Swe573 Software Development Practice, Spring 2024

Project Final Deliverables

Name: Efe Göçmen

Course: SWE 573 Software Development Practice, Spring 2024

Date: 19.05.2024

Project Name: DEVCOM

Deployment URL: http://13.53.129.204:8080/

Git Repository URL: https://github.com/efestrikesback/SWE-573-Software-Development-Practice

Git Tag Version URL:

HONOR CODE

Related to the submission of all the project deliverables for the SWE 573 2024 Spring semester project reported in this report, I Efe declare that:

- I am a student in the Software Engineering MS program at Bogazici University and am registered for the SWE 573 course during the Spring 2024 semester.

- All the material that I am submitting related to my project (including but not limited to the project repository, the final project report, and supplementary documents) have been exclusively prepared by myself.

- I have prepared this material individually without the assistance of anyone else with the exception of permitted peer assistance which I have explicitly disclosed in this report.

Efe Göçmen

Table Of Contents

1. Overview

Provide a summary of your project, its objectives, and the main features.

The Devcom App is designed to offer a versatile platform for users to create, manage, and engage with communities based on various interests. It aims to provide an intuitive and customizable environment where users can interact, share content, and efficiently moderate their communities.

Objectives:

Community Engagement: Foster user interaction within interest-based communities.

Ease of Use: Provide an intuitive user interface for easy navigation and community management.

Customization: Allow users to customize their profiles and community settings.

Main Features:

Ability to create communities with unique identifiers and descriptions.

Simplified login with username and password.

Profile customization with pictures and nicknames.

Template-driven content creation.

Content searching.

Owner and member roles.

Member count.

2. Software Requirements Specification

Detail the functional and non-functional requirements of your project.

Community Management

1) Create Communities: Allow users to create communities by defining community name, post templates, and description.

2) Post Templates: Provide a mechanism to create information schemas for posts to define shared information types.

3) Multiple Schemas: Allow builders to create multiple schemas as needed.

4) Data Fields: Enable adding data fields to schemas, marking them as required or optional, with specified data types (e.g., geolocation, date).

5) Unlimited Fields: No limit on the number of data fields in a schema.

6) Schema Management: Provide mechanisms to create, update, and delete schemas after community creation.

7) Enforced Schema Selection: Require users to choose a community-specific schema before posting.

Authorization, Roles, and Permissions

8) Roles: Include roles such as "Owner," "Creator," "Moderator," and "User," with "User" as the global default role.

9) Creator Role: Assign one "Creator" per community, who cannot be changed.

10) Promotion/Demotion: Allow "Creator" to promote/demote "Users" to/from "Owner."

11) Owner Restrictions: Prevent "Owners" from demoting or kicking other "Owners."

12) Moderator Role: Allow "Creators" and "Owners" to promote/demote "Users" to/from "Moderator."

13) Moderation Powers: Allow "Moderators" to kick "Users" and edit/delete their posts.

14) Reporting: Enable "Users" to report posts and "Moderators."

Identity Management

15) Profile Page: Provide a profile page for each "User."

16) Profile Customization: Allow "Users" to have an avatar, description, and username.

17) Manage Identity: Enable "Users" to create/update/delete profile information.

18) Community Roles: Display a list of all communities subscribed to and roles assigned on the profile page.

19) Manage Subscriptions: Allow "Users" to manage community subscriptions on the profile page.

Security and Interaction

20) No Direct Messaging: Disallow direct messaging between "Users."

21) View Profiles: Allow "Users" to view other "Users'" avatars, descriptions, and community subscriptions.

22) Post Interaction: Allow "Users" to like/upvote and comment on posts.

23) Authentication: Provide an authentication mechanism.

24) Post Priority: Prioritize most liked/upvoted posts in community main pages.

25) Community Tags: Allow "Creators" and "Owners" to tag/label communities.

26) Community Recommendations: Recommend communities to "Users" based on tags of subscribed communities.

27) Private Communities: Allow communities to be made private, with recruitment via invitation only.

28) Search Mechanism: Provide search mechanisms for finding communities and posts.

Non-Functional Requirements

1) Scalability: The system should handle a large number of users, communities, and posts efficiently.

2) Performance: Ensure quick response times for all actions, especially searching and schema-related operations.

3) Usability: The user interface should be intuitive and easy to navigate for all user roles.

4) Security: Implement robust security measures to protect user data and community content.

5) Reliability: Ensure high availability and minimal downtime.

6) Maintainability: Codebase should be modular and well-documented to facilitate maintenance and updates.

7) Data Integrity: Ensure accurate and consistent data, especially in schema management and role assignments.

8)Compliance: Adhere to relevant data protection and privacy regulations.

9) Options for content sorting: recent, trending, hot.

3. Design Documents

Include diagrams and descriptions of your system architecture, database schema, and any other relevant design documentation.

