Introduction

This is the technical documentation and the installation guide of the clan system. We will discuss the design decisions and will go through each requirement.

Installation Guide

Database Setup Instructions

Tools

I have used MongoDB 5.0 which was the latest at the time of development.

Instructions

First, create an empty database named “clan_db” by running the following command:

```
use clan_db
```

Now, let’s add a user just to make sure the database is created.

```
db.Users.insert({"UserName":"admin"})
```

Now the database is ready. Please note that in my situation I didn’t change the default port **27017**.

Application Solution Installation Guide

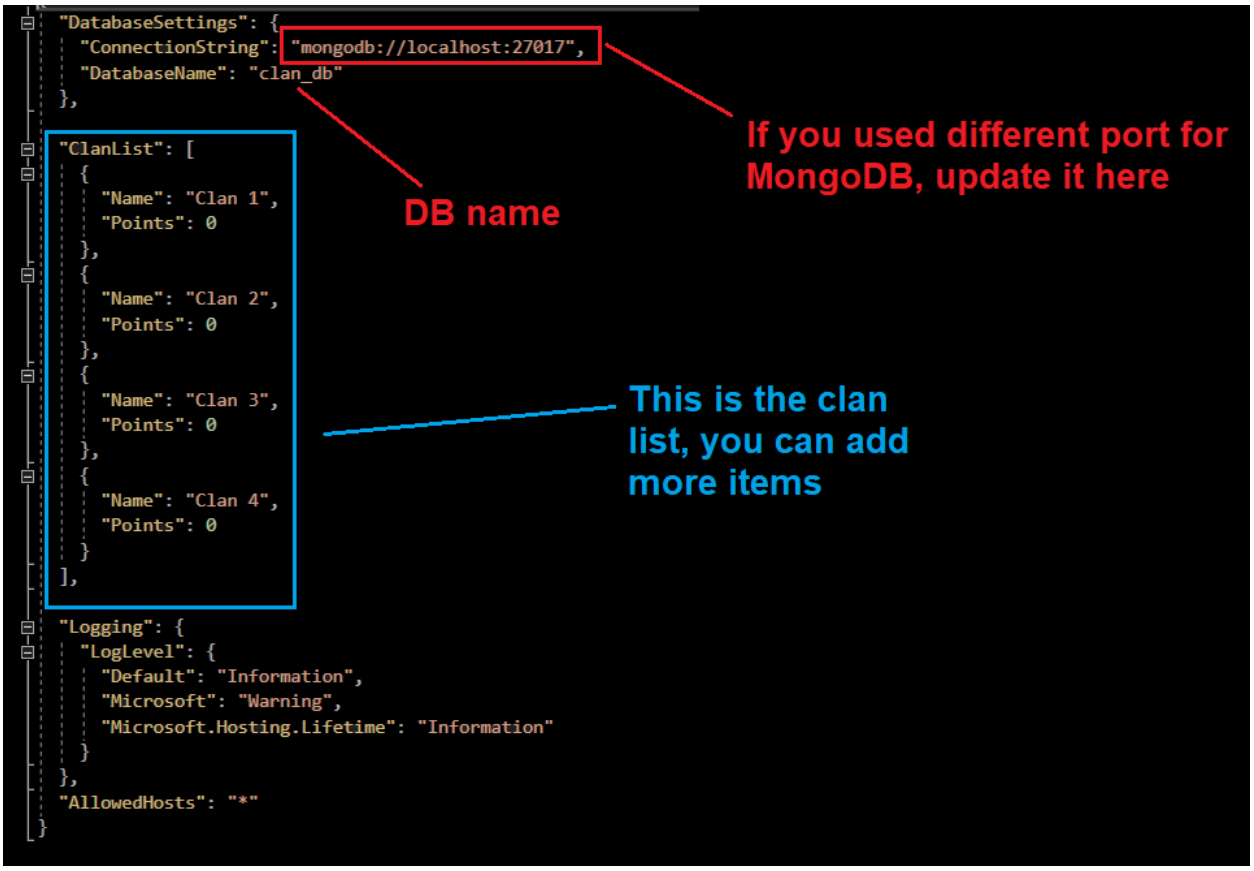
Tools

I have used Visual Studio 2019 (build 16.11.17) as an IDE and I have used ASP.net Core 3.1 (LTS) framework to build the solution.

