Purposeful Community Platform

SWE-573 Software Development Practice

Yunus Sina GÜLŞEN

2017719177

2019f

### Contents:

1. Introduction  
   1.1. Purpose  
   1.2. Overview  
   1.3. Environmental Characteristics  
   1.3.1. Hardware  
   1.3.2. Software  
   1.3.3. Usability
2. Goals of Implementation
3. Overall Description  
   3.1. Product Functions  
   3.2. Platform Users and Characteristics  
   3.3. Product Glossary
4. Requirements  
   4.1. Functional Requirements  
   4.2. Non-Functional Requirements
5. Assumptions
6. Use Cases  
   6.1 Actors  
   6.2 Use Case Definitions  
   6.3 Use Case Diagrams
7. Class Diagrams
8. Sequence Diagrams  
   8.1 Sequence Diagram for 1.2.4 Create Community  
   8.2 Sequence Diagram for 1.2.9 Create Custom Data Type  
   8.3 Sequence Diagram for 1.2.12 Create Post
9. Mockups  
   9.1 Register New User  
   9.2 Login  
   9.3 Browse Community  
   9.4 View Generic Post  
   9.5 View Custom Data Type Post  
   9.6 New Post - Select Community  
   9.7 New Custom Data Type Post  
   9.8 New Custom Data Type  
   9.9 New Custom Data Type Field
10. Abbreviations
11. Deployment  
    11.1 Local Deployment  
    11.2 Remote Deployment (AWS)
12. Solution Components  
    12.1 Front End  
    12.2 Back End  
    12.3 Remote Deployment  
    12.4 Database
13. Licenses
14. User Manual  
    14.1 Register New User  
    14.2 Login with existing user  
    14.3 Reset password  
    14.4 Browse communities  
    14.5 Join community  
    14.6 Leave community  
    14.7 Create new community  
    14.8 Create new generic post  
    14.9 Create new custom post type  
    14.10 Create new custom post  
    14.11 Update post  
    14.12 Update community  
    14.13 Delete post  
    14.14 WikiData Tags  
    14.15 Search

## 1. Introduction

### 1.1. Purpose

This document is created for the lecture "SWE573 - Software Development Practice" @ Boğaziçi University Software Engineering Department for fall 2019 semester.

This project is about Purposeful Community Platform where registered members submit content to the platform such as links, text posts, images and etc, on moderated topics.

Topics are created by community builders who are able to define the community specific custom datatypes.

The site recommends its users (registered or anonymous) related topics based on constructed relations between these topics using tagged WikiData information.

Intended use of these topics are;

* Ability to accumulate information stored on these topics.
* Ability to request/provide structured information in a fast and supervised way.

This document also provides an overview of the Software Requirements Specification for "SWE573 Class Project - Purposeful Community Platform" with system characteristics, description, functional and non-functional requirements, behavioral description and questions. The document aims to gather and analyze the specifications of Purposeful Community Platform.

### 1.2. Overview

The document examines the implementation goals in Section 2, overall description in Section 3, functional and non-functional requirements in Section 4.

### 1.3. Environmental Characteristics

#### 1.3.1. Hardware

There are no requirements for a bare-metal server. For ease of use, cloud-based service providers will be used. (PaaS and IaS services can be used, leaving out SaaS components)

Candidates for cloud providers:

* Microsoft azure
* Amazon Web Services

#### 1.3.2. Software

Web Server: Any container supporting Java/Python  
Programming Language: Java/Python  
Database: PostgreSQL/MongoDB  
Frontend: Angular/React/Bootstrap

In client side, the web application should be reachable from common web browsers with HTML 5.0 support.

#### 1.3.3. Usability

User interfaces of the platform should be friendly enough to enable non-technical people to use the platform from targeted client environment which can be mobile or web application.

## 2. Goals of Implementation

* Develop a timeline to insure milestone dates are met and the project is completed within the established timeline.
* Manage implementation within pre-determined time.
* Follow SDLC- Software/System Development Life Cycle
* Follow Agile project management practices.

The application development phases include requirements analysis (gathering requirements), design (high level design and detailed design), implementation (development to transfer a design to product), testing (unit and integration testing) and evolution (deployment and maintenance) phases.

## 3. Overall Description

### 3.1. Product Functions

#### 3.1.1 General

* Register new user
* Login
* Browse anonymously

#### 3.1.2 Community

* Create new community
* Delete community
* Search for a community
* View recommended communities
* Join community
* Leave community
* Create Custom Data Type
* Modify own Custom Data Type
* Delete own Custom Data Type

#### 3.1.3 Post

* Read all posts inside a community
* Modify own posts inside a community
* Submit own posts inside a community
* Delete own posts inside a community
* Query all posts inside a community

#### 3.1.4 Administration

* Delete users
* Modify users

#### 3.1.5 Generic Data Type

PCP provides a primitive datatype for all communities which is called "Generic Data Type". Common fields;

* Title - used for short name, required
* Description - used for detailed description
* Semantic Tag - used for semantic search and relations, required

#### 3.1.6 Custom Data Type

Community specific datatypes which inherit fields "Title" and "Semantic Tag" from primitive structure "Generic Data Type". A community can have multiple data types. Custom Data Type posts can use the following field types;

* Image
* Audio
* Video
* Text
* Enumeration
* Email
* URL
* Datetime
* Number

### 3.2. Platform Roles and Functions

#### 3.2.1 Roles

* Registered User
* Anonymous
* Administrator

#### 3.2.2 Roles & Functions

| **Function** | **Registered User** | **Anonymous** | **Administrator** |
| --- | --- | --- | --- |
| Register |  | x |  |
| Login | x |  | x |
| Browse | x | x | x |
| Create Community | x |  | x |
| Search Community | x | x | x |
| Join Community | x |  | x |
| Leave Community | x |  | x |
| Delete Community |  |  | x |
| Browse Community | x | x | x |
| View Recommended Community | x | x | x |
| Create Custom Data Type | x |  | x |
| Modify own Custom Data Type | x |  | x |
| Delete own Custom Data Type | x |  | x |
| Read Posts | x | x | x |
| Submit Post | x |  | x |
| Modify Own Posts | x |  | x |
| Delete Own Posts | x |  | x |
| Query All Posts | x | x | x |
| Delete Users |  |  | x |
| Modify Users |  |  | x |

### 3.3. Product Glossary

* Community - A group of Data Type's with subscribed users having a particular characteristic in common
* Data Type - A structure to hold information with predefined fields
* Community Builder - Community Users who create new communities or custom data types
* Community User - Registered user who has "Registered User" role
* Anonymous User - Unregistered users who has "Anonymous" role
* Field - Representation of data in specific types
* Primitive Data Type - A basic structure to hold the predefined format of a community (Title, Description, Semantic Tag)
* Generic Data Type - Initial common datatype and official channel of a community
* Custom Data Type - A predefined structure and sub category to gather data from community users in a structured way

## 4. Requirements

### 4.1. Functional Requirements

| **Requirement** | **Short Description** | **Perspective** | **Details** |
| --- | --- | --- | --- |
| 4.1.1 | Register | User | Users shall be able to register to the platform by providing First name, Last name, username, password details |
| 4.1.2 | Login | User | Users shall be able to login to the platform by providing username and password |
| 4.1.3 | Reset Password | User | Users shall be able to reset their password using provided email address during registration |
| 4.1.4 | Create Community | User | Registered users shall be able to create new communities |
| 4.1.5 | Define Data Types | User | Registered users shall be able to create predefined structures for storing information inside communities |
| 4.1.6 | Content Evolution | User | Users who are owners of a predefined structure shall be able to modify this predefined structure by adding or removing fields to evolve the stored information |
| 4.1.7 | Generic Data Type Structure | Product | Generic Data Type shall have Title, Description and Semantic Tag fields |
| 4.1.8 | Custom Data Type Structure | Product | Custom Data Type shall inherit Title and Semantic Tag fields and have user specified number of primitive types |
| 4.1.9 | WikiData Tags | User | Users shall be able to relate posts with recommendations from WikiData services on a query topic |
| 4.1.10 | Full-Text Search | User | Users shall be able to search all information stored inside communities using full-text search |
| 4.1.11 | Filtered Search | User | Users shall be able to search information stored in predefined structures using fields from these structures as filters |
| 4.1.12 | Primitive Types | Product | Platform shall provide users primitive types {Image, Audio, Video, Text, Enumeration, Email, URL, Datetime, Number} to be used when creating predefined structures to hold community specific information |
| 4.1.13 | WikiData Recommendations | Product | Platform shall provide community recommendations to users using membership information and WikiData tags |
| 4.1.14 | Browse Communities | User | Anonymous or registered users shall be able to view communities and read posts inside communities |
| 4.1.15 | Subscription | User | Registered users shall be able to subscribe to communities. Subscribed users shall be able to unsubscribe from communities |
| 4.1.16 | Post | User | Registered users shall be able to post information to communities using generic or predefined structures |
| 4.1.17 | Enumeration Field | User | When creating predefined structures to hold information inside a community, user shall be able to use a special field which allows them to select values from predefined options |
| 4.1.18 | Upload Rich Content | User | Users shall be able to store rich content (Video, audio or image) inside communities |
| 4.1.19 | Update Community | User | Community owner users shall be able to update community information |
| 4.1.20 | Update Post | User | Users shall be able to update own posts |
| 4.1.21 | Delete Post | User | Users shall be able to delete own posts |
| 4.1.22 | Required Field | User | Users shall be able to define required fields when creating predefined structures to store information inside a community |

### 4.2. Non-Functional Requirements

| **Requirement No** | **Short Description** | **Type** | **Details** |
| --- | --- | --- | --- |
| 4.2.1 | Database | Product | Database used for the platform shall be MongoDB or PostgreSQL |
| 4.2.2 | Programming Language | Product | Programming language used to implement the platform shall be Java or Python |
| 4.2.3 | Front-end | Product | Platform front-end shall be mobile or web application |

## 5. Assumptions

| **Assumption No** | **Short Description** | **Details** |
| --- | --- | --- |
| 5.1 | Orphaned Values | When builders alter Custom Data Type definition, the platform shall behave optimistic about orphaned information |

## 6. Use Cases

### 6.1 Actors

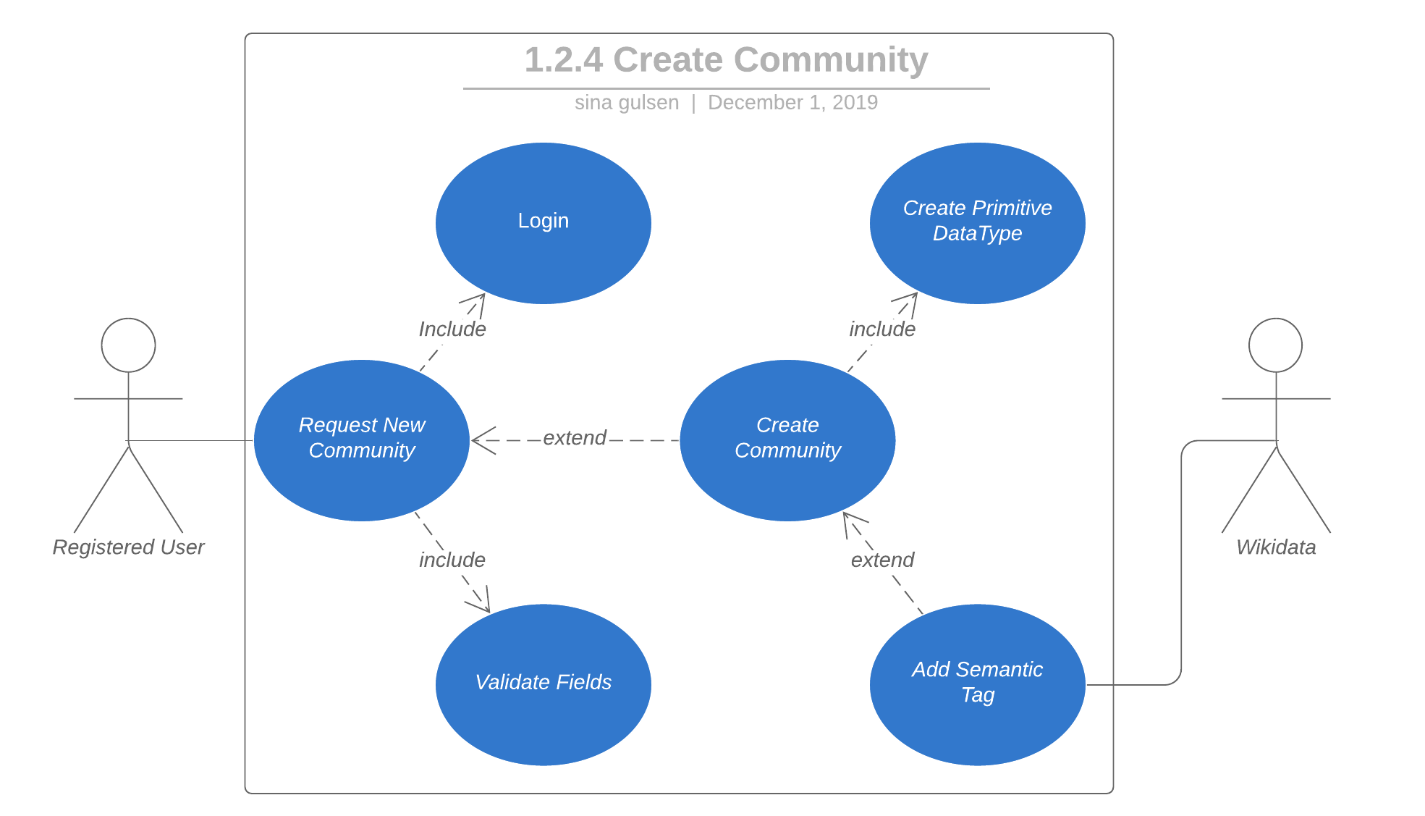
| **Id** | **Name** |
| --- | --- |
| 1.1.1 | Administrator |
| 1.1.2 | Anonymous |
| 1.1.3 | Registered User |
| 1.1.4 | WikiData |

### 6.2 Use Case Definitions

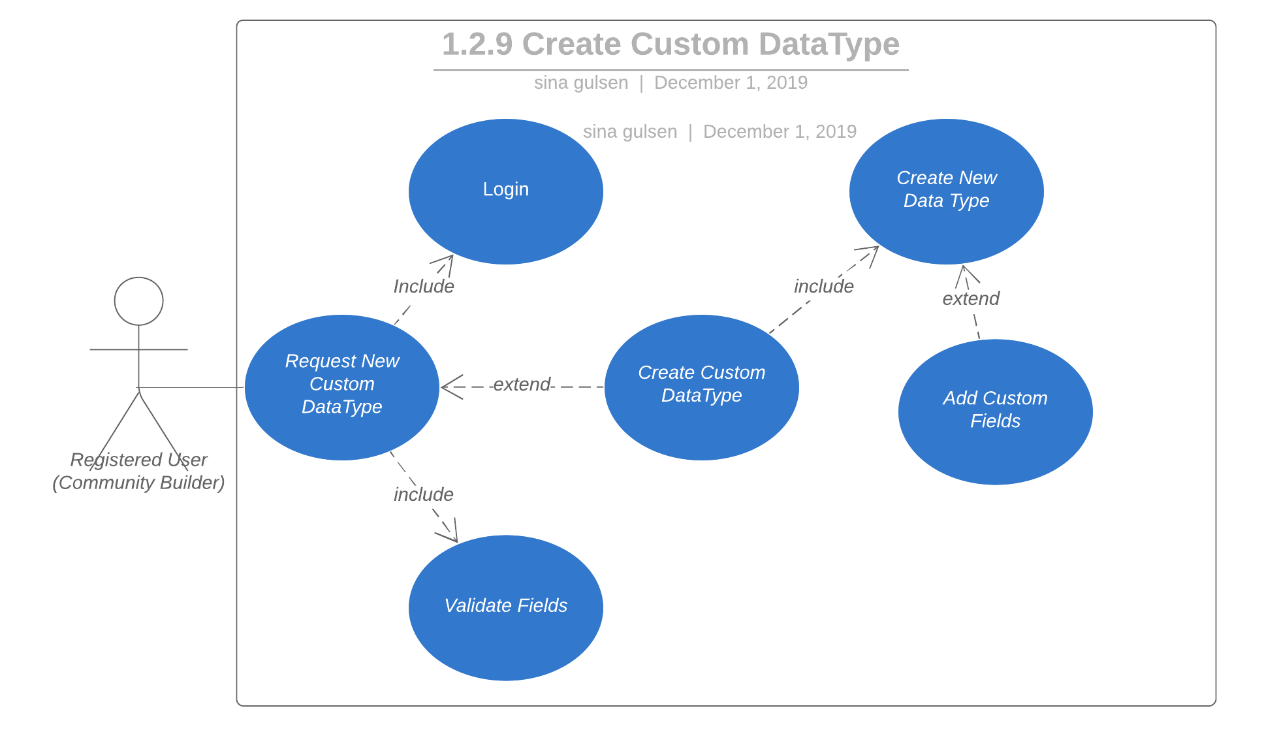
| **Id** | **Name** |
| --- | --- |
| 1.2.1 | Create User |
| 1.2.3 | Login |
| 1.2.4 | Create Community |
| 1.2.5 | Browse Community |
| 1.2.6 | Join Community |
| 1.2.7 | Leave Community |
| 1.2.9 | Create Custom Data Type |
| 1.2.10 | Modify Custom Data Type |
| 1.2.12 | Create Post |
| 1.2.13 | Modify Post |
| 1.2.14 | Browse Posts |
| 1.2.15 | Delete Posts |
| 1.2.16 | Delete Users |
| 1.2.17 | Modify Users |
| 1.2.18 | Get Semantic Tags |
| 1.2.19 | Get Recommended Communities |
| 1.2.20 | Generic Search |
| 1.2.21 | Custom Search |

### 6.3 Use Case Diagrams

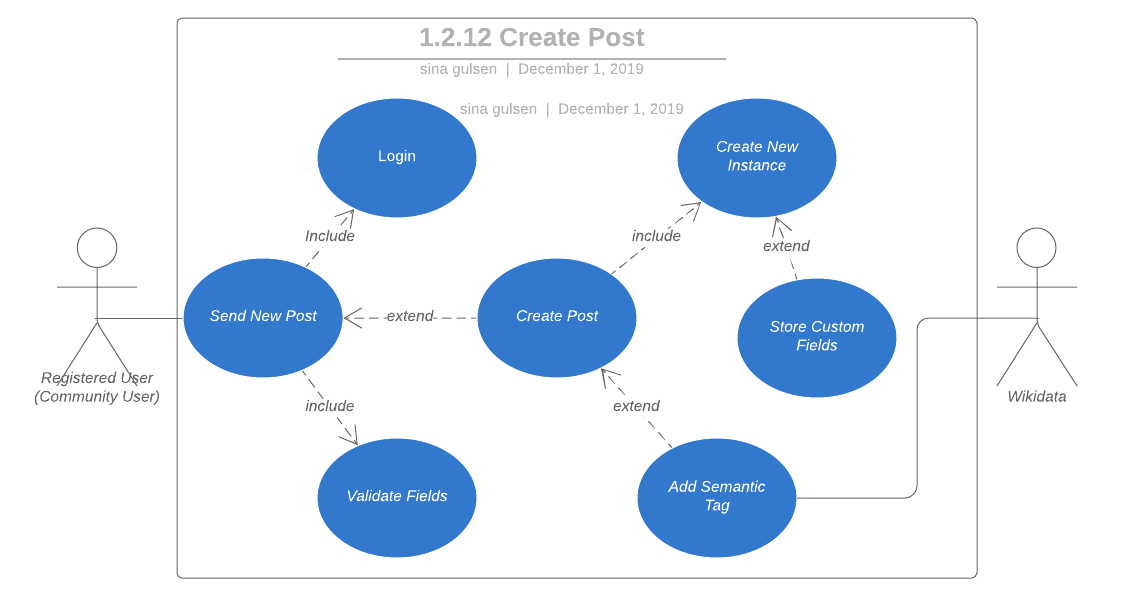
#### 6.3.1 Create Community (Use case 1.2.4)



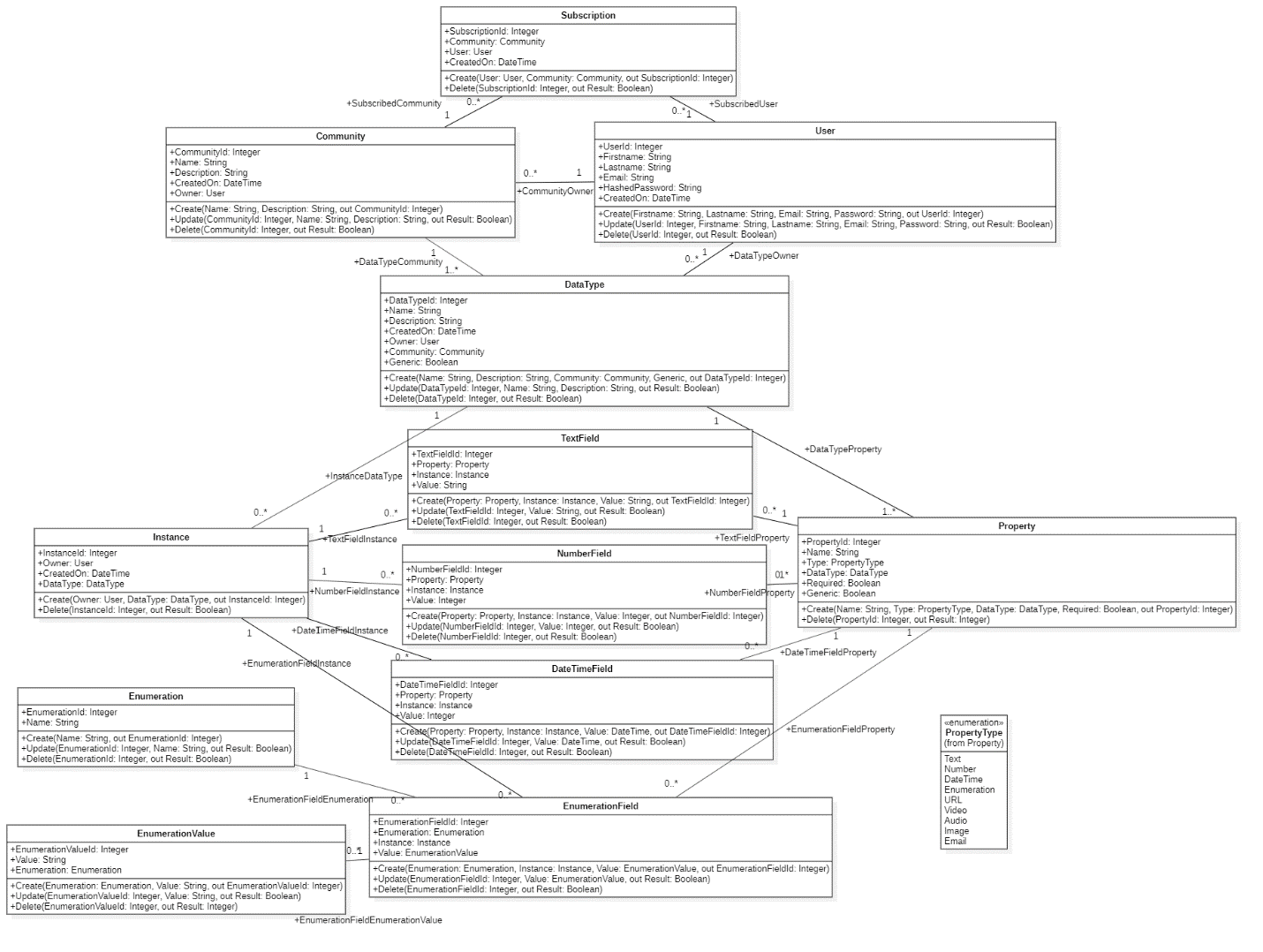
#### 6.3.2 Create Custom Data Type (Use case 1.2.9)



