Boğaziçi University CMPE451 Group 3 Milestone 1 Report

Table of Contents

- Executive Summary
  - Project Description
  - Project Status
  - Moving Forward
- Deliverables
- Work Done by Each Team Member
- Challenges Met During DevOps
- Requirements
- Design documents
- API documentation
- Project plan
- User scenarios
- Evaluation of Tools
  - Backend Tools
  - Frontend Tools
- Evaluation of Managing
- Assessment of the customer presentation
- The code structure and group process

Executive Summary

Project Description

Paperlayer is an online collaboration platform for academics, in which people can create new projects and write papers, while collaborating with other users and scholars. Moreover, in PaperLayer people can access public projects and keep track of the trends in their academic research area, search for co-authors for their research and follow upcoming events in academia such as conferences or journals. Projects can also be made public/private depending on the user's wishes. Users can invite others to projects in order to collaborate and search for an optimal collaborator for their project. Also users can rate other users, comment on their profiles and send each other messages. To sum up, it's a great platform for collaboration for academics.

Project Status

After Cmpe352, we have lost 3 members and 6 new members have joined us. So, we had an adaptation process but this process was very useful for us to review what we did in Cmpe352. We started the semester by reviewing our requirements and project plan with the contribution of the different perspectives of our new members. As a result of this review and adaptation process, we made necessary changes and we made sure that all members in the team are on the same line.

After the completion of these first steps, we divided into 3 subgroups, namely Backend, Frontend and Android. Firstly, we decided the technologies that we will be using and CI/CD workflow in order to avoid any problems that we could face during the implementation.

So far, we have managed to deploy our first MVP (Minimum Viable Product) for the Milestone 1.

- Our web project dockerized and deployed at http://paperlayer.tech/
- Our backend api documentation can be found at http://ec2co-ecsel-1egsefcb8a7i3-34132836.us-east-2.elb.amazonaws.com/swagger/
- We demonstrated a demo of our Android Application at our representation.

First weeks were challenging since our team was not experienced in web development, we struggled learning new technologies and new design patterns. Also constructing a "team" means great effort: Getting used to each other, welcoming new members, getting used to each other's way of working. Beyond that, we had to work hard in order to define a workflow that will ease our future implementations and deployments.

Within the context of our first Milestone, we managed to keep our wiki up to date, define a convenient workflow and stick to our project plan.

Moving Forward

Until the first Milestone, we have faced many challenges and naturally made some mistakes. We learned from our mistakes and realized that we can do better with some issues.

Firstly, we saw that the Backend team had to make some changes in API regarding requests from Frontend and Android teams. So, we decided to make weekly inter-team meetings in order to pre-define API endpoints' details together. We hope that these meetings will improve our efficiency.

Secondly, we noticed at our pre-milestone meeting that the opinions of the members of other subteams were very useful for progress. So, we decided that we should make inter-team reviews in order to recognize shortcomings earlier and take actions.

In terms of the first milestone, we believe that we are in a good situation but we could achieve more. Lessons learned from our mistakes will hopefully help us move forward faster.

Deliverables

Deliverable	Update Frequency	Status
Wiki Page	Weekly	Complete
Issues/Pull Requests	Daily	In Progress
Requirements	As needed	Complete
Design Documents	As needed	Complete
Project Plan	As needed	Complete
User scenarios	As needed	Complete
API Documentation	As we implement/update new endpoints	In Progress
Frontend Project	Daily	In Progress
Backend Project	Daily	In Progress

Deliverable	Update Frequency	Status
Android Project	Daily	In Progress

1- Wiki Page

We've updated our personal wiki pages, wiki home page and we share our meeting notes occasionally.

2- Issues/Pull Requests

We open issues and pull request for each requested feature implementation, bug fix, documentation. We also attach them with Projects so that we can track our progress on a board.

3- Requirements

We've updated requirements according to the feedback from our new members and customer.

4- Design Documents

We've updated our class diagrams according to the changes in the implementation.

5- Project Plan

We've planned our tasks according to which milestone will include which requirement. It also includes due date and which person is responsible for that task.

6- User scenarios

We've prepared two user scenarios for the milestone 1 presentation. These scenarios show the current functionality of the application.

7- API Documentation

We're generating our API documentation automatically with swagger.

8- Frontend Project

We're still implementing the frontend project. You can see the source files in /front-end directory.

9- Backend Project

We're still implementing the backebd project. You can see the source files in /backend directory.

9- Android Project

We're still implementing the android project. You can see the source files in /android directory.

Work Done By Each Member

Name	Task
Mahir Efe KAYA	[Android]Register page UI [Android] Register Page Functionality [Android] Register's Connection to Backend [Android] Register Error Checking Merging the individual parts of the milestone Searching for the implementation of the dagger and session manager principle in kotlin
Ramazan Koç	[Android] Implementing project main page, project page and login page. Connecting project main page,project page and login page with backend. Preparing the main scenario that used in android presentation. Searched retrofit,dagger,rxjava,materialui.
Ahmet Emir Kocağa	- Created first empty version of the Django Project. - Researched about Static Code Analysis and CI. - Set up gitignore file for backend. - Implemented "Profile" model and created CRUD endpoints. - Implemented "Project" model and created CRUD endpoints. - Created the permission check mechanism for these objects so that only the owner can edit. - Created Backend, Frontend, Andorid and General Projects in GitHub Projects section. - Prepared Milestone 1 presentation. - Reviewed work done by other subteams. - Wrote the Executive Summary of Milestone Report 1.
Yahya Bedirhan Pak	[Android] Implementing profile view and edit pages. Connecting profile view and edit pages with backend. Creating an initial retrofit and moshi configurations for API requests. Adding initial dependency injection structure to the project with dagger2. [Backend] Configuring poetry Creating Dockerfile and docker-compose Creating GitHub Actions for backend continuous integration. Helping Furkan for continuous delivery of the backend and frontend. [Frontend] Helping Barış with the continuous integration of the frontend.
Furkan Cansever	Implemented a script file compatible with Linux and macOS (no longer needed after usingpoetry) Implemented configurations for deployment to Heroku Handled CD process to Amazon ECS for backend and fronted Reviewed work is done by backend team members. Implemented Swagger UI for API documentation.
Buse Gilederelli	Creating issue template Researching and creating a document about MySQL Database Creating Milestone model Creating Tag model Adding DjangoFilterBackend to the project Updating project plan

