# C# Web Development Basics

# Exam – MeTube

**MeTube** is like YouTube, “ama ne bash” like an old Bulgarian Sage once said. You have been tasked to implement this application for literally no money, by the richest man on earth. There are several requirements you must follow in the implementation.

## Technological Requirements

* Use the Web Server
* Use the MVC Framework
* Use Entity Framework

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 **2 entities**:

### User

* Has an Username
* Has a Password
* Has an Email
* Has Tubes (a collection of tubes)

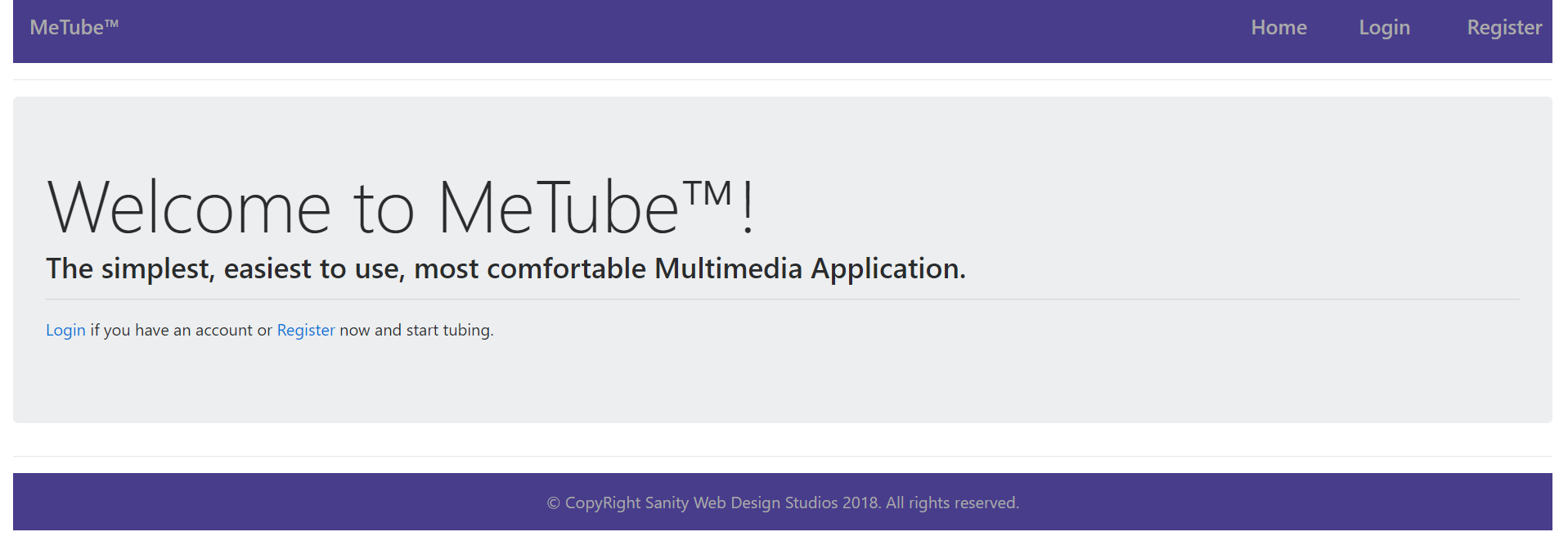
### Tube

* Has a Title
* Has an Author
* Has an Description
* Has an Youtube Id – This Id is only for the **youtube video**. It is not the **Entity’s id**.
* Has Views (an **integer**, by default – **0**)
* Has an Uploader (a User).

