Milestone Report 2

Table of Contents

Executive Summary

Summary of the Project

This project is a social media platform which lets people benefit from cumulation of content while they are sharing experiences and knowledge. The main building block of the platform is communities. The communities allow people with similar interests and purposes to contact each other. Users can share and reach contents of common interests in the communities. There are different types of users with different privileges. Community moderators are the ones who created the community or are assigned by another community moderator. They can create data types so that the users post community specific contents. They are able to remove posts, comments or users from their communities if needed. They can also restrict access to the community via making it private. Private communities are only accessible to the community members and common users can join them only if the community moderators accept their requests. However, common users are able to join the public communities without a request. Users can post contents, comment and like the existing posts and report the inappropriate ones to the moderators. The things that are offered to the users are not limited to these. The platform provides search and filter mechanisms to the users so that they do not have difficulties with finding the communities and the users they want to interact with. They are able to follow other users, visit their profiles and message them privately. The platform, of course, does not disregard the privacy of its users. It allows users to change their privacy settings and let them decide who can follow, message and view their profiles. All in all, this application aims to reach the next level of what we know as social media by letting self-structured communities be formed by themselves.

In this milestone, we developed a practice app as a guideline to the future implementation of our social media platform using the things we have learned in our lectures and PS hours. Spring Boot framework was used to implement the application. Each group member developed functions for our API and used external APIs to include in the application for practice. Unit tests for the API functions were also performed by group members to detect possible defects and to make sure everything works as desired. Swagger was used to test and document our RESTful API as a whole. A basic user interface was developed to make end users interact with our application. Since each group member had to contribute to the code, the problems with the code were inevitable. The problems were easily handled with effective use of Git tools like code reviews and pull requests. The application was dockerized to "produce complete and executable version" of it. Lastly, the docker image of the application was deployed into the AWS machine.

API URL: http://34.238.122.101:8080/

Frontend URL : http://34.238.122.101:8080/home

Documentation: http://34.238.122.101:8080/swagger-ui/#

Overall Status

In this Milestone, we have started to work on the implementation part. Our main goal was to implement a practice app that provides basic functionalities using GET and POST methods. As a starting point there were many decisions to take such as which framework to use for the backend and which database to use. After the long discussions at meetings and having research on the difference between the frameworks, we decided to use MongoDB for the database and Spring Boot for the backend. At later steps we decided to use Thymeleaf for the frontend part.

After deciding the frameworks, some of our team members shared some resources to learn those frameworks. We studied for a while. Then we set a deadline for the backend part. As a group we decided which endpoints to implement and distributed them to everyone. Each of us also had research on 3rd party APIs and chose one of them to use in the project. Some of the functions that consume 3rd party API were directly related to our API functions yet the others provide extra features for our app.

Each team member opened a new branch and pushed his or her code with unit tests to that branch following the shared repository model of Github. Then opened a pull request to merge into the dev branch and assigned reviewers. Each reviewer gave some suggestion, requested changes or made comments on the code. Modifications are made accordingly or the branch directly merged to the dev branch if everything is implemented well.

After implementing the large part of the backend we discussed how to design the frontend and we decided on the pages that should be implemented. We again divided the pages among the team. A new branch named frontend was opened and we worked on the frontend at that branch. After all implementations are completed we merged it to the dev branch. At final we dockerized the app and created a tag in Github as v.0.1.

During the implementation process we used Git and experienced lots of scenarios in Git like working in branches, reviewing, merging and handling conflicts. Therefore in this milestone our work was more dependent on each other. So we had a stronger communication and worked by consulting each other. We also arranged a meeting with our TA and we received feedback about the first milestone. After those feedbacks we tried to improve our work in this milestone.

For now our app provides the basic functionalities and extra features and we hope it will be useful in the next semester. Even if we start over again, we believe our experience of working together in consistency will help us to progress quickly.

Challenges

As we did in all of our other courses, we suffered from the difficulties the pandemic has brought to our lives throughout this milestone. It was impossible to meet when we needed to discuss something in person. We even had difficulties setting up an online meeting due to our unstable agendas. Luckily, we managed to help and inform each other all the time by using the communication tools efficiently. Second major challenge we encountered in this milestone was deciding the framework to use. There was no framework that all of us are familiar with unsurprisingly. We chose the one majority of our group know well enough, Spring Boot. Competent ones provided tutorials and a demo to other members of our group so that they feel comfortable to work with the framework. Last major challenge was the conflicts, which are inherent in these kinds of projects. We did not have a difficult time to handle them thanks to the Git PS hours provided to us prior to the implementation. During these challenges ,we experienced how it is to be a part of a software development and learned about what kind of problems one can encounter during the process.

Moving Forward

Since milestone-1 we have gained great experience both individually and as a group. Each of us have taken responsibility to learn and use new tools to implement new features to our practice app. Through the development of the main project we will be taking advantage of these newly learned skills. With that fresh knowledge it will be easier for us to compare and decide on the tools we may use to improve both efficiency and quality of the main project once we start working on it. So, we will have further discussions about what tools and languages to use for the main project. For the main project we will be doing mobile, front-end, and back-end development. So, we will be assigning each member to one of these three roles. For the back-end we will be implementing the required functionalities and expanding our unit tests. For mobile and front-end we are planning to provide all the requirements and create a UI that is consistent within mobile and web. Also, we will decide on the name and design a logo for our app. As we did through these milestones, we will keep the communication strong and continue having meetings when needed to make sure that everyone is on the same page. We are looking forward to improving ourselves and creating a great project for the next semester.

Basic Functionality of the Project

As stated in previous sections, our project aims to create a social media platform where users can post content in organized communities. The functionalities of the project differs among user types. There are common users, community members, community moderators, and project admins. In order to become a common user people have to register to the app through multiple options. Once a user is registered they can start joining communities of their interest. If a community is public, users can join them directly and see the posts shared in that community in their feed. If a community is private, users have to send a request to moderators to join that community, then they can access and create posts there.

The communities are created and ruled by their moderators. A moderator has the permission to change the name, topics, rules, logo etc. of the community they moderate. They can remove the posts shared in their communities if one violates the community rules and ban users from posting in their community. A moderator can also create something called “Data Types” for their community. A data type provides templates for users to share their content in an organized manner with tags and fields.

Users can change their own profile and the privacy settings of the data they share in their profile. Users can follow other users and communicate with each other via a direct message functionality. Users can search/filter for posts, communities and data types. Users can like the posts they enjoy, or comment on them. Users can also report posts or comments they find abusive or inappropriate.

There are also the project admins. They have the power to suspend any user, post, community, comment etc. from the system if any of those violates the regulations of the app itself.

Overall this app provides a great range of functionalities to ensure that every user can easily find the content they are looking for, unite among communities, meet and communicate with each other, and express themselves the way they want.

List and Status of Deliverables

Deliverable	Status	Update Frequency
Communication Plan	Completed	As improvement needed
GitHub Wiki	In Progress	Daily
GitHub Issues	In Progress	As improvement needed
Meeting Notes	In Progress	Weekly
Project Plan & RAM	In Progress	As improvement needed
Practice-app API	Completed	As improvement needed
API Documentation	Completed	As improvement needed

Evaluation of the Status of Deliverables

Communication Plan:

A well-designed communication plan is important to ensure healthy communication between members, the controlled progress of the project, and to ensure that everyone receives the necessary information regularly. Video calls improve communication and production. On the other hand, messaging provides faster and instant conversation. Hence, we decided to use different communication channels for different purposes such as Google Meet for weekly meetings and distribution of tasks, Slack for discussing details of tasks, WhatsApp for daily talk. In addition to our weekly meetings, we organized separate meetings for subtasks

You can view our communication plan.

GitHub Wiki

We have tried to keep our wiki pages up to date to show all the work we have done. While preparing our wiki home page, we considered people who might join our team or are interested in our project. To make it organized, we gathered our work under certain headings and prepared templates.While preparing our wiki home page, we also thought of friends who might just join our team or who might want to review our project.

You can visit our wiki home page.

GitHub Issue and Pull Requests

We created an issue for each task and assigned at least one team member. Every issue has a milestone that defines its week and labels which define component, type, priority, and status. We commented on issues and updated their status labels to inform the team about the process. We preferred communication channels instead of discussing under the issues, as some topics should remain team specific. But we commented on the final reviews and the results of the discussions in the issues.
We waited for at least one person to approve pull requests and assigned at least two people as reviewers. We made tracking easier by linking our pull requests with issues.

You can access our issues and pull requests.

Meeting Notes

We created a template for meeting notes. After every meeting, we documented the time, duration, attendees, agenda, discussion and action items.

You can access template and meeting notes from our repository.

Project Plan & RAM

Project Plan contains all deliverables with their data, duration and contributor. We also tried to plan our future work too.
With the Responsibility Assignment Matrix, we showed the participation of members in tasks with their roles. As you can see each team member took a role in most of the tasks as a contributor or a reviewer. Since planning is a key part of project management, we do our best to stick to our plans.

You can access our project plan and RAM.

Practice-app

We have developed a practice-app web application to get ready for the application we will write next year. Practice-app had two parts, backend and frontend. We used Spring Boot to write the backend and every member contributed to the RESTful API. Each member wrote unit tests for the API he wrote and provided Swagger documentation. Then we implemented a simple user interface using thymeleaf. We have deployed our project in AWS successfully.

API Documentation

The API documentation provides detailed information on how to use our API. In this way, it speeds up the integration process for people who want to use our API. During the development process, this documentation was very useful to the front-end team. In addition, since we use Swagger to create API documentation, it was possible for us to read the documentation and quickly test API from the same platform. When we want to review the practice app again in the next semester, we will benefit from this documentation.

Evaluation of Tools and Processes

Tools

GitHub

We used GitHub as our main platform. We uploaded all project related documents, opened issues to assign tasks, created branches and used its version management system, created pull requests to review code before merging etc. As we move forward with our project, our repository and its wiki page becomes more and more detailed. GitHub is a very useful platform to develop our project and keep track of what we have done. In the next step, we are planning to create automated workflows using GitHub actions.

Spring Boot Framework

We used Spring Boot to implement our backend. Although it seems confusing at first sight with lots of classes and connections, after getting used to it, it utilized our development process perfectly. Having encapsulated controller and service classes made code updates easier since updates in different classes did not trigger each other, each construct worked independently without touching each other. We also had the chance to customize the structures we implemented with beneficial annotations. To put in a nutshell, frameworks like Django and Flask may look simpler, however we are glad to learn Spring Boot and we can assert that it will help us with bigger projects.

Thymeleaf

We used thymeleaf to connect the backend and frontend. It was easy to configure and it enables us to use for loop or if else statements in the HTML. We think it reduces the complexity of the front end. It is a good choice for simple front-end development.

MongoDB

MongoDB is a document oriented NoSQL database program. We thought choosing a DB that uses JSON-like objects is a better way to handle our data, instead of using a relational database. Using a relational database would have been a better option if we had tightly coupled data.

IntelliJ IDEA

IntelliJ is an integrated development environment for Java. This IDE is a great choice for writing Java programs as it supports various frameworks and has a smart code completion. IntelliJ also includes version control which becomes very handy when working on a large scale project.

Projectlibre

Projectlibre is an open-source project management software. We used this tool to create and assign tasks, keep track of the project and distribute resources efficiently. This tool especially gained importance as we are not able to meet in person and tend to lose track in our project given the current circumstances. But we think that this platform is not
user friendly enough. Although we came to the same conclusion in the previous milestone too, we plan to continue using this tool as it is free.

Microsoft Excel

We use Microsoft Excel to create our responsibility assignment matrix. It enables us to create structured tables which can be easily modified very fast.