Name	Task
Barış Başmak	I work in the front-end team. For the front-end project: I created the project. Designed and coded a simple architecture so that pages could be added in an understandable and effortless way. Added the dependencies. Coded the login page and the login functionality. Coded the registration page. I implemented the navigation bar for logged in and guest states. Coded the project creation page. Coded event creation page Before the deployment I implemented "visual design" changes to the homepage. Helped researching about deploying the front-end project.
Ali Furkan Budak	- Implemented 4 endpoints: "register", "auth", "logout", and "reset_password" - Modified the "auth" endpoint so that it returns the logged-in user's ID - Modified the "register" endpoint so that it accepts name fields and creates a new Profile with the name information for the registered user. - Created CRUD endpoints for "Event" objects. - Added an optional "events" field to the "Project" model
Adil Numan Çelik	Created File model Created "retrieve_file" endpoint for getting the file Wrote test cases for File model. Implemented the first version of endpoints for registration, authorization, password reset but they are replaced with Ali Furkan Budak's implementation due to design choices. Tested the endpoints.
Ahmet Mert Tahran	Coding the home page and listing projects and events functionality. Coding the profile page and listing details functionality. Coding the project page and listing details functionality. Coding event page and listing details functionality.
Emirhan Yasin Cetin	Edited the profile page code, so that editing will made user friendly Made the page routing settings Coding the user profile editing page and the functionalities Web security and using chrome for development settings
Yusuf Bayam	- Created the starter project using Kotlin. We decided on using MVP architecture. Therefore I created the base classes(BaseFragment, BasePresenter etc.) - Implemented the MainActivity of the project which will be the host for other fragments. - Implemented default styling for Button and TextView. - Implemented bottom navigation bar to the project. - Did some refactor on MainFragment and updated the layout with Material Components. - Implemented project creation page. - Implemented dependency injection to project creation page. - Added PaperLayer logo to MainFragment and application launcher icon.

Challenges Met During DevOps

As we are a big development team, we needed CI for process mechanics and some automation. For CI, we used the Action feature of Github. By following the steps of Github Action documentation, we implemented some YML files that contain some commands running at Ubuntu. When we open a pull request from our branch to the master branch, these actions are being triggered to run. Implementing scripts for CI was an easy part for us because these scripts contain all commands that we run on a local computer and these were working fine. All we did was automate them. We didn't encounter any challenges for CI.

For CD, we need to dockerize our repository. After we looked at the Docker documentation, we implemented Dockerfile and we want to dockerize frontend and backend separately. When we tried to create a docker image on a local computer and run this image, some of us encountered some issues. This issue was caused by the OS. In the macOS and Linux, there was no problem, but in the Windows environment, we took some exceptions and didn't run this backend image. After researching the error, we found that **'standard_init_linux.go:211'** is a well-known issue for Windows users and we edited all .sh files in the repo and change the EOL from CRLF to LF. After we handled this issue, all members of the team were able to build and run the frontend and backend images with Dockerfile. Our next step was to deploy our repository to Amazon ECS. For this configuration, we used a workflow presented by AWS under the Github Action tab.

According to this workflow, we needed an ECR repository to store our images, a task-definiton.json file, an ECS cluster, and an ECS service to create a container in this service and run images in this cluster. On our first try, we encountered port issues. In the amazon services, security issues are general problems. We have Dockerfiles for frontend and backend and it contains the port number to run in this port. Also, there was port configuration while creating a container and this port had to match with our port in Dockerfile. This was one of the issues we encountered in the CD process.

After that, we needed the task-definition.json file in our repository. It consists of some information about service, image, compatibilities, CPU, and memory, etc. When we pushed the image to ECR after the build process, the ECS service created a task for an image that places in the 'task-definition' JSON file but couldn't find the image we send to the ECR repository. We realized that it was caused by the absence of a match between the image in ECR and the image in the task-definition file. We changed the 'image' value to the URI of the ECR repository in the task-definition file. So, images were matched, a task was created, and run successfully. We had a Public IP and we connected the backend service and frontend web application. Also, we faced another issue that was about Public IP address. When we took a release of our repository, CD workflows are being triggered to run.

After completed the CD process, images of the frontend and backend were sent to the ECR and were run in the clusters. But I realized that Public IP always changed whenever any deployment process finished. When we took a release as v1.0.0, it gave a Public IP and it changed at release v1.0.1. To handle this issue, we created a load balancer that receives public network traffic and routes it to your Amazon ECS container instances. We reconfigured the ECR cluster and service with the load balancer name. The Load Balancer has a DNS name and with this URL, we can reach backend service even if Public IP changed. We also created one more load balancer with a different DNS name for the frontend web application.

Requirements

We've updated our requirements according to the feedback from our customer and new team members. You can see updated glossary terms and requirements as highlighted.

Glossary

- **Guest User** : The user who is only able to view the profiles that are public
- **Registered User** : The user who is registered to the system. **User** in the rest of glossary and requirements imply registered user.
- **Project Creator (Poster)** : The user who created a certain project.
- **Member** : Users that collaborate on a certain project.
- **Team** : A group of members.
- **Teammate** : The relation between two members.
- **Project Coordinators** : Members with the highest rank on a certain project.
- **Profile Page** : A page that stores information about a certain user.
- **Workspace** : The environment where members perform collaboration within a project.

- **Description** : The content describes the mechanism and the policy of the regarding the project
- **File** : The material that is a part of a certain project.
- **Search Engine** : A search tool which guests and users search content with keywords. Also contains advanced search with filters.
- **Semantic Search** : Semantic search denotes search with meaning, as distinguished from lexical search where the search engine looks for literal matches of the query words or variants of them, without understanding the overall meaning of the query.
- **Filter** : Detailed search criterion used in advanced search.
- **Event** : Journal submission activities, academic conferences, funded projects (e.g Tubitak Projects).
- **Custom Event** : Events that are created by users.
- **Public** : The content that can be reached by anyone.
- **Private** : The content that can be reached only by the ones who are allowed.
- **Secure Enough Password** : Consists of at least six characters (and the more characters, the stronger the password) that are a combination of letters, numbers, and symbols.

1. Functional Requirements

1.1. User Requirements

- 1.1.1.Guest User
  - **1.1.1.1.** Guests shall be able to register to the system.
  - **1.1.1.2.** Guests shall be able to search for users, papers or projects.
  - **1.1.1.3.** Guests shall be able to see upcoming events, project titles and descriptions.
  - **1.1.1.4.** Guests shall be able to see public profile pages.

- 1.1.2. Registration and Sign in
  - **1.1.2.1.** Users shall be able to register with a unique e-mail address, password, and full name.
  - **1.1.2.2.** Users should be able to sign up with their Google accounts.
  - **1.1.2.3.** Users shall be able to sign in using an email address and a password.
  - **1.1.2.4.** Users shall be able to agree to Terms of Service and Privacy Policy while registering.
  - **1.1.2.5.** Users shall be able to reset their forgotten password.
  - **1.1.2.6.** A verification email shall sent to users after they sign up.

