MILESTONE REPORT 1

PROJECT DEVELOPMENT IN SOFTWARE ENGINEERING

Group Members

Berke Argın
Emrah Doğan
Muhammed Göktepe
Emir Hıfsı Öztoprak
Bilal Tekin
Elif Akalın
Yunus Emre Topal
Orkan Akısu

CMPE451
Fall 2021

1.Executive Summary

Project Description

We are designing a social media platform called Protopost that enables its users to create and share information in a community by creating posts in specific formats. Each user can create or join a community of their liking and create posts in some predetermined post format of their community. These predetermined post formats, which are called post templates, are created by community moderators and used in categorizing posts in that community. Each post template contains several compulsory or optional data fields to be filled, which can be of many types including location, image, video, text, date or a selection. Users can also filter posts based on their post templates and use queries on several data fields on that post template. Post template feature aims to provide an effective and accessible information sharing by providing a well-defined and characteristic outline for each piece of information in that community. Each post must belong to a community and put into that community's post feed. Users can also add other users as a friend after mutual consent and see their friends posts in their own user feed. Communities can be public or private, which in turn, will also define the privacy setting of the posts in that community. Currently, we aim to develop this platform both as a web and mobile application.

In summary, we aim to create a digital platform that enables its users to be a part of a purposeful community with each having their own specialized language for sharing information. In our platform, each community can define distinct and well-defined blocks of information which paves way for users to create specialized communities around a purpose.

Project Status

As we decided on our core requirements and created some design documents for our project, we’ve implemented and realized some of these requirements based on the design documents for Milestone 1. While there are some issues regarding the integration of Backend API to Frontend services, we think that these setbacks are only temporary.

As the backend team, we have implemented most of the core API functionalities such as creating communities, posts and post templates as well as retrieving these data structures. We also implemented a registration system that utilizes the built-in Django authentication system. We also launched an AWS instance and deployed our API. We also included a Swagger page as a dynamic documentation for the Backend API which can be reached here. Currently, we have some issues regarding authentication cookies and CSRF tokens which were detected late and we did not have enough time to fix them. Also, our endpoints are not configured with respect to JSON-LD and Activity Streams standards.

As the frontend team, we have implemented Register and Login pages, Create Community, Create Post, Create Post Template, My Post, My Communities, Communities and Feed Page. Due to some problems in backend connection Create Post, Create Post Template Page and Create Community pages do not respond.

Moving Forward

Our primary objective is to fix integration problems between Backend API and Frontend services. After that we will be looking to implement features on privacies, commenting, interactions between users, and extensive filtering and query mechanisms.

As the backend team, we first aim to address issue regarding authentication and CSRF tokens by configuring our current project for token-based authorization instead of the default session based system. We also deploy our API service as HTTPS, by getting a SSL certificate. We are also planning to write a CI/CD pipeline that includes automatic testing, dockerization, and AWS deployment. As for implementing future features such as privacy measures and advanced filtering, we expect a smooth development process now since we have the core system to built upon.

As the mobile team, we have some features we couldn't manage to implement in time. They are post creating, post template creating and community creating. We will start with working on them first. After that, we will focus on some additional features that are missing on our screens such as: searching communities, searching posts, additional setting options. Also, we know that our applications user interface can use some polishing to look better. We will create better styles to make our application beautiful.

As the frontend team, we implemented most of the pages but there are some problems in backend connections due to CSRF token and authentication cookies. We will try to solve those problems as soon as possible. Then we will start to implement new features such as search and filtering, profile page and settings. We will improve our styling to provide a better view for our users.

2.List and Status of Deliverables

Deliverable	Status	Frequency	Description
Github Wiki Page	In Progress	Updated as needed	Documentation for the project and up-to-date information on team members, weekly meeting notes, weekly assessments
Github Issues	In Progress	Daily	Current and past issues about the project, with helpful and colorful labels.
Meeting Notes	In progress	Weekly	A summary of the discussions and decisions of the weekly scheduled meeting. A table of the division of weekly assignments can be seen at the end of every page.
Requirements	Complete	Completed	Requirements to be satisfied by the project. This page also includes a glossary and comparison of our project with other similar social networking websites.,
User Scenarios & Mockups	Complete	Completed	This page includes three scenarios that showcase the several uses of our project. A mock ui for both android and web, as well as detailed scenarios can be found here.
Design Diagrams	Complete	If Needed	These diagrams include a use case diagram, a class diagram and a sequence diagram.
Milestone Report	Complete	Completed	This document.
Backend API Django Project	Partially complete	Weekly	Django project that contains our code, configurations and logic behind our backend API.
Deployed Backend API Service	Partially complete	Weekly	Our backend API service which it's Swagger page can be reached on http://3.249.82.166:8000/api/schema/swagger-ui/

Mobile Application	Partially complete	Weekly	Mobile part of our application. You can find the latest version of it in our github pages mobile branch.
Web Application	Partially complete	Weekly	Web part of our application. You can find the latest version of it in our github pages Web_ReactJS branch.

3.Evaluation of the Status of Deliverables

- **Github Wiki Page:** The wiki page is continuously updated. A summary of our scheduled meetings are added every week. Pages such as “requirements”, “scenarios” and “design documents” are added according to the type of our weekly assignments. This pages provides an overall view of our current work and our progress in a compact manner.
- **Github Issues:** All open and closed issues regarding our project can be found here. The progression of our weekly assignments can be seen here. Custom labels such as “help needed” and “review needed” are used to increase productivity and alert other team members to a member’s progress in an assignment.
- **Meeting Notes:** Meeting notes are taken every week during scheduled meetings. They are then assembled by a team member as a summary in markdown language and uploaded on the wiki.
- **Requirements:** The requirements elicitation of the project is completed and can be easily accessed using the sidebar of the wiki.
- **User Scenarios and Mockups:** A mockup of the web and android UIs as well as example use scenarios can be found here. They can be accessed using the sidebar on the wiki.
- **Design Diagrams:** Use case, class and sequence diagrams have been created for the project. They are more “first draft” than a “final version”, and some changes might be required as the project proceeds.
- **Milestone Report:** This report.
- **Backend API Django Project:** We created and pushed the Django project that contains implementations for our backend API. Script in the Backend-Emrah-Deployment branch currently runs in our AWS instance. It currently contains core functionalities of our application including community, post and post template creation, subscription, and post retrieval.
- **Deployed Backend API Service:** We deployed our backend service to the AWS instance. It’s swagger page can be reached here. Currently, it contains some token issues that creates some complications during Frontend communication which will be fixed as soon as possible.
- **Mobile Application:** We connected our mobile application to our backend server. Users can Register and Login to our application. When logged in, they can view all communities, their subscribed communities, their feed and posts created by them. They can click on posts to see their details and they can click on communities to see community pages. They can view community posts, subscribe to or leave from a community. Also, they can logout from the application. But there are some issues on the backend related to registration and community subscription/unsubscription functionalities. Once they are fixed, they will work as intended.
- **Web Application:** We connected our web application to the backend server. Users can register and login to our application. When logged in they will be redirected to the feed page and will see the posts of the communities they have subscribed to. There is a sidebar at the left side of the screen. From there they can go to the Communities page where they can see all the communities that have been created. Also there is Create Post Page to create posts in a community. To create a post they need to create a post template first so there is a Post Template Create page to fulfil this requirement. Users can see his posts by clicking the My Posts label in the sidebar. Also users can create Communities by clicking Create Community label. Finally, users can log out from the app.

4.Evaluation of Tools and Processes Used

- **Github:** Github is the main platform where our project will be developed. We have only used the wiki and issue management features of github so far. It will be incredibly useful once we start coding, since it will let us keep record of our progress, and give the option to branch and/or merge as we try and add more features to our code base.
- **Slack:** We use slack due to its tight integration with other platforms. Channels and integration are the main features that make slack invaluable for us.
- **Zoom:** We use zoom for our weekly meetings, since everyone has zoom installed and configured on their computers for the lectures.
- **Discord:** We use discord for asynchronous communication. In addition to the channels, voice chat and livestream capabilities, we also use discord to communicate with our TA.
- **Moqups:** This tool was used to create the mockup UI for the android version of our project. Moqups is a streamlined web app that helps create and collaborate in real-time on mockups and diagrams.
- **Lucidchart:** Lucidchart was used for the use case, class and sequence diagrams. It allows multiple team members to work on the same diagram simultaneously, and offers UML support.
- **Google Docs:** Google docs was used to create this document. Similar to lucidchart, it allows multiple team members to work on the same document.
- **ProjectLibre:** We used the project management software ProjectLibre to produce a Gantt chart of our team effort until now. ProjectLibre is an open-source project management software that supports many features like Gantt charts, network diagrams and PERT graphs.

Backend:

- **Django Framework:** Django Framework is one the most popular python web framework that is built on developing web applications with Model-View-Template software pattern. It is a very flexible tool with many documentations and a vast library. We decided to use this framework as we had some experience with it from last semester and were largely satisfied with its functionalities.

- **Django REST framework**: Django REST Framework is the most widely used library for building Web APIs in Django Framework. It provided us an additional layer of abstraction over Django framework such as serializer classes that enabled us to manipulate and retrieve data with ease. It also provides built-in request and response wrappers that reduce boilerplate code and simplify managing API entrypoints and endpoints, which was quite useful for our project.

- **Docker:** Docker is a container based virtualization tool that enables developers to develop and deploy their applications on clean, isolated environments called containers. We decided that proper dockerization of our application would be vital together with a CI/CD pipeline.

- **Swagger:** Swagger is an open source automatic documentation tool for RESTful APIs. This documentation is readable not only for humans but also for machines. It is easy to use and help developers to understand API endpoints fast by providing supported operations, parameters and outputs, authorization requirements, licenses needed.

- **MySQL:** MySQL is an open source relational database management system. It uses SQL for query language and is most commonly used in Web Servers.

- **Amazon AWS:** AWS is a cloud computing provider that provides a combination of many systems such as data storage, data processing, content delivery services to meet the service needs of businesses.

Web Frontend:

- **create-react-app:** This is a very important framework which is for creating single page react applications.

- **React-Pro-Sidebar:** This is an npm package which is created for React applications. This can be installed via npm tools.

- **React-Bootstrap:** This is a very popular style framework which is used for creating a very good looking web screens.

- **React Hooks:** React hook APIs provide an alternative to writing class-based components, and offer an alternative approach to state management and lifecycle methods.

- **React Axios:** Axios is a library that serves to create HTTP requests that are present externally.

- **React Redux:** React Redux is the official React binding for Redux. It allows React components to read data from a Redux Store, and dispatch Actions to the Store to update data. Redux helps apps to scale by providing a sensible way to manage state through a unidirectional data flow model.

- **React Navigation:** This library is used for redirecting users to any screen with parameters without refreshing the webpage.

Android:

- **React Native:** React Native is a framework to develop mobile applications using React features as well as some other features developed for mobile. React Native is a powerful but hard tool to use. We struggled at the start but we learned how to use it effectively.
- **React Navigation:** React Navigation is a package developed for handling navigation inside the application. It has several different types of navigators such as Stack, Drawer, Tab. We used Stack Navigator for navigating between Welcome, Login, Registration Screens and we used Drawer to show 5 screens that a logged in user can access. It helped us a lot since navigation is an essential part for our application.
- **Expo Go:** Expo is a framework that lets us develop and deploy React applications much faster. Instead of running our application on a virtual device, we can just connect our phones to test our application and that accelerates the development time quite a lot. But while using that feature, we missed something. That is the difference between iOS and Android. Some stylings changed from iOS to Android on React and we tested our application on our Iphones which resulted 2 white buttons on our demo.
- **React Axios:** Axios is a package to make API calls and fetch the result from an

external website. Using it instead of the default fetch() function helps us to manage requests from a single page and with that, we don’t have to worry about changing the URL of our server. Because instead of fixing the URL in every request code, we can just change it in one place.

5.Summary of Work Done

Member Name	Contributions
Elif Akalın	● Reviewing and updating requirements page. ● Attending group meetings ● Creating an Axios client to use project-wide in mobile app ● Implementing AllCommunitiesScreen API call ● Implementing UserCommunitiesScreen API call ● Implementing CommunityScreen API call ● Implementing PostScreen API call ● Adding the logo to the mobile welcome screen ● Adding a button to the CommunityScreen that works as a toggle to join and leave a community ● Designing a pretty UI for the login and register screens ● Adding constraints for registering, such as password requirements ● Reviewing pull requests for mobile ● Creating pull requests for mobile ● Reviewing issues ● Testing and detecting bugs in the backend service
Berke Argın	● Reviewing and updating requirements page. ● Reviewing and updating Wiki Page for CMPE451. ● Creating a doodle for the weekly meeting schedule. ● Discussed on the project name and commented my suggestion on the discussion thread. ● Drawing the UI design sketch and adding it to wiki. ● Attending group meetings. ● Attending lab sessions. ● Writing the basic sketch for backend API. ● Initializing the Django project for backend API. ● Migrating and updating the existing model from the practice app to the new project. ● Writing initial functions for post, post template creation as well as login, logout and register. ● Reconfiguring code for Django REST framework. ● Writing serializer classes for the existing models. ● Implementing Login, Register API functions. ● Implementing CreatePost, CreatePostTemplate functions. ● Implementing GetCommunities, GetPosts, GetPostTemplates functions. ● Implementing UserSubscriptionStatus functions.

	● Writing unit tests for CreatePost and CreatePostTemplate. ● Dockerizing the Backend application. ● Adding environ library usage to protect secrets ● Integrating the code for auto-schema generation using drf_spectacular library ● Creating Swagger page for the Backend API documentation. ● Taking and uploading Backend Meeting Notes 1,2 ● Reviewing pull requests made by Emrah Doğan. ● Reconfiguring the project database to MySQL. ● Deploying the API to AWS. ● Fixing the bugs in the Backend service pointed out by the frontend team. ● Filling out the backend section of Project Status in executive summary. ● Filling out the backend section of Moving On in executive summary. ● Adding descriptions ● Filling out my part on the RAM. ● Writing tool evaluations for Django, Django REST framework and Docker. ● Filling out the roadmap section for the backend team.
Orkan Akısu	
Emrah Doğan	● Reviewing requirements page. ● Reviewing Wiki Page for CMPE451. ● Researching and Opening issues for determining determining Mobile and Web Frontend ● Learning Django ● Initializing the Django project for the backend endpoints ● Learning Django Rest API ● Learning W3 Activity Streams ● Learning Json-LD ● Implementing ListCommunityPosts function ● Implementing ListPostTemplates function ● In home.py, implementation of get_community_data, try_create_community, get_subscription_status, set_subscription_status, search_communities, get_user_posts functions ● Writing unit tests for the Backend API ● Adding Swagger auto-documentation functionality to the code. ● Configuring endpoints for JSON-LD standards ● Making database migration from SQLite to MySQL ● Learning Amazon AWS ● Deploying backend services on Amazon AWS ● Working on fixing CORS errors ● Working on authentication errors ● Configuring endpoints for JSON-LD standards ● Reviewing pull requests made by Berke Argın

	● Taking and uploading Backend Meeting Notes 3,4 ● Fixing the bugs in the Backend service specified by the frontend teams ● Attending group meetings. ● Attending lab sessions.
Muhammed Göktepe	● Updating System and User Requirements ● Updating Sequence Diagram ● Updating Wiki Page for CMPE451 ● Updating Sidebar for CMPE451 ● Learning React-Bootstrap ● Learning React Hooks ● Learning React Axios ● Learning React Redux ● Learning React Navigation ● Creating Registration Page ● Creating Login Page ● Creating My Posts Page ● Creating Create Post Template Page with Emir ● Creating Create Post Page with Emir ● Creating Feed Page with Emir and Bilal ● Creating Create Community Page with Bilal ● Taking and uploading Meeting Notes 12, 14, 16 ● Taking and uploading Frontend Meeting Notes 1 and 2 ● Reviewing pull requests made by Bilal and Emir ● Creating issues for Frontend development status ● Creating issue for project development ● Creating pull requests for Frontend
Emir Hıfsı Öztoprak	● Updating sequence diagram following the feedback from our TA and uploading them back to our project page ● Learning Javascript basics ● Learning React ● Creating Create Post Template Page with Muhammed ● Creating Create Post Page with Muhammed ● Creating Community Feed Page ● Creating My Communities Page ● Creating All Communities Page ● Reviewed 2 pull requests ● Creating a pull request to Web_ReactJS branch
Bilal Tekin	• Creating the main branch Web_ReactJS • Initializing the Web project and pushing to github. • Implementation of Redux-Thunk • Implementation of React Navigation • Implementation of React Bootstrap and creating examples • Installation ProSideBar and Implementation of it. • Implementation of Axios template and using it for backend integration • Making some meetings for frontend team to tell them my experiences about react and teach them the details. • Creating an AWS account and deploying frontend application to the server.