Swagger

Swagger is a tool that combines both API testing and API documentation. Swagger eliminates the need for changing tools for checking API parameters and return values while testing the API. This speeds up the testing process and decreases the chance to make errors. One of the best features is that it automatically creates the API documentation after it is configured.

Slack and Zoom

A good method of communication is a must have in a group effort. However, due to the pandemic, we are unable to conduct face-to-face meetings. Therefore, for our communication needs, we decided to use Slack and Zoom. Slack is a primarily text based chatting service and is used for quick and asynchronous chatting between two people or a group. It also enables us to create different channels for different tasks. On the other hand, we used Google Meets to conduct video meetings, where we decided on more general topics and talked about general issues that concerned every member of the team.

Processes

API Development
Implementation
Each member should create their own branch and work there. They should comment their code in an understandable way and provide Swagger documentation to their APIs. When their implementation is complete, they should open a pull request to merge their branch into the “dev” branch.
We can improve this process by opening the branch for each different task instead of opening the branch per member.

Pull Requests
After the implementation is complete, members should open a pull request to merge into dev. Pull requests must have a descriptive title, a summary, type, and component labels. At least one person must be assigned as a reviewer for each pull request. It must be approved by at least one person in order to be merged. Those working on the frontend should merge into the frontend branch before the dev.

Reviewing
Reviewers should first evaluate the functionality of the code and then judge the readability of the code. Edge cases should be considered. Reviewers should provide feedback for the pull request. At least one reviewer should approve the request before it can be merged.

Table of Work Done by Each Member

Member Name	Contributions
Kıymet Akdemir	● Implemented endpoints ○ GetPostsController.getPosts ○ GetPostsController.getDefinition ● Implemented services ○ CommunityService.getPosts ○ CommunityService.getDefinition ● Implemented unit test ○ GetPostsControllerTest ● Third party API Utility ○ DictionaryApi ● Milestone Report ○ Wrote overall status ○ Updated RAM with Zeynep ○ Updated Project Plan with Zeynep
İrem Zeynep Alagöz	● Implemented API functionality: Creating Community. Implemented POST method to create community. I used third-party DetectLanguageApi.detectLanguage(String text) to detect language of the community ○ **Result**: CreateCommunityController, CommunityService, DetectLanguageApi ○ Frontend URI: http://34.238.122.101:8080/home/communities/create ○ API URI: http://34.238.122.101:8080/community/create ● Implemented API functionality: Get Communities By Publicity. Implemented GET method to get communities based on their publicities. ○ Frontend URI:

	http://34.238.122.101:8080/home/communities/search to search http://34.238.122.101:8080/home/communities/?public=true (public) http://34.238.122.101:8080/home/communities/?public=false (private) ○ API URI: http://34.238.122.101:8080/communities/?public=true (public communities) http://34.238.122.101:8080/communities/?public=false (private communities) ○ **Result**: GetCommunitiesByPublicityController, CommunityService, CommunityRepository ● I implemented following methods of the CommunityService: ○ CommunityService.createCommunity ○ CommunityService.exists ○ CommunityService.findByPublicty ○ CommunityService.detectLanguage ○ CommunityService.getAllLanguages ○ CommunityService.getByName ○ **Result**: CommunityService ● Implemented Unit Tests: Community Service ○ **Result**: CommunityServiceTest ● Implemented Unit Tests: CreateCommunityController ○ **Result**: CreateCommunityControllerTest ● Implemented Unit Tests: GetCommunitiesByPublicityController ○ **Result**: GetCommunitiesByPublicityControllerTest ● Implemented Community entity and relations ○ **Result**: CommunityEntity ● Provided Swagger documentation for community entity ○ **Result**: http://34.238.122.101:8080/swagger-ui/#/ In the model parts. ● Implemented frontend: Creating Community ○ **Result**: CreateCommunityViewController, community_create_form, community ● Implemented frontend: Getting Community ○ **Result**: GettingCommunitiesByPublicityViewController, community_create_form, communities_search_form, communities, community ● Implemented frontend: See posts with Orhun ○ **Result**: GetPostsViewController, posts ● Designed a generic template for the website. Besides the already mentioned templates refactored community_search_form, dictionary_form. Hence all written/updated templates: ○ community.html ○ communities.html ○ community_create_form.html ○ community_search_form.html ○ communities_search_form.html ○ dictionary_form.html ○ posts.html ● Storing third-party API keys: created JSON file to store keys with Orhun and Halil

	● Updated Issue Labels: added new issue labels with the help of Halil, completed missing descriptions. You can view the labels. ● Milestone 2 Report: I wrote Status and Evaluation of Deliverables. ● Milestone 2 Report: I wrote Evaluation of Tools and Processes. ● Project Plan: practice-app: Kıymet Akdemir and I added the practice-app plan to our project plan. You can view the plan. ● Responsibility Assignment Matrix (RAM): I and Kıymet created RAM for milestone 2 and merged it with the previous one from milestone 1. ● Deployment: Even though the main work belongs to Halil and Sevde I attended deployment of our practice-app during the meeting with all team members.
Ufuk Arslan	● Updating the Slack channels and organizing them so that it is easier to search for the current and previous tasks. ● Deciding on which tools to use for the practice app and searching for backend, frontend, database options. ● Creating API functions for the practice app. Here is the the endpoints I implemented: UpdatePostController.updatePost(PostEntity postEntity): **PUT** method to update a post. Frontend URI: http://34.238.122.101:8080/home/post/update API URI: http://34.238.122.101:8080/post/update ● Implementing and documenting the **udatePost** function in **PostService**. ● Discussing about which entities and endpoints to use for the practice app ● Searching for third-party APIs to be used in the practice app. I chose Kanye Rest Api, created controllers and utilities for the endpoint. Here is the endpoint I implemented: UpdatePostController.getRandomKanyeQuote(): **GET** method that returns a random Kanye West quote. Frontend URI: http://34.238.122.101:8080/home/kanyequote API URI: http://34.238.122.101:8080/post/kanyequote ● Implementing unit tests for update post functionality. Those unit tests check whether the endpoints return the expected or not by mocking the update post scenarios. Here are the functions and files I created for that: UpdatePostControllerTest.java PostServiceTest.updatePost ● For the second milestone I wrote the **Moving Forward** and **Basic Functionality of the Project** sections of the Executive Summary. ● Reviewing the Docker deployment process with the group.
Halil Baydar	
Burak Onur Duman	● Implemented the API Endpoints ○ - JoiningCommunityController ○ - JoiningCommunityService ● Implemented the Unit Tests: ○ - JoiningCommunityControllerTest ○ - JoiningCommunityServiceTest ● Implemented the Usage of Third Party API: ○ - AdviceController ● Implemented the Frontend View Controllers: ○ - AdviceViewController ○ - CreateUserViewController

	○ - ProfileViewController ● Implemented the Frontend HTML Templates: ○ - advice.html ○ - getjoinedcommunities.html ○ - getprofile.html ○ - getuserbyname.html ○ - saveuser.html ○ - searchadvice.html ○ - setrandompic.html ○ - updateprofile.html
Orhun Görkem	1. I implemented API Functionalities after deciding the format and frameworks with the group. We used Spring Boot for the backend. Here are the endpoints I worked on: ● CreatePostController.savePost(PostEntity postEntity) Frontend URI: http://34.238.122.101:8080/home/post/create API URI: http://34.238.122.101:8080/post/save POST method to create posts. ● GetPlacesController.getPlaces(String input) Frontend URI: http://34.238.122.101:8080/home/places/search API URI: http://34.238.122.101:8080/places/{input} GET method to get relevant locations with given text input 2. We decided on the entities necessary for the practice-app. I implemented the post entity. 3. I implemented services to handle API requests from controllers. Service functions are the following: ● PostService.getPost ● PostService.getPlaces 4. I researched Third Party API's to make use of in our project. I decided on Google Places API. I implemented the PlacesAPI class to fetch the data and shape it into a useful format. 5. I implemented unit tests for controller and service classes assigned to me. Unit tests check whether the request is fetched with a given URI and the necessary backend operation is completed. Test classes are: ● CreatePostControllerTest ● PostServiceTest.savePost 6. View Controllers are necessary to connect the backend to UI. I implemented the view controllers: ● DictionaryViewController ● GetCommunityViewController ● GetPlacesViewController ● GetPostsViewController ● IndexViewController ● PostViewController ● UniversitiesViewController 7. UI templates are implemented to give frontend its shape: ● community.html ● community_search_form.html ● dictionary.html ● dictionary_form.html ● getprofile.html ● index.html ● places.html ● posts_search_form.html ● universities.html

	8. Reviewed deployment with the group. 9. Got the output of API documentation from Swagger to provide in the report.
Bekir Keldal	Implemented the API endpoint: ● GetCommunityController Implemented services: ● CommunityService.getByName ● CommunityService.getUniversitiesList Implemented the unit tests: ● UniverisitiesListControllerTest ● GetCommunityControllerTest Implemented third party API: ● UniversitiesListApi Attended the meeting for deployment with all members. Milestone 2 Group Report: ● Wrote the introduction ● Wrote the challenges we met
Kutay Saran	
Sevde Sarıkaya	**URI & API Functions** **Profile Controller:** ● **getProfile():** **Frontend:** http://34.238.122.101:8080/home/getprofile **Backend:** http://34.238.122.101:8080/profile/{username} GET REQUEST ● **updateProfile():** **Frontend**: http://34.238.122.101:8080/home/updateprofileform **Backend:** http://34.238.122.101:8080/profile/updateProfile POST REQUEST ● **setRandomPic():** **Frontend:** http://34.238.122.101:8080/home/setrandompic **Backend:** http://34.238.122.101:8080/profile/setRandomPic/{username} POST REQUEST **Major Work** ● **Endpoints implemented:** ProfileController ● **Entity implemented:** Profile Profile Repository ● **Services implemented:** ProfileService.getProfile ProfileService.updateProfile ProfileService.setRandomPic ● **Third Party Utilities implemented:** PicturesApi ● **Unit tests implemented:** ProfileServiceTest ● **Swagger:** ProfileController & ProfileEntity Report ● **Deployment:** Dockerized a project as a sample: https://hub.docker.com/repository/docker/practiceapp2/cmpe352 Ran an EC2 instance Deployed the Docker image in the AWS machine ● **Frontend:** Created a sample Angular project

	Implemented Save User and Get User components
Abdurrahman Emre Yılmaz	

Deliverables

Responsibility Assignment Matrix (RAM)

You can access the pdf file from here.

