MILESTONE REPORT 2

05.12.2020

PLATON

THE ACADEMIC COLLABORATION PROGRAM

BACKEND | HALIL UMUT ÖZDEMİR, HÜSEYİN CAN BÖLÜKBAŞ, HİLAL DEMİR, ALPEREN DİVRİKLİOĞLU
ANDROID | BURAK ÖMÜR, ERTUĞRUL BÜLBÜL, ÖYKÜ YILMAZ, ORKAN AKISU
FRONTEND | AHMET DADAK, RAMAZAN YURT, BURHAN AKKUŞ, UMUTCAN UVUT

GROUP 7

CMPE 451

Contents

EXECUTIVE SUMMARY

INTRODUCTION

In the general frame of work, Platon is an academic collaboration platform that provides an environment for academicians to collaborate. The platform is open to anyone who has an interest in joining an academic research project. It is available on the web and Android, and free to use.

After registering and creating a profile, users can follow other users, and meet new people to collaborate with. If an user has a well-defined paper topic or a project proposal in their mind, they can post their ideas and search for collaborators. They can create private or public workspaces and upload files, and edit those files with the other collaborators of that workspace.

Users can view the Activity Stream on the home page, which includes news from the accounts the user is following. They can use the recommendation system, which recommends users to workspaces and workspaces to users. They can do an advanced search using filters and sorting with respect to criterias like deadline and number of collaborators needed for the workspace. System also supports semantic search in workspace, user and upcoming event search.

Anyone using Platon as a guest can search for workspaces, users, and upcoming events and see only the descriptions of the search results. To fully use the platform, they will have to register.

WHERE ARE WE

Basics

After getting feedback from the first customer milestone, we've decided to start early. We've changed the order of tasks in Project Plan in order to deliver Workspace endpoints to Android and Frontend team early. In our next meeting with the customer, our customer requested a short demo to see our progress. In a week, we've prepared a demo that mainly concentrates on critical functionalities like file system and workspace system. After getting good feedback from the customer, we've delivered most of our deliverables to the Customer Milestone 2. Detailed list of status of the deliverables can be found in the following sections.

Planning

In the previous customer milestone, due to late delivery of endpoints, Android and Frontend teams had stressful schedules. To prevent this situation, Backend team was expected to deliver the endpoints as quickly as possible. Also not every endpoint counts as the same. Some endpoints such as user search were critical. These endpoints were helping us to test other endpoints. Therefore, we've decided to change the order of terms in Project Plan to deliver User Search and Workspace System in advance. During Backend team develops these endpoints, partially delivered systems from Customer Milestone 1 are completed. Another good decision was to prepare a demo for the customer before customer milestone 2. It was very relieving to see positive feedback from the customer. Also we've managed to arrange a meeting for presentation preparation. By doing this, we've prepared a better storyline and showed most of our functionalities compared to the Customer Milestone 1 presentation.

WHAT IS NEXT

We will:

- evenly divide the tasks
- fix the currently existing bugs
- deliver the partially delivered deliverables

- create "activity stream" to allow our users to see the activities of the people they follow
- increase our code coverage by increasing the number of tests

DELIVERABLES

LIST OF DELIVERABLES

	DELIVERABLE	DUE DATE	STATUS
1	PROJECT PLAN		DELIVERED
2	BACKEND		
	REGISTER ENDPOINTS	29.12.2020	DELIVERED
	FOLLOW SYSTEM ENDPOINTS	29.12.2020	DELIVERED
	PROFILE MANAGEMENT ENDPOINTS	29.12.2020	DELIVERED
	SEARCH ENGINE ENDPOINTS	29.12.2020	DELIVERED
	WORKSPACE SYSTEM ENDPOINTS	29.12.2020	DELIVERED
	FILE SYSTEM ENDPOINTS	29.12.2020	DELIVERED
	ISSUE&MILESTONE SYSTEM ENDPOINTS	29.12.2020	DELIVERED
	UPCOMING EVENTS SYSTEM ENDPOINTS	29.12.2020	DELIVERED
	COMMENT&RATE SYSTEM ENDPOINTS	29.12.2020	DELIVERED
	API DOCUMENTATION(SWAGGER)	29.12.2020	DELIVERED
3	FRONTEND		
	LANDING PAGE	29.12.2020	DELIVERED
	REGISTER PAGE	29.12.2020	DELIVERED
	PROFILE PAGE	29.12.2020	DELIVERED
	FOLLOW SYSTEM	29.12.2020	DELIVERED
	WORKSPACE PAGES	29.12.2020	DELIVERED
	INVITATION-APPLICATION PAGES	29.12.2020	DELIVERED
	FILE SYSTEM PAGES	29.12.2020	DELIVERED
	NOTIFICATIONS PAGE	29.12.2020	DELIVERED
	TRENDING WORKSPACE PAGE	29.12.2020	DELIVERED
	UPCOMING EVENTS PAGE	29.12.2020	DELIVERED
	MILESTONE PAGE	29.12.2020	DELIVERED
	ISSUE PAGE	29.12.2020	PARTIALLY
	SEARCH SYSTEM PAGE	29.12.2020	PARTIALLY
	COMMENT-RATE PAGE	29.12.2020	NOT DELIVERED
4	ANDROID		
	LANDING PAGE	29.12.2020	DELIVERED
	REGISTER PAGE	29.12.2020	DELIVERED
	PROFILE PAGE	29.12.2020	DELIVERED
	FOLLOW SYSTEM	29.12.2020	DELIVERED
	WORKSPACE PAGES	29.12.2020	DELIVERED
	INVITATION-APPLICATION SECTIONS	29.12.2020	DELIVERED
	FILE SYSTEM PAGES	29.12.2020	PARTIALLY
	NOTIFICATIONS SECTION	29.12.2020	DELIVERED
	TRENDING WORKSPACE SECTION	29.12.2020	DELIVERED
	UPCOMING EVENTS SECTION	29.12.2020	DELIVERED
	MILESTONE SECTION	29.12.2020	DELIVERED

	ISSUE PAGE	29.12.2020	DELIVERED
	SEARCH SYSTEM PAGE	29.12.2020	DELIVERED
	COMMENT-RATE SECTION	29.12.2020	DELIVERED

EVALUATION OF DELIVERABLES

BACKEND

Register Endpoints

Explanation

There were some missing parts for the registration system in the Customer Milestone 1. Uploading profile photo functionality was missing. We implemented this functionality until this milestone.

Covered Requirements

1.1.1.1, 1.1.5.1, 1.2.6.5

Follow System Endpoints

Explanation

Users are able to follow public accounts and send requests to follow private accounts. Users with private accounts can accept/reject follow requests. Users are able to comment & rate users that are collaborated with.

Covered Requirements

1.1.3.1, 1.1.3.2, 1.1.3.3, 1.1.3.4

Profile Managements Endpoints

Explanation

There were some missing parts for the profile management system in the Customer Milestone 1. Until Customer Milestone 2, adding skills and job information to a profile implemented. Also some updates on profile privacy was done.

Covered Requirements

1.1.1.1, 1.1.5.1, 1.2.3.2

Search Engine Endpoints

Explanation

In this milestone we implemented basic semantic search for Users in our system. Also in our system we implemented an advanced search mechanism which is based on filtering search results according to some fields. Also, results can be sorted according to the criterias that are specified in the software requirements. Additionally, we implemented a search history mechanism for our system.

Users and guests are able to search public workspaces using keywords. Users and guests are able to do an advanced search on only public workspaces by specifying filters such as the founder of the project, starting date, submission deadline and skills. Users and guests are able to sort search results of workspaces according to the sorting criteria as date, and number of collaborators needed, alphabetical order. System supports semantic search.

Users and guests are able to search upcoming events using keywords. Users and guests are able to do an advanced search on only public workspaces by specifying filters such as starting and ending dates & deadlines. Users and guests are able to sort search results of workspaces according to the sorting criteria as date and alphabetical order. System supports semantic search.

Covered Requirements

1.1.1.4, 1.1.6.1, 1.1.6.2, 1.1.6.3, 1.1.6.4, 1.1.6.5, 1.2.2.1, 1.2.2.2, 1.2.2.3, 1.2.2.4, 1.2.2.5, 2.1.2

Workspace System Endpoints

Explanation

The necessary endpoints for a user to create, read, update and delete workspaces have been implemented. Using the “create workspace” endpoint (by sending POST requests to /workspaces), users can create a new workspace. Users, while creating a workspace, can set the title, description, privacy status, maximum number of collaborators, deadline, requirements for a prospective contributor to join that workspace, skills that a prospective contributor needs to have to join that workspace, the upcoming events of that workspace. Users can change the state (and the other workspace information mentioned above) of a workspace using “update workspace” endpoint (by sending PUT request to /workspaces).Users can view the workspaces that they are permitted to view (all public workspaces and the private workspaces of which they’re currently a contributor), and users can delete the workspaces that they’ve created. Using the invitations, a user who created a workspace can invite other users for collaboration for that specific workspace. Using the applications, a user can apply for collaboration in the workspace.

Covered Requirements

1.1.4.1, 1.1.4.2, 1.1.4.3, 1.1.4.4, 1.1.4.5, 1.1.4.6, 1.1.4.9, 1.1.4.10, 1.1.4.11, 1.1.4.14, 1.1.4.15, 1.2.5.1, 1.2.5.2, 1.2.5.3

File System Endpoints

Explanation

In our workspace system, there is a file system which provides users the capability of uploading, downloading, editing and deleting files which are related to their workspace. Also to improve the user experience, a folder system was implemented. By using this folder system files can be stored in an organized way for each workspace in our system.

Covered Requirements

1.1.4.7, 1.1.4.8

Issue and Milestone System Endpoints

Explanation

Collaborators are able to create, close and reopen issues and assign any collaborator. Collaborators are able to create milestones and add them to the project timeline to distribute requirements and tasks, and also to keep track of the progress.

Covered Requirements

1.1.4.12, 1.1.4.13

Upcoming Events System Endpoints

Explanation

According to software requirements, our system has to consist of a mechanism which provides some information about the upcoming conferences, journal special issues, submission deadlines, and CFPs. Another requirement about this system is updating this information periodically. As a result, this information was periodically (once a day) fetched from a website which is cited in our website and Android application. We stored their acronym, title, date, deadline, and a link which provides additional information about the event.

Covered Requirements

1.1.1.5, 1.2.1.2

Comment Rate System Endpoints

Explanation

Users can comment and rate users that are collaborated with

Covered Requirements

1.1.3.4

FRONTEND

Landing Page

Explanation

(This was partially delivered in the Milestone1) Guests can see upcoming events and their details, and project descriptions. Guests can see trending workspace.

Covered Requirements

1.1.1.5, 1.1.1.6

Register Page

Explanation

(This was partially delivered in the Milestone1) Guests can add his/her jobs while registering.

Covered Requirements

1.1.1.1

Profile Page

Explanation

(This was partially delivered in the Milestone1.) Users shall be able to follow public accounts. Users shall be able to send requests to follow private accounts. Users with private accounts shall be able to accept/reject follow requests. Users shall be able to edit their account information which includes name, surname, e-mail, affinities, profile photo, research, ResearchGate and Google Scholar accounts. Users shall be able to set their profiles as either public or private. Users shall be able to link their Google Scholar & ResearchGate accounts. A user shall be able to view public profile pages. Users shall be able to view private profiles that they are following. Users shall be able to view sections as followers, comments, ratings, photo, name and e-mail in a profile. The platform shall provide definitive labels according to the user's skills. A profile page shall be public or private. Profile Page of any user shall be different from each other at least by e-mail information. Profile page shall include the research information which is fetched from Google Scholar or ResearchGate account of the user if this accounts are provided to the system. Profile page shall provide the follower information which is the number of followers, the number of followed users and links to profile pages followers and followed users. Profile page shall provide account information which includes name, surname, e-mail certainly and affinities, profile photo, research, ResearchGate and Google Scholar accounts if these information is provided to the system. A public profile page and all information provided (described in 1.2.6.3, 1.2.6.4, 1.2.6.5) shall be able to be viewed by any user in the system. A private profile page and all information provided (described in 1.2.6.3, 1.2.6.4, 1.2.6.5) shall be able to be viewed by only the followers of the owner of the profile page.

Covered Requirements

1.1.3.1, 1.1.3.2, 1.1.3.3, 1.1.5.1, 1.1.5.2, 1.1.5.3, 1.1.5.4, 1.1.5.5, 1.1.5.7, 1.2.3.2, 1.2.6.2, 1.2.6.3, 1.2.6.4, 1.2.6.5, 1.2.6.6, 1.2.6.7

Workspace Page

Explanation

Users can create a workspace by providing required fields as title, description, and optional fields as workspace privacy, deadline, the maximum number of collaborators, requirements, and skill tags. Users can set the workspace privacy as public and private. Users can edit all fields of the workspaces. Users can delete their workspaces. Users can update the state of their workspaces. Users can quit the workspaces. Users can attach upcoming events to their workspaces. Users can view the files of the public workspaces that they are not a collaborator in Search for Collaborators and Published State.

Covered Requirements

1.1.4.1, 1.1.4.6, 1.1.4.9, 1.1.4.10, 1.1.4.11, 1.1.4.14, 1.1.4.15

Invitation-Application Pages

Explanation

Users can invite other users to their workspaces at Search for Collaborators. Users can apply to the workspaces at Search for Collaborators State or Ongoing State. Users can accept/reject workspace invitation requests. Users can accept/reject workspace application requests.

Covered Requirements

1.1.4.2, 1.1.4.3, 1.1.4.4, 1.1.4.5

File System

Explanation

Users can upload/view/edit/delete files. Users can organize files as folder structure.

Covered Requirements

1.1.4.7, 1.1.4.8

Notifications Page

Explanation

Users can see their notifications. Users can delete their notifications.

Covered Requirements

1.2.4.2

Trending Workspace Page

Explanation

Guests and users can see the trending workspaces.

Covered Requirements

1.1.1.6.

Upcoming Events Page

Explanation

Guests and users can see the upcoming events on the landing and the home page respectively. The upcoming events data is taken from CFP API. They can see the details, deadlines, locations and visit the provided link.

Covered Requirements

1.1.1.5, 1.1.1.6, 1.2.1.2

Milestones Page

Explanation

Users can add milestones providing a title, description and a deadline. Users can edit the milestones. Users can delete the milestones.

Covered Requirements

1.1.4.13

Issues Page

Explanation

Members of a workspace can create issues and assign members of the workspace to work on issues. Members can also comment on any issue and add a deadline.

Covered Requirements

1.1.4.12

Search System

Explanation

A user and a guest shall be able to search public workspaces, upcoming events, and users using keywords. Users and guests shall be able to do an advanced search on only public workspaces, upcoming events, and any profile by specifying filters such as founder of the project, starting date, submission deadline, research area/topic and labels. Users and guests should be able to sort search results of workspaces according to the sorting criteria as date, and number of collaborators needed, alphabetical order. Users and guests should be able to sort search results of upcoming events according

to the sorting criteria as date, alphabetical order. Users and guests should be able to sort search results of users according to the sorting criteria as alphabetical order. System shall provide a mechanism that users can use to search. Search engine shall search content concerning the research area, topic, and scope. System shall support searching for semantically related content.

Covered Requirements

1.1.6.1, 1.1.6.2, 1.1.6.3, 1.1.6.4, 1.1.6.5, 1.2.2.1, 1.2.2.2, 1.2.2.4

ANDROID

Landing Page

Explanation

(This was partially delivered in the Milestone1). Guests are able to do basic & advanced search using filters, see upcoming events and their details, see trending projects. Requirements are fully satisfied.

Covered Requirements

1.1.1.4 1.1.1.5 1.1.1.6

Register Page

Explanation

(This was partially delivered in the Milestone1) Users can now add their institution and job information while registering. Requirements are fully satisfied.

Covered Requirements

1.1.1.1.

Profile Page

Explanation

(This was partially delivered in the Milestone1). Users were able to view profile information, followers and following, research information and link ResearchGate & Google Scholar accounts in the first milestone. Now, they can see comments, invite users to workspaces, see the rating, upload a profile photo, add, update and delete research information, add and delete skills, see their followers and following. Users are able to view details of the profiles which are public, or prive but they are following. Users are able to see only the profile photo, name and surname of the private profiles they are not following. Requirements are fully satisfied.

Covered Requirements

1.1.5.1, 1.1.5.2, 1.1.5.3, 1.1.5.4, 1.1.1.5, 1.1.1.6, 1.1.5.7

Follow System

Explanation

Users are able to follow and unfollow users, and send follow requests to private accounts. Users are able to accept or reject follow requests. Requirements are fully satisfied.

Covered Requirements

1.1.3.1 1.1.3.2 1.1.3.3

Workspace Page

Explanation

Users are able to create a workspace by providing title, description, deadline, maximum number of collaborators, requirements, and skill tags. Users can set the privacy of the workspace as public and private. Users can edit and delete their workspace, add or delete skills and requirements afterwards. Users can update the state of their workspace. Users are able to assign or remove upcoming events to their workspaces. Users are able to view only the public workspaces. Users are able to quit workspace if they are not the creator. Only the creator can delete the workspace. Requirements are fully satisfied.

Covered Requirements

1.1.4.1,, 1.1.4.6, 1.1.4.14, 1.1.4.15, 1.2.5.1, 1.2.5.3

Invitation-Application Sections

Explanation

Users can send invitations for one of their workspaces to other users by going to their profile page. Users are able to accept and reject invitations.

Users are able to send applications to the public workspace in search for collaborators or ongoing state workspaces on the workspace page. Users are able to accept and reject applications. Requirements are fully satisfied.

Covered Requirements

1.1.4.2, 1.1.4.3, 1.1.4.4, 1.1.4.5

File System

Explanation

Users are able to upload files such as c, cpp, py, md, txt and etc, delete files, create folders, delete folders to the workspaces that they belong to as a collaborator. Users are able to navigate between folders by changing directory.

Covered Requirements

1.1.4.8, 1.1.4.7

Notifications

Explanation

Users are able to see their notifications such as somebody followed, or sent follow request, sent workspace invitation etc. They can also choose to remove the notifications that they do not want to see anymore. Also, notifications are sorted in the descending date order, and notifications are paginated. Requirements are fully satisfied.

Covered Requirements

1.2.4.2, 1.2.4.3, 1.2.4.4, 1.2.4.5

Trending Workspace Section

Explanation

Guests and users are able to see the trending workspaces. Only the public workspaces can be a trending workspace. Requirements are fully satisfied.

Covered Requirements

1.1.1.6.

Upcoming Events Section

Explanation

Users and guests are able to see the upcoming events in the home page. The data is taken from CFP API. They can see the details, deadlines, locations and go to the provided link. Requirements are fully satisfied.

Covered Requirements

1.1.1.5, 1.2.1.2

Milestones Section

Explanation

Users can add milestones providing a title, description and a deadline. Users are able to delete and edit the milestones. Requirements are fully satisfied.

Covered Requirements

1.1.4.13

Issues Page

Explanation

Users can add issues to their workspace by providing a title, description, and deadline. Users are able to see the issues of their workspace. Users are able to add comments to the issues. Users are able to assign collaborators to the issues, and change the assignees. Requirements are fully satisfied.

