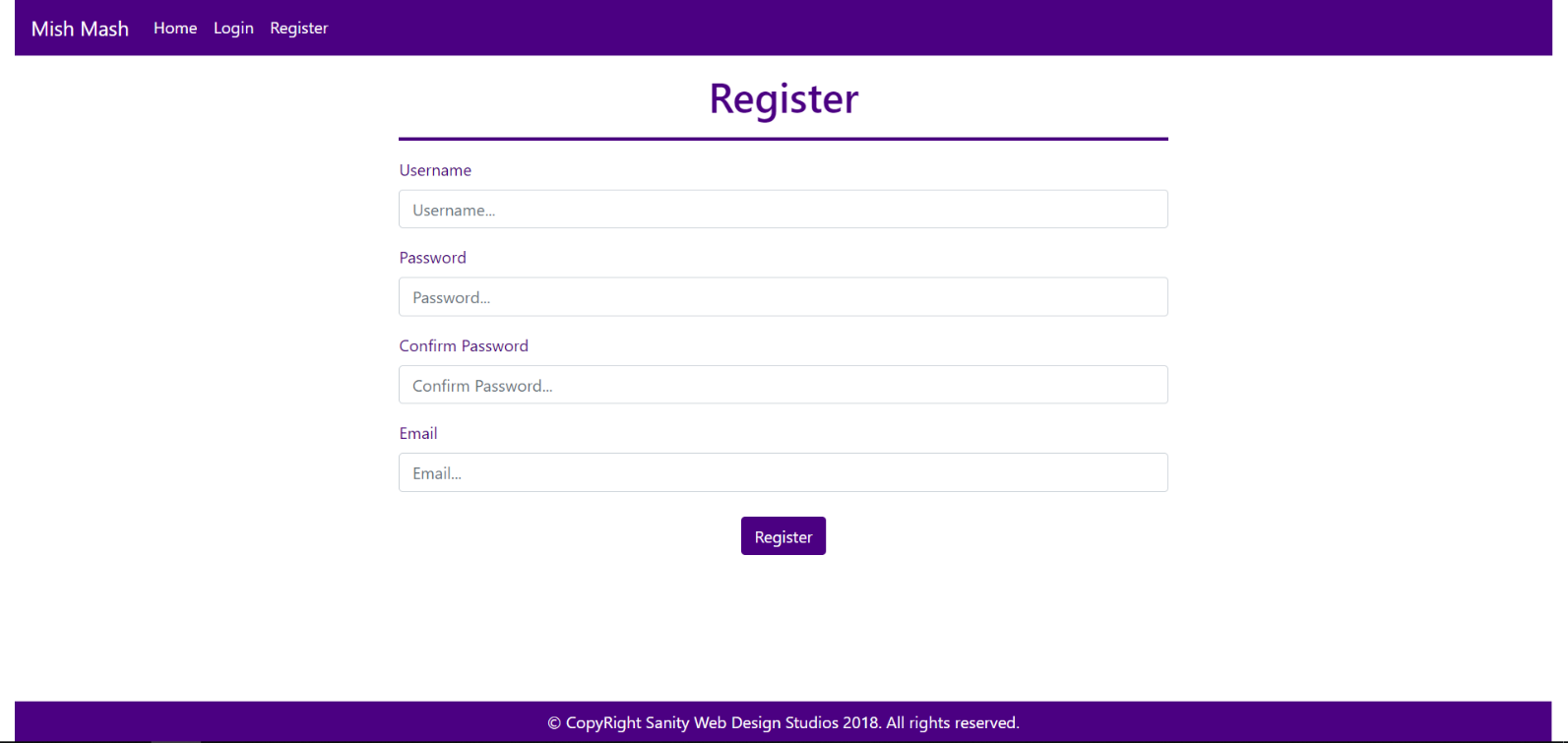
#### Index Template (route = “/”) (logged out user)