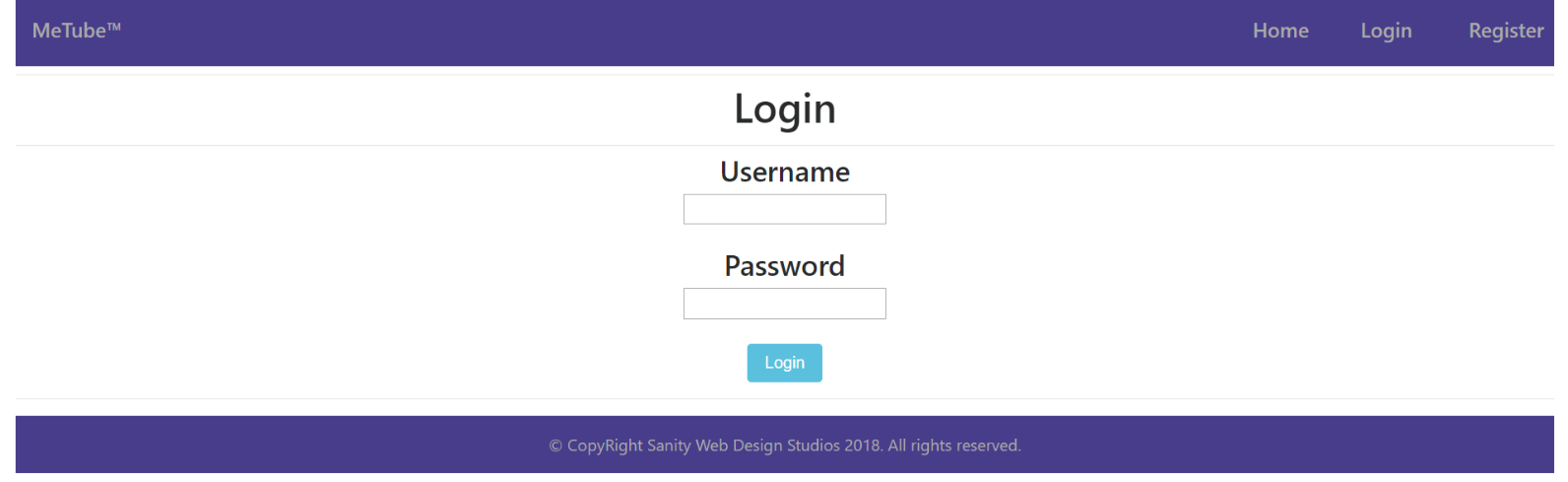
Implement the entities with the **correct datatypes**, and implement **repositories** for them.

## Template Requirements

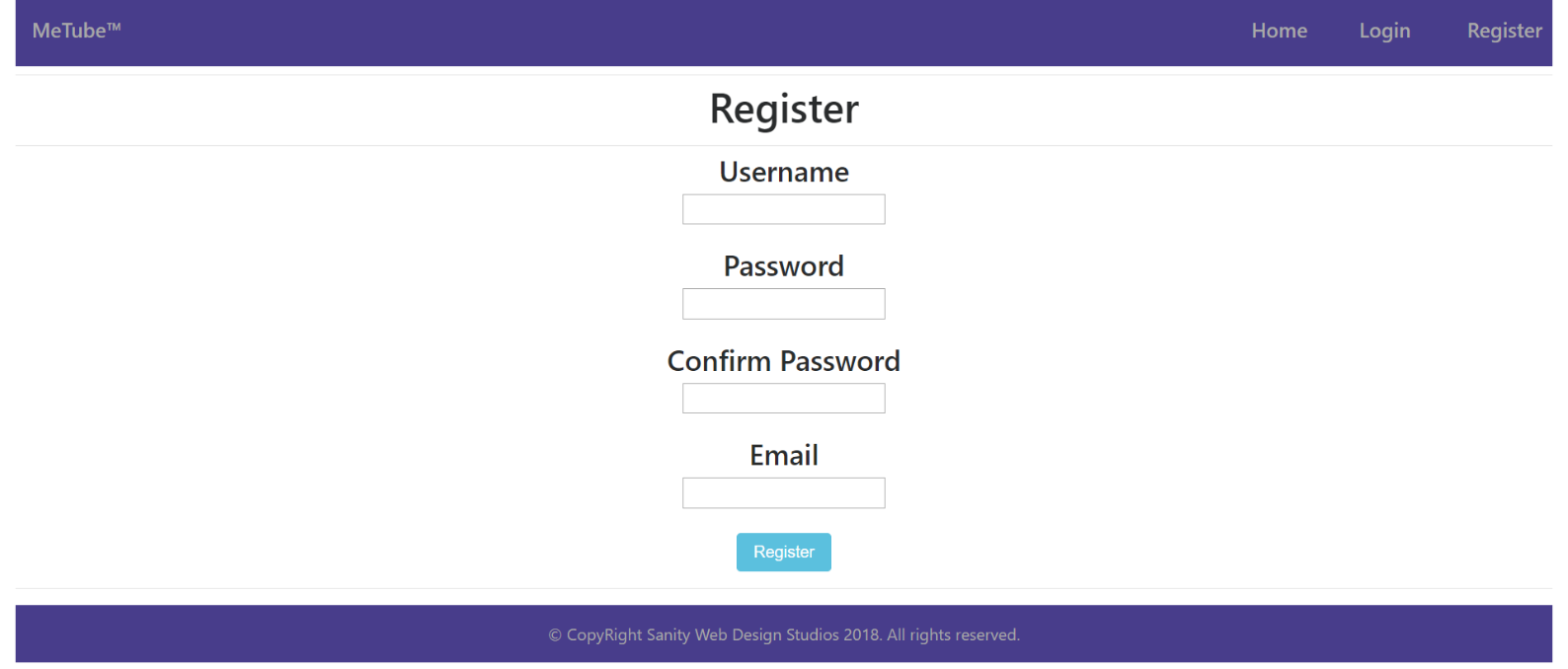
### Index Template (logged out user)