Covered Requirements

1.1.4.12

Search System

Explanation

Users and guests are able to search workspaces using keywords. Only public workspaces are visible in the search results. Users and guests are able to use filters to do advanced search by specifying filters as the founder of the project, starting dates, deadlines, assigned upcoming events, and skills. Users and guests are able to sort the search results according to the date (ascending and descending), alphabetical order (ascending and descending), and the number of collaborators needed (ascending and descending). When the user or guest clicks on the search result, they are directed to the workspace's page.

Users and guests are able to search users using keywords. Both public and private users are visible in the search results. Users and guests are able to use filters to do advanced search by specifying a filter as a job of the user. Users and guests are able to sort the search results according to alphabetical order (ascending and descending). When the user or guest clicks on the search result, they are directed to the user's page.

Users and guests are able to search upcoming events using keywords. Users and guests are able to use filters to do advanced search by specifying filter as deadline and starting date. Users and guests are able to sort the search results according to alphabetical order (ascending and descending) and date (ascending and descending). When the user or guest clicks on the search result, they are directed to the wikicfp.com.

Users are able to see their search history while typing to the search bar for users, workspaces, and upcoming events separately. Requirements are fully satisfied.

Covered Requirements

1.2.2.3 1.1.1.4 1.1.6.1, 1.1.6.2, 1.1.6.3 1.1.6.4 1.1.6.5

Comment Rate Sections

Explanation

Users can add issues to their workspace by providing a title, description, and deadline. Users are able to see the issues of their workspace. Users are able to add comments to the issues. Users are able to assign collaborators to the issues, and change the assignees. Requirements are fully satisfied.

Covered Requirements

1.1.4.12

WORK DONE

SUMMARY OF WORK DONE

BACKEND

TEAM MEMBER	CONTRIBUTION
HALİL UMUT ÖZDEMİR	Implemented a class which provides Semantic Search Functionality Implemented UserSearchAPI(GET) which provides semantic search and advanced search and also its unit tests Implemented semantic search functionality of WorkspaceSearchAPI(GET) Implemented SearchHistoryAPI(GET) which stores and provides search history of each user and its unit tests Implemented uploading profile photo functionality in UserAPI(PUT), ProfilePhotoAPI(GET), DefaultProfileAPI(GET) Implemented a FileSystemAPI(GET,POST,PUT,DELETE) and FolderSystemAPI(GET,POST,PUT,DELETE) Implemented UpcomingEventsAPI(GET) and a class which fetches and updates upcoming event information periodically and their unit tests Implemented complete version of the comment system(GET,POST,DELETE) and solved its bugs Implemented privacy mechanism of the workspaces Updated profile privacy mechanism Implemented TrendingWorkspacesAPI(GET) and a background process which calculates the trending scores according to number of clicks Implemented GetWorkspacesAPI(GET) Implemented GetUserWorkspaces(GET) Implemented QuitWorkspaceAPI(GET) Reviewed pull requests of the backend team Solved some bugs of the backend Solved the problem of deployment

TEAM MEMBER	CONTRIBUTION
HÜSEYİN CAN BÖLÜKBAŞ	Implemented IssueAPI(GET, POST, PUT, DELETE) and their unit tests. Implemented IssueAssigneeAPI(GET, POST, DELETE) and their unit tests. Implemented IssueCommentAPI(GET, POST, DELETE) and their unit tests. Implemented MilestoneAPI(GET, POST, PUT, DELETE) and their unit tests. Implemented WorkspaceSearchAPI(GET) with H. Umut Özdemir and implemented its unit tests. Reviewed merge requests of the backend team.
HİLAL DEMİR	Implemented Upcoming Events Search(GET) and its unit tests. Implemented the basic version of the comment system(GET,POST,DELETE) Implemented UserSkillsAPI(GET,POST,DELETE) and their unit tests. Kept track of the Github Issues and reviewed merge requests of the backend team.
ALPEREN DİVRİKLİOĞLU	Implemented the RESTful API (create, read, update, delete) for the Workspace resource Implemented the RESTful API (create, read, update, delete) for the Invitations (for Workspaces) resource Implemented the RESTful API (create, read, update, delete) for the Application (for Workspaces) resource Reviewed the pull requests of other team members

FRONTEND

TEAM MEMBER	CONTRIBUTION
HASAN RAMAZAN YURT	Completed implementation of profile page. Implemented follow request and follow response mechanism. Implemented profile editing component. Implemented add and edit skill component. Implemented filtered search interface for user, workspaces, and upcoming events. Reviewed merge request of the frontend team.
AHMET DADAK	-Implemented create/edit/delete research information section in profile page. -Implemented the page routing of the app. -Implemented notification section and applied pagination. -Implemented workspace application section. -Implemented workspace invitation section with Umutcan. -Implemented all pages of workspace. -Implemented the file system where upload/preview/edit/delete operations are handled. -Implemented folder structure in file system. -Fixed the logout bug, homepage related bugs, job and skills bug in the profile page. -Fixed the out of memory error with the backend team members Alperen, Can, and Umut. -Responsible for the whole deployment process of frontend. -Reviewed pull requests of the frontend team
BURHAN AKKUS	-Implemented Issues part. -Added styling for issues. -Added page by page display of issues. -Added reply functionality to issues. -Reviewed pull requests.
UMUTCAN UVUT	-Implemented Trending Projects -Implemented Upcoming Events -Implemented milestone page to Workspace -Implemented invitation to workspace functionality to Profile Page with Ahmet. -Added styling for all implementations. -Added pagination for trending projects and upcoming events. -Reviewed pull request and attended debug sessions with the team.

ANDROID

TEAM MEMBER	CONTRIBUTION
BURAK ÖMÜR	Participated in the sessions for converting MVP to MVVM. We coded simultaneously. Changes were made on the Home activity. Converted the login, registration, home fragments to MVVM. Implemented the workspace search, user, and upcoming event search with their own special filters. Implemented Resource class for request to wrap in. Implemented the pagination of researches at profile page, search results, notifications. Implemented the comment rate section in the profile. Implemented progress bar and changed the previous code where it was not used. Implemented the pagination for followers and followings pages with Oyku. Created new activity for workspaces with Oyku. Implemented delete research information. Implemented notifications for users. Implemented add skills section for users. Implemented job section to register and edit profile. Implemented get requests for upcoming events. Implemented get requests for trending projects. Implemented the file upload for workspaces. Implemented the profile photo upload section. Converted some fragments into dialogs such as edit profile, add research etc. Implemented the workspace invitation notifications at notifications section. Implemented the workspace application's dialog user interface.
ERTUĞRUL BÜLBÜL	Implemented issue and issue detail pages. Implemented issue's dialog user interfaces. Implemented issue add comment and add/delete assignee mechanism. Researched about lottie. Implemented lottie animation.

TEAM MEMBER	CONTRIBUTION
ÖYKÜ YILMAZ	Participated in the sessions for converting MVP to MVVM. We coded simultaneously. Changes were made on the Home activity. Converted profile fragments to MVVM. Implemented the pagination for followers and followings pages with Burak. Implemented add edit research information. Implemented changes on the backend in followers and follower sections, others profile pages. Implemented unfollow. Helped Burak implementing add skills section to users. Implemented the workspace list page for users' workspaces. Implemented add, delete, edit, and listing milestones for workspaces. Implemented quit workspace. Converted edit workspace to dialogs after Burak converted the profile edit into dialogs. Implemented add, delete, list, and edit folders. Implemented the UI for the files. Created new activity for workspaces with Burak. Implemented accept and reject workspace invitations. Implemented send, accept and reject workspace applications. Implemented UI change and showing the details of upcoming events, and the pagination. Implemented UI change and showing the details of trending projects. Implemented accept and reject follow requests.
ORKAN AKISÜ	Test and debugged pagination Test and debugged workspace, user and event searches Test and debugged lottie

UNIT TESTS

HILAL DEMIR (BACKEND)

Upcoming Event Search Unit Tests (sha: 40a7007f3da8447fb211035e3a12375284df9d6c):

I implemented 4 unit tests for Upcoming Events. All tests have passed with their expected results and expected status codes which is 200.

1. (GET) test_name_search for searching a keyword in the names of the upcoming events. Keyword “simulation” is passed as a searching query and the expected result is the two of the upcoming events contain “simulation” in their event names from the test environment database. The test has passed.
2. (GET) test_location_search for searching a keyword in the location of the upcoming events. Keyword “malta” is passed as a search query. It may seem like the result should contain only one event but since there is semantic search in the system, “portugal” keyword is also searched in the upcoming events. So, the expected result is two of the upcoming events contain “malta” or “portugal” in their location names from the test environment database. The test has passed.
3. (GET) test_deadline_search for advanced search with specific deadlines. Keyword “simulation” is passed as a searching query and (2020,12,1,23,59,59) datetime data is passed as deadline_filter_start. This searches the events that contain simulation in their names and filters the results which their deadlines start after the deadline_filter_start. There are 2 results that pass these conditions. Expected result is true so the test has passed.
4. (GET) test_sort for sorting the results. Keyword “simulation” is passed as a searching query and “1” is passed as sorting_criteria which means sorting according to starting dates. In the test_name_search test the results were displayed according to semantic rating. In this test, results are same but their appearance is different since this time results are sorted according to their starting dates. Expected result is true so the test has passed.

User Skill Unit Tests (sha: c6e10808b68dd8c54648a1ad256fa82ce5fe537a):

I implemented 1 unit test for User Skills. The test has passed with its expected result and expected status code which is 200.

1. (POST) test_post_userskill for posting a skill into a user's profile. The test generates a valid token since the POST endpoint requires authentication token as header. Then it sends the skill “javascript” with the valid token to the endpoint and receives the response which is 'Skill is successfully added'. Expected result is true so the test has passed.

HALIL UMUT OZDEMIR (BACKEND)

Base Test(sha: 0ff7b14bd97b45db572debd408979164fe464887)

I implemented a Base test class to configure the test database and a testing mechanism which removes all of the contents of the test database after each test case.

Login Unit Tests (sha: 83d5c59890936211a0e67c4ba267db14da0e80b4, fbd9fa97e94fa36da9380994d778c0a16c39fb01)

I implemented 3 test cases to validate the functionality of the login endpoint. At the beginning of each test I created artificial data on the test database. Artificial data contains possible users of the system which contains different properties with respect to account validity.

1. test_valid_login: Checks the login endpoint with the valid data. The expected result is a valid token and 200 HTTP status code.
2. test_invalid_login: Checks the login endpoint with the invalid data. Given password is not the real password of the user. The expected result is an error message and 401 HTTP status code.
3. test_not_activated_user: Checks the login endpoint with the valid data. But the user which has these credentials are not valid. Therefore, the expected result is an error message and 401 HTTP status code.

Reset Password Test(sha: 2005e36d687704551038539 63efe8ab49ff6e6e4)

I implemented 4 test cases to validate the functionality of reset password endpoint. At the beginning of each test I created artificial data on the test database. Artificial data contains possible users of the system which contains different properties with respect to account validity.

1. test_reset_password_valid: Checks the reset password endpoint with the data. The expected result is a valid response message and 200 HTTP status code.
2. test_reset_password_invalid_input: Checks the reset password endpoint with the invalid data. In the given data, the passwords do not match. The expected result is an error message and 400 HTTP status code.
3. test_reset_password_invalid_input2: Checks the reset password endpoint with the invalid data. In the given data password is not written twice. The expected result is an error message and 400 HTTP status code.
4. test_reset_password_invalid_token: Checks the reset password endpoint with the invalid data. In the given data, the given token is not valid. So it is an unauthorized call. The expected result is an error message and 401 HTTP status code.

Research Information Tests(sha: 8fedb92915d181a3ccfccf437612869e913b43b7, 81ef390fa900a24f5af2f9982f9f3033844b300c)

I implemented 6 test cases to validate the functionality of the research information system. At the beginning of each test I created artificial data on the test database. Artificial data contains possible users of the system and their possible research information in our system.

1. test_add_research_info_valid: Checks the POST request of research information endpoint with the valid data. The expected result is a valid response message and 201 HTTP status code. Also new record must be added to the test database.

2. test_add_research_info_invalid: Checks the POST request of research information endpoint with the invalid data. The year information is missing in the data. The expected result is an error message and 400 HTTP status code. Also new record must not be added to the test database.
3. test_remove_research_info_valid: Checks the DELETE request of research information endpoint with the valid data. The expected result is a valid response message and 201 HTTP status code. Also new record must be deleted from the test database.
4. test_remove_research_info_invalid: Checks the DELETE request of research information endpoint with the invalid data. The owner of the token and the owner of the research information are not the same. The expected result is an error message and 400 HTTP status code. Also new record must not be deleted from the test database.
5. test_fetch_RG_info: Checks the fetch ResearchGate information with artificial data. Ther expected result of the function is specified in the unit test.
6. test_fetch_GS_info: Checks the fetch GoogleScholar information with artificial data. Ther expected result of the function is specified in the unit test.

Notification Tests (sha: 47c82110470d5fc81ab29999da57621aefbea4de)

I implemented 5 test cases to validate the functionality of the notification system. At the beginning of each test I created artificial data on the test database. Artificial data contains possible users of the system and their possible notifications in our system.

1. test_add_notification: Checks the POST request of notification endpoint with the valid data. The expected result is a valid response message and 201 HTTP status code. Also new record must be added to the test database.
2. test_get_notification_valid: Checks the GET request of notification endpoint with the valid data. The expected result is the list of notifications and 200 HTTP status code.
3. test_get_notification_invalid: Checks the GET request of notification endpoint with the invalid data. Given token does not have the true format. The expected result is an error message and 401 HTTP status code.
4. test_delete_notification_valid: Checks the DELETE request of notification endpoint with the valid data. The expected result is a valid response message and 200 HTTP status code. Also new record must be deleted from the test database.
5. test_delete_notification_invalid: Checks the DELETE request of notification endpoint with the invalid data. Given notification id does not exist in the test database. The expected result is an error message and 404 HTTP status code. Also new record must not be deleted from the test database.

Search History Tests (sha: 434401ae41de337d11834db24d4f12715fcf1e5b)

I implemented 4 test cases to validate the functionality of the search history system. At the beginning of each test I created artificial data on the test database. Artificial data contains possible users of the system and their possible search history in our system.

1. test_user_search_history, test_workspace_search, test_ue_search_history: Checks the GET request of search history endpoint for 3 types of searhc in our system with the valid data. The expected result is the list of search history ordered by number of use and 200 HTTP status code.
2. test_empty_result: Checks the GET request of search history endpoint for the empty response. The expected result is an empty list and 200 HTTP status code.

User Search Tests (sha: 7bf8dd85144c6edb8fd4cef8453e70435d9c1a10)

I implemented 7 test cases to validate the functionality of the search engine. At the beginning of each test I created artificial data on the test database. Artificial data contains possible users of the system and their possible search history in our system.

1. test_name_search: Checks the search engine by searching a user with the name of this user.
2. test_no_result_search: Checks the search engine by a search query which does not have any related user.
3. test_no_result_search_filter: Checks the search engine by a search query which does not have any related user. The reason why there is no related user is the given filters to the search engine.
4. test_skill_search: Checks the search engine by searching a user with s skill of this user.
5. test_job_search: Checks the search engine by searching a user with the job of this user.
6. test_semantic_search: Checks the search engine by searching a user with the "Schoolman" search query. Because a user has a job of "PhD Student" this user matches with the search query.
7. test_institution_search: Checks the search engine by searching a user with the institution of this user.

Upcoming Events Test (sha: 67e3a56568a99e746f6fae4ad0553fefece0f1ac)

I implemented 3 test cases to validate the functionality of the upcoming events system.

1. test_parse_event: Checks the function which parses an upcoming event from the HTML of the given event.
2. test_get_number_of_pages: Checks the function that fetches the number of pages of the website that upcoming events are fetched.
3. test_parse_page: Checks the function that fetches the upcoming events with the given page index.

HUSEYIN CAN BOLUKBAS (BACKEND)

Workspace Search (sha: 516b83e5065472bda6fa0f26d547b8b5a1807b02)

Unit tests checks below cases:

- if the searched element appears in the workspace title.
    - if searched element is one word query
    - if searched element is multiple words
    - if searched element is multiple words and contains white spaces
- if searched element appears in the workspace description
    - if searched element is one word query
    - if searched element is multiple words
    - Check if upper case letters are ignored while searching.
- Check if private workspaces do not appear in the search.
- Check if the searched element does not appear in the title but is similar to a word in title semantically.
- Check if the searched element does not appear in description but is similar to a word in description semantically.
- Check if the searched element does not appear in the title or description. It should return no values.
- Check if the skill filter works.
    - Check that without a skill, workspace_ids 3 and 5 is found with "bravest" query.
    - Check that with a skill, only workspace_id=5 is found with "bravest" query.
- Check if the starting date filter works.
    - Check that without a date filter, workspace_ids 3 and 5 is found with "bravest" query.
    - Check that with a date filter, nothing is found.
- Check if the deadline filter works.
    - Check that without a deadline filter, workspace_ids 3 and 5 is found with "bravest" query.
    - Check that with a deadline filter, only workspace_id=5 is found with "bravest" query.
- Check if sorting criterias work.
    - check if sorting with respect to date works
    - check if sorting with “number of collaborators needed” works
    - check if sorting with alphabetical order works
- Check if the founder name and surname filter works.

Milestone API (sha: 0a43f2a32a95e08ecdc2bf66afa662fd58f26e83)

Unit tests checks below cases:

- Can will try to get the milestones of a public workspace in which there is no milestone.
- Can will try to get the milestones of a public workspace. It should be successful.
- Umut will try to get the milestones of a private workspace. It should be unsuccessful.
- Can will try to get the milestones of a private workspace in which he is an active contributor. It should be successful.
- Can will try to get the milestones of a private workspace in which he is not an active contributor anymore. It should be unsuccessful.
- Can will try to post a milestone into a workspace in which he is a contributor. Success.
- Can will try to post a milestone into a workspace in which he is not a contributor. Fail.
- Can will try to update a milestone at a workspace in which he is a contributor. Success.

- Can will try to update a milestone at a workspace in which he is not a contributor. Fail.
- Can will try to delete a milestone at a workspace in which he is a contributor. Success.
- Can will try to delete a milestone at a workspace in which he is not a contributor. Fail.

Issue API (sha: 3d089c8069ef6027ce743e6c3db2f8ba1ec4ffbb)

Unit tests checks below cases:

- Can will try to get the issues of a public workspace. It should be successful.
- Umut will try to get the issues of a private workspace. It should be unsuccessful.
- Can will try to get the issues of a private workspace in which he is an active contributor. It should be successful.
- Can will try to get the issues of a private workspace in which he is not an active contributor anymore. It should be unsuccessful.
- Can will try to post an issue into a workspace in which he is a contributor. Success.
- Can will try to post an issue into a workspace in which he is not a contributor. Fail.
- Can will try to update an issue at an workspace in which he is a contributor. Success.
- Can will try to update an issue at a workspace in which he is not a contributor. Fail.
- Can will try to delete an issue at a workspace in which he is a contributor. Success.
- Can will try to delete an issue at a workspace in which he is not a contributor. Fail.