#### 6.3.3 Create Post (Use case 1.2.12)

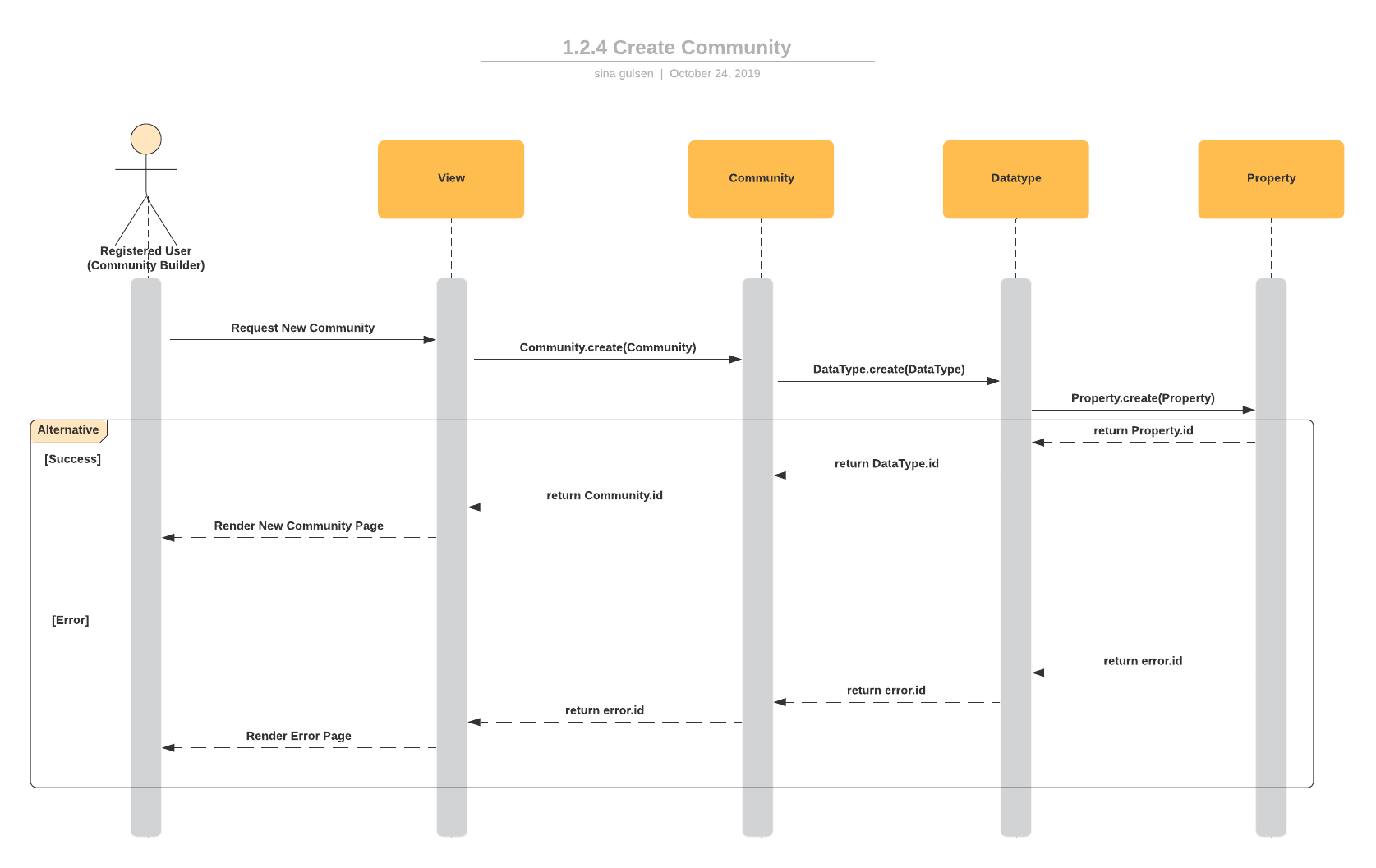


## 7. Class Diagrams

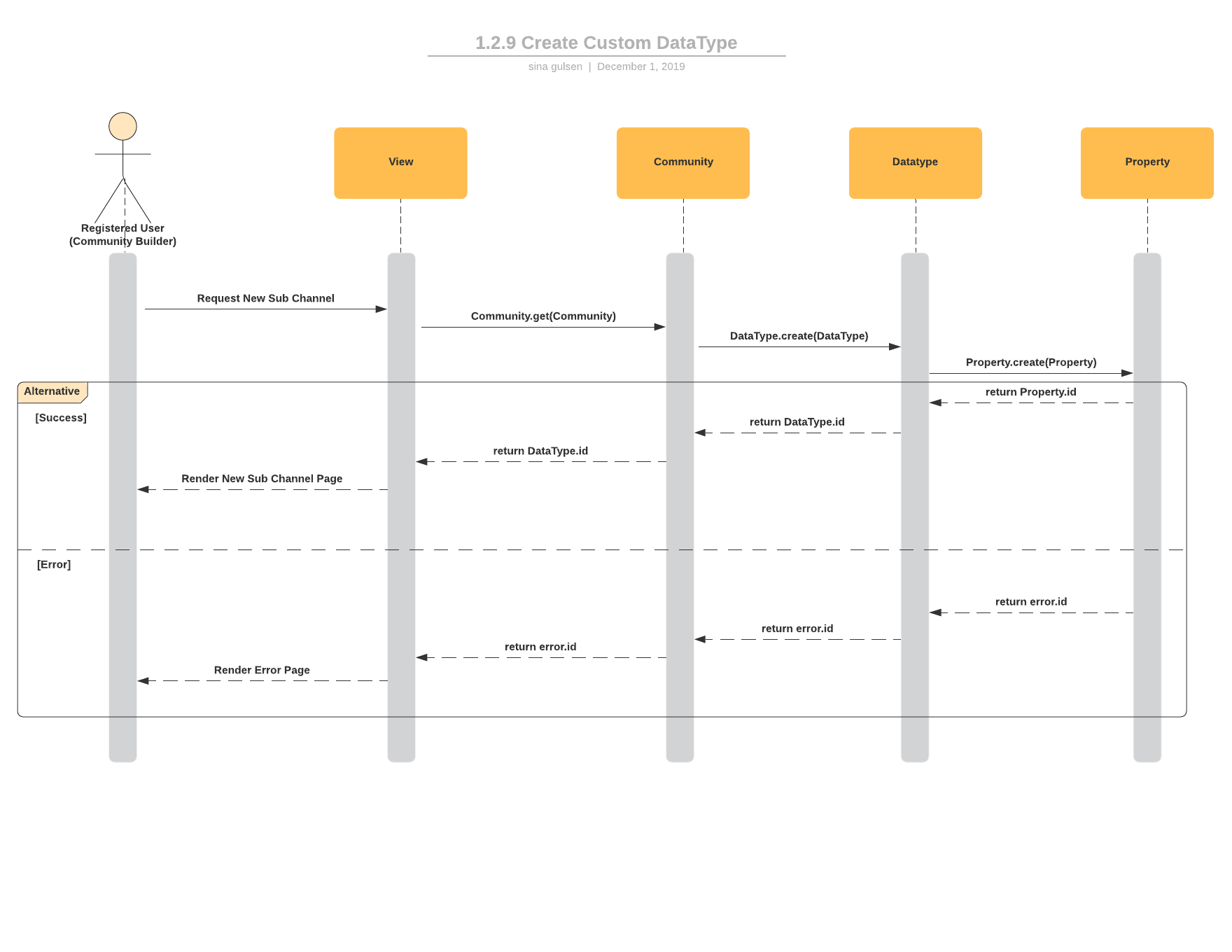
****

## 8. Sequence Diagrams

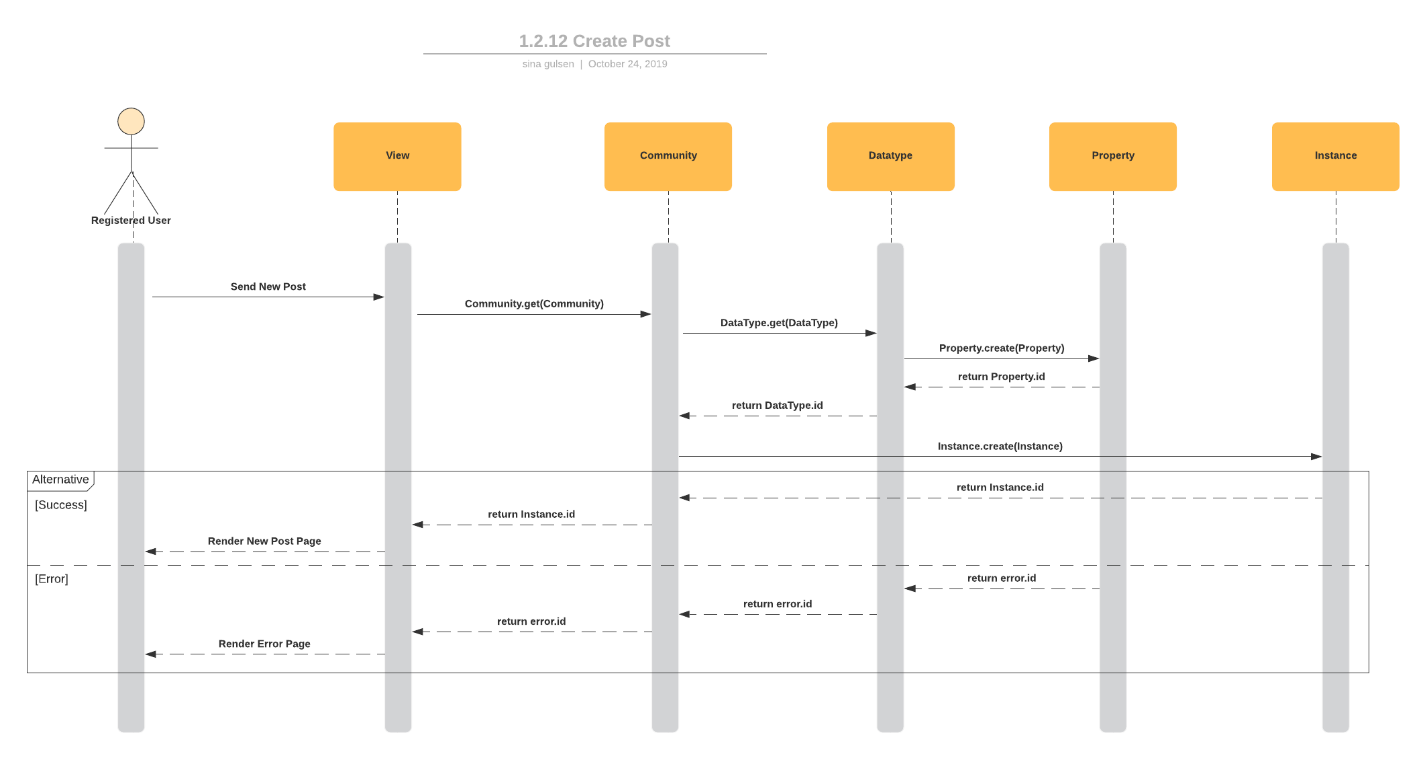
### 8.1 Sequence Diagram for 1.2.4 Create Community



### 8.2 Sequence Diagram for 1.2.9 Create Custom Data Type

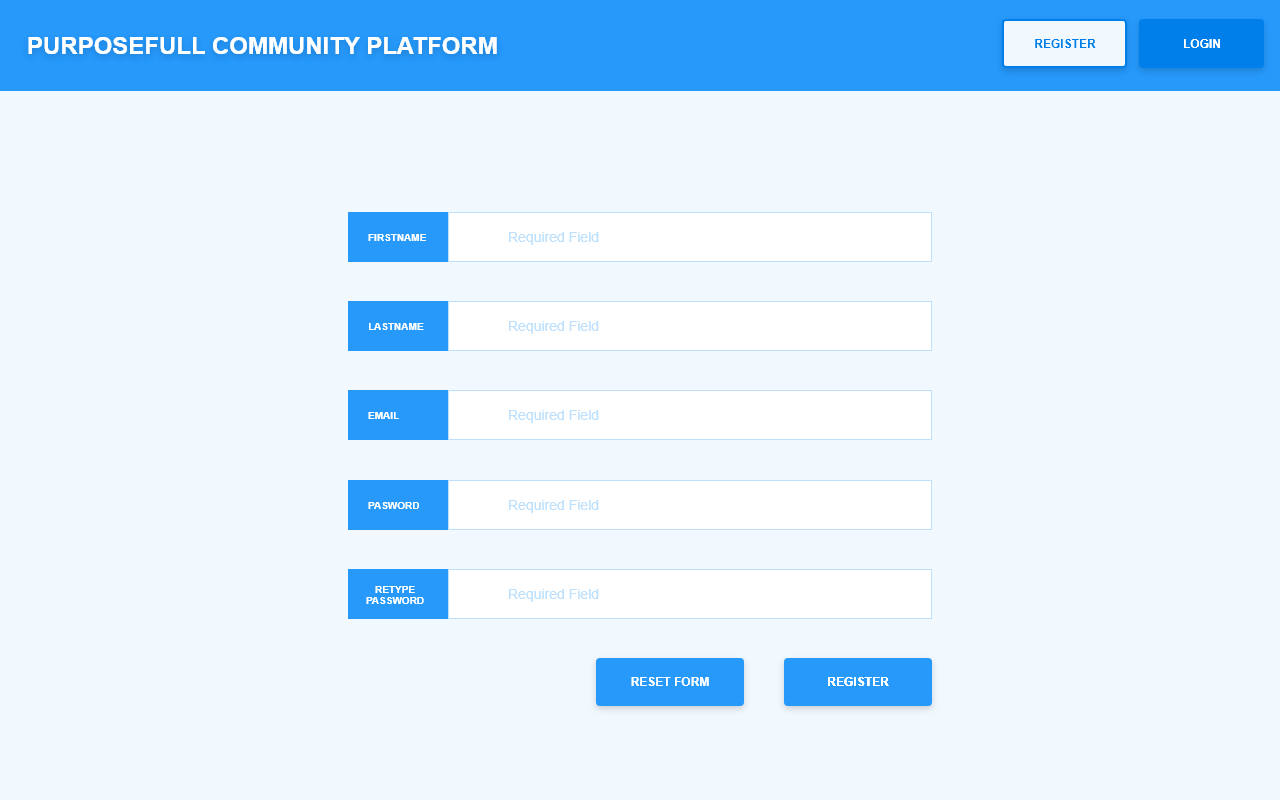


### 8.3 Sequence Diagram for 1.2.12 Create Post

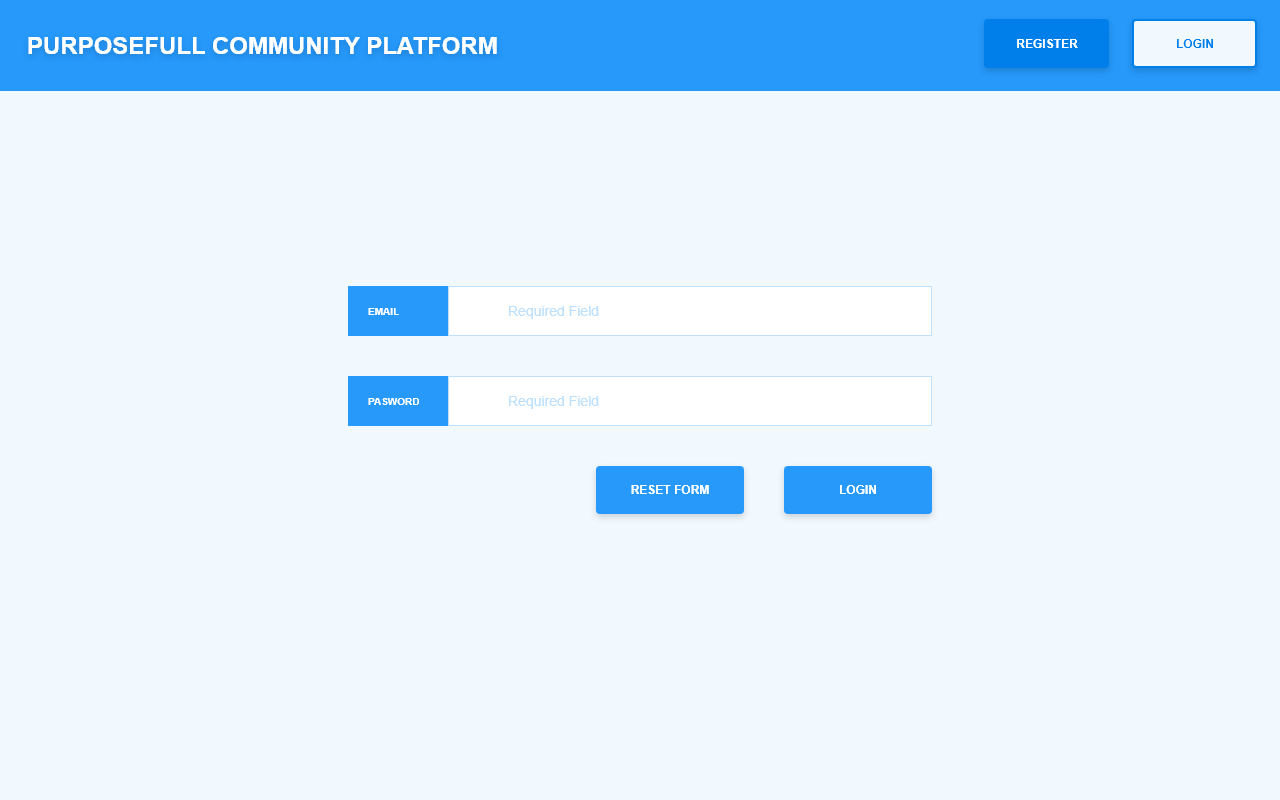


## 9. Mockups

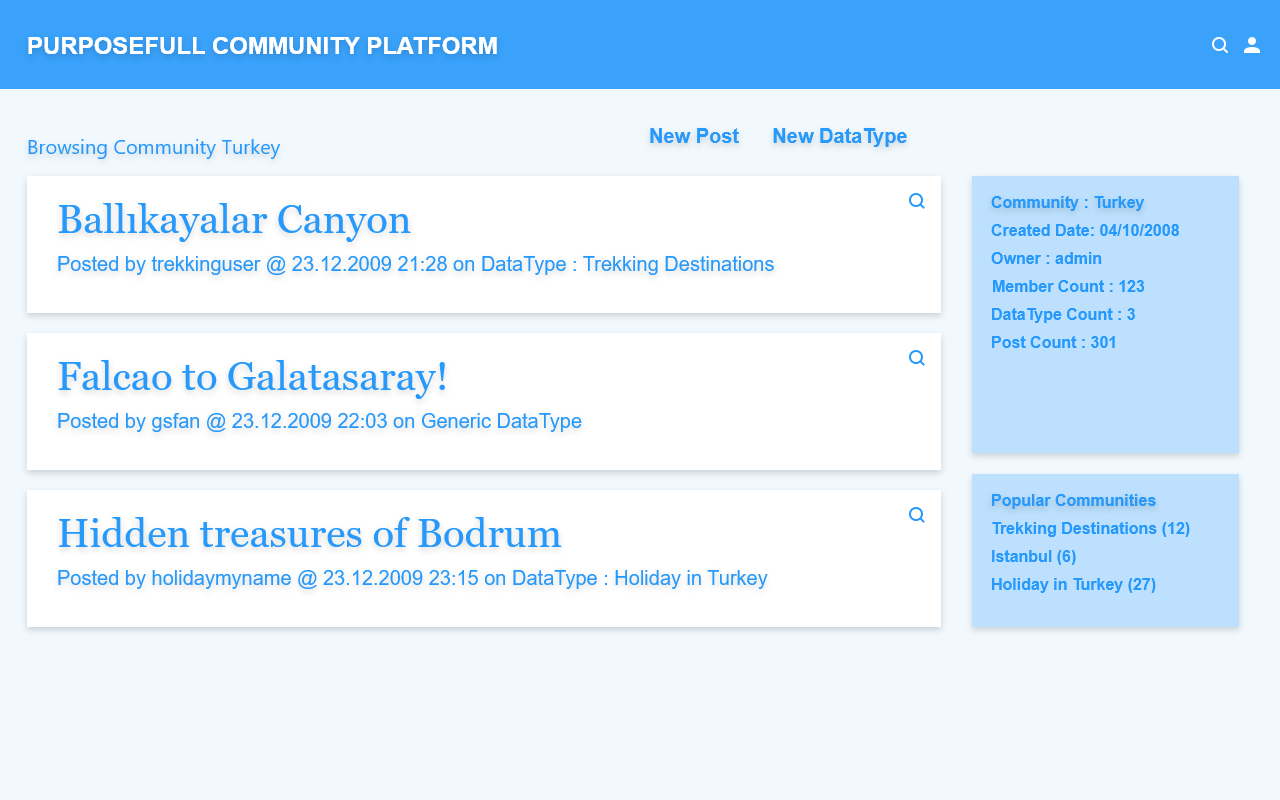
### 9.1 Register New User



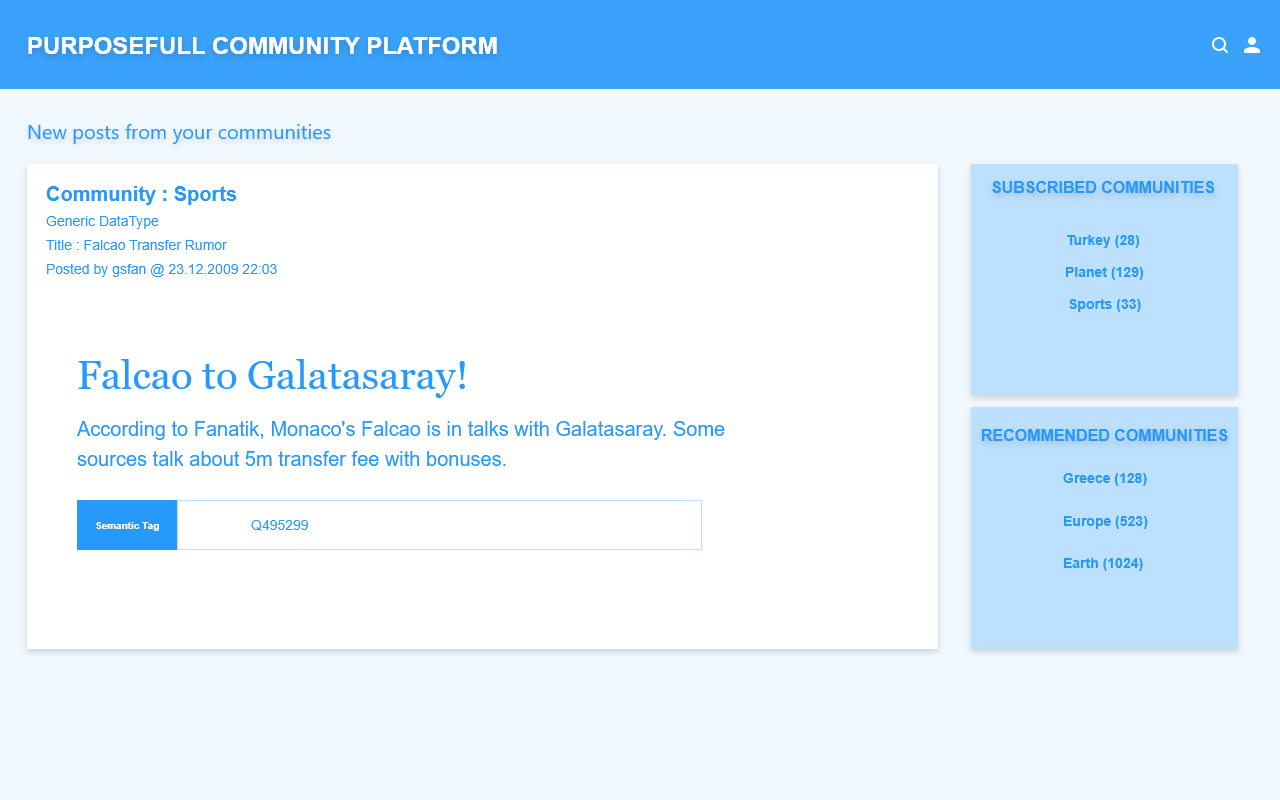
### 9.2 Login



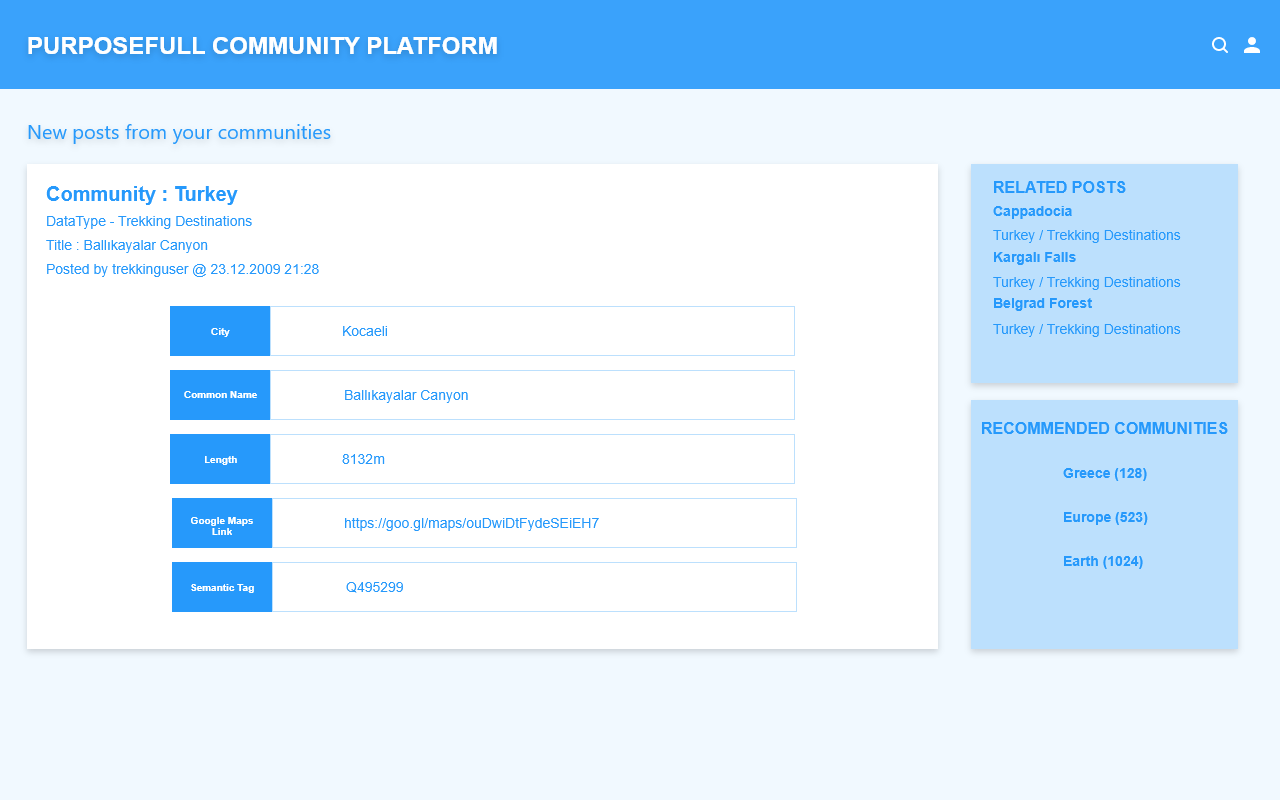
### 9.3 Browse Community



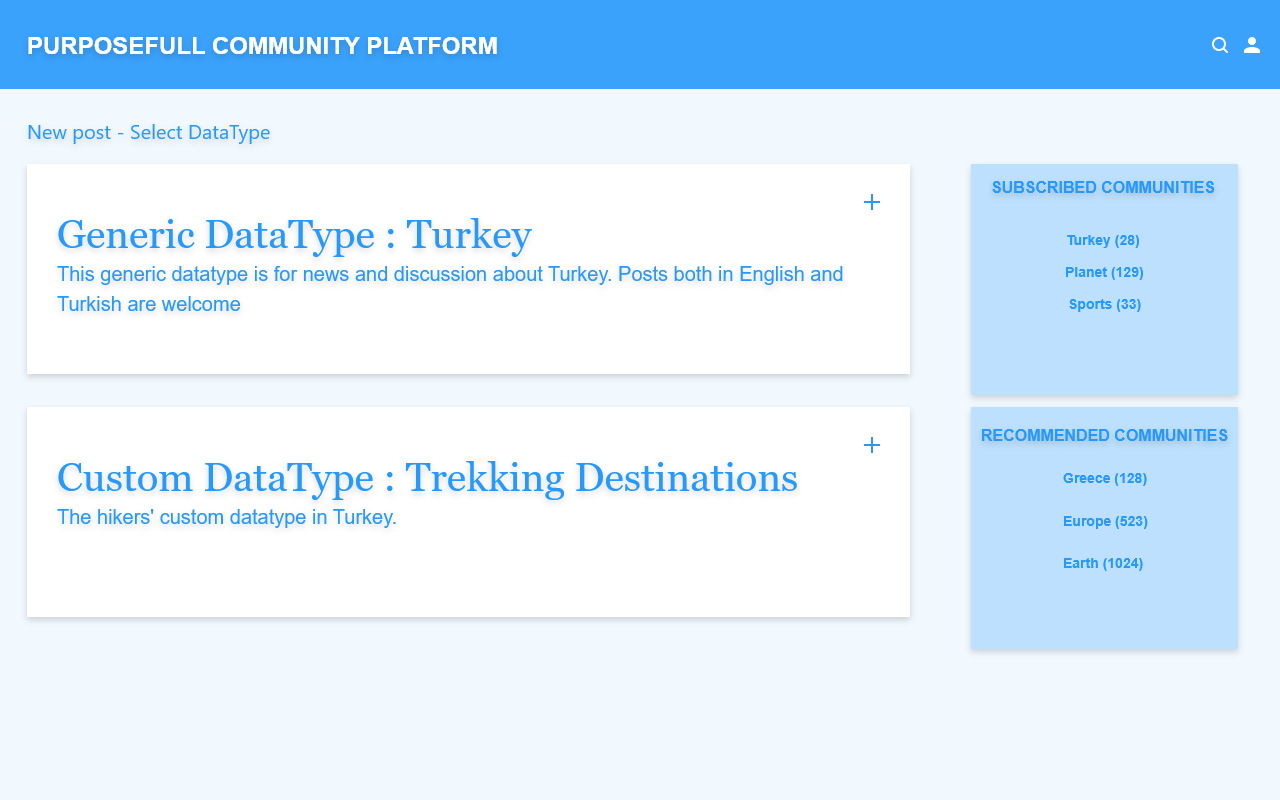
### 9.4 View Generic Post



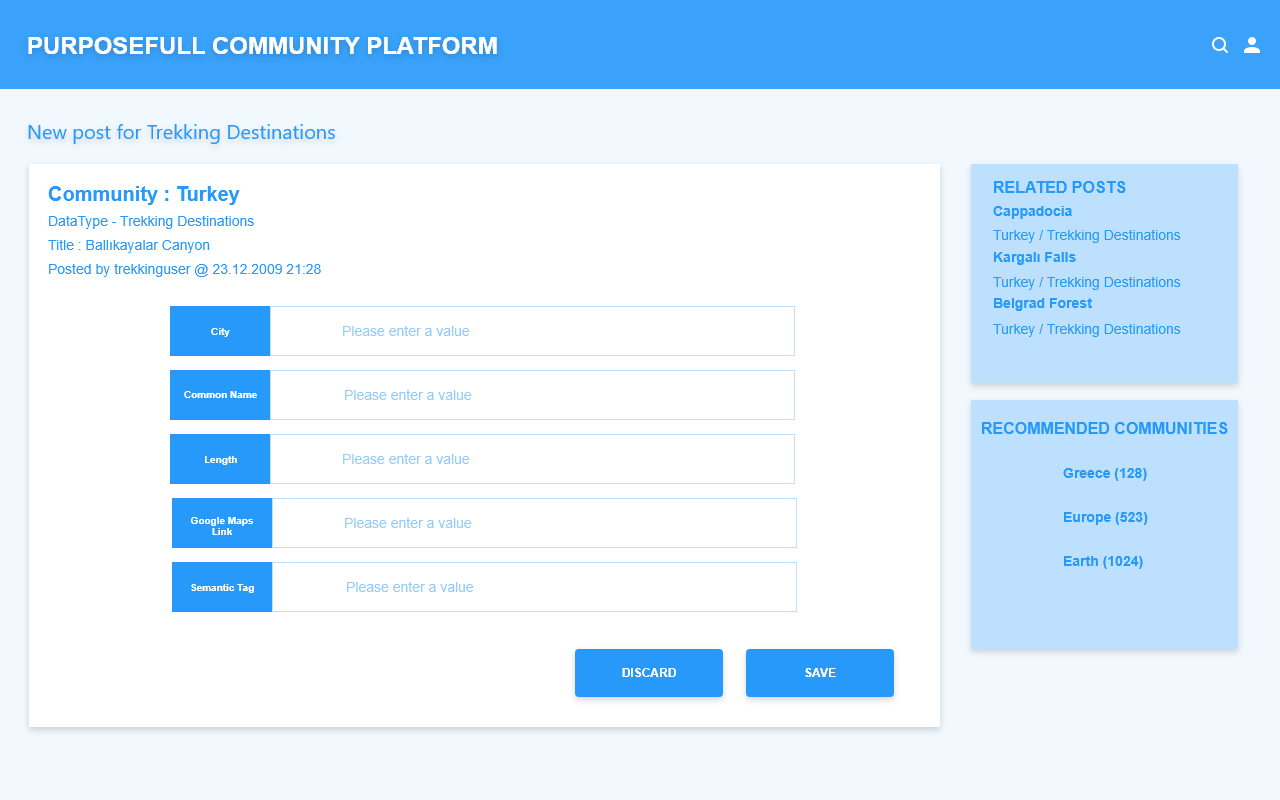
### 9.5 View Custom Data Type Post



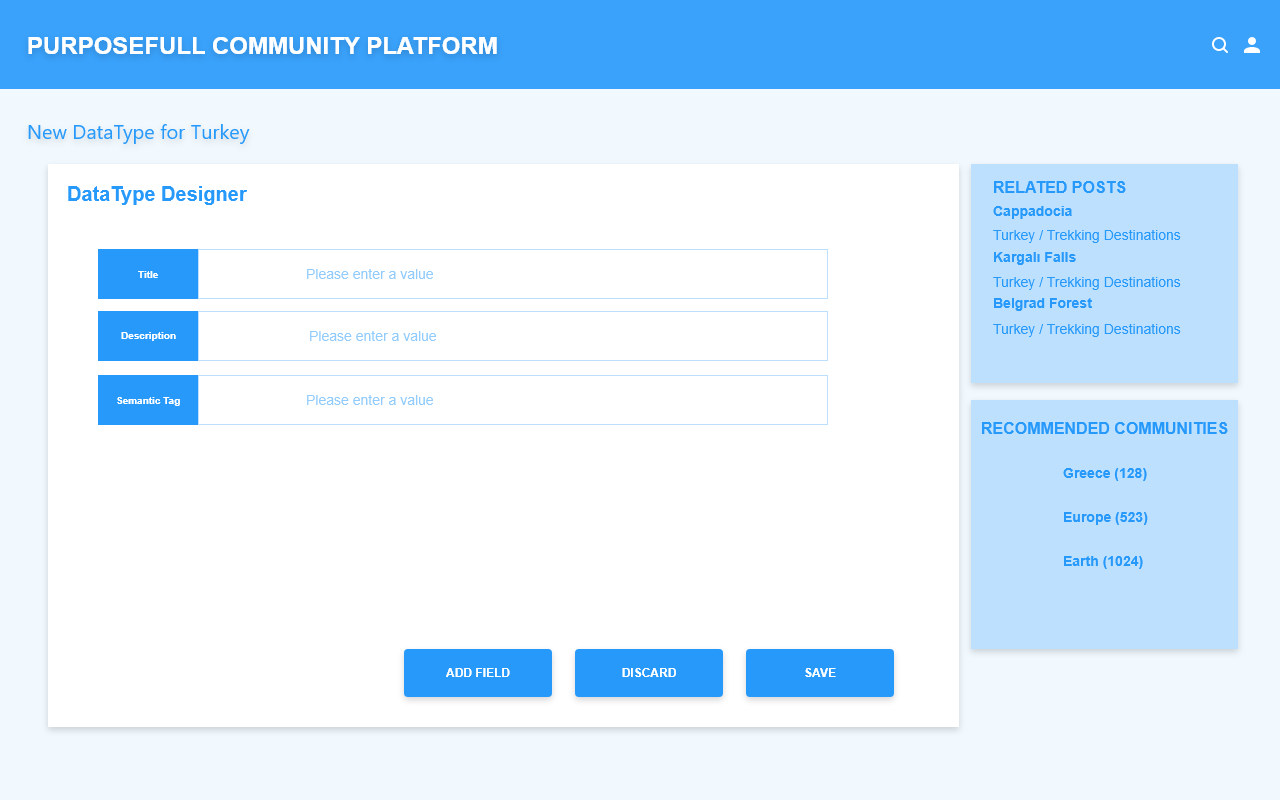
### 9.6 New Post - Select Community



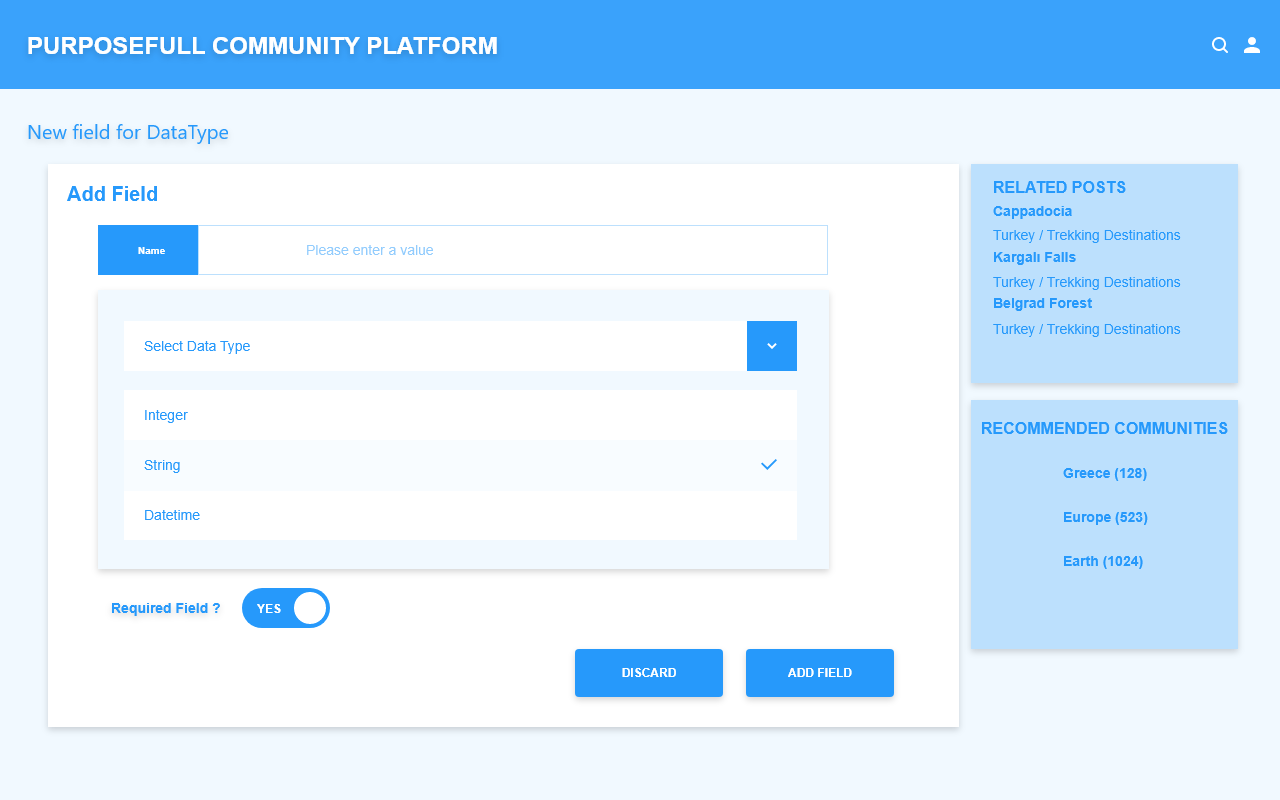
### 9.7 New Custom Data Type Post



### 9.8 New Custom Data Type



### 9.9 New Custom Data Type Field



## 10. Abbreviations

List of commonly used abbreviations used in this documentation.