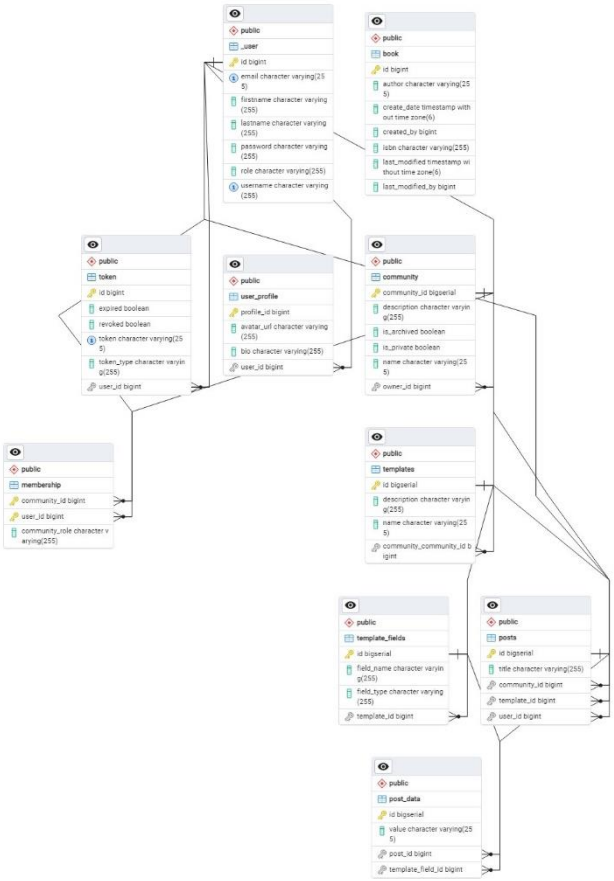

Figure 1: Database entity relation diagram

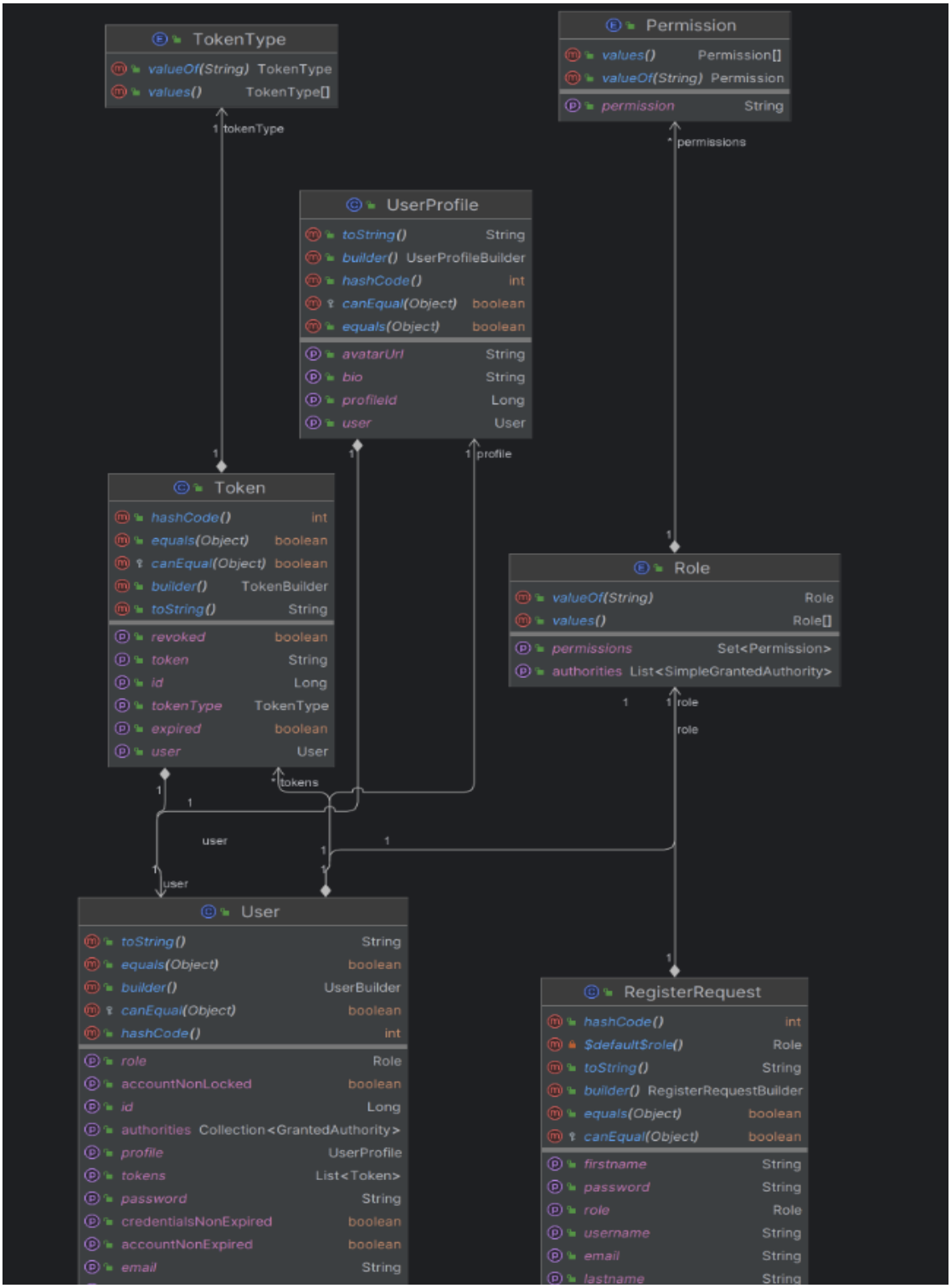

Figure 2: *Small part of final class diagram please see https://github.com/efestrikesback/SWE-573-Software-Development-Practice/wiki/Final-Deliverables for high resolution version.*

4. Project Status

Status of Each Requirement

List each requirement and its current status (completed/not completed). Include notes on documentation, testing, and deployment for completed requirements.

Requirement (Functional)	*Status*	*Notes*
Create Communities	*completed*	*See* 7.User Manual & Test Results
Post Templates	*completed?*	*See* 7.User Manual & Test Results
Multiple Schemas	*completed*	*See* 7.User Manual & Test Results
Data Fields	*completed?*	*See* 7.User Manual & Test Results *Only two data types are allowed.*
Unlimited Fields	*completed*	*See* 7.User Manual & Test Results *It is possible to add unlimited number of fields to a template.*
Schema Management	*not completed*	*It is only possible to create a template. Update and delete are not implemented.*
Enforced Schema Selection	*completed*	*See* 7.User Manual & Test Results
Roles	*completed*	*See* 7.User Manual & Test Results Owner and member roles and role based access is fully functional.
Creator Role	*completed*	*See* 7.User Manual & Test Results
Promotion/Demotion	*not completed*	*not implemented.*
Owner Restrictions	*completed*	*See* 7.User Manual & Test Results
Moderator Role	*completed*	*Implemented as "Owner"*
Moderation Powers	*not completed*	*not implemented.*
Reporting	*not completed*	*not implemented.*
Profile Page	*completed*	*See* 7.User Manual & Test Results
Profile Customization	*completed*	*See* 7.User Manual & Test Results
Manage Identity	*not completed*	*not implemented.*
Community Roles	*completed*	*See* 7.User Manual & Test Results Owner and member roles and role based access is fully functional.
Manage Subscriptions	*completed*	*See* 7.User Manual & Test Results is possible to leave & join communities.
No Direct Messaging	*completed*	*Implemented*
View Profiles	*completed?*	*User can only view their own profile.*
Post Interaction	*not completed*	*not implemented.*
Authentication	*completed*	*See* 7.User Manual & Test Results
Post Priority	*not completed*	*not implemented.*
Community Tags	*not completed*	*not implemented.*
Community Recommendations	*not completed*	*not implemented.*

Private Communities	*not completed*	*not implemented.*
Search Mechanism	*completed*	*See* 7.User Manual & Test Results Basic keyword based search is fully functional

5. Deployment & Dockerization Status

Dockerized. Deployed on Amazon EC2. http://13.53.129.204:8080/

Docker Files:

https://github.com/efestrikesback/SWE-573-Software-Development-Practice/blob/main/Dockerfile

https://github.com/efestrikesback/SWE-573-Software-Development-Practice/blob/main/docker-compose.yml

6. Installation Instructions

Local testing

Clone the Git repository: git clone https://github.com/efestrikesback/SWE-573-Software-Development-Practice.git

Download PostgreSQL and create a database named "DEVCOM": PostgreSQL

Change properties from application.yml if needed

Change Javascript API calls to from http://13.53.129.204:8080/ to http://localhost:8080/

Make sure you have OpenJDK 17: java -version

Build the application using Maven from Maven Goals: mvn clean install

Docker files content

DOCKER COMPOSE

```
version: '3.8'

services:

  db:

    container_name: postgres

    image: postgres

    ports:

      - "3000:5432"

    environment:

      POSTGRES_PASSWORD: 123

      POSTGRES_DB: DEVCOMDB

      POSTGRES_USER: postgres

    volumes:

      - ./pgdata:/var/lib/postgresql/data

    restart: always


  backend:

    container_name: backend

    build:

      context: .

      dockerfile: Dockerfile

    ports:

      - "8080:8080"

    environment:

      SPRING_DATASOURCE_URL: jdbc:postgresql://db:5432/DEVCOMDB

      SPRING_DATASOURCE_USERNAME: postgres

      SPRING_DATASOURCE_PASSWORD: 123

    depends_on:
```

```
    - db
  restart: always
  privileged: true
```

DOCKER

```
FROM openjdk:17.0.2-slim-bullseye
WORKDIR /app
COPY . .
RUN ls -la
RUN chmod +x mvnw
#RUN sed -i 's/\r$//' mvnw
#RUN /bin/sh mvnw dependency:resolve
RUN ./mvnw clean install -DskipTests
CMD java -jar ./target/DEVCOM-0.0.1-SNAPSHOT.jar
```

7. User Manual & Test Results

Describe how to use your system, including any necessary credentials for testing.

Email	Password	Role
ahmet@ahmet.com	ahmet@ahmet.com	Owner of “EU4 Fans”
fer@fer.com	fer@fer.com	Owner of “GYM RATZ”
dira@gmailcom	dira@gmailcom	Has no profile, has no ownership, has no membership
efx99@gmail.com	efx999	Member of all communities, has profile
cankut@outlook.com	aaa	Has no profile, has no ownership, has no membership
efeG@outlook.com	efe	Has a profile, owner of “Java Developers” , member of “GYM RATZ”

Register

Visit http://13.53.129.204:8080/ click “Register!”, fill the form.

Each user has to have a unique username and a unique email & password pair!

After filling the form click **“Register”** button at right hand side.

Note: Dropdown selection is for a WIP feature, which grants admin role to user to enable him/her to test every feature freely. Currently does not work.

Let’s register

```
{
   "firstname": " Efe",
   "lastname": " Gocmen",
   "username": " efeG",
   "email": " efeG@outlook.com",
   "password": "efe"
}
```

Figure 3: *Register Form*

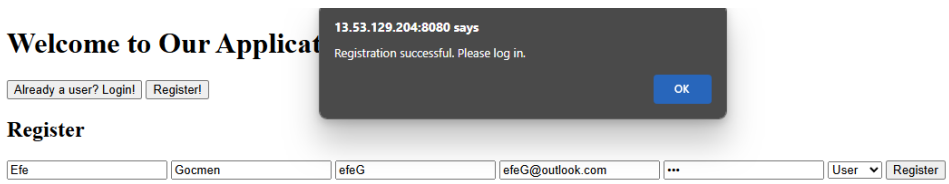

Figure 4: *Successful registeration*

When we click **"Ok"** we will be redirected to login form.