Issue Assignee API (sha: 3d089c8069ef6027ce743e6c3db2f8ba1ec4ffbb)

Unit tests checks below cases:

- Can will try to get the issue assignees at a public workspace. Success.
- Can will try to get the issue assignees at a private workspace. Fail.
- Can will try to get the issue assignees at a private workspace in which he is a contributor. Success.
- Hilal will try to post an issue assignee at a workspace in which she is a contributor. Assignee(Umut) is also the contributor of the workspace. Success.
- Can will try to post an issue assignee at an workspace in which he is a contributor. Assignee is not a contributor of the workspace. Fail.
- Can will try to post issue assignee at a workspace in which he is not a contributor. Fail.
- Can will try to post an issue assignee at an workspace in which he is a contributor. However, that person already exists in the issue assignees. Fail.
- Can will try to remove issue assignee at a workspace in which he is a contributor. Success.
- Can will try to remove issue assignee at a workspace in which he is not a contributor. Fail.

Issue Comment API (sha: 3d089c8069ef6027ce743e6c3db2f8ba1ec4ffbb)

Unit tests assures below cases with their expected result:

- Can will try to get the issue comments at a public workspace. Success.
- Can will try to get the issue comments at a private workspace. Fail.
- Can will try to get the issue comments at a private workspace in which he is a contributor. Success.
- Can will try to post issue comments at a workspace in which he is a contributor. Success.
- Can will try to post issue comments at a workspace in which he is not a contributor. Fail.
- Can will try to remove issue comments at a workspace in which he is a contributor. Also it is his comment. Success.

- Can will try to remove issue comments at a workspace in which he is a contributor. However it is not his comment.Fail.
- Can will try to remove issue comments at a workspace in which he is not a contributor. Fail.

GIT WORKFLOW

BACKEND

Introduction

As the backend team, our choice of web application framework to use is Flask, which has given us a greater flexibility in creating our directory structure. Therefore, instead of the exact directory structure shown in the official Flask documentation, we use a modified and a Django-like version of it: we have factored our app into modules -where related functionalities are located together. (for example, "Create Workspace", "Create Issue" and "Create Milestone" functionalities are all located in the "workspace_system" folder/module.) In each module, we have "models.py", "views.py", "forms.py" and "helpers.py".

"models.py"

Contains the description of the data models to be used in that module,

"views.py"

Contains the methods for each endpoint,

"forms.py"

Contains the forms for the endpoints that receive data from the client, those forms makes it convenient to validate input data.

"helpers.py"

Contains the methods that are not the direct equivalent of the endpoints but that help them.

This directory structure has streamlined our development process, as we know where we should write the code of a new feature or where a bug to be fixed is located.

Group Process

We have utilized Git and its branching feature for better version control. We have a root branch named "backend-dev", and each of our team members branches from that root branch to implement the functionality that they're assigned to implement. For example, the branch named "backend-dev-workspaces" from "backend-dev" was created for the implementation of the RESTful API for "Workspace" resource. As another example, "backend-dev-userComment" was created for the implementation of the commenting mechanism (for users to be able to comment on each other after collaborating).

Task Specific Branches

backend-dev

The root backend branch

backend-dev-workspaces

The branch for the implementation of the workspace related endpoints

backend-dev-ws-search

The branch for the implementation of the workspace search

backend-dev-userComment

The branch for the implementation of the commenting mechanism (for users to be able to comment on each other after collaborating)

backend-dev-user-search

The branch for the implementation of the user search

backend-dev-upcomingEventSearch

The branch for the implementation of the upcoming event search

backend-dev-upcoming-event

The branch for the implementation of the upcoming events related endpoints

backend-dev-trending-project

The branch for the implementation of the mechanism to find and show the trending projects

backend-dev-search-history

The branch for the implementation of the mechanism to store the search history of users

backend-dev-profile-photo

The branch for the implementation of the mechanism to upload profile photos

backend-dev-file-system

The branch for the implementation of the mechanism to store files and folders

FRONTEND

As Frontend team, we put in effort to use Git version-control system to the best we can. To accomplish this, we created a main branch called 'frontend', which holds the latest ready to deploy state of our subteam's work. We also created other branches for task specific purposes. Each task's assignee was also responsible for creating the relevant branch to contain his/her work. However, we couldn't follow through with this plan and ended up falling prey for one of the most common pitfalls of git usage: We failed to separate tasks adequately and ended up using frontend-dev-profilepage for majority of our work. To prevent this issue in the future we decided that we would give dedicated branches to each one of our developers. Each developer would work on his/her own branch and after a feature is completed he/she would merge it to frontend-dev branch for others to syn to. This approach might be problematic when there are too many features or when a feature needs more than one developer but because our team has only 4 developers and our features are quite atomic we decided this new approach would be more suitable for our case.

Task Specific Branches

frontend-dev-notification:

This branch is for notifications. This branch contains the notification API and related code for appropriate.

frontend-dev-page-routing:

This branch contains the work related to routing and navigation between pages of our website. We keep this branch up to date by adding each recently completed page's url in link form.

frontend-dev-workspace:

This is the branch that contains what we have done for the workspace functionality and file system. This is currently the biggest branch in frontend. Dividing this branch into a few smaller branches might be more easily manageable. However we didn't have any trouble because all the work related to workspace was done by one team member.

frontend-dev:

This branch is our team's main branch. Content of this branch is the latest ready to deploy state of our subteam's work. After merges to this branch is confirmed by the team, this branch is ready to be merged with our product's main branch 'release'.

Developer Specific Branches

frontend-dev-burhan

This branch contains work of our team's member Burhan Akkus. Currently there is HomePage functionality and Issues implemented on this branch. We plan to create new branches for each of our developer's in our next implementation cycle.

frontend-ramazan:

This branch contains the work of our team's member Ramazan Yurt.

ANDROID

As the android team, we continued to use the git version control system. Here I would like to mention again about our two base branches that we believe to be best practices.The basest branch is called android-release. The purpose of this branch is to keep the project that has become deliverable in a way that will not change until the next deliverable. The second most important one is the android-dev which is the branch where we combine our simultaneous developments. According to the tasks given to us, we made the necessary implementation by opening a new branch by taking the android-dev branch as a base, and then at the end of the implementation we merged our task branch with android-dev.

The most difficult point when using the version control system for this milestone was the architecture change. Because the architecture change affects all parts of the code, it would be very difficult to work here by moving to separate branches. So although we know it is not a good use, there have been times when we commit to the same branch but all of these were the pains of migrating to a new architecture. However, we think that we successfully managed this exchange process without any trouble even though we added so many features between two milestones.

Task Specific Branches

android-dev-file:

This is a branch that we've developed a lot on. The main purpose of the operations on the files belonging to the workspace such as uploading and downloading files, which is also mentioned in the requirements, was implemented here. At the same time, the milestone mechanism was implemented in this branch. Also, ui updates and bug fixes related to workspace were also implemented.

android-dev-issue :

The issue mechanism of the workspaces was implemented in this branch. Added features such as adding assignee to issues, removing and commenting.

android-dev-ws :

Operations such as adding/deleting/updating workspaces, adding/deleting skills are implemented in this branch. In general, most of the operations we can do on the workspace are implemented in this branch.

android-dev-newarch:

This is our branch where the most important changes were implemented before the milestone came. We experienced an architecture change in this branch. We switched from the MVP design, which we later learned to be an old architecture, to the MVVM structure. We also implemented the pagination, which we think is a very important feature, in this branch.

android-dev-newarch-imp:

Notification mechanism implemented on this branch. Also some search and ui bug fixes were handled. Also, improvements made on the general architecture, where imp implies the improvements.

android-release-lottie:

We used this branch to make our app a Christmas edition before the presentation.

android-dev-newarch-ue:

Upcoming events UI updated and pagination added. Also get upcoming events and get trending projects tied to the backend. Lastly some bug fixes were made.

android-dev-newarch-ws:

Fixes related to workspace have been made. Also add workspace and edit workspace added. Also some UI updates were handled.

android-dev-newarch-others-pp:

Other profile UI logic changed and follow mechanism added.

android-dev-newarch-imp-req:

Follow request accept and reject handled. Also some UI updates were made.

android-dev-newarch-research:

Edit research implemented. Also some bug fixes were made.

android-dev-newarch-imp-workspace-rw:

Workspace recyclerview added. Also register and edit profile updated according to new endpoints.

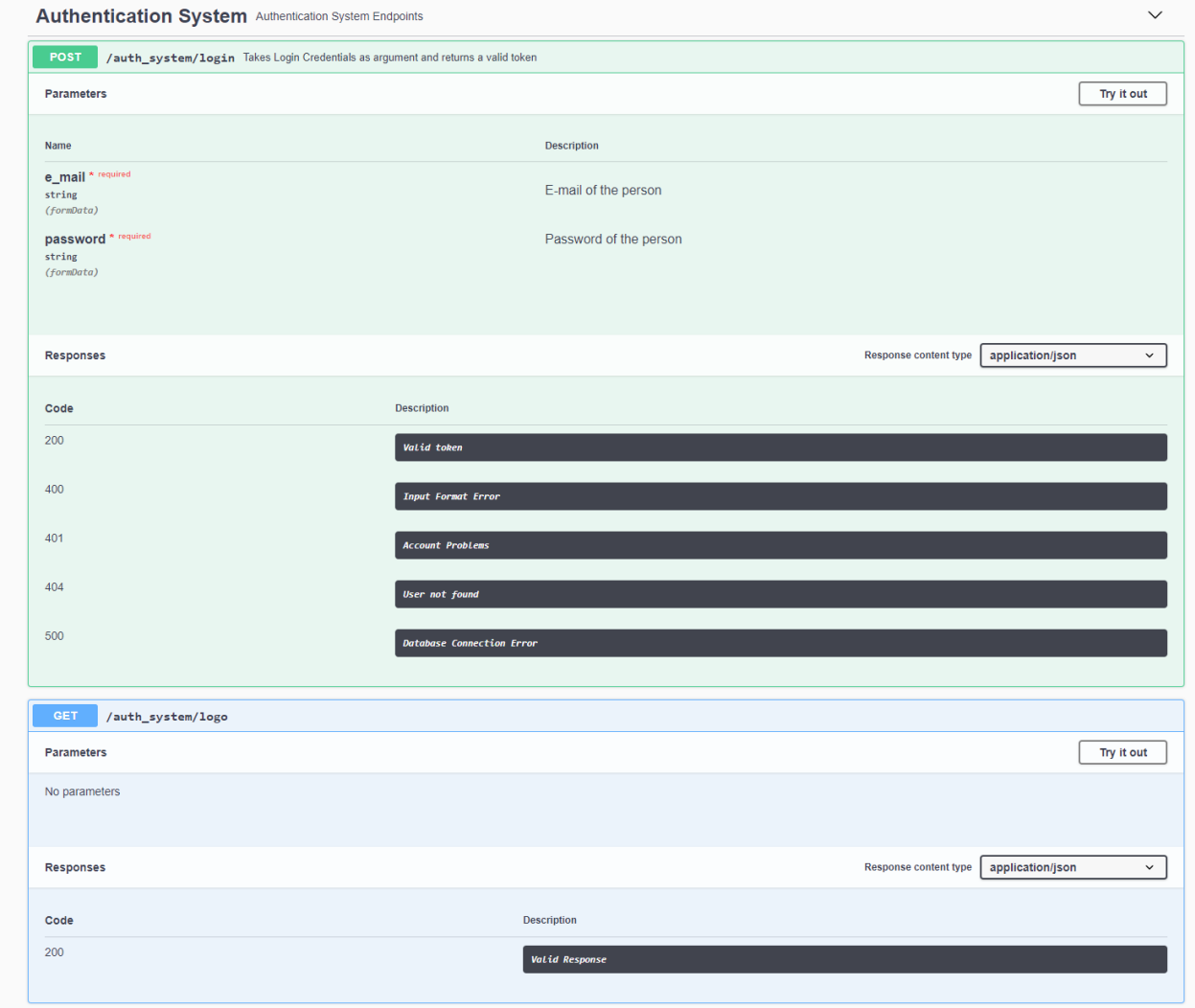
CHALLENGES DURING DEPLOYMENT, DOCKERIZING AND CI/CD PROCESS

Until Customer Milestone 2 we have encountered some problems in our deployment process. Our production server gave out of memory errors periodically. Initially, we do not try to find the source of the problem. Because the frequency of the errors was not so high. However, as we increased the size of code that runs on the server, the frequency of out of memory error increased. Then we searched for the source of the problem. As our first milestone report states, we used docker-compose to deploy database, backend, and frontend on the same machine. After some searching, we see that the source of the problem is this. After the database, and backend were deployed, the frontend was starting to be deployed, however, the node.js script (start.js) that started the app requires more memory than our machine had while it was running. In other words, we deployed everything in one machine, but its memory is not enough to run all of them.

After we realized that deployment of React App causes the out of memory error, we decided to separate the FlaskServer/MySQLDatabase and React App. Then, we created a new Amazon EC2 Instance located in Frankfurt. We set its config files. Then, we installed Docker and Git (because of the dependency issues) to that instance. Then, we wrote a script containing Docker commands in order to automate the deployment process. After doing all these steps, we deployed our React App.

In the case of our CI/CD process, we do not have any update until the Customer Milestone 1. As we stated in its report, we use Travis-CI for our CI/CD process.

API DOCUMENTATION



Authentication System

Authentication System Endpoints

POST `/auth_system/login` Takes Login Credentials as argument and returns a valid token

Parameters — Try it out

Name	Description
e_mail * required string *(formData)*	E-mail of the person
password * required string *(formData)*	Password of the person

Responses — Response content type: application/json

Code	Description
200	*Valid token*
400	*Input Format Error*
401	*Account Problems*
404	*User not found*
500	*Database Connection Error*

GET `/auth_system/logo`

Parameters — Try it out

No parameters

Responses — Response content type: application/json

Code	Description
200	*Valid Response*

GET `/auth_system/profile_photo`

Parameters — Try it out

Name	Description
user_id * required string *(query)*	ID of the User

Responses — Response content type: application/json

Code	Description
200	*Valid Response*
400	*Input Format Error*
404	*Profile Photo is not Found*
500	*Database Connection Error*

POST `/auth_system/reset_password` Changes password of the user

Parameters — Try it out

Name	Description
new_password * required string *(formData)*	Enter new password
new_password_repeat * required string *(formData)*	Enter new password(Again)
auth_token * required string *(header)*	Token that sent via email as a URL link

Responses — Response content type: application/json

Code	Description
200	*Password Successfully Changed*
400	*Passwords are not matched*
401	*Authorization Error*
500	*Database Connection/E-mail Server Error*

GET `/auth_system/reset_password` Sends a reset password email to the user

Parameters Try it out

Name	Description
e_mail * required string *(query)*	E-mail of the person

Responses Response content type: application/json

Code	Description
200	*Valid e-mail* Example Value | Model `{ "mgs": "string" }`
400	*Input Format Error*
401	*Account Problems*
404	*E-mail not found*
500	*Database Connection/E-mail Server Error*

GET `/auth_system/self`

Parameters Try it out

Name	Description
auth_token * required string *(header)*	Authentication token

Responses Response content type: application/json

Code	Description
200	*User has been found.* Example Value | Model `{ "id": 0, "name": "string", "surname": "string", "profile_photo": "string", "e_mail": "string", "job_id": 0 }`
404	*The user is not found.*
500	*The server is not connected to the database.*

POST `/auth_system/skills` Adds a new skill to user's skills with name

Parameters Try it out

Name	Description
skill * required string *(formData)*	Skill name
auth_token * required string *(header)*	Authentication token

Responses Response content type: application/json

Code	Description
200	*Skill is Successfully Added*
400	*Input Format Error*
404	*User is not found*
500	*Database Connection*

GET `/auth_system/skills` Returns a list of user's skills with id and name

Parameters Try it out

Name	Description
user_id * required integer *(query)*	ID of the requested user.
auth_token * required string *(header)*	Authentication token

Responses Response content type: application/json

Code	Description
200	*Skills are Successfully Returned*
206	*Partial Content*
400	*Input Format Error*
404	*Skills are empty*
500	*Database Connection*

DELETE `/auth_system/skills` Deletes the skill from user's skills with name

Parameters — Try it out

Name	Description
skill * required string *(formData)*	Skill name
auth_token * required string *(header)*	Authentication token

Responses — Response content type: application/json

Code	Description
200	*Skill is Successfully Deleted*
400	*Input Format Error*
404	*Skill or UserSkill is not Found*
500	*Database Connection*

POST `/auth_system/user`

Parameters — Try it out

Name	Description
e_mail * required string *(formData)*	E-mail address of the new user
password * required string *(formData)*	Password of the new user
name * required string *(formData)*	Name of the new user
surname * required string *(formData)*	Surname of the new user
job * required string *(formData)*	Job of the new user
google_scholar_name string *(formData)*	URL of the Google Scholar page of the user
researchgate_name string *(formData)*	URL of the ResearchGate page of the user
institution string *(formData)*	Institution of the user

Responses — Response content type: application/json

Code	Description
201	*User has been successfully created.*
206	*Partial Content*
400	*Missing data fields or invalid data.*
409	*User with the given e-mail address already exists.*
500	*The server is not connected to the database.*
503	*The server could not send the account activation e-mail.*

GET /auth_system/user

Returns the profile information of the requested user

Parameters — Try it out

Name	Description
user_id * required integer *(query)*	ID of the requested user.
auth_token * required string *(header)*	Authentication token

Responses — Response content type: application/json

Code	Description
200	*User has been found.* Example Value | Model `{ "id": 0, "name": "string", "surname": "string", "profile_photo": "string", "e_mail": "string", "job_id": 0 }`
206	*Partial Content*
400	*Missing data fields or invalid data.*
404	*The user is not found.*
500	*The server is not connected to the database.*

DELETE /auth_system/user

Parameters — Try it out

Name	Description
password * required string *(formData)*	Password of the user
auth_token * required string *(header)*	Authentication token

Responses — Response content type: application/json

Code	Description
200	*User account has been successfully deleted.*
206	*Partial Content*
400	*Missing data fields or invalid data.*
401	*Wrong password.*
404	*The user is not found.*
500	*The server is not connected to the database.*

PUT /auth_system/user

Parameters

Try it out

Name	Description
name string *(formData)*	Name of the user
surname string *(formData)*	Surname of the user
e_mail string *(formData)*	E-mail address of the user
job string *(formData)*	Job of the user
is_valid integer *(formData)*	The flag that shows whether the user account is activated or not. (0: Inactive, 1: Active)
is_private integer *(formData)*	The flag that shows whether the user's profile is public or private. (0: Public, 1: Private)
profile_photo file *(formData)*	Upload File Here
google_scholar_name string *(formData)*	URL of the Google Scholar page of the user
researchgate_name string *(formData)*	URL of the ResearchGate page of the user
institution string *(formData)*	Institution of the user
auth_token * required string *(header)*	Authentication token