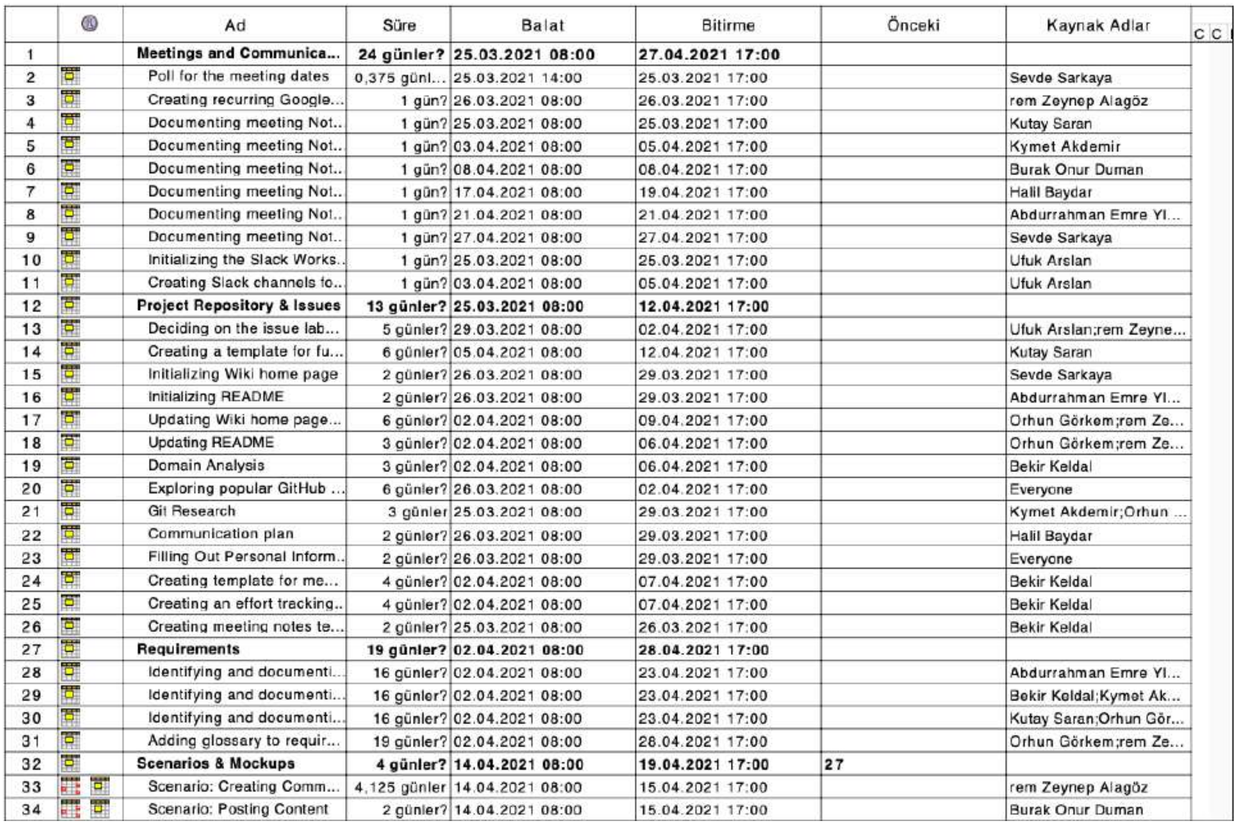
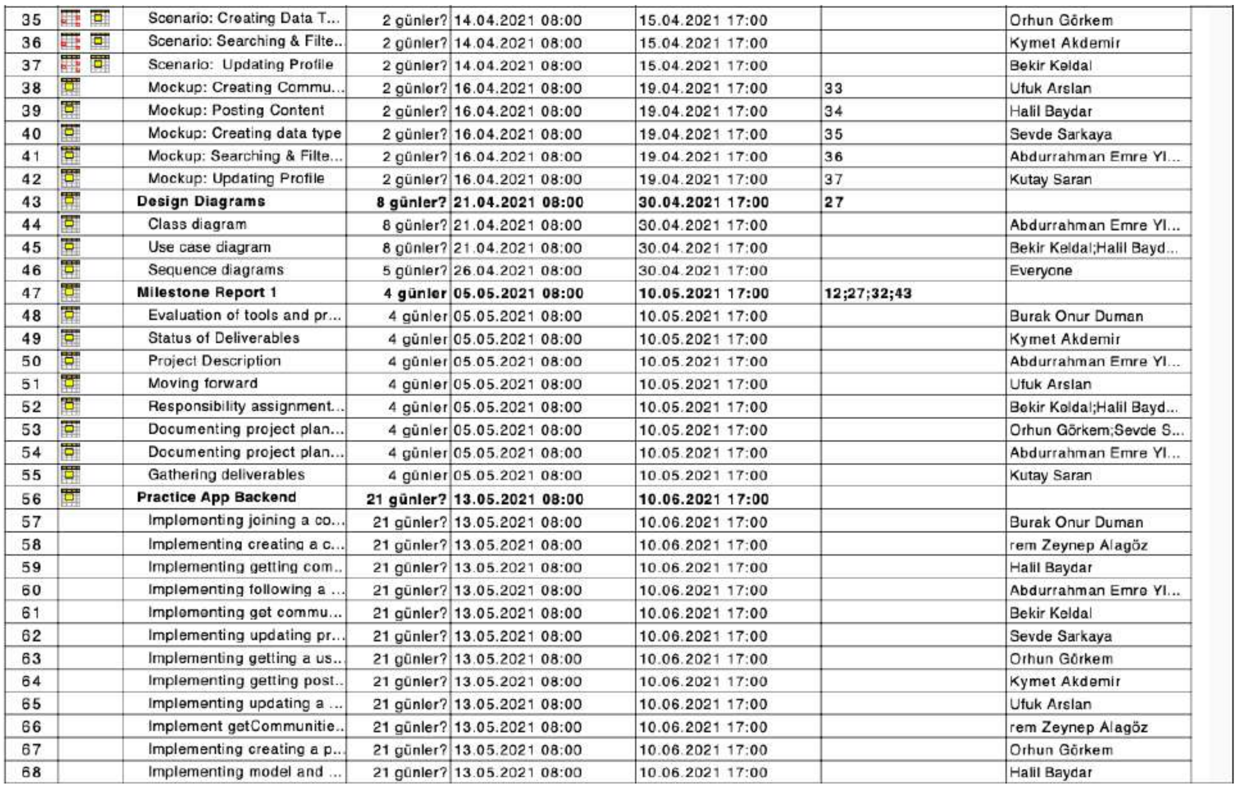
L: Lead C: Contributor R: Reviewer N: None	Kıymet Akdemir	İrem Zeynep Alagöz	Ufuk Arslan	Halil Baydar	Burak Onur Duman	Orhun Görkem	Bekir Keldal	Kutay Saran	Sevde Sarıkaya	Abdurrahman Emre Yılmaz
Customer Meeting										
Identifying questions to ask customer	L	C	C	C	C	C	C	C	C	C/R
Organize questions page	C	N	N	N	N	N	N	R	R	L
Organizing customer meeting	N	R	N	N	N	L	N	N	N	N
Taking note of the customer meeting #1	N	L	N	R	N	N	N	N	N	N
Documenting answers of questions for meeting #1	N	R	N	N	N	N	N	N	N	L
Taking notes of customer meeting #2	N	L	N	N	N	N	N	N	N	N
Updating Customer Meeting Documentation	C	L	N	N	N	N	R	N	N	N
Meetings & Communication										
Meeting Date Poll	N	N	N	R	N	N	N	N	L	N
Creating recurring Google Meet session	N	L	N	N	N	R	N	N	N	N
Documenting meeting Notes #1	N	N	N	N	N	N	R	L	N	N
Documenting meeting Note #2	N	N	N	N	L	N	R	N	N	N
Documenting meeting Notes #3	L	N	N	N	N	N	N	N	N	R
Documenting meeting Notes #4	N	N	N	L	N	R	N	N	N	N
Documenting meeting Notes #5	R	N	N	N	N	N	N	N	N	L
Documenting meeting Notes #6	N	N	N	N	N	N	N	R	L	N
Documenting meeting Notes #7	R	N	N	N	N	N	L	N	N	N
Documenting meeting Notes #8	N	R	N	N	N	L	N	N	N	N
Documenting meeting Notes #9		L	N	R	N		N	N	N	N
Initializing the Slack Workspace	N	R	L	N	N	N	N	N	N	N
Creating Slack channels for decided tasks	N	R	L	N	N	N	N	N	N	N
Requirements										
Identifying and documenting user requirements	R	C	N	C	C	R	N	N	R	C
Identifying and documenting system requirements	C	R	C	N	R	R	C	N	C	R
Identifying and documenting non-functional requirements	R	R	N	N	R	C	N	C	R	R
Adding glossary to requirements page	N	C	R	N	N	C	N	R	N	N
Scenarios & Mockups										
Scenario: Creating Community	N	L	N	N	R	N	N	N	N	N
Scenario: Posting Content	N	N	R	N	L	N	N	N	N	N
Scenario: Creating Data Type	R	N	N	N	N	L	N	N	N	N
Scenario: Searching & Filtering	L	N	N	N	N	N	R	N	N	N
Scenario: Updating Profile	N	N	N	N	R	N	L	N	N	N
Mockup: Creating Community	N	R	L	N	N	N	N	N	N	N
Mockup: Posting Content	N	N	N	L	R	N	N	N	N	N
Mockup: Creating data type	N	N	N	N	N	R	N	N	L	N
Mockup: Searching & Filtering	R	N	R	N	N	N	N	N	N	L
Mockup: Updating Profile	N	N	R	N	N	N	R	L	N	N
Design Diagrams										
Class diagram	C	C	C	N	C	R	N	N	N	C
Use case diagram	N	R	N	C	N	C	C	C	C	N
Sequence diagram: Login	N	L	N	N	R	N	N	N	N	N
Sequence diagram: Registration	N	L	N	N	R	N	N	N	N	N
Sequence diagram: Notification settings	N	R	N	N	N	N	N	N	L	N
Sequence diagram: Liking post	N	N	N	N	N	R	L	N	N	N
Sequence diagram: Private messaging	R	N	N	L	N	N	N	N	N	N
Sequence diagram: Creating datatype	N	R	N	N	N	L	N	N	N	N
Sequence diagram: Searching community	L	N	N	N	N	R	N	N	N	N
Sequence diagram: Reporting comment	N	N	R	N	L	N	N	N	N	N
Sequence diagram: Posting content	N	N	L	R	N	N	N	N	N	N
Sequence diagram: Editing profile	N	N	N	N	N	N	N	L	R	N
Project Repository & Issues										

Deciding on the issue labels and creating them	N	L	C	N	N	N	N	N	N	R
Creating a template for future issues	N	N	N	R	N	N	N	L	N	N
Initializing Wiki home page	N	N	N	N	N	N	R	N	L	C
Initializing README	N	N	N	N	N	N	R	N	N	L
Updating Wiki home page and sidebar	N	L	R	N	N	C	N	N	N	N
Updating README	N	C	N	R	N	L	N	N	N	N
Domain Analysis	N	N	C	C	N	R	N	N	C	N
Exploring popular GitHub repositories	C	C/R	C	C	C	C	C	C	C	C
Git Research	C	N	N	N	N	C	N	N	R	N
Communication plan	R	N	R	L	N	N	N	N	N	N
Filling Out Personal Information Wiki Page	C	C	C	C	C/R	C	C	C	C	C
Creating template for member wiki pages	N	N	N	N	R	N	L	N	N	N
Creating an effort tracking table template	N	N	N	N	N	N	L	R	N	N
Creating meeting notes template	N	N	R	N	N	N	L	N	N	N
Updating Issue Labels	R	L	R	C	R	R	R	R	R	R
practice-app										
RESTFULL API										
Implementing joining a community API	N	N	N	N	L	N	R	N	R	N
Implementing creating a community API	N	L	N	N	N	R	N	N	R	N
Implementing creating a user API	N	N	N	L	N	N	N	N	N	N
Implementing following a user API	N	R	N	N	N	R	N	N	N	L
Implementing get communities by name API	R	N	R	N	N	N	L	N	N	N
Implementing updating profile API	N	N	R	N	R	N	N	N	L	N
Implementing getting a user API	N	N	N	L	N	L	N	N	N	N
Implementing getting posts by a community API	L	R	N	N	N	R	N	N	R	N
Implementing updating a post API	R	N	L	N	N	N	N	N	N	N
Implement getCommunitiesByPublicity API	R	L	N	N	N	N	N	N	R	N
Implementing creating a post API	N	R	N	N	N	L	N	N	N	N
DB Entites										
Implementing model and repository for User	R	R	R	L	R	R	R	R	R	R
Implementing model and repository for Community	R	L	R	N	N	R	N	N	N	N
Implement profile model	N	R	N	N	N	R	N	N	L	N
Implementing model and repository for Post	N	R	N	N	N	L	N	N	N	N
Configuration										
Implement Swagger Configuration	N	N	N	L	N	N	N	N	N	N
Storing third-party API Keys	N	L	N	C	N	L	N	N	N	N
Creating Docker file and image	N	N	N	L	N	N	N	N	C	N
AWS Deployment	C	C	C	L	C	C	C	R	L	C