* IAS - Infrastructure as a service
* PaaS - Platform as a service
* SaaS - Software as a service
* SDLC - Software development life cycle
* PDT - Primitive data type
* CSDT - Community specific data type
* PCP - Purposeful Community Platform

## 11. Deployment

## 11.1 Local Deployment

## Requirements

* Python 3.6
* pip
* virtualenv
* awsebcli
* PostgreSQL 11

## Steps

1. Set Up a Python Virtual Environment and Install Django

~$ virtualenv ~/pcp

1. Activate the virtual environment

~$ source ~/pcp/bin/activate

(pcp) ~$

1. Use pip to install Django

(pcp) ~$ pip install django==2.1.1

1. To verify that Django is installed, enter the following

(pcp) ~$ pip freeze

Django==2.1.1

...

1. Clone repository

git clone https://github.com/sinag/SWE573-PurposefulCommunityPlatform

1. Run PCP site locally with manage.py runserver.

(pcp) ~$ cd SWE573-PurposefulCommunityPlatform

(pcp) ~/SWE573-PurposefulCommunityPlatform$ python manage.py runserver

1. In a web browser, open <http://127.0.0.1/> to view the site

## 11.2 Remote Deployment (AWS)

## Requirements

* virtualenv
* awsebcli

## Steps

1. Activate your virtual environment.

~/SWE573-PurposefulCommunityPlatform$ source ~/pcp/bin/activate

1. Run pip freeze, and then save the output to a file named requirements.txt

(pcp) ~/SWE573-PurposefulCommunityPlatform$ pip freeze > requirements.txt

1. Initialize your EB CLI repository with the eb init command

~/pcp$ eb init -p python-3.6 pcp

Application pcp has been created.

1. Create an environment and deploy your application to it with eb create

~/pcp$ eb create pcp-env

## 12. Solution Components

## 12.1 Front End

* Django Forms 2.2.1
* Django Templates 2.2.1
* jQuery 3.3.1
* Bootstrap 4.3.1
* Font Awesome 5.5.0
* Select2 4.0.12

## 12.2 Back End

* Django Models 2.2.1
* Django Views 2.2.1

## 12.3 Remote Deployment

* Amazon EC2 Instance
* Amazon Elastic Beanstalk

## 12.4 Database

* Amazon RDS for PostgreSQL 11

## 13. Licenses

* Purposeful Community Platform License

https://github.com/sinag/SWE573-PurposefulCommunityPlatform/blob/master/LICENSE

* PostgreSQL License

https://opensource.org/licenses/postgresql

* Django License

https://github.com/django/django/blob/master/LICENSE

* jQuery License

https://jquery.org/license

* Bootstrap License

https://getbootstrap.com/docs/4.0/about/license

* Select2 License

https://github.com/select2/select2/blob/develop/LICENSE.md

* Font Awesome License

<https://fontawesome.com/license/free>

## 14. User Manual

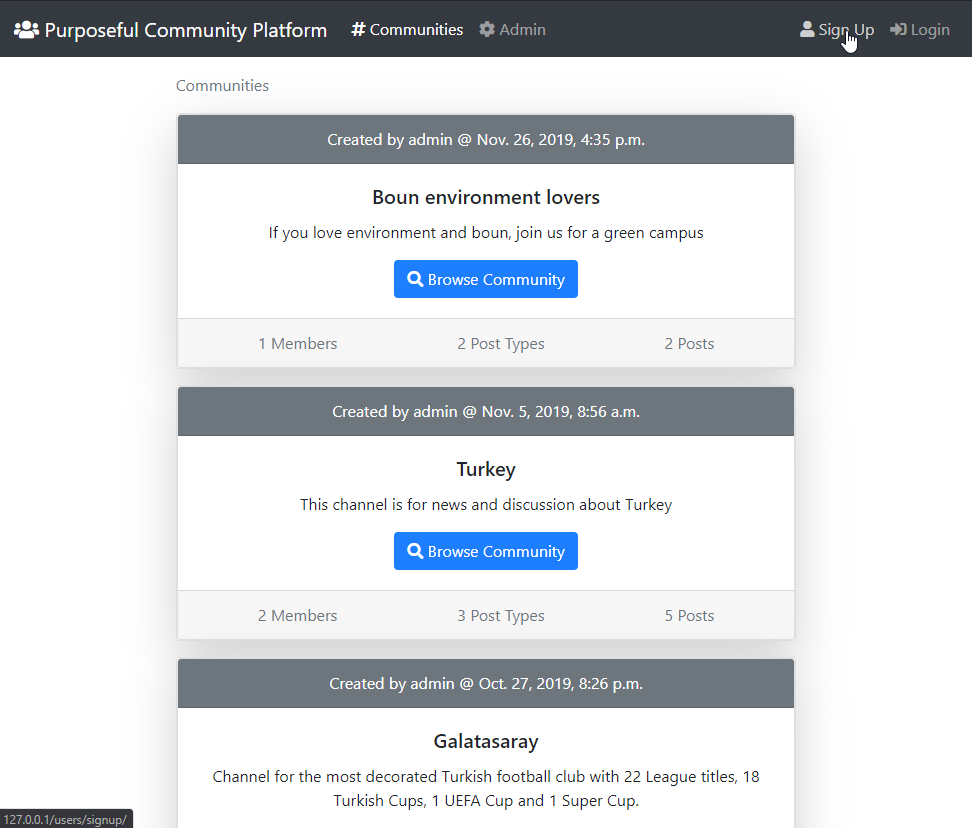
## 14.1 Register new user

### Related functional requirements

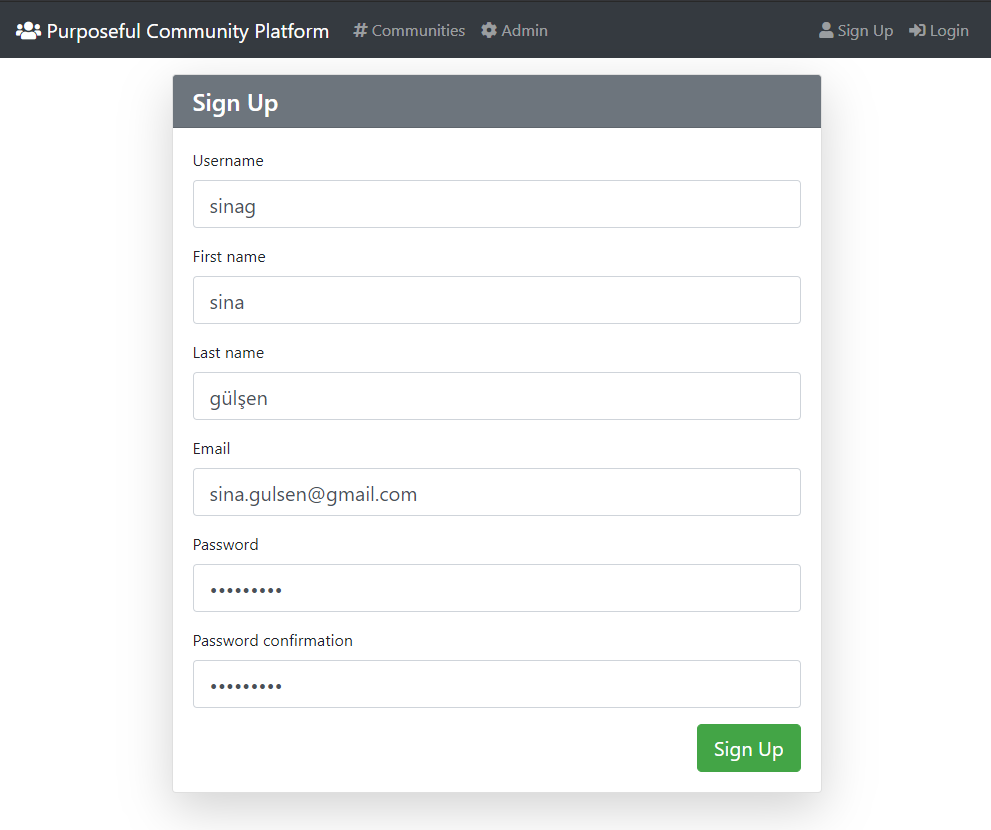
* 4.1.1 - Register

### Steps

#### 1) As an anonymous user, click "Sign-Up" button on homepage



#### 2) Fill the form and click "Sign-Up"



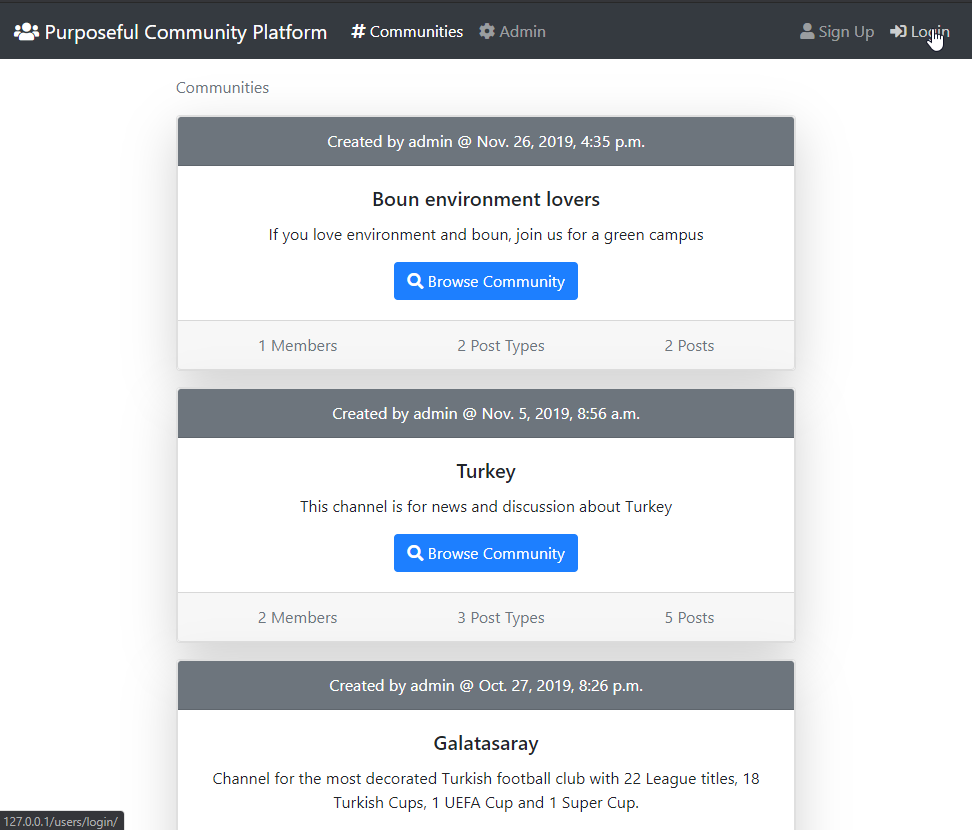
## 14.2 Login with existing user

### Related functional requirements

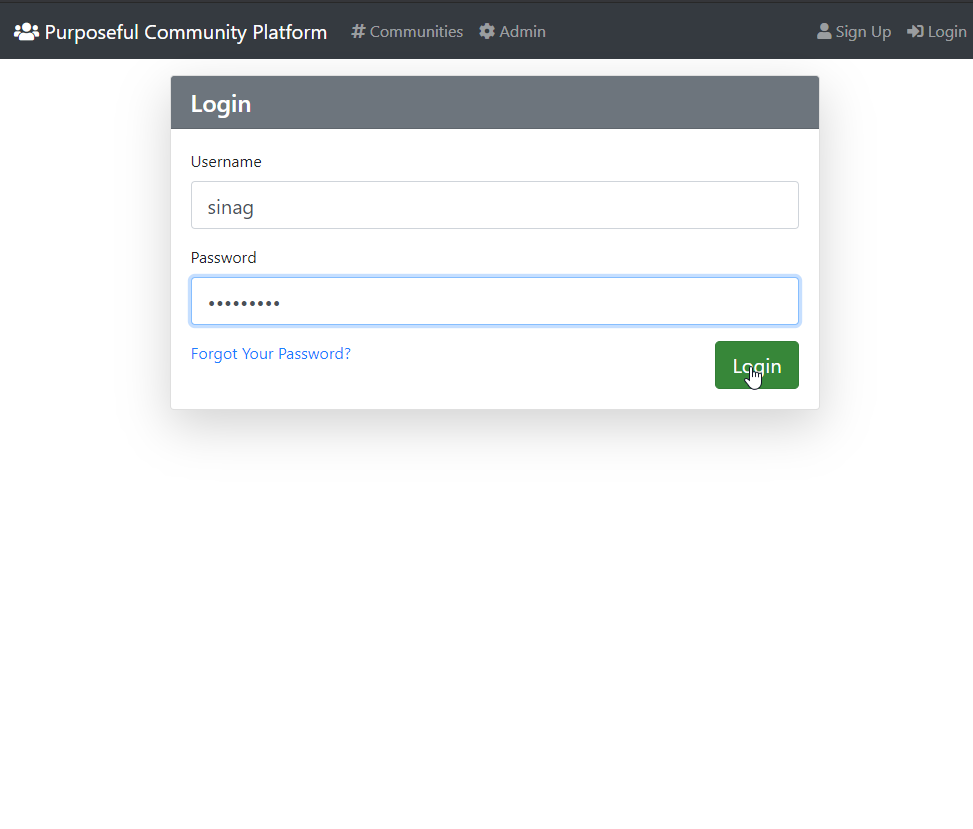
* 4.1.2 - Login

### Steps

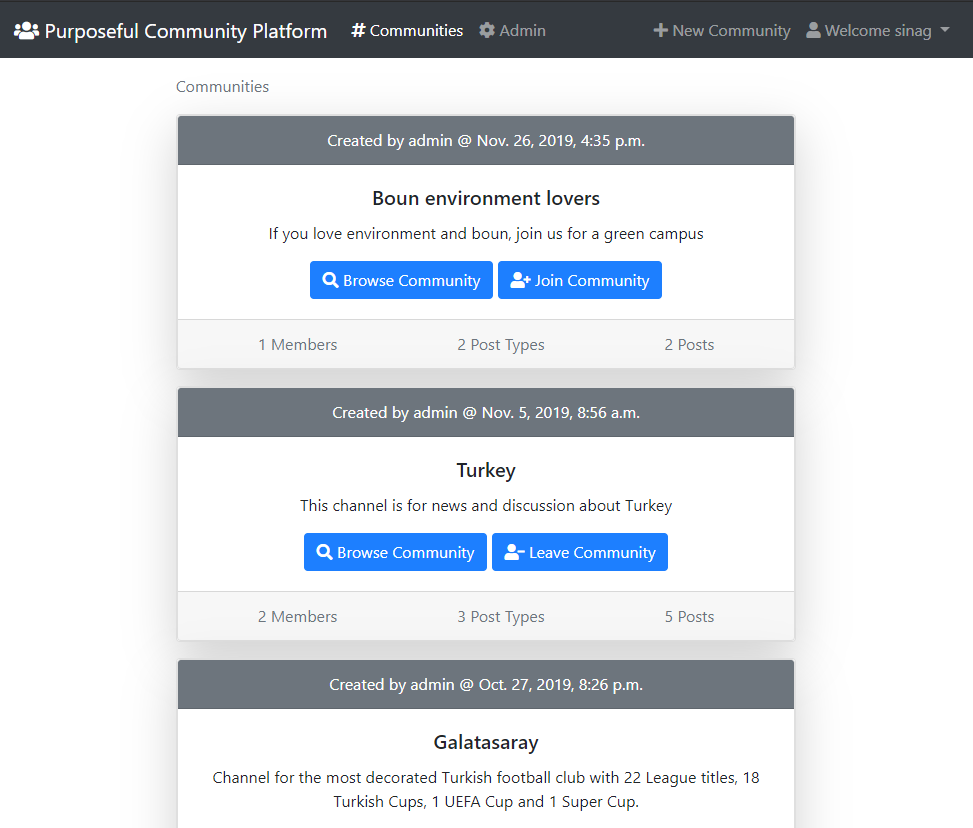
#### 1) As an anonymous user, click "Login" button on homepage



#### 2) Provide username and password and click "Login"



#### 3) User is redirected to home page and logged-in user menu is available on top right portion of the page inside navigation menu



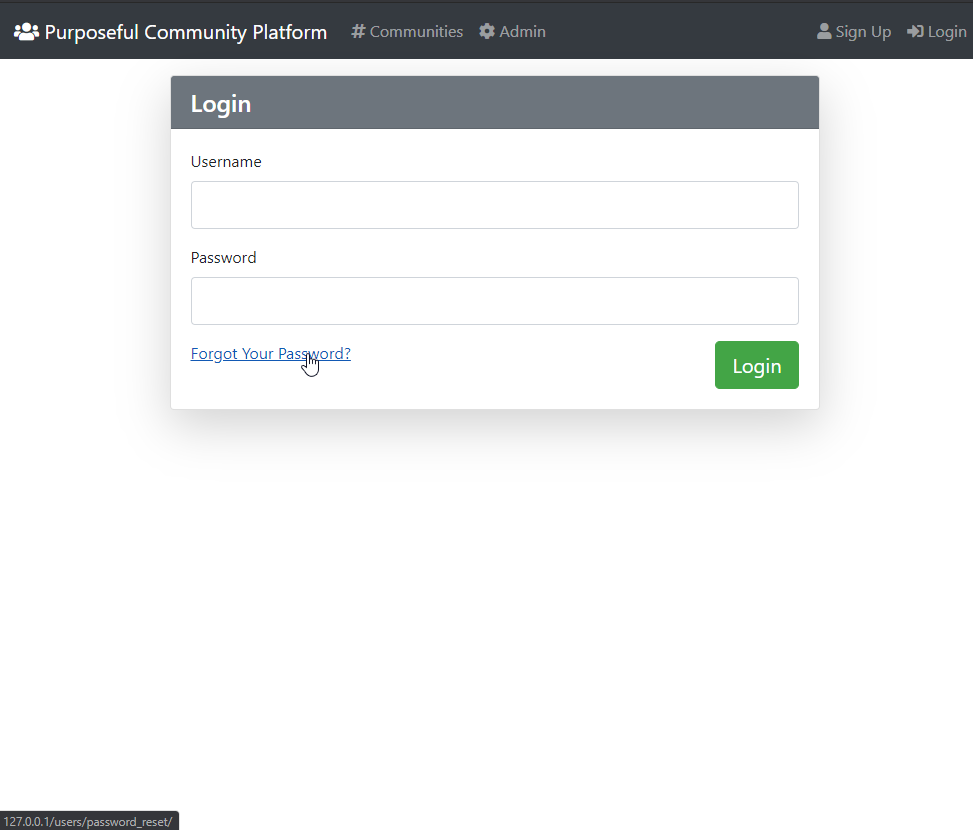
## 14.3 Reset password

### Related functional requirements

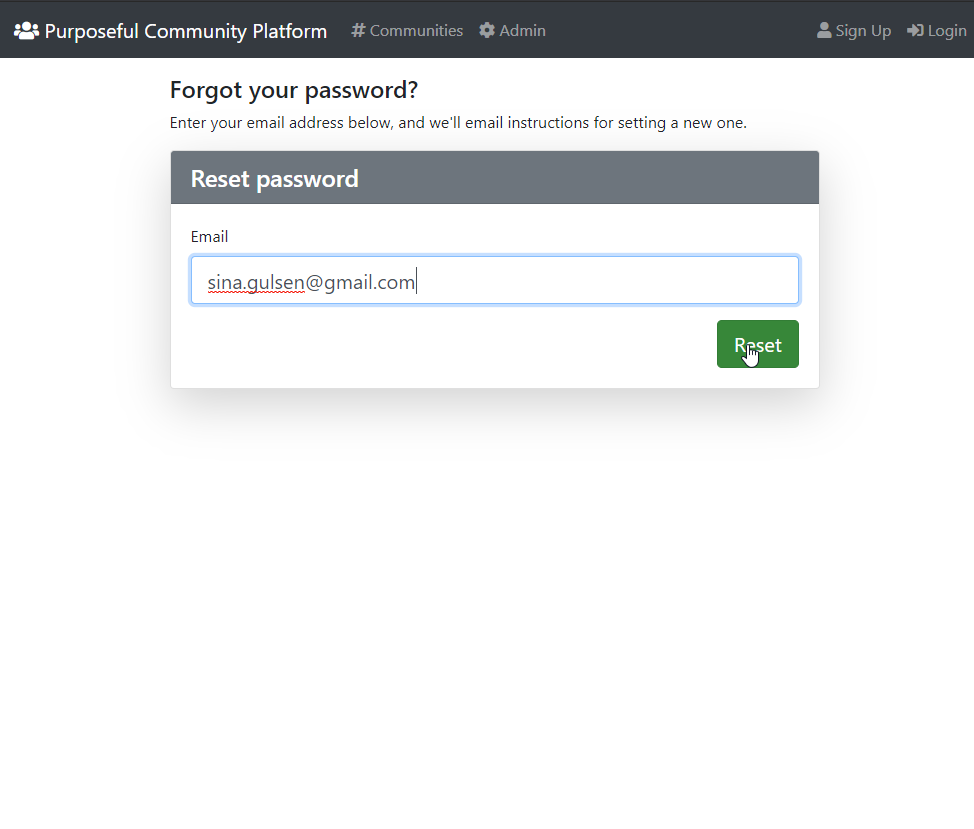
* 4.1.3 - Reset password

### Steps

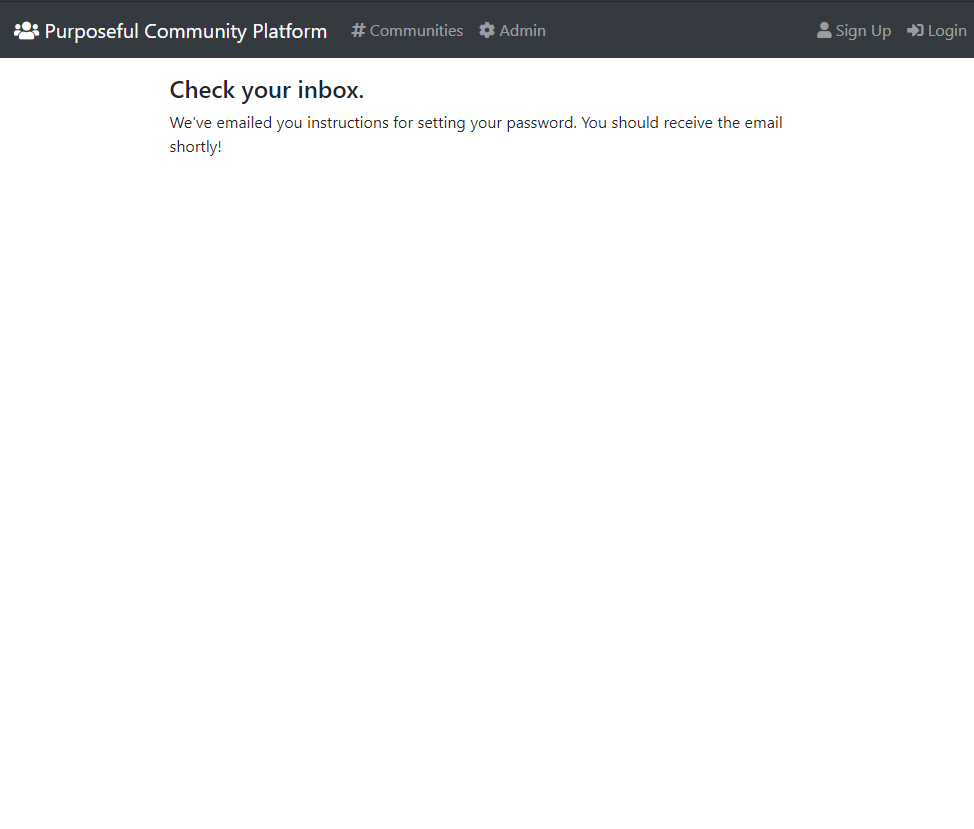
#### 1) As an anonymous user, click "Forgot your password?" link on login page



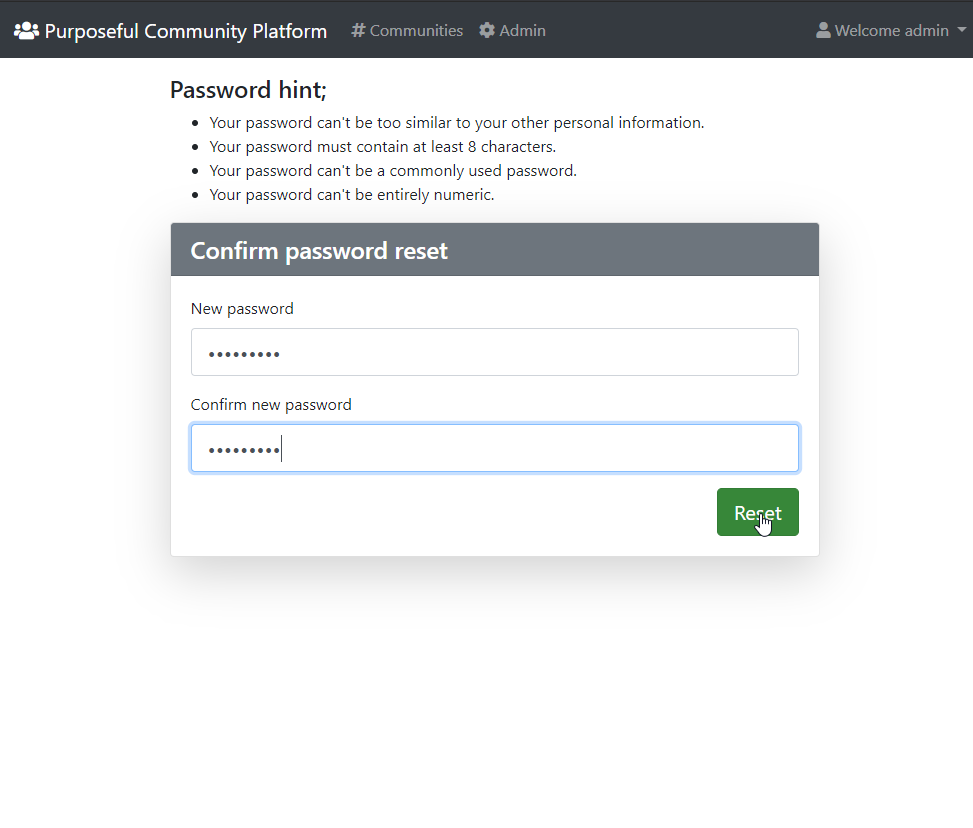
#### 2) Provide email account to send password reset instructions