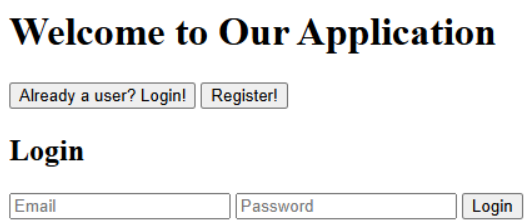

Figure 5: *Login & authentication*

We can immediately login or leave, comeback later, and login. **"Registration successful."** Shows successful registration.

Let's try to register (changed nothing)

{

"firstname": " Efe",

"lastname": " Gocmen",

"username": " efeG",

"email": " efeG@outlook.com",

"password": "efe"

}

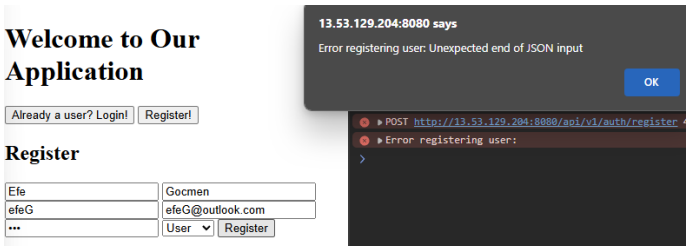

Figure 5: *Trying duplicate registeration*

We get generic error for unique constraint errors.

Let's try to register (changed username but email & password pair is same)

{

"firstname": " Efe",

"lastname": " Gocmen",

"username": " efeG2",

"email": " efeG@outlook.com",

"password": "efe"

}

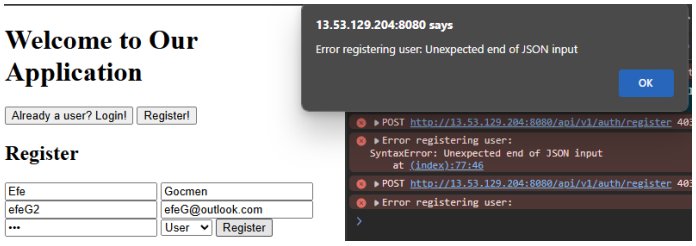

Figure 5: *Trying duplicate* email & password

Once again, we get generic unique constraint error.

Login

We can login immediately after registration or any other time we like if we are already registered.

Let's login with our registered user. We click **"Already a user? Login!"** button and fill the form with our previously registered user information, then we click **"Login"** button at the right-hand side. Let's make typo on purpose to see message.

{

"email": "efeg@outlook.com",

"password": "efe"

}

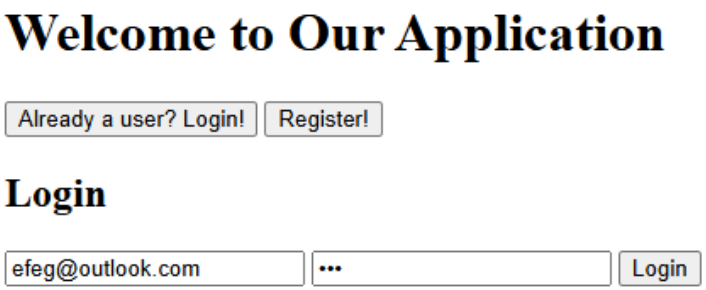

Figure 6: *Trying authentication with email typo.*

Figure 7: *Login failed response.*

We got **"Login failed:"** generic error which indicates failed authentication.

Now with correct values

```
{
   "email": "efeG@outlook.com",
   "password": "efe"
}
```

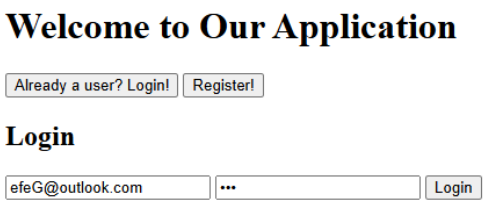

Figure 8: *Login with correcet credentials.*

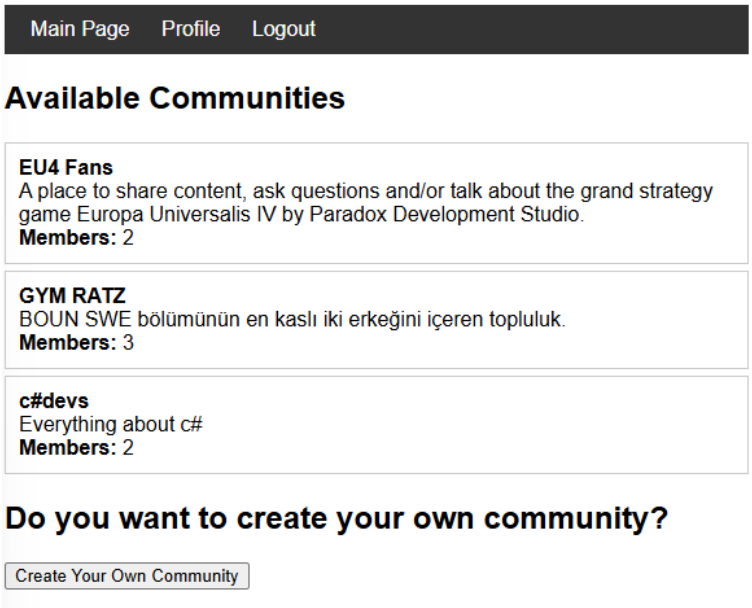

Figure 9: *Main page.*

After successful login we are immediately redirected to main page.

Create Profile

Let's create a profile for our user. From navigation bar we select **"Profile"** fill the forms then click "Create Profile" button.

{

"bio": "efe's Profile",

"avatarUrl": " https://wallpaperset.com/w/full/7/3/8/29781.jpg"

}

Figure 10: *User profile form.*

Figure 11: *Successful creation of user profile form.*

When we click **"OK"** on the popup message we immedialty see our profile.

Let's edit our profile before returning to main page.

To edit we click **"Update Profile"** button.

Figure 11: *Profile update.*

During editing process, if desired we can cancel editing by clicking **“Cancel”** button, if we want to proceed with update, we edit the form then click **“Update Profile”** button.

Let’s set our new profile as
{

"bio": "efe's NEW Profile",

"avatarUrl": " https://wallpaperset.com/w/full/7/3/8/29781.jpg”

}

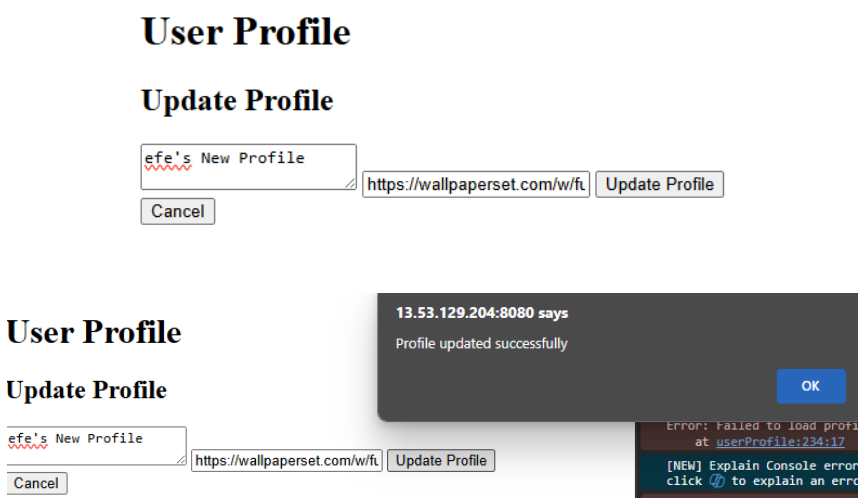

Figure 12: *Profile update success message.*

Once again when we click **“OK”** we will see the udpated profile.