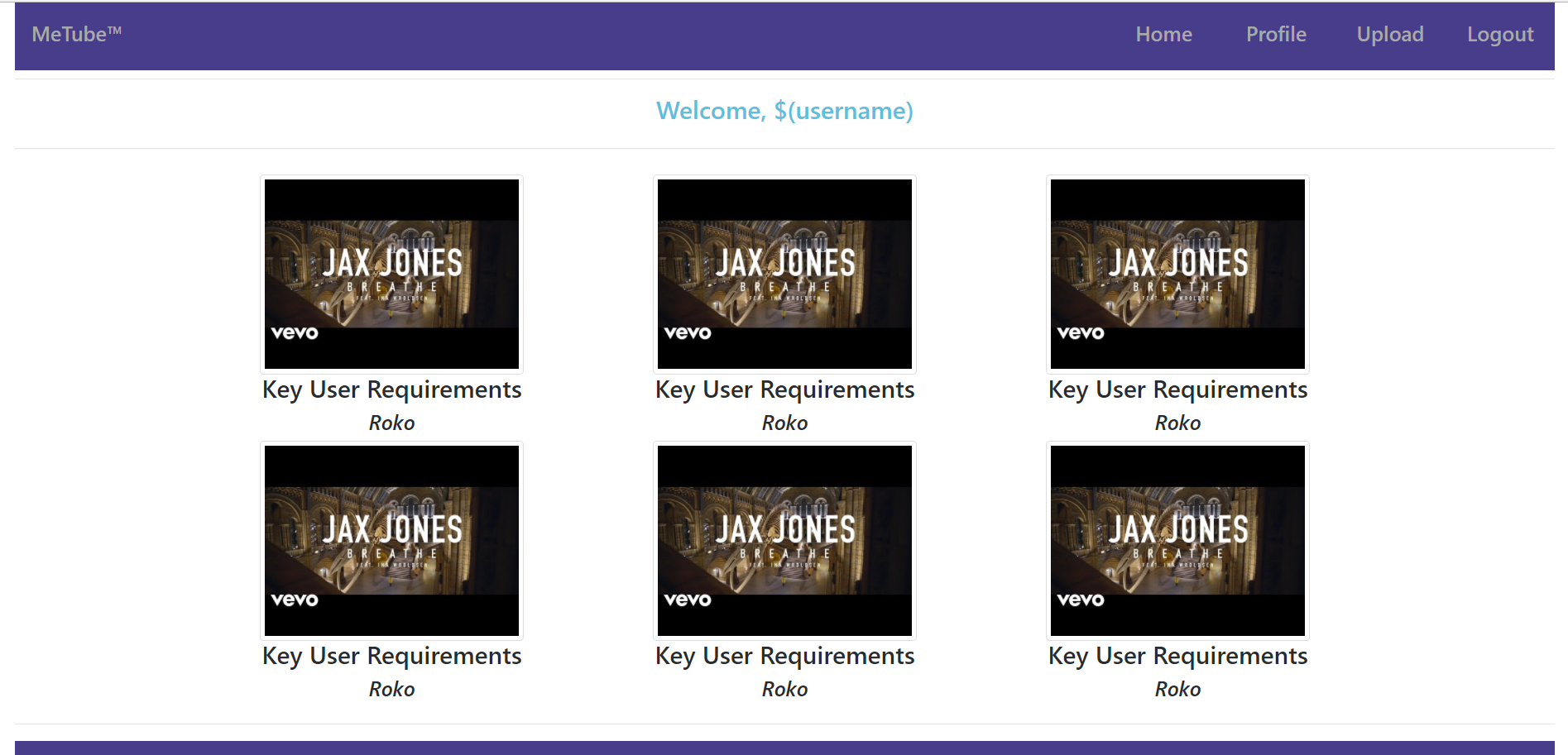
### Login Template (logged out user)