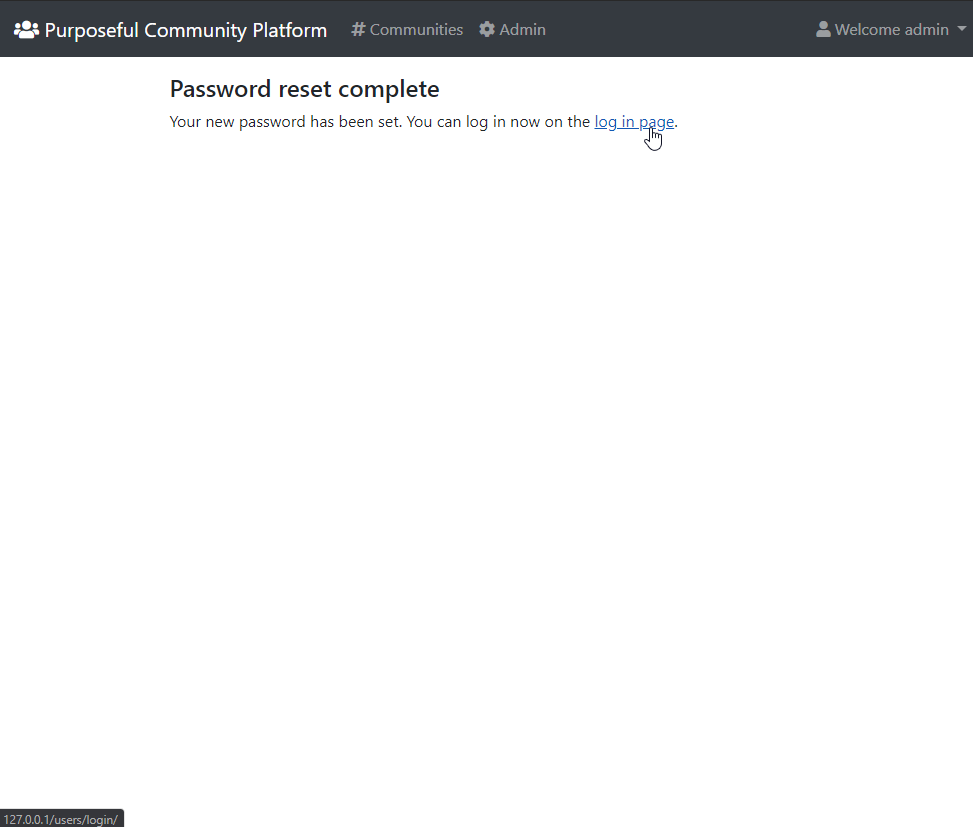
#### 3) Password reset instructions sent to provided email address is email address is valid



#### 4) Password reset link displays new password dialog



#### 5) Password reset is complete, clicking "login page" link takes user to login page



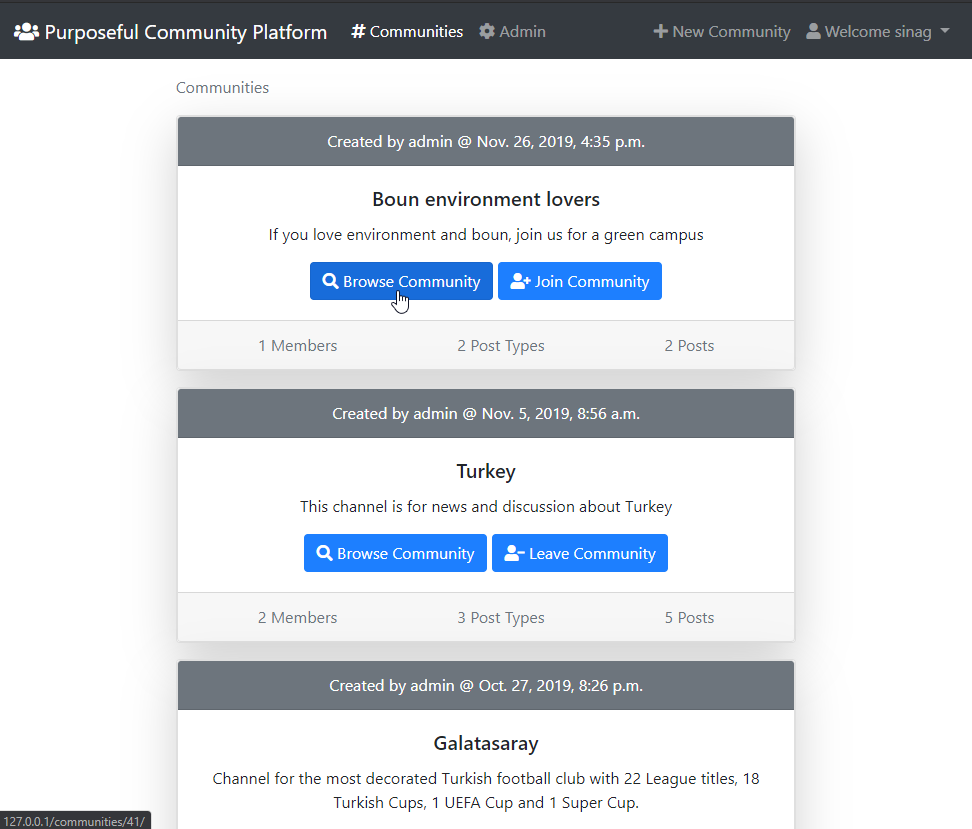
## 14.4 Browse communities

### Related functional requirements

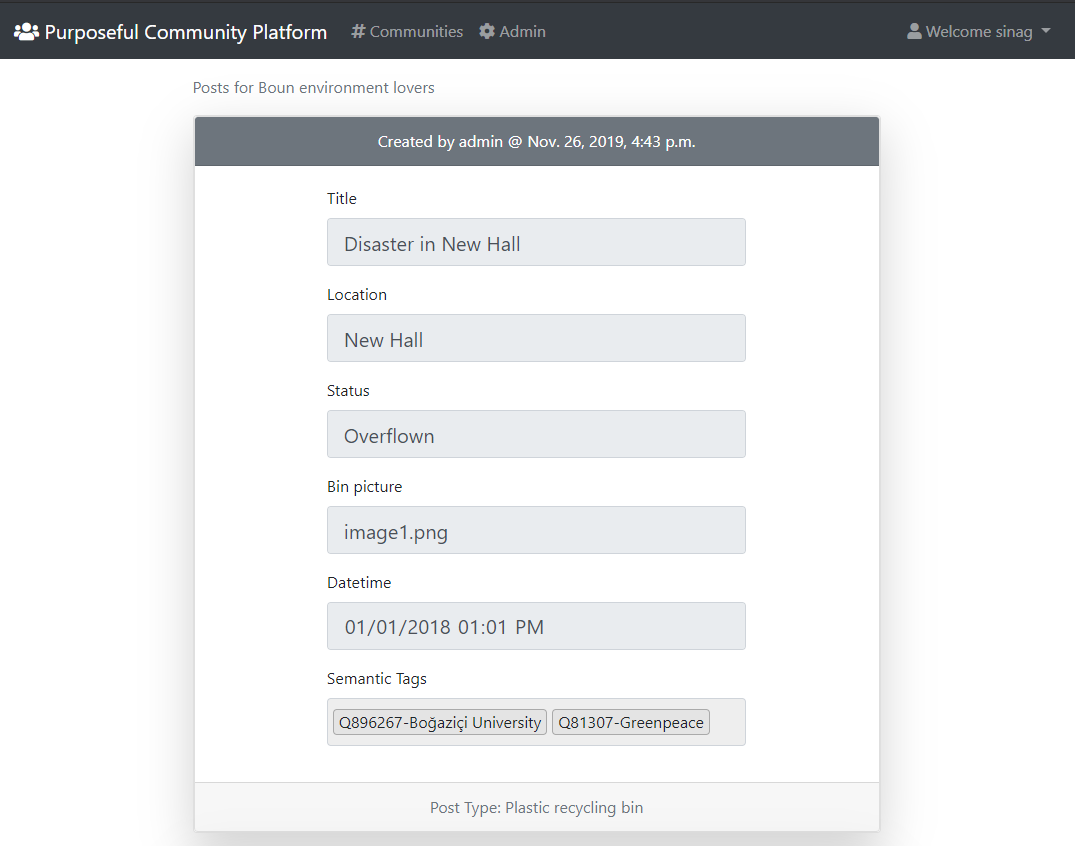
* 4.1.14 - Browse Communities

### Steps

#### 1) As an anonymous/registered user, click "Browse Community" button for community to be browsed on homepage



#### 2) All posts inside this community (Generic/Custom) will be displayed as a list



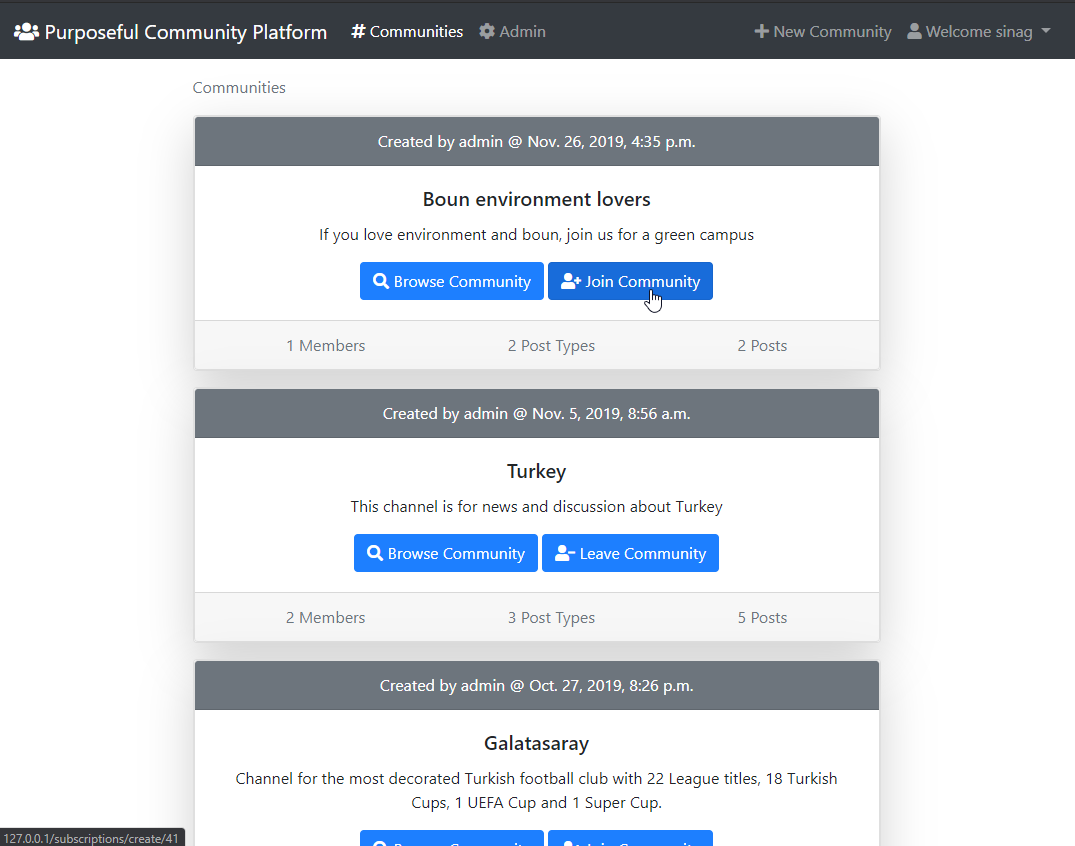
## 14.5 Join community

### Related functional requirements

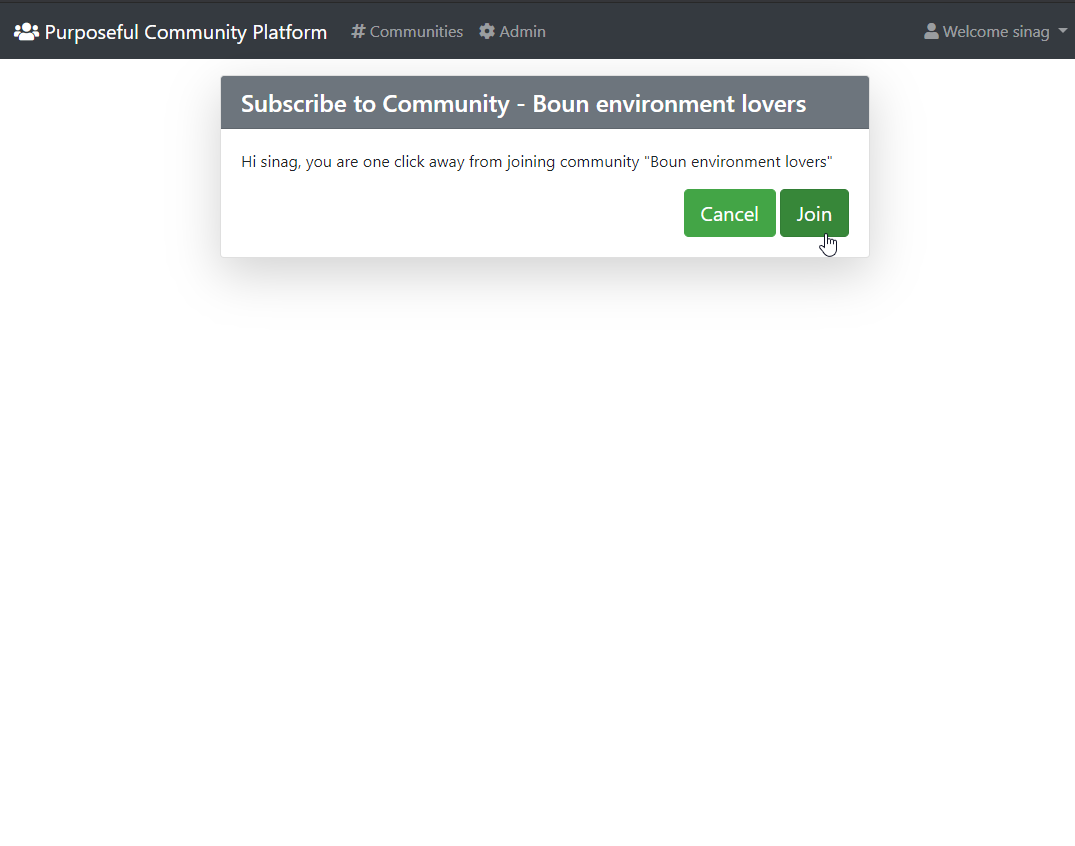
* 4.1.15 - Subscription

### Steps

#### 1) As a registered user, click "Join Community" button on homepage to subscribe for a community



#### 2) Confirm community subscription



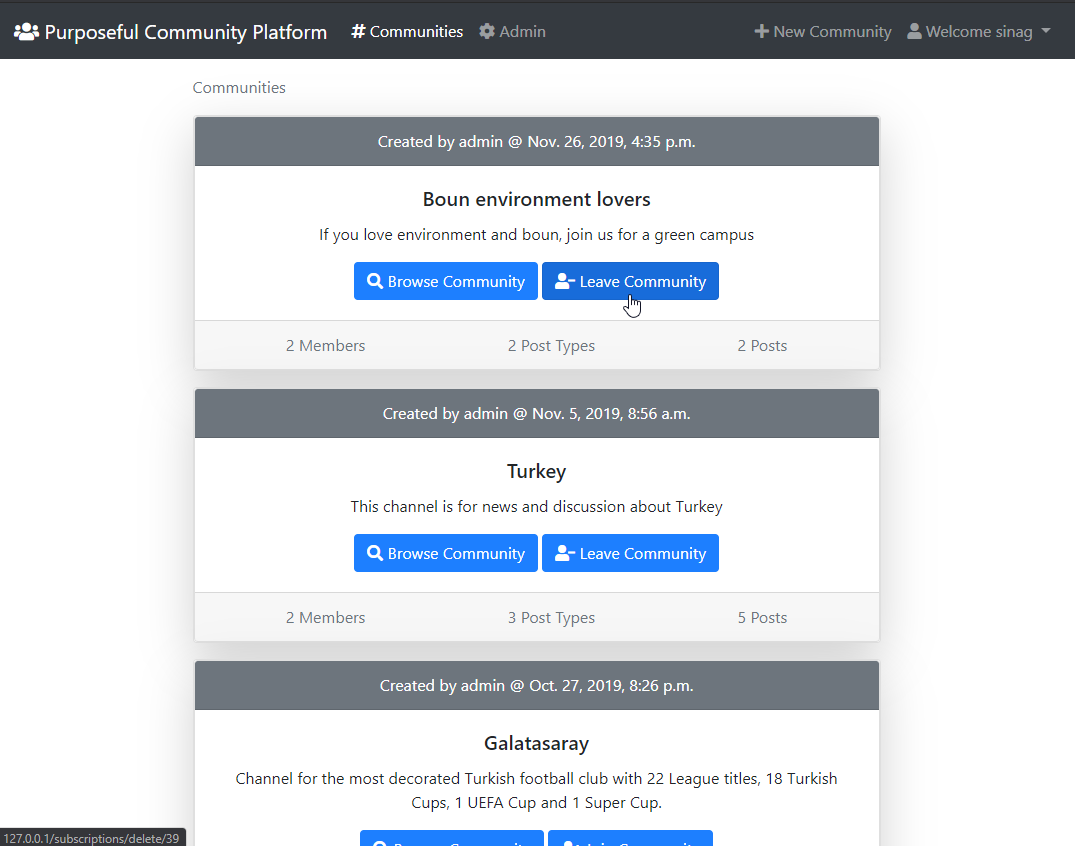
## 14.6 Leave community

### Related functional requirements

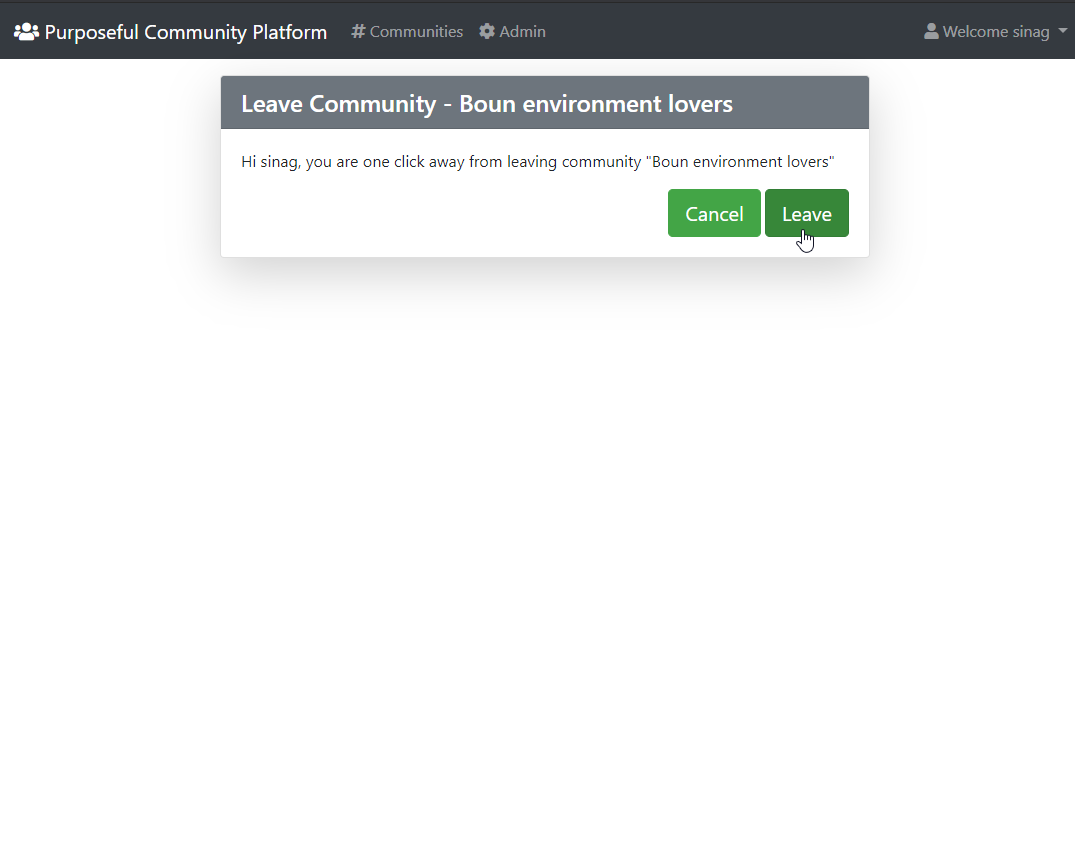
* 4.1.15 - Subscription

### Steps

#### 1) As a registered and already subscribed user, click "Leave Community" button on homepage to unsubscribe from a community



#### 2) Confirm operation



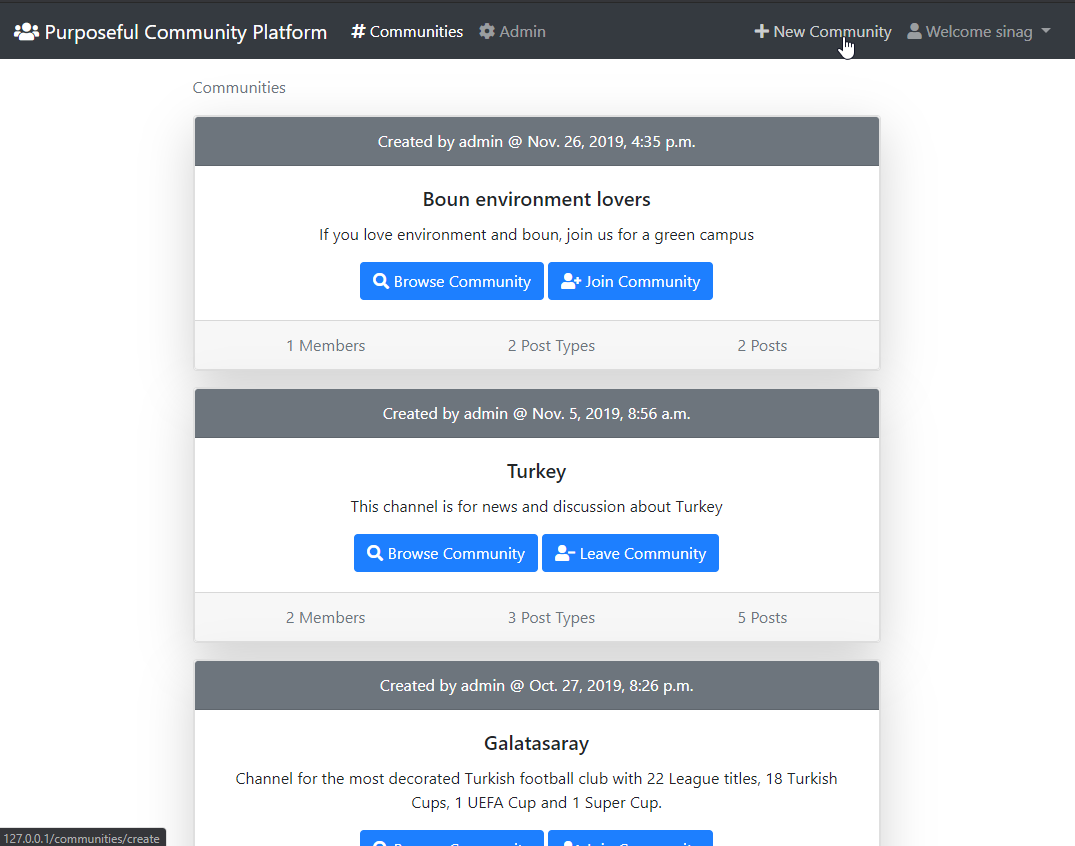
## 14.7 Create new community

### Related functional requirements

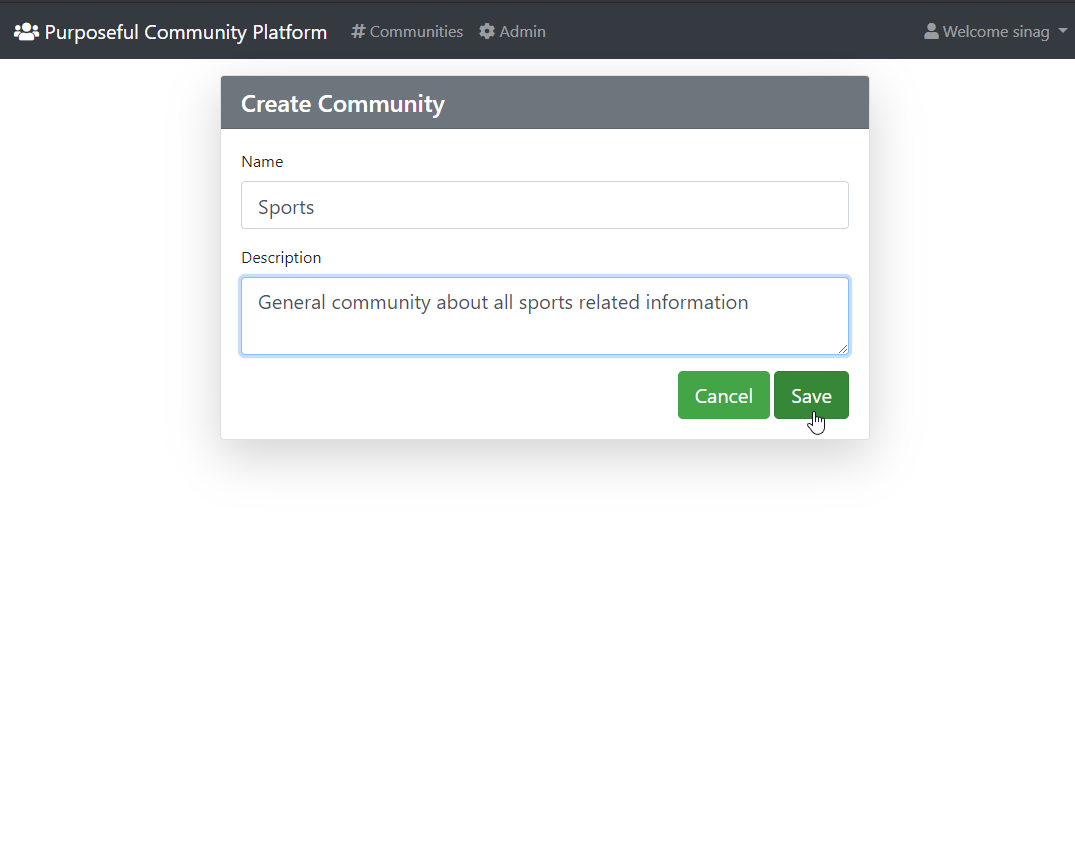
* 4.1.4 - Create Community

### Steps

#### 1) As a registered user, click "Create Community" button from navigation bar on homepage to create a new community



#### 2) Provide title and description for community to be created and click "Save"



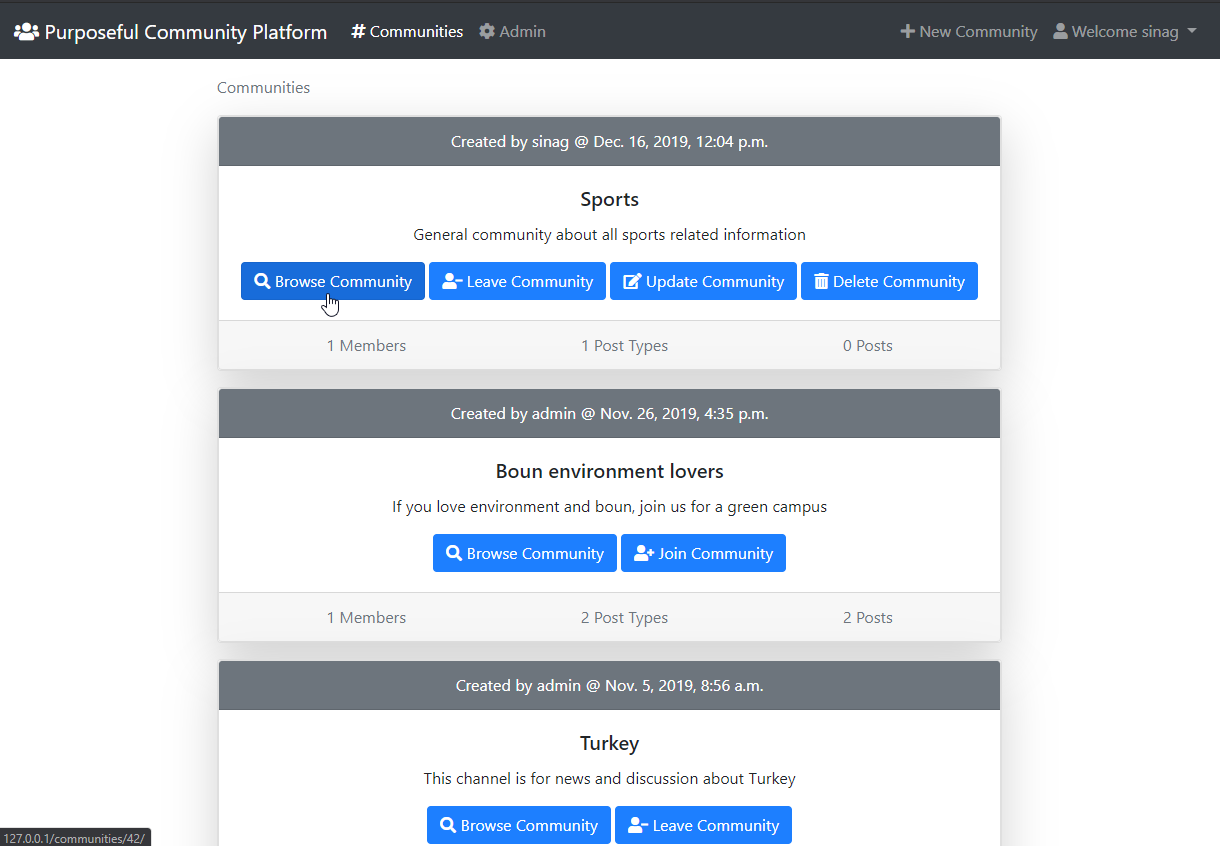
## 14.8 Create new generic post

### Related functional requirements

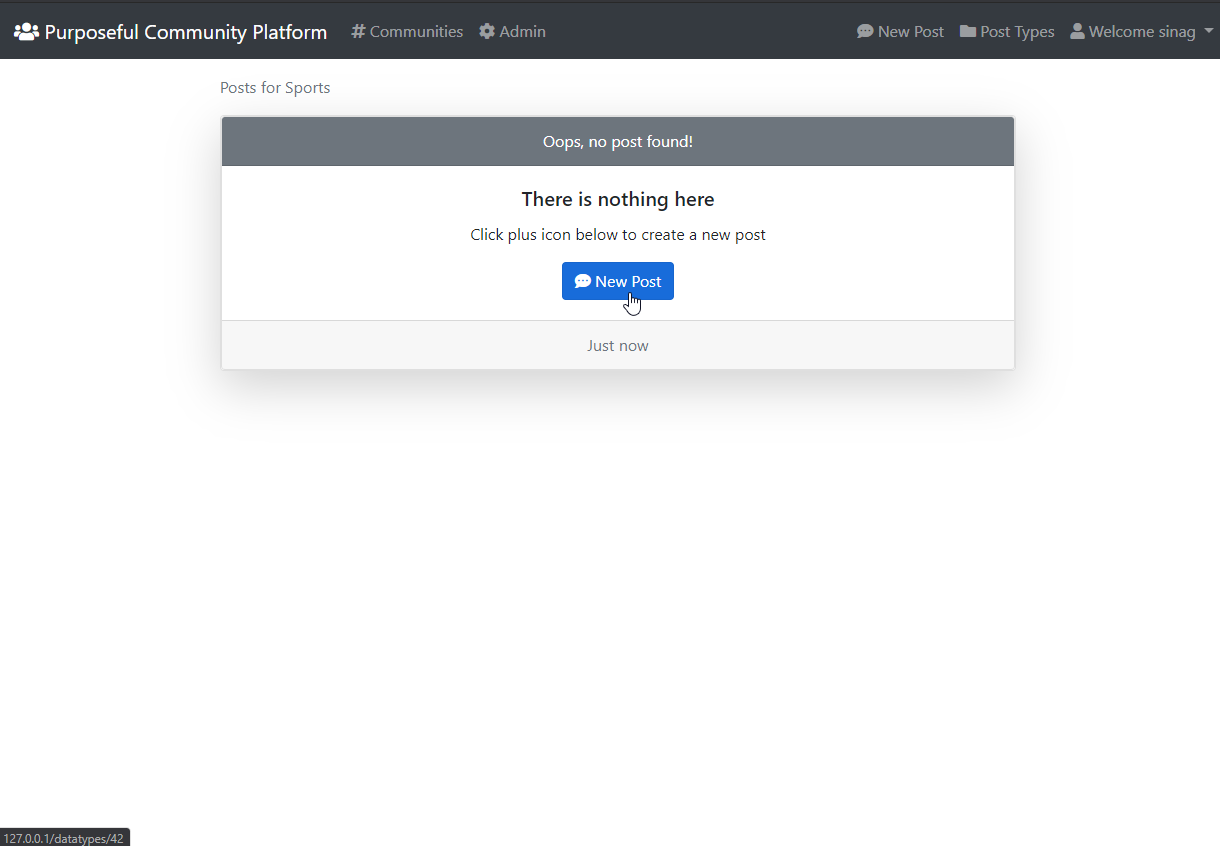
* 4.1.16 - Post
* 4.1.9 - WikiData Tags
* 4.1.7 - Generic Data Type Structure