Unit Tests										
Implement Test: CreateCommunityController	N	L	N	N	N	R	R	N	N	N
Implement Test: PostService	N	R	N	N	N	L	R	N	N	N
Implement Test: CommunityService	N	L	N	N	N	R	R	N	N	N
Implement Test: CreatePostController	N	R	N	N	N	L	R	N	N	N
Implement Test: UpdatePostController	N	R	L	N	N	R	N	N	R	N
Implement Test: GetPostByCommnunityController	L	N	N	N	N	N	N	N	R	N
Implement Test: GetCommnunityController	N	R	N	N	N	R	L	N	N	N
Implement Test: JoinCommnunityController	N	N	N	N	L	R	N	N	N	N
Implement Test: GetCommunitiesByPublicityController	N	L	N	N	N	N	N	N	R	N
Implement Test: UserService	N	R	N	L	N	N	N	N	N	N
Implement Test: UserController	N	N	N	L	N	N	N	N	N	N
Implement Test: GetProfile	N	R	R	N	N	R	N	N	L	N
Implement Test: UpdateProfile	N	R	R	N	N	R	N	N	L	N
Frontend										
Implementation: Frontend for Profile & Create user	C	R	N	N	L	R	N	N	N	N
Implementation: Frontend for Creating & Getting Community	N	L	N	N	N	L	N	N	N	N
Implementation: Frontend for Creating & Updating Posts	N	C	L	N	N	C	N	N	N	L
Implementation: Frontend for Third Party APIs	N	R	N	L	C	C	N	N	N	N
Implementation: Frontend for Getting Posts & Joining Community	N	R	N	N	N	C	N	C	L	N
Milestone 1 Report										
Evaluation of tools and processes	R	N	N	N	L	N	N	N	N	N
Status of Deliverables	L	N	N	N	N	R	N	N	N	N
Project Description	N	R	N	N	N	N	N	N	N	L
Moving forward	N	N	L	N	R	N	N	N	N	N
Responsibility assignment matrix (RAM)	N	L	N	L	N	N	L	N	R	N
Documenting project plan for past work	N	L	N	R	R	L	R	R	L	R
Documenting project plan for future work	R	R	R	L	N	N	N	N	N	L
Gathering deliverables	N	N	N	N	N	N	N	L	R	N
Milestone 2 Report										
Evaluation of tools and processes	N	L	N	N	N	N	R	N	N	N
Status of Deliverables	N	L	N	N	N	N	R	N	N	N
Overall Status	L	R	N	N	N	N	N	N	N	N
Challenges	N	R	N	N	N	N	L	N	N	N
Introduction	N	N	R	N	N	N	L	N	N	N
Look ahead	N	N	L	N	N	N	N	N	N	N
Responsibility assignment matrix (RAM)	C	L	R	R	R	R	R	R	R	R
Documenting project plan for the practice-app	L	C	R	N	N	N	N	N	N	N
Basic functionality of the practice-app	N	N	L	N	N	N	N	N	R	N

Updated Project Plan

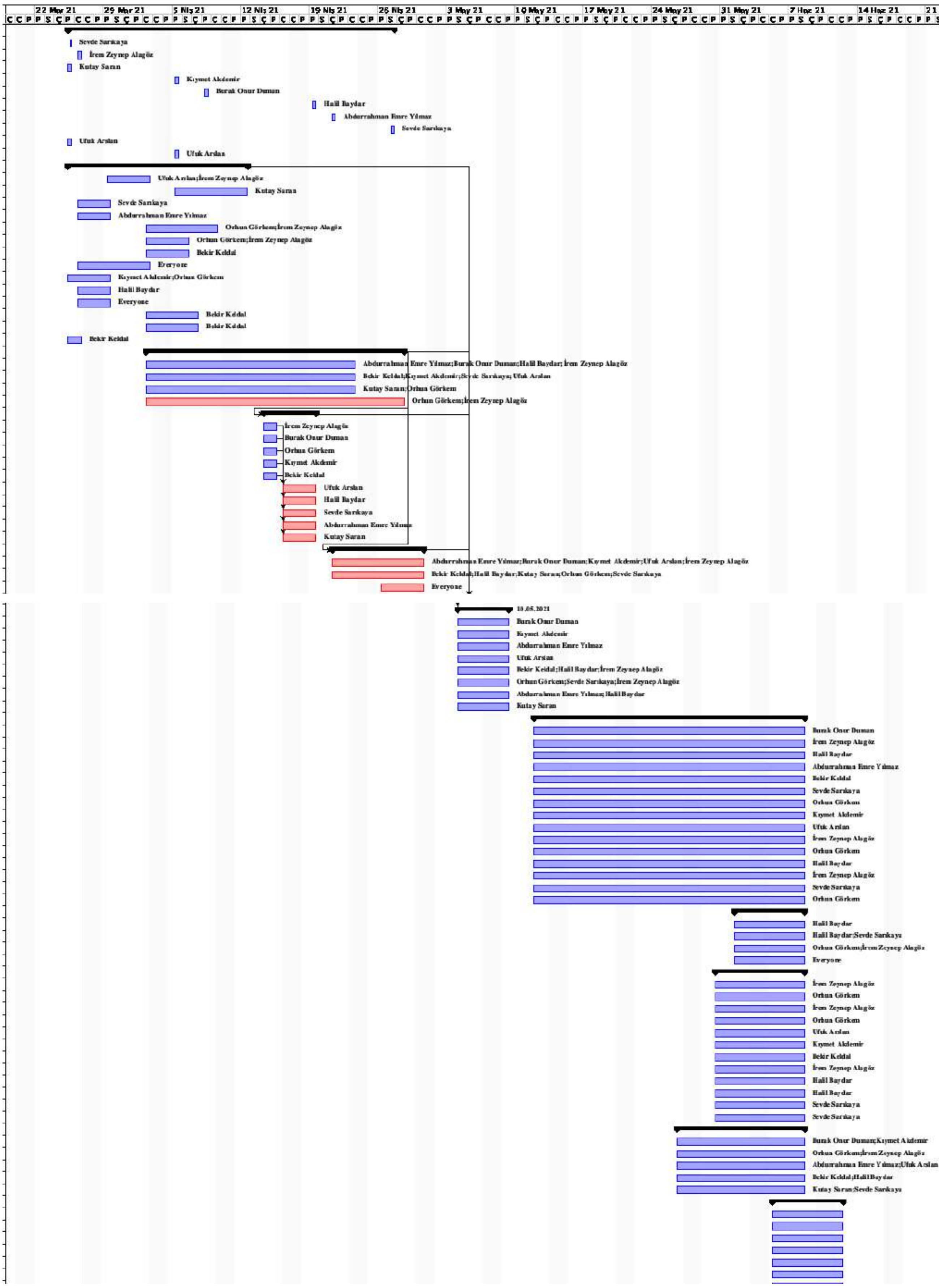
You can access the pdf file from here.