### Register Template (logged out user)



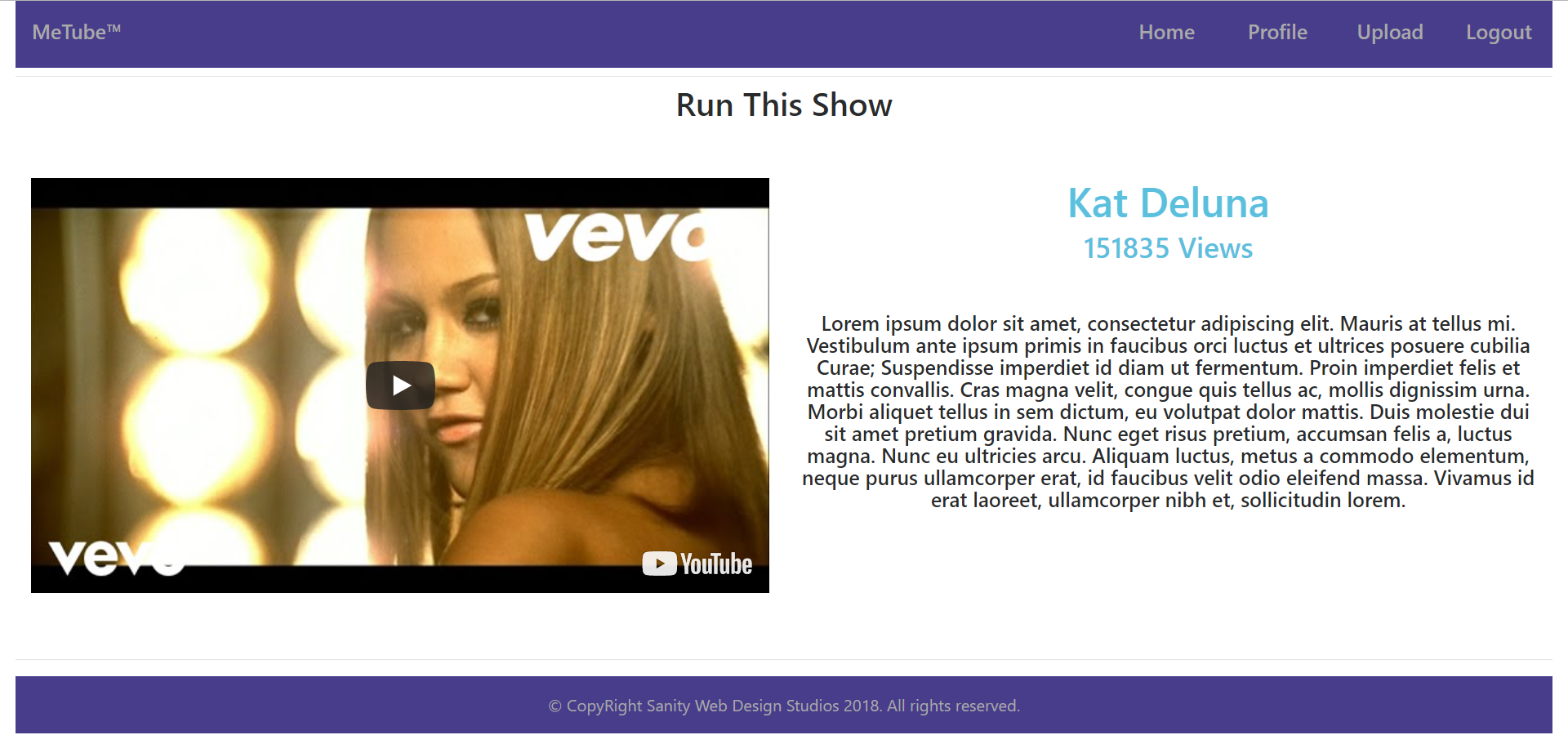
### Home Template (logged in user)