### Steps

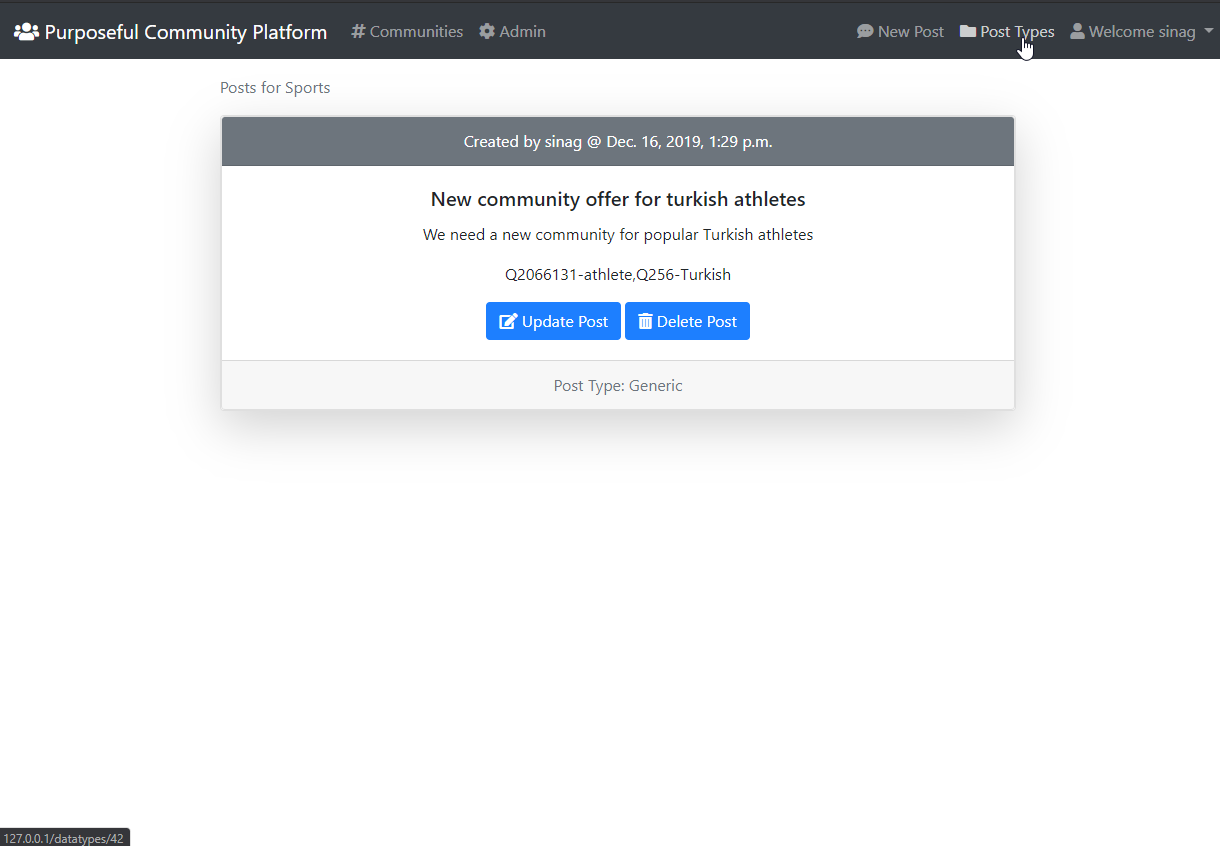
#### 1) As a registered user, click "Browse Community" to select the community where new post will be created



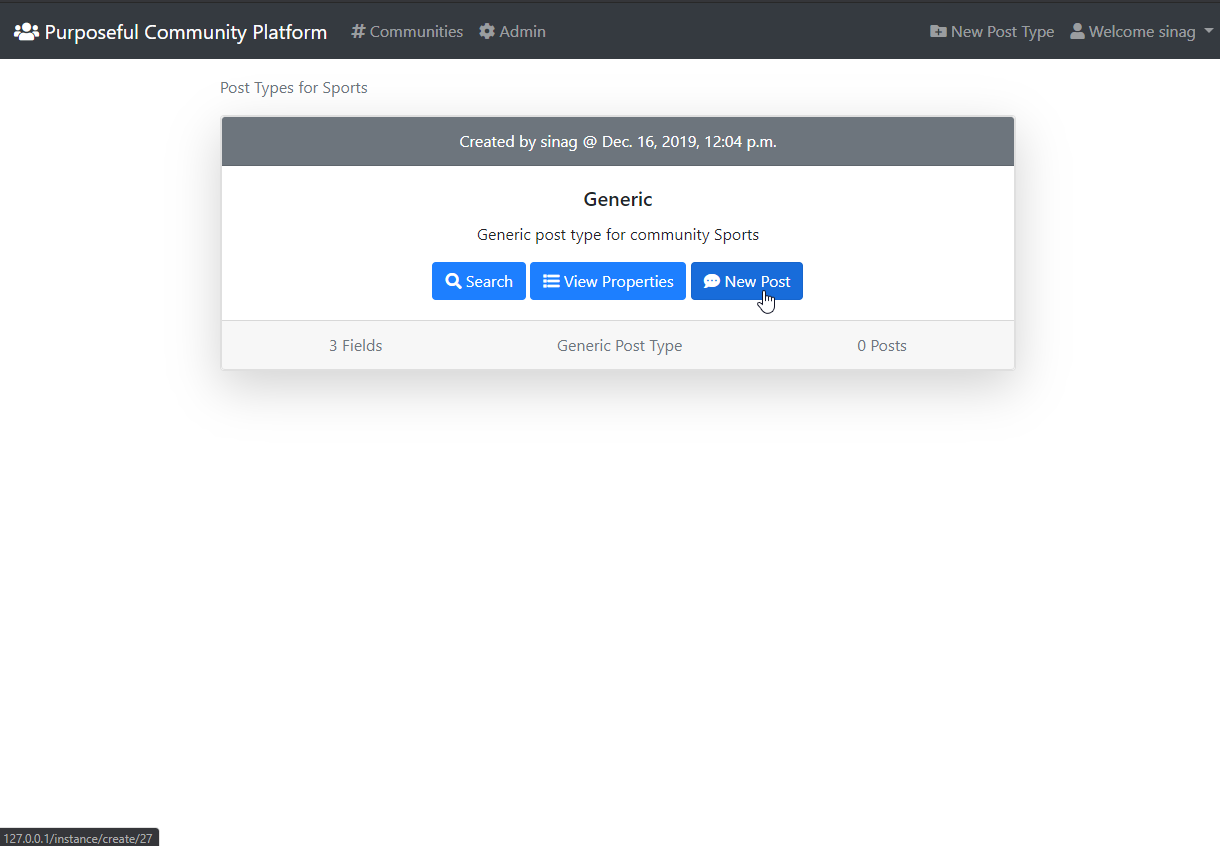
#### 2) If there are no posts inside a community, "There is nothing here" message will be display with a "New Post" button. Click "New Post button"



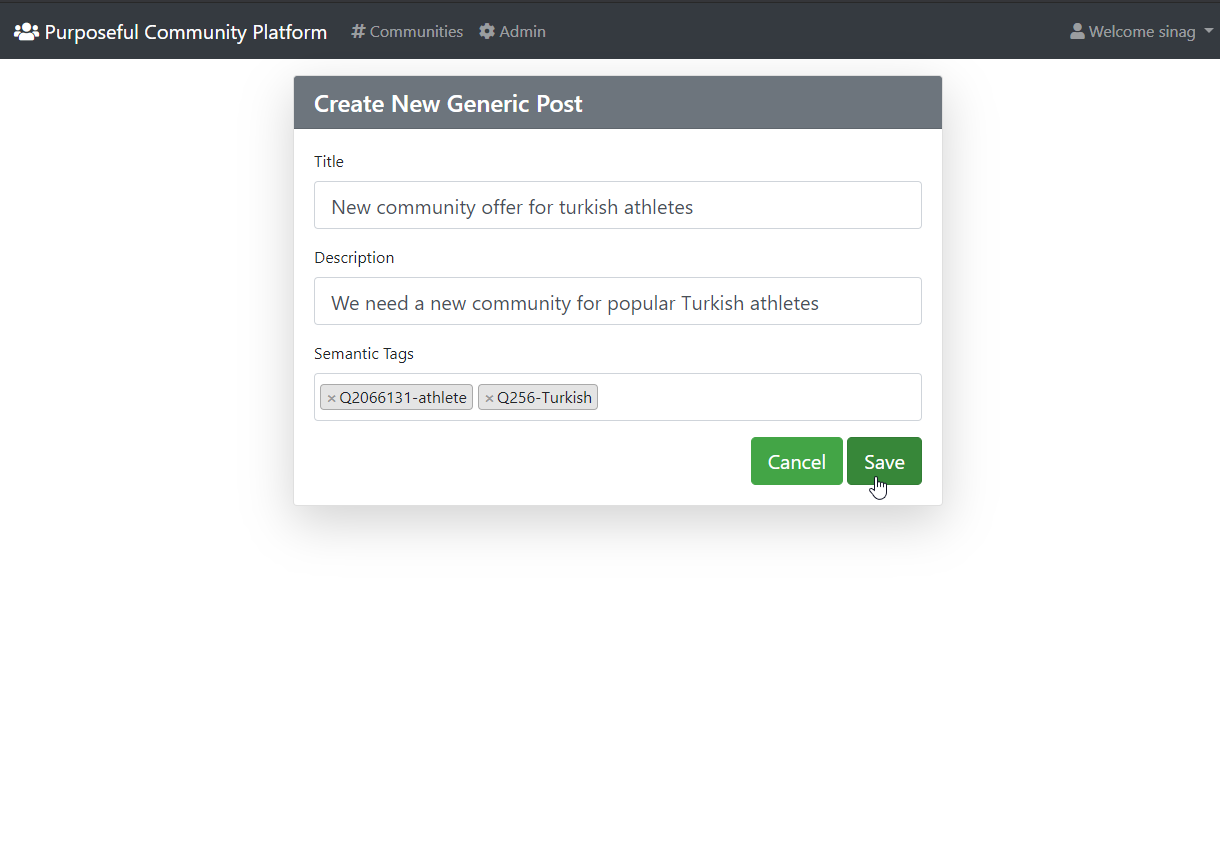
#### 3) If there are posts inside a community, click "Post Types" button on navigation bar



#### 4) Click "New Post" button on generic data type



#### 5) Provide title, description and semantic tags for generic post and click "Save" button



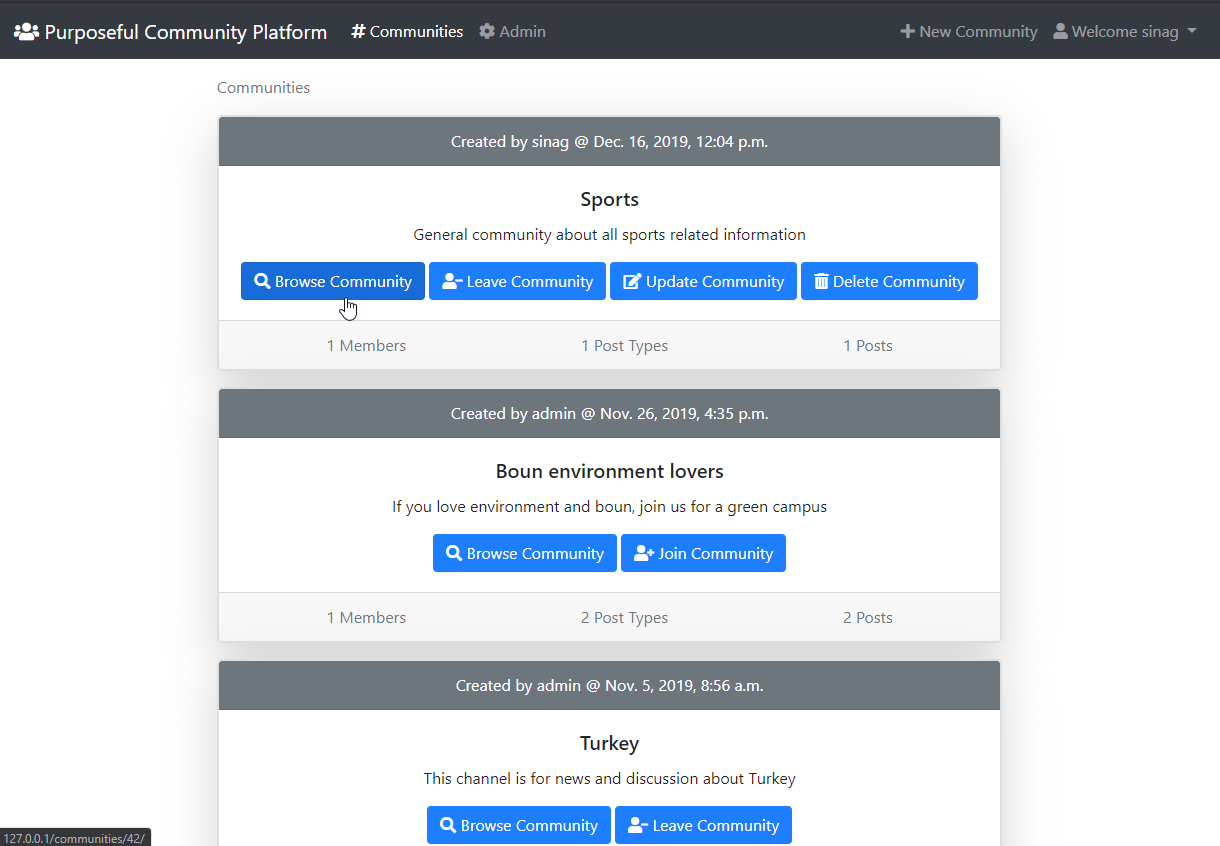
## 14.9 Create new custom post type

### Related functional requirements

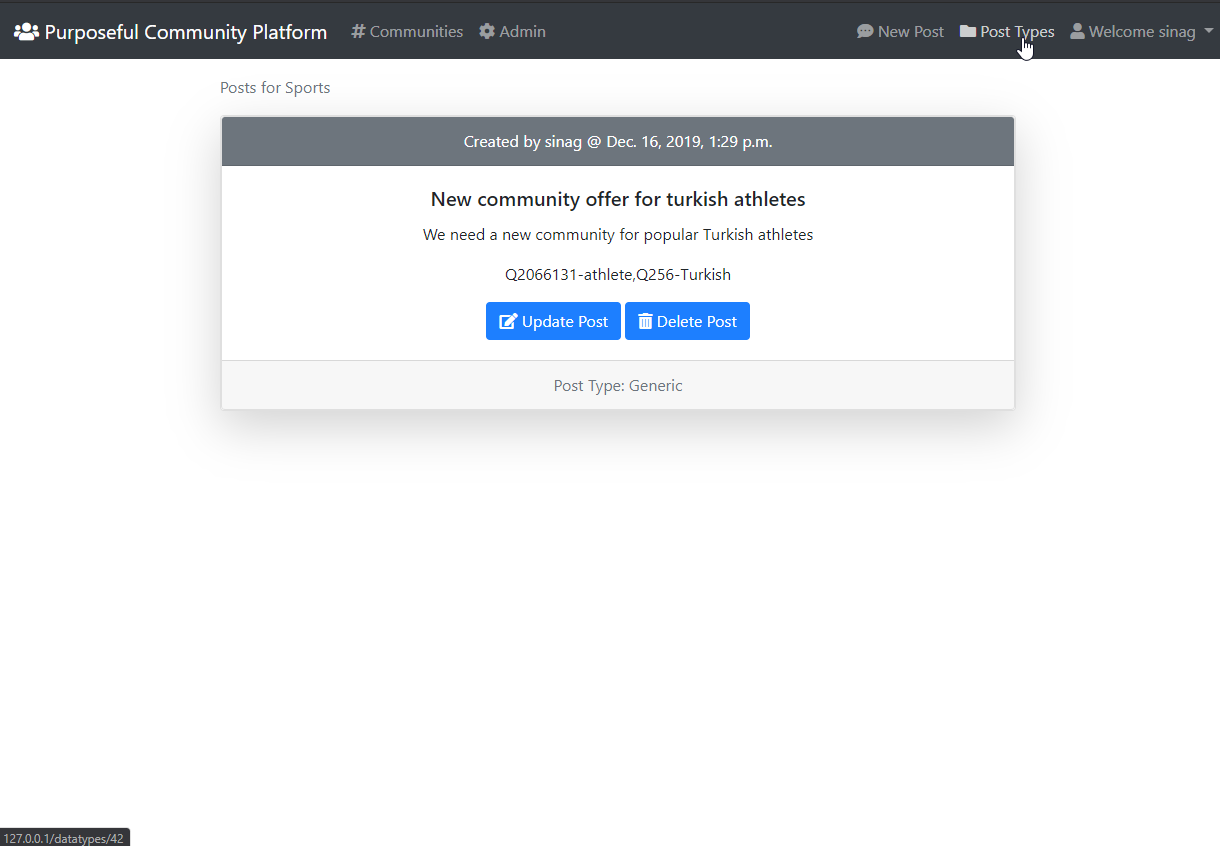
* 4.1.5 - Define Data Types
* 4.1.6 - Content Evolution
* 4.1.8 - Custom Data Type Structure
* 4.1.22 - Required Field
* 4.1.12 - Primitive Types

### Steps

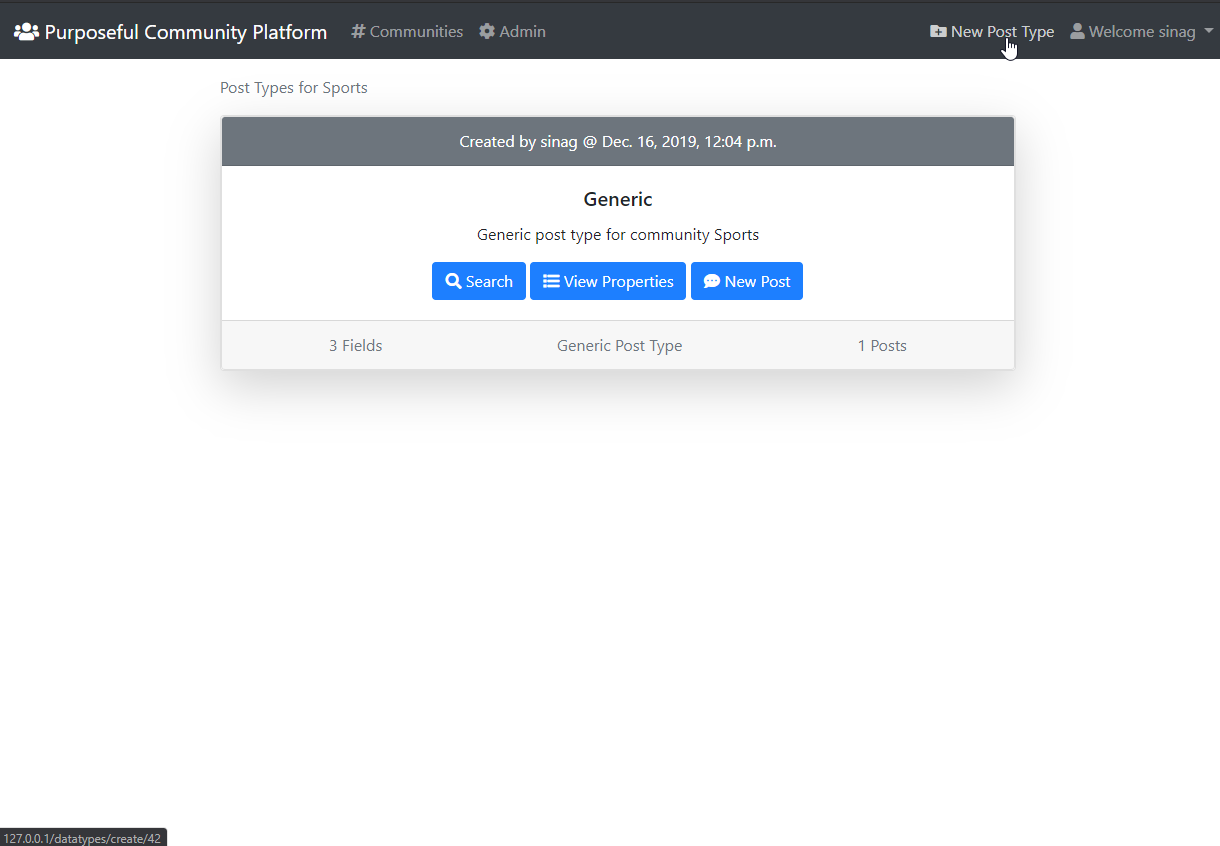
#### 1) As a registered user, click "Browse Community" to select the community where new custom post type will be created



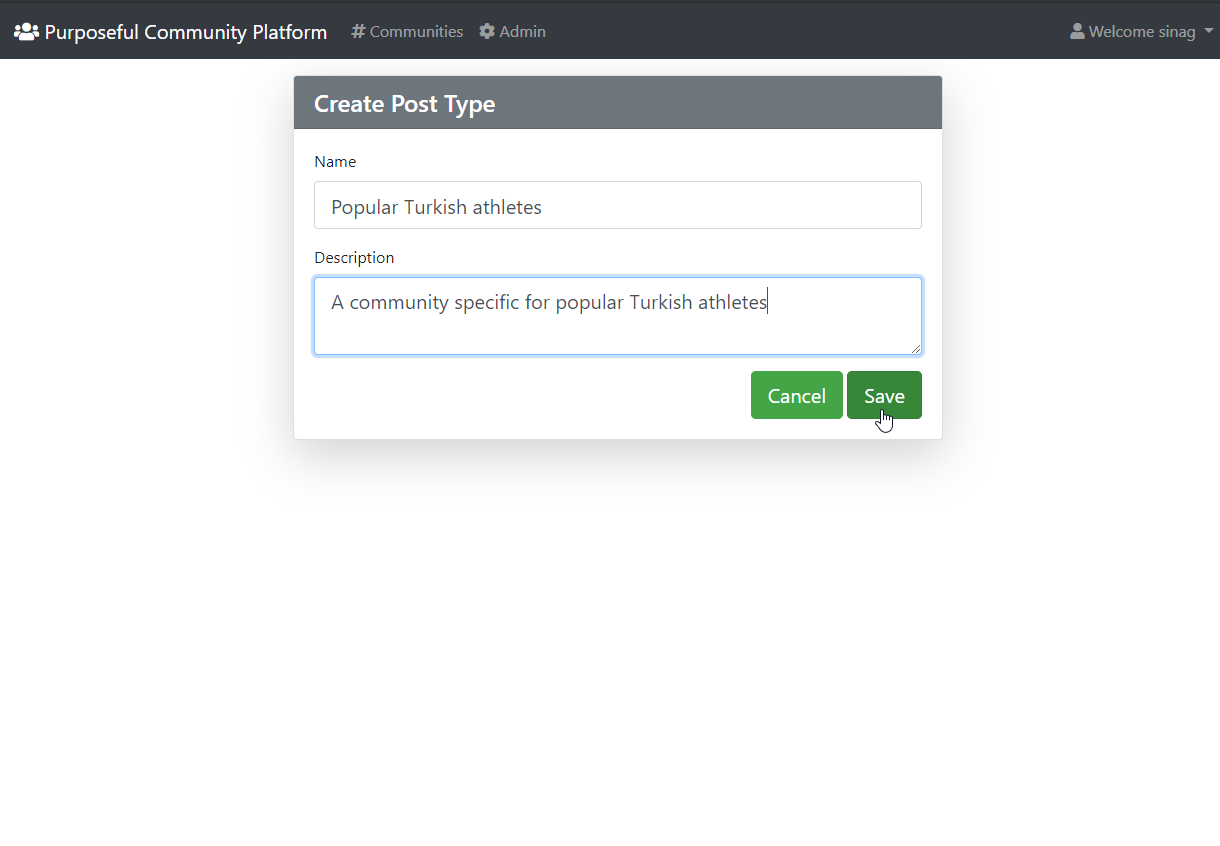
#### 2) Click "Post Types" button on navigation bar



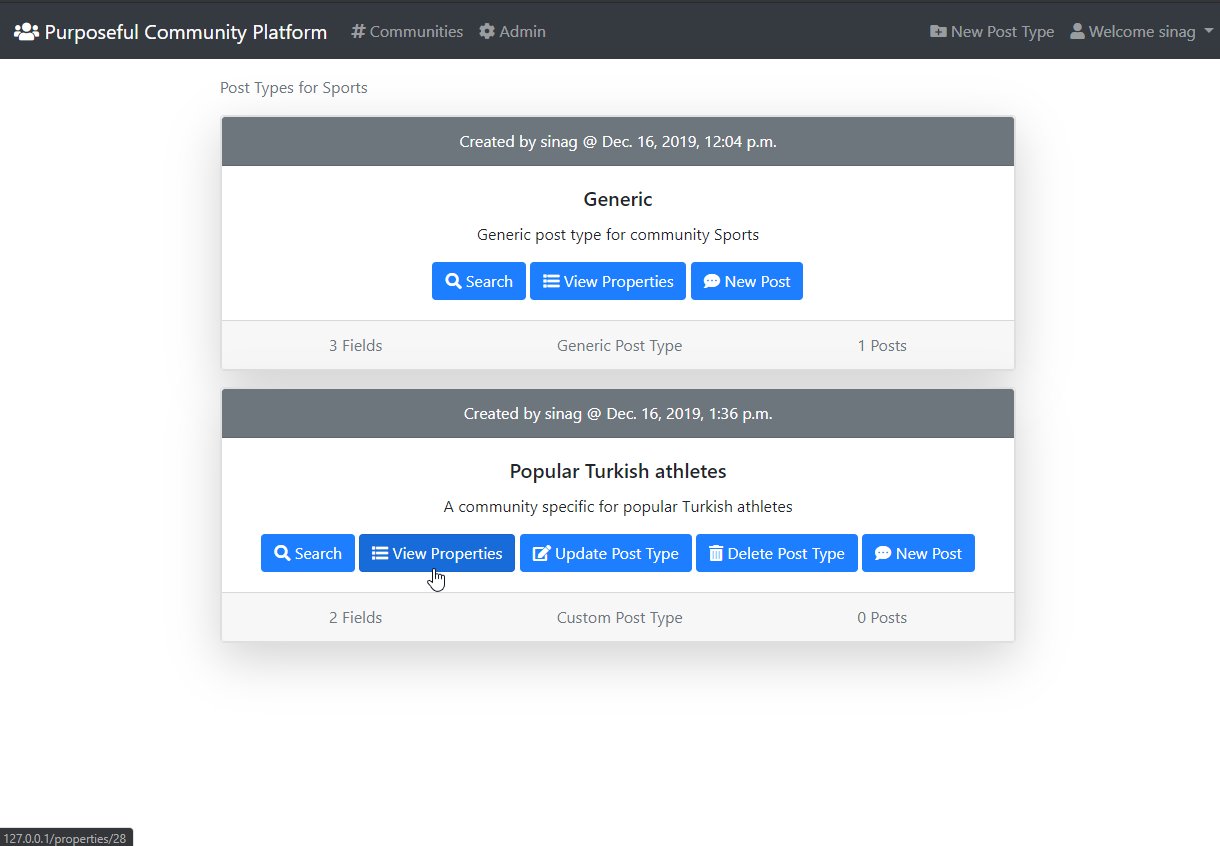
#### 3) Click "New Post Type" button on navigation bar