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



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



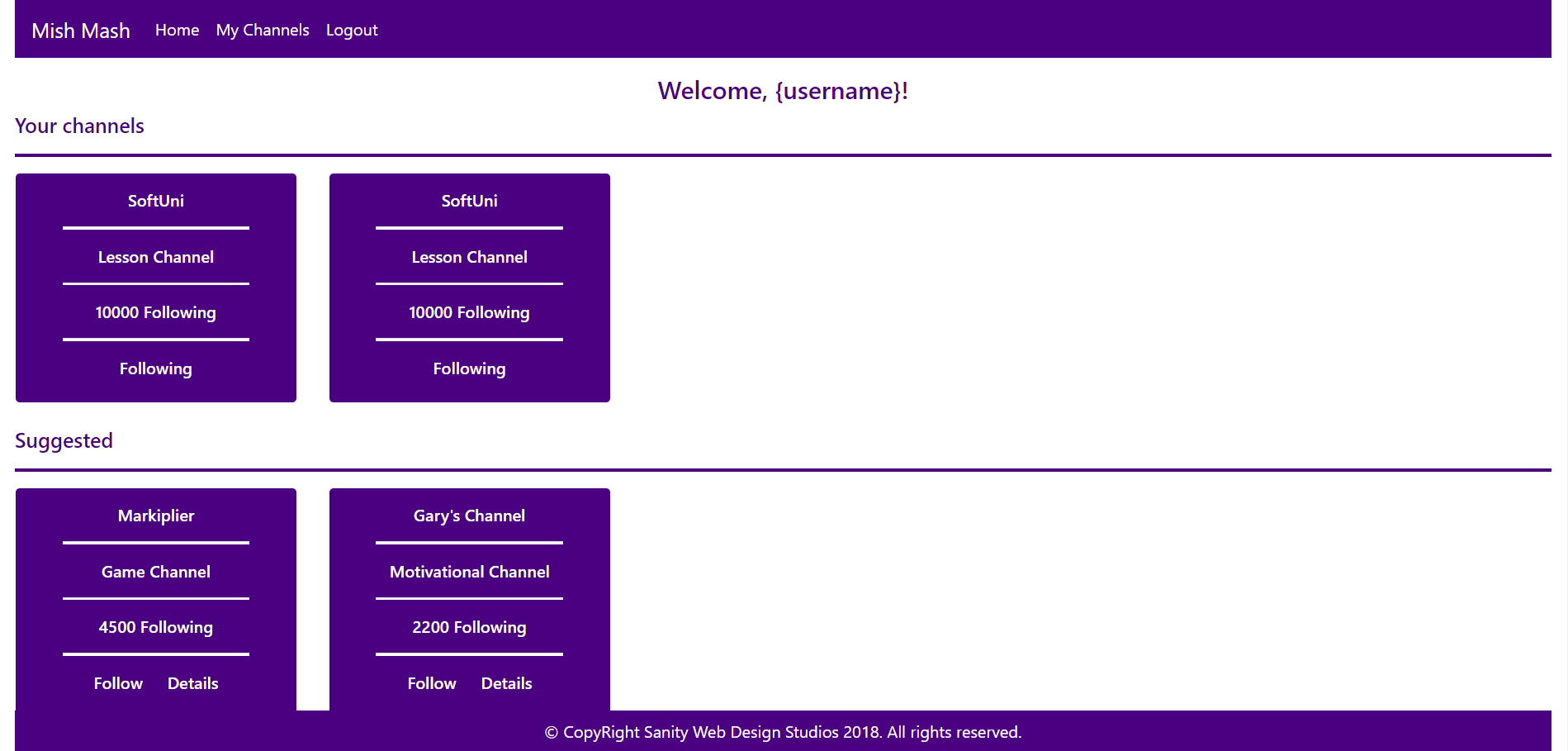
### User Templates

These are the **templates** and **functionalities**, accessible by Users (**logged in** users with Role - User).

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

**NOTE**: As you can see the **elements** are **aligned** to the **left**, **regardless of their count**. The **maximum count** is **5 per** **row**, however they are **always aligned** to the **left**.

