Java Web Development Basics

Exam Preparation

Casebook

Exam Preparation problems for the "Java Web Development Basics" course @ SoftUni. Submit your solutions on the course page, so that you can be evaluated by your fellow colleagues.

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.

1. Database Requirements

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

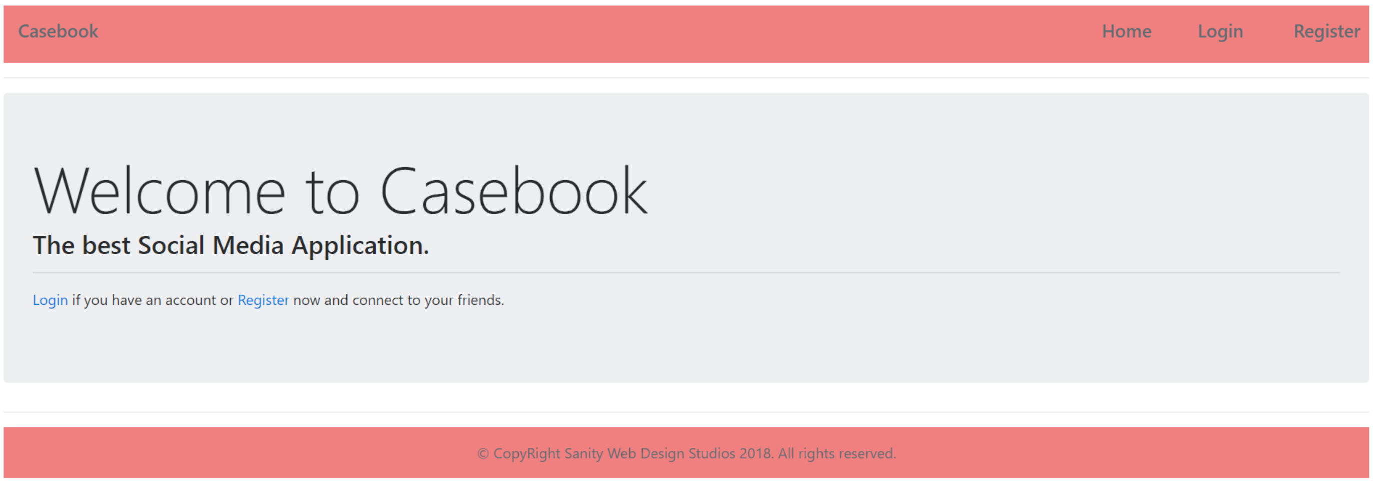
User

- 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.

2. Pages

Index Page (logged out user)



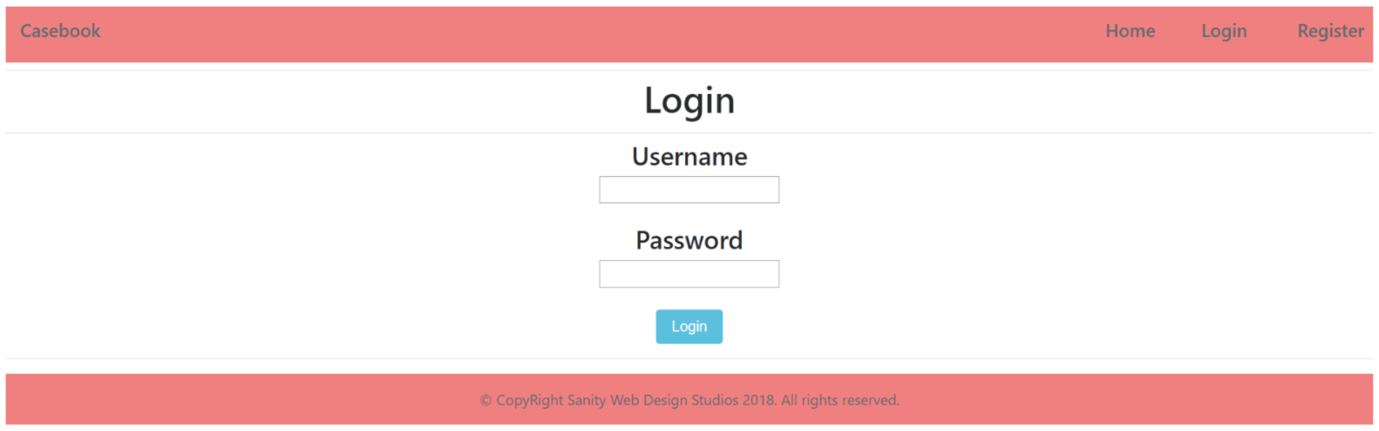
Welcome to Casebook

The best Social Media Application.