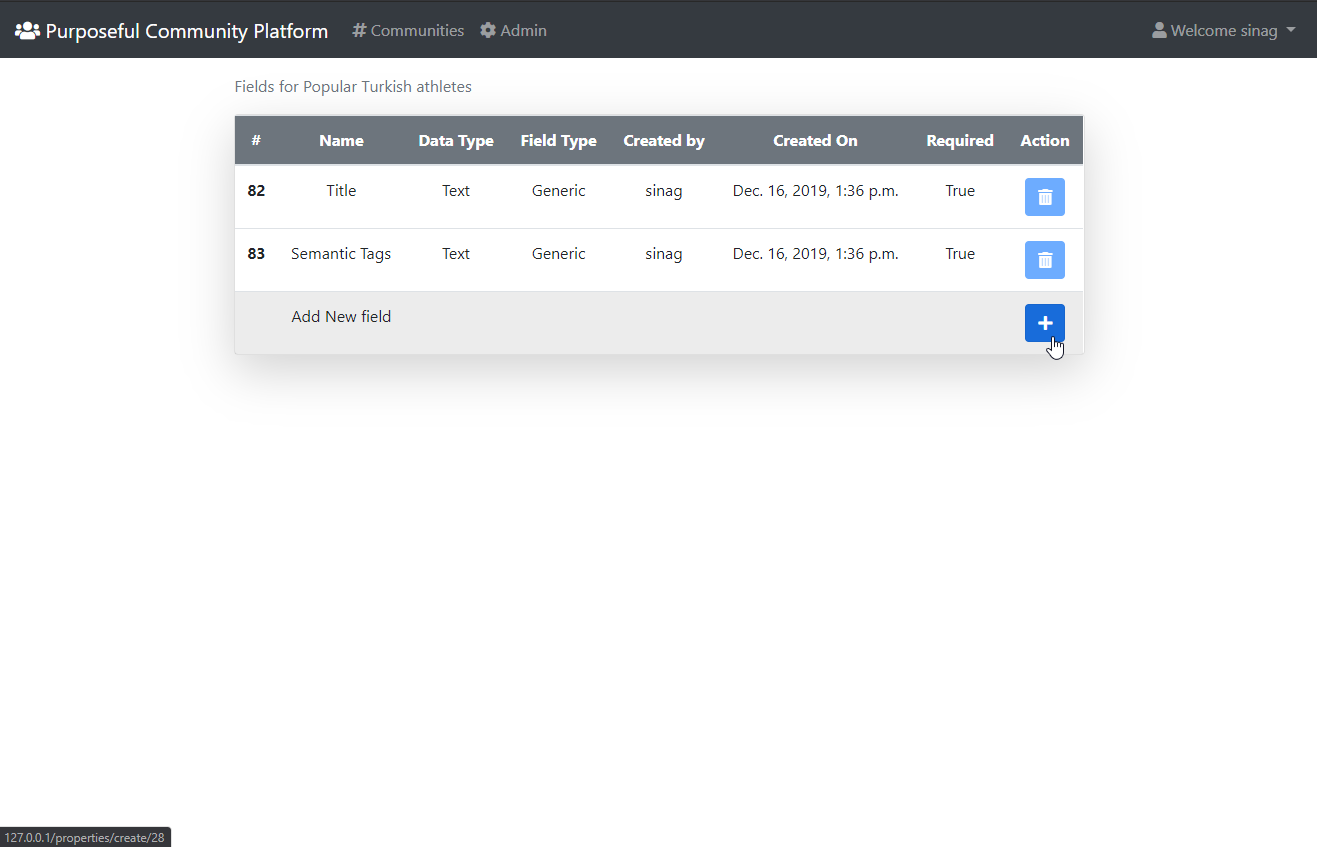
#### 4) Provide title and description and for custom post type to be created and click "Save" button



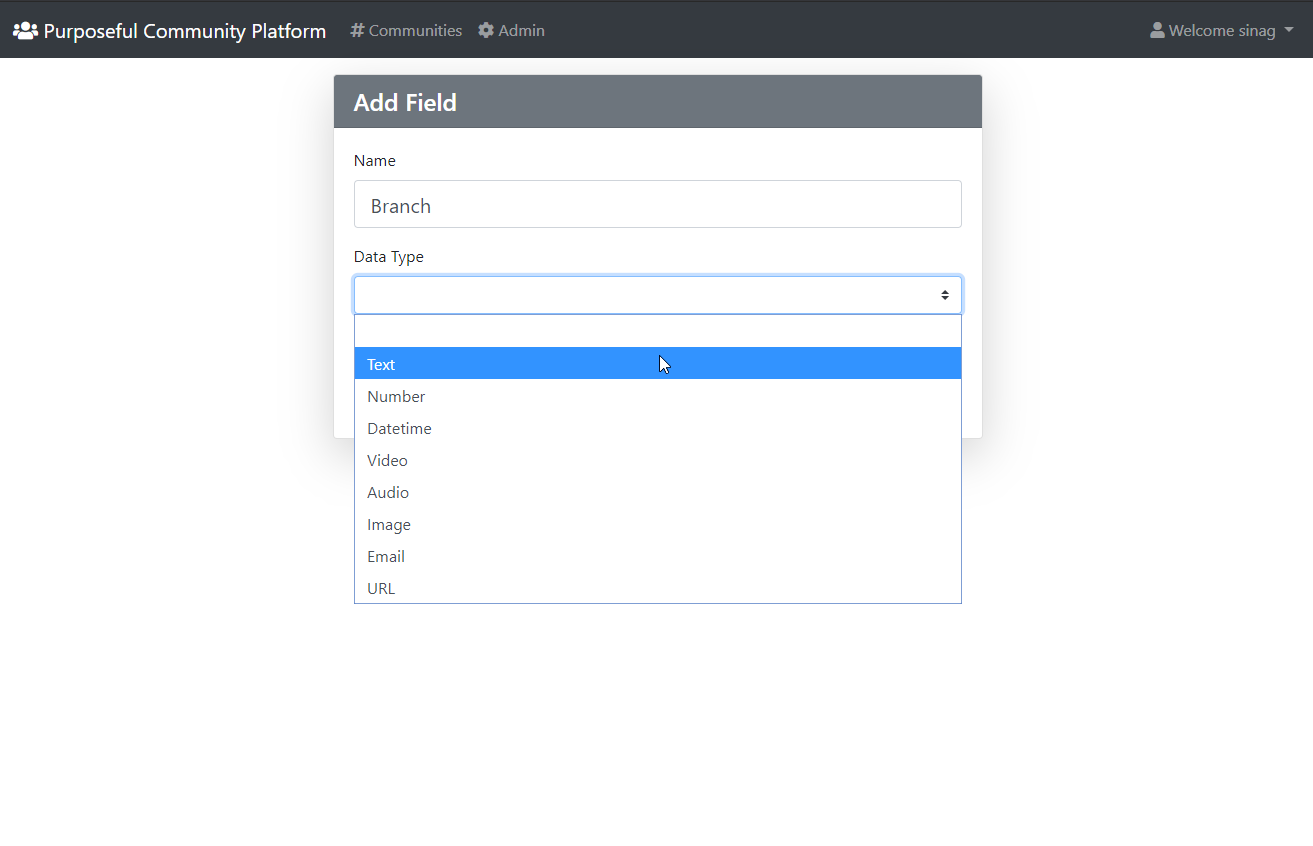
#### 5) User will be redirected to post type listing page for selected community, click "View Fields" to add new properties to created post type



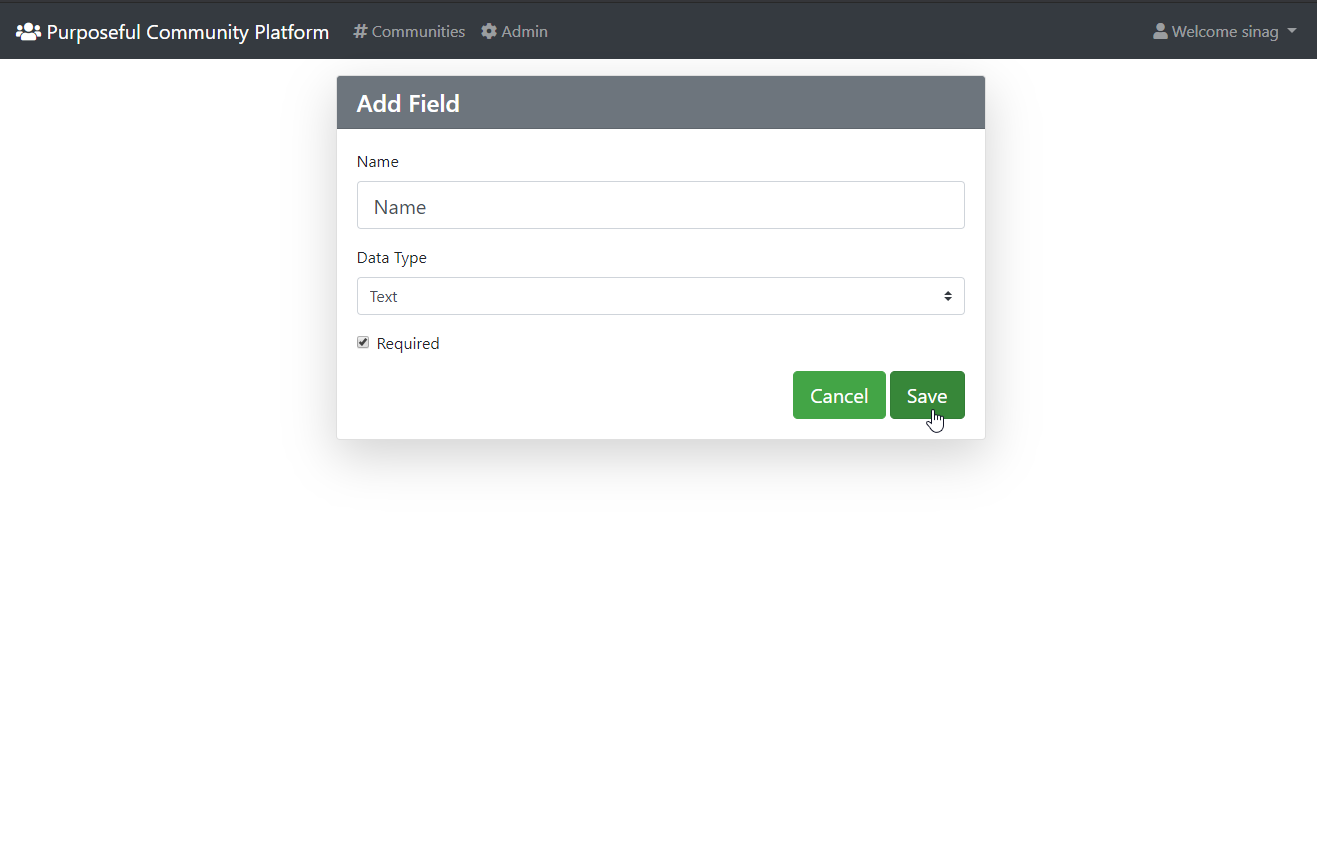
#### 6) Inherited fields from generic post type (title, description and semantic tags) cannot be modified. Click "+" button to add a new field



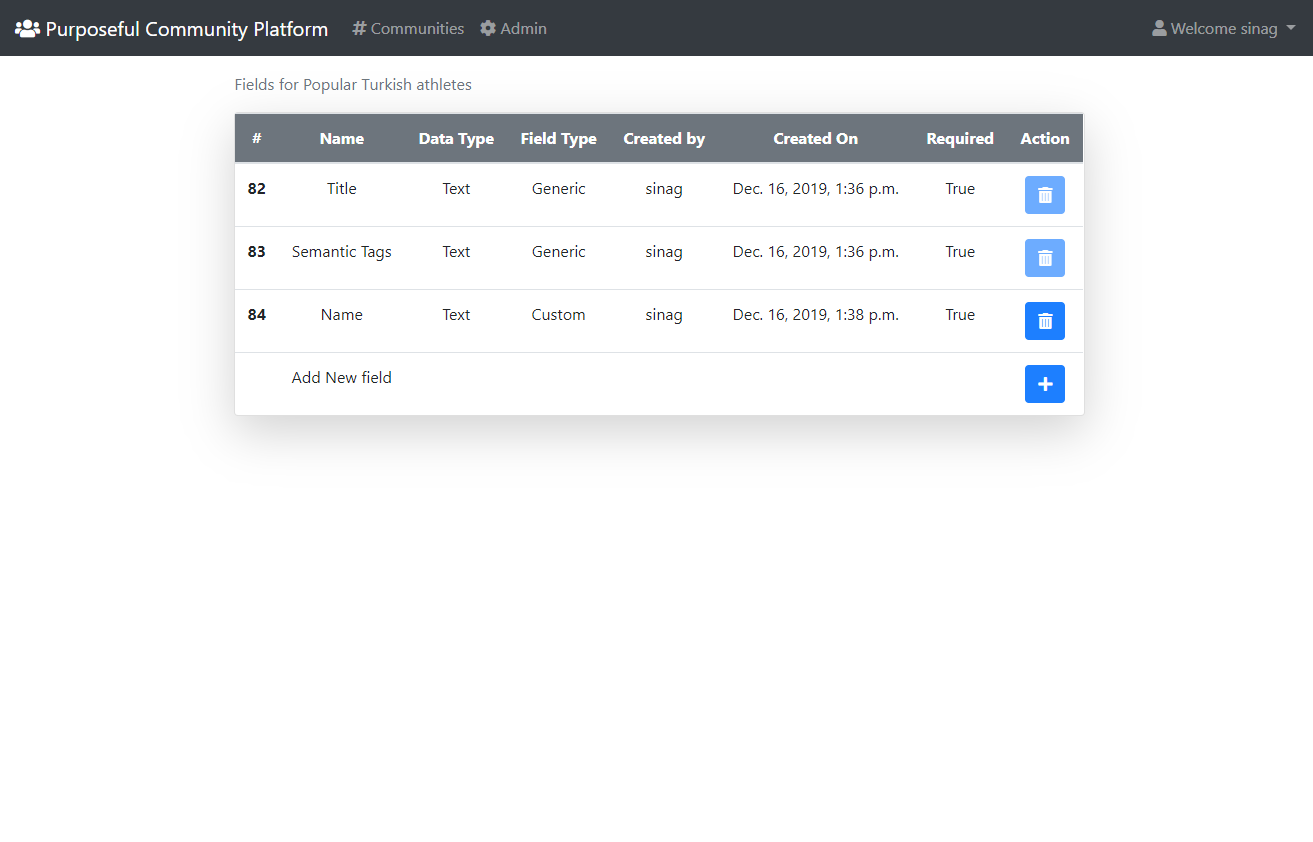
#### 7) Custom fields support these primitive types



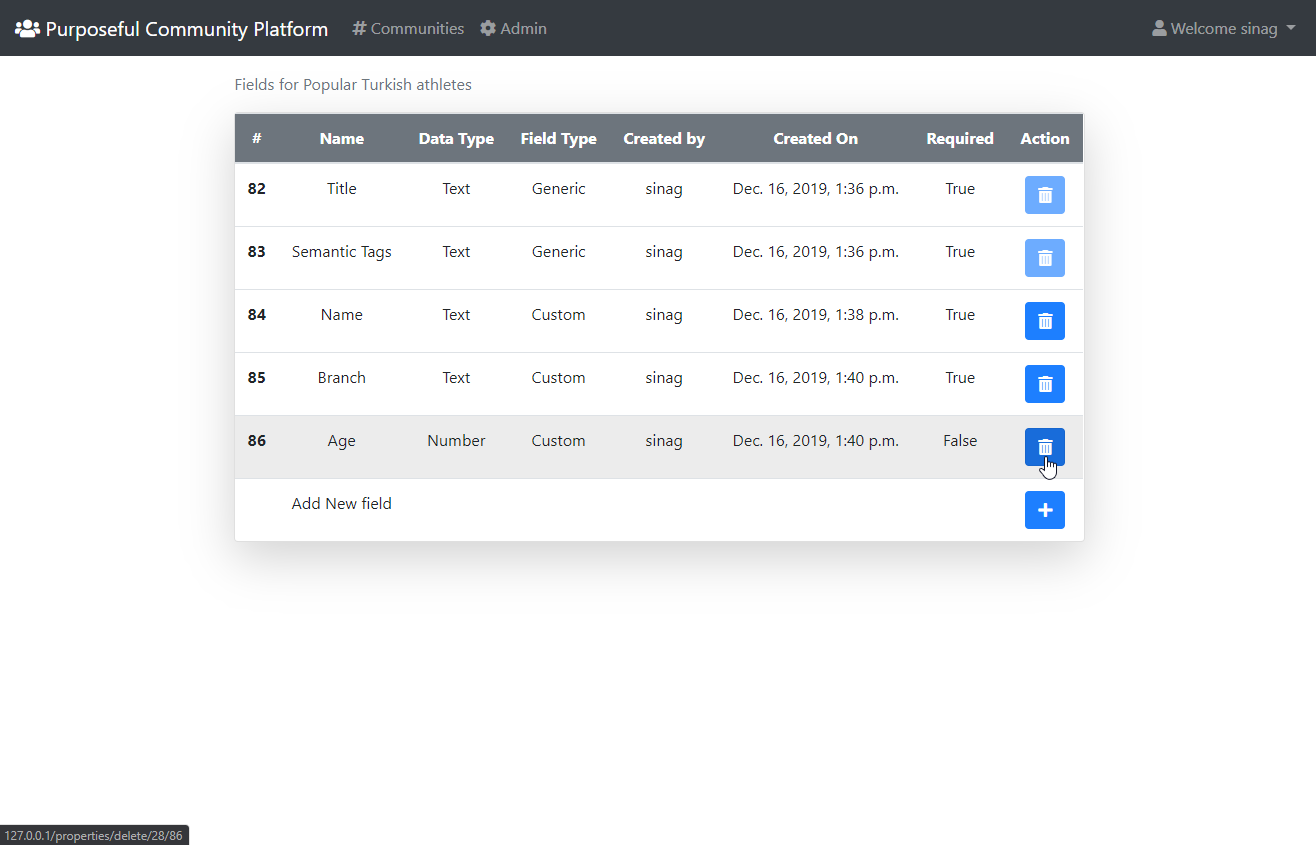
#### 8) Provide name, select primitive type and if needed check required for field to be added and click save



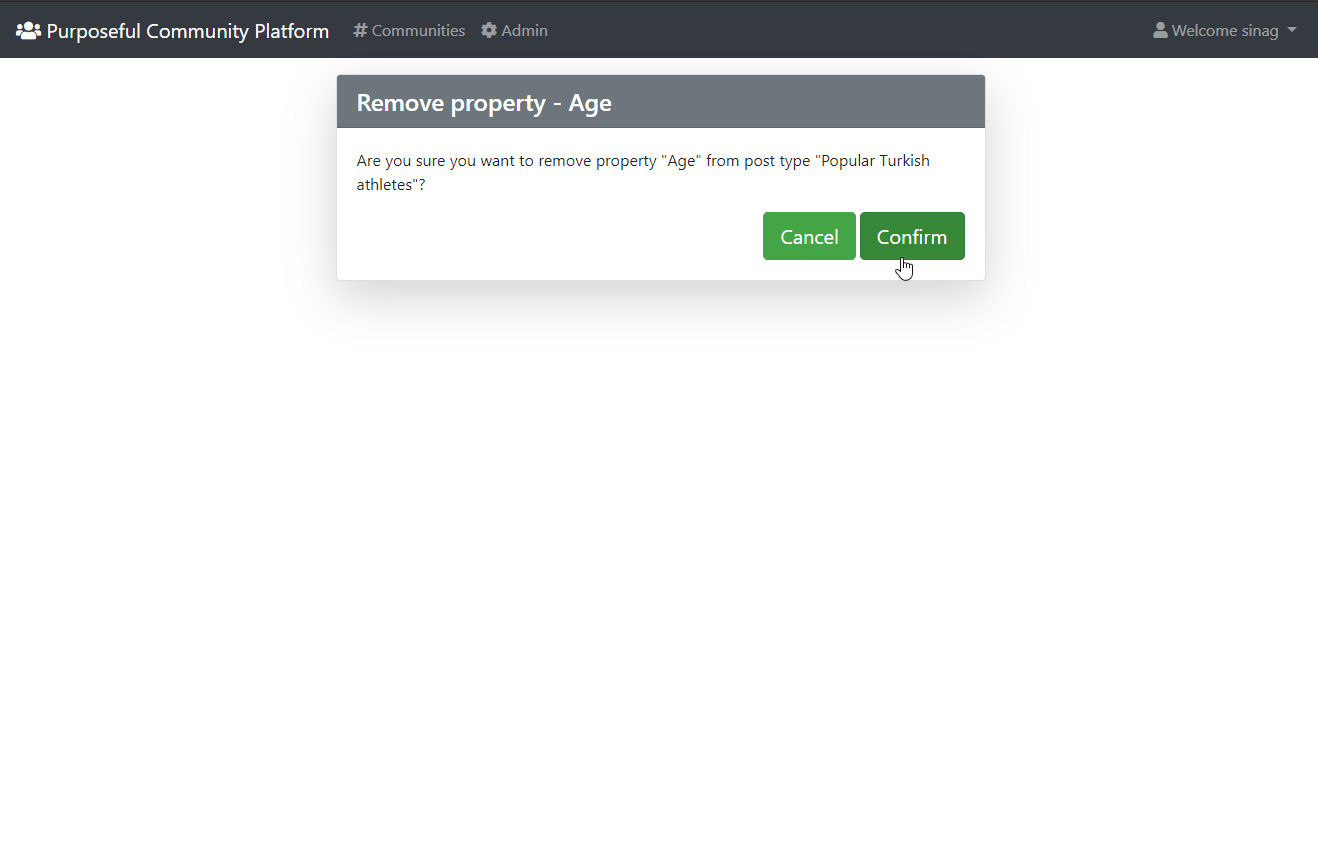
#### 9) Added field appears on the list of fields for selected post type



#### 10) Custom fields can be removed if necessary. Click trashcan image next to the property to be removed



#### 11) Confirm field removal



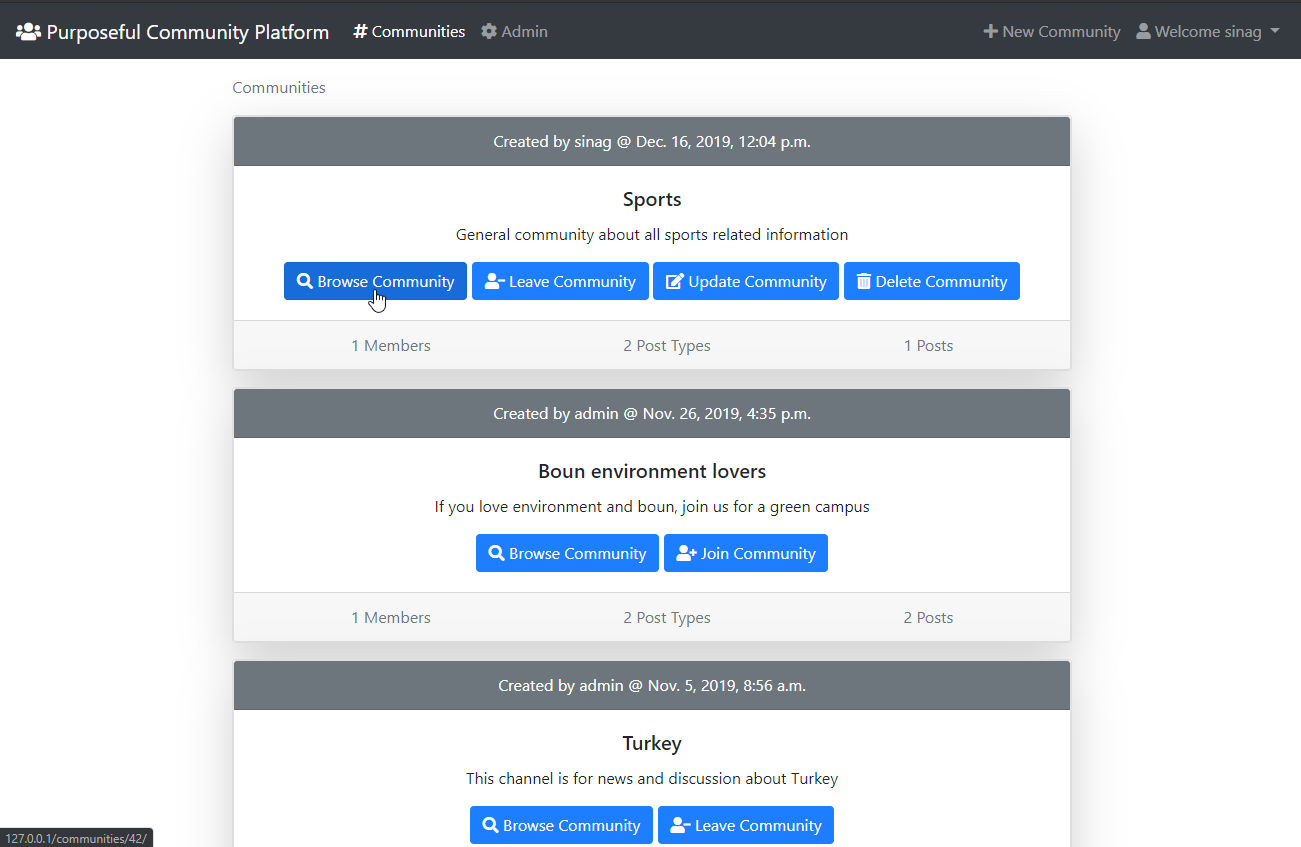
## 14.10 Create new custom post

### Related functional requirements

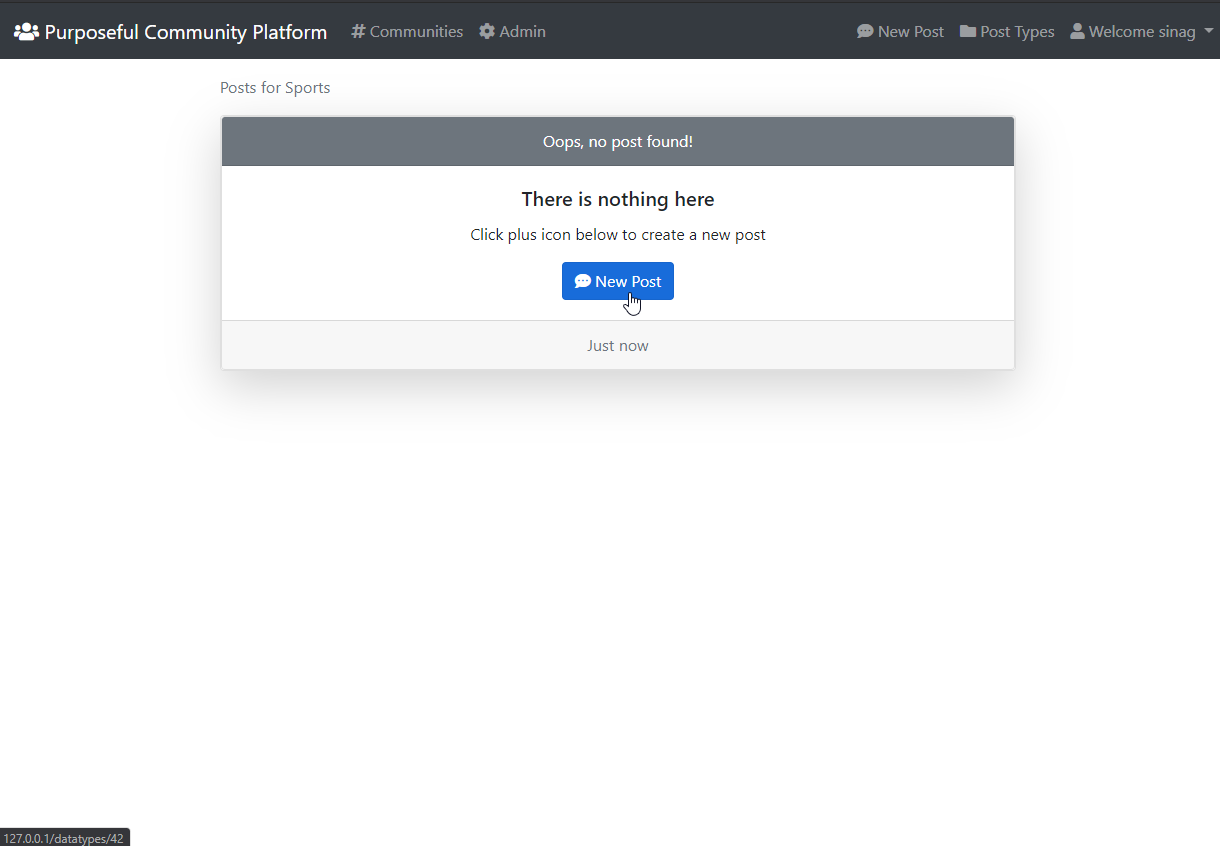
* 4.1.16 - Post
* 4.1.9 - WikiData Tags
* 4.1.8 - Custom Data Type Structure