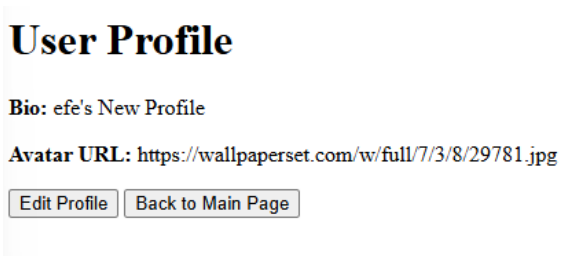

Figure 13: *Updated profile.*

Updated profile with new values. Let’s click **“Back to Main Page”** to retrun to the main page.

Visit a Community

At main page we can see a list of communities with basic information about them, their name, description and member count respectively.

Let's visit "c#devs". To visit a community, we simply click on them.

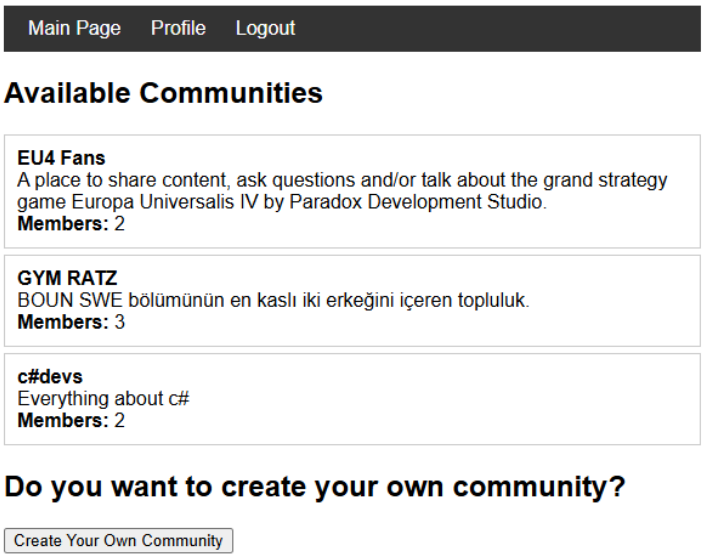

Figure 14: *Community list.*

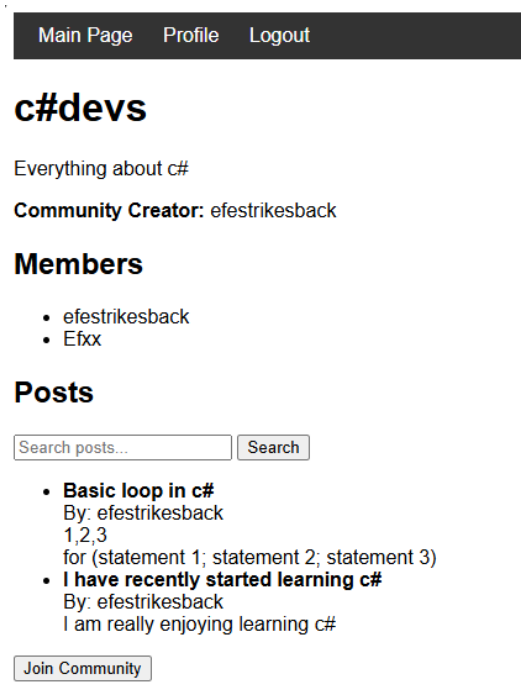

Figure 14: *Community page view.*

Since we are not a member only basic search **"Search"** and joining **"Join Community"** is available to us.

Become a Member

To join a community, we simply visit their page by clicking on their name at the main page. Then we click **"Join Community"** button.

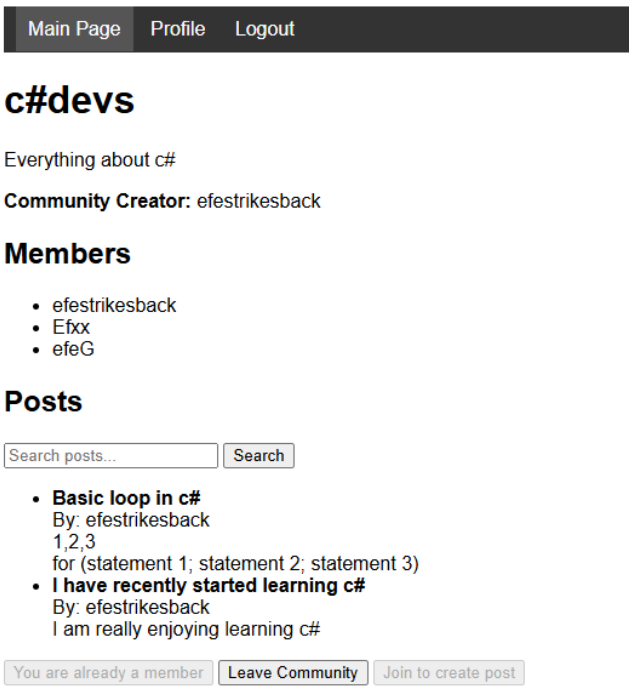

Figure 15: *A member's view of community page.*

After joining, we can see that member list is updated and shows our username as well.

[BUG]: Even though we have just joined we see a "Join to create post" box. It is due to a small overlook and will be fixed in next release.
Simply click F5 or return to main page and comeback to community again for temporary solution.

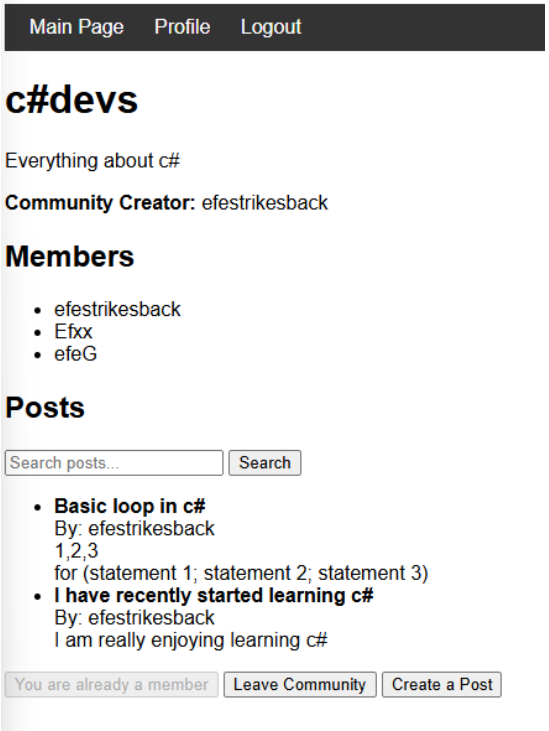

Figure 15: *Create a post is visible after refresh.*

After refreshing the page with F5 key we can see that "Create a Post" button is now available, it checks user membership status correctly.

Create Your First Post

To create a post in community the user has to be a member! Visitors cannot create posts!

Let's create a post in "c#devs" with our community member user "efeG"
We click **"Create a Post"** button, then **"Create a Post"** card (gray area) with post creation form shows up.

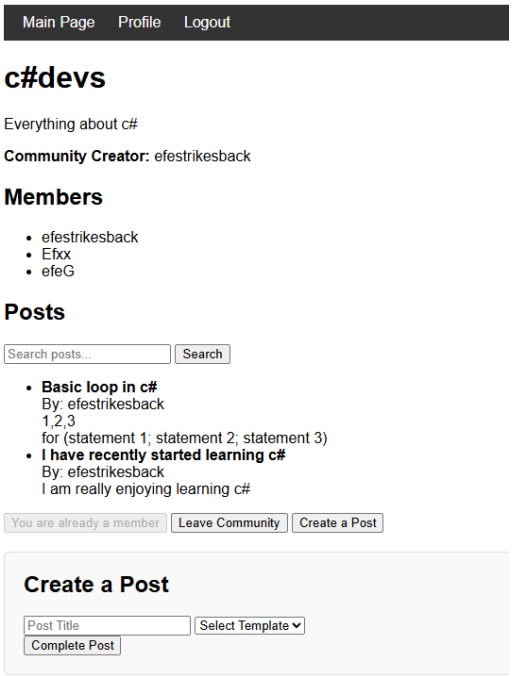

Figure 16: *Post creation card.*

Before creating our post we have to choose a template created by community owner.

We can choose a post template from **"Select Template"** dropdown.