- 1.1.3. Project Creation and Gathering
  - **1.1.3.1.** Users shall be able to post their paper/project ideas/topics to collaborate with other users.
  - **1.1.3.2.** Posters shall be able to set the state of the post to "Seeking for Collaborators".
  - **1.1.3.3.** Users shall be able to specify requirements for collaboration on their posts.
  - **1.1.3.4.** Users shall be able to state their posts as public or private.
  - **1.1.3.5.** Users shall be able to send a collaboration request for a public post that is in the "Seeking for Collaborators" state.
  - **1.1.3.6.** Posters shall be able to send an invitation to another user to contribute to the post they created that is in the "Seeking for Collaborators" state.
  - **1.1.3.7.** Members shall be able to create a suggestion for inviting new users if the project is in "Seeking for Collaborators" state.
- #### 1.1.4. Project Development
  - **1.1.4.1.** Users shall be able to share and monitor the submission document, codes, and any other file for the projects they are collaborating on.
  - **1.1.4.2.** Posters shall be able to add a custom event to the post with a title, description, date, link, location, submission deadline, and type.
  - **1.1.4.3.** Posters shall be able to add a milestone to the post with a description and date.
  - **1.1.4.4.** Posters shall be able to change the state of the post.
  - **1.1.4.5.** Posters shall be able to link an existing event to their project.
  - **1.1.4.6.** Members shall be able to determine and specify the requirement of the project that they are assigned to.

- 1.1.5. Profile System
  - **1.1.5.1.** Users shall be able to edit their profile information.
  - **1.1.5.1.1.** Users shall be able to change or add new photos for their profiles
  - **1.1.5.1.2.** Users shall be able to change their shown biological gender with one of the given next: 'male','female','do not want to share'
  - **1.1.5.1.3.** Users shall be able to decide whether to give the information of their age or not
  - **1.1.5.1.4.** Users shall be able to share their interests unrelated to their academic profession such as cosmology, astrology in their profiles.
  - **1.1.5.2.** Users shall be able to provide the information regarding expertise, bio, affiliation, recent publications, research area manually or by linking their Google Scholar or ResearchGate accounts
  - **1.1.5.3.** Users shall be able to add ratings and comments to their current or previous teammates.
  - **1.1.5.4.** Users shall be able to set their information either private or public to all users.
  - **1.1.5.5.** Users shall be able to be report other user profiles for these reasons: Disturbing other users, Sharing unrelated or disturbing posts, Spam, Fake Profile, Stolen Account .
  - **1.1.5.6.** Users shall be able to decide whether it shall appear or be hidden on the profile page's certain sections: Bio, Age, Gender, Affiliations

- 1.1.6. Search
  - **1.1.6.1.** Guests and users shall be able to search events, projects and other users with the search engine.
  - **1.1.6.2.** Users shall be able to utilize the advanced search facility to customize the search with filters.
  - **1.1.6.3.** Users should be able to sort search results
    - **1.1.6.3.1** Users should be able to sort user-related search results with these criteria: alphabet order, connection with common teammates, rating.
    - **1.1.6.3.2** Users should be able to sort project-related search results with these criteria: alphabet order, relation with users' interest.
    - **1.1.6.3.3** Users should be able to sort event-related search results with these criteria: alphabet order, date, submission deadline.

- 1.1.7. Follow
  - **1.1.7.1.** Users shall be able to follow other users with public profile pages.
  - **1.1.7.2.** Users shall be able to send follow requests for private profile pages.
  - **1.1.7.3.** Users with private profile pages shall be able to accept or decline the following requests.
  - **1.1.7.4.** Users shall be able to keep track of the activities that belong to the followed users and collaborators.
  - **1.1.7.5.** Users shall be able to unfollow other users.

- 1.1.8. Project Page:
  - **1.1.8.1.** Project Coordinator shall be able to provide the information regarding project manually.
    - **1.1.8.1.1.** The coordinator shall be able to provide the title of project.
    - **1.1.8.1.2.** The coordinator shall be able to provide the description of project.
    - **1.1.8.1.3.** The coordinator shall be able to provide the type of project that what project is for, such as conference, the project for an institution or journal.
    - **1.1.8.1.4.** The coordinator shall be able to provide the tags of project.
    - **1.1.8.1.5.** The coordinator shall be able to provide the due date of project.
    - **1.1.8.1.6.** The coordinator shall be able to provide the accessibility of project.
    - **1.1.8.1.7.** The coordinator shall be able to provide the state of project such as Open For Collaborators or Seeking For Collaborators, In Progress.
    - **1.1.8.1.8.** The coordinator shall be able to add people to the collaborators of project.
  - **1.1.8.2.** Those who're not collaborators can view the project's public information

1.2. System Requirements

- 1.2.1. Search Engine

- **1.2.1.1** The system shall provide a search engine that supports basic and advanced search.
  - **1.2.1.1.1** Basic search shall support searching with name and tag.
  - **1.2.1.1.2** Advanced search shall support searching with the institution, rating, and skills for the user; project stage, due date and linked event for projects; date, submission deadline, location and type for events.
- **1.2.1.2** Search engine shall support searching among user profiles, projects, and events.
- **1.2.1.2.1** Search results shall include the public and followed profiles.
- **1.2.1.2.2** Search results shall include public projects.
- **1.2.1.3** Search engine shall support semantic search.
- **1.2.1.3.1** Semantically related content about the search keywords shall be included in search results.

- 1.2.2. Recommendation
  - **1.2.2.1** System shall support a recommendation system to provide related content to users.
  - **1.2.2.1.1** Recommendation system shall be based on users' previous projects, interest areas, ratings, and skills.
  - **1.2.2.1.2** System shall recommend possible collaborators to project creators.
  - **1.2.2.1.3** System shall provide possible project recommendations to users.

- 1.2.3. Notifications
  - **1.2.3.1** System shall provide notifications about user feeds and updates on projects and events.
  - **1.2.3.1.1** System shall notify users in case of follows, follow requests and ratings on their profiles.
  - **1.2.3.1.2** System shall notify users in case of project collaboration requests and updates on milestones, files and polls about collaborated projects.
  - **1.2.3.1.3** System shall notify users in case of stage changes about followed and collaborated projects.

- 1.2.4. Profile Page
  - **1.2.4.1** System shall ensure that private profile pages are displayed to followers.
  - **1.2.4.2** System shall provide necessary mechanism for users to link their profile page with Google Scholar or ResearchGate.