	• Meeting with the backend team to help them in deployment of backend to aws servers. • Creating necessary issues for frontend about react. • Reviewing frontend team code • Reviewing pull requests made by Muhammed and Emir • Creating issues for Frontend development status • Creating issue for project development • Creating pull requests for Frontend • Research about how to fix Cors Origin Policy and secure cookie problems • Filling necessary parts of the Group Milestone 1 • Attending group meetings.
Yunus Emre Topal	• Revising Functional Requirements • Updating Customer Questions • Taking meeting notes 15 • Reviewing changes done by my groupmates on Requirements • Reviewing changes done by my groupmates on Sequence Diagram • Attending group meetings (i missed 2 meetings) • Creating road map for frontend • Creating issues for project planning • Creating issues for mobile developing • Discussing project development tools with group members. • Attending customer meetings. • Presenting mobile part of our app during the demo • Creating Welcome Screen on mobile • Creating Login Screen on mobile • Implementing API Call for Login Feature • Creating Registration Screen on mobile • Implementing API Call for Registration Feature • Putting stack navigation components to our app • Creating drawer navigation component to app • Creating Feed Screen on mobile • Implementing API Call for getting Feed data • Creating Communities Screen on mobile • Creating User Posts Screen on mobile • Implementing API Call for getting User Posts • Creating User Communities Screen on mobile • Creating Settings Screen on mobile • Creating Post Screen on mobile • Creating Community Screen on mobile • Reviewing pull requests for mobile • Filling Mobile Application part on Evaluation of Deliverables. • Filling Mobile part on Evaluation of Tools and Processes Used • Filling my personal Summary of Work Done (This part!) • Creating Roadmap for Mobile Part. • Filling my personal part on RoadMap • Creating Mobile part on RAM • Filling my personal part on RAM • Filling Mobile part of Moving Forward

6.Deliverables

a.Communication Plan

Participants	Place	Purpose	When
Team Members	Zoom	General Discussion	Every Thursday at 20.00
Team Members	Slack	Weekly Progress Review	When Needed
Team Members	Discord	General Discussion	When Needed
Team Members	WhatsApp	Urgent Communication	When Needed

b.Requirements

1. Functional Requirements

1.1 User Requirements

- **1.1.1 Registration and Log in**
  - 1.1.1.1 A user shall be able to register by providing a valid e-mail address, a username, and a strong password.
  - 1.1.1.2 Users shall be able to register by providing their twitter or facebook social media accounts.
  - 1.1.1.3 Users shall be able to log in to their account with their email/username and password.
  - 1.1.1.4 If a user logged in to their account, then the next time they open the app they will be logged in automatically.
  - 1.1.1.5 Users shall be able to login by providing their social media account which they used during registration.
- **1.1.2 Home Page**
  - 1.1.2.1 Users should be able to see posts from their communities in their home page.
  - 1.1.2.2 Users should be able to see the communities that they're joined.
  - 1.1.2.3 Users should be able to name search and list other users provided that their accounts are public.

  - 1.1.2.4 Users should be able to name search and list communities.
  - 1.1.2.5 Users should be able to title search and list post in their home post feed.
- **1.1.3 Settings Page**
  - 1.1.3.1 Users should be able to change mode of the app (light and dark mode).
  - 1.1.3.2 Users should be able to disable notifications.
  - 1.1.3.3 Users should be able to log out from the app.
  - 1.1.3.4 Users should be able to add profile pictures from settings menu.
  - 1.1.3.5 Changing password/username
    - 1.1.3.5.1 Users shall be able to change their username if the username they wish to use is available.
    - 1.1.3.5.2 Users shall be able to change their password with e-mail confirmation.
  - 1.1.3.6 Account Privacy
    - 1.1.3.6.1 Users shall be able to make their profile public or private.
    - 1.1.3.6.2 Users shall be able to accept or send follow requests from/to other users.
    - 1.1.3.6.3 Users shall be able to block other users. Blocked users can't see the activities of that user and also can't message them.
- **1.1.4 Community Features**
  - 1.1.4.1 A user shall be able to create a community by specifying a unique name, description and privacy status (public or private). After a successful creation, user shall be a moderator of that community.
  - 1.1.4.2 A user shall be able to view a community page that will contain community post feed and information.
  - 1.1.4.3 Users shall be able to join to any public community.
  - 1.1.4.4 Users shall be able to leave a community that they're currently joined.
- **1.1.5 Moderating Features**
  - 1.1.5.1 Moderators can ban users if the users engage in an ill behaviour.
  - 1.1.5.2 Moderators should be able to assign other users as moderators.
  - 1.1.5.3 Moderators should be able to send users email invitations to the community.
  - 1.1.5.4 Moderators should be able to delete the community that they moderate.
  - 1.1.5.5 Moderators should be able to create community specific post tags that can be used in the posts.
- **1.1.6 Post Templates**

    - 1.1.6.1 Community moderators should be able to create post templates for their communities.
    - 1.1.6.2 Community moderators should be able to select and add multiple data fields to the post templates.
    - 1.1.6.3 Users should be able to list and select post templates during post creation.
- **1.1.7 Posts**
    - 1.1.7.1 Users should be able to see post feed of a community in its respective community page.
    - 1.1.7.2 Users should be able to create a post in a joined community by selecting a post template and filling out necessary fields.
    - 1.1.7.3 Users should be able to sort post by various criteria in post feeds.
    - 1.1.7.4 Users should be able to add existing tags to their posts.
    - 1.1.7.5 Users should be able to search for titles of posts with keywords using search bar in both home and community post feed.
    - 1.1.7.6 Users should be able to filter posts in terms of their communities, post templates, and the query methods defined for the data field types in the post template.
    - 1.1.7.7 Users should be able to like or comment to posts in both home and community post feed.

1.2 System Requirements

- **1.2.1 Notification**
    - 1.2.1.1 System notifies users when there is a new post in their communities.
    - 1.2.1.2 System notifies users when people like their posts.
    - 1.2.1.3 System notifies users when people comment their posts.
    - 1.2.1.4 System notifies users when people like comments.
    - 1.2.1.5 System notifies users when people comment comments.
    - 1.2.1.6 System gives an option to users disallow notifications that come from
        - 1.2.1.6.1 posts
        - 1.2.1.6.2 comments
        - 1.2.1.6.3 likes
- **1.2.2 Users and Communities**
    - 1.2.2.1 System must ensure uniqueness of the username and e-mail before registration.
    - 1.2.2.2 Users shall enter their password 2 times to prevent typos in registration.
    - 1.2.2.3 To register, users must confirm their e-mail adress using a link sent by the system.

    - 1.2.2.4 System must ensure uniqueness of the name of a community before community creation.
    - 1.2.2.5 System will not permit usage of priveledged moderator actions by regular users of the community.
    - 1.2.2.6 Users shall be redirected to their homepage after a successful login.
- **1.2.3 Post Templates and Data Fields**
    - 1.2.3.1 System must support post templates that contain multiple data field types including but not limited to text, image, location.
    - 1.2.3.2 System must be able to render posts that contain different data field types correctly in the UI.
    - 1.2.3.3 System must ensure the exclusivity of post templates to their respective communities (i.e. System should not allow post template usage outside its community.)
- **1.2.4 Recommendation**
    - 1.2.4.1 System must be able to recommend new communities to the users according to their current communities.
    - 1.2.4.2 System must be able to recommend the most followed communities to the users.
    - 1.2.4.3 System must be able to recommend the most liked posts in the users joined communities.
- **1.2.5 Search/Filter**
    - 1.2.5.1 System must support text search on names/ titles of users, communities and posts.
    - 1.2.5.2 System must support tag based filtering on posts.
    - 1.2.5.3 System must support queries on data fields based on their type. These queries will be used by users to filter specific posts.

2. Nonfunctional Requirements

2.1 Availability

- 2.1.1 Application shall be available on Android based mobile application stores.
- 2.1.2 System shall be accessible using modern web browsers.

2.2 Standards

- 2.2.1 The semantic taggings should be supported with Wikidata.org
- 2.2.2 The system must follow the rules of W3C Activity Streams Standards 2.0

2.3 Privacy

- 2.3.1 Users shall agree to the User Terms & Policy Agreement before registration.

2.4 Security

- 2.4.1 User's password will be encrypted with SHA-256 and only encrypted passwords will be contained in the database.
- 2.4.3 JWT or Authentication Tokens will be used to authenticate users.

c.Scenarios and Mockups

1. Scenario 1

Persona

- John Doe
- 23 Years Old
- Boğaziçi University Junior Student
- Lives in Hisarüstü
- Currently Pursuing Bachelor's Degree in Computer Engineering
- Admin of the Boğaziçi Housing Community

Story

- John Doe is an enthusiastic user of our app.

- He had to change 4 houses in 2 semesters, which he did not enjoy. He wants to make sure everyone can find a house in Hisarüstü at reasonable prices.
- He wants to make sure the Boğaziçi Housing community is a spam-free place.
- He checks for spam content daily and bans user accordingly.
- He just broke his old phone, and he is using a brand new phone.

Preconditions

- He is not a first-time user. He actually founded the Boğaziçi Housing community. However, he just broke his phone and he has to sign in for the first time on this phone.

Goals

- He wants to login with his new phone.
- He wants to check for spam content and ban users.

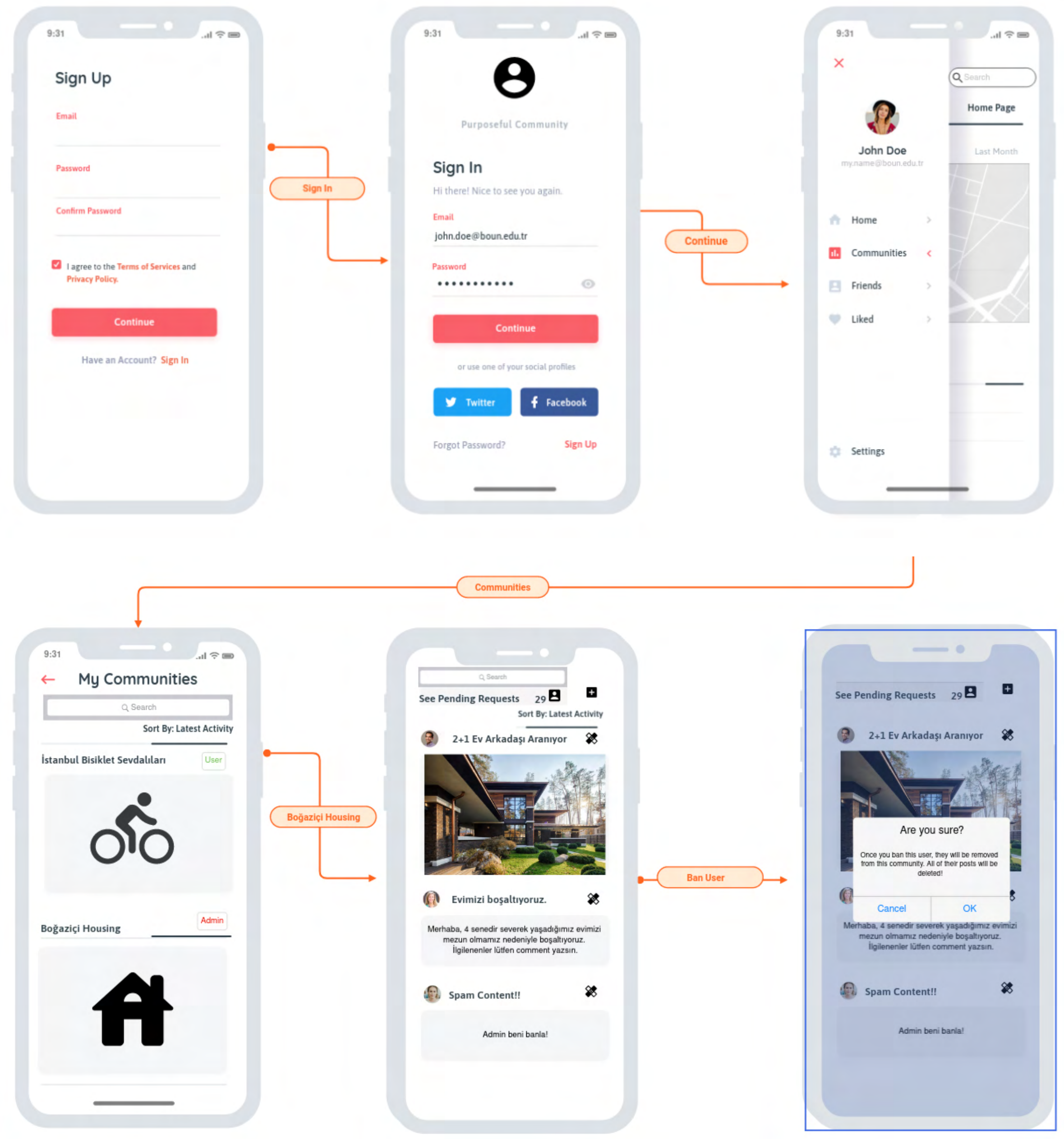
Acceptance Criteria

- (1.1.1.1) Unregistered users shall be able to register either using an username/password or their social media accounts.
- (1.1.1.2) Registered users shall be able to login either using their username/password or social media.
- (1.1.1.3.2) Users who forgot their password shall be able to reset their passwords by clicking "Forgot Password?".
- (1.1.1.2.4) Users shall be initially directed to their homepages.
- (1.2.5.3) Users shall be able to switch to their homepage, communities, friends and liked posts.
- (1.2.5.2 and 1.2.5.5) Users shall be able to search and filter posts according to several criteria.
- (1.2.8) Users shall be able to configure privacy options as well as app options from the settings menu.
- (1.2.6.2 and 1.2.6.4) Users shall be able to search for and sort communities by several criteria.
- (1.2.6.3) Users shall be able to see which communities they moderate.
- (1.2.7.5 and 1.2.7.6) Users shall be able to search for and sort posts by several criteria.
- (1.2.7.3) Users shall be able to report posts they deem unfit for the community.
- (1.2.7.4) Users shall be able to post on communities they belong to.

- (1.2.7.7) Community admins shall see an admin-only toolbar where they see pending requests, reported posts and so on.
- (1.2.7.8) Community admins shall be able to either delete posts or ban users.

Scenario with Mockups

- John Doe was an admin, but he broke his phone.
- He bought a new phone and installed the app.
- He opened the app for the first time on this phone. Since he never logged in on this phone before, he saw the sign-up screen.
- He tapped sign-in, since he was already an active user.
- He entered his username and password. He did not login using his social media, since he uses this app exclusively.
- He tapped continue.
- He was logged in and directed to his homepage. He used the sidebar to switch to communities view.
- He saw the 2 communities he was a part of: Istanbul bisiklet sevdalilari and Boğaziçi Housing. The communities were sorted by latest activity.
- He tapped Boğaziçi Housing, where he is the admin.
- He saw 29 pending requests, as well as a spam post.
- He decided to not only remove the post, but ban the user from the community.
- He saw a popup to confirm that he was sure.
- He banned the user from the community.



Merhaba, 4 senedir severek yaşadığımız evimizi mezun olmamız nedeniyle boşaltıyoruz. İlgilenenler lütfen comment yazsın.

Admin beni banla!

Once you ban this user, they will be removed from this community. All of their posts will be deleted!

2. Scenario 2

Sebastian Vettel

Persona

- Sebastian Vettel
- 33 years old
- He is from Heppenheim, West Germany
- He is married and has two children
- He is a Formula 1 driver for Aston Martin

Story

- Sebastian Vettel downloaded our app months ago and uses it to follow the news about F1 and F1 communities.
- Since he has been racing for 15 years, he has lots of friends who are Formula 1 drivers as well.
- Since F1 drivers travel around the world often, he can only keep in touch with his friends via our app.
- He wants to buy gifts for his friend Daniel Ricciardo for his birthday, but he forgot the exact date.
- He wants to make sure the date.
- He also wants to learn more about Daniel's hobbies so that he can buy a better gift for him.

Preconditions

- He is not a first-time user. He signs in with email and password.

Goals

- He wants to login with his phone.
- He wants to enter Daniel's profile.
- He wants to tap the "about" button to see Daniel's personal info.