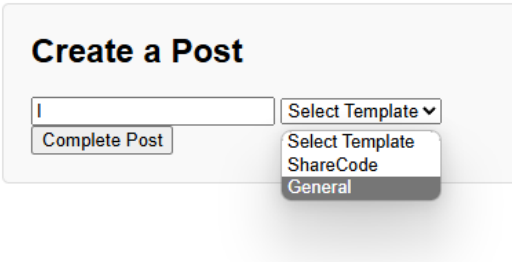

Figure 17: *Template dropdown.*

Let's choose "ShareCode" template for demonstration.

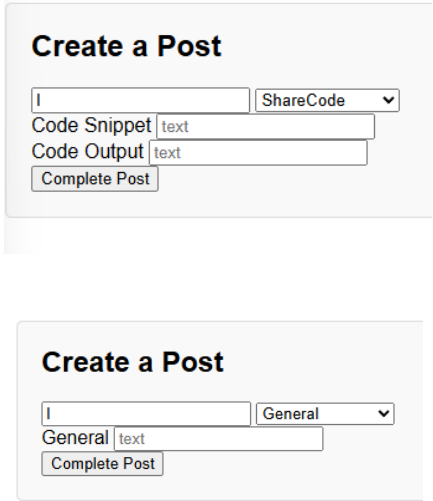

Figure 18: *Comparison of fields of two different templates.*

The mandatory fields of ShareCode showed up, let's try "General" template now.

It just requires us to fill a general form.

Let's continue with "ShareCode" and fill the forms. After filling we will click **"Complete Post"** to finish posting.

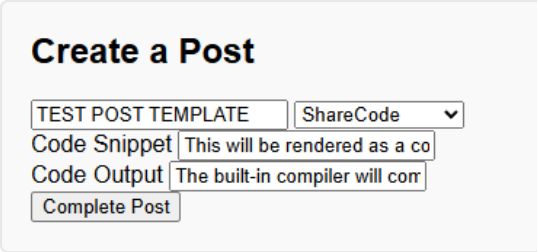

Figure 19: *Filling post fields.*

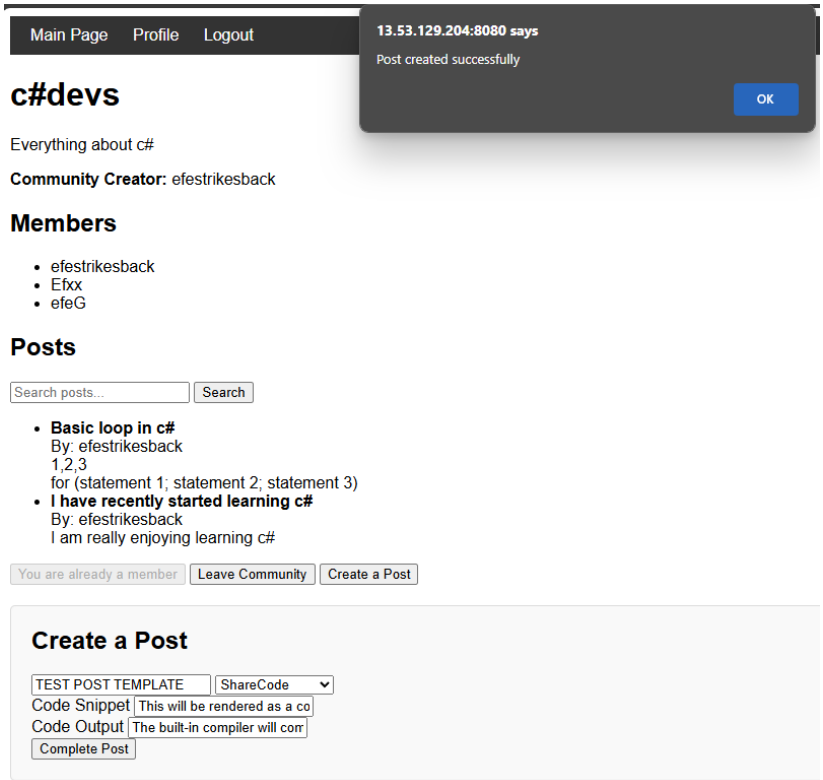

Figure 20: *Success message for post creation.*

When we click **"OK"** the page will update Posts and we will see our post!

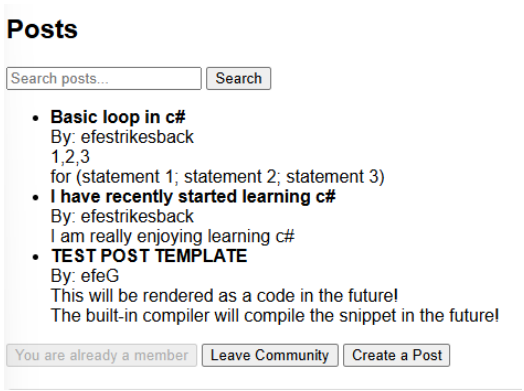

Figure 21: *Created post is now visible among other posts.*

We can see our newly created post now!

Create a Community

Let's create our own community. We click **"Create Your Own Community"** button.

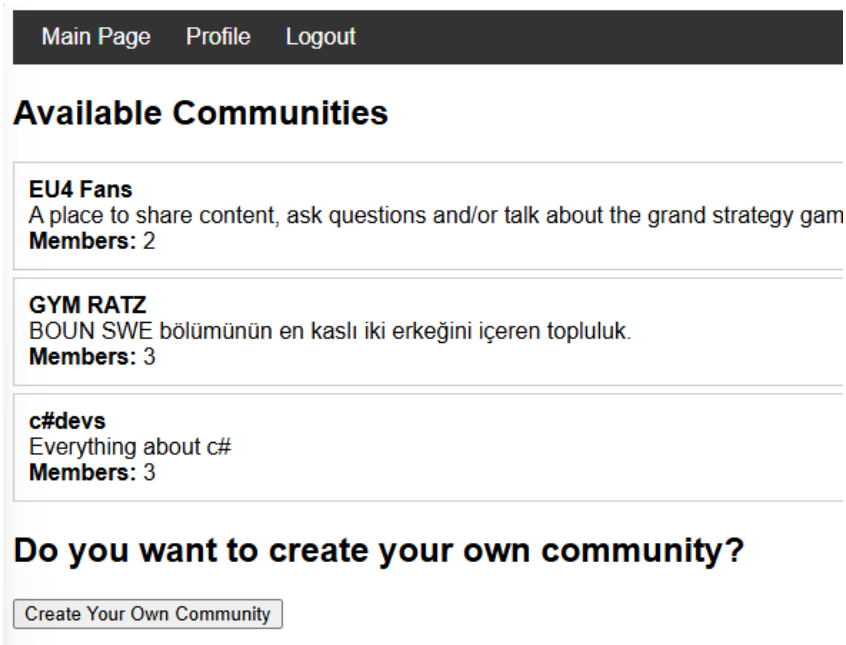

Figure 22: *Community creation.*

After clicking we can either fill the form and create community or modify url "http://13.53.129.204:8080/createCommunity" to "http://13.53.129.204:8080/mainPage" to return to the main page (fully functional navigation bar is already implemented I just need to copy that code to this page, which will be done at the next release).

Private communities and archiving are not implemented yet therefore we will not check the checkboxes.

Let's fill the forms and click **"Create"**

Figure 22: *Community creation forms and the success message.*

After clicking **“OK”** we get redirected to main page and we see our community listed among others.

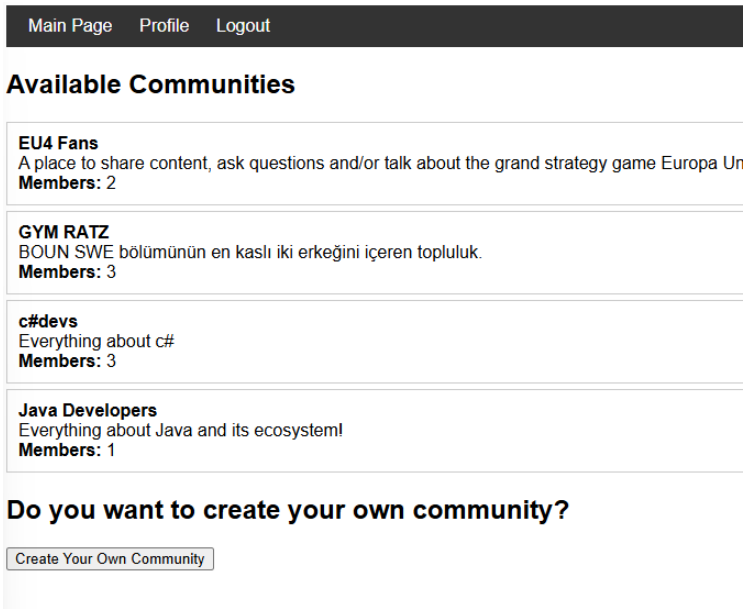

Figure 23: *New community is added to the list.*

Create a Post Template

Let’s create a post template for our community so that members can use it to create posts.