- 1.2.5. Project Structure
  - **1.2.5.1** Projects shall consist of stages "Open for collaborators", "In Progress", "Submitted to event", "Published".
  - **1.2.5.2** Projects should support stages "Cancelled", "Reopened".
  - **1.2.5.3** System shall provide a text editor to edit project files.
  - **1.2.5.4** System shall support an upload file functionality for the files that are less than 5MBs.
  - **1.2.5.5** System shall make private projects to be visible by only the collaborators.
- #### 1.2.6. Project Page
  - **1.2.6.1** System shall provide a project page to the collaborators to be viewed by all the users in the system.
  - **1.2.6.2** System shall ensure that project's private information isn't shown to those who are not collaborators.
  - **1.2.6.3** System shall ensure that guest users can not collaborate or request collaboration.
  - **1.2.6.4** System shall provide necessary mechanism for users to link their profiles with the project. ## 2. Non-Functional Requirements

2.1.Performance Requirements

- **2.1.1.** Respond times should be less than 3 seconds.
- **2.1.2.** The system should work 7/24 with no more than 1% downtime.

2.2.Legal Constraints

- **2.2.1.** Not allowed the processing of personal data outside the legitimate purpose for which the personal data was collected.
- **2.2.2.** Personal data shall be deleted once the legitimate purpose for which it was collected is fulfilled.
- **2.2.3.** The controller of personal data has the accountability to ensure that personal data is protected and GDPR requirements respected, even if the processing is being done by a third party.
- **2.2.4.** Intention to process personal data beyond the legitimate purpose for which that data was collected, a clear and explicit consent must be asked from the data subject.

2.3.Security Requirements

- **2.3.1.** System should store hashed passwords.
- **2.3.2.** System objects should be encrypted with MD5.
- **2.3.3.** System should use HTTPS Protocol.
- **2.3.4.** System should be backed up to AWS at the end of each day.

2.4.Portability Requirements

- **2.4.1.** Mobile Application should support Android 6 or later.
- **2.4.2.** Mobile Browsers shall redirect to the mobile application.
- **2.4.3.** Web Application should support Chrome, Firefox, Safari or Opera.

2.5.Implementation-Requirements

- **2.5.1.** The application should be dockerized.
- **2.5.2.** The implementation of the system should follow W3C standards and W3C Activity Streams Protocol.
- **2.5.3.** There shall be a web platform and a native Android application that supports the same functionalities.
- **2.5.4.** The color designs of the platform will be made to improve the experience of color-blind people.

Design Documents

We haven't changed our mockups. You can see them here: https://github.com/bounswe/bounswe2020group3/wiki/Mockups

We haven't changed our use-case diagrams. You can see it here: https://github.com/bounswe/bounswe2020group3/wiki/Use-Case-Diagram

We've updated our class diagram and uploaded to the wiki page. Since it's a huge image and hard to read without zoom in, we don't include the image here. Red text shows updated fields and methods. We've updated them according to the needs of the updated requirements and feedback from the customer. Here you can see the latest version of the class diagram: https://github.com/bounswe/bounswe2020group3/wiki/Class-Diagram

We haven't changed our sequence diagrams. You can see it here: https://github.com/bounswe/bounswe2020group3/wiki/Sequence-Diagrams

API documentation

We generated our API documentation automatically with swagger. You can check the latest version by using this link: http://ec2co-ecsel-1egsefcb8a7i3-34132836.us-east-2.elb.amazonaws.com/swagger/

We've also exported it after the milestone 1 presentation. Since it's a long document, we placed it at the end of this file. You can check it at Appendix

Project Plan

Task	Requirement	Details	Start Date	Deadline	Assignee(Frontend)	Assignee(Backend)	Assignee(Android)	Predecessor
Registration/Sign in	1.1.1.1.	Registration	03.11.2020	10.11.2020	Barış	Ali / Adil	Mahir	
	1.1.1.4.	Profile pages			Mert	Ahmet Emir	Yahya	
	1.1.2.1.	Registration info			Barış	Ali / Adil	Mahir	
	1.1.2.3.	Sign in info			Barış	Ali / Adil	Ramazan	
	1.1.2.5	Password reset			-	Ali / Adil	Mahir	
Profile Page	1.1.5.1.1	Profile info	03.11.2020	10.11.2020	Emirhan	Ahmet Emir	Yahya	
	1.1.5.1.2				Emirhan	Ahmet Emir	Yahya	
	1.1.5.1.3				Emirhan	Ahmet Emir	Yahya	
	1.1.5.1.4				Emirhan	Ahmet Emir	Yahya	
	1.1.5.6	Hidden info			Emirhan	Ahmet Emir	Yahya	
Project Creation	1.1.3.1.	Posting a project	10.11.2020	17.11.2020	Barış	Buse / Ahmet Emir	Yusuf	
	1.1.3.2.	Seeking for collaborators state			Barış	Buse / Ahmet Emir	Yusuf	
	1.1.3.3.	Post project requirements			Barış	Buse / Ahmet Emir	Yusuf	
	1.1.3.4.	Set project public/private			Barış	Buse / Ahmet Emir	Yusuf	
	1.1.4.1.	Posting files			Mert	Adil	Ramazan	
	1.1.4.3.	Add milestone			-	Buse / Ahmet Emir	Ramazan	
	1.1.4.4.	Change state of post			Mert	Buse / Ahmet Emir	Ramazan	
	1.1.4.5.	Link event to project			Mert	Ali	Mahir	
Project Page	1.1.8.1	Setting project details	10.11.2020	17.11.2020	Mert	Buse / Ahmet Emir	Mahir	
Event Creation	1.1.4.2.	Adding an event	17.11.2020	24.11.2020	Barış	Ali	Mahir	
Presentation	-	Prepare milestone presentation	17.11.2020	24.11.2020	All team	All team	All team	
Report	-	Prepare milestone report	24.11.2020	29.11.2020	All team	All team	All team	
Project Page	1.1.8.1	Tag improvements	01.12.2020	08.12.2020	Yunus	Buse	Mahir	Earlier 1.1.8.1
Event Creation	1.1.4.2.	Event improvements	01.12.2020	08.12.2020	Yunus	Ali	Mahir	Earlier 1.1.4.2.
Editing Project	1.1.4.3.	Milestone improvements	01.12.2020	08.12.2020	Emirhan	Buse	Ramazan	Earlier 1.1.4.3.
	1.1.4.4.	Change state of post			Emirhan	Ahmet Emir	Ramazan	Earlier 1.1.4.4.
	1.1.4.5.	Link event to project			Emirhan	Ali	Ramazan	Earlier 1.1.4.5.
Profile System	1.1.5.4	Set profile public/private	01.12.2020	08.12.2020	Barış	Ahmet Emir	Yahya	
	1.1.5.6	Hide hidden fields from profile			Barış	Ahmet Emir	Yahya	Earlier 1.1.5.6.
Project Structure	1.2.5.1	Add new stages	01.12.2020	08.12.2020	Mert	Furkan	Yusuf	1.1.4.4.