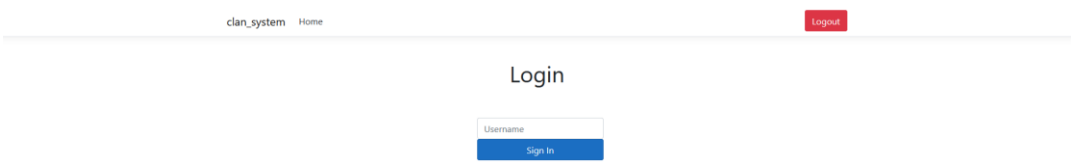
Instructions

After opening the solution, please navigate to appsetttings.json.

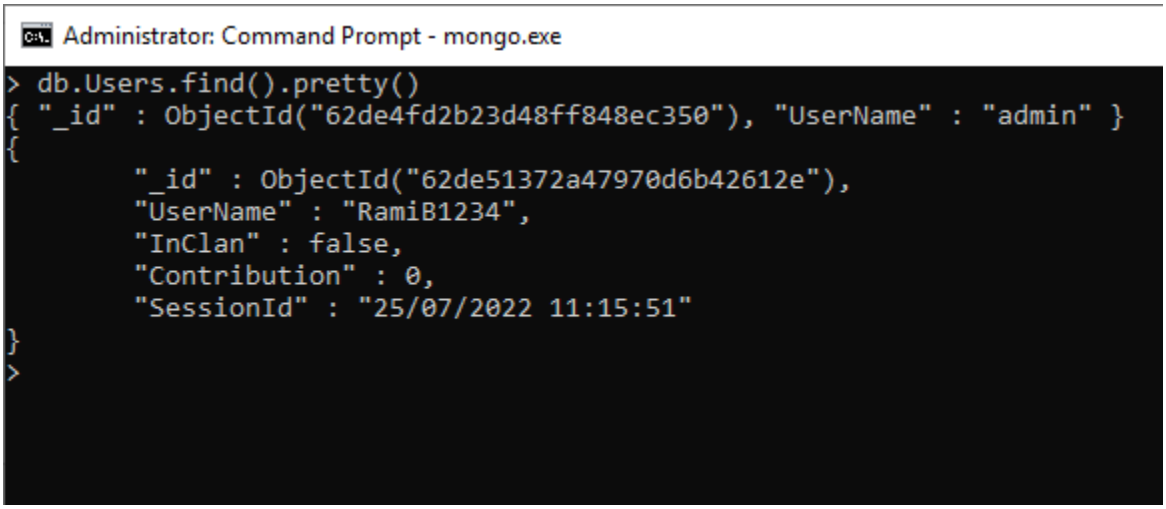
If you need to update the database name, default port or the clan list. Please check the following screenshot.



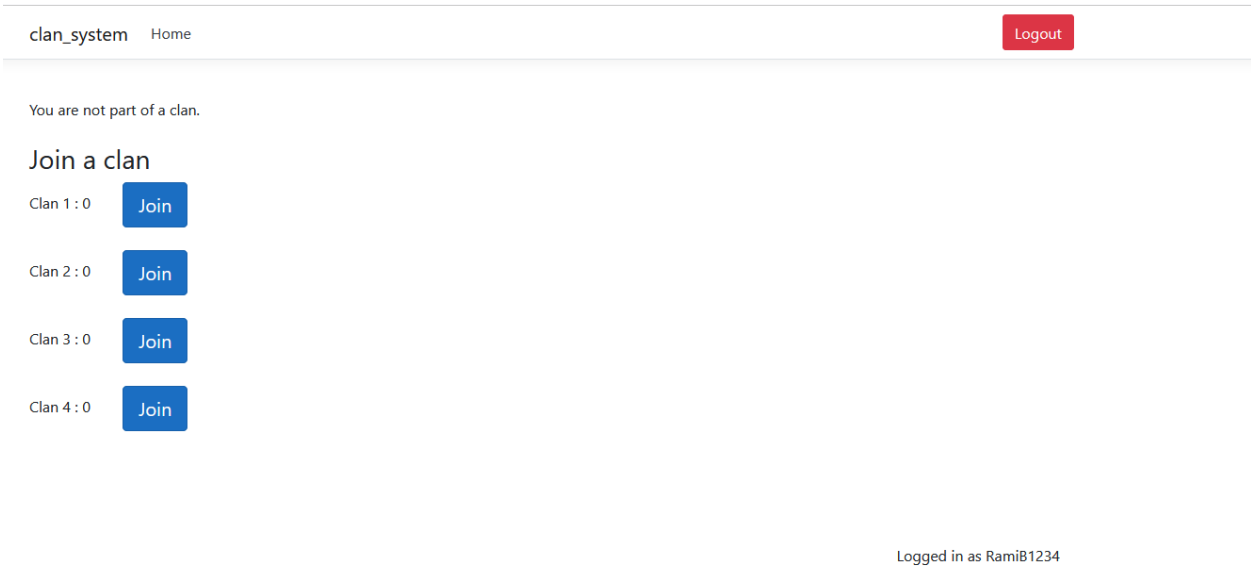
After making necessary changes, press (CTRL+F5) to run the application without debugging. You'll be greeted by the login page as shown below:



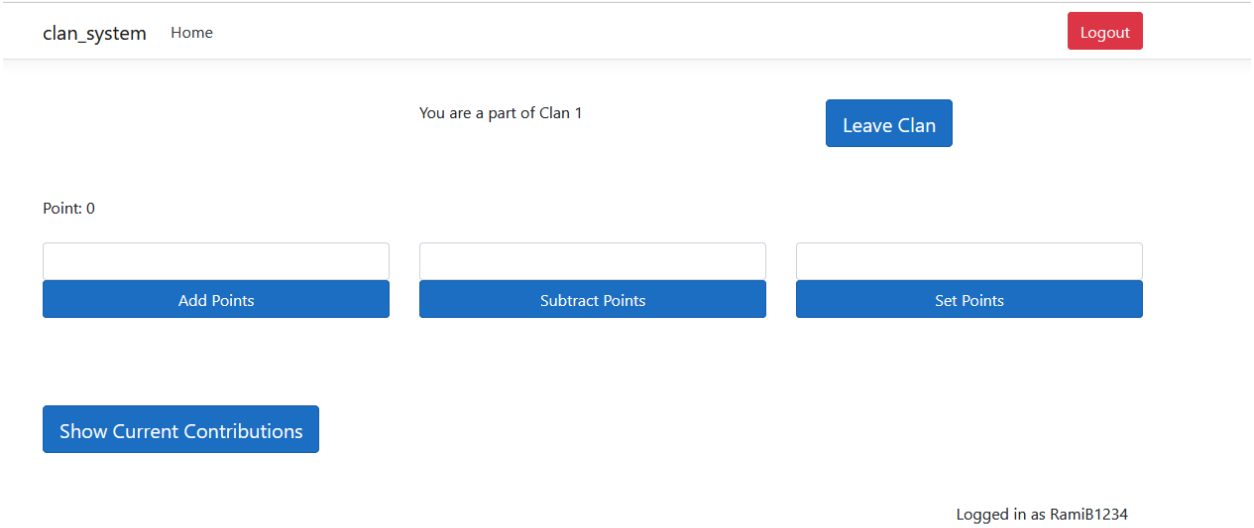
Now login with a username. Please note that if the username is not yet created, it'll be created after login in the Users collections as shown below:



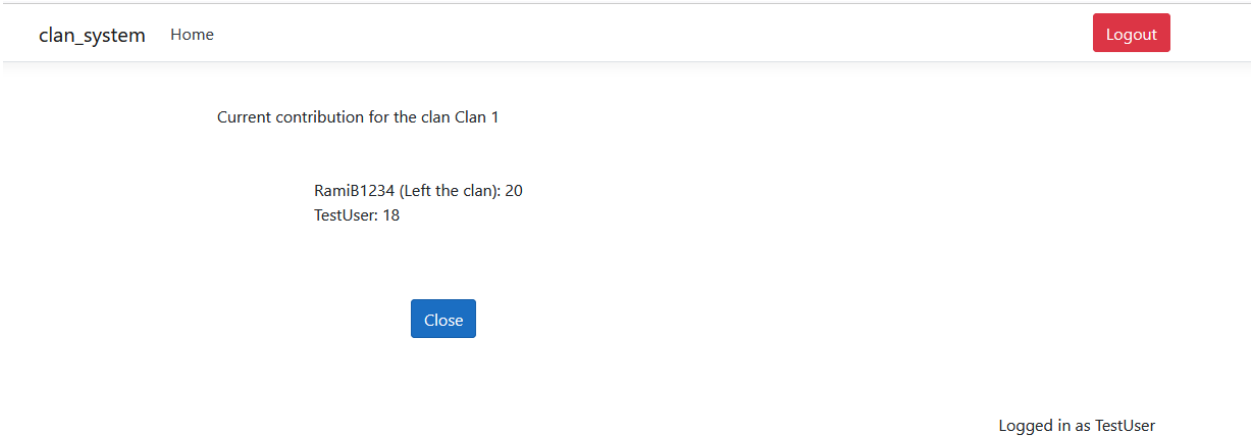
After you log in, you will see the landing page for users who are not yet in a clan, where you can click on any "Join" button, assuming that the clan capacity is not yet full.



After joining any clan, you'll be redirected the landing page for users who already joined a clan as shown below:

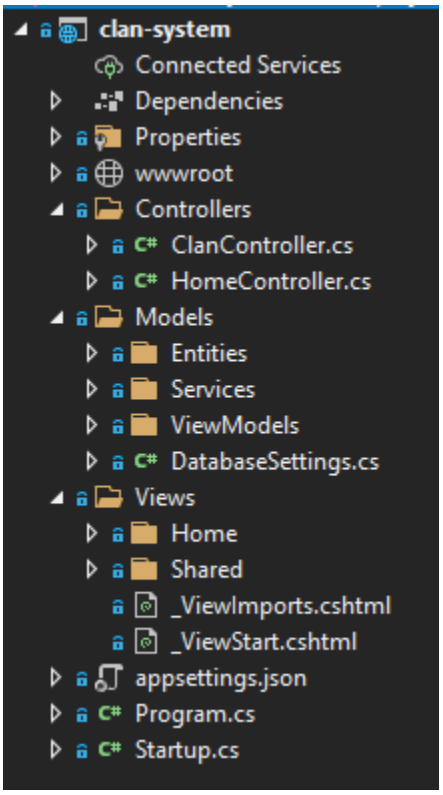


In this page, the user can add, subtract, or set points of clan. The player can leave the clan or display the contributions by clicking on “Show current Contributions” as shown below:



Technical Documentation

Project Structure



We have developed the application using **Core 3.1 MVC framework**. These are the important project directories and their descriptions:

- **Controllers**: This folder contains the controllers. We have 2 controllers. HomeController that is responsible for rendering the pages and logging in and out. The other controller is the ClanController that contains the APIs to perform clan related operations.
- **Models**: This folder contains 3 sub directories as follows:
    - **Entities**: Where all entities that are mapped to database collections reside
    - **Services**: They are responsible for communicating with our data source (in our case the mongoDB). You can think of it as the data access layer
    - **ViewModels**: That contains few complex classes that need to be passed to the view
- **Views**: Pretty self-explanatory. These are the views that will be rendered to the client.
- **appsettings.json:** This JSON file contains several important options that can be edited, as the db name, port and the clan list that will be migrated to the database at runtime.
- **Startup**: This class has several configurations such as session time, dependency injection mapping, and route configurations.

Requirement Checklist

Below are the set of requirements for developing the clan system and the status of each one:

Screen	Requirement	Status	Remarks
Login	A player enters the username, and on clicking on login, the player would be logged in	**Done**	
	If another player with the same name is already logged in, the old user will be logged out automatically and receive an error "You just logged on from a different device."	**Done**	The old user is logged out but due to time constraint he/she doesn't receive an error message
Landing / No Clan	If a user is not a part of any clan, the user will see the above page	**Done**	
	A user will see their name on the bottom	**Done**	
	The user will see the list of created clans	**Done**	
	The user should be able to click on the join button to join a clan	**Done**	
	The clan list can be retrieved from the config file	**Done**	To modify the list of clans, navigate to appsettings.json in the root of the solution
	Each Clan can only hold 10 users	**Done**	
Landing / No Clan	This page will be shown when a player is already a part of the Clan	**Done**	
	Every Clan has its points, and it's shown as "Points."	**Done**	
	Each clan member can add, remove, and set clan points	**Done**	
	A user can enter the number of points in the text input and then click on the respective button to execute the action	**Done**	
	When clicking on "Show Current Contributions," you should see the clan members' current contributions	**Done**	
	The "Set Points" action will clear the contributions list and set the current points as the contribution for the logged-in user	**Done**	
	Users can click on leave clan, and they will leave the Clan; you don't have to clear the contribution for the player who already left	**Done**	
Contribution	The contribution screen will contain the list of players and the points they contributed	**Done**	
	If a user contributed points and left, his name will contain the "Left the clan" suffix	**Done**	

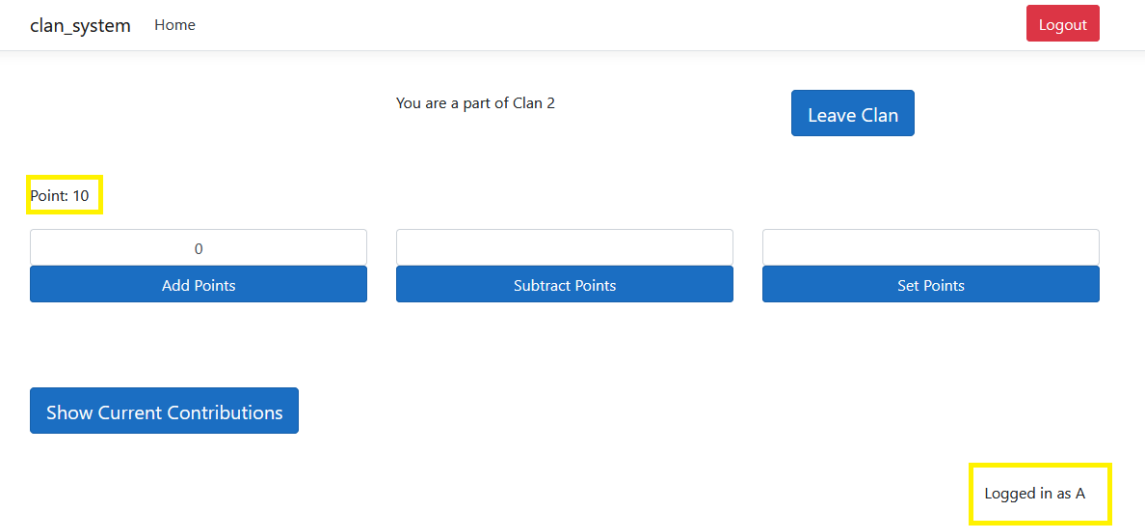
Race Condition Test

We'll test a simple race condition scenario. These are the steps taken to simulate the scenario:

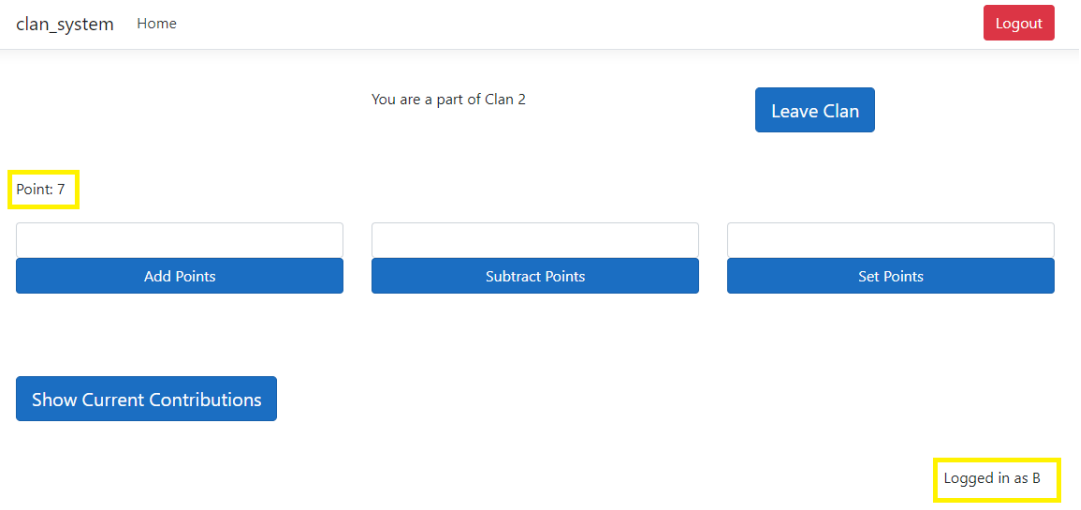
- User A login and goes to a clan page
- User A adds some points
- User B login and goes to the same clan page
- User B subtracts some points
- User A adds some points