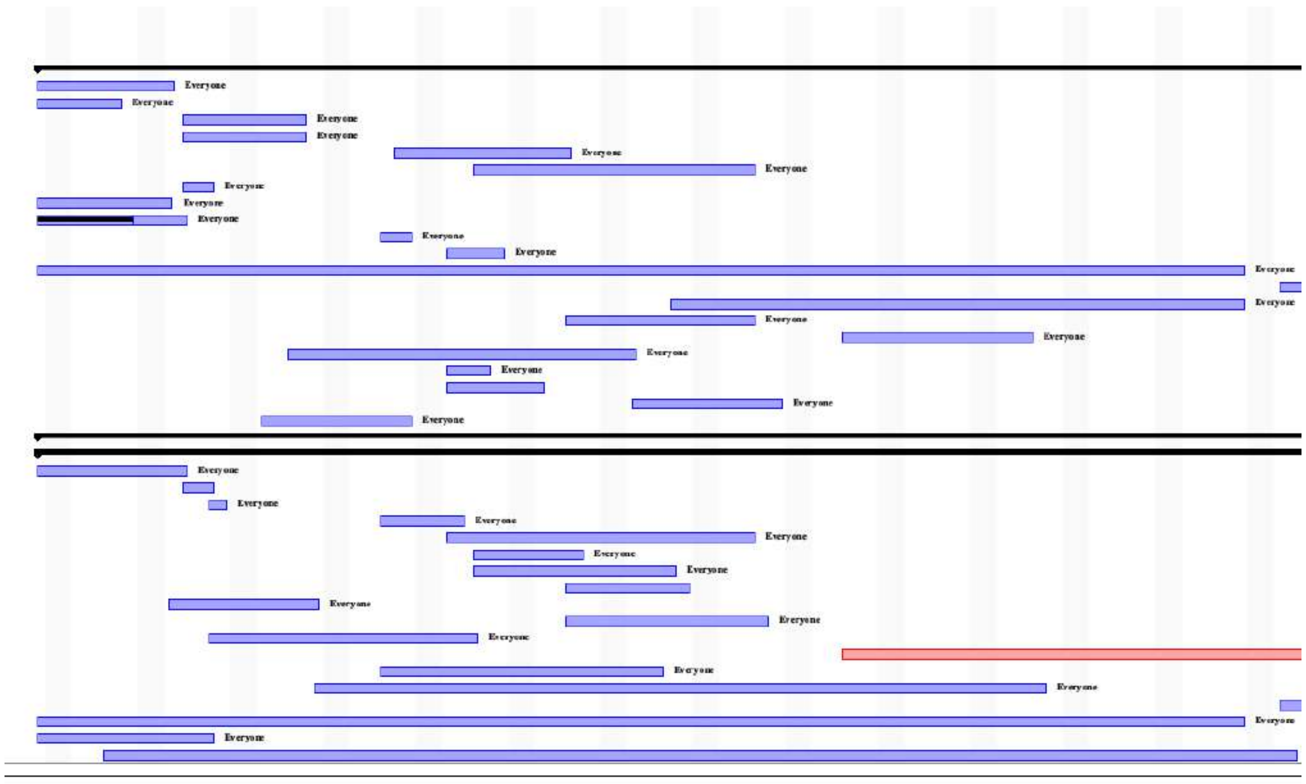



		Ad	Süre	Balat	Bitirme	Önceki	Kaynak Adlar	C C
1		**Meetings and Communica...**	**24 günler?**	**25.03.2021 08:00**	**27.04.2021 17:00**			
2		Poll for the meeting dates	0,375 günl...	25.03.2021 14:00	25.03.2021 17:00		Sevde Sarkaya	
3		Creating recurring Google...	1 gün?	26.03.2021 08:00	26.03.2021 17:00		rem Zeynep Alagöz	
4		Documenting meeting Not..	1 gün?	25.03.2021 08:00	25.03.2021 17:00		Kutay Saran	
5		Documenting meeting Not..	1 gün?	03.04.2021 08:00	05.04.2021 17:00		Kymet Akdemir	
6		Documenting meeting Not..	1 gün?	08.04.2021 08:00	08.04.2021 17:00		Burak Onur Duman	
7		Documenting meeting Not..	1 gün?	17.04.2021 08:00	19.04.2021 17:00		Halil Baydar	
8		Documenting meeting Not..	1 gün?	21.04.2021 08:00	21.04.2021 17:00		Abdurrahman Emre Yl...	
9		Documenting meeting Not..	1 gün?	27.04.2021 08:00	27.04.2021 17:00		Sevde Sarkaya	
10		Initializing the Slack Works..	1 gün?	25.03.2021 08:00	25.03.2021 17:00		Ufuk Arslan	
11		Creating Slack channels fo..	1 gün?	03.04.2021 08:00	05.04.2021 17:00		Ufuk Arslan	
12		**Project Repository & Issues**	**13 günler?**	**25.03.2021 08:00**	**12.04.2021 17:00**			
13		Deciding on the issue lab...	5 günler?	29.03.2021 08:00	02.04.2021 17:00		Ufuk Arslan;rem Zeyne...	
14		Creating a template for fu...	6 günler?	05.04.2021 08:00	12.04.2021 17:00		Kutay Saran	
15		Initializing Wiki home page	2 günler?	26.03.2021 08:00	29.03.2021 17:00		Sevde Sarkaya	
16		Initializing README	2 günler?	26.03.2021 08:00	29.03.2021 17:00		Abdurrahman Emre Yl...	
17		Updating Wiki home page...	6 günler?	02.04.2021 08:00	09.04.2021 17:00		Orhun Görkem;rem Ze...	
18		Updating README	3 günler?	02.04.2021 08:00	06.04.2021 17:00		Orhun Görkem;rem Ze...	
19		Domain Analysis	3 günler?	02.04.2021 08:00	06.04.2021 17:00		Bekir Keldal	
20		Exploring popular GitHub ...	6 günler?	26.03.2021 08:00	02.04.2021 17:00		Everyone	
21		Git Research	3 günler	25.03.2021 08:00	29.03.2021 17:00		Kymet Akdemir;Orhun ...	
22		Communication plan	2 günler?	26.03.2021 08:00	29.03.2021 17:00		Halil Baydar	
23		Filling Out Personal Inform..	2 günler?	26.03.2021 08:00	29.03.2021 17:00		Everyone	
24		Creating template for me...	4 günler?	02.04.2021 08:00	07.04.2021 17:00		Bekir Keldal	
25		Creating an effort tracking..	4 günler?	02.04.2021 08:00	07.04.2021 17:00		Bekir Keldal	
26		Creating meeting notes te...	2 günler?	25.03.2021 08:00	26.03.2021 17:00		Bekir Keldal	
27		**Requirements**	**19 günler?**	**02.04.2021 08:00**	**28.04.2021 17:00**			
28		Identifying and documenti...	16 günler?	02.04.2021 08:00	23.04.2021 17:00		Abdurrahman Emre Yl...	
29		Identifying and documenti...	16 günler?	02.04.2021 08:00	23.04.2021 17:00		Bekir Keldal;Kymet Ak...	
30		Identifying and documenti...	16 günler?	02.04.2021 08:00	23.04.2021 17:00		Kutay Saran;Orhun Gör...	
31		Adding glossary to requir...	19 günler?	02.04.2021 08:00	28.04.2021 17:00		Orhun Görkem;rem Ze...	
32		**Scenarios & Mockups**	**4 günler?**	**14.04.2021 08:00**	**19.04.2021 17:00**	**27**		
33		Scenario: Creating Comm...	4,125 günler	14.04.2021 08:00	15.04.2021 17:00		rem Zeynep Alagöz	
34		Scenario: Posting Content	2 günler?	14.04.2021 08:00	15.04.2021 17:00		Burak Onur Duman	
35		Scenario: Creating Data T...	2 günler?	14.04.2021 08:00	15.04.2021 17:00		Orhun Görkem	
36		Scenario: Searching & Filte..	2 günler?	14.04.2021 08:00	15.04.2021 17:00		Kymet Akdemir	
37		Scenario: Updating Profile	2 günler?	14.04.2021 08:00	15.04.2021 17:00		Bekir Keldal	
38		Mockup: Creating Commu...	2 günler?	16.04.2021 08:00	19.04.2021 17:00	33	Ufuk Arslan	
39		Mockup: Posting Content	2 günler?	16.04.2021 08:00	19.04.2021 17:00	34	Halil Baydar	
40		Mockup: Creating data type	2 günler?	16.04.2021 08:00	19.04.2021 17:00	35	Sevde Sarkaya	
41		Mockup: Searching & Filte...	2 günler?	16.04.2021 08:00	19.04.2021 17:00	36	Abdurrahman Emre Yl...	
42		Mockup: Updating Profile	2 günler?	16.04.2021 08:00	19.04.2021 17:00	37	Kutay Saran	
43		**Design Diagrams**	**8 günler?**	**21.04.2021 08:00**	**30.04.2021 17:00**	**27**		
44		Class diagram	8 günler?	21.04.2021 08:00	30.04.2021 17:00		Abdurrahman Emre Yl...	
45		Use case diagram	8 günler?	21.04.2021 08:00	30.04.2021 17:00		Bekir Keldal;Halil Bayd...	
46		Sequence diagrams	5 günler?	26.04.2021 08:00	30.04.2021 17:00		Everyone	
47		**Milestone Report 1**	**4 günler**	**05.05.2021 08:00**	**10.05.2021 17:00**	**12;27;32;43**		
48		Evaluation of tools and pr...	4 günler	05.05.2021 08:00	10.05.2021 17:00		Burak Onur Duman	
49		Status of Deliverables	4 günler	05.05.2021 08:00	10.05.2021 17:00		Kymet Akdemir	
50		Project Description	4 günler	05.05.2021 08:00	10.05.2021 17:00		Abdurrahman Emre Yl...	
51		Moving forward	4 günler	05.05.2021 08:00	10.05.2021 17:00		Ufuk Arslan	
52		Responsibility assignment...	4 günler	05.05.2021 08:00	10.05.2021 17:00		Bekir Keldal;Halil Bayd...	
53		Documenting project plan...	4 günler	05.05.2021 08:00	10.05.2021 17:00		Orhun Görkem;Sevde S...	
54		Documenting project plan...	4 günler	05.05.2021 08:00	10.05.2021 17:00		Abdurrahman Emre Yl...	
55		Gathering deliverables	4 günler	05.05.2021 08:00	10.05.2021 17:00		Kutay Saran	
56		**Practice App Backend**	**21 günler?**	**13.05.2021 08:00**	**10.06.2021 17:00**			
57		Implementing joining a co...	21 günler?	13.05.2021 08:00	10.06.2021 17:00		Burak Onur Duman	
58		Implementing creating a c...	21 günler?	13.05.2021 08:00	10.06.2021 17:00		rem Zeynep Alagöz	
59		Implementing getting com..	21 günler?	13.05.2021 08:00	10.06.2021 17:00		Halil Baydar	
60		Implementing following a ...	21 günler?	13.05.2021 08:00	10.06.2021 17:00		Abdurrahman Emre Yl...	
61		Implementing get commu...	21 günler?	13.05.2021 08:00	10.06.2021 17:00		Bekir Keldal	
62		Implementing updating pr...	21 günler?	13.05.2021 08:00	10.06.2021 17:00		Sevde Sarkaya	
63		Implementing getting a us..	21 günler?	13.05.2021 08:00	10.06.2021 17:00		Orhun Görkem	
64		Implementing getting post..	21 günler?	13.05.2021 08:00	10.06.2021 17:00		Kymet Akdemir	
65		Implementing updating a ...	21 günler?	13.05.2021 08:00	10.06.2021 17:00		Ufuk Arslan	
66		Implement getCommunitie..	21 günler?	13.05.2021 08:00	10.06.2021 17:00		rem Zeynep Alagöz	
67		Implementing creating a p...	21 günler?	13.05.2021 08:00	10.06.2021 17:00		Orhun Görkem	
68		Implementing model and ...	21 günler?	13.05.2021 08:00	10.06.2021 17:00		Halil Baydar	