Responses

Response content type: application/json

Code	Description
200	*Account information has been successfully updated.*
202	*Server has received the request but there was no information to be updated.*
206	*Partial Content*
400	*Missing data fields or invalid data.*
404	*The user is not found.*
500	*The server is not connected to the database.*

Follow System Follow System Endpoints

POST /follow/comment Create user comment

Parameters

Try it out

Name	Description
commented_user_id * required integer *(formData)*	ID of the commented user
rate * required integer *(formData)*	Rate for the User(Between 1 and 5)
text string *(formData)*	Comment for the User
auth_token * required string *(header)*	Authentication token

Responses

Response content type: application/json

Code	Description
400	*Input Format Error*
401	*Account Problems*
404	*Not found*
500	*Database Connection Error*

GET /follow/comment Get Comments

Parameters

Try it out

Name	Description
commented_user_id * required integer *(query)*	ID of the commented user
page integer *(query)*	Page index that you want(Starts from 0)
per_page integer *(query)*	Number of items in a page
auth_token * required string *(header)*	Authentication token

Responses

Response content type: application/json

Code	Description
200	*Success* Example Value | Model `{ "number_of_pages": 0, "result": [ { "comment_id": 0, "owner_id": 0, "commented_user_id": 0, "timestamp": "2021-01-05T19:14:28.210Z", "rate": 0, "text": "string" } ] }`
400	*Input Format Error*
401	*Account Problems*
404	*Not found*
500	*Database Connection Error*

DELETE `/follow/comment` Deletes an user comment

Parameters — Try it out

Name	Description
comment_id * required integer *(query)*	ID of the commented user
auth_token * required string *(header)*	Authentication token

Responses — Response content type: application/json

Code	Description
400	*Input Format Error*
401	*Account Problems*
404	*Not found*
500	*Database Connection Error*

POST `/follow/follow_requests` Creates FollowRequest record if profile is private, creates Follow record if profile is public

Parameters — Try it out

Name	Description
follower_id * required integer *(formData)*	ID of the follower. Follower is the one who follows someone.
following_id * required integer *(formData)*	ID of the following user. Following user is the one who is followed by someone.
auth_token * required string *(header)*	Authentication Token

Responses — Response content type: application/json

Code	Description
200	*Follow Request is sent successfully*
400	*Input Format Error*
401	*Current user is unauthorized to send Follow Request*
500	*Database Connection Error*

GET `/follow/follow_requests` Returns a list of dictionaries that contains id, name, surname, e_mail, rate and is_private

Parameters — Try it out

Name	Description
following_id * required integer *(query)*	ID of the following user. Following user is the one who is followed by someone.
page integer *(query)*	Page index that you want(Starts from 0)
per_page integer *(query)*	Number of items in a page
auth_token * required string *(header)*	Authentication Token

Responses — Response content type: application/json

Code	Description
200	*Follow Requests List is successfully returned* Example Value | Model `{ "number_of_pages": 0, "follow_requests": [ { "id": 0, "name": "string", "surname": "string", "profile_photo": "string", "e_mail": "string", "job_id": 0 } ] }`
400	*Input Format Error*
401	*Current user is unauthorized to see the Follow Requests*
404	*Follow Requests List is empty*
500	*Database Connection Error*

DELETE `/follow/follow_requests` Takes the follower_id and following_id as inputs and replies the corresponding Follow Request

Parameters — Try it out

Name	Description
follower_id * required integer *(formData)*	ID of the follower. Follower is the one who follows someone.
following_id * required integer *(formData)*	ID of the following user. Following user is the one who is followed by someone.
state * required integer *(formData)*	State is 1 if the reply is accept, state is 2 if the reply is reject.
auth_token * required string *(header)*	Authentication Token

Responses — Response content type: application/json

Code	Description
200	*Follow Request is replied successfully*
400	*Input Format Error*
401	*Current user is unauthorized to reply the Follow Request*
404	*Follow Request not found*
500	*Database Connection Error*

GET /follow/followers

Returns a list of dictionaries with id, name, surname, e_mail, rate and is_private informations

Parameters

Try it out

Name	Description
following_id * required integer *(query)*	ID of the following user. Following user is the one who is followed by someone.
page integer *(query)*	Page index that you want(Starts from 0)
per_page integer *(query)*	Number of items in a page
auth_token * required string *(header)*	Authentication Token

Responses

Response content type: application/json

Code	Description
200	***Followers List is successfully returned*** Example Value | Model `{ "number_of_pages": 0, "followers": [ { "id": 0, "name": "string", "surname": "string", "profile_photo": "string", "e_mail": "string", "job_id": 0 } ] }`
400	***Input Format Error***
404	***Followers List is empty***
500	***Database Connection Error***

GET /follow/followings

Returns a list of dictionaries with id, name, surname, e_mail, rate and is_private informations

Parameters

Try it out

Name	Description
follower_id * required integer *(query)*	ID of the follower. Follower is the one who follows someone.
page integer *(query)*	Page index that you want(Starts from 0)
per_page integer *(query)*	Number of items in a page
auth_token * required string *(header)*	Authentication Token

Responses

Response content type: application/json

Code	Description
200	***Followings List is successfully returned*** Example Value | Model `{ "number_of_pages": 0, "followings": [ { "id": 0, "name": "string", "surname": "string", "profile_photo": "string", "e_mail": "string", "job_id": 0 } ] }`
400	***Input Format Error***
404	***Followings List is empty***
500	***Database Connection Error***

DELETE `/follow/followings` Takes the following_id and authentication token as inputs and deletes corresponding Follow entry

Parameters — Try it out

Name	Description
following_id * required integer *(formData)*	ID of the unfollowed user
auth_token * required string *(header)*	Authentication Token

Responses — Response content type: application/json

Code	Description
200	*User is unfollowed successfully*
400	*Input Format Error*
404	*No follow information found with given authentication token and following_id*
500	*Database Connection Error*

Profile Management

Profile Management Endpoints

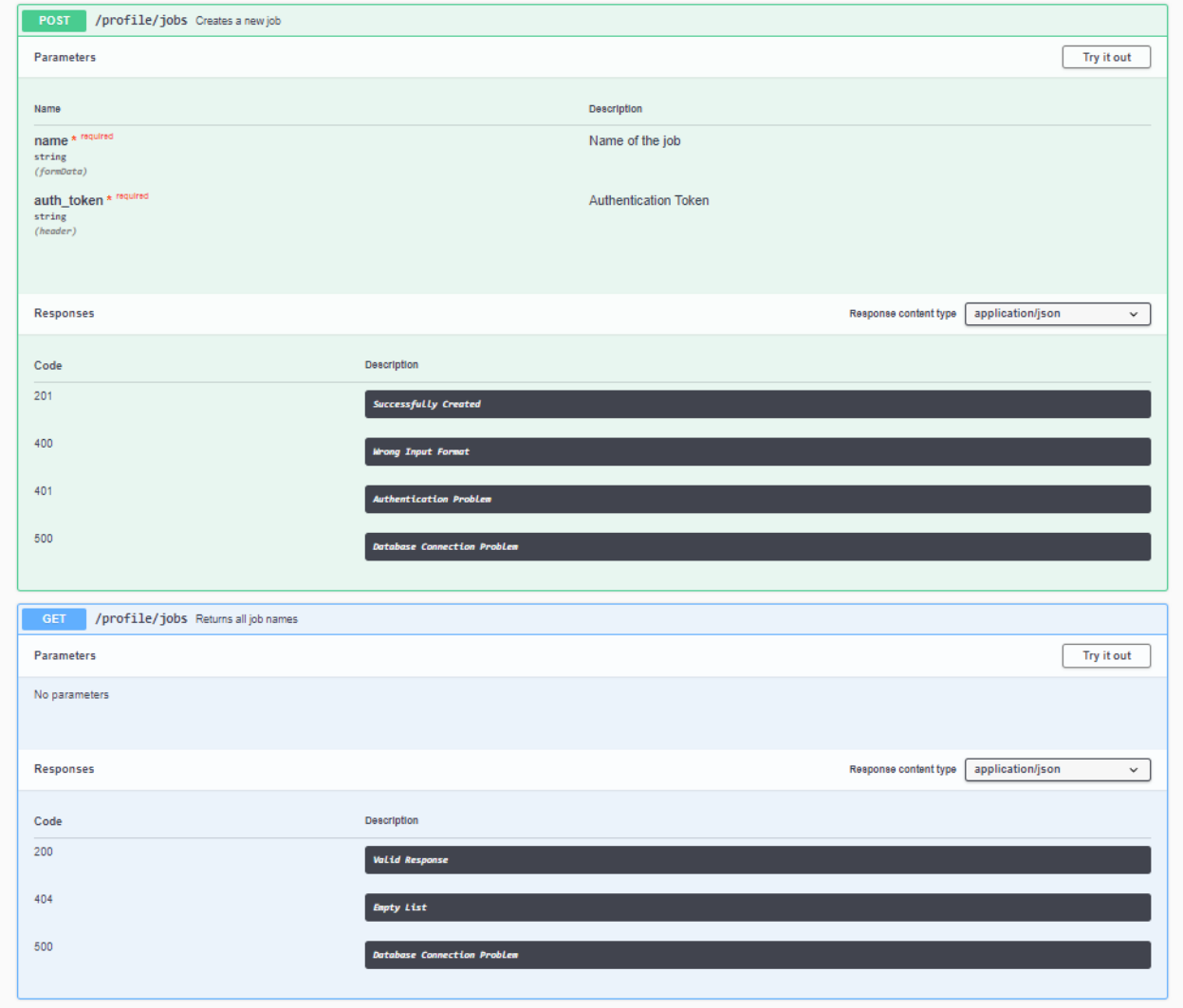
GET `/profile/front_page`

Parameters — Try it out

Name	Description
auth_token * required string *(header)*	Authentication Token

Responses — Response content type: application/json

Code	Description
200	*OK*
500	*The server does not respond.*



POST /profile/jobs Creates a new job

Parameters — Try it out

Name	Description
name * required string (formData)	Name of the job
auth_token * required string (header)	Authentication Token

Responses — Response content type: application/json

Code	Description
201	Successfully Created
400	Wrong Input Format
401	Authentication Problem
500	Database Connection Problem

GET /profile/jobs Returns all job names

Parameters — Try it out

No parameters

Responses — Response content type: application/json

Code	Description
200	Valid Response
404	Empty List
500	Database Connection Problem

DELETE `/profile/jobs` Deletes a job

Parameters Try it out

Name	Description
id * required integer *(formData)*	Job ID
auth_token * required string *(header)*	Authentication Token

Responses Response content type: application/json

Code	Description
200	*Successfully Deleted*
400	*Wrong Input Format*
401	*Authentication Problem*
500	*Database Connection Problem*

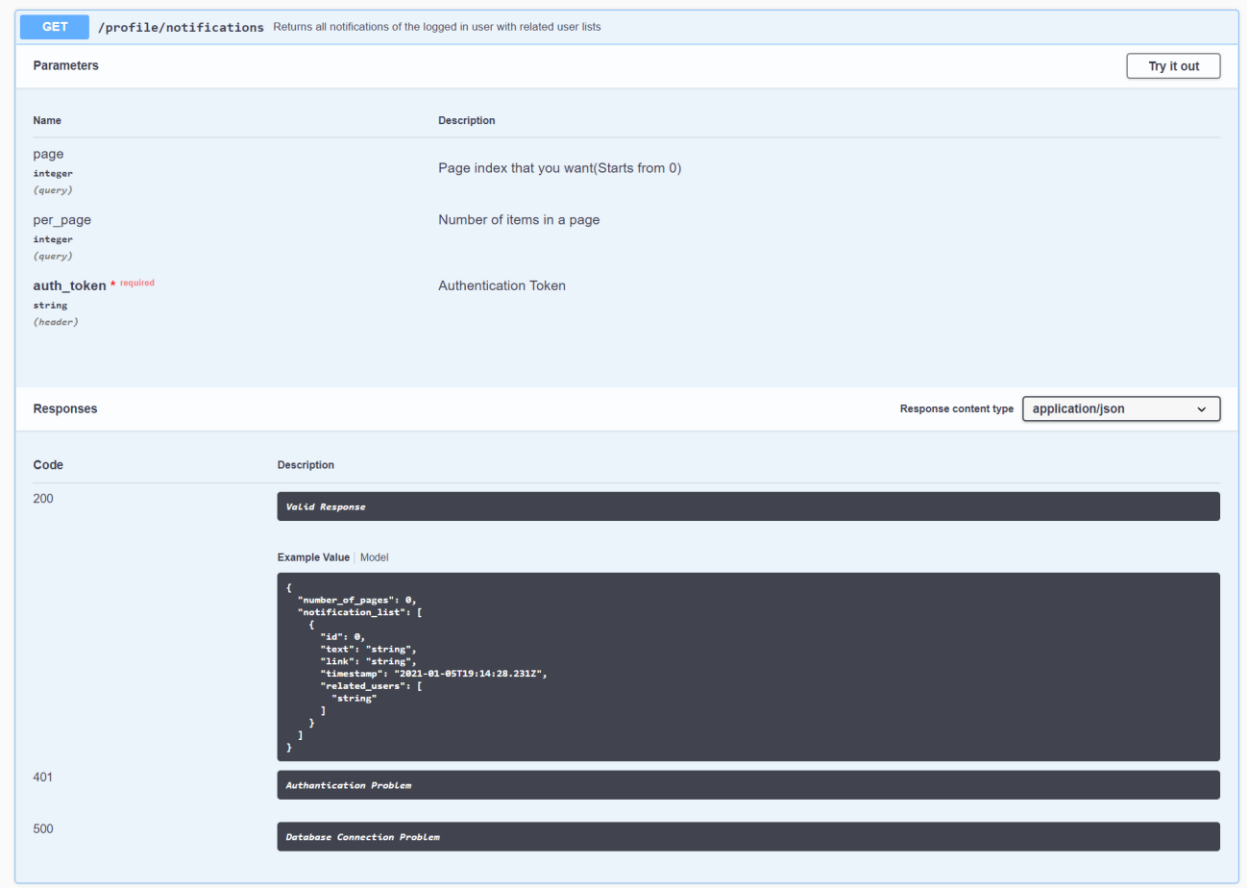
PUT `/profile/jobs` Updates a job

Parameters Try it out

Name	Description
id * required integer *(formData)*	Job ID
name * required string *(formData)*	Name of the job
auth_token * required string *(header)*	Authentication Token

Responses Response content type: application/json

Code	Description
201	*Successfully Updated*
400	*Wrong Input Format*
401	*Authentication Problem*
500	*Database Connection Problem*


GET
/profile/notifications
Returns all notifications of the logged in user with related user lists
Parameters
Try it out
Name
Description
page
integer
(query)
Page index that you want(Starts from 0)
per_page
integer
(query)
Number of items in a page
auth_token * required
string
(header)
Authentication Token
Responses
Response content type
application/json
Code
Description
200
Valid Response
Example Value | Model
{
"number_of_pages": 0,
"notification_list": [
{
"id": 0,
"text": "string",
"link": "string",
"timestamp": "2021-01-05T19:14:28.231Z",
"related_users": [
"string"
]
}
]
}
401
Authantication Problem
500
Database Connection Problem

DELETE `/profile/notifications` Deletes the given notification

Parameters Try it out

Name	Description
notification_id * required integer *(formData)*	ID of the Notification that will be deleted
auth_token * required string *(header)*	Authentication Token

Responses Response content type: application/json

Code	Description
200	*Successfully Deleted*
400	*Wrong Input Format*
401	*Authantication Problem*
404	*Notification Not Found*
500	*Database Connection Problem*

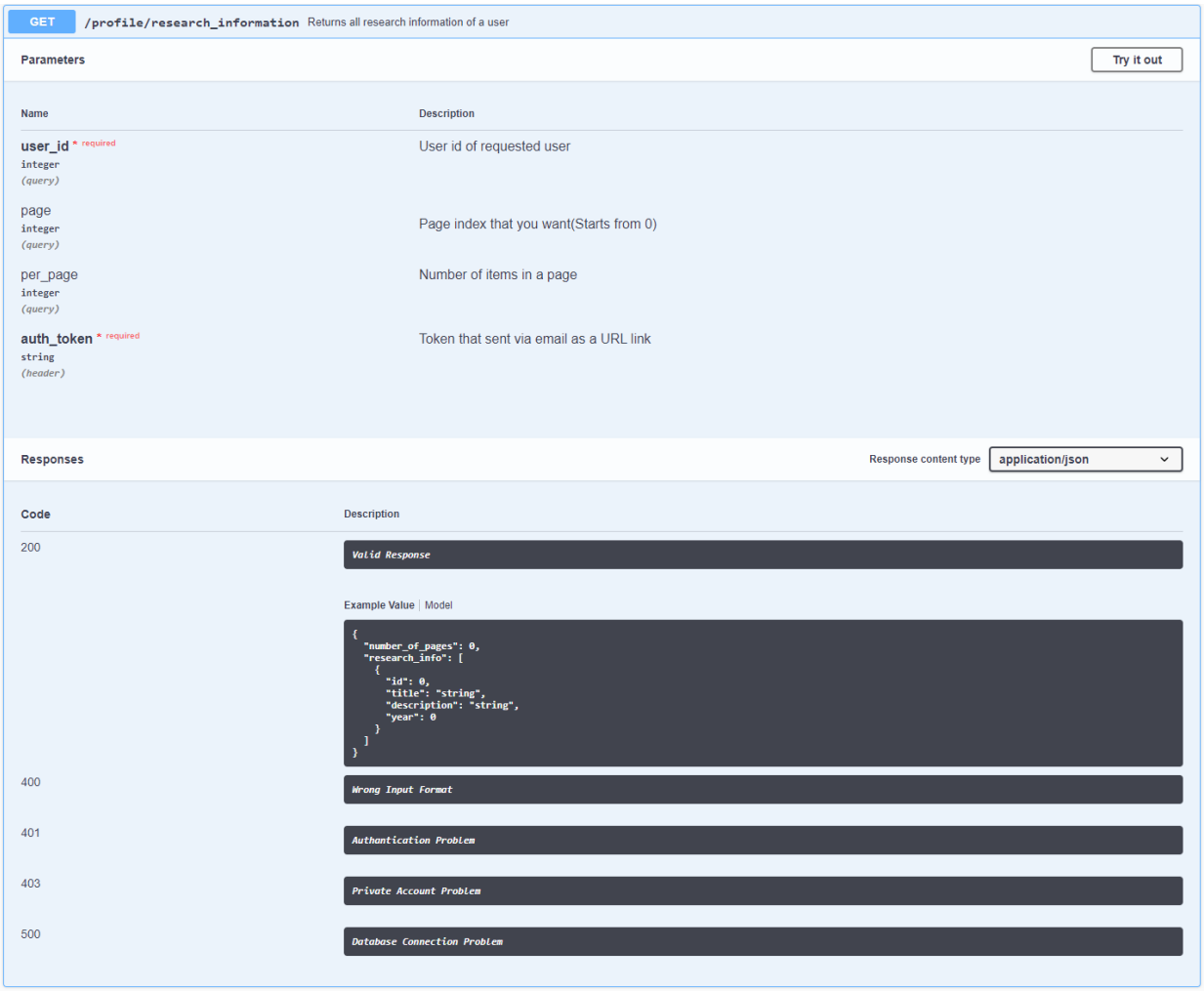
POST `/profile/research_information` Adds new Research Information