##### First



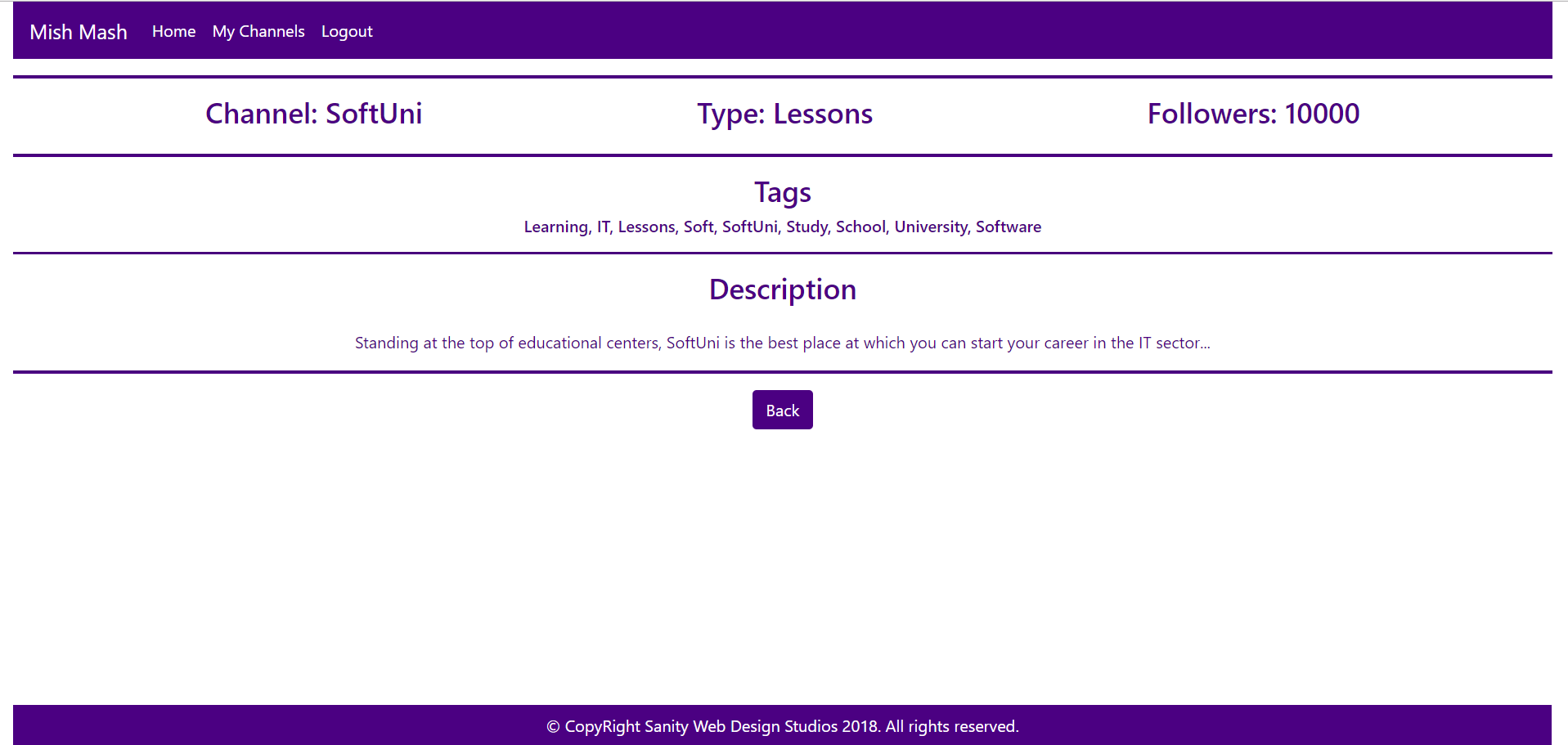
##### Second



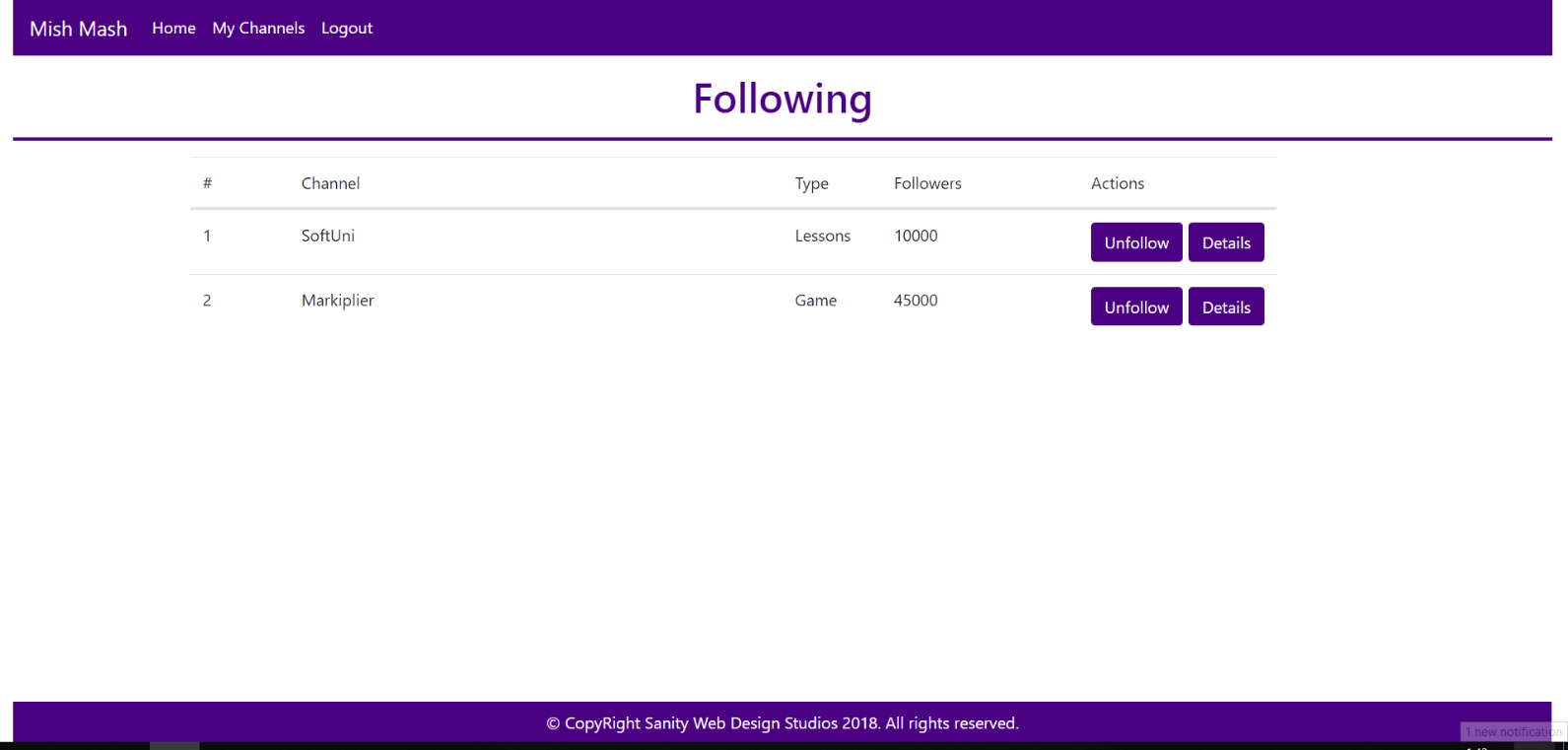
##### Third



#### Channel Details Template (route=”/channels/details/{id}”) (logged in user)



#### My Channels Template (route=”/channels/followed”) (logged in user)



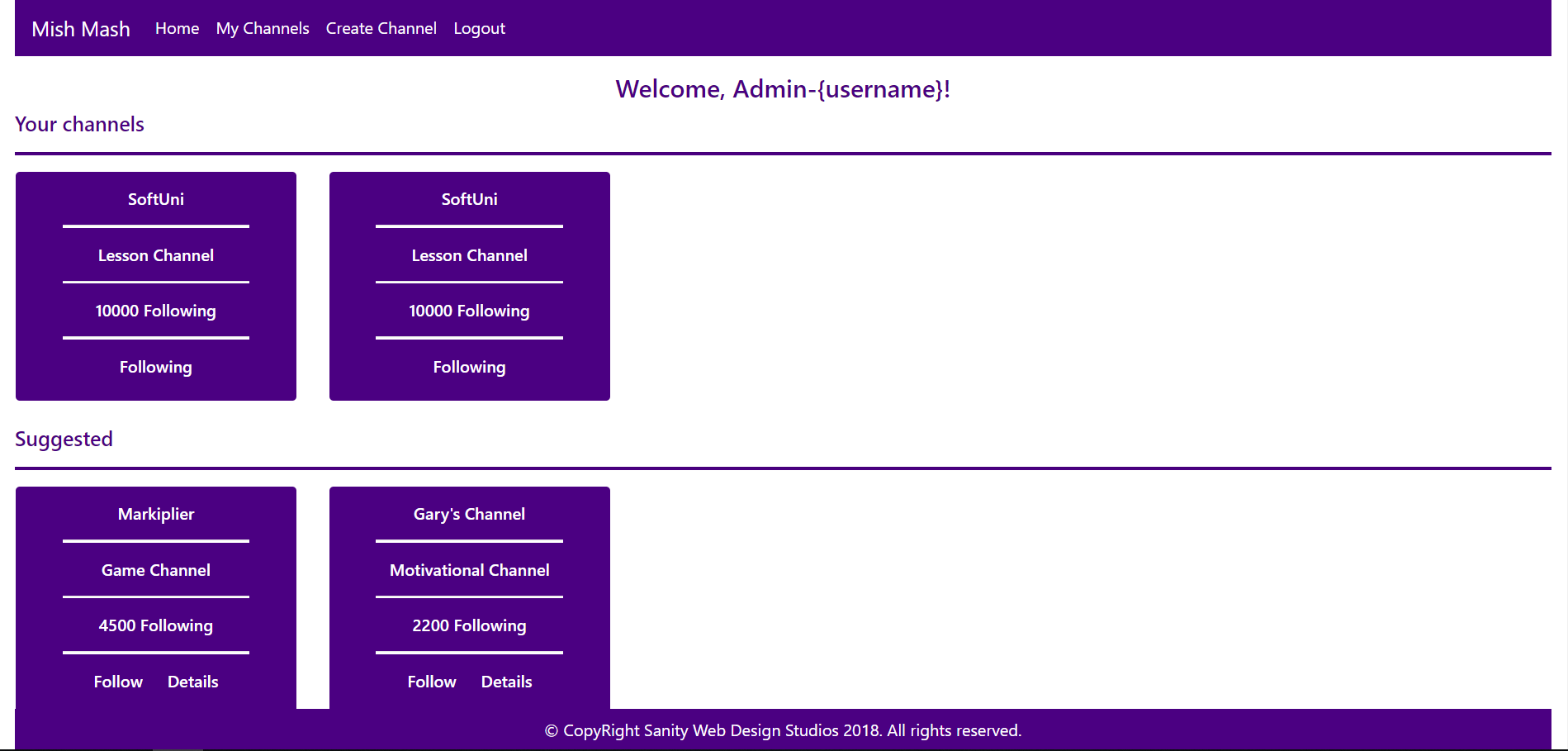
### Admin Templates

These are the **templates** and **functionalities**, accessible by Admins (**logged in** users with Role - Admin).

#### Admin Index Template (route=”/home”) (logged in admin)

**NOTE**: As you can see the **elements** are **aligned** to the **left**, **regardless of their count**. The **maximum count** is **5 per** **row**, however they are **always aligned** to the **left**.