The expected result is that user A must see the correct points after making an update, it should not conflict with the updated points made by user B

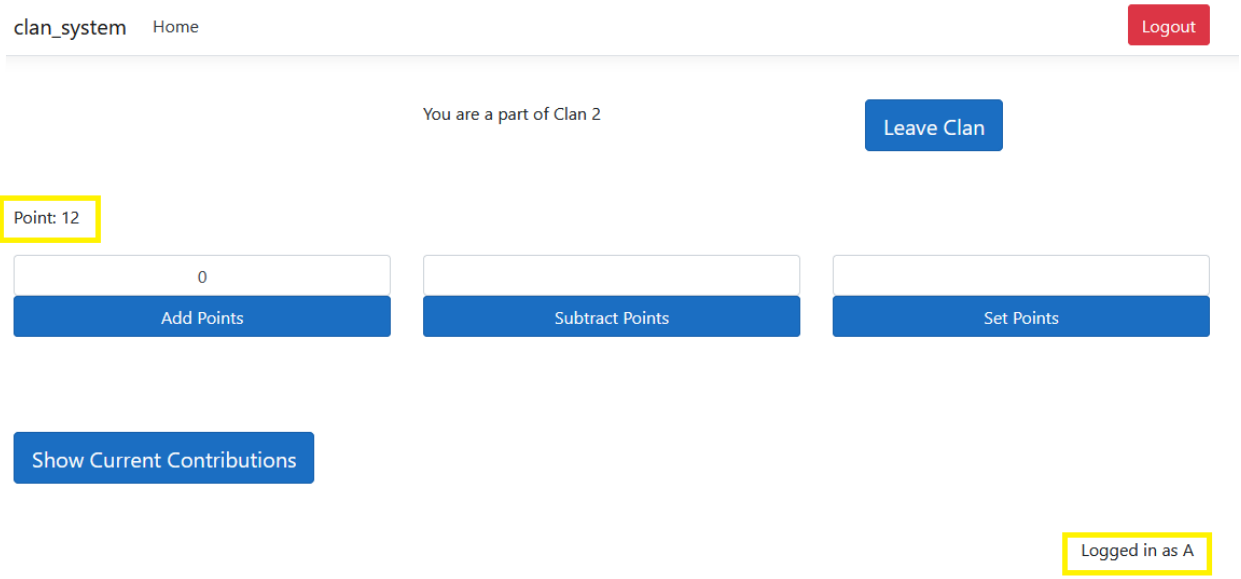
This is what happens after adding 10 points by user A:



Then, user B subtracts 3 points



Finally, this is what happens after user A adds 5 points:



This is the expected behavior, User A makes 2 actions but if there's another action in the middle, the second action must show correct result.

Known Issues and Bugs

Due to time constraints, there are few issues that were left unresolved. I would have fix them if I have more time. These are some of the found issues:

- The logout button should not appear in login page
- There's no validation that ensures entering correct information or any information at all. Ideally, the user should not be able to perform actions if fields are empty or having wrong data type
- If the user is logged in from another session, old session automatically logs out but there's no error message that appears as asked in the requirement. It will be silently logged out and redirected to the login page again.

Conclusion

I have enjoyed working on this small exercise. I never worked with MongoDB or any other NoSQL databases before, so this was a good learning experience. I like to always get out of the comfort zone and learn new things.

There was a lot of room of improvement. However, I'm kind of satisfied with what I could done in this short time.