Parameters Try it out

Name	Description
research_title * required string *(formData)*	Title of the research
description string *(formData)*	Description of the work *Default value* :
year * required integer *(formData)*	Year of the work
auth_token * required string *(header)*	Authentication Token

Responses Response content type: application/json

Code	Description
201	*Successfully Created*
400	*Wrong Input Format*
401	*Authantication Problem*
500	*Database Connection Problem*


GET
/profile/research_information
Returns all research information of a user
Parameters
Try it out
Name
Description
user_id * required
integer
(query)
User id of requested user
page
integer
(query)
Page index that you want(Starts from 0)
per_page
integer
(query)
Number of items in a page
auth_token * required
string
(header)
Token that sent via email as a URL link
Responses
Response content type
application/json
Code
Description
200
Valid Response
Example Value | Model
{
"number_of_pages": 0,
"research_info": [
{
"id": 0,
"title": "string",
"description": "string",
"year": 0
}
]
}
400
Wrong Input Format
401
Authantication Problem
403
Private Account Problem
500
Database Connection Problem

DELETE `/profile/research_information` Deletes Research Information

Parameters Try it out

Name	Description
research_id * required integer *(formData)*	Research ID of the Research
auth_token * required string *(header)*	Authentication Token

Responses Response content type: application/json

Code	Description
200	*Successfully Deleted*
400	*Wrong Input Format*
401	*Authantication Problem*
500	*Database Connection Problem*

PUT `/profile/research_information` Updates a Research Information

Parameters Try it out

Name	Description
research_id * required integer *(formData)*	Research ID of the Research
research_title string *(formData)*	Title of the research *Default value* :
description string *(formData)*	Description of the work *Default value* :
year integer *(formData)*	Year of the work *Default value* :
auth_token * required string *(header)*	Authentication Token

Responses Response content type: application/json

Code	Description
201	*Successfully Updated*
400	*Wrong Input Format*
401	*Authantication Problem*
500	*Database Connection Problem*

POST `/profile/skills` Creates a skill

Parameters Try it out

Name	Description
name * required string *(formData)*	Name of the skill
auth_token * required string *(header)*	Authentication Token

Responses Response content type: application/json

Code	Description
201	*Successfully Created*
400	*Wrong Input Format*
401	*Authentication Problem*
500	*Database Connection Problem*

GET `/profile/skills` Returns all skill names

Parameters Try it out

No parameters

Responses Response content type: application/json

Code	Description
200	*Valid Response*
404	*Empty List*
500	*Database Connection Problem*

DELETE `/profile/skills` Deletes a skill

Parameters Try it out

Name	Description
id * required integer *(formData)*	Skill ID
auth_token * required string *(header)*	Authentication Token

Responses Response content type: application/json

Code	Description
200	*Successfully Deleted*
400	*Wrong Input Format*
401	*Authentication Problem*
500	*Database Connection Problem*

PUT `/profile/skills` Updates a skill

Parameters Try it out

Name	Description
id * required integer *(formData)*	Skill ID
name * required string *(formData)*	Name of the skill
auth_token * required string *(header)*	Authentication Token

Responses Response content type: application/json

Code	Description
201	*Successfully Updated*
400	*Wrong Input Format*
401	*Authentication Problem*
500	*Database Connection Problem*

Search Engine Search Engine Endpoints

GET `/search_engine/search_history`

Parameters Try it out

Name	Description
search_type * required integer *(query)*	Search Type 0 => User, 1 => Workspace, 2 => Upcoming Event
auth_token * required string *(header)*	Authantication Token(If registered)

Responses Response content type: application/json

Code	Description
200	*Valid Search History* Example Value | Model `{ "search_history": { "query": "string", "number_of_use": 0 } }`
400	*Input Format Error*
401	*Account Problems*
500	*Database Connection Error*

GET `/search_engine/upcoming_events`

Parameters Try it out

Name	Description
search_query * required string *(query)*	Search Query
date_filter_start string *(query)*	Inclusive.
date_filter_end string *(query)*	Exclusive.
deadline_filter_start string *(query)*	Inclusive.
deadline_filter_end string *(query)*	Exclusive.
sorting_criteria integer *(query)*	None => Semantic Rating // 0 => Alphabetical Order(A=>Z) // 1 => Date Order
page integer *(query)*	Page ID(0-Indexed)
per_page integer *(query)*	Number of Records in a Page
auth_token string *(header)*	Authantication Token(If registered)

Responses Response content type application/json

Code	Description
200	*Valid Search Result* Example Value | Model `{ "number_of_pages": 0, "upcoming_events": [ { "id": 0, "acronym": "string", "title": "string", "location": "string", "link": "string", "date": "string", "deadline": "string" } ] }`
400	*Input Format Error*
401	*Account Problems*
500	*Database Connection Error*

GET `/search_engine/user`

Parameters — Try it out

Name	Description
search_query * required string *(query)*	Search Query
job_filter integer *(query)*	Job Filter(Give as ID)
sorting_criteria integer *(query)*	None => Semantic Rating // 0 => Alphabetical Order(A=>Z) // 1 => Alphabetical Order (Z=>A)
page integer *(query)*	Page ID(0-Indexed)
per_page integer *(query)*	Number of Records in a Page
auth_token string *(header)*	Authantication Token(If registered)

Responses — Response content type: application/json

Code	Description
200	*Valid Search Result* Example Value | Model `{ "number_of_pages": 0, "result_list": [ { "id": 0, "name": "string", "surname": "string", "profile_photo": "string", "e_mail": "string", "job_id": 0 } ] }`
400	*Input Format Error*
401	*Account Problems*
500	*Database Connection Error*

GET `/search_engine/workspace`

Parameters Try it out

Name	Description
search_query * required string *(query)*	Search Query
skill_filter string *(query)*	Skill Filter
creator_name string *(query)*	Name of the creator
creator_surname string *(query)*	Surname of the creator
starting_date_start string *(query)*	Inclusive.
starting_date_end string *(query)*	Exclusive.
deadline_start string *(query)*	Inclusive.
deadline_end string *(query)*	Exclusive.
sorting_criteria integer *(query)*	None: Semantic Rating, 0: Ascending Date, 1: Descending Date, 2: Ascending Number of Collaborators Needed, 3: Descending Number of Collaborators Needed, 4: Ascending Alphabetical Order, 5: Descending Alphabetical Order)
page integer *(query)*	Page ID(0-Indexed)
per_page integer *(query)*	Number of Records in a Page
auth_token string *(header)*	Authantication Token(If registered)

Responses Response content type application/json

Code	Description
200	*Valid Search Result* Example Value | Model `{ "workspaces": [ { "id": 0, "title": "string", "description": "string", "state": 0, "creator_id": 0, "contributors": [ { "id": 0, "name": "string", "surname": "string" } ] } ] }`
400	*Input Format Error*
401	*Account Problems*
500	*Database Connection Error*

Workspace System

Workspace System Endpoints

POST `/workspaces` Creates a new workspace

Parameters Try it out

Name	Description
title * required string *(formData)*	Title of the new workspace
description * required string *(formData)*	Description of the new workspace
is_private integer *(formData)*	Privacy status of the new workspace
max_collaborators integer *(formData)*	Maximum number of collaborators of the new workspace
deadline string *(formData)*	Deadline of the new workspace
requirements string *(formData)*	The list of the requirements to be able to join the new workspace
skills string *(formData)*	The list of the skills required to be able to join the new workspace
upcoming_events string *(formData)*	The list of the upcoming events that are related to the new workspace
auth_token * required string *(header)*	Authentication token

Responses Response content type application/json

Code	Description
201	*Workspace has been successfully created.*
400	*Missing data fields or invalid data.*
500	*The server is not connected to the database.*

GET `/workspaces` Returns the requested workspace(s)

Parameters Try it out

Name	Description
workspace_id * required integer *(query)*	ID of the requested workspace
auth_token * required string *(header)*	Authentication token

Responses Response content type: application/json

Code	Description
200	*Workspace(s) found.*
400	*Missing data fields or invalid data.*
401	*The user is not allowed to view this workspace.*
404	*Requested workspace is not found.*
500	*The server is not connected to the database.*

DELETE `/workspaces` Deletes the requested workspace

Parameters Try it out

Name	Description
workspace_id * required integer *(formData)*	ID of the requested workspace
auth_token * required string *(header)*	Authentication token

Responses Response content type: application/json

Code	Description
200	*Workspace has been successfully deleted.*
400	*Missing data fields or invalid data.*
401	*You are not the creator of this workspace, you cannot delete it.*
404	*The workspace is not found.*
500	*The server is not connected to the database.*

PUT `/workspaces` Updates the requested workspace

Parameters Try it out

Name	Description
workspace_id * required integer *(formData)*	ID of the workspace to be updated
title string *(formData)*	Updated title of the new workspace
description string *(formData)*	Updated description of the new workspace
is_private integer *(formData)*	Updated privacy status of the new workspace
max_collaborators integer *(formData)*	Updated maximum number of collaborators of the new workspace
deadline string *(formData)*	Updated deadline of the new workspace
requirements string *(formData)*	Updated list of the requirements to be able to join the new workspace
skills string *(formData)*	Updated list of the skills required to be able to join the new workspace
state integer *(formData)*	Updated state of the workspace
upcoming_events string *(formData)*	The list of the upcoming events that are related to the new workspace
auth_token * required string *(header)*	Authentication token

Responses Response content type application/json

Code	Description
200	*Workspace has been successfully updated.*
202	*Server has received the request but there was no information to be updated.*
400	*Missing data fields or invalid data.*
401	*You are not among the active contributors of this workspace, you cannot modify it.*
404	*The workspace is not found.*
500	*The server is not connected to the database.*

POST `/workspaces/applications` Creates an application for a workspace

Parameters Try it out

Name	Description
workspace_id * required integer *(formData)*	ID of the workspace that the application is sent to
auth_token * required string *(header)*	Authentication token

Responses Response content type: application/json

Code	Description
201	*Application has been successfully created.*
400	*Missing data fields or invalid data.*
404	*The workspace does not exist.*
409	*The user is already an active contributor of this workspace OR there is already an application sent to this workspace. OR the workspace has already reached its maximum number of collaborators OR this workspace is not in 'search for collaborators' state.*
500	*The server is not connected to the database.*

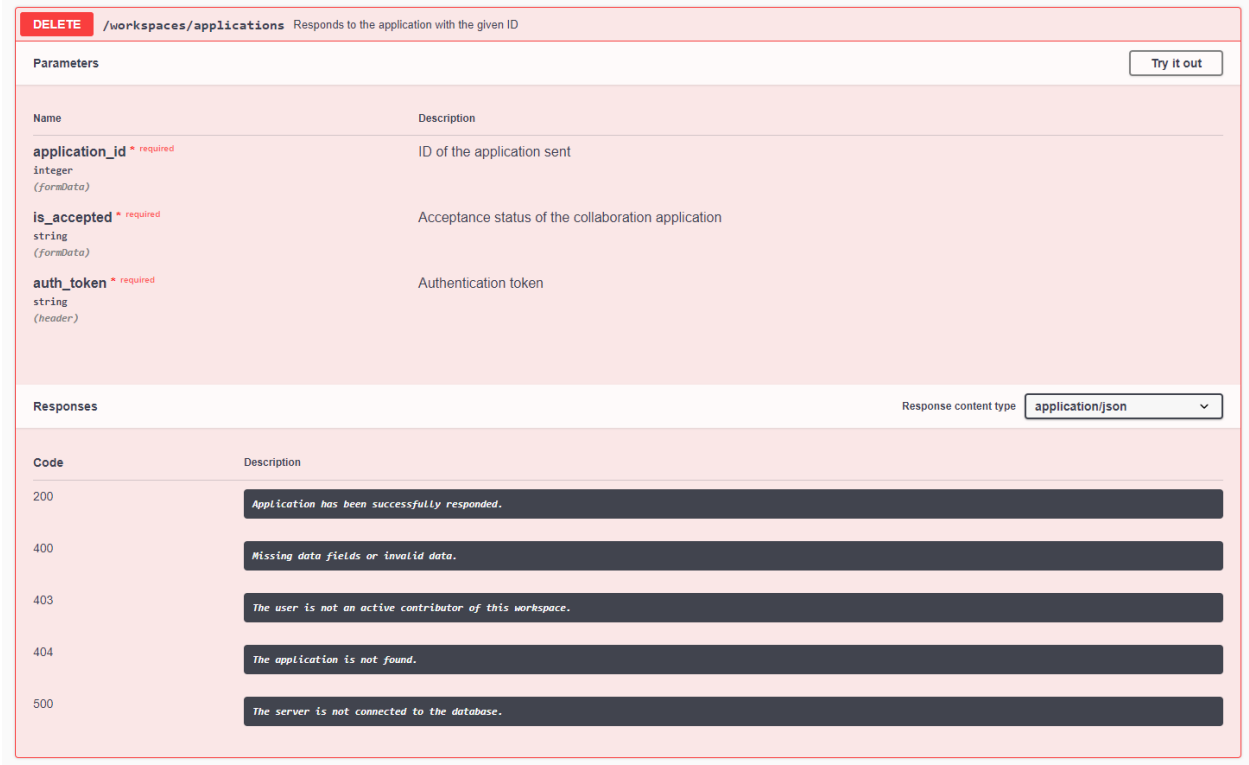
GET `/workspaces/applications` From the database, retrieves the requested pending application(s) sent to the workspace and returns them

Parameters Try it out

Name	Description
application_id integer *(query)*	ID of the application to be retrieved from the database
workspace_id * required integer *(query)*	ID of the workspace that the application(s) is sent to
auth_token * required string *(header)*	Authentication token

Responses Response content type: application/json

Code	Description
200	*Application(s) found.*
400	*Missing data fields or invalid data.*
403	*The user is not an active contributor of this workspace.*
404	*Application(s) not found.*
500	*The server is not connected to the database.*


DELETE
/workspaces/applications
Responds to the application with the given ID
Parameters
Try it out
Name
Description
application_id * required
integer
(formData)
ID of the application sent
is_accepted * required
string
(formData)
Acceptance status of the collaboration application
auth_token * required
string
(header)
Authentication token
Responses
Response content type
application/json
Code
Description
200
Application has been successfully responded.
400
Missing data fields or invalid data.
403
The user is not an active contributor of this workspace.
404
The application is not found.
500
The server is not connected to the database.

POST `/workspaces/invitations` Creates an invitation for a workspace

(i.e. an invitation for a workspace gets sent to the invitee user)

Parameters Try it out

Name	Description
workspace_id * required integer *(formData)*	ID of the workspace that the invitation is sent for
invitee_id * required integer *(formData)*	ID of the user that the invitation is sent to
auth_token * required string *(header)*	Authentication token

Responses Response content type application/json

Code	Description
201	*Invitation has been successfully created.*
400	*Missing data fields or invalid data.*
401	*You are not the creator of this workspace, you cannot invite users to this workspace.*
404	*The workspace or the invitee user does not exist.*
409	*Invitee user is already an active contributor of this workspace OR there is already an invitation sent to the invitee user for this workspace. OR the workspace has already reached its maximum number of collaborators OR this workspace is not in 'search for collaborators' state.*
500	*The server is not connected to the database.*

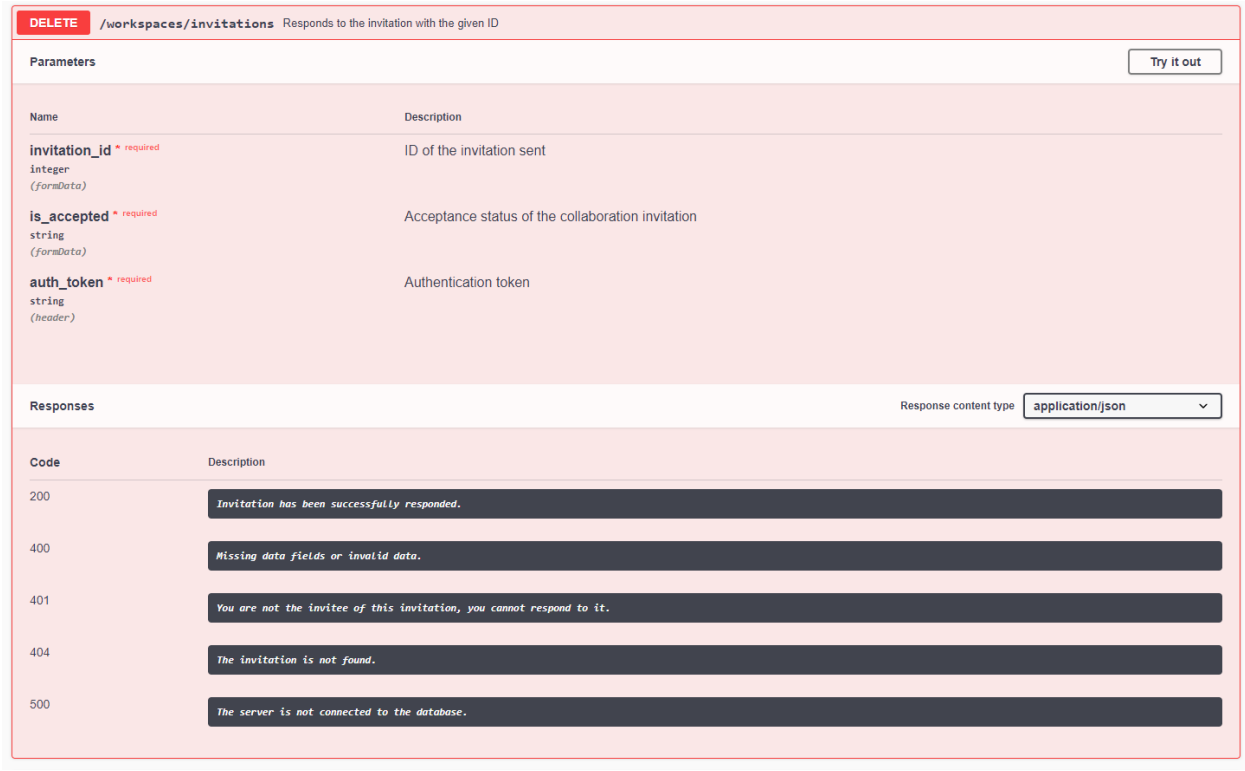
GET `/workspaces/invitations` From the database, retrieves the requested pending invitation(s) sent to the user and returns them