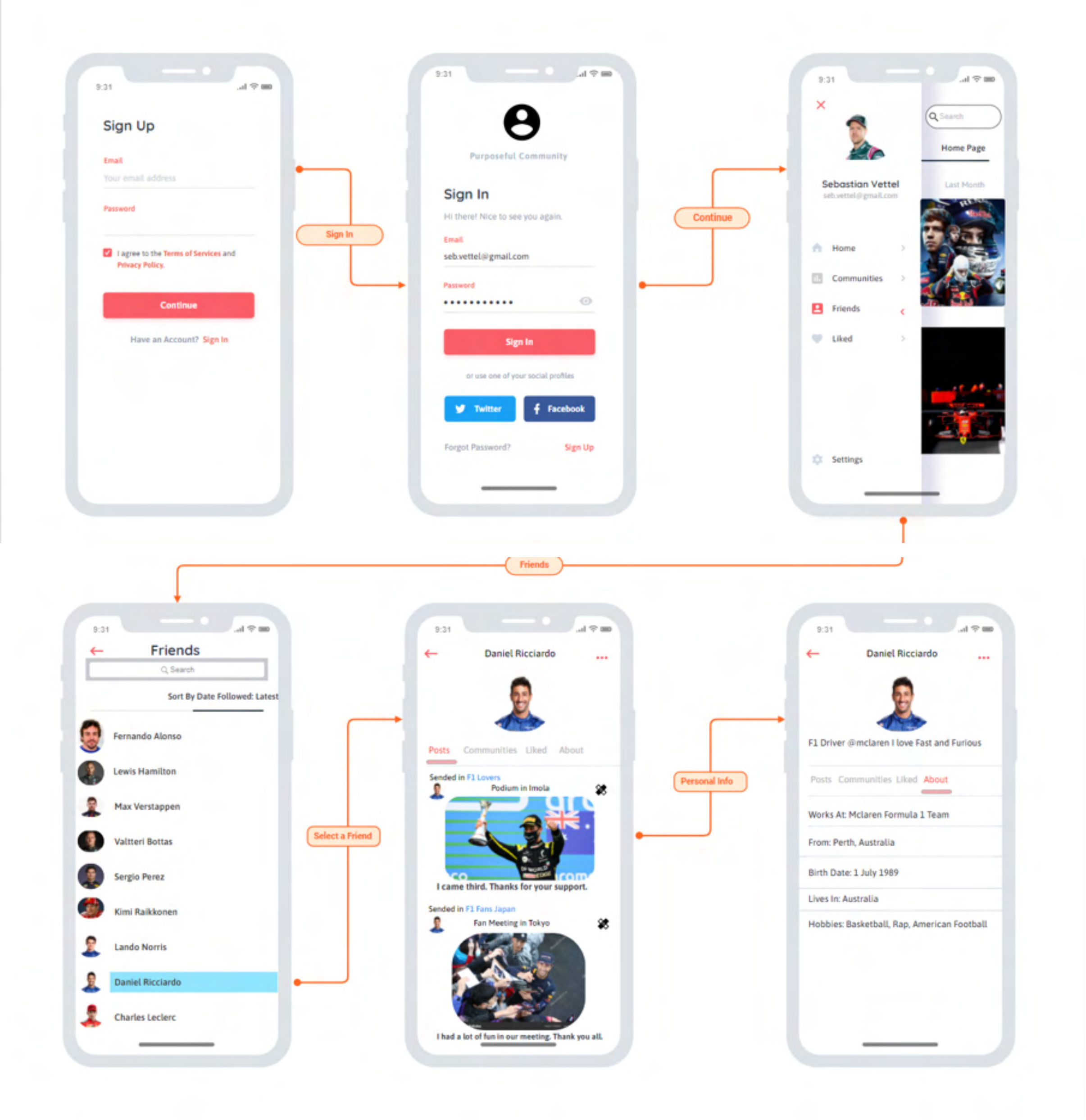
Acceptance Criteria

- (1.1.1.1.1 and 1.1.1.1.6) Unregistered users shall be able to register either using a username/password or their social media accounts.
- (1.1.1.2.1) Registered users shall be able to login either using their username/password or social media.
- (1.1.1.3.2) Users who forgot their password shall be able to reset their passwords by clicking "Forgot Password?".
- (1.1.1.2.4) Users shall be initially directed to their homepages.
- (1.2.5.3) Users shall be able to switch to their homepage, communities, friends and liked posts.
- (1.2.3 and 1.2.4) Users shall be able to search and filter posts according to several criteria.
- (1.1.1.4) Users shall be able to configure privacy options as well as app options from the settings menu.
- (1.2.6.4 and 1.2.7.5) Users shall be able to search for and sort friends by several criteria.
- (1.1.2.2) Users shall be able to post on communities they belong to.
- (1.1.1.4) Users shall be able to see information about their friends.
- (1.1.1.4.1) Users shall be able to see the public posts their friends shared.
- (1.1.1.4) Users shall be able to see the communities followed by their friends.
- (1.1.2.3) Users shall be able to see the posts that are liked by their friends.

Scenario with Mockups

- Sebastian Vettel was a user, and he has been using this app for four months.
- He opened the app on this phone and tapped sign-in, since he was already an active user.
- He entered his email and password.
- He tapped continue.
- He was logged in and directed to his homepage. He used the sidebar to switch to friends view.
- He saw his friends. They were sorted by date followed: latest.
- He tapped Daniel Ricciardo's page.
- He saw the posts Daniel sent to different public communities.
- He tapped the about button to see Daniel's personal information.

- He saw Daniel's bio and saw his birthday and hometown.
- He also saw the hobbies of Daniel and chose a gift based on these.



Email: seb.vettel@gmail.com

Password: ●●●●●●●●●●●●

I came third. Thanks for your support.

I had a lot of fun in our meeting. Thank you all.

F1 Driver @mclaren I love Fast and Furious

Works At: Mclaren Formula 1 Team

From: Perth, Australia

Birth Date: 1 July 1989

Lives In: Australia

Hobbies: Basketball, Rap, American Football

3. Scenario 3

Beth Harmon in 70's.

Persona

- Beth Harmon
- 73 years old
- Currently resides in Kentucky.
- Chess prodigy who currently holds a FIDE Elo 2673.

Story

- Beth Harmon is a professional chess player, mostly active in 60's and 70's.
- Even though her age, she likes to spend her time in a chess community to study chess and answer other peoples questions.
- She is currently retired, however she still follows the chess news and participates in tournaments.

Preconditions

- Beth already has an account and joined several communities.
- Beth is logged in to the website through a web browser.
- She is familiar with the community post filtering feature.

Goals

- Beth wants to browse the Chess Community page.
- She wishes to find a tournament which fits her schedule.

Acceptance Criteria

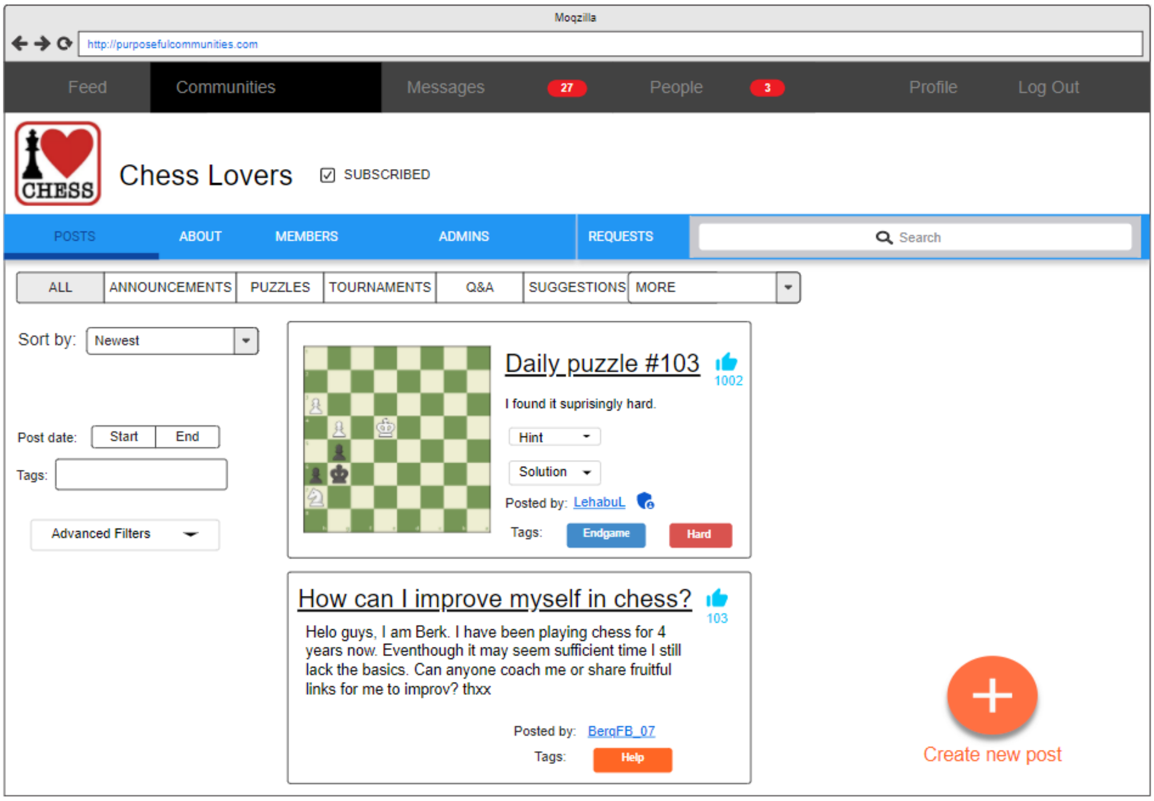
- (1.1.1.2.2) Users should be able to stay logged-in after closing the app or browser.
- (1.2.7.9) Posts must support multiple data field types including text, image, location.
- (1.2.4) For several types of data fields used in posts, there must be some predefined comparison -methods and queries to be used in filtering posts.
- (1.2.7.10) Users must use a predefined datatype template for posting.
- (1.2.4) Users should be able to add existing tags to their posts.
- (1.2.7.9) Post datatypes must support some text formatting including dropdown texts and hyperlinks.
- (1.2.3.3) Users should be able to browse posts in groups based on their datatypes.
- (1.2.7.11) Users should be search and filter specific types of posts with some predefined comparison methods and queries

Scenario with Mockups

- Beth opens the "Chess Lovers" community page.
- She selects the "Tournaments" tab.
- She filters the tournaments posts by their location, date and time.
- She wishes to attend a tournament on that same day between 11.00 and 18:00.
- But there is no tournament fitting her preferences.
- So she decides to host her own tournament and let her fellow chess players know on the community.
- She presses the create new post button.
- She fills the predetermined fields of the tournament datatype and posts it.

Mockup

- Beth views the "Chess Lovers" page.



Daily puzzle #103

1002

I found it suprisingly hard.

Posted by: LehabuL

Tags: Endgame Hard

How can I improve myself in chess?

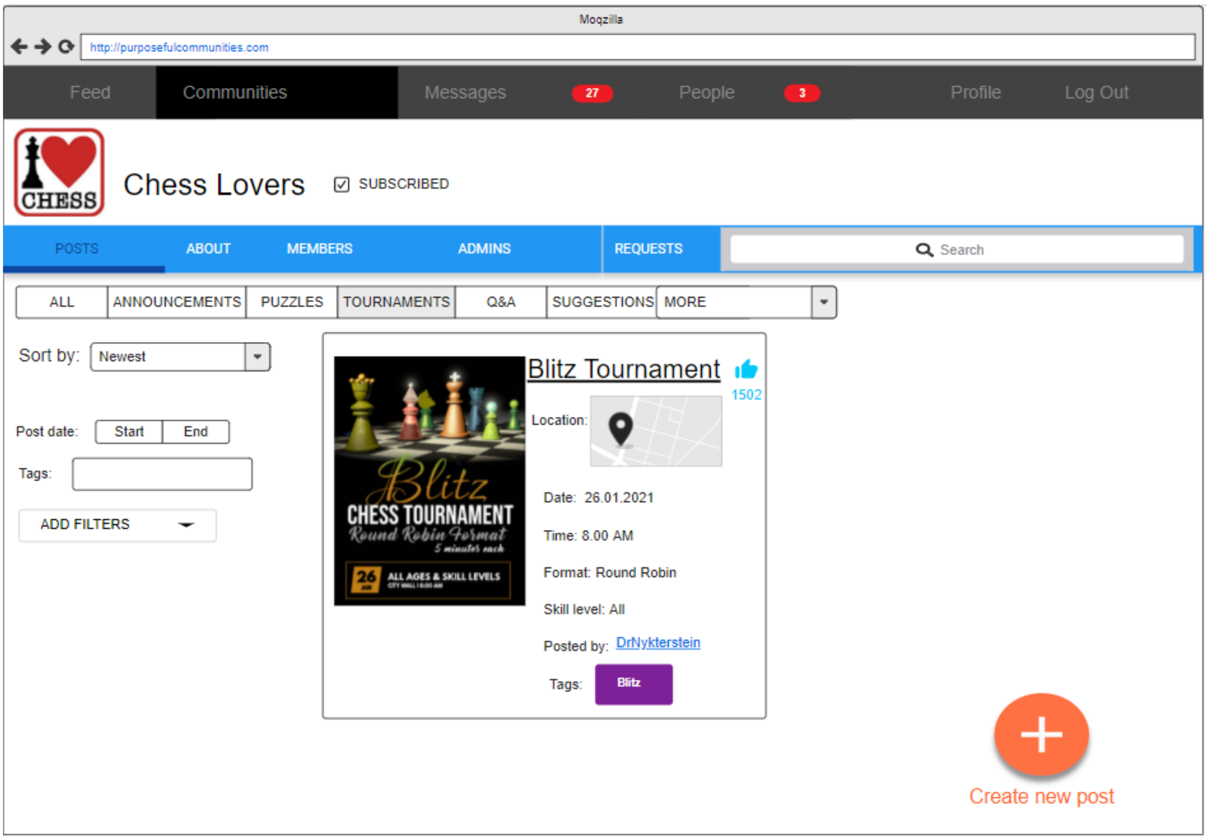
103

Helo guys, I am Berk. I have been playing chess for 4 years now. Eventhough it may seem sufficient time I still lack the basics. Can anyone coach me or share fruitful links for me to improv? thxx

Posted by: BergFB_07

Tags: Help

- She selects "Tournament" tab.



Blitz Tournament

1502

Location:

Date: 26.01.2021

Time: 8.00 AM

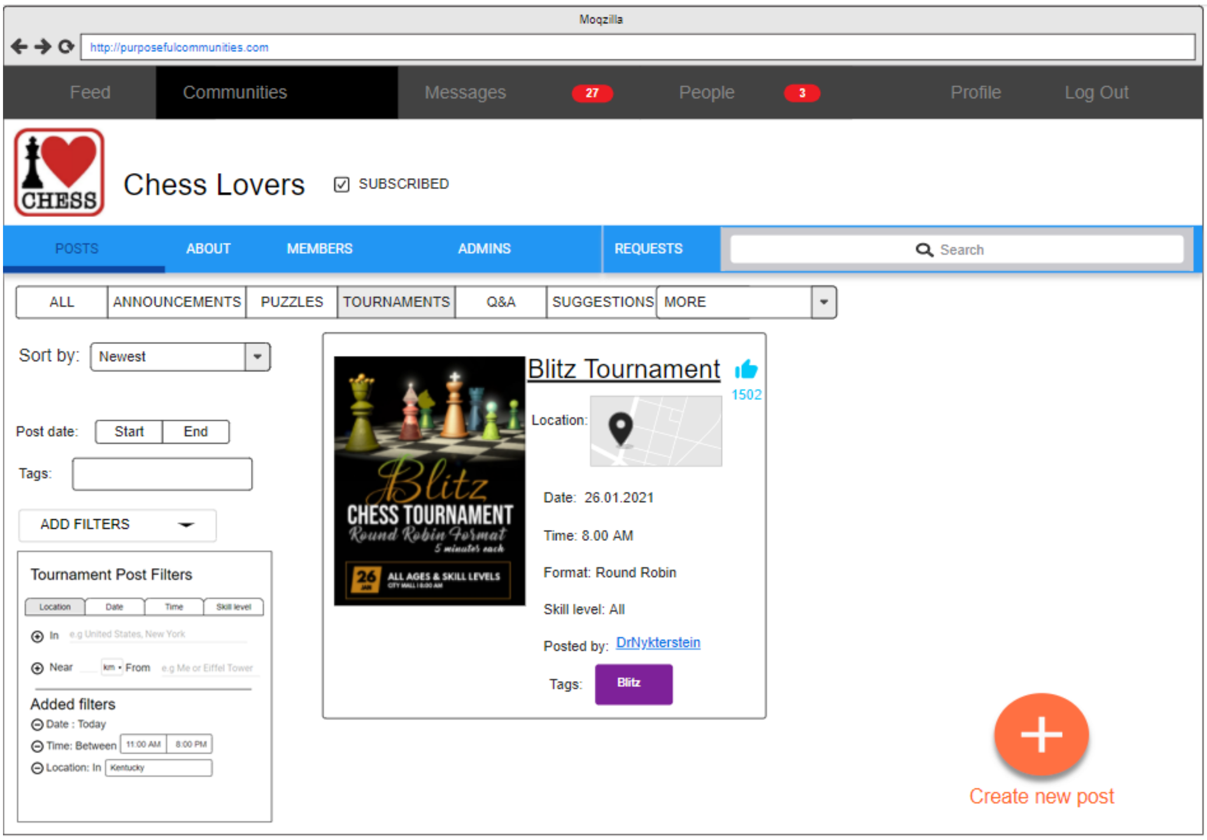
Format: Round Robin

Skill level: All

Posted by: DrNykterstein

Tags: Blitz

- She filters the posts on her preferences.