Task	Requirement	Details	Start Date	Deadline	Assignee(Frontend)	Assignee(Backend)	Assignee(Android)	Predecessor
	1.2.5.2	Add new stages			Mert	Furkan	Yusuf	1.1.4.4.
	1.2.5.4	Add file to project			Mert	Adil	Yusuf	1.1.4.1
Project Page	1.1.8.2	View the project's public information	01.12.2020	08.12.2020	Barış	Adil	Yahya	Earlier 1.1.8.1
Follow	1.1.7.1	Follow other users with public profile pages	08.12.2020	15.12.2020	Yunus / Emirhan	Adil	Ramazan	1.1.5
	1.1.7.2	Send follow requests for private profile pages			Yunus / Emirhan	Adil	Ramazan	1.1.5
	1.1.7.3	Accept/decline following requests			Yunus / Emirhan	Adil	Ramazan	1.1.5
	1.1.7.5	Unfollow			Yunus / Emirhan	Adil	Ramazan	1.1.5
Profile Page (Functional)	1.2.4.1	Private profile pages displayed to followers	08.12.2020	15.12.2020	Yunus / Emirhan	Adil	Ramazan	1.1.5
Search	1.1.6.1	Search events, projects and other users	08.12.2020	15.12.2020	Mert	Ali	Mahir	
	1.1.6.2	Customize the search with filters			Barış	Ali	Yusuf	
	1.1.6.3.1	Sort user-related search results with these criteria: alphabet order, connection with common teammates, rating			Mert	Ahmet Emir	Yahya	1.1.5
	1.1.6.3.2	Sort project-related search results with these criteria: alphabet order, relation with users' interest			Mert	Furkan	Yahya	1.1.3
	1.1.6.3.3	Sort event-related search results with these criteria: alphabet order, date, submission deadline			Mert	Buse	Yahya	1.1.4.2.
Search Engine (Functional)	1.2.1.1.1	Basic search shall support searching with name and tag.	08.12.2020	15.12.2020	Mert	Ali	Mahir	
	1.2.1.1.2	Advanced search shall support searching with the institution, rating, and skills for the user; project stage, due date and linked event for projects; date, submission deadline, location and type for events.			All team	All team	All team	
	1.2.1.2.1	Search results shall include the public and followed profiles.			All team	All team	All team	
	1.2.1.2.2	Search results shall include public projects.			All team	All team	All team	

Task	Requirement	Details	Start Date	Deadline	Assignee(Frontend)	Assignee(Backend)	Assignee(Android)	Predecessor
	1.2.1.3	Search engine shall support semantic search. Semantically related content about the search keywords shall be included in search results.			All team	All team	All team	
Project Gathering	1.1.3.5	Sending collaboration request	15.12.2020	22.12.2020	Mert	Ahmet Emir	Yusuf	1.1.3 - 1.1.5 - 1.1.6
	1.1.3.6	Owners sending invitation to users			Mert	Ahmet Emir	Yusuf	1.1.3 - 1.1.5 - 1.1.6
	1.1.3.7	Members suggesting new users to project			Mert	Ahmet Emir	Yusuf	1.1.3 - 1.1.5 - 1.1.6
Profile System	1.1.5.2 - 1.2.4.2	Linking Google Scholar or ResearchGate accounts.	15.12.2020	22.12.2020	Yunus	Adil	Yahya	
	1.1.5.3	Add ratings and comments to teammates			Emirhan	Ali	Ramazan	1.1.3
Follow	1.1.7.4	Activity page about followed users and collaborators	15.12.2020	22.12.2020	Barış	Buse	Mahir	1.1.7
Guest User	1.1.1.2	Guests search	15.12.2020	22.12.2020	Barış / Yunus	Ali	Ramazan	
	1.1.1.3	Guests home page			Barış / Yunus	Ali	Ramazan	
Registration/Sign in	1.1.2.4	Terms of Service and Privacy Policy	15.12.2020	22.12.2020	Emirhan	Buse	Yahya	
	1.1.2.6	Verification email			Barış	Furkan	Yahya	1.1.1
Project Page (Functional)	1.2.6.2	System shall ensure that project's private information isn't shown to those who are not collaborators.	22.12.2020	29.12.2020	All	All	All	
	1.2.6.3	System shall ensure that guest users can not collaborate or request collaboration.			All	All	All	
	1.2.6.4	System shall provide necessary mechanism for users to link their profiles with the project.			All	All	All	
Project Structure	1.2.5.5	System shall make private projects to be visible by only the collaborators.	22.12.2020	29.12.2020	All	All	All	
Annotation	-	Annotation Research	22.12.2020	29.12.2020	All	All	All	
Presentation	-	Prepare milestone presentation	22.12.2020	29.12.2020	All	All	All	
Report	-	Prepare milestone report	29.12.2020	05.01.2021	All	All	All	

Task	Requirement	Details	Start Date	Deadline	Assignee(Frontend)	Assignee(Backend)	Assignee(Android)	Predecessor
Profile System	1.1.5.5	Report other user profiles for these reasons: Disturbing other users, Sharing unrelated or disturbing posts, Spam, Fake Profile, Stolen Account .	29.12.2020	05.01.2021	Barış	Furkan	Yahya	
Project Structure	1.2.5.3	Text editor to edit project files	05.01.2021	12.01.2021	Barış / Mert	Adil / Buse	Yahya / Ramazan	1.1.4.1
Notifications	1.2.3.1.1	Notify users in case of follows, follow requests and ratings on their profiles	05.01.2021	12.01.2021	Emirhan	Furkan	Mahir	
	1.2.3.1.2	Notify users in case of project collaboration requests and updates on milestones, files and polls about collaborated projects.			Emirhan	Furkan	Mahir	
	1.2.3.1.3	Notify users in case of stage changes about followed and collaborated projects.			Emirhan	Furkan	Mahir	
Recommendation	1.2.2.1.1	Recommendation system shall be based on users' previous projects, interest areas, ratings, and skills.	05.01.2021	12.01.2021	Yunus	Ali / Ahmet Emir	Yusuf	1.1.3
	1.2.2.1.2	System shall recommend possible collaborators to project creators.			Yunus	Ali / Ahmet Emir	Yusuf	1.1.3
	1.2.2.1.3	System shall provide possible project recommendations to users.			Yunus	Ali / Ahmet Emir	Yusuf	1.1.3
Registration and Sign In	1.1.2.2	Users should be able to sign up with their Google accounts.	12.01.2021	19.01.2021	Yunus	Adil	Yahya	
Annotation	-		12.01.2021	19.01.2021	All	All	All	Annotation Research
Presentation	-	Prepare milestone presentation	12.01.2021	19.01.2021	All	All	All	

User Scenarios

Web Scenario

Background

Umut Cihat Kabasarı is a master's student. He has no prior research experience and he is working on his master's thesis. He wants his project advisor to see the latest version of the project at all times and he is considering submitting the outcome of his project as a paper to a journal.