If the community owner visits his own community page, he/she can see a “**Create Template”** card, this is an owner exclusive feature so regular members or visitors cannot see it.

Before adding fields, we need to create a template because builder pattern is used to create then add fields to a template.

Let’s try to add fields with no template.

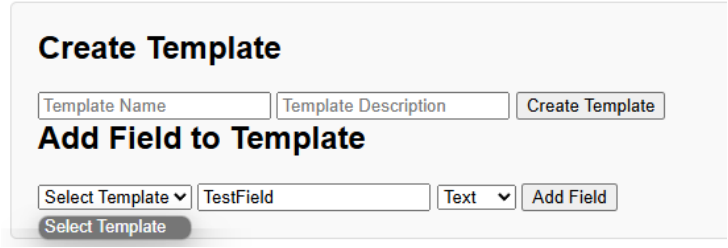

Figure 23: *Template and field card.*

After filling the field form and clicking **“Add Field”** we get an error message.

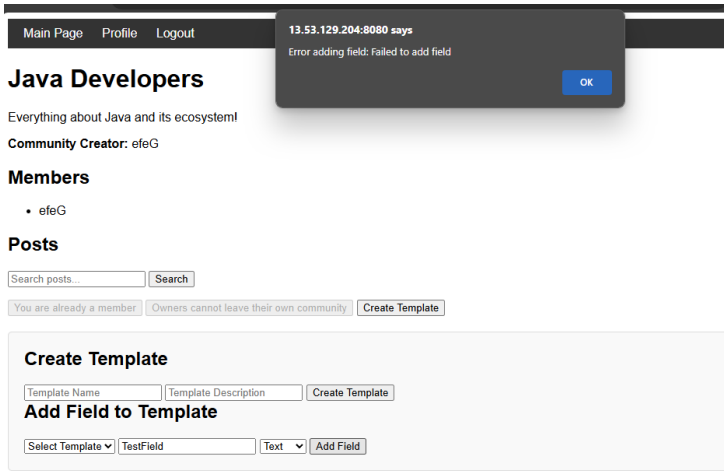

Figure 23: *Trying to add field without selecting a template.*

Let’s try after creating a template.

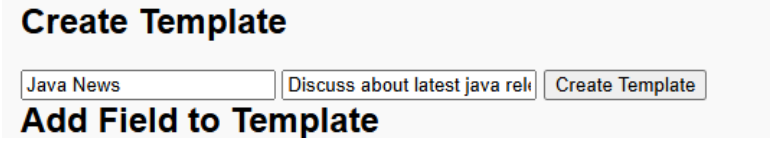

Figure 23: *Adding field with selecting a template.*

We fill the form and click **“Create Template”**

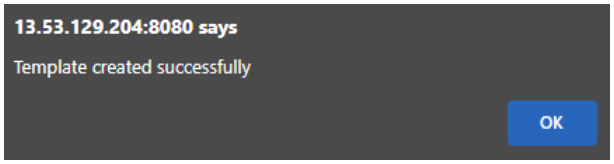

Figure 24: *Template creation success.*

We press **"OK"** and then we can see our new template is now available in **"Select Template"** dropdown.

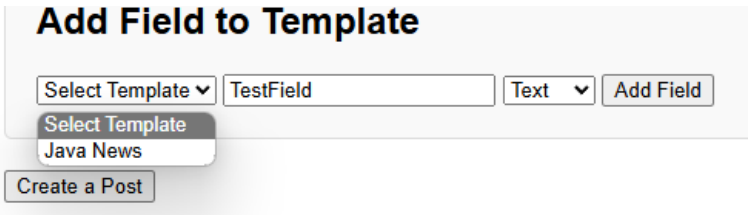

Figure 25: *Available templates.*

I have also added "Java Coding Help" template to further populate the dropdown.

Add Fields to Your Template

Just like template creation this feature is also owner exclusive!

Choose a template to add fields from **"Select Template"**, fill **"Field Name"** form choose data type from dropdown with **"text"** defaut value then **click "Add Field"** to complete addition.

Figure 26: *Field addition to template.*

The form does not automatically gets cleared after submission we can simply erase question and write another field name and choose another data type.

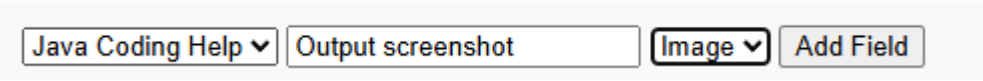

Figure 27: *Form is still full after addition.*

We see “Field added successfully” message once again and click “OK” to continue.

Create a Post w/ Newly Created Template

Let’s test our new template “Java Coding Help”, when we click **“Create a Post”** button, **“Create a Post”** card appears.

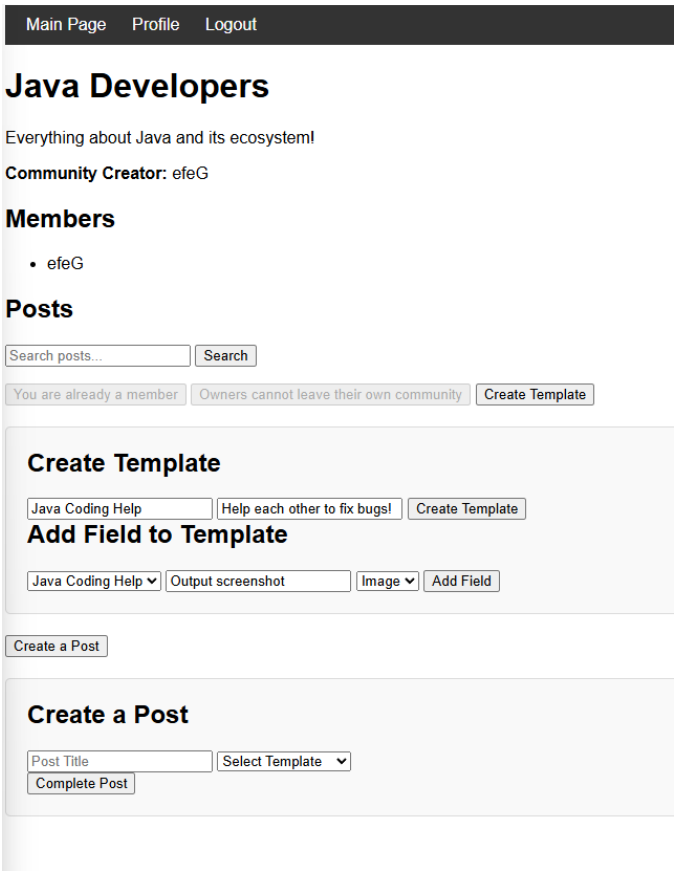

Figure 28: *Create a post card shows up.*

Each template selection brings their own fields.

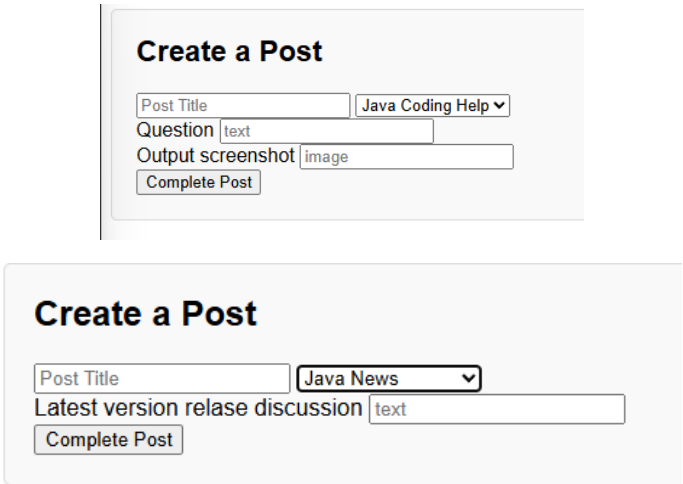

Figure 29: *Comparison of different templates.*

We choose “Java Coding Help”, fill the forms then click **“Complete Post”** button.