Blitz Tournament 1502

Location:

Date: 26.01.2021

Time: 8.00 AM

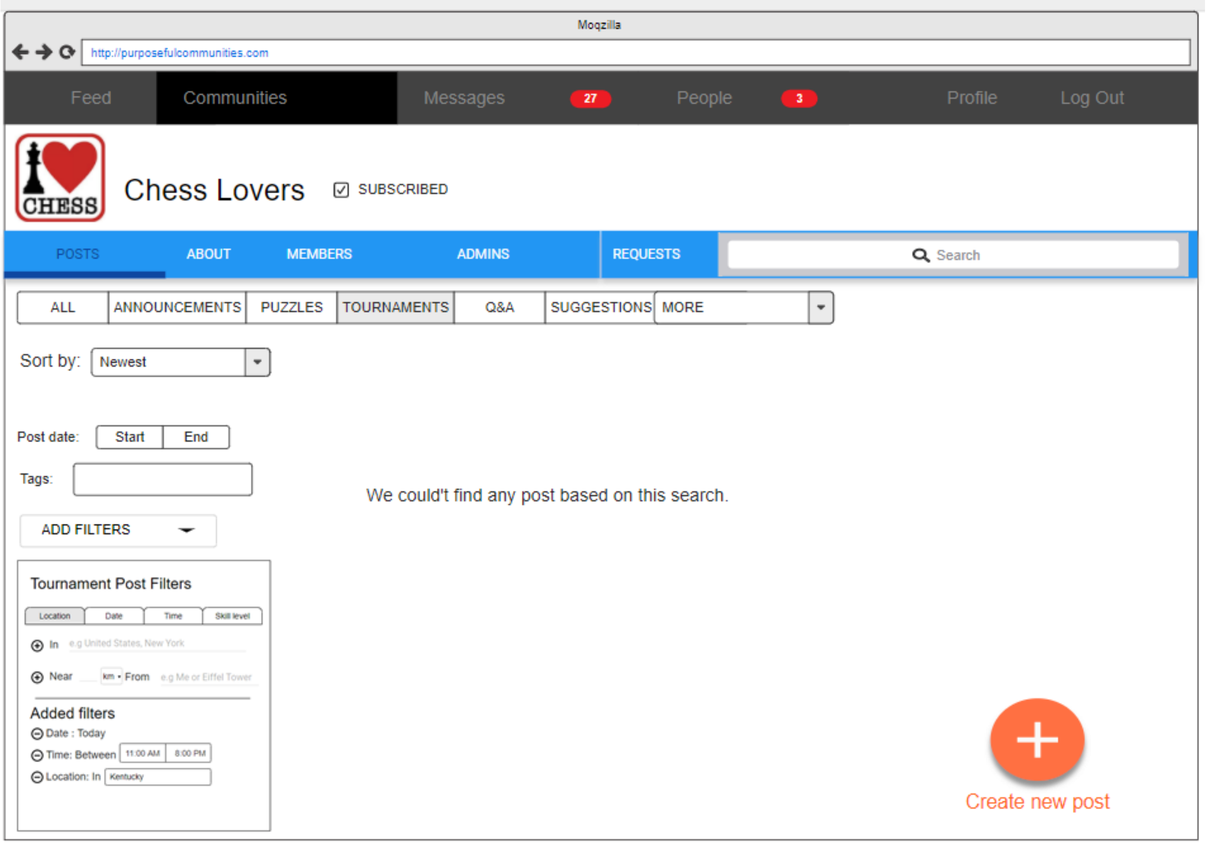
Format: Round Robin

Skill level: All

Posted by: DrNykterstein

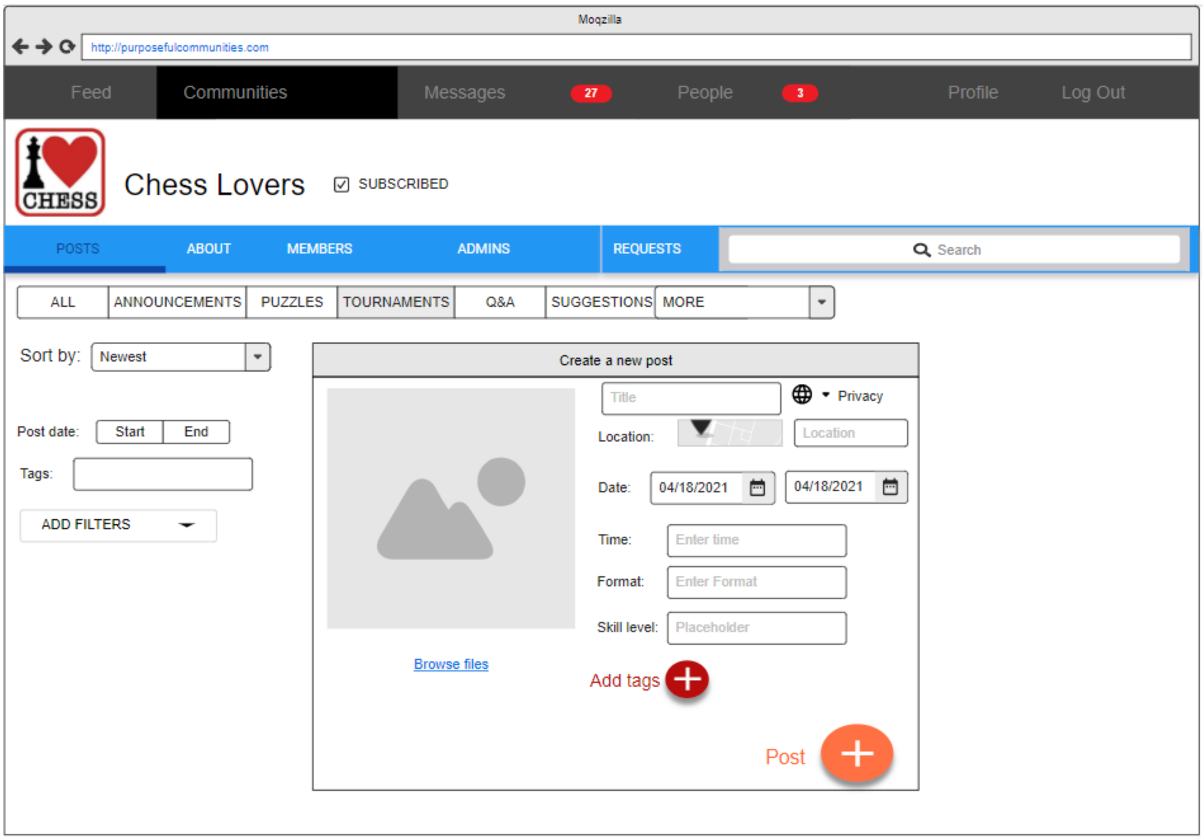
Tags: Blitz

- There is no tournament fitting her.



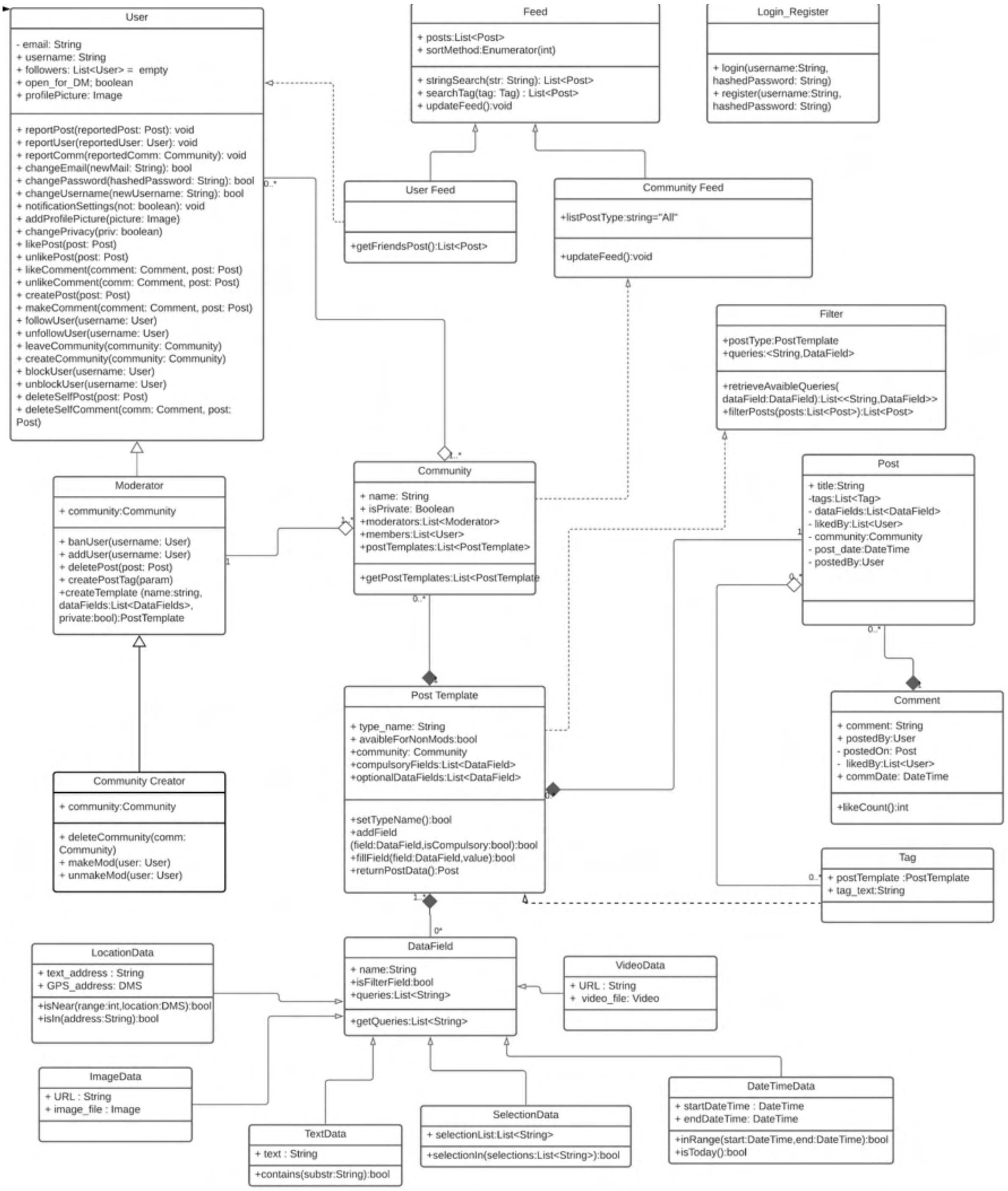
We could't find any post based on this search.

- She creates a new Tournament post.



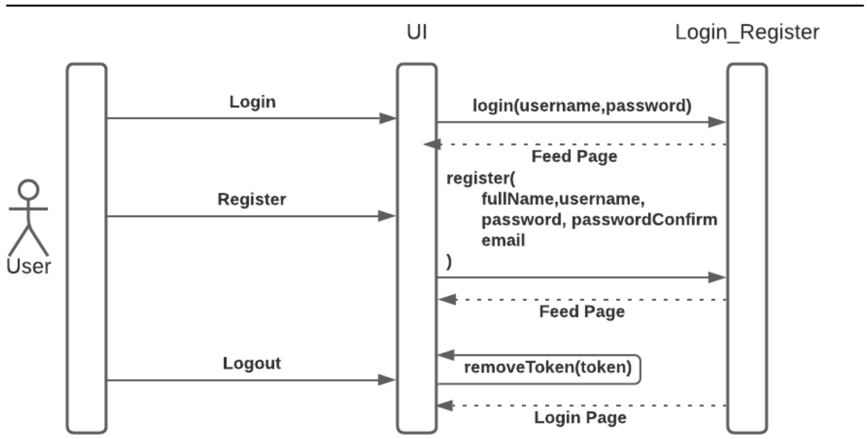
d.Software design documents in UML

1. Class Diagram

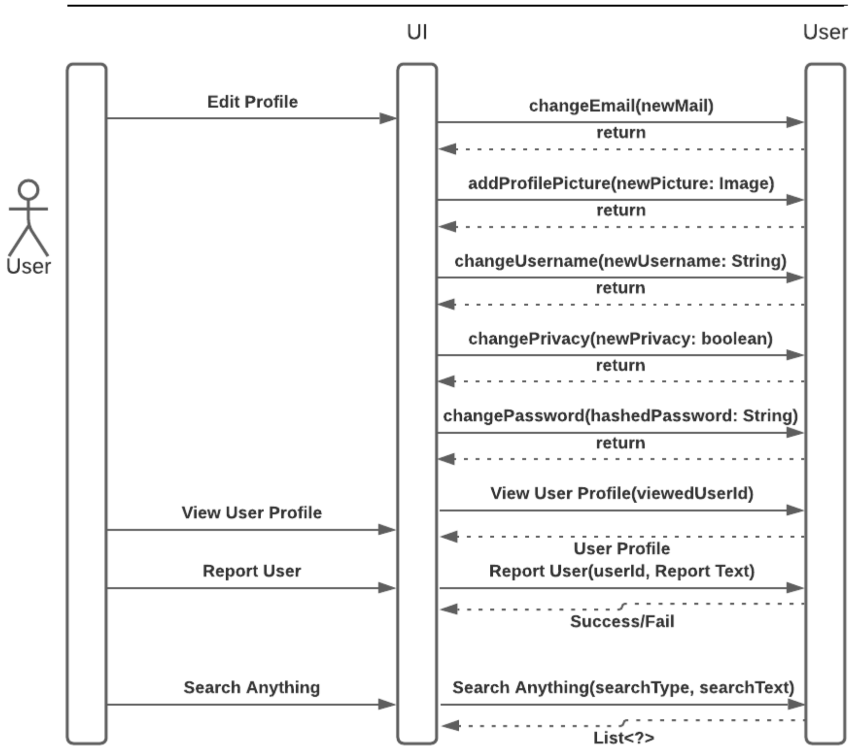

User
- email: String
+ username: String
+ followers: List<User> = empty
+ open_for_DM; boolean
+ profilePicture: Image
+ reportPost(reportedPost: Post): void
+ reportUser(reportedUser: User): void
+ reportComm(reportedComm: Community): void
+ changeEmail(newMail: String): bool
+ changePassword(hashedPassword: String): bool
+ changeUsername(newUsername: String): bool
+ notificationSettings(not: boolean): void
+ addProfilePicture(picture: Image)
+ changePrivacy(priv: boolean)
+ likePost(post: Post)
+ unlikePost(post: Post)
+ likeComment(comment: Comment, post: Post)
+ unlikeComment(comm: Comment, post: Post)
+ createPost(post: Post)
+ makeComment(comment: Comment, post: Post)
+ followUser(username: User)
+ unfollowUser(username: User)
+ leaveCommunity(community: Community)
+ createCommunity(community: Community)
+ blockUser(username: User)
+ unblockUser(username: User)
+ deleteSelfPost(post: Post)
+ deleteSelfComment(comm: Comment, post: Post)
0..*
Feed
+ posts:List<Post>
+ sortMethod:Enumerator(int)
+ stringSearch(str: String): List<Post>
+ searchTag(tag: Tag) : List<Post>
+ updateFeed():void
Login_Register
+ login(username:String, hashedPassword: String)
+ register(username:String, hashedPassword: String)
User Feed
+getFriendsPost():List<Post>
Community Feed
+listPostType:string="All"
+updateFeed():void
Filter
+postType:PostTemplate
+queries:<String,DataField>
+retrieveAvaibleQueries(dataField:DataField):List<<String,DataField>>
+filterPosts(posts:List<Post>):List<Post>
1..*
Community
+ name: String
+ isPrivate: Boolean
+moderators:List<Moderator>
+members:List<User>
+postTemplates:List<PostTemplate>
+getPostTemplates:List<PostTemplate
1..*
0..*
Moderator
+ community:Community
+ banUser(username: User)
+ addUser(username: User)
+ deletePost(post: Post)
+ createPostTag(param)
+createTemplate (name:string, dataFields:List<DataFields>, private:bool):PostTemplate
1
Post
+ title:String
-tags:List<Tag>
- dataFields:List<DataField>
- likedBy:List<User>
- community:Community
- post_date:DateTime
- postedBy:User
1
0..*
0..*
1
Comment
+ comment: String
+ postedBy:User
- postedOn: Post
- likedBy:List<User>
+ commDate: DateTime
+likeCount():int
Community Creator
+ community:Community
+ deleteCommunity(comm: Community)
+ makeMod(user: User)
+ unmakeMod(user: User)
1
Post Template
+ type_name: String
+ avaibleForNonMods:bool
+community: Community
+compulsoryFields:List<DataField>
+optionalDataFields:List<DataField>
+setTypeName():bool
+addField (field:DataField,isCompulsory:bool):bool
+fillField(field:DataField,value):bool
+returnPostData():Post
0..*
Tag
0..*
+ postTemplate :PostTemplate
+ tag_text:String
1..*
0*
DataField
+ name:String
+isFilterField:bool
+queries:List<String>
+getQueries:List<String>
LocationData
+ text_address : String
+ GPS_address: DMS
+isNear(range:int,location:DMS):bool
+isIn(address:String):bool
VideoData
+ URL : String
+ video_file: Video
ImageData
+ URL : String
+ image_file : Image
TextData
+ text : String
+contains(substr:String):bool
SelectionData
+ selectionList:List<String>
+selectionIn(selections:List<String>):bool
DateTimeData
+ startDateTime : DateTime
+ endDateTime: DateTime
+inRange(start:DateTime,end:DateTime):bool
+isToday():bool