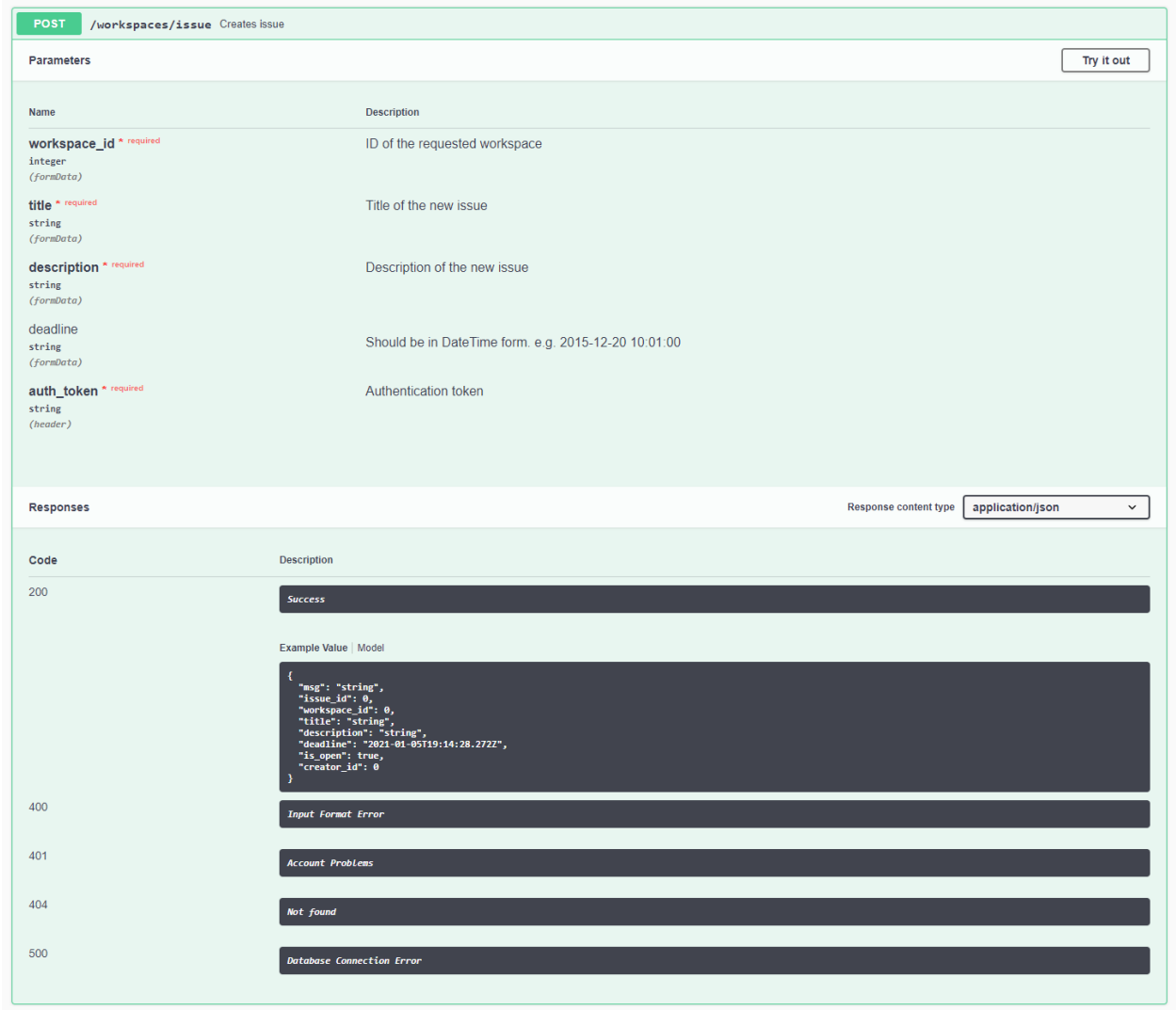
Parameters Try it out

Name	Description
invitation_id integer *(query)*	ID of the invitation to be retrieved from the database
auth_token * required string *(header)*	Authentication token

Responses Response content type application/json

Code	Description
200	*Invitation(s) found.*
400	*Missing data fields or invalid data.*
404	*Invitation(s) not found.*
500	*The server is not connected to the database.*


DELETE
/workspaces/invitations
Responds to the invitation with the given ID
Parameters
Try it out
Name
Description
invitation_id * required
integer
(formData)
ID of the invitation sent
is_accepted * required
string
(formData)
Acceptance status of the collaboration invitation
auth_token * required
string
(header)
Authentication token
Responses
Response content type
application/json
Code
Description
200
Invitation has been successfully responded.
400
Missing data fields or invalid data.
401
You are not the invitee of this invitation, you cannot respond to it.
404
The invitation is not found.
500
The server is not connected to the database.



POST /workspaces/issue Creates issue

Parameters

Try it out

Name	Description
workspace_id * required integer *(formData)*	ID of the requested workspace
title * required string *(formData)*	Title of the new issue
description * required string *(formData)*	Description of the new issue
deadline string *(formData)*	Should be in DateTime form. e.g. 2015-12-20 10:01:00
auth_token * required string *(header)*	Authentication token

Responses

Response content type application/json

Code	Description
200	*Success* Example Value | Model `{ "msg": "string", "issue_id": 0, "workspace_id": 0, "title": "string", "description": "string", "deadline": "2021-01-05T19:14:28.272Z", "is_open": true, "creator_id": 0 }`
400	*Input Format Error*
401	*Account Problems*
404	*Not found*
500	*Database Connection Error*

GET `/workspaces/issue` Get Issues

Parameters Try it out

Name	Description
workspace_id * required integer *(query)*	ID of the requested workspace
page integer *(query)*	Page index that you want(Starts from 0)
per_page integer *(query)*	Number of items in a page
auth_token * required string *(header)*	Authentication token

Responses Response content type application/json

Code	Description
200	*Success* Example Value | Model `{ "number_of_pages": 0, "result": [{ "issue_id": 0, "workspace_id": 0, "title": "string", "description": "string", "deadline": "2021-01-05T19:14:28.274Z", "is_open": true, "creator_id": 0, "creator_name": "string", "creator_surname": "string", "creator_e_mail": "string", "creator_rate": 0, "creator_job_name": "string", "creator_institution": "string", "creator_is_private": true, "creator_photo": "string" }] }`
400	*Input Format Error*
401	*Account Problems*
404	*Not found*
500	*Database Connection Error*

DELETE `/workspaces/issue` Deletes an issue

Parameters Try it out

Name	Description
workspace_id * required integer *(formData)*	ID of the requested workspace
issue_id * required integer *(formData)*	ID of the issue
auth_token * required string *(header)*	Authentication token

Responses Response content type application/json

Code	Description
400	*Input Format Error*
401	*Account Problems*
404	*Not found*
500	*Database Connection Error*

PUT `/workspaces/issue` Updates an issue

Parameters Try it out

Name	Description
workspace_id * required integer *(formData)*	ID of the requested workspace
issue_id * required integer *(formData)*	ID of the issue
title string *(formData)*	Title of the updated issue
description string *(formData)*	Description of the updated issue
deadline string *(formData)*	Updated deadline of the new issue
auth_token * required string *(header)*	Authentication token

Responses Response content type application/json

Code	Description
400	*Input Format Error*
401	*Account Problems*
404	*Not found*
500	*Database Connection Error*

POST `/workspaces/issue/assignee` Create issue assignee record

Parameters Try it out

Name	Description
workspace_id * required integer *(formData)*	ID of the requested workspace
issue_id * required integer *(formData)*	ID of the issue
assignee_id * required integer *(formData)*	ID of the assignee
auth_token * required string *(header)*	Authentication token

Responses Response content type application/json

Code	Description
400	*Input Format Error*
401	*Account Problems*
404	*Not found*
500	*Database Connection Error*

GET `/workspaces/issue/assignee` Get Issue Assignees

Parameters Try it out

Name	Description
workspace_id * required integer *(query)*	ID of the requested workspace
issue_id * required integer *(query)*	ID of the issue
page integer *(query)*	Page index that you want(Starts from 0)
per_page integer *(query)*	Number of items in a page
auth_token * required string *(header)*	Authentication token

Responses Response content type application/json

Code	Description
200	*Success* Example Value | Model `{ "number_of_pages": 0, "result": [ { "comment_id": 0, "comment": "string", "owner_id": 0, "owner_name": "string", "owner_surname": "string", "owner_e_mail": "string", "owner_rate": 0, "owner_photo": "string" } ] }`
400	*Input Format Error*
401	*Account Problems*
404	*Not found*
500	*Database Connection Error*

DELETE /workspaces/issue/assignee

Deletes an issue assignee

Parameters Try it out

Name	Description
workspace_id * required integer *(formData)*	ID of the requested workspace
issue_id * required integer *(formData)*	ID of the issue
assignee_id * required integer *(formData)*	ID of the assignee
auth_token * required string *(header)*	Authentication token

Responses Response content type application/json

Code	Description
400	*Input Format Error*
401	*Account Problems*
404	*Not found*
500	*Database Connection Error*

POST /workspaces/issue/comment

Create issue comment

Parameters Try it out

Name	Description
workspace_id * required integer *(formData)*	ID of the requested workspace
issue_id * required integer *(formData)*	ID of the issue
comment * required string *(formData)*	Comment
auth_token * required string *(header)*	Authentication token

Responses Response content type application/json

Code	Description
400	*Input Format Error*
401	*Account Problems*
404	*Not found*
500	*Database Connection Error*

GET `/workspaces/issue/comment` Get Issue Comments

Parameters Try it out

Name	Description
workspace_id * required integer *(query)*	ID of the requested workspace
issue_id * required integer *(query)*	ID of the issue
page integer *(query)*	Page index that you want(Starts from 0)
per_page integer *(query)*	Number of items in a page
auth_token * required string *(header)*	Authentication token

Responses Response content type application/json

Code	Description
200	*Success* Example Value | Model `{ "number_of_pages": 0, "result": [ { "comment_id": 0, "comment": "string", "owner_id": 0, "owner_name": "string", "owner_surname": "string", "owner_e_mail": "string", "owner_rate": 0, "owner_photo": "string" } ] }`
400	*Input Format Error*
401	*Account Problems*
404	*Not found*
500	*Database Connection Error*

DELETE `/workspaces/issue/comment` Deletes an issue comment

Parameters Try it out

Name	Description
workspace_id * required integer *(formData)*	ID of the requested workspace
issue_id * required integer *(formData)*	ID of the issue
comment_id * required integer *(formData)*	ID of the comment
auth_token * required string *(header)*	Authentication token

Responses Response content type application/json

Code	Description
400	*Input Format Error*
401	*Account Problems*
404	*Not found*
500	*Database Connection Error*

POST `/workspaces/milestone` Creates milestone

Parameters Try it out

Name	Description
workspace_id * required integer *(formData)*	ID of the requested workspace
title * required string *(formData)*	Title of the new milestone
description * required string *(formData)*	Description of the new milestone
deadline * required string *(formData)*	Should be in DateTime form. e.g. 2015-12-20 10:01:00
auth_token * required string *(header)*	Authentication token

Responses Response content type application/json

Code	Description
400	*Input Format Error*
401	*Account Problems*
404	*Not found*
500	*Database Connection Error*

GET `/workspaces/milestone` Get Milestones

Parameters Try it out

Name	Description
workspace_id * required integer *(query)*	ID of the requested workspace
page integer *(query)*	Page index that you want(Starts from 0)
per_page integer *(query)*	Number of items in a page
auth_token * required string *(header)*	Authentication token

Responses Response content type application/json

Code	Description
200	*Success* Example Value | Model `{ "number_of_pages": 0, "result": [{ "milestone_id": 0, "workspace_id": 0, "title": "string", "description": "string", "deadline": "2021-01-05T19:14:28.288Z", "creator_id": 0, "creator_name": "string", "creator_surname": "string", "creator_e_mail": "string", "creator_rate": 0, "creator_job_name": "string", "creator_institution": "string", "creator_is_private": true }] }`
400	*Input Format Error*
401	*Account Problems*
404	*Not found*
500	*Database Connection Error*

DELETE `/workspaces/milestone` Deletes a milestone

Parameters Try it out

Name	Description
workspace_id * required integer *(formData)*	ID of the requested workspace
milestone_id * required integer *(formData)*	ID of the milestone
auth_token * required string *(header)*	Authentication token

Responses Response content type application/json

Code	Description
400	*Input Format Error*
401	*Account Problems*
404	*Not found*
500	*Database Connection Error*

PUT `/workspaces/milestone` Updates a milestone

Parameters Try it out

Name	Description
workspace_id * required integer *(formData)*	ID of the requested workspace
milestone_id * required integer *(formData)*	ID of the issue
title string *(formData)*	Title of the updated milestone
description string *(formData)*	Description of the updated milestone
deadline string *(formData)*	Updated deadline of the milestone
auth_token * required string *(header)*	Authentication token

Responses Response content type application/json

Code	Description
400	*Input Format Error*
401	*Account Problems*
404	*Not found*
500	*Database Connection Error*

DELETE `/workspaces/quit` Use it to quit from a workspace

Parameters Try it out

Name	Description
workspace_id * required integer *(formData)*	ID of the Workspace
auth_token * required string *(header)*	Authentication token

Responses Response content type: application/json

Code	Description
201	*You successfully quited from workspace*
400	*Missing or invalid data*
401	*Login Required*

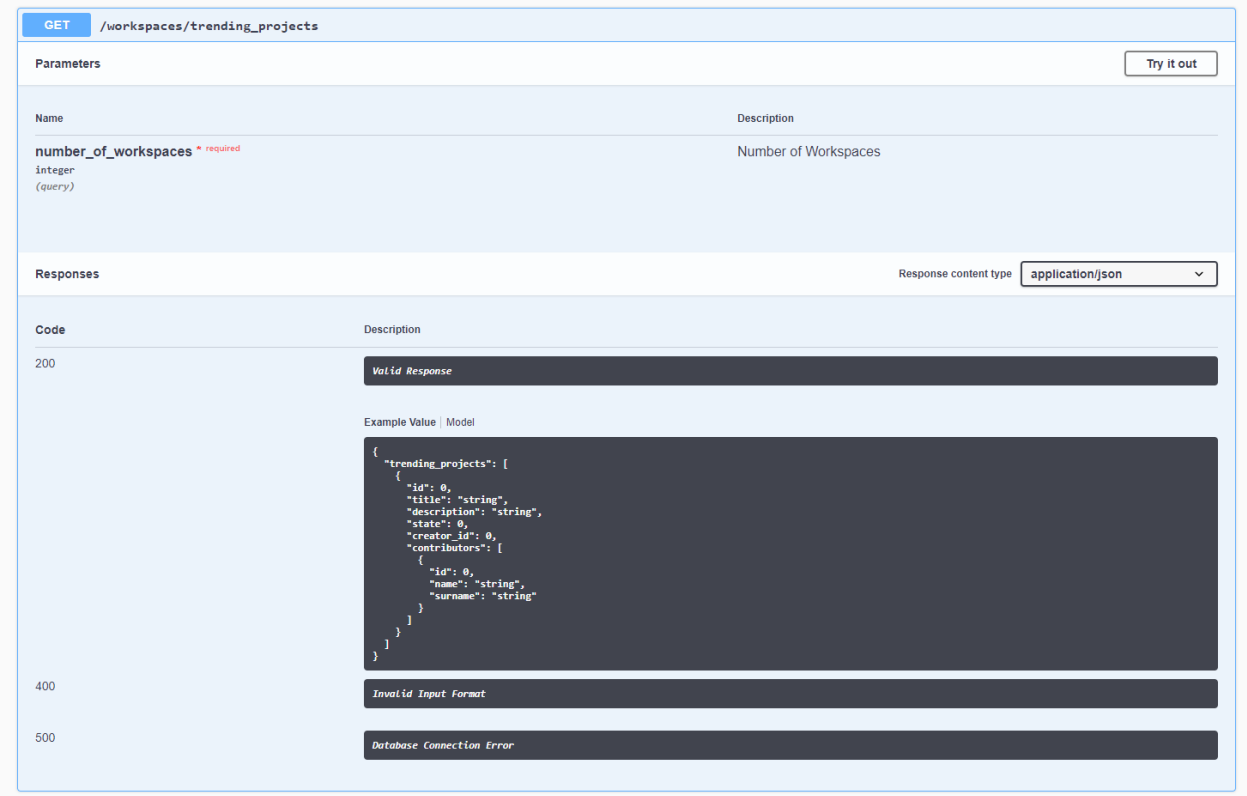
GET `/workspaces/self` Returns all workspaces of the user(Public and Private)

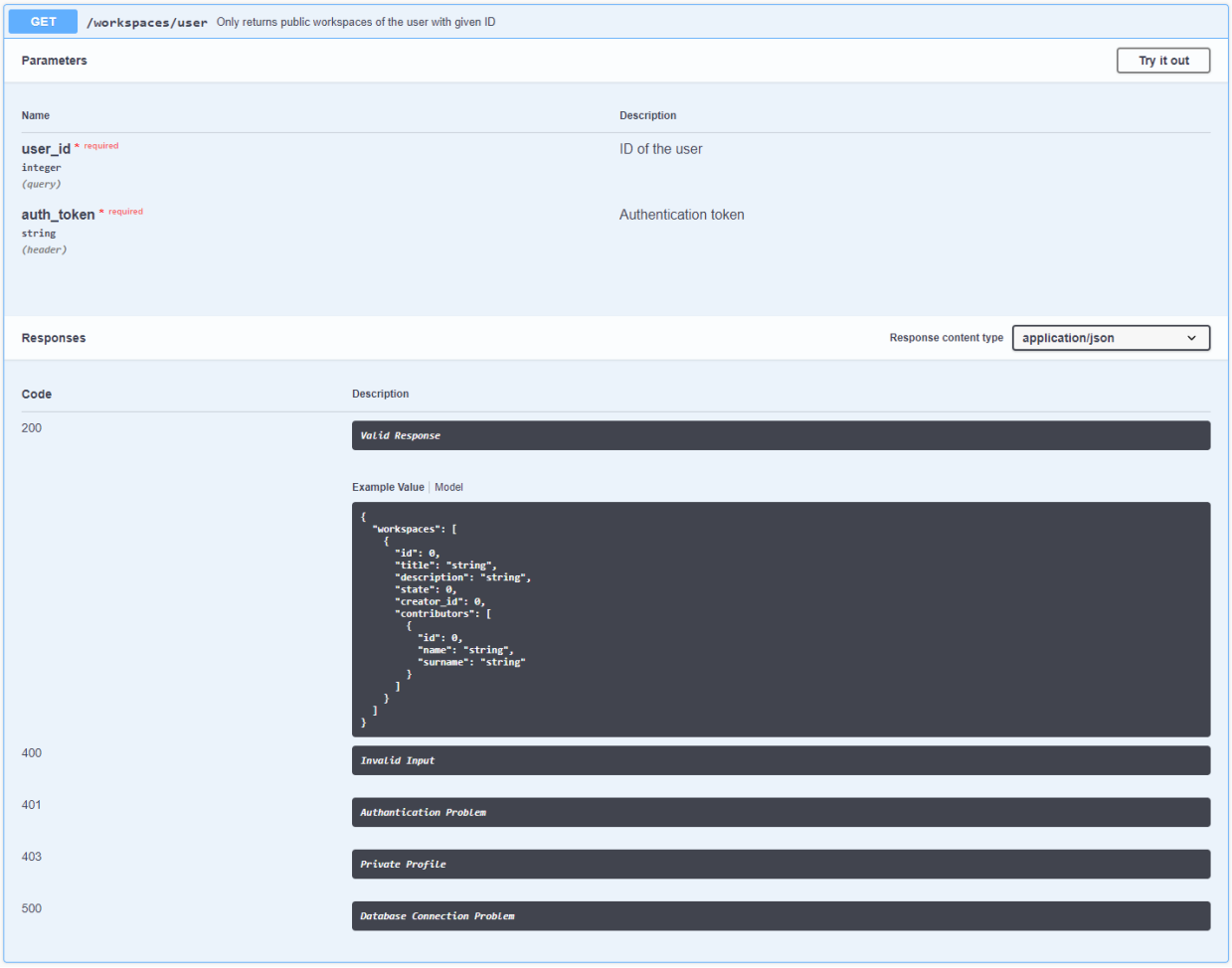
Parameters Try it out

Name	Description
auth_token * required string *(header)*	Authentication token

Responses Response content type: application/json

Code	Description
200	*Valid Response* Example Value | Model `{ "workspaces": [ { "id": 0, "title": "string", "description": "string", "state": 0, "creator_id": 0, "contributors": [ { "id": 0, "name": "string", "surname": "string" } ] } ] }`
401	*Authantication Problem*
500	*Database Connection Problem*