Steps Shown in Demo

1. He registers to paperlayer with a unique username, first name, (optional) middle name, last name,email and password.
2. He is forwarded to the login page and he logs in.
3. He is forwarded to the home screen. He clicks create a project to create a new project.
4. He enters the details of his project and sets it public so that his advisor will be able to see it.He also sets the project type in progress since he doesn't wany any collaborators. And he leaves

event blank because he doesn't know to which journal he is going to submit his paper to.
5. He then goes back to the homepage and scrolls through other projects.
6. He clicks a project and looks at the details of the project.

Android Scenario

Background

Özlem Türeci is a scientist. At the early stages of the pandemic, she wants to work on a project to develop a vaccine for covid19. She wants to work with a group of skilled and hardworking scientists to develop a vaccine as fast as possible. For this purpose she can use PaperLayer. Because on PaperLayer you can find other scientists to collaborate on your project or research. This way, she and other scientists can share their knowledge and work together for the same purpose. She searched for a platform to find other colleagues and found Paperlayer.

Preconditions

1. She is a registered user
2. She already downloaded the PaperLayer app to his phone

Steps Shown in Demo

1. She logins to PaperLayer app
2. She views his profile page
3. She views and edits his profile page
4. She opens project creation page
5. She enters project name, project description and requirements
6. She chooses to create public project
7. Since she needs another researchers, she selects project state as seeking for collaborators
8. She picks a due date for the research project
9. She creates a Covid19 research project
10. After that she checks her projects page to make sure project is created
11. She clicked details of project to check other descriptions

Evaluation of tools

Backend Tools

Django: Django makes the most part of the backend development easier as it has numerous bulit-in functionalities, but this has side effects like steep learning curve so we are still trying to fully control it.

Poetry: It creates a virtual environment to include only the same versions of python modules accross the backend team. This way, we have less erorrs and backend development is streamlined.

Docker: It keeps all required dependecies for the project such as python, python modules, binary files(Postgresql, OpenSSL etc.) in one place. So each team member can have the same development environment in their machines. Without Docker, installing dependencies one by one and making sure that everyone has the same version would be hard to accomplish.

Swagger: Swagger is a great tool for us to create endpoint documentation automatically. The endpoints we create in Django end-up in Swagger documentation with its input and output.

Django REST Framework: Django REST framework is a powerful and flexible toolkit for building Web APIs. We utilized it for every endpoint and it made the development process easier.

VS Code: VS code is a great user friendly code editor. It allow us to work more efficiently.

Github Desktop: It makes everything about Github easier and understandable with its easy-to-use UI.

Discord: We heavily utilize Discord. It is convenient to use because we can have one app for one-to-one and group chats, meetings and screen sharing.

Whatsapp: Whatsapp is used for instant communication in our group.

Frontend Tools

Github: We used GitHub actions, issues, and projects to plan, automate test process and check the health of the development process. To use GitHub effectively, we found a branch creation system that for each issue, we create a branch that its name is the one of the directly related issues.

Docker: We used Docker to easily deploy the application into the server and run faster on our own computers due to the fact that it automates installations and environment configurations

Nginx: We used Nginx to dynamic HTTP content handling

ESLint: We used ESLint for static code analysis on health test and code quality check

Discord: We heavily utilize Discord. It is convenient to use because we can have one app for one-to-one and group chats, meetings and screen sharing to plan, report, and discuss the state of development.

Whatsapp: We used Whatsapp for fast or direct communication to plan, report, and discuss the state of development.

VS Code: It is a powerful and user friendly IDE and we generally used this because it has a lot of extensions to design and develop a webpage with React, and its git integration, keymap support and simultaneous coding support is very nice. Thus, it allows us to work more efficiently.

React: We used React as the web framework to develop each web page of the platform. It makes the frontend development easier because it has a great and nice community, and this property led us to find a solution for the bugs etc. easily. Since React is a simple framework that it uses the DOM like HTML, it leads learning period to become shorter. As two sides of a coin, React has some problems: there is no direct supports for some features such as state management and routing. Therefore, we need to use additional libraries

Material.ui: We used material.ui library in order to make development and design processes faster and easier. For example, we use Snackbar, Toolbar, MuiAlert, Grid and many other components with some minor changes in the pages and they were very useful overall.

Axios: we used axios to handle promise-based HTTP request

Date-io: we used date-io for date management

Android Tools

Android Studio: Android Studio is the most powerful and most used IDE for Android development. It has Git, terminal integtrations. Also it has Gradle support.

Kotlin : We decided to use Kotlin for Android project of PaperLayer. There are several reasons behind these decision. One of them is Google announced Kotlin as official language for Android development in 2017. Some of the other reasons are there is lambda expressions, null safety and extension functions. These features are not supported on Java.

MVP Pattern: Before we started to implement Android project we discussed about whether we should use some software architecture or not. Then, we decided on using MVP architecture. MVP is Model-View-Presenter and what it does is basically seperate the functionality in three parts. Model is for managing API requests, View is for UI related operations and Presenter is for logic operations. Following this architecture we created a project which is easier to implement, debug and test. Also, by following MVP pattern we tried to accomplish S.O.L.I.D principles as much as possible.

Retrofit : We used Retrofit for network operations. Retrofit is a network provider library used in almost every Android project. We created Retrofit instance in our project and we called that instance whenever we need to send a request and get a response. Currently we didn't add AuthenticationInterceptor in our Retrofit instance but we will add it for easier implementation later.

RxJava: RxJava is a library that enables us to use reactive programming. We used RxJava to handle request and response with combination of Retrofit instance. Basically when we create and send a request our RxJava instance waits for the response. Whenever the response comes(either success or error) our RxJava is notified and after that we can handle the response.

Moshi : Moshi is a library for converting request or response to Kotlin objects. Using Moshi we created our requests as Kotlin objects and when sending request it automatically converts it to JSON object. Also our response is in JSON format and by using Moshi it is converted to Kotlin object. Moshi is great for handling request and response in object oriented programming.

Dagger : Dagger is a Dependency Injection library for Android. We used Dagger for this purpose.

Material Components : We used Material Components library for customizing UI elements.

Git : We used Git as version control system. It allowed us to work as a team.

Evaluation of managing

To properly manage the progress we decided that every subteam should have their own meeting besides the weekly team meeting. We discuss the outlines of tasks together in weekly meetings and after that every subteam namely backend, frontend, and Android, sets-up a special meeting where tasks are discussed deeply and assigned to members. Despite these attempts, controlling the progress of each team member and trying to be on the same page with them while being sure no task is delayed is challenging especially in a team with 12 students who has myriad of assignemnts, and homeworks from other courses and jobs. Initally, we thought that backend had to be ahead of other subteams so that frontend and Android can use the available endpoints. However, this resulted in backend team rushing tasks without setting the rules properly with other subteams for long-term convenience. After a while, we realized that this is costing us time because for almost every endpoint, there was a feedback from either the frontend, or the Android team suggesting modifications. Now we decided to set up a meetings for members who are working on the same functionality so that they can thoroughly plan and decide the expected outcome. We are hoping to apply this system for the tasks of following milestones.