### Steps

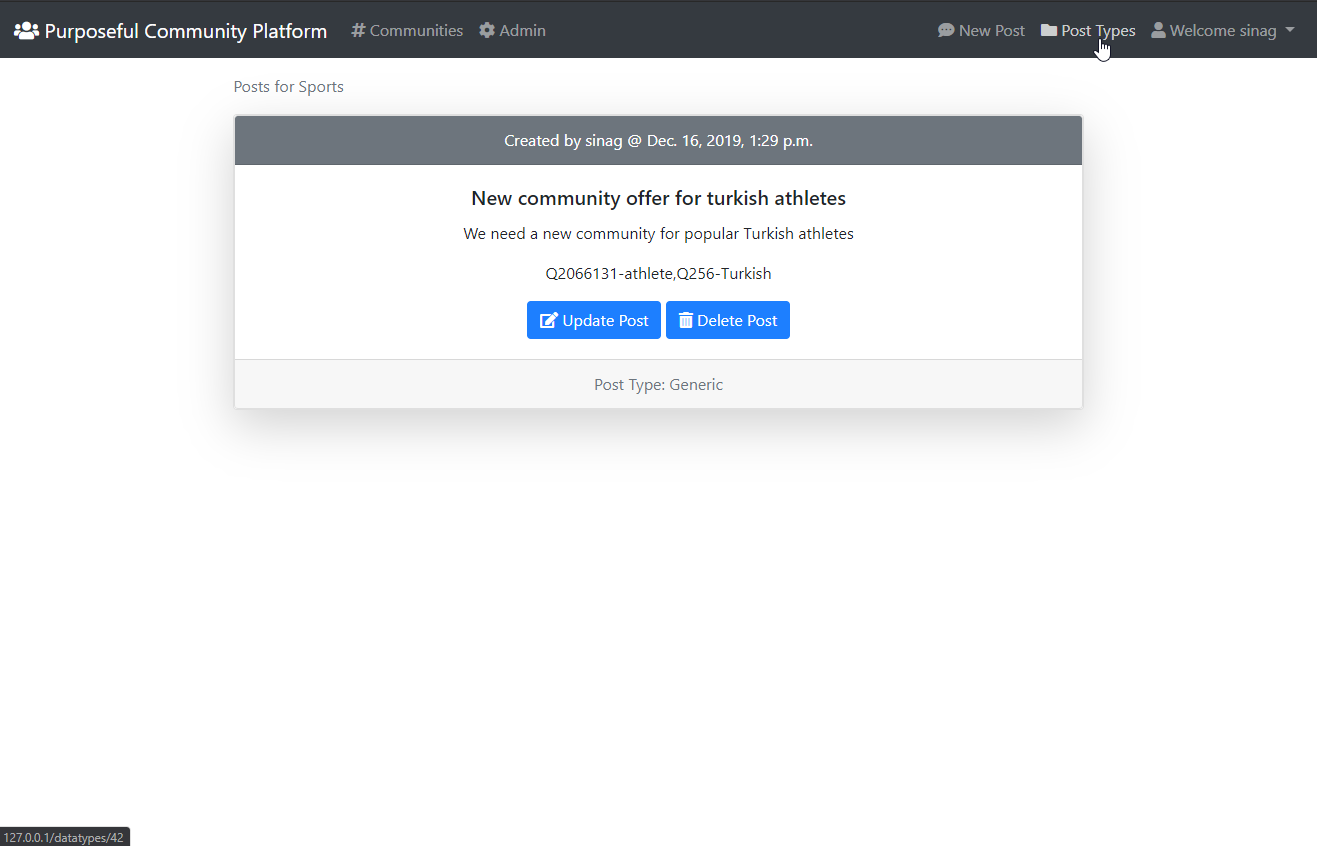
#### 1) As a registered user, click "Browse Community" to select the community where new custom post will be created



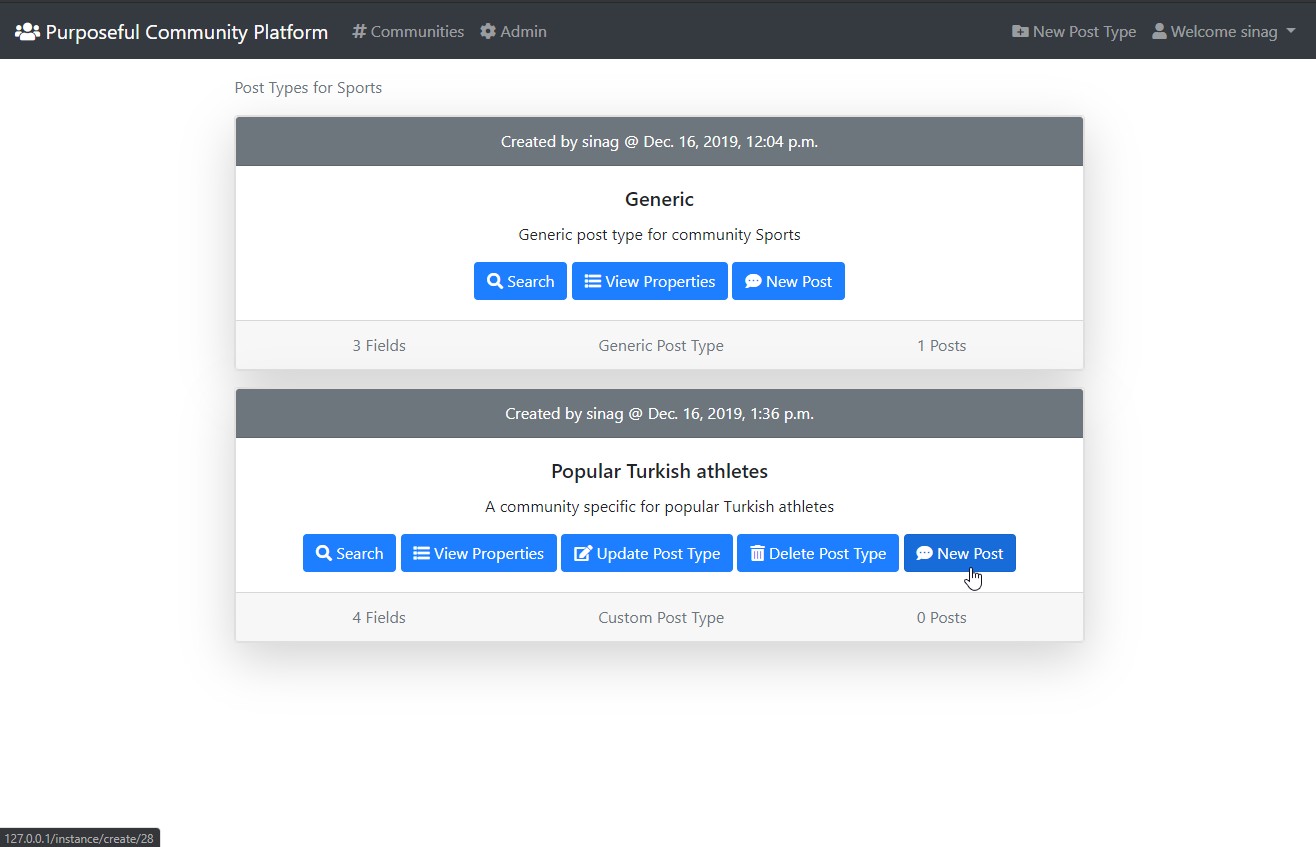
#### 2) If there are no posts inside a community, "There is nothing here" message will be display with a "New Post" button. Click "New Post button"



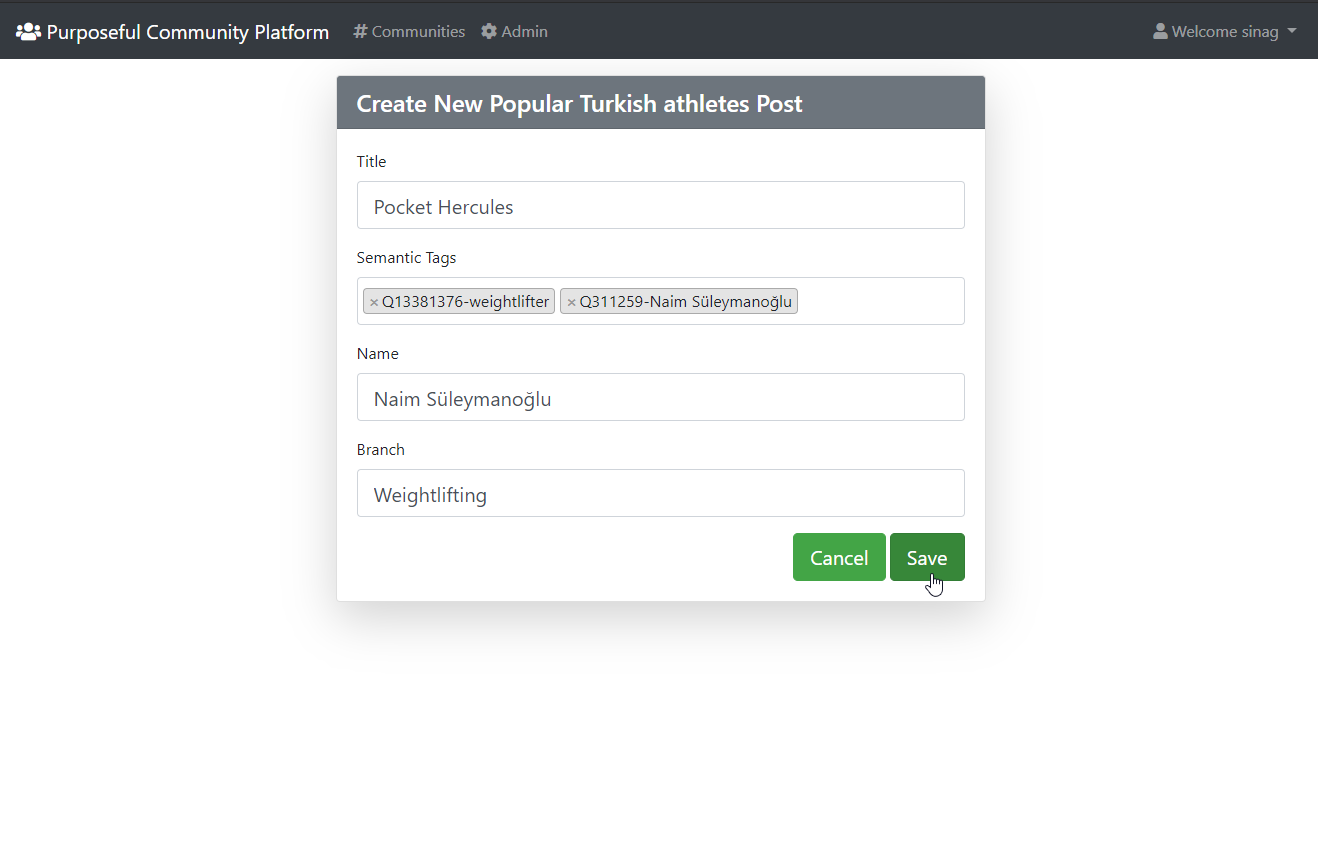
#### 3) If there are posts inside a community, click "Post Types" button on navigation bar