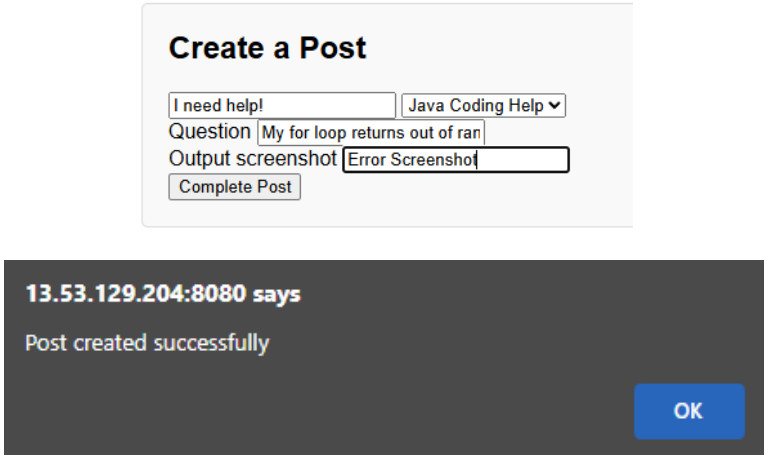

Figure 29: *Successful post creation.*

Our post is successfully created, we click **“OK”.**

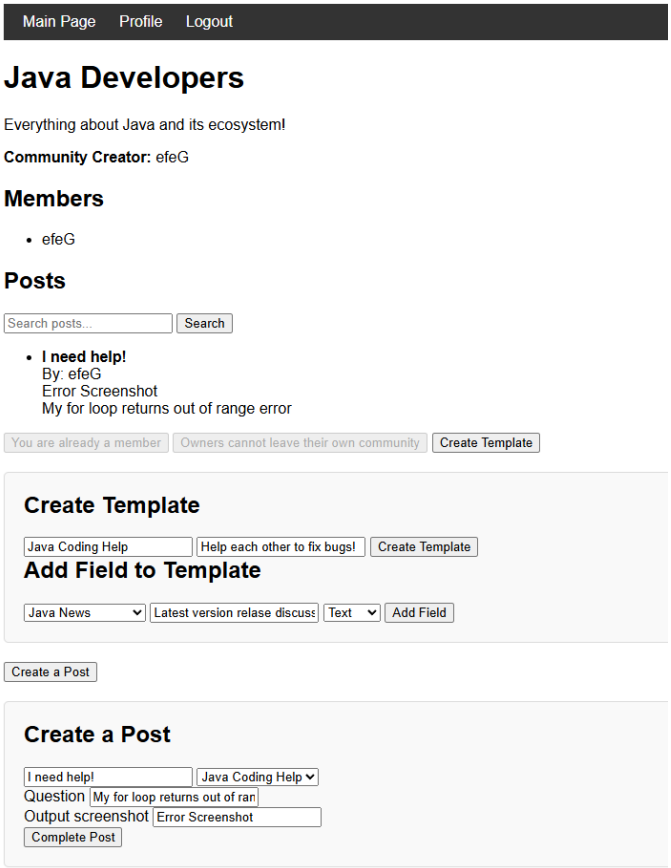

Figure 30: *post shows up.*

We have completed our posting!

Use Post Searching

Let's navigate back to "c#devs" community to try search feature. We fill the **search form** with a word, click **"Search"** then search engine checks every post and their content to retrieve related posts.

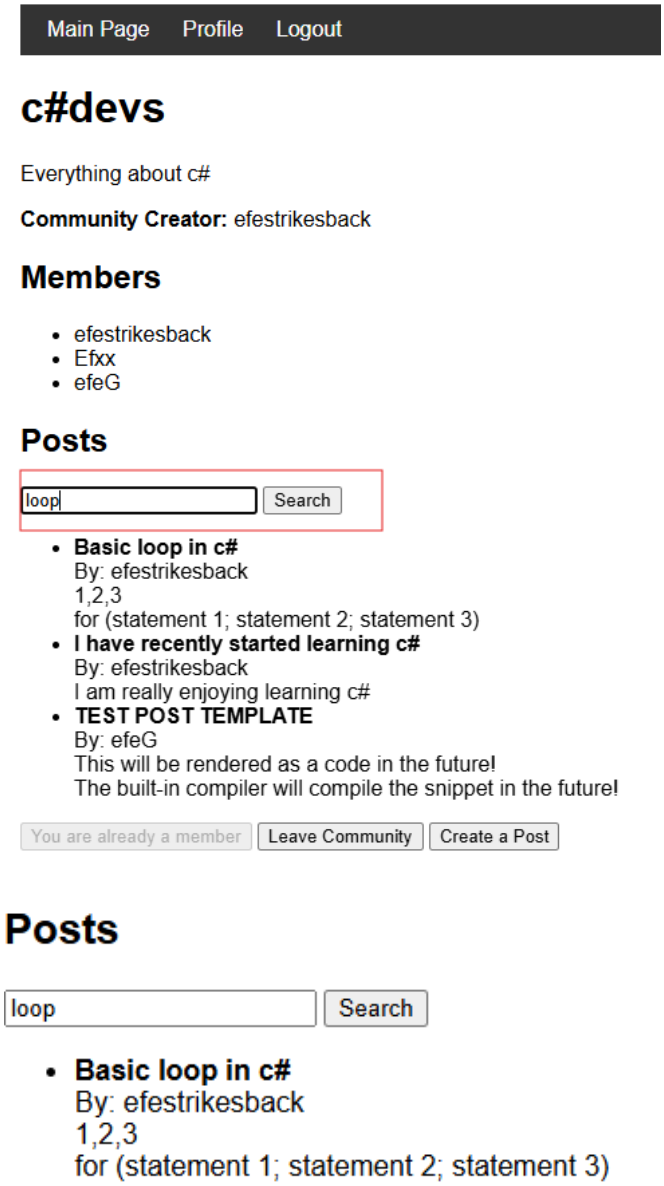

Figure 30: *Searching.*

We see the post with "loop" keyword inside.

Join, Leave, Join Again

Our test user efeG is a member of “c#devs” let’s see how the page render changes depending on user’s membership status.

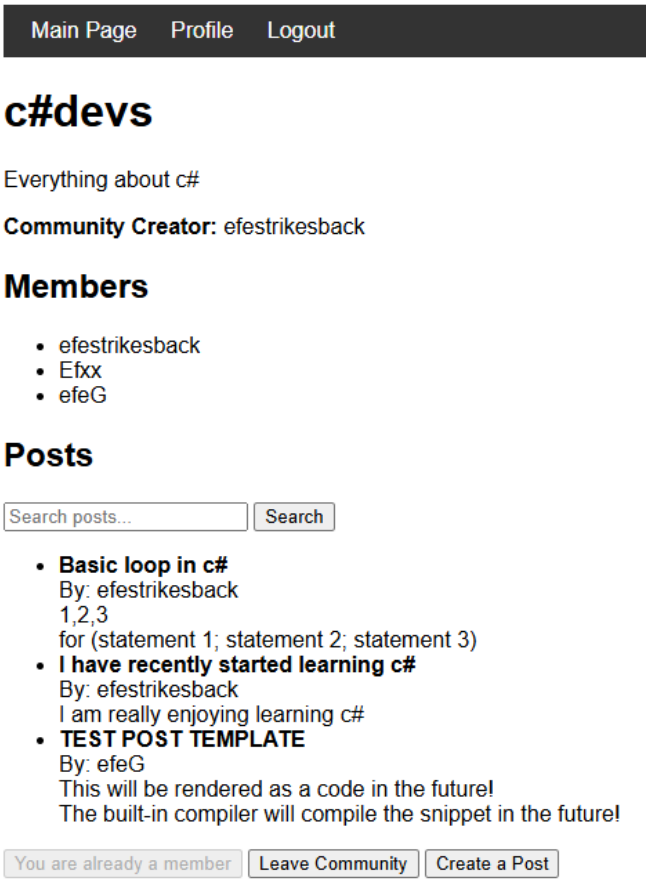

Figure 30: *Member view of commuity page.*

Let’s click **“Leave Community**”, we will see message “Successfully left the community”, then we will click **“OK”**