Assessment of the customer presentation

The presentation helped us set a clear direction for our project thanks to the many insigthful feedbacks and suggestions we got from our customers. The reaction from our customers was generally positive as they were following the presentation, and at the end our customer came up with good questions about our path moving forward. The main focus of the customer was the process of joining a project. Although there were some mixed opinions, the customer generally wanted the project owner to describe clear requirements and the collaborators to fit these requirements in order to join the project. This was a thought-provoking idea and led to some internal discussions in the team and has been helpful as it made the product clearer before further development.

During our earlier customer meetings, our customer warned us to include interesting functinalities to make sure we are on the same page about important features. So we included important functionalities like project creation, profile creation and profile view. In the presentation, our team did a good job coming up with realistic scenarios, and displaying the strong points of our product. We managed the time well and captured the attention of the audience with interesting scenarios. What helped us in this process was that we practiced the presentation earlier internally and made sure everything was working as planned. The lesson we learned for the next time is that we should focus more on the Frontend and communicate more across teams, like Android, Frontend and Backend.

The code structure and group process

We have three different teams namely Backend, Android and Frontend all of which are working separately having their preferences on how to implement their part of the project. Group process might differ from team to team. Developers have their own branches for implementing, improving or debugging a feature. We do not have strict rules on how to use branches if no one violates another developer's branch or the main branch. Every branch is created with a prefix which tags the team it belongs to. After completing their coding developers need to create a pull request. Pull requests can have tags representing how critical they are, what they change, which team they belong to, their status and their type all of which helps reviewers. Every pull request is tested by various continuous integration tools such as Github Actions and Docker. If every test is passed, the code should be reviewed by other developers before being merged to the main branch. Number of reviewers depends on the size and priority of the pull request. After passing, tests and getting confirmed by reviewers, the newly requested code can be merged into the main branch. Backend team follows pep-8 coding standards for a more structured, readable and understandable code. They use a tool called flake8 to test and ensure our code is up to those standards. Poetry and pytest are the other tools they use. Frontend team uses Eslint for a cleaner code. They make sure every pull request has explanatory comments so that reviewers have an easier time to review. They used material-ui themes and components for minimizing disorders in CSS code. Android team uses MVP (model-view-presenter) pattern to organize the presentation layer in the application. They require at least a reviewer for a pull request like the other teams.

Appendix

PaperLayer API

Overview

API documentation of PaperLayer

Version information

Version: v1

URI scheme

Host: http://ec2co-ecsel-1egsefcb8a7i3-34132836.us-east-2.elb.amazonaws.com

Base path: /api

Schemes: HTTP

Consumes

application/json

Produces

application/json

Security

Token

Please write "Token" before the key.

Type: apiKey
Name: Authorization
In: HEADER

Paths

POST /auth/

Parameters

Type	Name	Schema
Body	**data** *required*	AuthToken

Response

HTTP Code	Schema
201	AuthToken

POST /events/

Parameters

Type	Name	Schema
Body	**data** *required*	Event

Response

HTTP Code	Schema
201	Event

GET /events/

Parameters

Type	Name	Schema
Query	**event_type** *optional*	string

Response

HTTP Code	Schema
200	<Event> array

GET /events/{id}/

Parameters

Type	Name	Description	Schema
Path	**id** *required*	A unique integer value identifying this event.	integer

Response

HTTP Code	Schema
200	Event

PUT /events/{id}/

Parameters

Type	Name	Description	Schema
Path	**id** *required*	A unique integer value identifying this event.	integer
Body	**data** *required*		Event

Response

HTTP Code	Schema
200	Event

DELETE /events/{id}/

Parameters

Type	Name	Description	Schema
Path	**id** *required*	A unique integer value identifying this event.	integer

Response

HTTP Code	Schema
204	No Content

PATCH /events/{id}/

Parameters

Type	Name	Description	Schema
Path	**id** *required*	A unique integer value identifying this event.	integer
Body	**data** *required*		Event

Response

HTTP Code	Schema
200	Event

POST /files/

Parameters

Type	Name	Schema
FormData	**file** *required*	file
FormData	**project** *required*	string (uri)
FormData	**remark** *required*	string

Response

HTTP Code	Schema
201	File

GET /files/

Parameters

Type	Name	Schema
Query	**project** *optional*	string

Response

HTTP Code	Schema
200	<File> array

GET /files/{id}/

Parameters

Type	Name	Description	Schema
Path	**id** *required*	A unique integer value identifying this file.	integer

Response

HTTP Code	Schema
200	File

PUT /files/{id}/

Parameters

Type	Name	Description	Schema
Path	**id** *required*	A unique integer value identifying this file.	integer
FormData	**file** *required*		File
FormData	**project** *required*		string (uri)
FormData	**remark** *required*		string

Response

HTTP Code	Schema
200	File

DELETE /files/{id}/

Parameters

Type	Name	Description	Schema
Path	**id** *required*	A unique integer value identifying this file.	integer

Response

HTTP Code	Schema
204	No Content

PATCH /files/{id}/

Parameters

Type	Name	Description	Schema
Path	**id** *required*	A unique integer value identifying this file.	integer
FormData	**file** *required*		File
FormData	**project** *required*		string (uri)
FormData	**remark** *required*		string

Response

HTTP Code	Schema
200	File

GET /files/{id}/retrieve_file

Parameters

Type	Name	Description	Schema
Path	**id** *required*	A unique integer value identifying this file.	integer

Response

HTTP Code	Schema
200	File

POST /logout/

Description

Logs the user out, has to be authenticated

Response

HTTP Code	Schema
201	No Content

POST /milestones/

Parameters

Type	Name	Schema
Body	**data** *required*	Milestone

Response

HTTP Code	Schema
201	Milestone

GET /milestones/

Response

HTTP Code	Schema
200	<Milestone> array

GET /milestones/{id}/

Parameters

Type	Name	Description	Schema
Path	**id** *required*	A unique integer value identifying this milestone.	integer

Response

HTTP Code	Schema
200	Milestone

PUT /milestones/{id}/

Parameters

Type	Name	Description	Schema
Path	**id** *required*	A unique integer value identifying this milestone.	integer
Body	**data** *required*		Milestone

Response

HTTP Code	Schema
200	Milestone

DELETE /milestones/{id}/

Parameters

Type	Name	Description	Schema
Path	**id** *required*	A unique integer value identifying this milestone.	integer