#### 4) Click "New Post" button on custom post type



#### 5) Provide values for custom post and click "Save" button



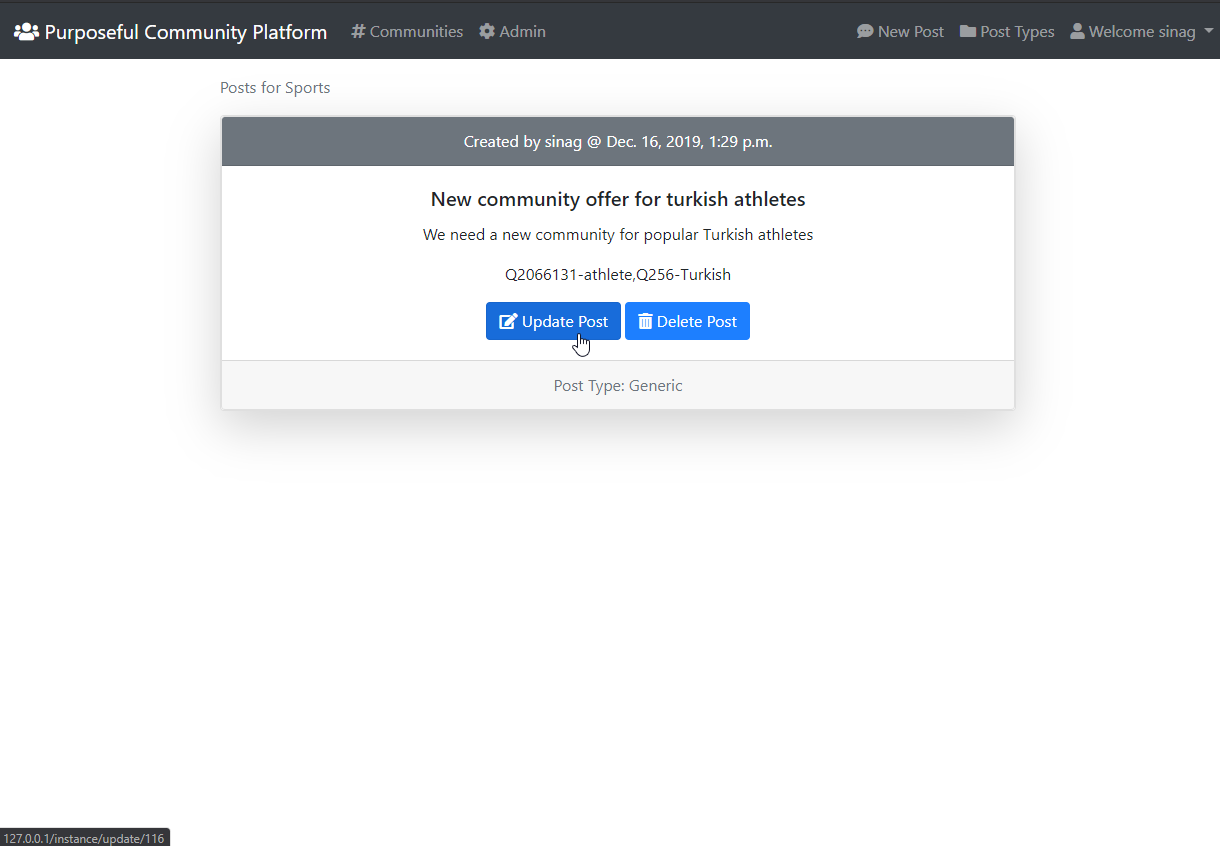
## 14.11 Update post

### Related functional requirements

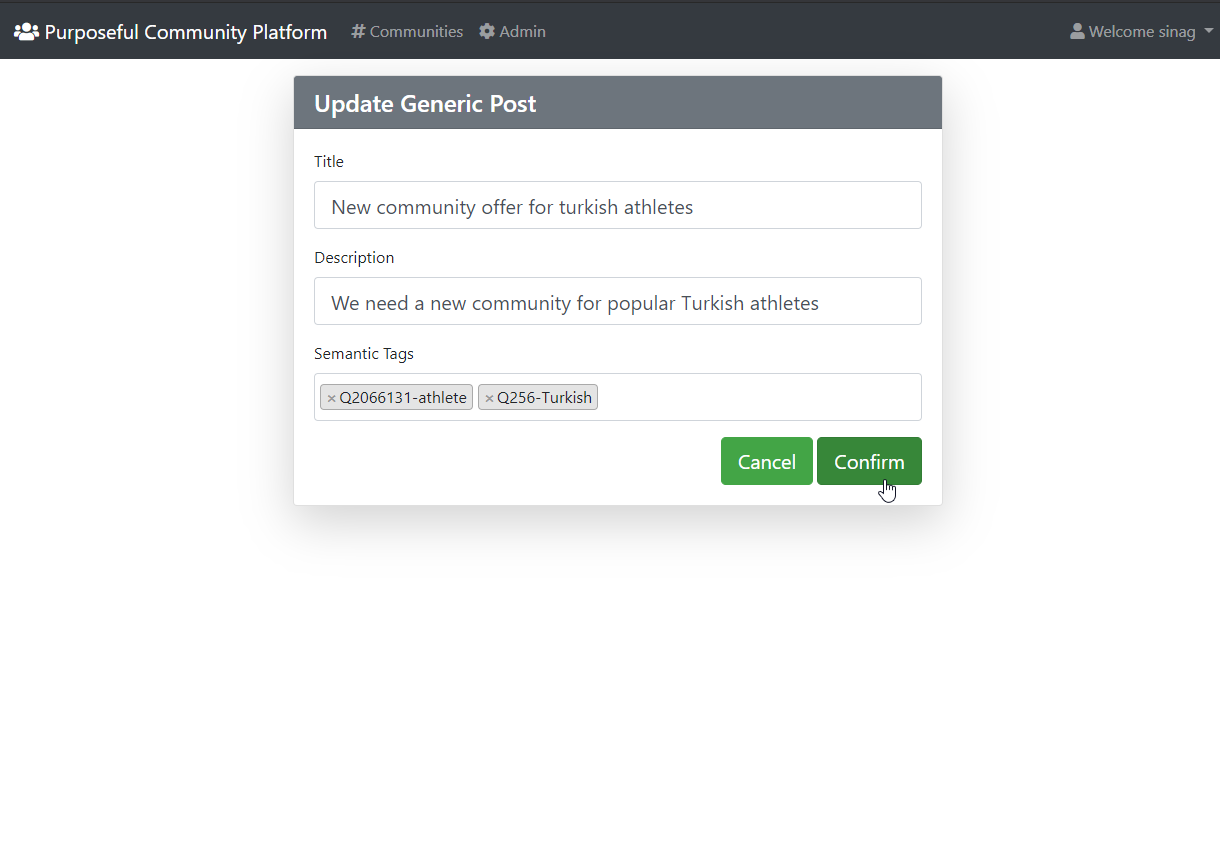
* 4.1.20 - Update Post

### Steps

#### 1) As the owner of the post, click "Update Post" when browsing the community which hosts the post to be updated



#### 2) Modify the values as required and click "Confirm" to save changes



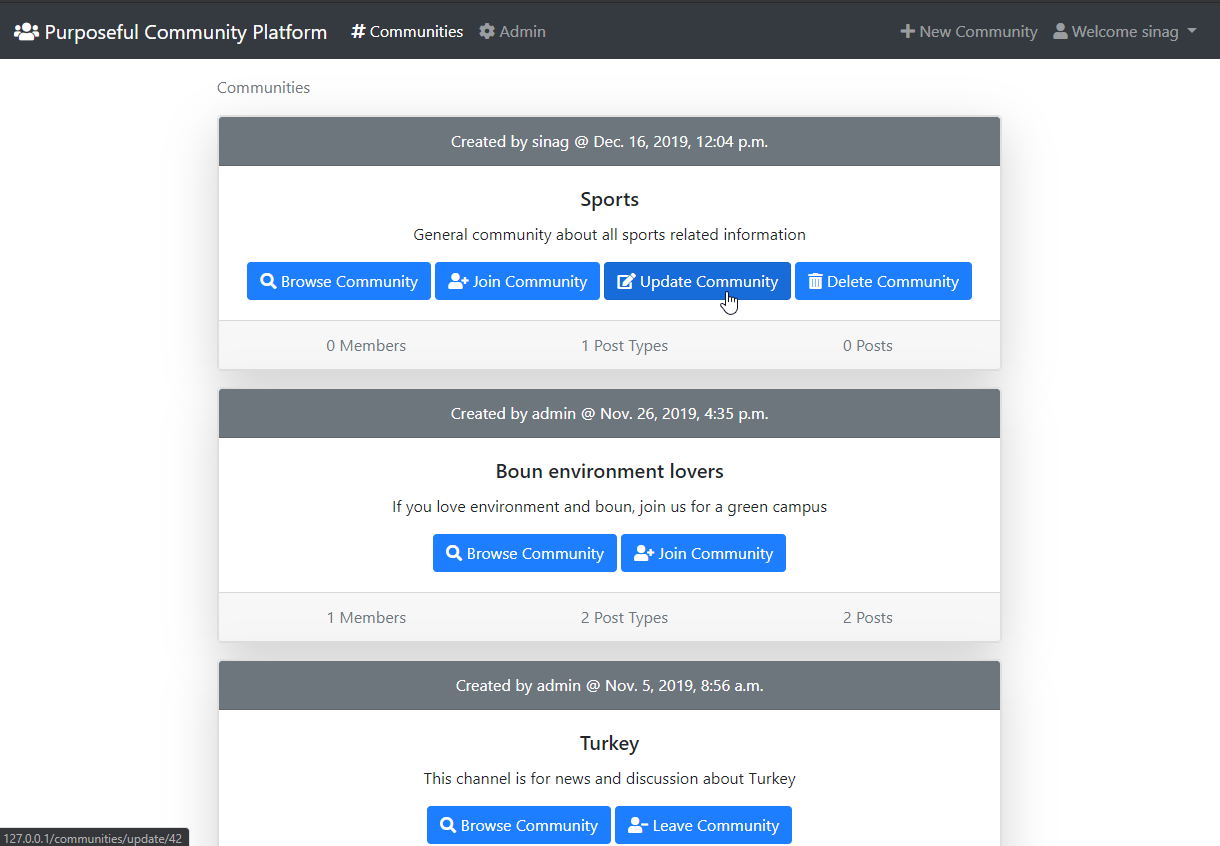
## 14.12 Update community

### Related functional requirements

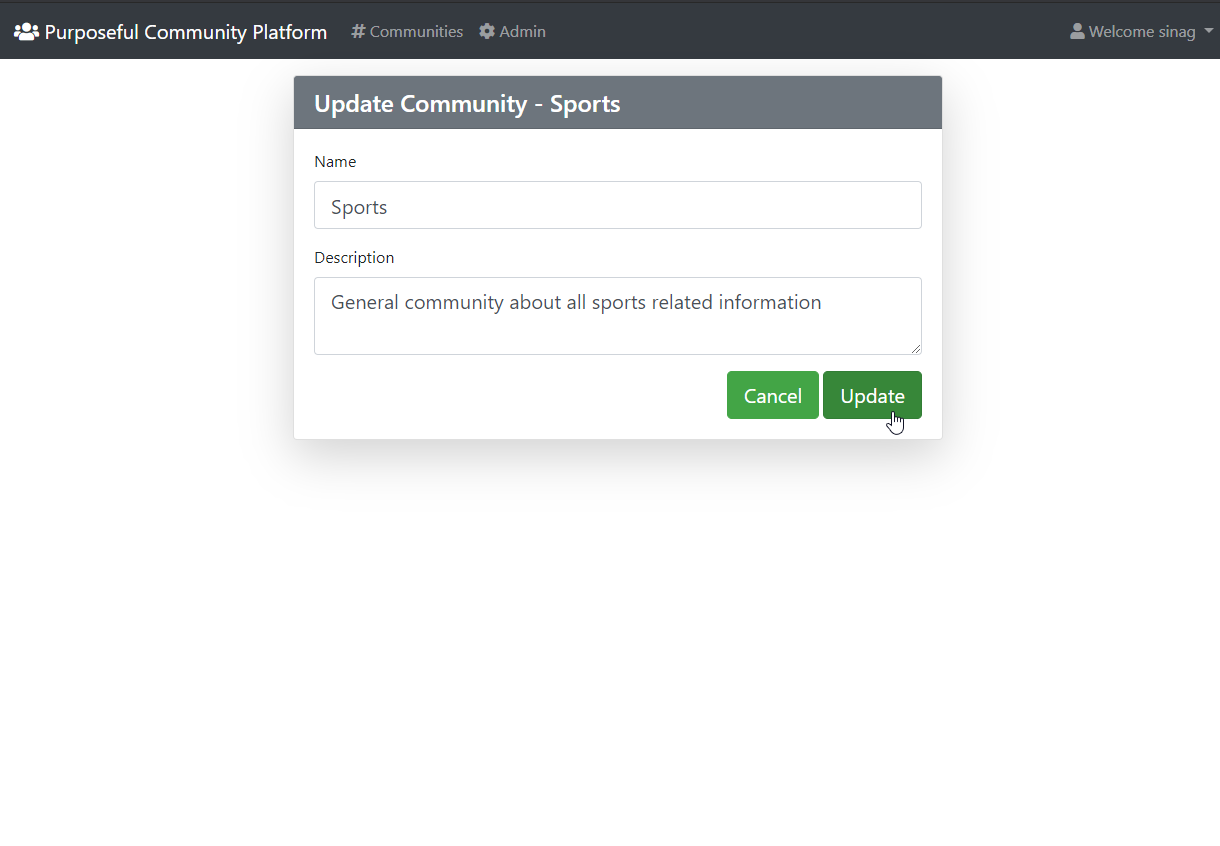
* 4.1.19 - Update Community

### Steps

#### 1) As the owner of the community, click "Update Community" when browsing the community to be updated



#### 2) Modify the values as required and click "Update" to save changes



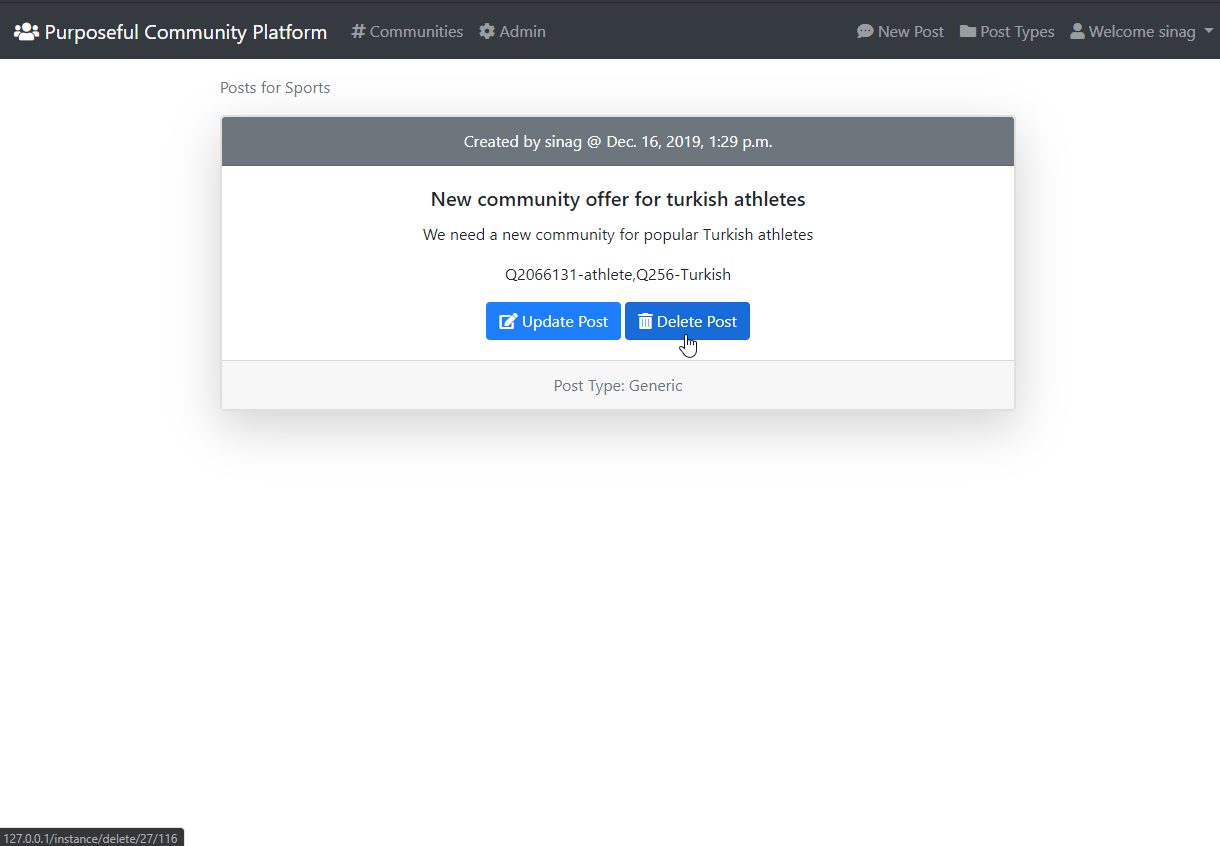
## 14.13 Delete post

### Related functional requirements

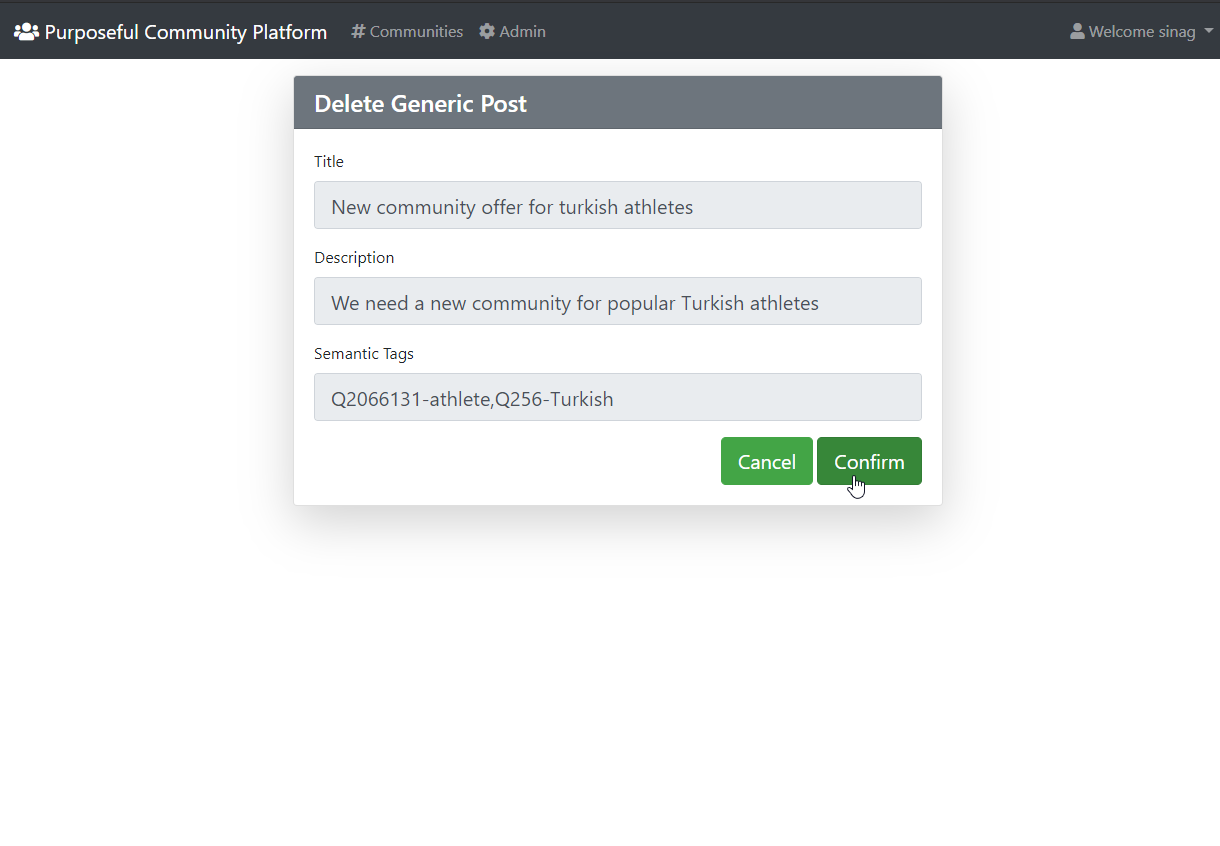
* 4.1.21 - Delete Post

### Steps

#### 1) As the owner of the post, click "Delete Post" when browsing the community which hosts the post to be deleted



#### 2) Click "Confirm" to delete post



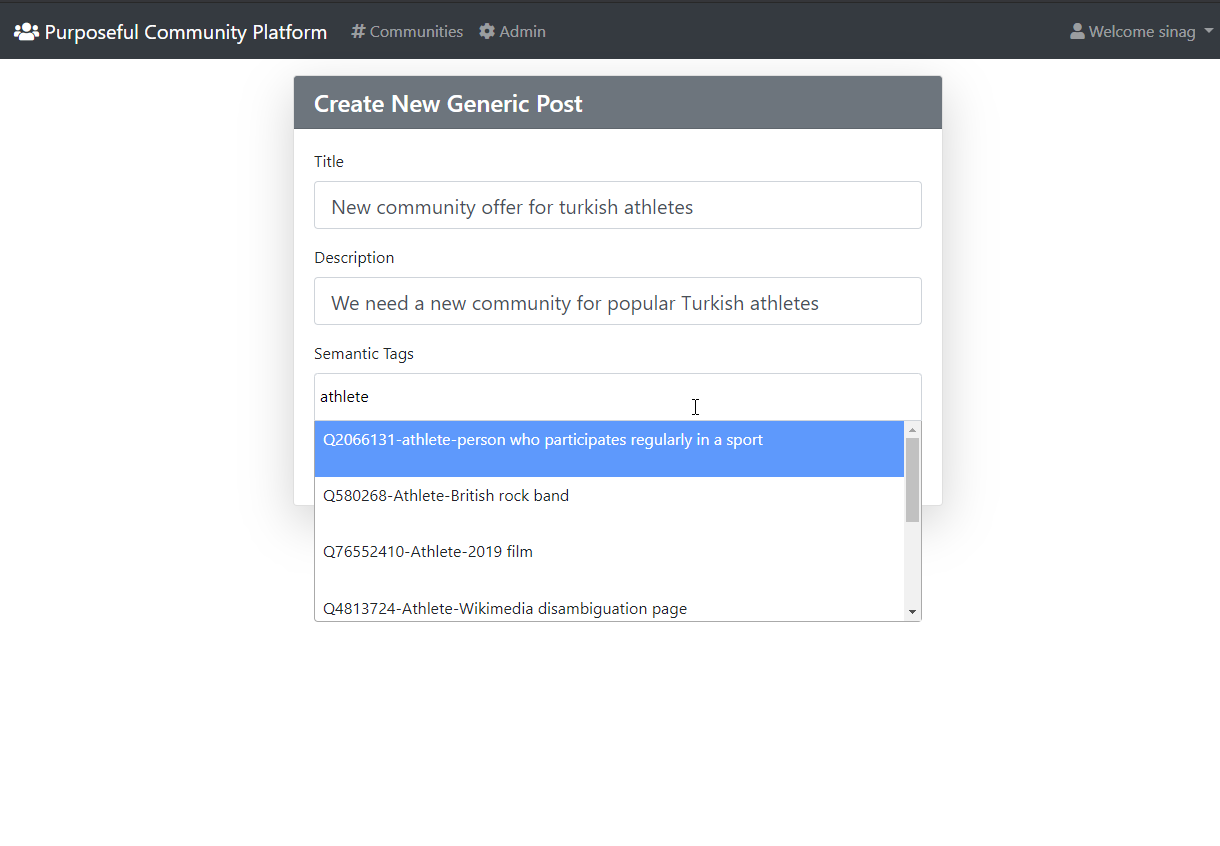
## 14.14 WikiData Tags

### Related functional requirements

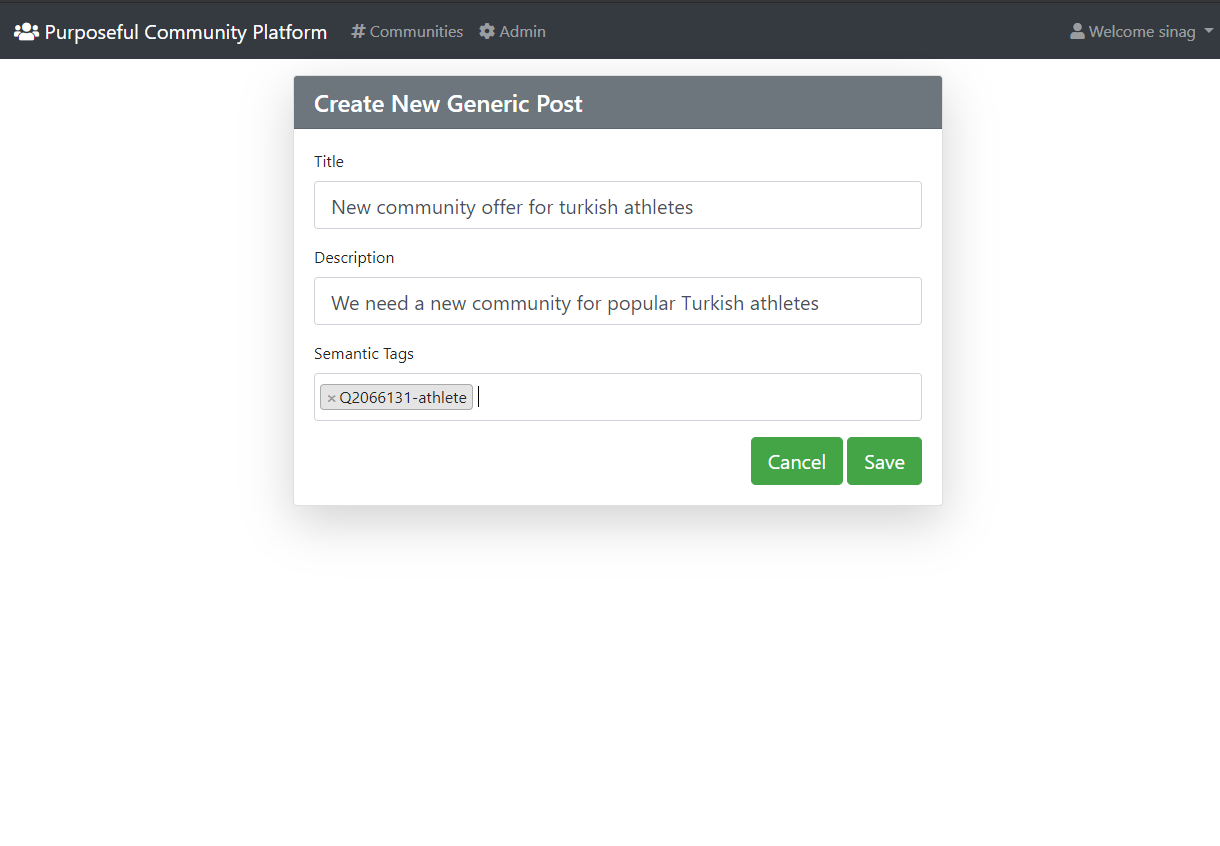
* 4.1.9 - WikiData Tags

### Steps

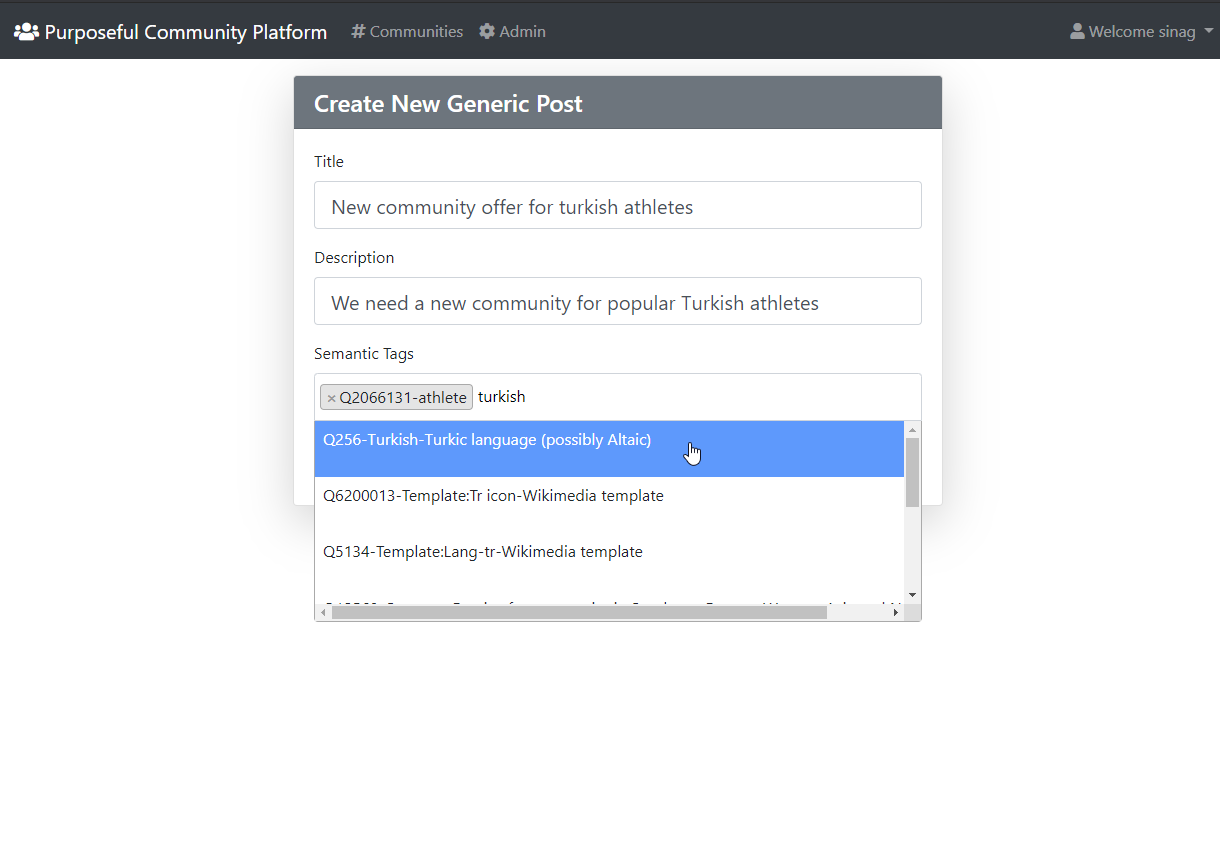
#### 1) When creating/updating posts "Semantic Tags" field has a special user interface to simplify tag selection. Type at least 3 characters in "Semantic Tags" field to query WikiData and show related results.



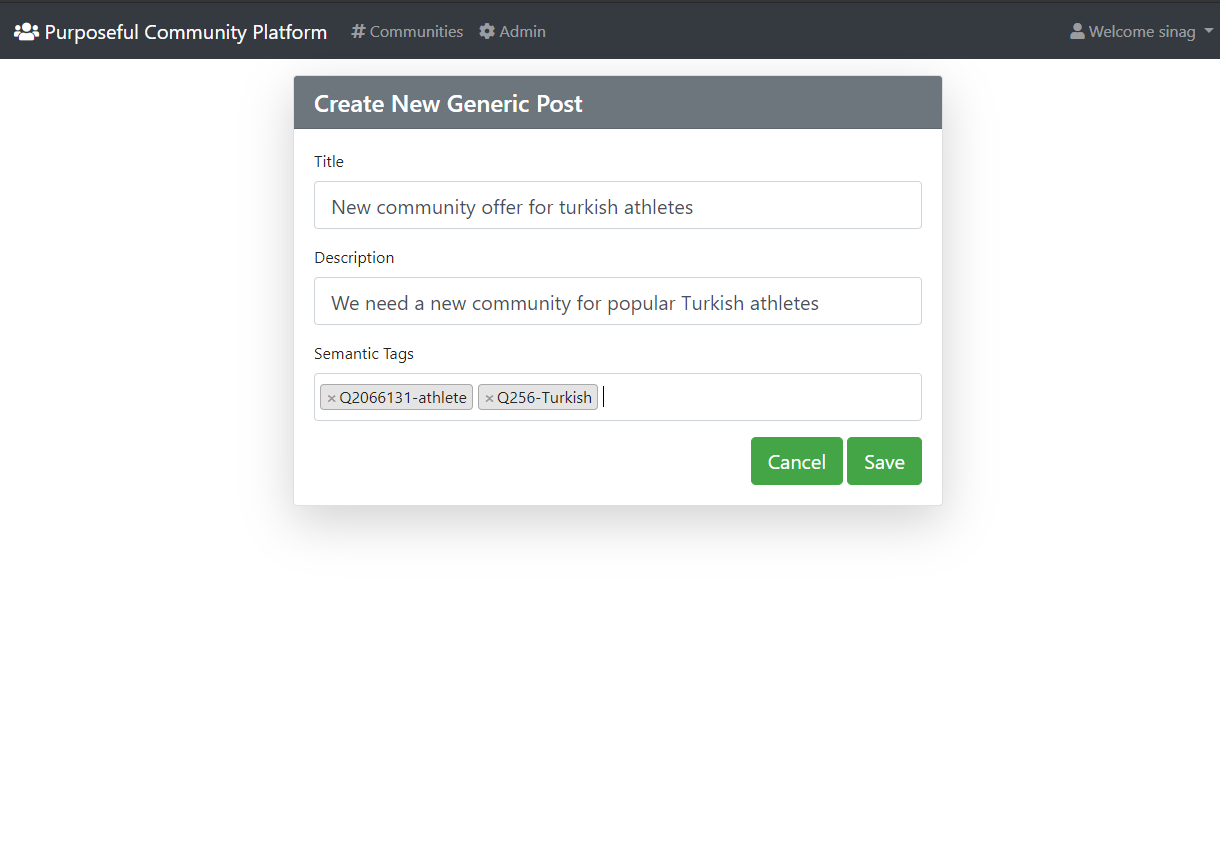
#### 2) Click item from list to add tag to validated tags



#### 3) "Semantic Tags" field supports multiple tags, make another query to fetch WikiData results



#### 4) Clicking the item from results list add the item to validated tags



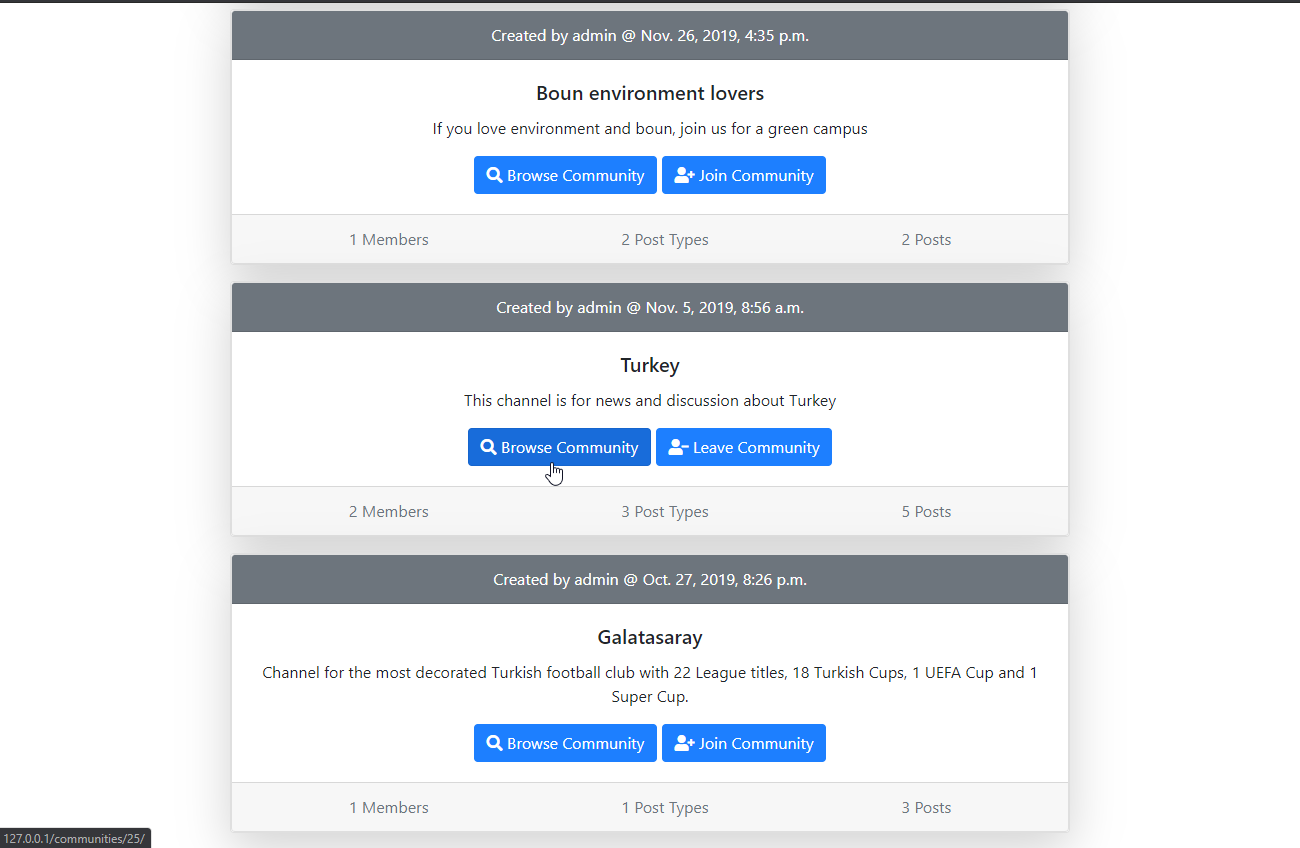
## 14.15 Search

### Related functional requirements

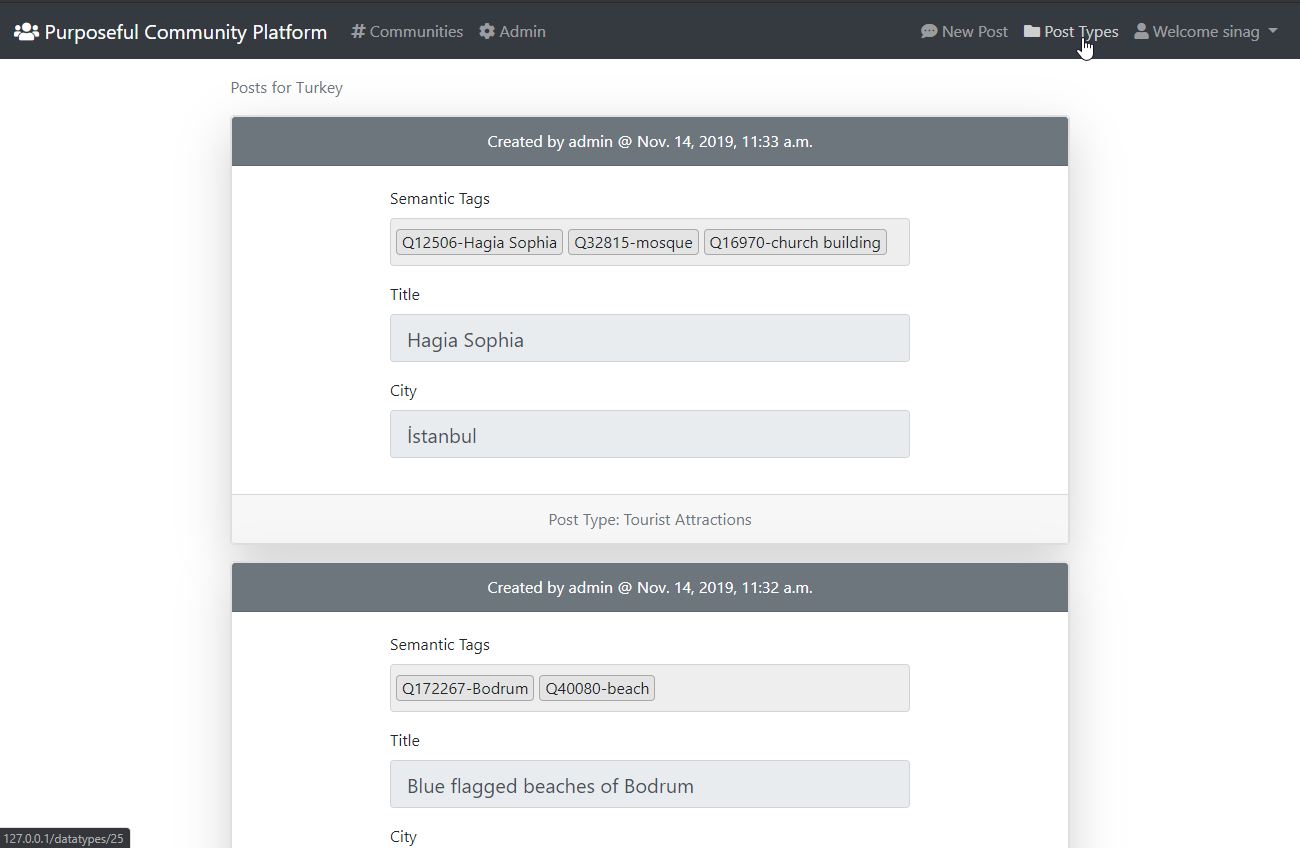
* 4.1.11 - Filtered Search

### Steps

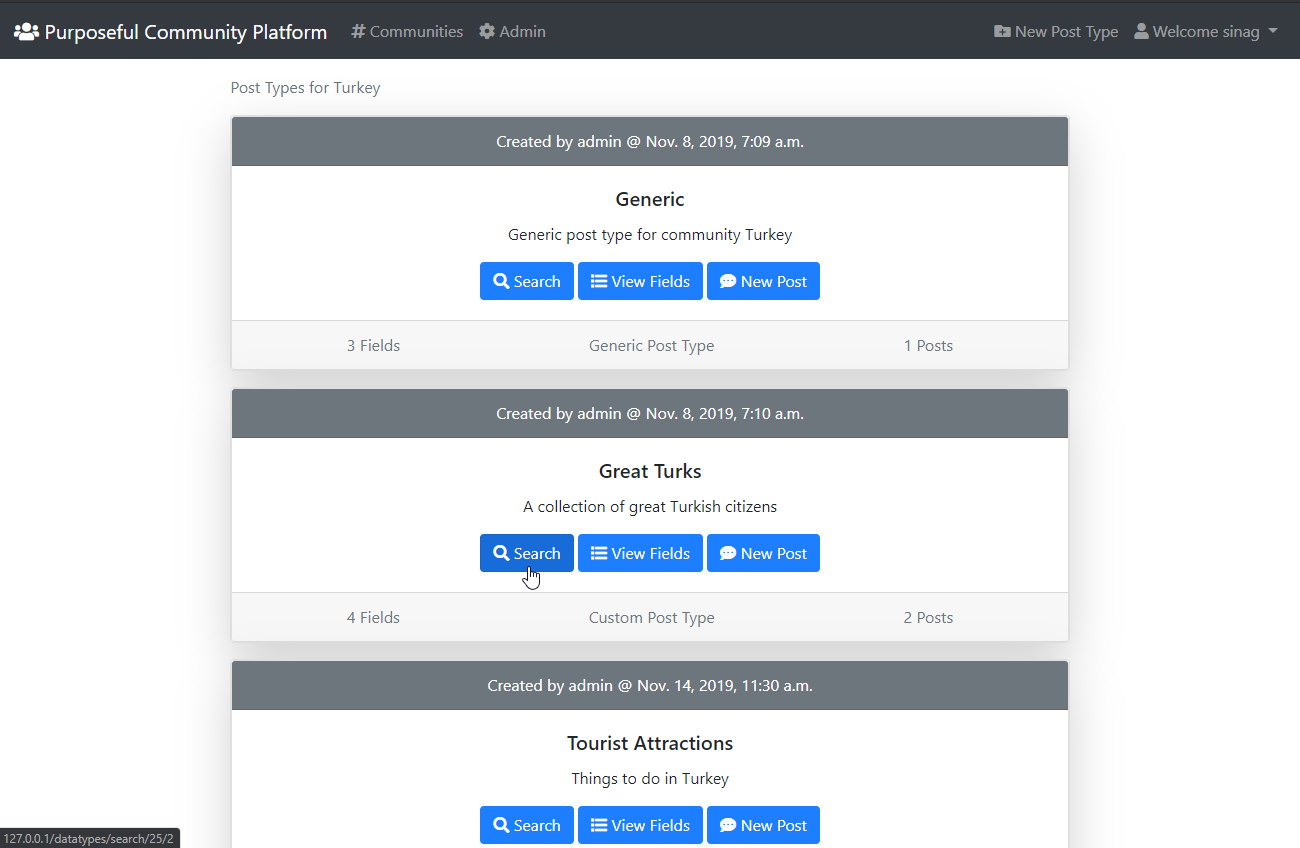
#### 1) Browse to the community which has the post type where filtered search will be run



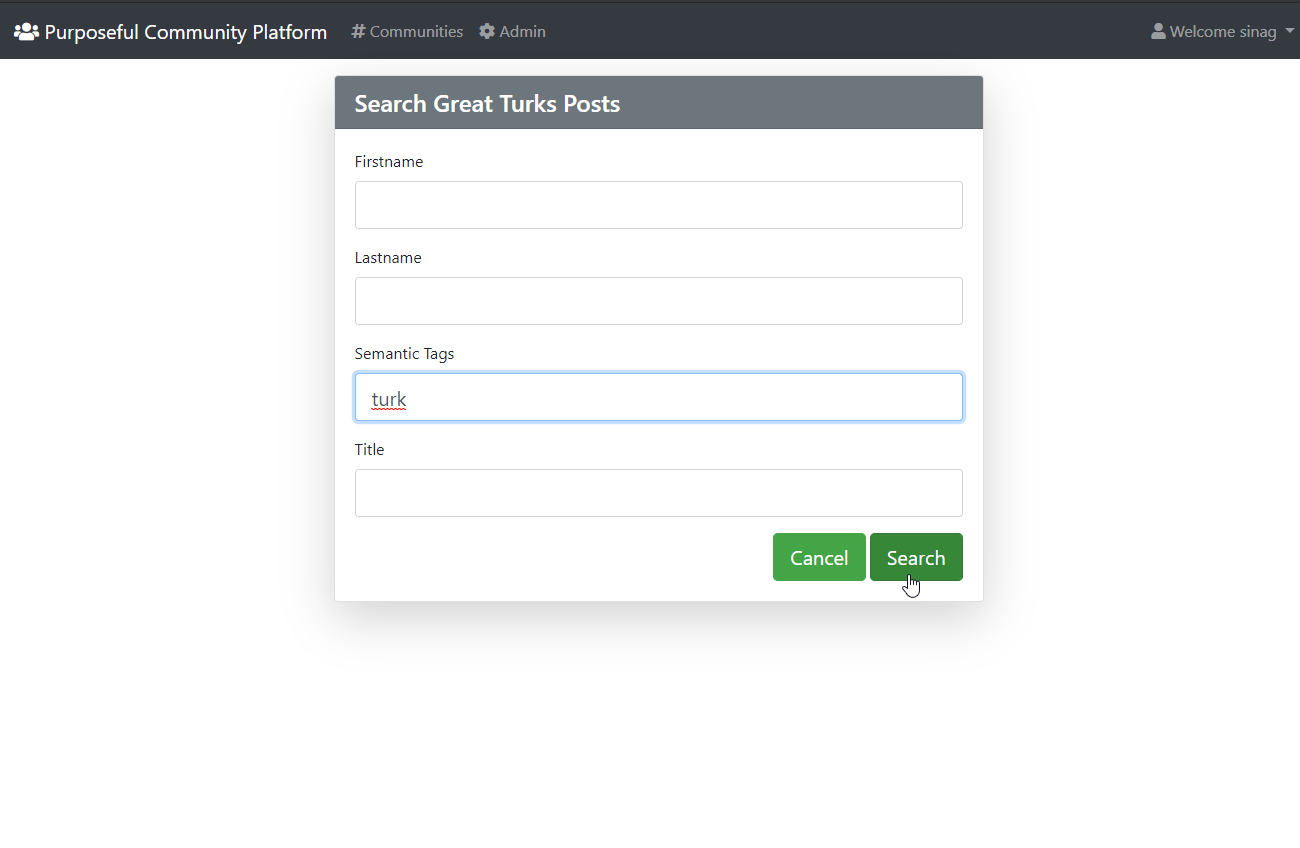
#### 2) Click "Post Types" button on navigation bar to view list of post types for this community



#### 3) Click "Search" button on the post type where the filtered search will be run



#### 4) Enter one or more filter values for post type and click "Search" to execute query



#### 5) Matching results for given filter will be listed