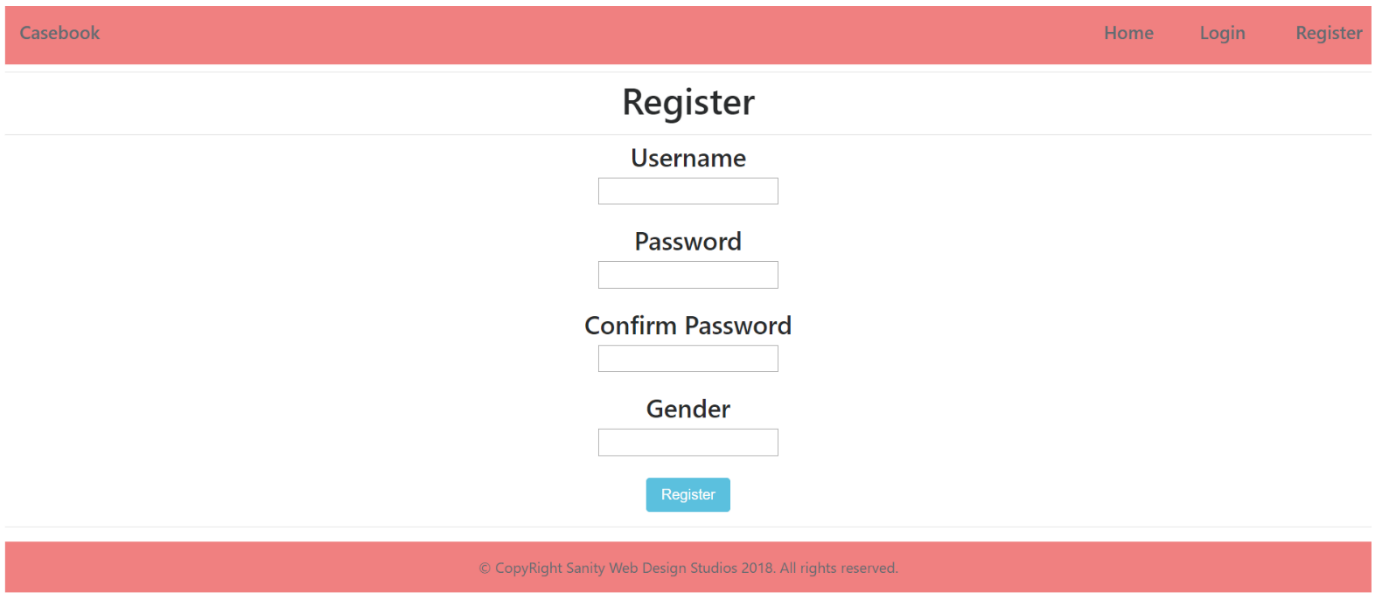
Login if you have an account or Register now and connect to your friends.

© CopyRight Sanity Web Design Studios 2018. All rights reserved.