69		Implementing model and ...	21 günler?	13.05.2021 08:00	10.06.2021 17:00		rem Zeynep Alagöz
70		Implement profile model	21 günler?	13.05.2021 08:00	10.06.2021 17:00		Sevde Sarkaya
71		Implementing model and ...	21 günler?	13.05.2021 08:00	10.06.2021 17:00		Orhun Görkem
72	📅	**Practice App Configuration**	**6 günler?**	**03.06.2021 08:00**	**10.06.2021 17:00**		
73		Implement Swagger Confi...	6 günler?	03.06.2021 08:00	10.06.2021 17:00		Halil Baydar
74		Creating Docker file and i...	6 günler?	03.06.2021 08:00	10.06.2021 17:00		Halil Baydar;Sevde Sar...
75		Storing third-party API keys	6 günler?	03.06.2021 08:00	10.06.2021 17:00		Orhun Görkem;rem Ze...
76		AWS Deployment	6 günler?	03.06.2021 08:00	10.06.2021 17:00		Everyone
77		**Practice App Unit Tests**	**8 günler**	**01.06.2021 08:00**	**10.06.2021 17:00**		
78	📅	Implement Test: CreateCo...	8 günler	01.06.2021 08:00	10.06.2021 17:00		rem Zeynep Alagöz
79	📅	Implement Test: PostServi...	8 günler	01.06.2021 08:00	10.06.2021 17:00		Orhun Görkem
80	📅	Implement Test: Commun...	8 günler	01.06.2021 08:00	10.06.2021 17:00		rem Zeynep Alagöz
81	📅	Implement Test: CreatePo...	8 günler	01.06.2021 08:00	10.06.2021 17:00		Orhun Görkem
82	📅	Implement Test: UpdateP...	8 günler	01.06.2021 08:00	10.06.2021 17:00		Ufuk Arslan
83	📅	Implement Test: GetPostB...	8 günler	01.06.2021 08:00	10.06.2021 17:00		Kymet Akdemir
84	📅	Implement Test: GetCom...	8 günler	01.06.2021 08:00	10.06.2021 17:00		Bekir Keldal
85	📅	Implement Test: GetCom...	8 günler	01.06.2021 08:00	10.06.2021 17:00		rem Zeynep Alagöz
86	📅	Implement Test: UserServi...	8 günler	01.06.2021 08:00	10.06.2021 17:00		Halil Baydar
87	📅	Implement Test: UserCont...	8 günler	01.06.2021 08:00	10.06.2021 17:00		Halil Baydar
88	📅	Implement Test: GetProfile	8 günler	01.06.2021 08:00	10.06.2021 17:00		Sevde Sarkaya
89	📅	Implement Test: UpdateP...	8 günler	01.06.2021 08:00	10.06.2021 17:00		Sevde Sarkaya
90		**Practice App Frontend**	**10 günler**	**28.05.2021 08:00**	**10.06.2021 17:00**		
91	📅	Implementation: Frontend...	10 günler	28.05.2021 08:00	10.06.2021 17:00		Burak Onur Duman;Ky...
92	📅	Implementation: Frontend...	10 günler	28.05.2021 08:00	10.06.2021 17:00		Orhun Görkem;rem Ze...
93	📅	Implementation: Frontend...	10 günler	28.05.2021 08:00	10.06.2021 17:00		Abdurrahman Emre Yl...
94	📅	Implementation: Frontend...	10 günler	28.05.2021 08:00	10.06.2021 17:00		Bekir Keldal;Halil Baydar
95	📅	Implementation: Frontend...	10 günler	28.05.2021 08:00	10.06.2021 17:00		Kutay Saran;Sevde Sar...
96	📅	**Milestone Report 2**	**6 günler?**	**07.06.2021 08:00**	**14.06.2021 17:00**		
97	📅	Creating the Table of Con...	6 günler?	07.06.2021 08:00	14.06.2021 17:00		
98	📅	Status of Deliverables	6 günler?	07.06.2021 08:00	14.06.2021 17:00		rem Zeynep Alagöz
99		Overall Status	6 günler?	07.06.2021 08:00	14.06.2021 17:00		Kymet Akdemir
100		Challenges	6 günler?	07.06.2021 08:00	14.06.2021 17:00		Bekir Keldal
101		Introduction	6 günler?	07.06.2021 08:00	14.06.2021 17:00		Bekir Keldal
102		Look ahead	6 günler?	07.06.2021 08:00	14.06.2021 17:00		Ufuk Arslan
103		Responsibility assignment...	3 günler?	07.06.2021 08:00	09.06.2021 17:00		Kymet Akdemir;rem Z...
104		Documenting project plan...	3 günler?	07.06.2021 08:00	09.06.2021 17:00		Kymet Akdemir;rem Z...
105		Basic functionality of the p...	6 günler?	07.06.2021 08:00	14.06.2021 17:00		Ufuk Arslan
106	📅	**BACKEND**	**91 günler?**	**01.10.2021 08:00**	**04.02.2022 17:00**		
107		Service Classes	7 günler?	01.10.2021 08:00	11.10.2021 17:00		Everyone
108		Controller classes	5 günler?	01.10.2021 08:00	07.10.2021 17:00		Everyone
109	📅	Abstract Service classes	8 günler?	12.10.2021 08:00	21.10.2021 17:00		Everyone
110	📅	Abstract Controller classes	8 günler?	12.10.2021 08:00	21.10.2021 17:00		Everyone
111	📅	Resource classes	10 günler?	28.10.2021 08:00	10.11.2021 17:00		Everyone
112	📅	Security configuration	16 günler?	03.11.2021 08:00	24.11.2021 17:00		Everyone
113	📅	Aws configuration	3 günler?	12.10.2021 08:00	14.10.2021 17:00		Everyone
114		Data transfer classes	6,5 günler?	01.10.2021 08:00	11.10.2021 13:00		Everyone
115		Entity Classes	8 günler?	01.10.2021 08:00	12.10.2021 17:00		Everyone
116	📅	Database configuration	3 günler?	27.10.2021 08:00	29.10.2021 17:00		Everyone
117	📅	Mapper interfaces	5 günler?	01.11.2021 08:00	05.11.2021 17:00		Everyone
118		Exception handling	66 günler?	01.10.2021 08:00	31.12.2021 17:00		Everyone
119	📅	Socket implementation	10 günler?	01.01.2022 08:00	14.01.2022 17:00		Everyone
120	📅	Notification implementation	32 günler?	18.11.2021 08:00	31.12.2021 17:00		Everyone
121	📅	Search implementation	11 günler?	10.11.2021 08:00	24.11.2021 17:00		Everyone
122	📅	Repository interfaces	11 günler?	01.12.2021 08:00	15.12.2021 17:00		Everyone
123	📅	Util classes	19 günler?	20.10.2021 08:00	15.11.2021 17:00		Everyone
124	📅	Swagger configuration	4 günler?	01.11.2021 08:00	04.11.2021 17:00		Everyone
125	📅	Email configuration	6 günler?	01.11.2021 08:00	08.11.2021 17:00		
126	📅	Authentication configuratior	10 günler?	15.11.2021 08:00	26.11.2021 17:00		Everyone
127	📅	Authorization configuration	10 günler?	18.10.2021 08:00	29.10.2021 17:00		Everyone
128	📅	**FrontEnd**	**91 günler?**	**01.10.2021 08:00**	**04.02.2022 17:00**		
129		**Application - Websites**	**91 günler?**	**01.10.2021 08:00**	**04.02.2022 17:00**		
130		Login page	8 günler?	01.10.2021 08:00	12.10.2021 17:00		Everyone
131	📅	Registration page	3 günler?	12.10.2021 08:00	14.10.2021 17:00		
132	📅	Confirmation page	2 günler?	14.10.2021 08:00	15.10.2021 17:00		Everyone
133	📅	Home page	5 günler?	27.10.2021 08:00	02.11.2021 17:00		Everyone
134	📅	Community page	18 günler?	01.11.2021 08:00	24.11.2021 17:00		Everyone
135	📅	Search page	7 günler?	03.11.2021 08:00	11.11.2021 17:00		Everyone
136	📅	Profile page	12 günler?	03.11.2021 08:00	18.11.2021 17:00		Everyone

137	▦	Settings pages	8 günler?	10.11.2021 08:00	19.11.2021 17:00			
138	▦	Chats page	10 günler?	09.10.2021 08:00	22.10.2021 17:00		Everyone	
139	▦	Message page	12 günler?	10.11.2021 08:00	25.11.2021 17:00		Everyone	
140	▦	Share post page	15 günler?	14.10.2021 08:00	03.11.2021 17:00		Everyone	
141	▦	Community admin panel	48 günler?	01.12.2021 08:00	04.02.2022 17:00		Everyone	
142	▦	Feedback page	16 günler?	27.10.2021 08:00	17.11.2021 17:00		Everyone	
143	▦	Admin panel	40 günler?	22.10.2021 08:00	16.12.2021 17:00		Everyone	
144	▦	Redux implementation	23 günler?	01.01.2022 08:00	02.02.2022 17:00		Everyone	
145		Axios implementation	66 günler?	01.10.2021 08:00	31.12.2021 17:00		Everyone	
146	▦	Permission configuration	10 günler?	01.10.2021 08:00	14.10.2021 17:00		Everyone	
147	▦	Delivery of Project	65 günler?	06.10.2021 08:00	04.01.2022 17:00			


22 Mar 21
29 Mar 21
5 Nis 21
12 Nis 21
19 Nis 21
26 Nis 21
3 May 21
10 May 21
17 May 21
24 May 21
31 May 21
7 Haz 21
14 Haz 21
Sevde Sarıkaya
İrem Zeynep Alagöz
Kutay Saran
Kıymet Akdemir
Burak Onur Duman
Halil Baydar
Abdurrahman Emre Yılmaz
Sevde Sarıkaya
Ufuk Arslan
Ufuk Arslan
Ufuk Arslan;İrem Zeynep Alagöz
Kutay Saran
Sevde Sarıkaya
Abdurrahman Emre Yılmaz
Orhun Görkem;İrem Zeynep Alagöz
Orhun Görkem;İrem Zeynep Alagöz
Bekir Keldal
Everyone
Kıymet Akdemir;Orhun Görkem
Halil Baydar
Everyone
Bekir Keldal
Bekir Keldal
Bekir Keldal
Abdurrahman Emre Yılmaz;Burak Onur Duman;Halil Baydar;İrem Zeynep Alagöz
Bekir Keldal;Kıymet Akdemir;Sevde Sarıkaya;Ufuk Arslan
Kutay Saran;Orhun Görkem
Orhun Görkem;İrem Zeynep Alagöz
İrem Zeynep Alagöz
Burak Onur Duman
Orhun Görkem
Kıymet Akdemir
Bekir Keldal
Ufuk Arslan
Halil Baydar
Sevde Sarıkaya
Abdurrahman Emre Yılmaz
Kutay Saran
Abdurrahman Emre Yılmaz;Burak Onur Duman;Kıymet Akdemir;Ufuk Arslan;İrem Zeynep Alagöz
Bekir Keldal;Halil Baydar;Kutay Saran;Orhun Görkem;Sevde Sarıkaya
Everyone
10.05.2021
Burak Onur Duman
Kıymet Akdemir
Abdurrahman Emre Yılmaz
Ufuk Arslan
Bekir Keldal;Halil Baydar;İrem Zeynep Alagöz
Orhun Görkem;Sevde Sarıkaya;İrem Zeynep Alagöz
Abdurrahman Emre Yılmaz;Halil Baydar
Kutay Saran
Burak Onur Duman
İrem Zeynep Alagöz
Halil Baydar
Abdurrahman Emre Yılmaz
Bekir Keldal
Sevde Sarıkaya
Orhun Görkem
Kıymet Akdemir
Ufuk Arslan
İrem Zeynep Alagöz
Orhun Görkem
Halil Baydar
İrem Zeynep Alagöz
Sevde Sarıkaya
Orhun Görkem
Halil Baydar
Halil Baydar;Sevde Sarıkaya
Orhun Görkem;İrem Zeynep Alagöz
Everyone
İrem Zeynep Alagöz
Orhun Görkem
İrem Zeynep Alagöz
Orhun Görkem
Ufuk Arslan
Kıymet Akdemir
Bekir Keldal
İrem Zeynep Alagöz
Halil Baydar
Halil Baydar
Sevde Sarıkaya
Sevde Sarıkaya
Burak Onur Duman;Kıymet Akdemir
Orhun Görkem;İrem Zeynep Alagöz
Abdurrahman Emre Yılmaz;Ufuk Arslan
Bekir Keldal;Halil Baydar
Kutay Saran;Sevde Sarıkaya



October 2021-January 2022

****Since view controllers are used for connection between backend and frontend, they do not belong to the API and they are not documented.**
****To view in browser: http://34.238.122.101:8080/swagger-ui/#**

Practice App 1.12.3

`[ Base URL: 34.238.122.101:8080/ ]`
http://34.238.122.101:8080/v2/api-docs

CmpE352 software of REST API

Apache 2.0

advice-controller Advice Controller

GET `/advice/` Getting an advice

This API call will return a random advice in json format.

Parameters Try it out

No parameters

Responses **Response content type** application/json

Code	Description
200	OK **Example Value** Model `string`
401	Unauthorized
403	Forbidden

Code	Description
404	Not Found

GET `/advice/search/{query}` Searching advices with a keyword

This API call will return a list of advices with the given keyword in json format.

Parameters Try it out

Name	Description
query * required `string` *(path)*	query query - query

Responses Response content type application/json

Code	Description
200	OK **Example Value** Model `string`
401	Unauthorized
403	Forbidden
404	Not Found

create-community-controller

Create Community Controller

POST `/community/create` Create Community

This method creates new community.
If language of community is not given, detects language using third-party API.
Community name and description can not be empty.
Name of the community should be **unique**.

Parameters Try it out

Name	Description
communityEntity * required `object` *(body)*	communityEntity **Example Value** Model `{ "description": "Community about the fundamentals of software engineering", "language": "string", "name": "CmpE352", "publicity": true }` **Parameter content type** application/json

```
{
  "description": "Community about the fundamentals of software engineer-
ing",
  "language": "string",
  "name": "CmpE352",
  "publicity": true
}
```

Responses **Response content type** */*

Code	Description
201	Successfully created community **Example Value** Model `string`
400	Bad Request: Ie.Mandatory fields are missing
401	Unauthorized
403	Forbidden: Name should be unique

Code	Description
404	Not Found

create-community-view-controller Create Community View Controller

create-post-controller Create Post Controller

POST `/post/save` Save Post

Implemented to create new posts and submitting to the database.