Response

HTTP Code	Schema
204	No Content

PATCH /events/{id}/

Parameters

Type	Name	Description	Schema
Path	**id** *required*	A unique integer value identifying this milestone.	integer
Body	**data** *required*		Milestone

Response

HTTP Code	Schema
200	Milestone

POST /profiles/

Parameters

Type	Name	Schema
Body	**data** *required*	Profile

Response

HTTP Code	Schema
201	Profile

GET /profiles/

Parameters

Type	Name	Schema
Query	**owner_id** *optional*	number

Response

HTTP Code	Schema
200	<Profile> array

GET /profiles/{id}/

Parameters

Type	Name	Description	Schema
Path	**id** *required*	A unique integer value identifying this profile.	integer

Response

HTTP Code	Schema
200	Profile

PUT /profiles/{id}/

Parameters

Type	Name	Description	Schema
Path	**id** *required*	A unique integer value identifying this profile.	integer
Body	**data** *required*		Profile

Response

HTTP Code	Schema
200	Profile

DELETE /profiles/{id}/

Parameters

Type	Name	Description	Schema
Path	**id** *required*	A unique integer value identifying this profile.	integer

Response

HTTP Code	Schema
204	No Content

PATCH /profiles/{id}/

Parameters

Type	Name	Description	Schema
Path	**id** *required*	A unique integer value identifying this profile.	integer
Body	**data** *required*		Profile

Response

HTTP Code	Schema
200	Profile

POST /projects/

Parameters

Type	Name	Schema
Body	**data** *required*	Project

Response

HTTP Code	Schema
201	Project

GET /projects/

Parameters

Type	Name	Schema
Query	**owner_id** *optional*	number

Response

HTTP Code	Schema
200	< Project > array

GET /projects/{id}/

Parameters

Type	Name	Description	Schema
Path	**id** *required*	A unique integer value identifying this project.	integer

Response

HTTP Code	Schema
200	Project

PUT /projects/{id}/

Parameters

Type	Name	Description	Schema
Path	**id** *required*	A unique integer value identifying this project.	integer
Body	**data** *required*		Project

Response

HTTP Code	Schema
200	Project

DELETE /projects/{id}/

Parameters

Type	Name	Description	Schema
Path	**id** *required*	A unique integer value identifying this project.	integer

Response

HTTP Code	Schema
204	No Content

PATCH /projects/{id}/

Parameters

Type	Name	Description	Schema
Path	**id** *required*	A unique integer value identifying this project.	integer
Body	**data** *required*		Project

Response

HTTP Code	Schema
200	Project

POST /tags/

Parameters

Type	Name	Schema
Body	**data** *required*	Tag

Response

HTTP Code	Schema
201	Tag

GET /tags/

Response

HTTP Code	Schema
200	< Tag > array

GET /tags/{id}/

Parameters

Type	Name	Description	Schema
Path	**id** *required*	A unique integer value identifying this tag.	integer

Response

HTTP Code	Schema
200	Tag

PUT /tags/{id}/

Parameters

Type	Name	Description	Schema
Path	**id** *required*	A unique integer value identifying this tag.	integer
Body	**data** *required*		Tag

Response

HTTP Code	Schema
200	Tag

DELETE /tags/{id}/

Parameters

Type	Name	Description	Schema
Path	**id** *required*	A unique integer value identifying this tag.	integer

Response

HTTP Code	Schema
204	No Content

PATCH /tags/{id}/

Parameters

Type	Name	Description	Schema
Path	**id** *required*	A unique integer value identifying this tag.	integer
Body	**data** *required*		Tag

Response

HTTP Code	Schema
200	Tag

POST /register/

Description

Registers a new user.

Parameters

Type	Name	Schema
Body	**data** *optional*	Register

Response

HTTP Code	Schema
201	Register

GET /users/

Response

HTTP Code	Schema
200	<User> array

GET /users/{id}/

Parameters

Type	Name	Schema
Path	**id** *required*	integer

Response

HTTP Code	Schema
200	User

Definitions

AuthToken

Name	Description	Schema
password *required*	**Min. Length:** 1	string
token *optional* read-*only*		string
username *required*	**Min. Length:** 1	string

Event

Name	Description	Schema
date *required*		string (date)
deadline *required*		string (date)
description *optional*	**Max. Length:** 200	string

event_type
required
		Enum {
		journal submission,
		academic conference,
		funded project
		}

id *optional* *read-only*		integer

title *required*	**Length:** 1 – 100	string
url *optional*	**Max. Length:** 200	string (uri)

File

Name	Description	Schema
file *optional* *read-only*		string (uri)
id *optional* *read-only*		integer
project *required*		string (uri)
remark *required*	**Length:** 1 – 20	string
timestamp *optional* *read-only*		string (date-time)

Milestone

Name	Description	Schema
date *required*		string (date)
description *optional*	**Min. Length:** 1	string
project *required*		string (uri)
url *optional* *read-only*		string (uri)

Profile

Name	Description	Schema
age *optional*		integer
bio *optional*	**Max. Length:** 1000	string
email *optional* *read-only*		string
expertise *optional*		string

gender
optional
		Enum {
		male,
		female,
		do not want share,
		}

id *optional* *read-only*		integer
interest *optional*		string
last_name *optional*	**Length:** 1 – 100	string
middle_name *optional*	**Max. Length:** 100	string
name *optional*	**Length:** 1 – 100	string
owner *optional* *read-only*		string
photo_url *optional*		string
share_affiliations *optional*		boolean
share_age *optional*		boolean
share_bio *optional*		boolean
share_gender *optional*		boolean

Project

Name	Description	Schema
description *optional*	**Max. Length:** 1	string
due_date *optional*	**Max. Length:** 1000	string (date)
events *optional*		< string (uri) > array
id *optional* *read-only*		integer
is_public *optional*		boolean
members *required*		< string (uri)> array
name *required*	**Length:** 1 – 500	string
owner *optional* *read-only*		string

project_type
optional
		Enum {
		Conference,
		Instutution,
		Journal
		}

requirements *optional*	**Min Length:** 1	String

state
optional
		Enum {
		Seeking for collaborators,
		Open for collaborator,
		In progress,
		Done
		}

Register

Name	Description	Schema
email *required*	**Min. Length:** 1	string (email)
first_name *required*	**Min. Length:** 1	string
last_name *required*	**Min. Length:** 1	string
middle_name *optional*		string
password *required*	**Min. Length:** 1	string
username *required*	**Min. Length:** 1	string

Tag

Name	Description	Schema
name *required*	***Length:*** *1 – 500*	string
projects *required*		< string(uri)> array
url *optional* *read-only*		string (uri)

User

Name	Description	Schema
email *optional*	***Max. Length:*** *254*	string (email)
id *optional* *read-only*		integer
profile *optional* *read-only*		<Profile> array
username *required*	**Length:** 1 – 50	string