GET
/workspaces/trending_projects
Parameters
Try it out
Name
Description
number_of_workspaces * required
integer
(query)
Number of Workspaces
Responses
Response content type
application/json
Code
Description
200
Valid Response
Example Value | Model
{
"trending_projects": [
{
"id": 0,
"title": "string",
"description": "string",
"state": 0,
"creator_id": 0,
"contributors": [
{
"id": 0,
"name": "string",
"surname": "string"
}
]
}
]
}
400
Invalid Input Format
500
Database Connection Error


GET
/workspaces/user
Only returns public workspaces of the user with given ID
Parameters
Try it out
Name
Description
user_id * required
integer
(query)
ID of the user
auth_token * required
string
(header)
Authentication token
Responses
Response content type
application/json
Code
Description
200
Valid Response
Example Value | Model
{
"workspaces": [
{
"id": 0,
"title": "string",
"description": "string",
"state": 0,
"creator_id": 0,
"contributors": [
{
"id": 0,
"name": "string",
"surname": "string"
}
]
}
]
}
400
Invalid Input
401
Authantication Problem
403
Private Profile
500
Database Connection Problem

File System

File System Endpoints

POST `/file_system/file`

Parameters

Try it out

Name	Description
workspace_id * required integer *(formData)*	Workspace ID
path * required string *(formData)*	Path of the File
filename * required string *(formData)*	Name of the File
new_file * required file *(formData)*	Upload File Here
auth_token * required string *(header)*	Authantication Token

Responses

Response content type: application/json

Code	Description
201	*File Successfully Uploaded*
400	*Invalid Input*
401	*Authantication Problem*
403	*Forbidden Path*
404	*User if not found*

GET /file_system/file

Parameters Try it out

Name	Description
workspace_id * required integer *(query)*	Workspace ID
path * required string *(query)*	Path of the File
filename * required string *(query)*	Name of the File
auth_token * required string *(header)*	Authantication Token

Responses Response content type application/json

Code	Description
200	*Valid Data*
400	*Invalid Input*
401	*Authantication Problem*
403	*Forbidden Path*
404	*User if not found*

DELETE /file_system/file

Parameters Try it out

Name	Description
workspace_id * required integer *(formData)*	Workspace ID
path * required string *(formData)*	Path of the File
filename * required string *(formData)*	Name of the File
auth_token * required string *(header)*	Authantication Token

Responses Response content type application/json

Code	Description
200	*File Successfully Deleted*
400	*Invalid Input*
401	*Authantication Problem*
403	*Forbidden Path*
404	*User if not found*

PUT `/file_system/file`

Parameters Try it out

Name	Description
workspace_id * required integer *(formData)*	Workspace ID
path * required string *(formData)*	Path of the File
filename * required string *(formData)*	Name of the File
new_file * required file *(formData)*	Upload File Here
auth_token * required string *(header)*	Authantication Token

Responses Response content type application/json

Code	Description
200	*File Name Successfully Changed*
400	*Invalid Input*
401	*Authantication Problem*
403	*Forbidden Path*
404	*User if not found*

POST `/file_system/folder`

Parameters — Try it out

Name	Description
workspace_id * required integer *(formData)*	Workspace ID
path * required string *(formData)*	Path of the Folder
new_folder_name * required string *(formData)*	Name of the new folder
auth_token * required string *(header)*	Authantication Token

Responses — Response content type: application/json

Code	Description
201	*Folder Successfully Created*
400	*Invalid Input*
401	*Authantication Problem*
403	*Forbidden Path*
404	*User if not found*

GET /file_system/folder

Parameters Try it out

Name	Description
workspace_id * required integer *(query)*	Workspace ID
path * required string *(query)*	Path of the Folder
auth_token * required string *(header)*	Authantication Token

Responses Response content type application/json

Code	Description
200	*Valid Folder* Example Value | Model `{ "files": [ "string" ], "folders": [ "string" ], "cwd": "string" }`
400	*Invalid Input*
401	*Authantication Problem*
403	*Forbidden Path*
404	*User if not found*

DELETE /file_system/folder

Parameters Try it out

Name	Description
workspace_id * required integer *(formData)*	Workspace ID
path * required string *(formData)*	Path of the Folder
auth_token * required string *(header)*	Authantication Token

Responses Response content type application/json

Code	Description
200	*Folder Successfully Deleted*
400	*Invalid Input*
401	*Authantication Problem*
403	*Forbidden Path*
404	*User if not found*

PUT `/file_system/folder`

Parameters — Try it out

Name	Description
workspace_id * required integer *(formData)*	Workspace ID
path * required string *(formData)*	Path of the Folder
new_folder_name * required string *(formData)*	Name of the new folder
auth_token * required string *(header)*	Authantication Token

Responses — Response content type: application/json

Code	Description
200	*Folder Name Successfully Changed*
400	*Invalid Input*
401	*Authantication Problem*
403	*Forbidden Path*
404	*User if not found*

Upcoming Events

Upcoming Events Endpoints

GET `/upcoming_events`

Parameters — Try it out

Name	Description
page integer *(query)*	Page ID(0-Indexed)
per_page integer *(query)*	Number of Records in a Page

Responses — Response content type: application/json

Code	Description
200	*Valid Response* Example Value | Model `{ "number_of_pages": 0, "upcoming_events": [ { "id": 0, "acronym": "string", "title": "string", "location": "string", "link": "string", "date": "string", "deadline": "string" } ] }`
500	*Database Connection Problem*

PROJECT PLAN

		Ad	Süre	Balat	Bitir...	Önc...	Kaynak Adlar
1		**Orientation**	**6 günler**	**10.02.2020 17:00**	**18...**		
2		Researching about version control system	6 günler	10.02.2020 17:00	18...		Burak Ömür;Hasan Ramazan Yurt;Halil Umut Özdemir;Öykü Ylmaz...
3		Researching about GitHub repositories	6 günler	10.02.2020 17:00	18...		Ahmet Dadak;Alperen Divrikliolu;Burak Ömür;Erturul Bülbül;Halil...
4		Researching about markdown	6 günler	10.02.2020 17:00	18...		Ahmet Dadak;Alperen Divrikliolu;Burak Ömür;Erturul Bülbül;Halil...
5		First team meeting	1 gün	13.02.2020 17:00	14...		Ahmet Dadak;Alperen Divrikliolu;Burak Ömür;Erturul Bülbül;Halil...
6		Creating a workspace on Slack	3 günler	13.02.2020 17:00	18...		Erturul Bülbül
7		**Documentation**	**5 günler**	**13.02.2020 08:00**	**19...**		
8		Creating the wiki page of the team	3 günler	13.02.2020 08:00	17...		Burak Ömür
9		Creating the readme.md for the repository	4 günler	13.02.2020 08:00	18...		Hasan Ramazan Yurt
10		Preparing communication plan	4 günler	13.02.2020 08:00	18...		Ahmet Dadak
11		Preparing favourite github repositories page	5 günler	13.02.2020 08:00	19...		Meltem Arslan
12		Preparing own personal wiki pages	5 günler	13.02.2020 08:00	19...		Ahmet Dadak;Alperen Divrikliolu;Burak Ömür;Erturul Bülbül;Halil...
13		Preparing version control system research page	5 günler	13.02.2020 08:00	19...		Öykü Ylmaz
14		**Requirements**	**13 günler**	**20.02.2020 08:00**	**09...**		
15		Requirement engineering	5 günler	20.02.2020 08:00	26...		Ahmet Dadak;Alperen Divrikliolu;Burak Ömür;Erturul Bülbül;Halil...
16		Reviewing the requirements before first version	1 gün	28.02.2020 08:00	28...		Halil Umut Özdemir;Meltem Arslan;Burak Ömür;Öykü Ylmaz
17		Adding/altering system requirements	3 günler	28.02.2020 08:00	03...		Meltem Arslan;Burak Ömür;Ahmet Dadak;Kerem Uslular;Hasan Ra...
18		Adding/altering user requirements and non-functional requir...	3 günler	28.02.2020 08:00	03...		Erturul Bülbül;Öykü Ylmaz;Alperen Divrikliolu;Mehmet Temizel;...
19		Customer meeting #1 for specifying requirements	3 günler	05.03.2020 08:00	09...		Halil Umut Özdemir;Kerem Uslular;Ahmet Dadak
20		**Logo and Name**	**4 günler**	**28.02.2020 08:00**	**04...**		
21		Preparing logo and name	4 günler	28.02.2020 08:00	04...		Ahmet Dadak;Alperen Divrikliolu;Burak Ömür;Erturul Bülbül;Halil...
22		**User Scenarios**	**11 günler**	**28.02.2020 08:00**	**13...**		
23		Creating scenario #1	4 günler	10.03.2020 08:00	13...	14	Öykü Ylmaz;Hasan Ramazan Yurt;Halil Umut Özdemir
24		Creating scenario #2	4 günler	28.02.2020 08:00	04...		Burak Ömür;Alperen Divrikliolu;Erturul Bülbül;Kerem Uslular
25		Creating scenario #3	4 günler	28.02.2020 08:00	04...		
26		**Mockups**	**4 günler**	**16.03.2020 08:00**	**19...**	**22**	
27		Creating mockup #1	4 günler	16.03.2020 08:00	19...		Halil Umut Özdemir;Hasan Ramazan Yurt;Öykü Ylmaz
28		Creating mockup #2	4 günler	16.03.2020 08:00	19...		Alperen Divrikliolu;Burak Ömür;Erturul Bülbül;Kerem Uslular
29		Creating mockup #3	4 günler	16.03.2020 08:00	19...		
30		**Diagrams**	**11 günler**	**12.03.2020 08:00**	**26...**	**14**	
31		Preparing Use Case Diagram	4 günler	12.03.2020 08:00	17...		Ahmet Dadak;Burak Ömür;Meltem Arslan;Erturul Bülbül;Alperen ...
32		Preparing Class Diagram	4 günler	12.03.2020 08:00	17...		Halil Umut Özdemir;Öykü Ylmaz;Kerem Uslular;Mehmet Temizel;H...
33		Preparing Sequence Diagrams	3 günler	24.03.2020 08:00	26...	31...	Burak Ömür;Hasan Ramazan Yurt;Halil Umut Özdemir;Öykü Ylmaz...
34		**Planning**	**5 günler**	**13.04.2020 13:00**	**20...**		
35		Until Planning	4 günler	13.04.2020 13:00	17...		Ahmet Dadak;Alperen Divrikliolu

		Ad	Süre	Balat	Bitir...	Önc...	Kaynak Adlar
36		After Planning	4 günler	13.04.2020 13:00	17...		Halil Umut Özdemir;Öykü Ylmaz;Meltem Arslan;Kerem Uslular
37		Merge of Plans	1 gün	17.04.2020 13:00	20...	35...	Burak Ömür;Hasan Ramazan Yurt;Mehmet Temizel;Erturul Bülbül
38		Milestone1	0 günler	04.05.2020 17:00	04...	26...	Ahmet Dadak;Alperen Divrikliolu;Burak Ömür;Erturul Bülbül;Halil...
39		**API Test & Study**	**12 günler?**	**23.04.2020 08:00**	**08...**		
40		API Meeting	1 gün?	23.04.2020 08:00	23...		
41		API Usage	5 günler?	23.04.2020 08:00	29...		
42		Design of New API	8 günler?	29.04.2020 08:00	08...		
43		Milestone2	0 günler	18.05.2020 17:00	18...	26...	Ahmet Dadak;Alperen Divrikliolu;Burak Ömür;Erturul Bülbül;Halil...
44		Arrange teams	1 gün	27.10.2020 08:00	27...		Burak Ömür;Hasan Ramazan Yurt;Halil Umut Özdemir;Öykü Ylmaz...
45		Revise Requirements, Design, Plans	6 günler	27.10.2020 08:00	03...		Ahmet Dadak;Alperen Divrikliolu;Burak Ömür;Erturul Bülbül;Halil...
46		**Group Meetings**	**1,833 günler**	**04.11.2020 08:00**	**05...**		
47		All members	0,833 günler	04.11.2020 08:00	04...		Ahmet Dadak;Alperen Divrikliolu;Burak Ömür;Erturul Bülbül;Halil...
48		Backend	1 gün	04.11.2020 15:39	05...	47	Alperen Divrikliolu;Halil Umut Özdemir;Hilal Demir;Hüseyin Can B...
49		Frontend	0,75 günler	04.11.2020 15:39	05...	47	Ahmet Dadak;Burhan Can Akku;Hasan Ramazan Yurt;Umutcan ...
50		Android	0,75 günler	04.11.2020 15:39	05...	47	Burak Ömür;Erturul Bülbül;Orkan Akisu;Öykü Ylmaz
51		**Pre Implementation**	**4,25 günler**	**05.11.2020 13:39**	**11...**		
52		**Backend**	**4 günler**	**05.11.2020 15:39**	**11...**		
53		Create Test Server	4 günler	05.11.2020 15:39	11...	48	Halil Umut Özdemir;Hüseyin Can Bölükba
54		**Frontend**	**1,5 günler**	**05.11.2020 13:39**	**09...**		
55		Create Initial Design	1,5 günler	05.11.2020 13:39	09...	49	Ahmet Dadak;Hasan Ramazan Yurt;Burhan Can Akku;Umutcan ...
56		**Android**	**0,25 günler**	**05.11.2020 13:39**	**05...**		
57		Create Initial Design	0,25 günler	05.11.2020 13:39	05...	50	Erturul Bülbül;Burak Ömür;Öykü Ylmaz;Orkan Akisu
58		**Test Case**	**5 günler**	**11.11.2020 15:39**	**18...**		
59		Implement Test Cases of Log in-Register and Profile Page	5 günler	11.11.2020 15:39	18...	53...	Ahmet Dadak;Alperen Divrikliolu;Burak Ömür;Erturul Bülbül;Halil...
60		**Log In - Register**	**20 günler**	**10.11.2020 08:00**	**07...**		
61		**Backend**	**20 günler**	**10.11.2020 08:00**	**07...**		
62		Implementation of Register System	4 günler	10.11.2020 08:00	13...		Alperen Divrikliolu
63		Implementation of Log In System	8 günler	16.11.2020 08:00	25...	62	Halil Umut Özdemir
64		Reset-Change Password	8 günler	26.11.2020 08:00	07...	62...	Halil Umut Özdemir
65		**Frontend**	**12 günler**	**10.11.2020 08:00**	**25...**		
66		Register Page	12 günler	10.11.2020 08:00	25...		Ahmet Dadak
67		Log In Page	12 günler	10.11.2020 08:00	25...		Ahmet Dadak
68		Reset-Change Password Page	6 günler	10.11.2020 08:00	17...		Ahmet Dadak
69		**Android**	**12 günler**	**10.11.2020 08:00**	**25...**		
70		Register Page	12 günler	10.11.2020 08:00	25...		Burak Ömür

		Ad	Süre	Balat	Bitir...	Önc...	Kaynak Adlar
71		Log In Page	12 günler	10.11.2020 08:00	25...		Burak Ömür
72		Reset- Change Password Page	6 günler	10.11.2020 08:00	17...		Burak Ömür
73		**Profile Page - User Actions**	**20 günler**	**16.11.2020 08:00**	**11...**		
74		**Backend**	**20 günler**	**16.11.2020 08:00**	**11...**		
75		Account Information System	4 günler	08.12.2020 08:00	11...	62...	Alperen Divrikliolu;Halil Umut Özdemir;Hilal Demir
76		User Action - Following System	8 günler	16.11.2020 08:00	25...		Hüseyin Can Bölükba
77		User Action - Reporting System	3 günler	16.11.2020 08:00	18...		Alperen Divrikliolu;Halil Umut Özdemir;Hilal Demir;Hüseyin Can B...
78		Profile Privacy	4 günler	16.11.2020 08:00	19...		Halil Umut Özdemir
79		Linking Google Scholar-ResearchGate Account	12 günler	16.11.2020 08:00	01...		Halil Umut Özdemir
80		**Frontend**	**12 günler**	**16.11.2020 08:00**	**01...**		
81		User Profile Page	12 günler	16.11.2020 08:00	01...		Hasan Ramazan Yurt
82		Follower and Following Accounts Pages	6 günler	16.11.2020 08:00	23...		Hasan Ramazan Yurt
83		Follow Requests Pages	6 günler	16.11.2020 08:00	23...		Hasan Ramazan Yurt
84		Reporting Page	3 günler	16.11.2020 08:00	18...		Hasan Ramazan Yurt
85		**Android**	**4 günler**	**16.11.2020 08:00**	**19...**		
86		User Profile Page	4 günler	16.11.2020 08:00	19...		Burak Ömür;Erturul Bülbül;Öykü Yılmaz
87		Follower and Following Accounts Pages	3 günler	16.11.2020 08:00	18...		Burak Ömür;Erturul Bülbül;Öykü Yılmaz
88		Follow Requests Pages	1 gün	16.11.2020 08:00	16...		Burak Ömür;Erturul Bülbül;Öykü Yılmaz
89		Reporting Page	1 gün	16.11.2020 08:00	16...		Burak Ömür;Erturul Bülbül;Öykü Yılmaz
90		**Testing**	**1 gün**	**16.11.2020 08:00**	**16...**		
91		Test Log in-Register and Profile Page	1 gün	16.11.2020 08:00	16...	62	Ahmet Dadak;Alperen Divrikliolu;Burak Ömür;Erturul Bülbül;Halil...
92		Internal Milestone 1	0 günler	11.12.2020 17:00	11...	58...	Ahmet Dadak;Alperen Divrikliolu;Burak Ömür;Erturul Bülbül;Halil...
93		Milestone3	0 günler	11.12.2020 17:00	11...	51...	Ahmet Dadak;Alperen Divrikliolu;Burak Ömür;Erturul Bülbül;Halil...
94		**Search Engine**	**9 günler**	**30.11.2020 08:00**	**10...**		
95		**Backend**	**8 günler**	**30.11.2020 08:00**	**09...**		
96		Basic User- Workspace- Event Search (Semantic)	2,667 günler	30.11.2020 08:00	02...		Halil Umut Özdemir;Hilal Demir;Hüseyin Can Bölükba
97		Advanced User- Workspace- Event Search (Semantic)	4 günler	30.11.2020 08:00	03...		Halil Umut Özdemir;Hilal Demir;Hüseyin Can Bölükba
98		Search History	4 günler	04.12.2020 08:00	09...	96...	Halil Umut Özdemir
99		**Frontend**	**9 günler**	**30.11.2020 08:00**	**10...**		
100		Basic User- Workspace- Event Search Section	6 günler	30.11.2020 08:00	07...		Hasan Ramazan Yurt
101		Advanced User- Workspace- Event Search Section	9 günler	30.11.2020 08:00	10...		Hasan Ramazan Yurt
102		Search History Section	6 günler	01.12.2020 08:00	08...		Hasan Ramazan Yurt
103		Search Results Page	3 günler	03.12.2020 08:00	07...		Hasan Ramazan Yurt
104		**Android**	**9 günler**	**30.11.2020 08:00**	**10...**		
105		Basic User- Workspace- Event Search Section	6 günler	30.11.2020 08:00	07...		Burak Ömür