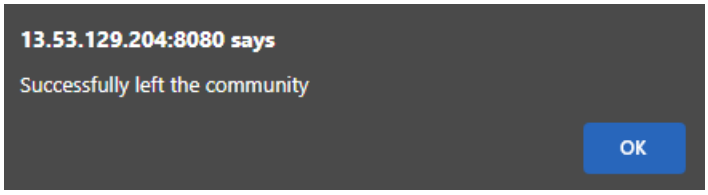

Figure 31: *Community leave message.*

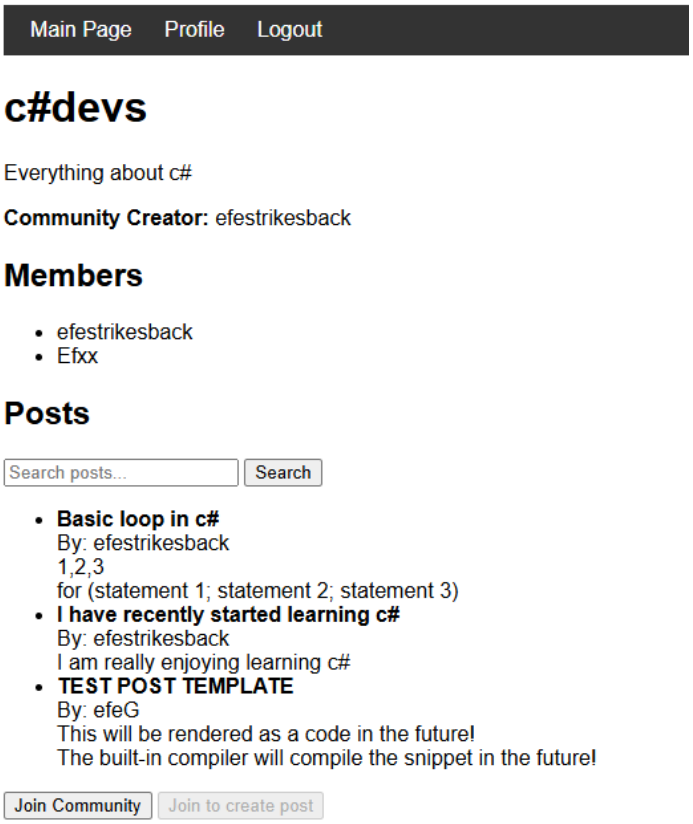

Figure 32: *Visitor view of commuity page*

We are no longer a member so we can not create a post but now we can **"Join Community"** let's join again. (After joining we need to press f5 to see new action boxes, see the bug discussed previously)

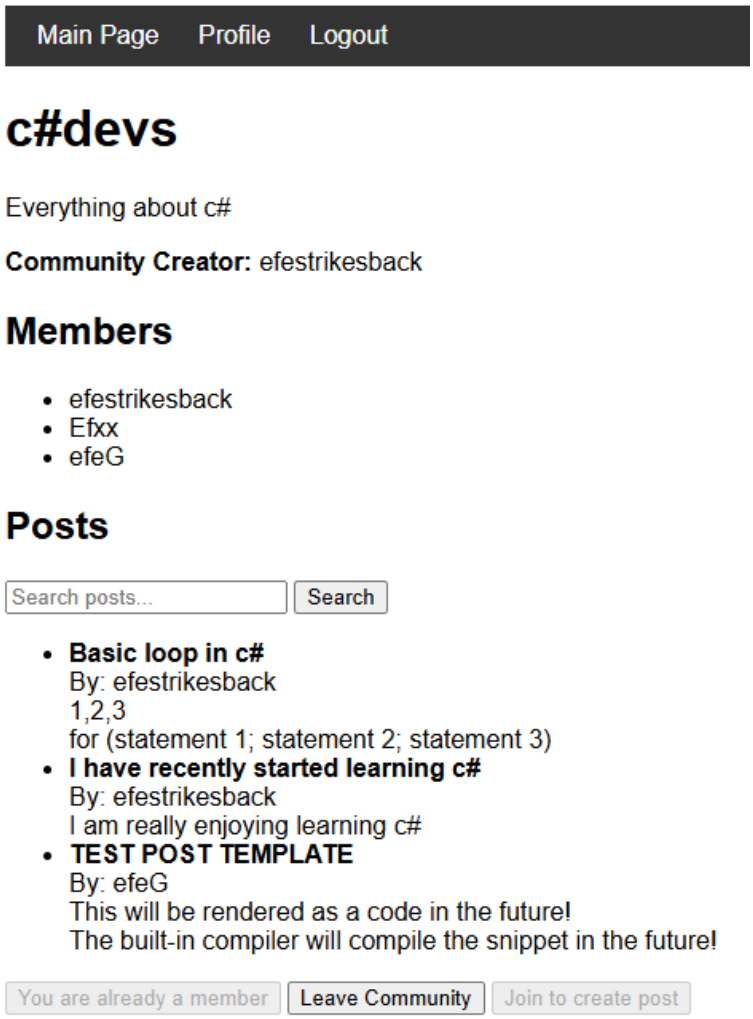

Figure 33: *Minor bug, explained previously.*

After joining again, we can see our user in the member list we refresh the page.

Figure 34: *After resfresh everthing works as intended.*

Action boxes are now at their proper status.

Logout

Currently only way to log out is to visit a community page and use **"Logout"** on navigation bar. During main page **"Logout"** on navigation bar" do not respond, as discussed previously I need to add a one-line code snippet.

Let's logout

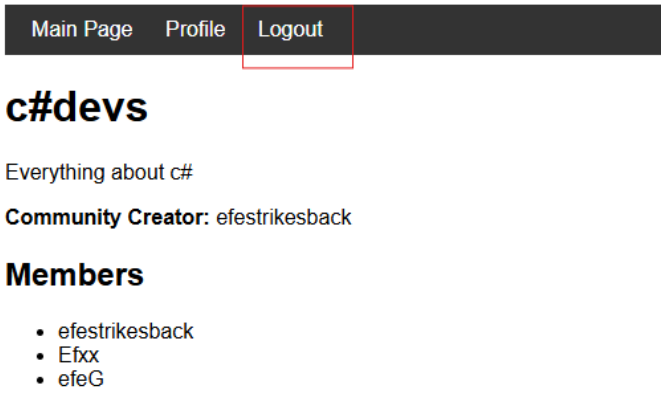

Figure 35: *Logout is functinonal when used in community page.*

When we click "Logout" we are logged out and redirected to authentication page.

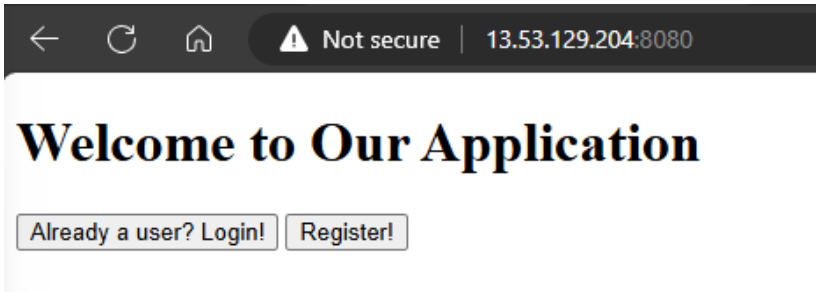

Figure 36: *Logged out user is redirected to authentication.*

8. References

"Nearly all subpages of the link provided can be regarded as reference considering my extensive use of these excellent documentations."

Springboot:

https://www.jetbrains.com/help/idea/spring-boot.html?hl=pl

https://docs.spring.io/spring-boot/docs/2.2.0.RELEASE/reference/htmlsingle/

https://www.baeldung.com/spring-boot

https://www.geeksforgeeks.org/spring-boot/

https://github.com/spring-guides/gs-spring-boot

https://github.com/springboot-samples

For frontend design:

https://www.thymeleaf.org/