##### First



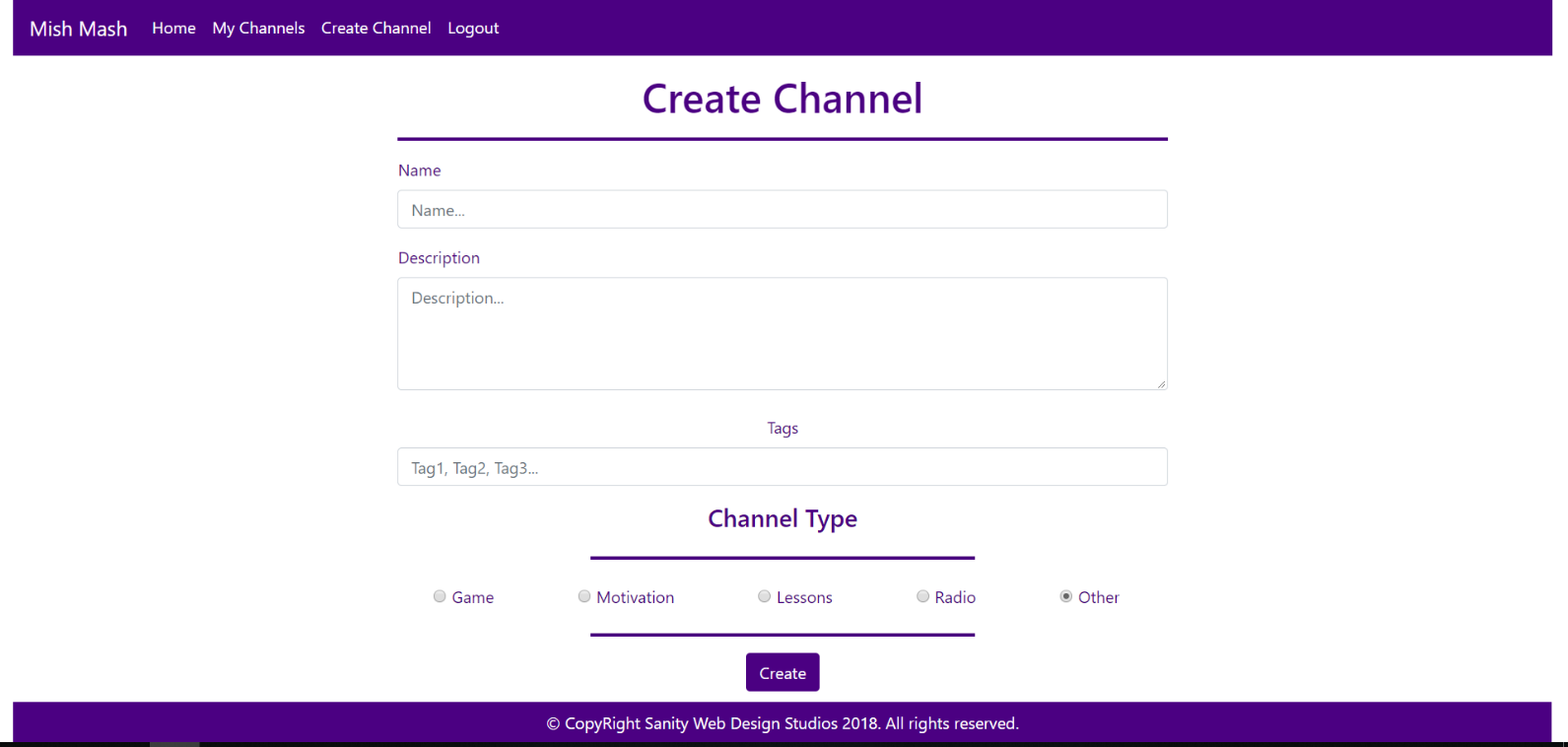
##### Second



##### Third



#### Channel Create Admin Template (route=”/channels/create”) (logged in admin)



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**. Only **bootstrap** is 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 and **view** the Index page.

The **application** should provide Users (logged in) with the functionality to logout, and view all channels, and **follow / unfollow** a Channel, and **view details** about a Channel.

The **application** should provide Users (logged in) with the functionality to logout, and view all channels, and **follow / unfollow** a Channel, and **view details** about a Channel, **create** a Channel.

When you register a new User, it should be assigned with a **role** – User.

### Users

Users can **follow** or **unfollow** Channels. If a Channel is **NOT followed**, it should be represented on the Index page as a rectangular element, which holds **2 buttons** at its bottom – [Follow] and [Details].

* Upon clicking the [Follow] button, the Channel should be added to the Followed Channels of the **current user**.
* Upon clicking the [Details] button, the **current user** should be **redirected** to the Channel’s Details Page.

If a Channel is **followed**, it should be represented on the Index page as a rectangular element, which holds **a simple text** at its bottom – “Following”, with no **functionality** behind it.

### Channels

There are **3 sections** on the Index page:

* Your Channels
* Suggested
* See Other

The Your Channels section holds all of the current user’s Followed Channels. If there are **no such** Channels, it should just be left empty.

The Suggested section holds any Channels, which have at least **1 common tag** with **one** of the **current user’s** Followed Channels.

The See Other section holds all Channels, which are **NOT included** in the Your Channels and Suggested sections.

## 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 and functionality.
* Guest (not logged in) users can access Login page and functionality.
* Guest (not logged in) users can access Register page and functionality.
* Users (logged in) can access User LoggedIn Index page and functionality.
* Users (logged in) can access User Channel Details page and functionality.
* Users (logged in) can access the My Channels functionality.
* Users (logged in) can access the Channel Follow functionality.
* Users (logged in) can access the Channel Unfollow functionality.
* Users (logged in) can access Logout functionality.
* Admins (logged in) can access **every functionality** a **normal** logged in User can.
* Admins (logged in) can access Admin LoggedIn Index page and functionality.
* Admins (logged in) can access the Channel Create page and functionality.