2. Sequence Diagram

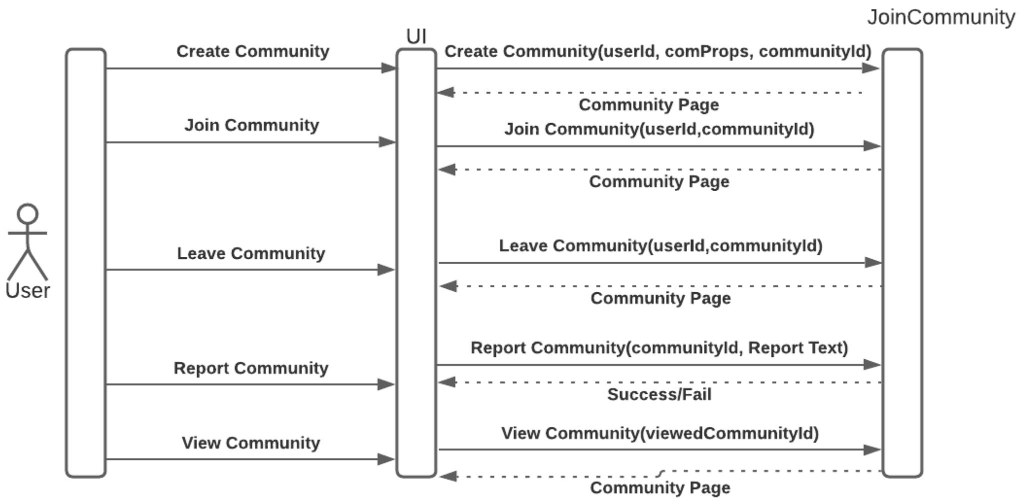
AUTHENTICATION



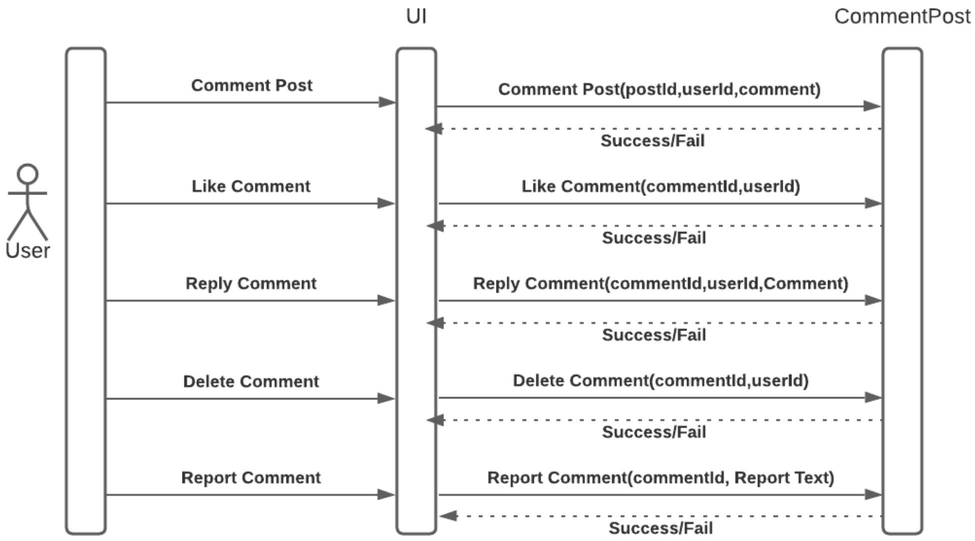
PROFILE


UI
User
User
Edit Profile
changeEmail(newMail)
return
addProfilePicture(newPicture: Image)
return
changeUsername(newUsername: String)
return
changePrivacy(newPrivacy: boolean)
return
changePassword(hashedPassword: String)
return
View User Profile
View User Profile(viewedUserId)
User Profile
Report User
Report User(userId, Report Text)
Success/Fail
Search Anything
Search Anything(searchType, searchText)
List<?>

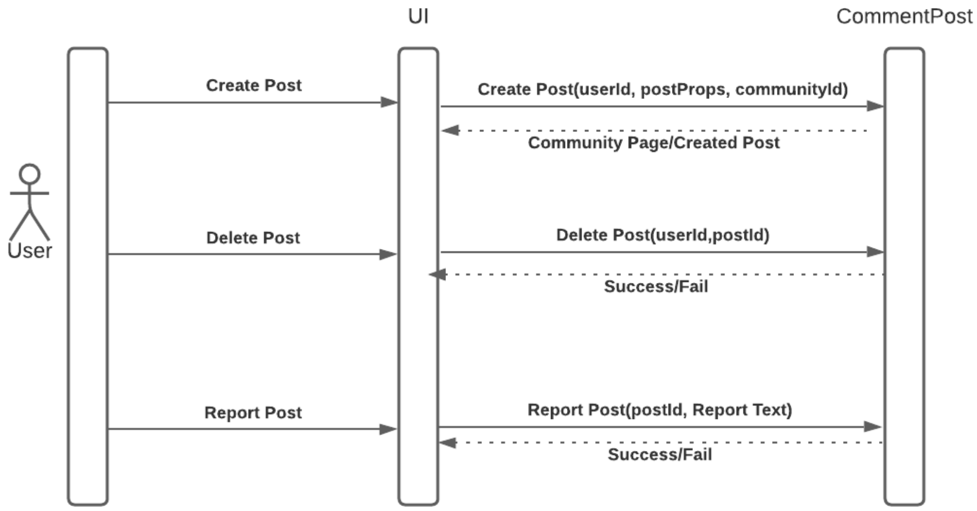
Community


JoinCommunity
UI
User
Create Community
Create Community(userId, comProps, communityId)
Community Page
Join Community
Join Community(userId,communityId)
Community Page
Leave Community
Leave Community(userId,communityId)
Community Page
Report Community(communityId, Report Text)
Report Community
Success/Fail
View Community(viewedCommunityId)
View Community
Community Page

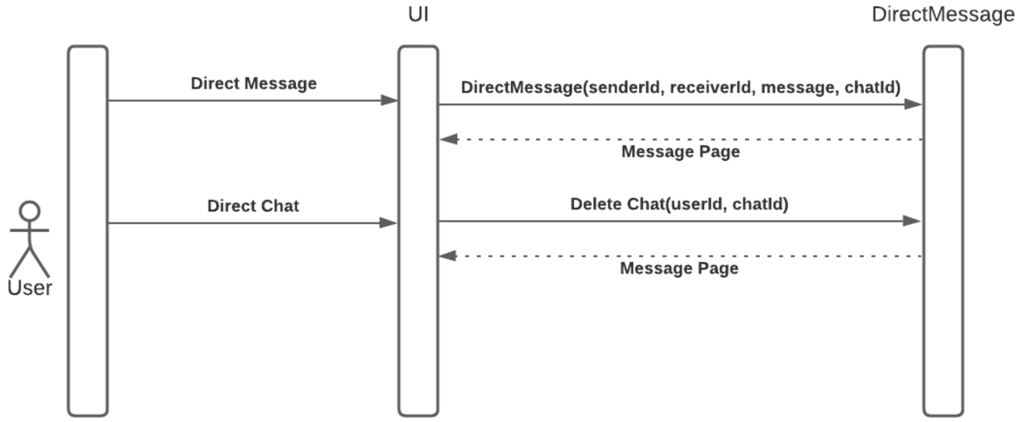
COMMENT


UI
CommentPost
User
Comment Post
Comment Post(postId,userId,comment)
Success/Fail
Like Comment
Like Comment(commentId,userId)
Success/Fail
Reply Comment
Reply Comment(commentId,userId,Comment)
Success/Fail
Delete Comment
Delete Comment(commentId,userId)
Success/Fail
Report Comment
Report Comment(commentId, Report Text)
Success/Fail

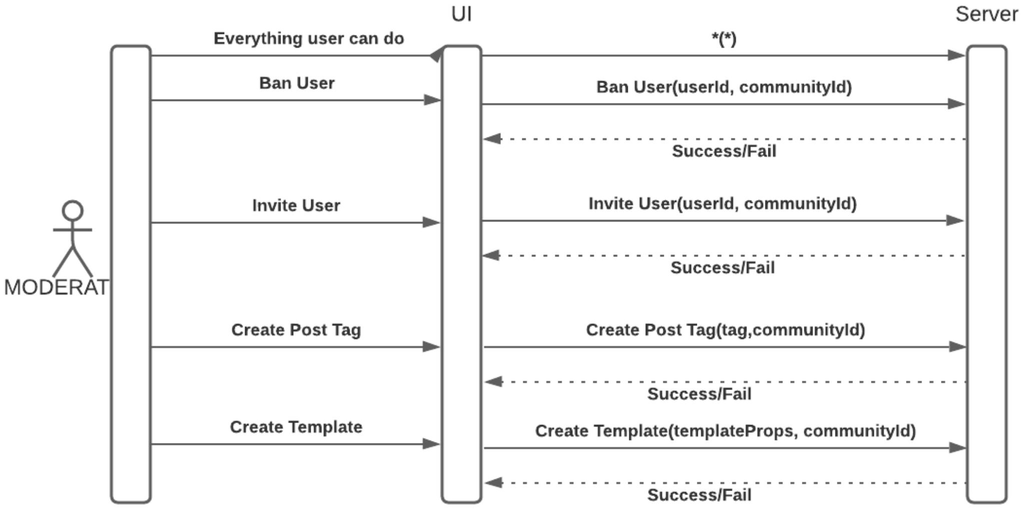
POST

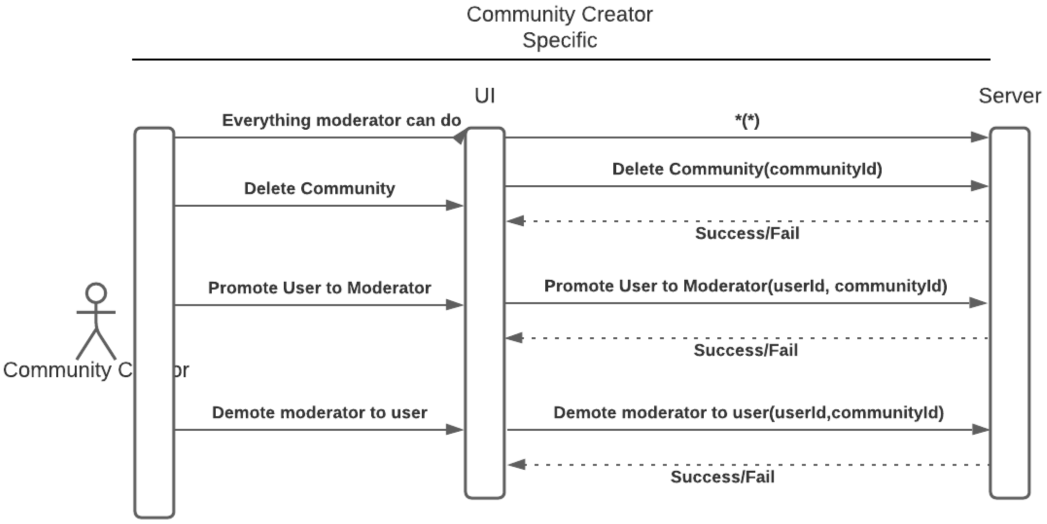

UI
CommentPost
User
Create Post
Create Post(userId, postProps, communityId)
Community Page/Created Post
Delete Post
Delete Post(userId,postId)
Success/Fail
Report Post
Report Post(postId, Report Text)
Success/Fail

CHAT / MESSAGE


UI
DirectMessage
User
Direct Message
DirectMessage(senderId, receiverId, message, chatId)
Message Page
Direct Chat
Delete Chat(userId, chatId)
Message Page

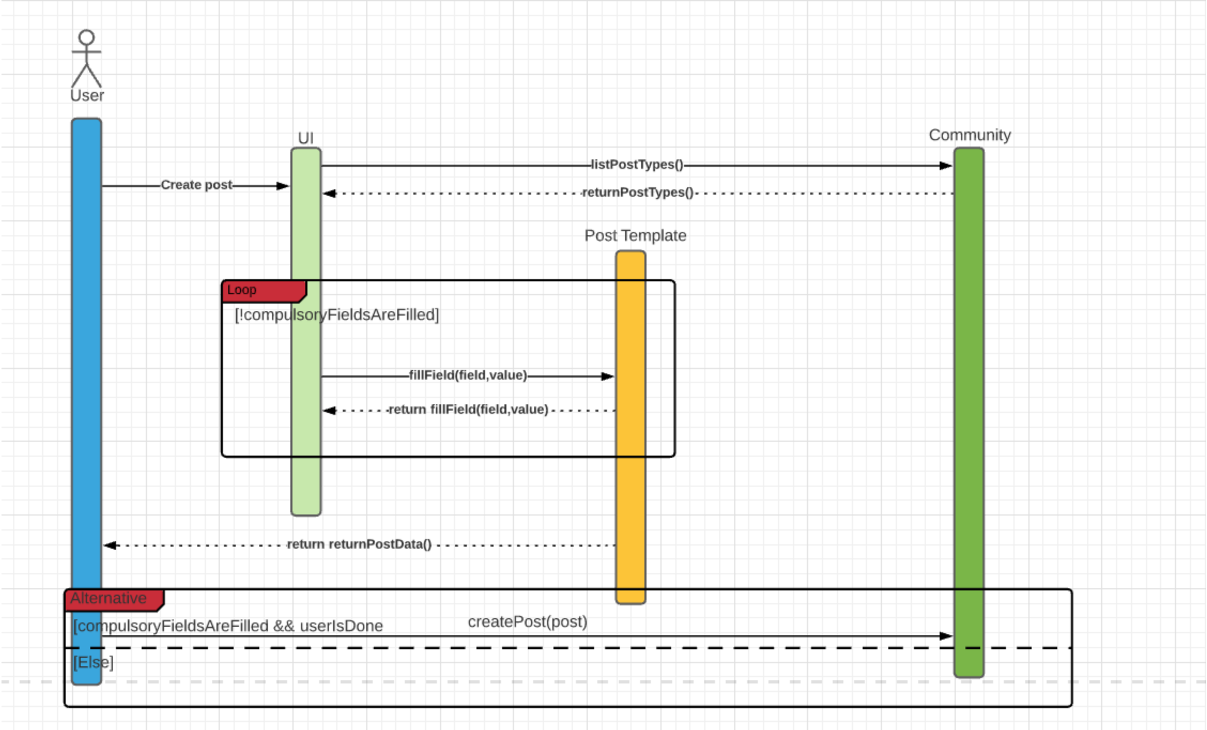
MODERATOR
SPECIFIC




Community Creator Specific
UI
Server
Everything moderator can do
*(*)
Delete Community(communityId)
Delete Community
Success/Fail
Promote User to Moderator
Promote User to Moderator(userId, communityId)
Success/Fail
Community C   or
Demote moderator to user
Demote moderator to user(userId,communityId)
Success/Fail

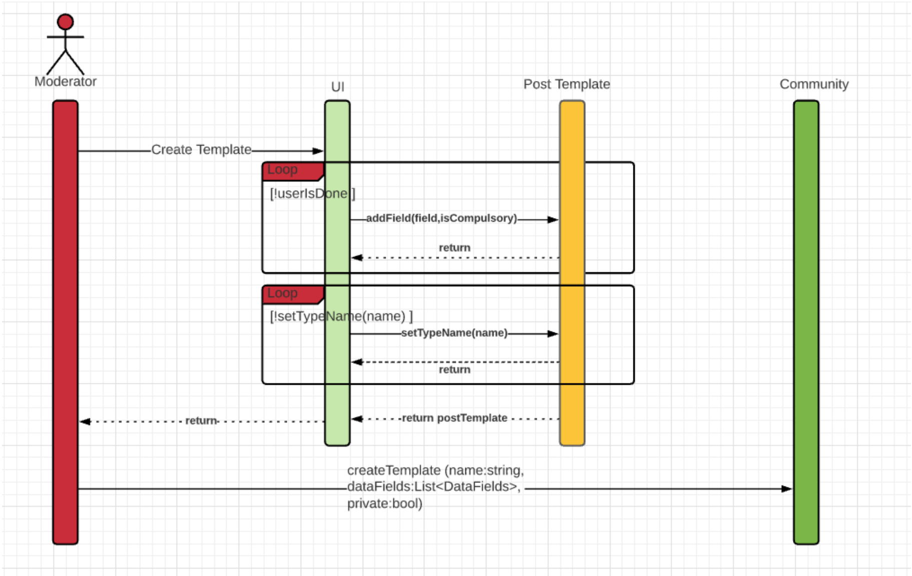
1. Creating Post in a Community

- User can create a post using existing post templates in community.