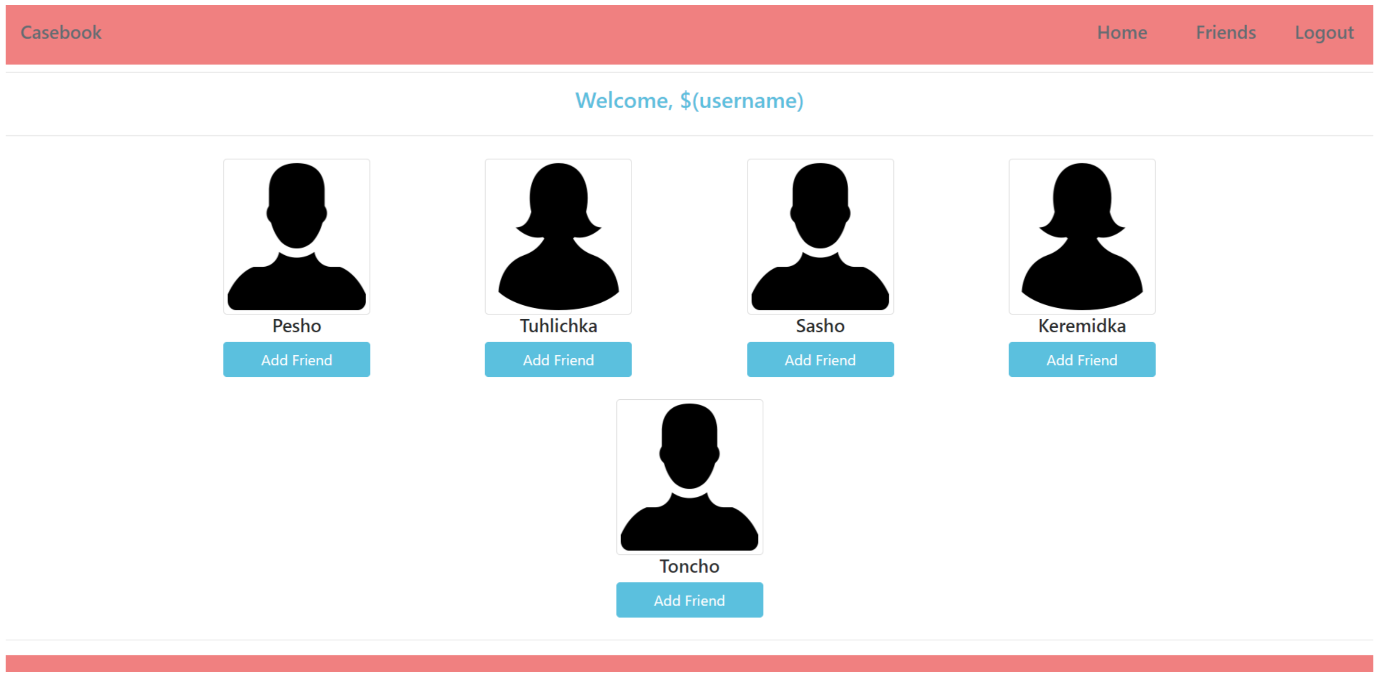
Login Page (logged out user)


Casebook
Home
Login
Register
Login
Username
Password
Login
© CopyRight Sanity Web Design Studios 2018. All rights reserved.

Register Page (logged out user)