Parameters Try it out

Name	Description
postEntity * required object *(body)*	postEntity **Example Value** Model

```
{
  "author": {
    "email": "string",
    "id": {
      "date": "2021-06-13T16:23:17.324Z",
      "timestamp": 0
    },
    "profile": {
      "description": "string",
      "id": {
        "date": "2021-06-13T16:23:17.324Z",
        "timestamp": 0
      },
      "isPhotoPublic": true,
      "isPublic": true,
      "name": "string",
      "photo": "string"
    },
    "username": "string"
  },
  "communities": [
    {
      "description": "Community about the fundamentals of software
engineering",
      "language": "string",
      "name": "CmpE352",
      "publicity": true
    }
```

Parameter content type

application/json

Responses Response content type */*

Code	Description
200	Post created **Example Value** Model `0`
201	Created
400	Bad Request: Ie.Mandatory fields are missing
401	Unauthorized
403	Forbidden
404	Not Found

create-user-controller

Create User Controller

GET `/user/` getUserByName

Parameters Try it out

Name	Description
userName * required `string` *(query)*	userName userName - userName

Responses Response content type **application/json**

Code	Description
200	OK

Example Value Model

```
{
  "email": "string",
  "id": {
    "date": "2021-06-13T17:33:13.089Z",
    "timestamp": 0
  },
  "profile": {
    "description": "string",
    "id": {
      "date": "2021-06-13T17:33:13.089Z",
      "timestamp": 0
    },
    "isPhotoPublic": true,
    "isPublic": true,
    "name": "string",
    "photo": "string"
  },
  "username": "string"
}
```

401	Unauthorized
403	Forbidden
404	Not Found

GET `/user/{id}` getUser

Parameters **Try it out**

Name	Description
id * required `string` *(path)*	id id - id

Responses Response content type **application/json**

Code	Description
200	OK

Example Value Model

```
{
  "email": "string",
  "id": {
    "date": "2021-06-13T17:33:13.093Z",
    "timestamp": 0
  },
  "profile": {
    "description": "string",
    "id": {
      "date": "2021-06-13T17:33:13.093Z",
      "timestamp": 0
    },
    "isPhotoPublic": true,
    "isPublic": true,
    "name": "string",
    "photo": "string"
  },
  "username": "string"
}
```

Code	Description
401	Unauthorized
403	Forbidden
404	Not Found

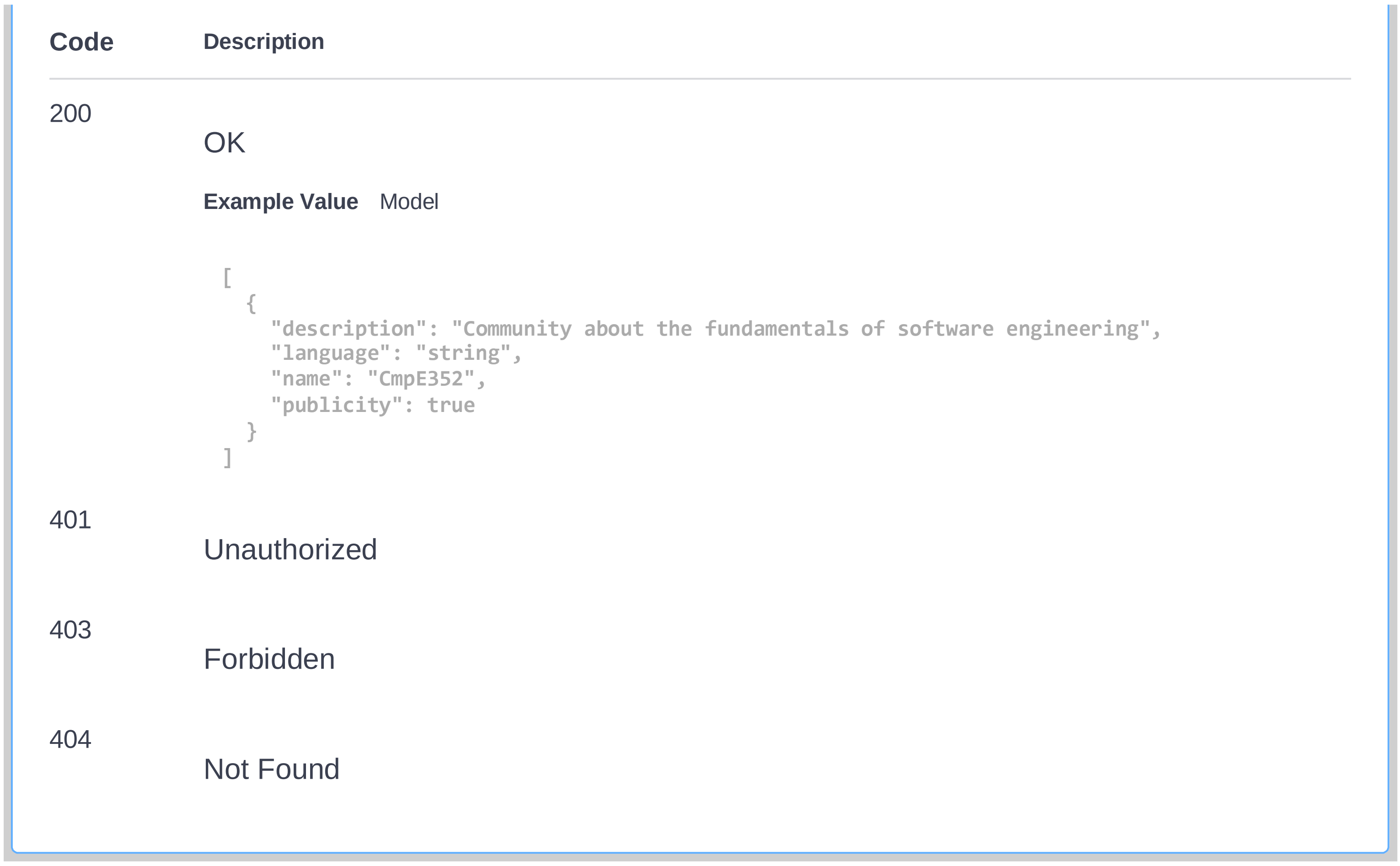
GET `/user/createdcommunities` getUserCreatedCommunitiesByName

Parameters Try it out

Name	Description
userName * required `string` *(query)*	userName userName - userName

Responses Response content type application/json

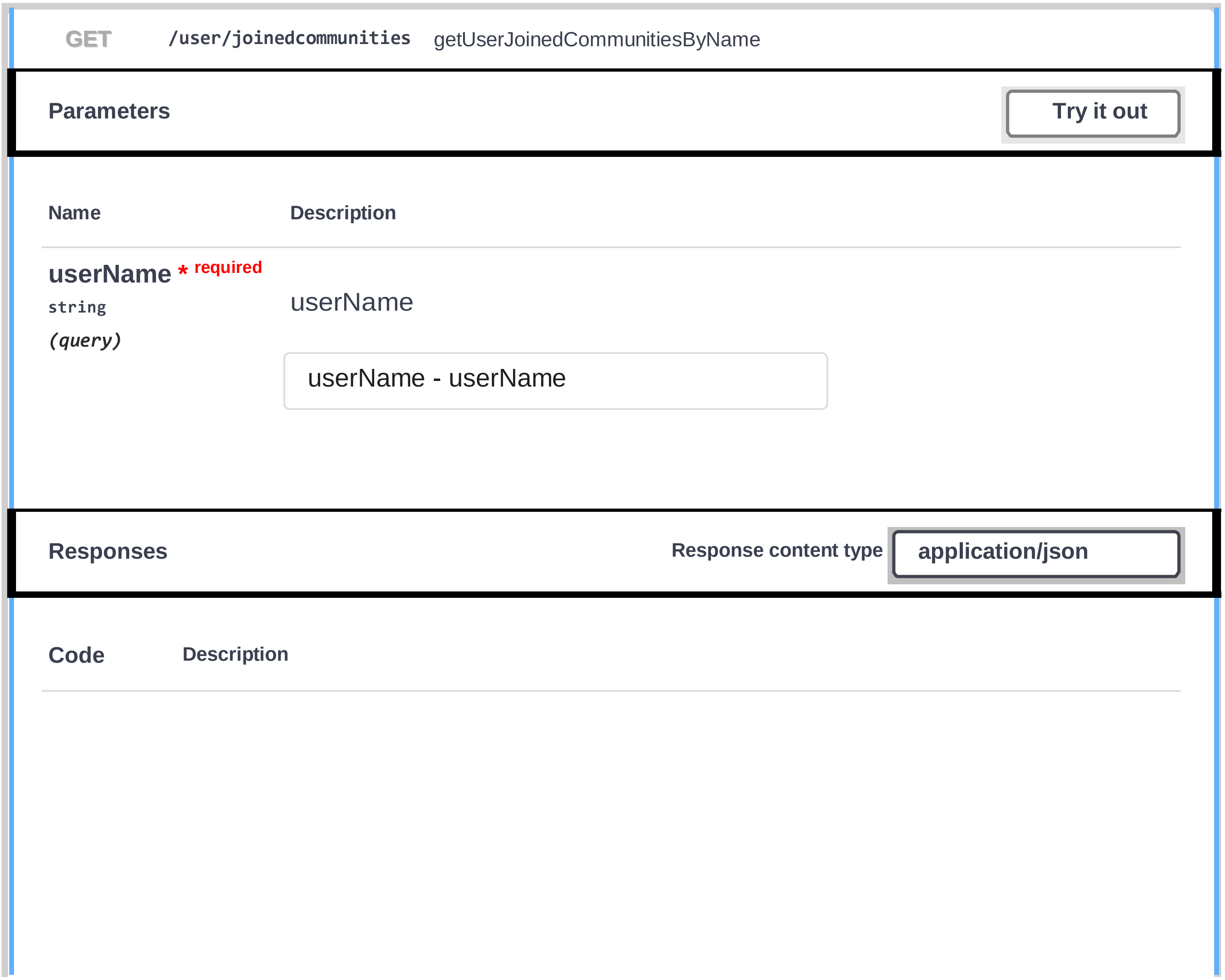
Code	Description



Code	Description
200	OK
401	Unauthorized
403	Forbidden
404	Not Found

Example Value Model

```
[
  {
    "description": "Community about the fundamentals of software engineering",
    "language": "string",
    "name": "CmpE352",
    "publicity": true
  }
]
```



GET `/user/joinedcommunities` getUserJoinedCommunitiesByName

Parameters

Try it out

Name	Description
userName * required string *(query)*	userName userName - userName

Responses

Response content type application/json

Code	Description

Code	Description
200	OK
401	Unauthorized
403	Forbidden
404	Not Found

For 200, **Example Value** Model:

```
[
  {
    "description": "Community about the fundamentals of software engineering",
    "language": "string",
    "name": "CmpE352",
    "publicity": true
  }
]
```

POST `/user/save` Save User

User can be verify by verification token

Parameters Try it out

Name	Description

Name	Description
userEntity * required `object` *(body)*	userEntity **Example Value** Model Parameter content type: application/json

```
{
  "email": "string",
  "id": {
    "timestamp": 0
  },
  "profile": {
    "description": "string",
    "id": {
      "timestamp": 0
    },
    "isPhotoPublic": true,
    "isPublic": true,
    "name": "string",
    "photo": "string"
  },
  "username": "string"
}
```

Parameter content type

application/json

Responses **Response content type** */*

Code	Description
200	OK **Example Value** Model `string`
201	Created
401	Unauthorized
403	Forbidden
404	Not Found

create-user-view-controller Create User View Controller

dictionary-view-controller Dictionary View Controller

get-communities-by-publicity-controller Get Communities By Publicity Controller

GET `/communities/` Get Communities by publicity

This method returns public or private communities based on selection
true: Public
false: Private

Parameters Try it out

Name	Description
public `boolean` *(query)*	public *Default value* : true true

Responses **Response content type** application/json

Code	Description

Code	Description
200	Successfully returned communities **Example Value** Model `string`
401	Unauthorized
403	Forbidden
404	Not Found

get-community-controller Get Community Controller

GET `/community/` getCommunityByName

Parameters **Try it out**

Name	Description

name * required
`string`
`(query)`
	name
	name - name

Responses **Response content type** **application/json**

Code	Description
200	OK

Example Value Model

```
{
  "description": "Community about the fundamentals of software engineering",
  "language": "string",
  "name": "CmpE352",
  "publicity": true
}
```

Code	Description
401	Unauthorized
403	Forbidden
404	Not Found

get-community-view-controller Get Community View Controller

get-kanye-view-controller Get Kanye View Controller

get-places-controller Get Places Controller

GET `/places/{input}` Get places

Returns the relevant places to the given input text.