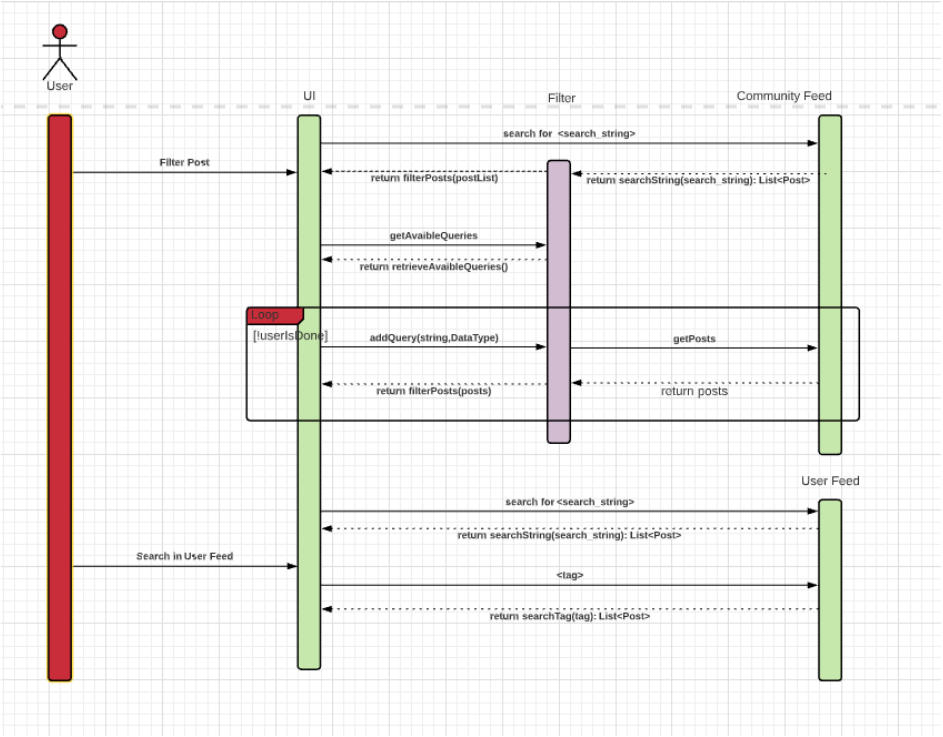
2. Creating Post Templates in a Community

- Moderators can create post templates in community.
- Moderators can add many fields to templates.

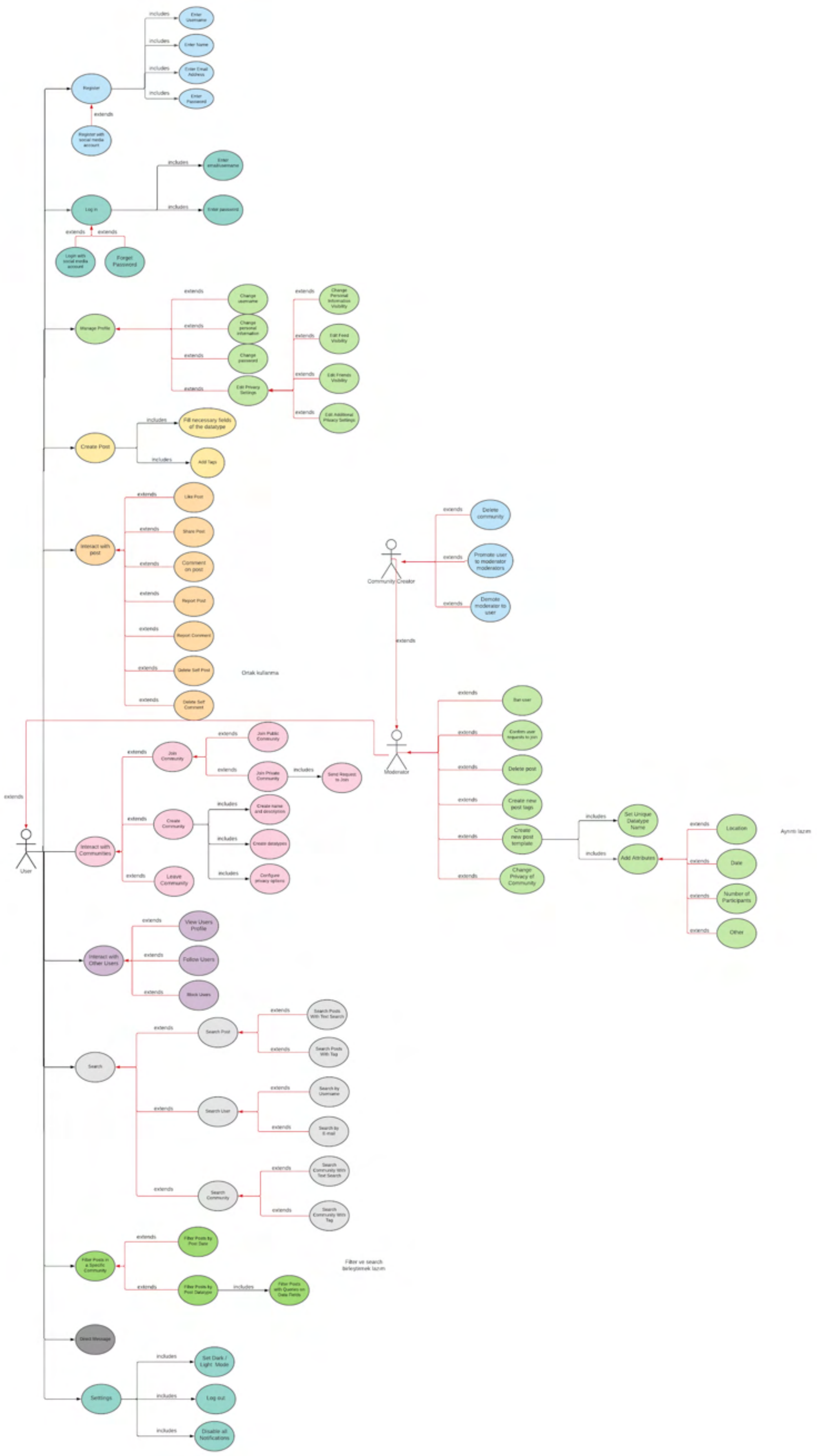


3. Filtering and Searching in Feeds

- Users can filter posts in communities.
- Users can search posts in their feed.


User
UI
Filter
Community Feed
search for <search_string>
Filter Post
return filterPosts(postList)
return searchString(search_string): List<Post>
getAvaibleQueries
return retrieveAvaibleQueries()
Loop
[!userIsDone]
addQuery(string,DataType)
getPosts
return filterPosts(posts)
return posts
User Feed
search for <search_string>
return searchString(search_string): List<Post>
Search in User Feed
<tag>
return searchTag(tag): List<Post>

3. Use Case Diagram



Ortak kullanma

Ayrım lazım

Filter ve search birleştirmek lazım

e.Project Plan and RAM

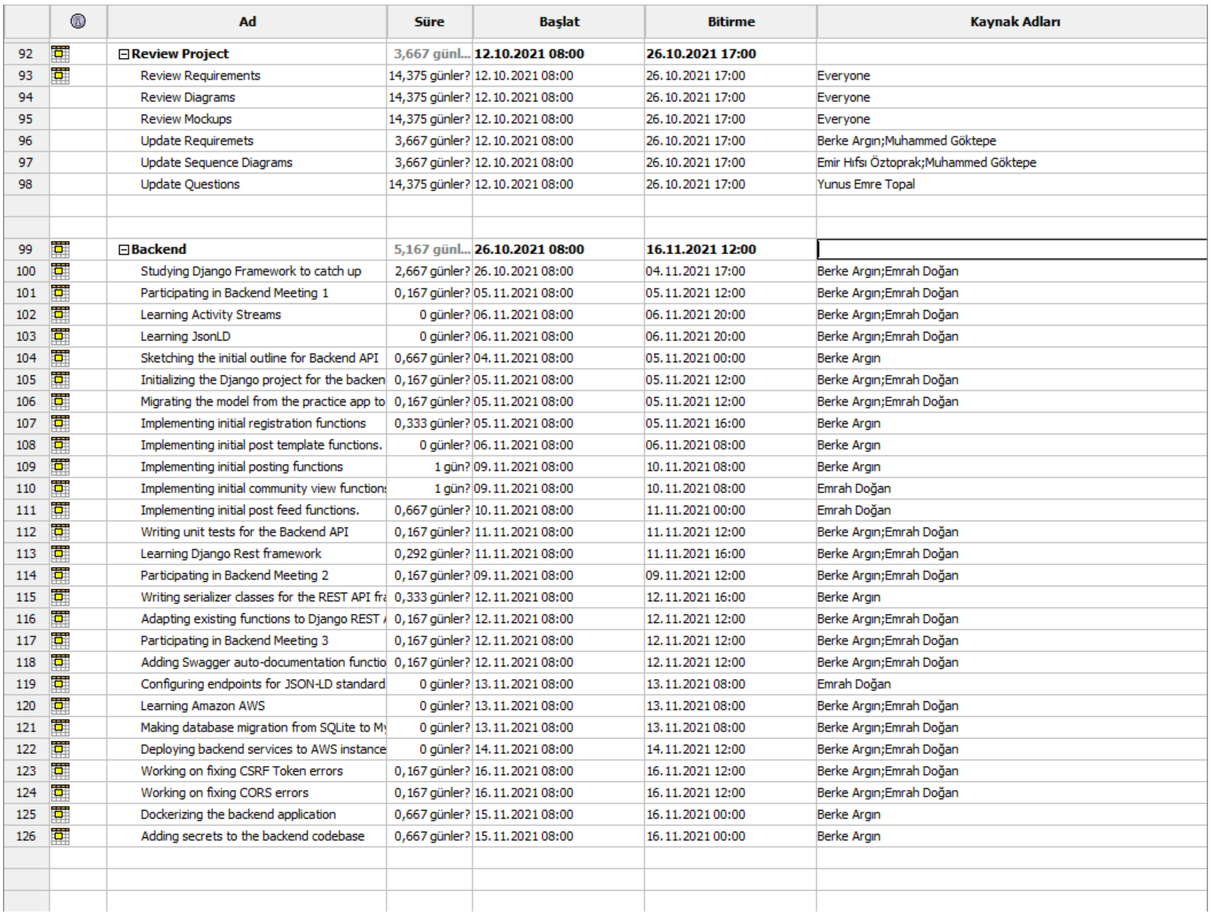
Green -> Lead Blue -> Secondary Yellow -> Contributor Magenta -> Approval Orange -> Review Red -> No contribution	Orkan Akisu	Elif AKALIN	Emrah DOĞAN	Berke ARGIN	Muhammed GÖKTEPE	Emir Hıfsı ÖZTOPRAK	Bilal TEKİN	Yunus Emre TOPAL
Initialization								
Reviewing the group repository		🟧	🟧	🟧	🟧	🟧	🟨	🟧
Refreshing knowledge on Github platform		🟧	🟧	🟧	🟧	🟧	🟨	🟧
Updating wiki page for the current semester		🟧	🟧	🟧	🟩	🟧	🟧	🟧
Requirement & Design Documents								
Reviewing UML diagrams		🟧	🟧	🟧	🟧	🟧	🟧	🟧
Reviewing requirements document from last semester		🟧	🟧	🟧	🟧	🟧	🟧	🟧
Reviewing mockups		🟧	🟧	🟧	🟧	🟥	🟥	🟥
Updating requirements document		🟧	🟧	🟩	🟦	🟧	🟧	🟨
Updating Sequence diagrams		🟧	🟧	🟥	🟨	🟩	🟥	🟧
Updating Customer Questions		🟧	🟧	🟥	🟥	🟥	🟧	🟩
Backend								
Studying Django Framework to catch up		🟥	🟩	🟩	🟥	🟥	🟨	🟥
Participating in Backend Meeting 1		🟥	🟦	🟩	🟥	🟥	🟨	🟥

Learning Activity Streams		🟥	🟩	🟦	🟥	🟥	🟥	🟥
Learning JSON-LD		🟥	🟩	🟦	🟥	🟥	🟥	🟥
Sketching the initial outline for Backend API		🟥	🟧	🟩	🟥	🟥	🟥	🟥
Initializing the Django project for the backend app		🟥	🟩	🟦	🟥	🟥	🟥	🟥
Migrating the model from the practice app to the new Django project.		🟥	🟩	🟦	🟥	🟥	🟥	🟥
Implementing initial registration functions		🟥	🟧	🟩	🟥	🟥	🟥	🟥
Implementing initial post template functions.		🟥	🟧	🟩	🟥	🟥	🟥	🟥
Implementing initial posting functions		🟥	🟧	🟩	🟥	🟥	🟥	🟥
Implementing initial community view functions		🟥	🟩	🟧	🟥	🟥	🟥	🟥
Implementing initial post feed functions.		🟥	🟩	🟧	🟥	🟥	🟥	🟥
Writing unit tests for the Backend API		🟥	🟨	🟨	🟥	🟥	🟥	🟥
Learning Django Rest framework		🟥	🟩	🟩	🟥	🟥	🟥	🟥
Participating in Backend Meeting 2		🟥	🟦	🟩	🟥	🟥	🟥	🟥
Writing serializer classes for the REST API framework		🟥	🟩	🟩	🟥	🟥	🟥	🟥
Adapting and reimplementing existing functions with respect to Django REST API framework.		🟥	🟩	🟩	🟥	🟥	🟥	🟥
Participating in Backend Meeting 3		🟥	🟩	🟦	🟥	🟥	🟥	🟥
Adding Swagger auto-documentation functionality to the code.		🟥	🟦	🟩	🟥	🟥	🟥	🟥
Configuring endpoints for JSON-LD standards		🟥	🟩	🟧	🟥	🟥	🟥	🟥
Learning Amazon AWS		🟥	🟩	🟦	🟥	🟥	🟨	🟥
Making database migration from SQLite to MySQL		🟥	🟦	🟩	🟥	🟥	🟥	🟥
Deploying backend services to AWS instance		🟥	🟩	🟦	🟥	🟥	🟨	🟥
Working on fixing CSRF Token errors		🟥	🟨	🟨	🟥	🟥	🟧	🟥
Working on fixing CORS errors		🟥	🟨	🟨	🟥	🟥	🟨	🟥
Dockerizing the backend application		🟥	🟥	🟩	🟥	🟥	🟨	🟥
Configuring secrets on backend codebase		🟥	🟥	🟩	🟥	🟥	🟥	🟥
Web Frontend								
Studying ReactJS-Bootstrap		🟥	🟥	🟥	🟨	🟨	🟩	🟥