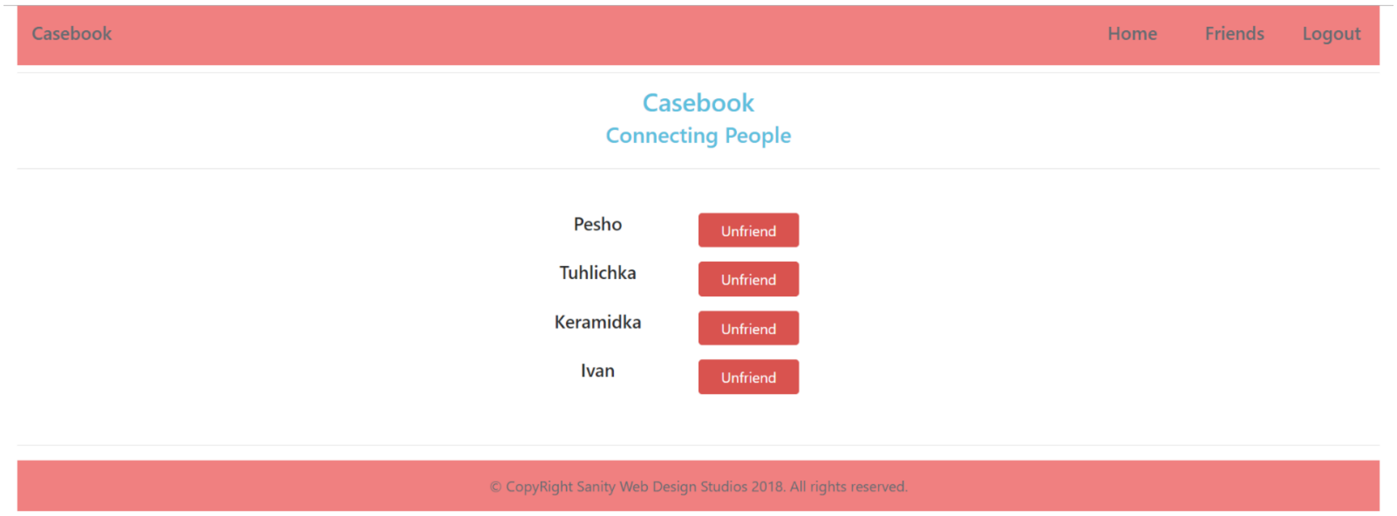


Home Page (logged in user)



NOTE: People are visualized on the **Home Page** in **rows** by **4**.

Friends Page (logged in user)



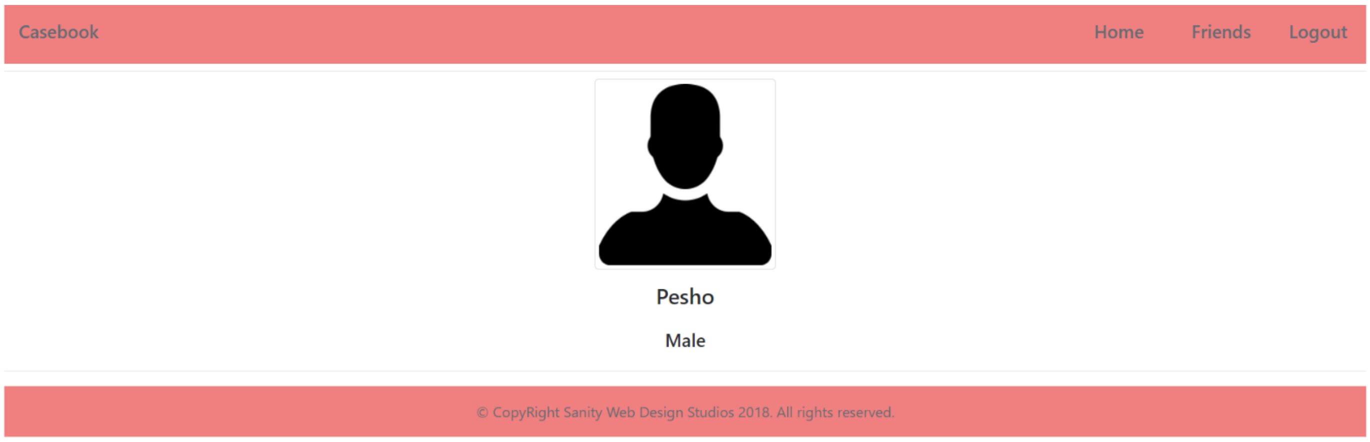
Casebook | Home | Friends | Logout

Casebook
Connecting People

Pesho [Unfriend]
Tuhlichka [Unfriend]
Keramidka [Unfriend]
Ivan [Unfriend]

© CopyRight Sanity Web Design Studios 2018. All rights reserved.

Profile Page (logged in user)



Casebook | Home | Friends | Logout

Pesho

Male

© CopyRight Sanity Web Design Studios 2018. All rights reserved.

The templates have been given to you in the application skeleton, so make sure you implement the pages correctly.

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.

3. Functionality

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.

4. Security

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) cannot access **Guest** pages.
- **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.

5. Code Quality

Make sure you provide the best architecture possible. Structure your code into different modules, divide and conquer, follow the principles of high-quality code. You will be scored for the Code Quality and architecture of your project.

6. Scoring

Database – 10 points.

Pages – 15 points.

Functionality – 30 points.

Security – 15 points.

Code Quality – 30 points.