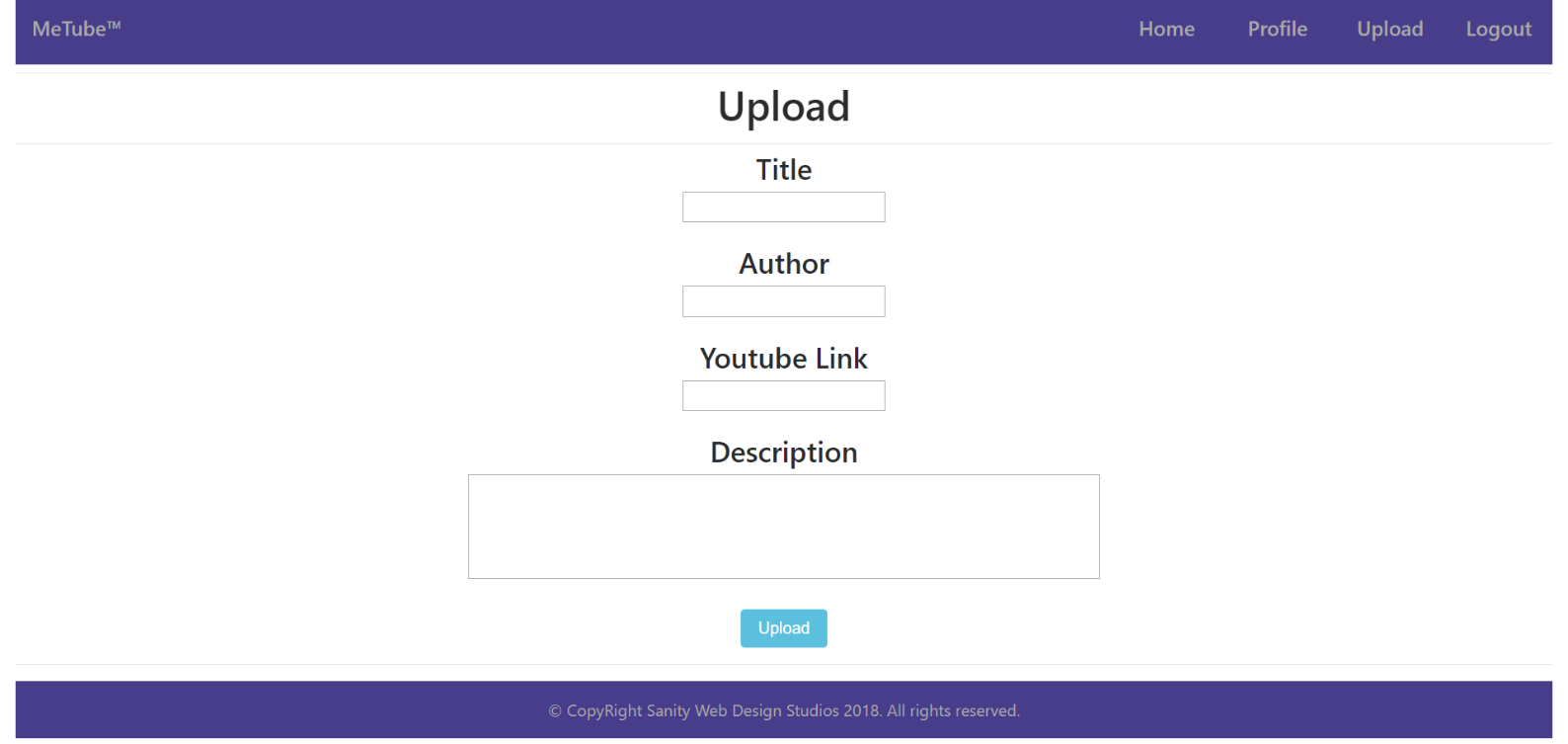
### Profile Template (logged in user)



### Tube Details Template (logged in user)



### Tube Upload Template (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 Tubes (Home page)
  + The Home page holds **image thumbnails**.
* **View** their **Profile** (Clicking on [Profile] button on Home page)
  + **Only** the **Tubes** uploadedbythe **User** should be **viewed.**
  + The **username** and the **email** of the user should be viewed in the **format specified** in the screenshots above**.**
* **View** **details** about Tube
  + (Clicking on a Tube’s picture on Home page)
  + (Clicking on the [Details] button on Profile page)
  + **Each time** you view details about a Tube, you should increment it’s views by 1.
  + The Details page holds a **video iframe**.
* Upload a Tube.
  + The upload is done with a youtube video link.
    - (Example: “https://www.youtube.com/watch?v=uGhKqb2Ow3E”)
  + The last segment of the video is its id - uGhKqb2Ow3E.
  + You can use that id to **extract** the thumbnail, in order to view it on the Home page.
  + You can use that id to **create** an iframe, in order to view it on the Details page.

The **application** should **store** its **data** into a MSSQL database, using EF Core.

## 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 Profile page.
* Users (logged in) can access Upload functionality.
* Users (logged in) can access Details functionality.
* Users (logged in) can access Logout functionality.

## Framework Requirements

Extend the given skeleton (Web Server, MVC Framework), by adding a validation before the mapping of the binding model. The validation result should be accessed by adding it as a parameter to a controller action.

The validation should be a class, which holds a map of errors which happened during the mapping of the binding model and the **request body**.

The validation should be very simple, for example – “Mapping of the \”title\” field failed.”. You don’t need to have custom messages, annotations or type & value limitations.

## Scoring

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

### Database Requirements – 10 points.

### Template Requirements – 30 points.

### Functional Requirements – 40 points.

### Security Requirements – 10 points.

### Framework Requirements – 10 points.