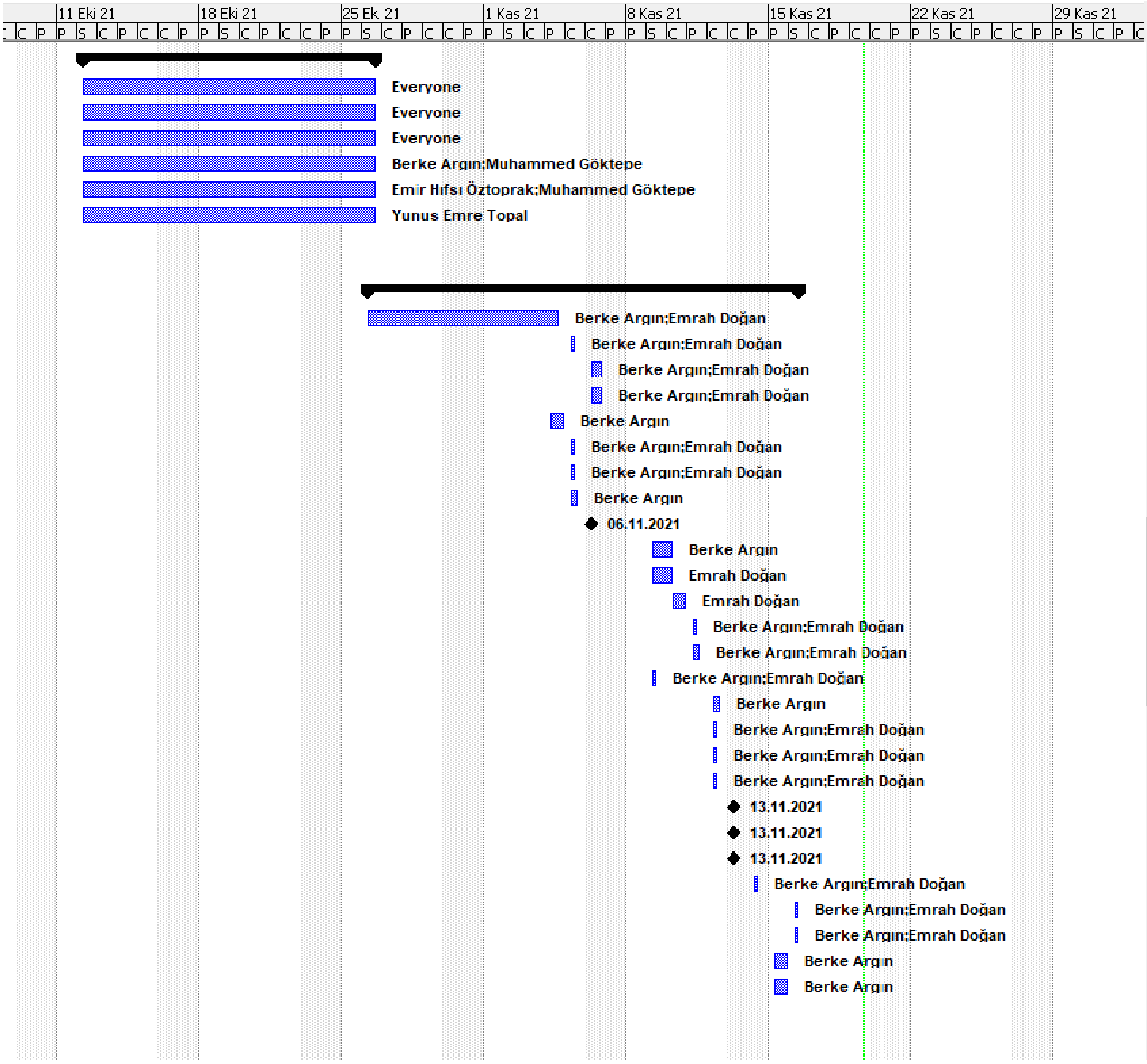
Participating in Frontend Meeting 1		🟥	🟥	🟥	🟨	🟨	🟨	🟥
Learning React Hooks		🟥	🟧	🟥	🟨	🟨	🟩	🟥
Learning React Axios		🟥	🟧	🟥	🟨	🟨	🟩	🟥
Learning React Navigation		🟥	🟥	🟥	🟨	🟨	🟩	🟥
Learning React Redux-thunk		🟥	🟥	🟥	🟨	🟨	🟩	🟥
Participating in Frontend Meeting 2		🟥	🟥	🟥	🟨	🟨	🟨	🟥
Creating React App		🟥	🟥	🟥	🟥	🟥	🟩	🟥
Creating Register Page		🟥	🟥	🟥	🟩	🟥	🟧	🟥
Creating Login Page		🟥	🟥	🟥	🟩	🟥	🟧	🟥
Creating Create Post Template Page		🟥	🟥	🟥	🟦	🟦	🟧	🟥
Creating Create Post Page		🟥	🟥	🟥	🟩	🟦	🟧	🟥
Creating My Posts Page		🟥	🟥	🟥	🟩	🟥	🟧	🟥
Creating Communities Page		🟥	🟥	🟥	🟥	🟩	🟧	🟥
Creating Feed Page		🟥	🟥	🟥	🟦	🟧	🟧	🟥
Creating Create Community Page		🟥	🟥	🟥	🟩	🟧	🟧	🟥
Creating My Communities Page		🟥	🟥	🟥	🟥	🟩	🟧	🟥
Creating Sidebar		🟥	🟥	🟥	🟧	🟧	🟩	🟥
Backend Integration		🟥	🟧	🟧	🟧	🟧	🟩	🟥
Login Backend Integration		🟥	🟧	🟧	🟧	🟧	🟩	🟥
Register Backend Integration		🟥	🟧	🟧	🟧	🟧	🟩	🟥
Communities Backend Integration		🟥	🟧	🟧	🟧	🟧	🟩	🟥
Deployment of Frontend to AWS		🟥	🟥	🟥	🟧	🟧	🟩	🟥
Mobile Frontend								
Learning React Native		🟩	🟥	🟥	🟥	🟥	🟥	🟩
Creating Initial App		🟧	🟥	🟥	🟥	🟥	🟥	🟩
Creating Welcome Screen		🟨	🟥	🟥	🟥	🟥	🟥	🟩
Creating Login Screen		🟨	🟥	🟥	🟥	🟥	🟥	🟩
Implementing API Call for Login Feature		🟨	🟥	🟥	🟥	🟥	🟥	🟩
Creating Registration Screen		🟨	🟥	🟥	🟥	🟥	🟥	🟩

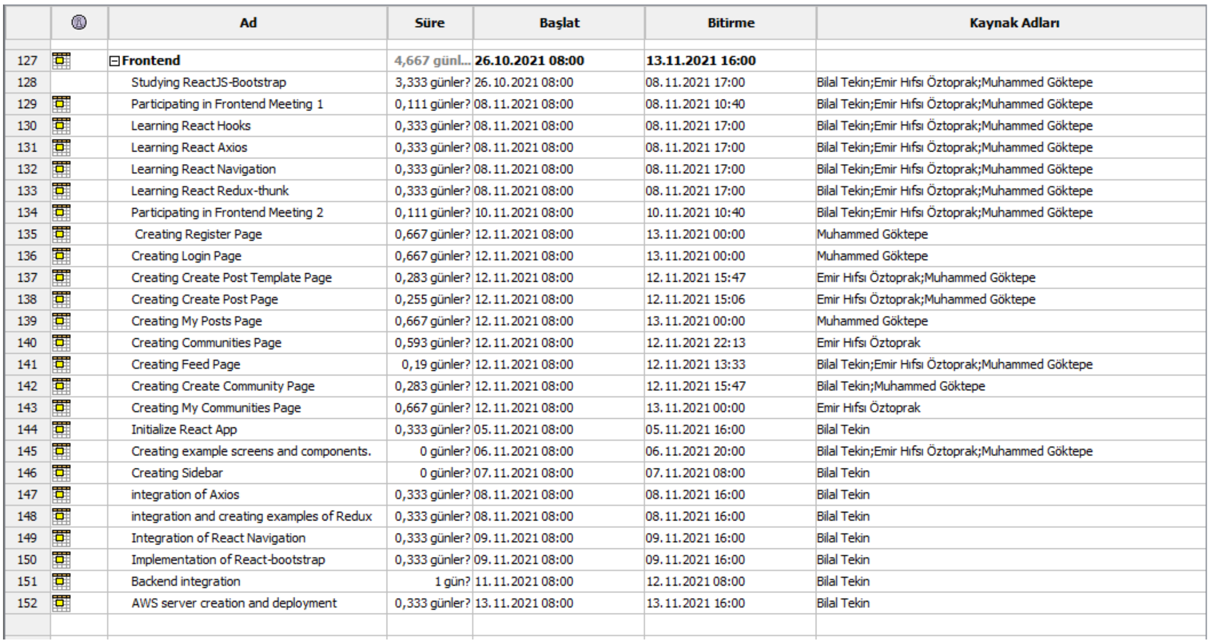
Implementing API call for Registration feature		🟧	🟥	🟥	🟥	🟥	🟥	🟩
Creating Feed Screen		🟧	🟥	🟥	🟥	🟥	🟥	🟩
Implementing API call for getting Feed data		🟧	🟥	🟥	🟥	🟥	🟨	🟩
Creating Communities Screen		🟨	🟥	🟥	🟥	🟥	🟥	🟩
Implementing API call for getting Communities		🟩	🟥	🟥	🟥	🟥	🟨	🟧
Creating User Posts Screen		🟨	🟥	🟥	🟥	🟥	🟥	🟩
Implementing API call for getting User Posts		🟩	🟧	🟥	🟥	🟥	🟨	🟩
Creating User Communities Screen		🟨	🟥	🟥	🟥	🟥	🟥	🟩
Implementing API call for getting User Communities		🟩	🟧	🟥	🟥	🟥	🟨	🟧
Creating Settings Screen		🟧	🟥	🟥	🟥	🟥	🟥	🟩
Creating Post Screen		🟨	🟥	🟥	🟥	🟥	🟥	🟩
Creating Community Screen		🟨	🟥	🟥	🟥	🟥	🟥	🟩
Creating Drawer Navigation		🟧	🟥	🟥	🟥	🟥	🟥	🟩
Creating Axios Service		🟩	🟥	🟥	🟥	🟥	🟨	🟧

7.RoadMap

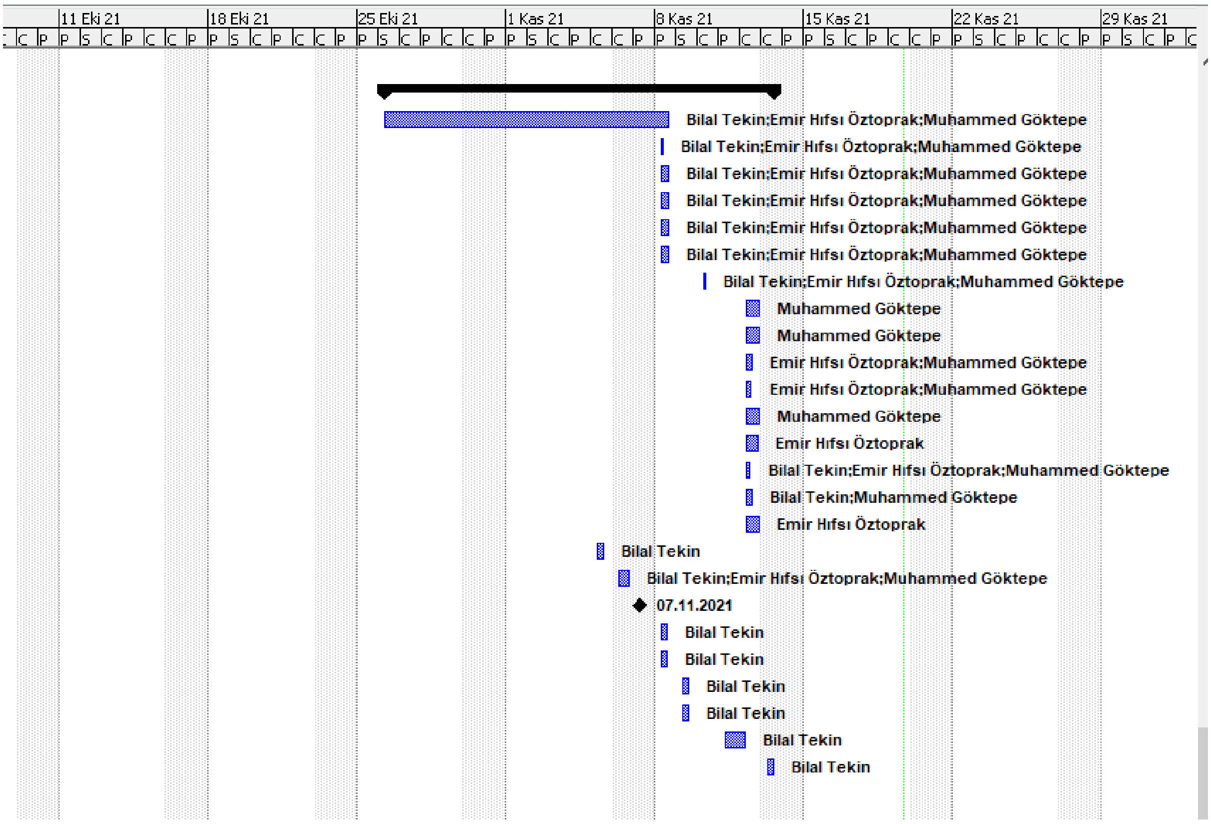


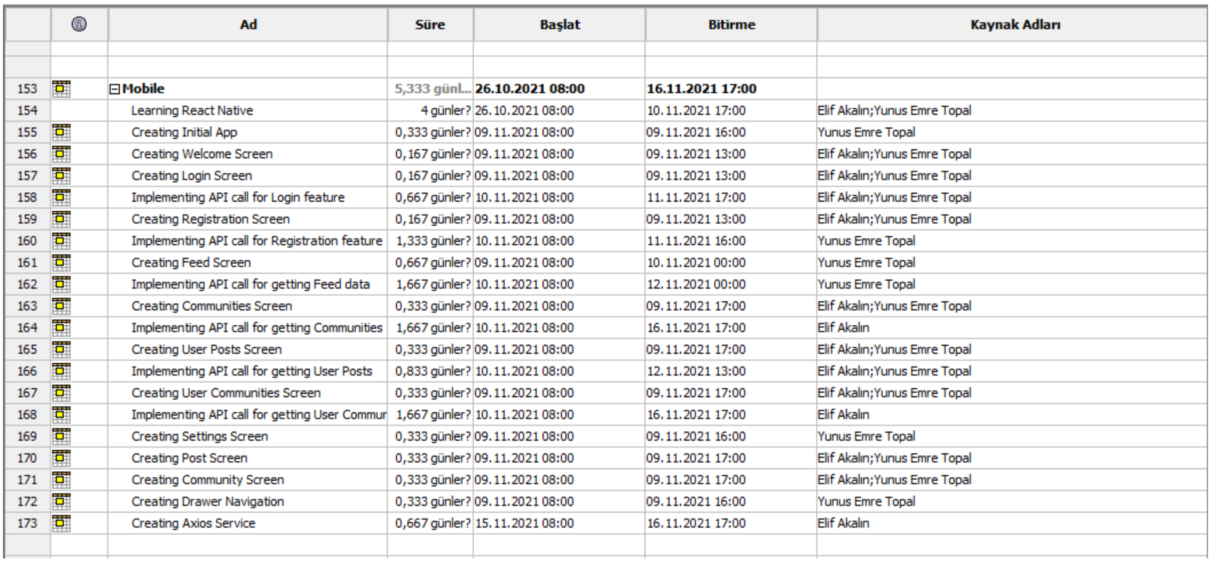
	ⓘ	Ad	Süre	Başlat	Bitirme	Kaynak Adları
92	📅	⊟ **Review Project**	3,667 günl...	**12.10.2021 08:00**	**26.10.2021 17:00**	
93	📅	Review Requirements	14,375 günler?	12.10.2021 08:00	26.10.2021 17:00	Everyone
94		Review Diagrams	14,375 günler?	12.10.2021 08:00	26.10.2021 17:00	Everyone
95		Review Mockups	14,375 günler?	12.10.2021 08:00	26.10.2021 17:00	Everyone
96		Update Requiremets	3,667 günler?	12.10.2021 08:00	26.10.2021 17:00	Berke Argın;Muhammed Göktepe
97		Update Sequence Diagrams	3,667 günler?	12.10.2021 08:00	26.10.2021 17:00	Emir Hıfsı Öztoprak;Muhammed Göktepe
98		Update Questions	14,375 günler?	12.10.2021 08:00	26.10.2021 17:00	Yunus Emre Topal
99	📅	⊟ **Backend**	5,167 günl...	**26.10.2021 08:00**	**16.11.2021 12:00**	
100	📅	Studying Django Framework to catch up	2,667 günler?	26.10.2021 08:00	04.11.2021 17:00	Berke Argın;Emrah Doğan
101	📅	Participating in Backend Meeting 1	0,167 günler?	05.11.2021 08:00	05.11.2021 12:00	Berke Argın;Emrah Doğan
102	📅	Learning Activity Streams	0 günler?	06.11.2021 08:00	06.11.2021 20:00	Berke Argın;Emrah Doğan
103	📅	Learning JsonLD	0 günler?	06.11.2021 08:00	06.11.2021 20:00	Berke Argın;Emrah Doğan
104	📅	Sketching the initial outline for Backend API	0,667 günler?	04.11.2021 08:00	05.11.2021 00:00	Berke Argın
105	📅	Initializing the Django project for the backen	0,167 günler?	05.11.2021 08:00	05.11.2021 12:00	Berke Argın;Emrah Doğan
106	📅	Migrating the model from the practice app to	0,167 günler?	05.11.2021 08:00	05.11.2021 12:00	Berke Argın;Emrah Doğan
107	📅	Implementing initial registration functions	0,333 günler?	05.11.2021 08:00	05.11.2021 16:00	Berke Argın
108	📅	Implementing initial post template functions.	0 günler?	06.11.2021 08:00	06.11.2021 08:00	Berke Argın
109	📅	Implementing initial posting functions	1 gün?	09.11.2021 08:00	10.11.2021 08:00	Berke Argın
110	📅	Implementing initial community view function	1 gün?	09.11.2021 08:00	10.11.2021 08:00	Emrah Doğan
111	📅	Implementing initial post feed functions.	0,667 günler?	10.11.2021 08:00	11.11.2021 00:00	Emrah Doğan
112	📅	Writing unit tests for the Backend API	0,167 günler?	11.11.2021 08:00	11.11.2021 12:00	Berke Argın;Emrah Doğan
113	📅	Learning Django Rest framework	0,292 günler?	11.11.2021 08:00	11.11.2021 16:00	Berke Argın;Emrah Doğan
114	📅	Participating in Backend Meeting 2	0,167 günler?	09.11.2021 08:00	09.11.2021 12:00	Berke Argın;Emrah Doğan
115	📅	Writing serializer classes for the REST API fr	0,333 günler?	12.11.2021 08:00	12.11.2021 16:00	Berke Argın
116	📅	Adapting existing functions to Django REST	0,167 günler?	12.11.2021 08:00	12.11.2021 12:00	Berke Argın;Emrah Doğan
117	📅	Participating in Backend Meeting 3	0,167 günler?	12.11.2021 08:00	12.11.2021 12:00	Berke Argın;Emrah Doğan
118	📅	Adding Swagger auto-documentation functio	0,167 günler?	12.11.2021 08:00	12.11.2021 12:00	Berke Argın;Emrah Doğan
119	📅	Configuring endpoints for JSON-LD standard	0 günler?	13.11.2021 08:00	13.11.2021 08:00	Emrah Doğan
120	📅	Learning Amazon AWS	0 günler?	13.11.2021 08:00	13.11.2021 08:00	Berke Argın;Emrah Doğan
121	📅	Making database migration from SQLite to M	0 günler?	13.11.2021 08:00	13.11.2021 08:00	Berke Argın;Emrah Doğan
122	📅	Deploying backend services to AWS instance	0 günler?	14.11.2021 08:00	14.11.2021 12:00	Berke Argın;Emrah Doğan
123	📅	Working on fixing CSRF Token errors	0,167 günler?	16.11.2021 08:00	16.11.2021 12:00	Berke Argın;Emrah Doğan
124	📅	Working on fixing CORS errors	0,167 günler?	16.11.2021 08:00	16.11.2021 12:00	Berke Argın;Emrah Doğan
125	📅	Dockerizing the backend application	0,667 günler?	15.11.2021 08:00	16.11.2021 00:00	Berke Argın
126	📅	Adding secrets to the backend codebase	0,667 günler?	15.11.2021 08:00	16.11.2021 00:00	Berke Argın