		Ad	Süre	Balat	Bitir...	Önc...	Kaynak Adlar
106		Advanced User- Workspace- Event Search Section	9 günler	30.11.2020 08:00	10...		Burak Ömür
107		Search History Section	3 günler	02.12.2020 08:00	04...		Burak Ömür
108		Search Results Page	6 günler	03.12.2020 08:00	10...		Burak Ömür
109		**Testing**	**0,833 günler**	**07.12.2020 08:00**	**07...**		
110		Implement Test Cases of Search Engine	0,833 günler	07.12.2020 08:00	07...		Ahmet Dadak;Alperen Divrikliolu;Burak Ömür;Burhan Can Akku;...
111		**Test Case**	**4 günler**	**07.12.2020 08:00**	**10...**		
112		Implement Test Cases of Search Engine	4 günler	07.12.2020 08:00	10...		Ahmet Dadak;Alperen Divrikliolu;Burak Ömür;Erturul Bülbül;Halil...
113		**Workspace**	**21 günler**	**03.12.2020 08:00**	**31...**		
114		**Backend**	**20 günler**	**04.12.2020 08:00**	**31...**		
115		Workspace Information System	12 günler	04.12.2020 08:00	21...		Alperen Divrikliolu
116		File Storage System	8 günler	22.12.2020 08:00	31...	115	Halil Umut Özdemir
117		Issue - Milestone System	8 günler	22.12.2020 08:00	31...	115	Hüseyin Can Bölükba
118		Comment-Rate System	8 günler	22.12.2020 08:00	31...	115	Hilal Demir
119		Invitation Mechanism	4 günler	22.12.2020 08:00	25...	115	Alperen Divrikliolu
120		**Frontend**	**12 günler**	**03.12.2020 08:00**	**18...**		
121		Workspace Pages for all States of Workspaces	12 günler	03.12.2020 08:00	18...		Ahmet Dadak
122		Edit File Page	6 günler	03.12.2020 08:00	10...		Ahmet Dadak
123		Issue - Milestone Pages	4 günler	03.12.2020 08:00	08...		Burhan Can Akku;Umutcan Uvut
124		Add Comment Rate Sections to Profile Pages	0 günler	03.12.2020 08:00	03...		Hasan Ramazan Yurt
125		Invitation Section of Workspace	2 günler	03.12.2020 08:00	04...		Ahmet Dadak
126		Add Invitation Section to User Profiles	1 gün	03.12.2020 08:00	03...		Ahmet Dadak;Umutcan Uvut
127		**Android**	**12 günler**	**03.12.2020 08:00**	**18...**		
128		Workspace Pages for all States of Workspaces	12 günler	03.12.2020 08:00	18...		Öykü Ylmaz
129		Edit File Page	3 günler	03.12.2020 08:00	07...		Burak Ömür;Öykü Ylmaz
130		Issue - Milestone Pages	1,5 günler	03.12.2020 08:00	04...		Erturul Bülbül;Öykü Ylmaz
131		Add Comment Rate Sections to Profile Pages	3 günler	03.12.2020 08:00	07...		Burak Ömür
132		Invitation Section of Workspace	1,5 günler	03.12.2020 08:00	04...		Burak Ömür;Öykü Ylmaz
133		Add Invitation Section to User Profiles	1,5 günler	03.12.2020 08:00	04...		Burak Ömür;Öykü Ylmaz
134		**Testing**	**2 günler**	**15.12.2020 08:00**	**16...**		
135		Test Upcoming Events-Workspace	2 günler	15.12.2020 08:00	16...		Ahmet Dadak;Alperen Divrikliolu;Burak Ömür;Erturul Bülbül;Halil...
136		**Upcoming Events**	**16 günler**	**21.12.2020 08:00**	**11...**		
137		**Backend**	**16 günler**	**21.12.2020 08:00**	**11...**		
138		Upcoming Events API	16 günler	21.12.2020 08:00	11...		Halil Umut Özdemir
139		**Frontend**	**6 günler**	**21.12.2020 08:00**	**28...**		
140		Upcoming Event Page	6 günler	21.12.2020 08:00	28...		Umutcan Uvut

		Ad	Süre	Balat	Bitir...	Önc...	Kaynak Adlar
141		Upcoming Events Calendar Widget	6 günler	21.12.2020 08:00	28...		Umutcan Uvut
142		**Android**	**3 günler**	**21.12.2020 08:00**	**23...**		
143		Upcoming Event Page	3 günler	21.12.2020 08:00	23...		Burak Ömür;Öykü Yılmaz
144		Upcoming Events Calendar Widget	3 günler	21.12.2020 08:00	23...		Burak Ömür;Öykü Yılmaz
145		Internal Milestone 2	0 günler	22.12.2020 08:00	22...		
146		**Test Case**	**3 günler**	**24.11.2020 08:00**	**26...**		
147		Implement Test Cases of Upcoming Events-Workspace	3 günler	24.11.2020 08:00	26...		Ahmet Dadak;Alperen Divrikliolu;Burak Ömür;Erturul Bülbül;Halil...
148		Milestone4	0 günler	11.01.2021 17:00	11...	11...	Ahmet Dadak;Alperen Divrikliolu;Burak Ömür;Erturul Bülbül;Halil...
149		**Test Case**	**3,333 günler**	**31.12.2020 08:00**	**05...**		
150		Implement Test Cases of Recommendation System-Home Pag	3,333 günler	31.12.2020 08:00	05...		Ahmet Dadak;Alperen Divrikliolu;Burak Ömür;Burhan Can Akku;...
151		**Recommendation System**	**3 günler**	**05.01.2021 08:00**	**07...**		
152		**Backend**	**3 günler**	**05.01.2021 08:00**	**07...**		
153		Recommendation System	3 günler	05.01.2021 08:00	07...		Halil Umut Özdemir
154		**Frontend**	**3 günler**	**05.01.2021 08:00**	**07...**		
155		Add Recommendation Section to Profile Pages	2 günler	05.01.2021 08:00	06...		Hasan Ramazan Yurt
156		Add Recommendation Section to Workspace Pages	3 günler	05.01.2021 08:00	07...		Ahmet Dadak
157		**Android**	**3 günler**	**05.01.2021 08:00**	**07...**		
158		Add Recommendation Section to Profile Pages	3 günler	05.01.2021 08:00	07...		Burak Ömür;Erturul Bülbül;Öykü Yılmaz
159		Add Recommendation Section to Workspace Pages	2 günler	05.01.2021 08:00	06...		Burak Ömür;Erturul Bülbül;Öykü Yılmaz
160		**Home Page**	**9 günler**	**07.01.2021 08:00**	**19...**		
161		**Backend**	**3 günler**	**07.01.2021 08:00**	**11...**		
162		Email Notification System	2 günler	07.01.2021 08:00	08...		Alperen Divrikliolu
163		Activity Stream System	3 günler	07.01.2021 08:00	11...		Hüseyin Can Bölükba
164		**Frontend**	**9 günler**	**07.01.2021 08:00**	**19...**		
165		Home Page	9 günler	07.01.2021 08:00	19...		Burhan Can Akku
166		Add Upcoming Event Calendar Widget	3 günler	07.01.2021 08:00	11...		Umutcan Uvut
167		**Android**	**9 günler**	**07.01.2021 08:00**	**19...**		
168		Home Page	9 günler	07.01.2021 08:00	19...		Erturul Bülbül
169		Add Upcoming Event Calendar Widget	1 gün	07.01.2021 08:00	07...		Burak Ömür;Erturul Bülbül;Öykü Yılmaz
170		**Testing**	**1 gün**	**07.01.2021 08:00**	**07...**		
171		Test Recommendation System and Home Page	1 gün	07.01.2021 08:00	07...		Ahmet Dadak;Alperen Divrikliolu;Burak Ömür;Erturul Bülbül;Halil...
172		System Testing	6 günler?	10.01.2021 08:00	18...		
173		Internal Milestone 3	0 günler	19.01.2021 17:00	19...	94...	Ahmet Dadak;Alperen Divrikliolu;Burak Ömür;Burhan Can Akku;...
174		Milestone5 (Final Milestone)	0 günler	19.01.2021 17:00	19...	15...	Ahmet Dadak;Alperen Divrikliolu;Burak Ömür;Burhan Can Akku;...

SECNARIOS IN THE PRESENTATION

Until the Customer Milestone 2, according to our project plan we implemented a workspace system which includes a file system, an upcoming events system and a search engine. By using the scenarios of Christophe Daussy and Gwen Stacy we tried to present some of the possible use cases of these functionalities.

User Scenario of Christophe Daussy - Android

Christophe Daussy is a Professor of Computer Science in Chicago State University. He has been mentoring 2 grad students for the last 2 years. The project they are working on is Estimating MCvD Channel Coefficients. Lately one of the students has lost his drive to further continue with the project. Realizing this made him decide to quit the project and start over.

The spirit of Christmas makes him change his profile photo. First, he changes his profile picture to show off his favorite Christmas hat. Then, he finds the skill Time management irrelevant, since he believes everyone must have that skill. Instead he adds the skill "Team Work" and "Task Distribution". Then he goes to his workspace named Estimating MCvD Channel Coefficients. He realizes that the upcoming event related to the workspace is not what it should have been. He changes it to the correct one. He moves on to check his students' work on Issues section. He is shocked when he realizes that Can Bolukbas's issue is overdue. Becoming enraged at this, he decides to take this issue from Can and assigns it to Umut. He goes to Can's profile to leave a comment about his work ethic. He searches for Can on the search engine of Platon, then clicks on his profile. He leaves a comment and a low rate to Can's profile. He is still frustrated, so he also unfollows him. He is still a dedicated researcher. So, he decides to start from scratch. He creates a private workspace named Predicting Earthquakes with DL. He sets its requirements and also its required skills. After creating a workspace, he uploads the first file:readme.txt to inform new collaborators about the project. To add collaborators, he checks his following. He thinks Gwen Stacy is a good match. He sends a collaboration invitation to Gwen Stacy. He logs out from the system.

User Scenario of Gwen Stacy - Web

Gwen Stacy is a graduate Student of Computer Science in Chicago State University. She is interested in AI, Deep Learning. She is looking for a project to work on.

First, she checks her follow requests, accepts and rejects them. Her dream has always been to prepare for a conference. So she searches for an upcoming event. She thinks that it is enough of dreaming, it is time to get back to work. She types "prediction" to the search bar. Since Platon supports semantic search, she gets results similar to the word "prediction" like "estimating" and "forecasting". She finds one workspace interesting and visits the workspace. She quickly scans the contents of this workspace, then she applies to this workspace. After that, she checks her workspace invitations to see if one interests her. She finds one that is too similar to the workspace that she just applied. She gets excited and immediately accepts. She goes to her new workspace. She quickly scans the workspace. She previews readme.txt and sees a typo. She edits it. Then she checks the milestones of the project. She realizes that the workspace does not have any milestones. To show that she is dedicated to the project, she adds a milestone. Then to study it later, she downloads the readme file. At the end she logs out from Platon.

EVALUATION OF TOOLS AND MANAGING PROJECT

BACKEND

We've used Discord as our communication platform, Python as our main programming language, Flask as our web framework, MySQL as our database server, Draw.io as a tool that is used to design database structure, PyCharm and VS Code as our IDE, Swagger as our API documentation tool.

We find Discord useful, because it combines the text channels of Slack and the video and voice communication of Zoom. As a result, we continued to use it as our main communication platform. Every wednesday we gathered as the backend team and discussed the task allocation and decided deadlines together.

We continued to use Python due to its relatively easy syntax and also it is the most widely used programming language in our team. We've also continued with Flask. Compared to the first milestone, in this milestone we were more experienced with Flask. This experience paid-off with lots of functionalities within a limited amount of time.

In the case of MySQL, we continued to use SQLAlchemy as an ORM tool to use our database. We've used the Workbench tool widely, and we did not encounter any problem. We've continued to use Draw.io to design our database structure. We found it useful to design the database of the project before the implementation part. Because it creates an abstraction between the modules that we implement individually. If any member knows the exact content of the database, it will be easy to implement each module individually which makes division of tasks easier. Also we think that, by using Draw.io, we can achieve sufficient designs, so we will continue to use it.

We do not use a specific IDE on development of the backend. Some of our group members used Visual Studio Code, and some of them used PyCharm as their IDE. We decided not to select a specific IDE, because we thought that every member has its own IDE that s/he is familiar with. During this period, we did not meet with any problems. So we decided to use Pycharm and Visual Studio Code.

In the case of Swagger, we used the Flask Restplus module of the flask which creates an automated API documentation for our code. Because it removes the obligation that we had to write API documentation by hand, we found Swagger and Flask Restplus very useful and we will continue to create our API Documentation by Swagger.

Also during the development process, we used git as a version management system. We created a development branch for the backend and for each functionality that we implemented, we created another branch from the development branch. At the end we merged all fully functioning branches to the development branch of backend. After the integration of all branches, we tested the code in the development branch. If it works properly, we merge it to the release branch which is the branch that is being deployed to the server.

We've used Trello in the previous milestone. However, we didn't continue with that. Instead we've used Github Issues. Our usage of Trello was mainly for dividing the tasks and assigning them to a contributor. This functionality of Trello was helpful, but it also exists in Github Issues. In addition to that, Github Issues is in our repository. This means we can add bugs in certain lines, we can relate a task with a Pull Request and see comments on the issue from other contributors. These functionalities were

making Github Issues as a better candidate. Therefore we've decided to stop using Trello and continue with Github Issues.

FRONTEND

Frontend team's main communication tool is WhatsApp. Talks about building the interface, discussions about the code, and meeting arrangements are provided by using this application. On the other hand, Discord is used for team meetings. Meetings are usually held to share tasks and to inform about the given task. The given tasks are also written in the WhatsApp group and an issue is opened for the task on GitHub. The team member with a specific task informs the team and opens a branch for herself/himself. Then, the team member in charge performs the coding process through this branch.

As explained in the previous milestone, the front-end team used React to create the interface and continues to use it. Since React has a structure consisting of components, team members usually create a component for themselves while coding their own parts of the project. Majority of the team members used Visual Studio Code during the implementation process, but there were also some using different IDEs such as WebStorm.

The team member who has completed the task assigned to her/his informs the team by writing the part she has finished to WhatsApp. To review the finished part locally, the finisher and another teammate meet on the frontend channel on Discord. If the finished part is as desired, a merge request is created from that branch to the main frontend branch on GitHub. In this way, two people in the meeting will both examine the code and resolve merge conflicts, if any. The relevant issue is closed after the merge transaction takes place.

After all tasks are completed, a pull request is opened from the main branch to the release branch to see the system online. After the team members in charge review the code and resolve any conflicts, the code is merged into the release branch. The system that is online is examined by other teams. After receiving their feedback, if anything is missing, it is corrected, and the system is finalized.

ANDROID

We have used discord as our main voice chat application and whatsapp as our main text chat application. We have used the Kotlin language on Android Studio. We have been using MVP architecture since beginning. However, we have decided to switch MVVM architecture for mainly testability because MVP is mixing UI elements with logic elements. Therefore, we needed a better solution that is MVVM which separates UI and logic distinctly. We have used the issues section at github repository to track other teams and see issues in android.

We have used a custom "Pagination Listener" to implement pagination for all required places such as researches, followers, followings, comments, issues etc. It has internal variables to control pages such as "retrieval of new pages", "current page number", "isLastPage", and "isLoading" to handle UI and thread syncron faultless.

We have switched to "LiveData" objects to receive responses from backed and ViewModel, which was very promising and working well in most cases. However, sometimes we have faced with some double observation of "LiveData" objects which cause duality in UI. We have solved this problem by using a separate Resource object that has 4 states as "Loading", "Success", "Error", and "Done". After switching this resource object we have handled most of the errors and bugs that we've faced.

We have used “Retrofit2” for sending HTTP requests in android, the reason for this is that Retrofit opens a separate thread and handles most of the work successfully. We only handle the results from retrofit with resource objects that we have mentioned before.

We have used “Glide” to retrieve image objects from the server and show it to the user in a nice manner which is also working in another thread and shows results on the main thread. At this point, there were similar libraries that do this, we have read many articles about their comparison and selected the glide for its usability.

We have used the “Android-FilePicker” library to show users an interface that s/he can use to select documents from its locale storage. We know that it will be deprecated for next android versions, therefore we seek for new libraries.

We have always used git to push and pull the new versions of the project. We have given feedback via testing each branch locally using our own emulators and android devices.

For all the libraries and changes, we have stayed in contact and taught everything to each other via github, discord, whatsapp etc.

CUSTOMER MEETING

LESSONS FROM PRESENTATION

The most important lesson learned from the presentation for our case was even though the preparation was good and well thought, we may encounter problems during the presentation. When this happens, what to do next is staying calm and continuing from that point. It is important to have comprehensive knowledge of the system by everybody in the team for this kind of situation. Also, user scenario planning should be done carefully for displaying all of the implemented features in a right amount of time.

In the presentation, we experienced a bug in our system but we managed it well and figured out the problem quickly. This way, we could continue our presentation and all the features, pages, functionalities and mechanisms are shown to the customer through our web and android applications. We have not encountered any other problems. For the future, we can practice even more for our presentation so that chances of encountering any problems or bugs during presentation will be minimum.

LESSONS FROM THE CUSTOMER

From our interaction with the customer we discovered that there may be some usability problems in our system but not major ones. We clarified the confusion between the project creator and the project applicant due to similar permissions are granted to both of them.

For our application’s direction we decided that it would be good if there is a pop-up kind of thing in the workspace creation page for added skills and requirements after the customer's suggestion.

We could not display all the features we have implemented but we managed to apply the project plan in the given deadlines. Still, most of the important features are shown to the customer.

Generally, our current status seemed to satisfied the customers' needs and maybe more. There was no major problems or failure of fulfilment issued by customers.