Parameters Try it out

Name	Description
input * required `string` *(path)*	input input - input

Responses **Response content type** */*

Code	Description

Code	Description
200	Places returned **Example Value** Model `[ "string" ]`
401	Unauthorized
403	Forbidden

get-posts-controller Get Posts Controller

GET /community/definition/{lang_code}/{word} Get Definition

This method gets definition of a word.

Parameters Try it out

Name	Description
lang_code * required string *(path)*	lang_code lang_code - lang_code
word * required string *(path)*	word word - word

Responses **Response content type** */*

Code	Description

Code	Description
200	Successfully returned the definition. **Example Value** Model `string`
401	Unauthorized
403	Forbidden
404	Not Found
500	Internal server error. Word is not found.

GET `/community/posts/{name}` Get Posts

This method gets posts of a community.

Parameters Try it out

Name	Description
name * required `string` *(path)*	name name - name

Responses Response content type application/json

Code	Description

Code	Description
200	Successfully returned the posts. **Example Value** Model `[ {} ]`
401	Unauthorized
403	Forbidden
404	Not Found
500	Internal server error. There is no such community.

get-posts-view-controller Get Posts View Controller

index-view-controller Index View Controller

join-community-view-controller Join Community View Controller

joining-community-controller Joining Community Controller

PUT `/joiningCommunity/{user_name}/{community_name}` Joining Community

Community will be added to the user's joined communities list and the user will be added to the community's members list.

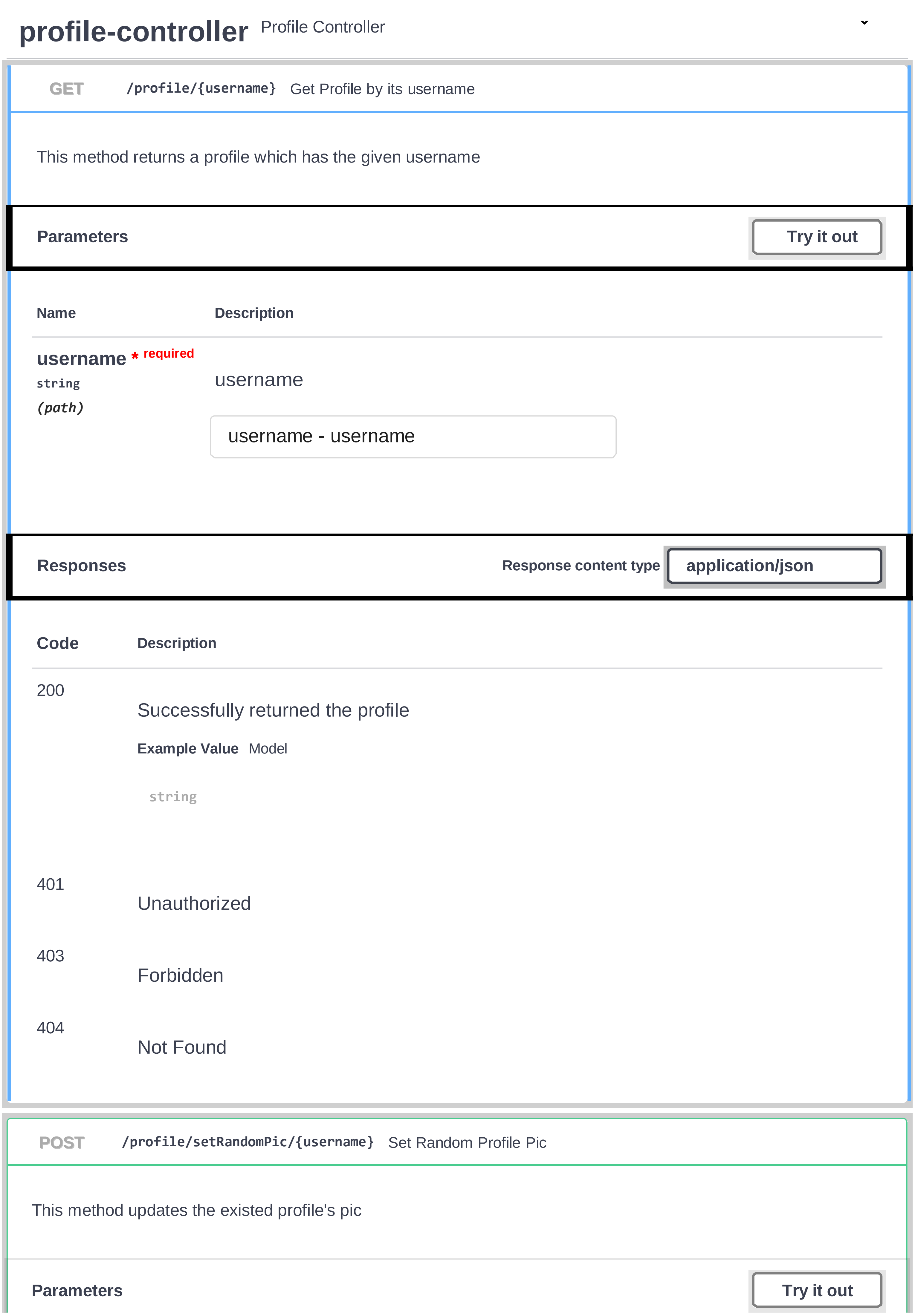
Parameters Try it out

Name	Description
community_name * required string *(path)*	community_name community_name - community_name
user_name * required string *(path)*	user_name user_name - user_name

Responses Response content type */*

Code	Description
200	OK **Example Value** Model `string`
201	Created
401	Unauthorized
403	Forbidden
404	Not Found

post-view-controller Post View Controller



profile-controller Profile Controller

GET `/profile/{username}` Get Profile by its username

This method returns a profile which has the given username

Parameters Try it out

Name	Description
username * required `string` *(path)*	username username - username

Responses Response content type application/json

Code	Description
200	Successfully returned the profile **Example Value** Model `string`
401	Unauthorized
403	Forbidden
404	Not Found

POST `/profile/setRandomPic/{username}` Set Random Profile Pic

This method updates the existed profile's pic

Parameters Try it out

Name	Description
username * required `string` *(path)*	username username - username

Responses Response content type **application/json**

Code	Description
200	OK
201	Successfully updated
400	Bad Request: Ie.Mandatory fields are missing
401	Unauthorized
403	Forbidden

200

OK

Example Value Model

```
{
  "description": "string",
  "id": {
    "date": "2021-06-13T17:33:13.131Z",
    "timestamp": 0
  },
  "isPhotoPublic": true,
  "isPublic": true,
  "name": "string",
  "photo": "string"
}
```

201

Successfully updated

Example Value Model

```
{
  "description": "string",
  "id": {
    "date": "2021-06-13T17:33:13.132Z",
    "timestamp": 0
  },
  "isPhotoPublic": true,
  "isPublic": true,
  "name": "string",
  "photo": "string"
}
```

Code	Description
404	Not Found

POST /profile/updateProfile Update Profile

This method updates the existed profile or creates new one

Parameters

Try it out

Name	Description
profile * required object *(body)*	profile **Example Value** Model

```
{
  "description": "string",
  "id": {
    "timestamp": 0
  },
  "isPhotoPublic": true,
  "isPublic": true,
  "name": "string",
  "photo": "string"
}
```

Parameter content type

application/json

Responses

Response content type application/json

Code	Description

Code	Description
200	OK **Example Value** Model
201	Successfully updated **Example Value** Model
400	Bad Request: Ie.Mandatory fields are missing
401	Unauthorized
403	Forbidden
404	Not Found

200 example:

```
{
  "description": "string",
  "id": {
    "date": "2021-06-13T17:33:13.136Z",
    "timestamp": 0
  },
  "isPhotoPublic": true,
  "isPublic": true,
  "name": "string",
  "photo": "string"
}
```

201 example:

```
{
  "description": "string",
  "id": {
    "date": "2021-06-13T17:33:13.137Z",
    "timestamp": 0
  },
  "isPhotoPublic": true,
  "isPublic": true,
  "name": "string",
  "photo": "string"
}
```

profile-view-controller Profile View Controller

universities-list-controller Universities List Controller

GET `/universitieslist/{country}` getUniversitiesList

Parameters

Try it out

Name	Description
country * required `string` *(path)*	country country - country

Responses

Response content type application/json

Code	Description
200	OK **Example Value** Model `[ "string" ]`
401	Unauthorized
403	Forbidden
404	Not Found

universities-view-controller Universities View Controller

update-post-controller Update Post Controller

GET `/post/kanye-quote` getRandomKanyeQuote

Parameters

Try it out

No parameters

Responses

Response content type */*

Code	Description
200	OK **Example Value** Model `string`
401	Unauthorized
403	Forbidden
404	Not Found

PUT `/post/update` Update Post

Implemented to update posts in database.

Parameters

Try it out

Name	Description

Name	Description
postEntity * required `object` *(body)*	postEntity

Example Value Model

```
{
  "author": {
    "email": "string",
    "id": {
      "timestamp": 0
    },
    "profile": {
      "description": "string",
      "id": {
        "timestamp": 0
      },
      "isPhotoPublic": true,
      "isPublic": true,
      "name": "string",
      "photo": "string"
    },
    "username": "string"
  },
  "communities": [
    {
      "description": "Community about the fundamentals of software
engineering",
      "language": "string",
      "name": "CmpE352",
      "publicity": true
    }
  ],
  "date": "2021-06-13T16:29:48.180Z",
```

Parameter content type

application/json

Responses

Response content type */*

Code	Description

Code	Description
200	Post updated **Example Value** Model (see example below)

```
{
  "author": {
    "email": "string",
    "id": {
      "date": "2021-06-13T17:33:13.144Z",
      "timestamp": 0
    },
    "profile": {
      "description": "string",
      "id": {
        "date": "2021-06-13T17:33:13.144Z",
        "timestamp": 0
      },
      "isPhotoPublic": true,
      "isPublic": true,
      "name": "string",
      "photo": "string"
    },
    "username": "string"
  },
  "communities": [
    {
      "description": "Community about the fundamentals of software engineering",
      "language": "string",
      "name": "CmpE352",
      "publicity": true
    }
  ],
```

Code	Description
201	Created
400	Bad Request
401	Unauthorized
403	Forbidden
404	Not Found

user-follow-controller User Follow Controller

PUT `/userfollow/{follower_name}/{followed_name}` Follow User

Parameters Try it out

Name	Description
followed_name * required `string` *(path)*	followed_name followed_name - followed_name
follower_name * required `string` *(path)*	follower_name follower_name - follower_name

Responses | **Response content type** | ***/***

Code	Description
200	OK **Example Value** Model `string`
201	Created
401	Unauthorized
403	Forbidden
404	Not Found

user-follow-view-controller User Follow View Controller

Models

```
CommunityEntity  {
    description*        string
                        example: Community about the fundamentals of software engineering

                        Description of the community

    language            string

                        Preferred language in the community

    name*               string
                        example: CmpE352

                        Name of the community

    publicity           boolean

                        Privacy setting of the community

}
```

```
PostEntity  {
    author              UserEntity  {...}

    communities             [...]
    date                string($date-time)
    location            string
    post_id             ObjectId  {...}

    text*               string
}
```

```
Profile    {
    description        string
    id                 ObjectId    {...}
    isPhotoPublic      boolean
    isPublic           boolean
    name               string
    photo              string
}
```

```
UserEntity    {
    email              string
    id                 ObjectId    {...}
    profile            Profile    {...}
    username           string
}
```