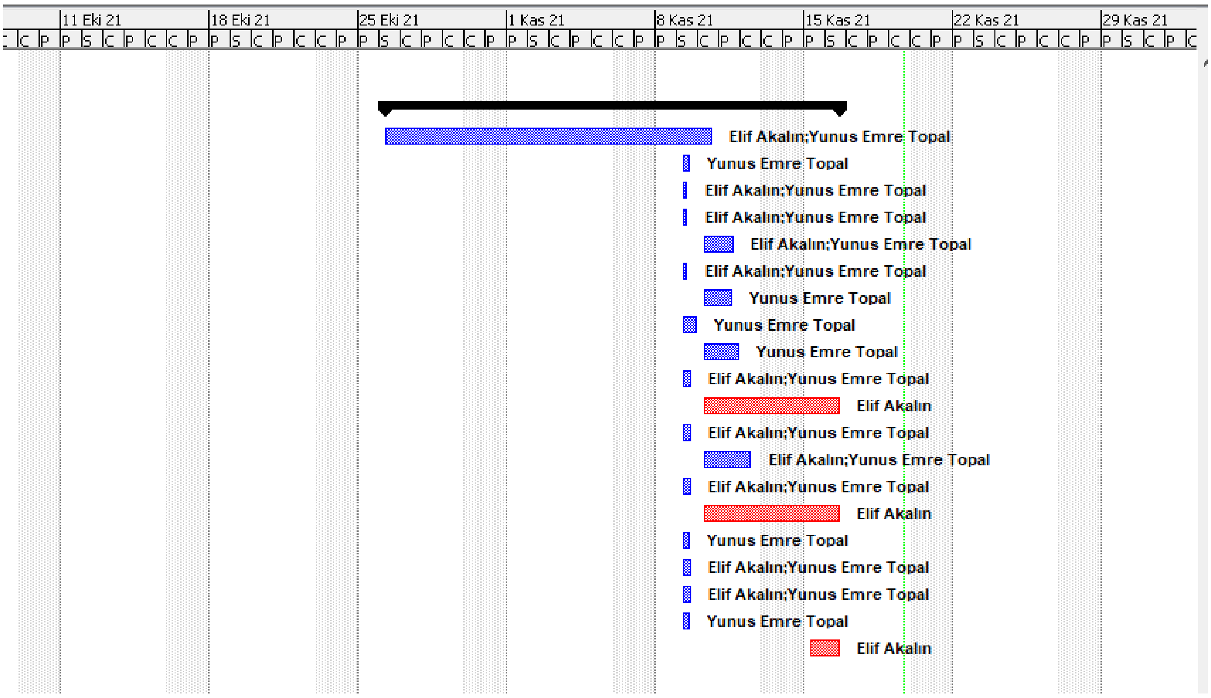


	ⓘ	Ad	Süre	Başlat	Bitirme	Kaynak Adları
127	▦	⊟ **Frontend**	4,667 günl...	**26.10.2021 08:00**	**13.11.2021 16:00**	
128		Studying ReactJS-Bootstrap	3,333 günler?	26.10.2021 08:00	08.11.2021 17:00	Bilal Tekin;Emir Hıfsı Öztoprak;Muhammed Göktepe
129	▦	Participating in Frontend Meeting 1	0,111 günler?	08.11.2021 08:00	08.11.2021 10:40	Bilal Tekin;Emir Hıfsı Öztoprak;Muhammed Göktepe
130	▦	Learning React Hooks	0,333 günler?	08.11.2021 08:00	08.11.2021 17:00	Bilal Tekin;Emir Hıfsı Öztoprak;Muhammed Göktepe
131	▦	Learning React Axios	0,333 günler?	08.11.2021 08:00	08.11.2021 17:00	Bilal Tekin;Emir Hıfsı Öztoprak;Muhammed Göktepe
132	▦	Learning React Navigation	0,333 günler?	08.11.2021 08:00	08.11.2021 17:00	Bilal Tekin;Emir Hıfsı Öztoprak;Muhammed Göktepe
133	▦	Learning React Redux-thunk	0,333 günler?	08.11.2021 08:00	08.11.2021 17:00	Bilal Tekin;Emir Hıfsı Öztoprak;Muhammed Göktepe
134	▦	Participating in Frontend Meeting 2	0,111 günler?	10.11.2021 08:00	10.11.2021 10:40	Bilal Tekin;Emir Hıfsı Öztoprak;Muhammed Göktepe
135	▦	Creating Register Page	0,667 günler?	12.11.2021 08:00	13.11.2021 00:00	Muhammed Göktepe
136	▦	Creating Login Page	0,667 günler?	12.11.2021 08:00	13.11.2021 00:00	Muhammed Göktepe
137	▦	Creating Create Post Template Page	0,283 günler?	12.11.2021 08:00	12.11.2021 15:47	Emir Hıfsı Öztoprak;Muhammed Göktepe
138	▦	Creating Create Post Page	0,255 günler?	12.11.2021 08:00	12.11.2021 15:06	Emir Hıfsı Öztoprak;Muhammed Göktepe
139	▦	Creating My Posts Page	0,667 günler?	12.11.2021 08:00	13.11.2021 00:00	Muhammed Göktepe
140	▦	Creating Communities Page	0,593 günler?	12.11.2021 08:00	12.11.2021 22:13	Emir Hıfsı Öztoprak
141	▦	Creating Feed Page	0,19 günler?	12.11.2021 08:00	12.11.2021 13:33	Bilal Tekin;Emir Hıfsı Öztoprak;Muhammed Göktepe
142	▦	Creating Create Community Page	0,283 günler?	12.11.2021 08:00	12.11.2021 15:47	Bilal Tekin;Muhammed Göktepe
143	▦	Creating My Communities Page	0,667 günler?	12.11.2021 08:00	13.11.2021 00:00	Emir Hıfsı Öztoprak
144	▦	Initialize React App	0,333 günler?	05.11.2021 08:00	05.11.2021 16:00	Bilal Tekin
145	▦	Creating example screens and components.	0 günler?	06.11.2021 08:00	06.11.2021 20:00	Bilal Tekin;Emir Hıfsı Öztoprak;Muhammed Göktepe
146	▦	Creating Sidebar	0 günler?	07.11.2021 08:00	07.11.2021 08:00	Bilal Tekin
147	▦	integration of Axios	0,333 günler?	08.11.2021 08:00	08.11.2021 16:00	Bilal Tekin
148	▦	integration and creating examples of Redux	0,333 günler?	08.11.2021 08:00	08.11.2021 16:00	Bilal Tekin
149	▦	Integration of React Navigation	0,333 günler?	09.11.2021 08:00	09.11.2021 16:00	Bilal Tekin
150	▦	Implementation of React-bootstrap	0,333 günler?	09.11.2021 08:00	09.11.2021 16:00	Bilal Tekin
151	▦	Backend integration	1 gün?	11.11.2021 08:00	12.11.2021 08:00	Bilal Tekin
152	▦	AWS server creation and deployment	0,333 günler?	13.11.2021 08:00	13.11.2021 16:00	Bilal Tekin





	ⓘ	Ad	Süre	Başlat	Bitirme	Kaynak Adları
153	▦	⊟ **Mobile**	5,333 günl...	**26.10.2021 08:00**	**16.11.2021 17:00**	
154		Learning React Native	4 günler?	26.10.2021 08:00	10.11.2021 17:00	Elif Akalın;Yunus Emre Topal
155	▦	Creating Initial App	0,333 günler?	09.11.2021 08:00	09.11.2021 16:00	Yunus Emre Topal
156	▦	Creating Welcome Screen	0,167 günler?	09.11.2021 08:00	09.11.2021 13:00	Elif Akalın;Yunus Emre Topal
157	▦	Creating Login Screen	0,167 günler?	09.11.2021 08:00	09.11.2021 13:00	Elif Akalın;Yunus Emre Topal
158	▦	Implementing API call for Login feature	0,667 günler?	10.11.2021 08:00	11.11.2021 17:00	Elif Akalın;Yunus Emre Topal
159	▦	Creating Registration Screen	0,167 günler?	09.11.2021 08:00	09.11.2021 13:00	Elif Akalın;Yunus Emre Topal
160	▦	Implementing API call for Registration feature	1,333 günler?	10.11.2021 08:00	11.11.2021 16:00	Yunus Emre Topal
161	▦	Creating Feed Screen	0,667 günler?	09.11.2021 08:00	10.11.2021 00:00	Yunus Emre Topal
162	▦	Implementing API call for getting Feed data	1,667 günler?	10.11.2021 08:00	12.11.2021 00:00	Yunus Emre Topal
163	▦	Creating Communities Screen	0,333 günler?	09.11.2021 08:00	09.11.2021 17:00	Elif Akalın;Yunus Emre Topal
164	▦	Implementing API call for getting Communities	1,667 günler?	10.11.2021 08:00	16.11.2021 17:00	Elif Akalın
165	▦	Creating User Posts Screen	0,333 günler?	09.11.2021 08:00	09.11.2021 17:00	Elif Akalın;Yunus Emre Topal
166	▦	Implementing API call for getting User Posts	0,833 günler?	10.11.2021 08:00	12.11.2021 13:00	Elif Akalın;Yunus Emre Topal
167	▦	Creating User Communities Screen	0,333 günler?	09.11.2021 08:00	09.11.2021 17:00	Elif Akalın;Yunus Emre Topal
168	▦	Implementing API call for getting User Commur	1,667 günler?	10.11.2021 08:00	16.11.2021 17:00	Elif Akalın
169	▦	Creating Settings Screen	0,333 günler?	09.11.2021 08:00	09.11.2021 16:00	Yunus Emre Topal
170	▦	Creating Post Screen	0,333 günler?	09.11.2021 08:00	09.11.2021 17:00	Elif Akalın;Yunus Emre Topal
171	▦	Creating Community Screen	0,333 günler?	09.11.2021 08:00	09.11.2021 17:00	Elif Akalın;Yunus Emre Topal
172	▦	Creating Drawer Navigation	0,333 günler?	09.11.2021 08:00	09.11.2021 16:00	Yunus Emre Topal
173	▦	Creating Axios Service	0,667 günler?	15.11.2021 08:00	16.11.2021 17:00	Elif Akalın



Tabular Format

Backend

Task	Start Date	End Date	Assignees
Studying Django Framework to catch up	26.10	04.11	Berke, Emrah
Participating in Backend Meeting 1	05.11	05.11	Berke, Emrah
Learning Activity Streams	06.11	10.11	Berke, Emrah
Learning JsonLD	06.11	10.11	Berke, Emrah
Sketching the initial outline for Backend API	04.11	05.11	Berke
Initializing the Django project for the backend app	05.11	05.11	Berke, Emrah
Migrating the model from the practice app to the new Django project.	05.11	06.11	Berke, Emrah
Implementing initial registration functions	06.11	07.11	Berke
Implementing initial post template	09.11	11.11	Berke

functions.			
Implementing initial posting functions	09.11	11.11	Berke
Implementing initial community view functions	10.11	11.11	Emrah
Implementing initial post feed functions.	11.11	11.11	Emrah
Writing unit tests for the Backend API	09.11	12.11	Berke, Emrah
Learning Django Rest framework	11.11	12.11	Berke, Emrah
Participating in Backend Meeting 2	09.11	09.11	Berke, Emrah
Writing serializer classes for the REST API framework	12.11	12.11	Berke
Adapting existing functions to Django REST API framework.	12.11	13.11	Berke, Emrah
Participating in Backend Meeting 3	12.11	12.11	Berke, Emrah
Adding Swagger auto-documentation functionality to the code.	13.11	13.11	Berke, Emrah
Configuring endpoints for JSON-LD standards	13.11	13.11	Emrah
Learning Amazon AWS	13.11	14.11	Berke, Emrah
Making database migration from SQLite to MySQL	13.11	13.11	Berke, Emrah
Deploying backend services to AWS instance	14.11	15.11	Berke, Emrah
Working on fixing CSRF Token errors	16.11	16.11	Berke, Emrah
Working on fixing CORS errors	16.11	16.11	Berke, Emrah

Dockerizing the backend application	15.11	16.11	Berke
Adding secrets to the backend codebase	15.11	16.11	Berke

Frontend - Web

Studying ReactJS-Bootstrap	26.10	08.11	Bilal, Muhammed, Emir
Participating in Frontend Meeting 1	08.11	08.11	Bilal, Muhammed, Emir
Learning React Hooks	08.11	10.11	Bilal, Muhammed, Emir
Learning React Axios	08.11	10.11	Bilal, Muhammed, Emir
Learning React Navigation	08.11	10.11	Bilal, Muhammed, Emir
Learning React Redux-thunk	08.11	10.11	Bilal, Muhammed, Emir
Participating in Frontend Meeting 2	10.11	10.11	Bilal, Muhammed, Emir
Creating Register Page	12.11	15.11	Muhammed
Creating Login Page	12.11	15.11	Muhammed
Creating Create Post Template Page	12.11	15.11	Muhammed, Emir
Creating Create Post Page	12.11	15.11	Muhammed, Emir
Creating My Posts Page	12.11	15.11	Muhammed
Creating Communities Page	12.11	15.11	Emir
Creating Feed Page	12.11	15.11	Muhammed, Emir,Bilal
Creating Create Community Page	12.11	15.11	Muhammed, Bilal
Creating My Communities Page	12.11	15.11	Emir
Initialize React App	05.11	05.11	Bilal

Creating example screens and components.	06.11	15.11	Muhammed, Emir, Bilal
Creating Sidebar	07.11	07.11	Bilal
integration of Axios	08.11	08.11	Bilal
integration and creating examples of Redux	08.11	08.11	Bilal
Integration of React Navigation	09.11	09.11	Bilal
Implementation of React-bootstrap	09.11	09.11	Bilal
Backend integration	11.11	15.11	Bilal
AWS server creation and deployment	13.11	15.11	Bilal

Frontend - Mobile

Learning React Native	26.10	10.11	Yunus Emre, Elif
Creating Initial App	09.11	09.11	Yunus Emre,
Creating Welcome Screen	09.11	09.11	Yunus Emre, Elif
Creating Login Screen	09.11	09.11	Yunus Emre, Elif
Implementing API call for Login feature	10.11	15.11	Yunus Emre, Elif
Creating Registration Screen	09.11	09.11	Yunus Emre, Elif
Implementing API call for Registration feature	10.11	15.11	Yunus Emre,
Creating Feed Screen	09.11	10.11	Yunus Emre,
Implementing API call for getting Feed data	10.11	16.11	Yunus Emre,
Creating Communities Screen	09.11	10.11	Yunus Emre, Elif
Implementing API call for getting Communities	10.11	16.11	Elif
Creating User Posts	09.11	10.11	Yunus Emre, Elif

Screen			
Implementing API call for getting User Posts	10.11	16.11	Yunus Emre, Elif
Creating User Communities Screen	09.11	10.11	Yunus Emre, Elif
Implementing API call for getting User Communities	10.11	16.11	Elif
Creating Settings Screen	09.11	09.11	Yunus Emre,
Creating Post Screen	09.11	10.11	Yunus Emre, Elif
Creating Community Screen	09.11	10.11	Yunus Emre, Elif
Creating Drawer Navigation	09.11	09.11	Yunus Emre,
Creating Axios Service	15.11	16.11	Elif