MILESTONE REPORT 1

29.11.2020

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

Table of 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, a user can follow other users, and meet new people to collaborate with. If a 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, also the system supports semantic search.

Anyone using Platon as a guest can only 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

We started by meeting our newcomers. After introducing our project to them, we reviewed the requirements, diagrams, and the project plan and made some changes to them. Then, we divided our team into subgroups so that every subgroup can professionalize in one of the following: backend, frontend, and android. The backend consists of Halil Umut Özdemir, Hüseyin Can Bölükbaş, Hilal Demir, Alperen Divriklioğlu; android members are Burak Ömür, Ertuğrul Bülbül, Oyku Yilmaz, Orkan Akisu; and in the frontend subteam Ahmet Dadak, Ramazan Yurt, Burhan Akkuş, and Umutcan Uvut are the members.

For communication, we decided to meet in the lab hours and every subteam will arrange their meetings according to the tasks.

Planning

First, every subteam decided on which technologies they will use. In the Backend, Flask framework and MySQL are selected. For Android, we used Kotlin language, and decided to use MVP architecture. Frontend team decided to use React.js. After deciding on those basics, every subteam started by making initial designs. In Android and Frontend, initial design means selecting colors, designing layouts. In the backend, designs of the databases are made.

In a meeting we created an AWS account. By using this AWS account we took an EC2 instance for our production server. Then by using Docker and Docker Compose we created a script for automated deployment of our frontend, backend and database to our production server. At the end by using Travis-CI Continuous Integration tool we created a CI/CD pipeline that firstly runs tests then makes the deployment to our server.

Implementation

According to the project plan, we were supposed to implement the login registration system, profile page, follow mechanism. To do that, every subteam worked hard. Every subteam had meetings to make the division of labor. After the tasks are divided, implementation has begun. Login and registration system flawlessly works in all subteams. Profile page, and follow mechanism are partially delivered. That has some reasons, first, that was the first milestone so we spent too much time on getting to know each other, planning the road ahead. But what took the most time is to create projects from scratch. Second, since it is the first time we tried to merge those subteams, our project plan lacked of time management.

WHAT IS NEXT

For the upcoming milestone, we want to

- evenly divide the tasks
- make time management in a way that it no one has to wait for a week then has to do all the job in the last week
- fix the currently existing bugs,
- deliver the partially delivered deliverables
- create the "workspaces" functionality to allow our users start their research projects on our platform,
- 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	12.11.2020	DELIVERED
2	PROJECT REQUIREMENTS	12.11.2020	DELIVERED
3	ANDROID		
	HOME PAGE	22.11.2020	PARTIALLY
	REGISTER PAGE	22.11.2020	PARTIALLY
	LOGIN PAGE	22.11.2020	DELIVERED
	RESET PASSWORD PAGE	22.11.2020	DELIVERED
	PROFILE PAGE	22.11.2020	PARTIALLY
	LANDING PAGE	22.11.2020	PARTIALLY
4	FRONTEND		
	HOME PAGE	22.11.2020	PARTIALLY
	REGISTER PAGE	22.11.2020	PARTIALLY
	LOGIN PAGE	22.11.2020	DELIVERED
	RESET PASSWORD PAGE	22.11.2020	DELIVERED
	PROFILE PAGE	22.11.2020	PARTIALLY
	LANDING PAGE	22.11.2020	PARTIALLY
5	BACKEND		
	REGISTER ENDPOINTS	22.11.2020	PARTIALLY
	LOGIN ENDPOINTS	22.11.2020	DELIVERED

	RESEARCH INFORMATION ENDPOINT	22.11.2020	DELIVERED
	NOTIFICATION ENDPOINTS	22.11.2020	DELIVERED
	FOLLOW SYSTEM ENDPOINTS	22.11.2020	PARTIALLY
	PROFILE MANAGEMENT ENDPOINTS	22.11.2020	PARTIALLY
	API DOCUMENTATION (SWAGGER)	22.11.2020	DELIVERED

EVALUATION OF DELIVERABLES

PROJECT PLAN

Our project plan from Cmpe352 had different dates, our first subject was to change them. Then after the announcement of milestone presentations, we made changes on the milestone dates. Also we added internal milestones to we will be front of the deadlines.

REQUIREMENTS

Requirements are reviewed according to the feedback of the old and the new members, and also to the feedback of the customer.

BACKEND

Register Endpoint

Explanation

Our backend server provides a mechanism to register to our system by providing name, surname, e-mail, an affinity. And also there is an e-mail based account verification system of our server. Also any user can not register to our system using same e-mail address. In the future, we will add the optional part of the registration information and create a mechanism to add multiple affinities.

Covered Requirements

1.1.1.1, 1.2.1.5, 1.2.6.2

Login Endpoint

Explanation

Our backend server provides a mechanism to login to the system.

Covered Requirements

1.1.2.1

Research Information Endpoint

Explanation

Our backend server provides a mechanism to fetch data from Google Scholar and Research Gate. Also a user can add a custom research information to our system. Also Edit and Deletion operations are supported from our backend server. Lastly, our server updates the information that is fetched from Google Scholar and Research Gare on every 1 hour.

Covered Requirements

1.1.5.3, 1.2.6.3

Notification Endpoint

Explanation

We created a notification manager that manages the notification of the users of our system. Any user can get the list of notifications that a user has. For now only the notifications about the Follow System is added as Notification.

In the future, we planned to add a related notification of each functionality, while implementing that functionality.

Covered Requirements

1.2.4.2

Follow System Endpoint

Explanation

Using our backend server, a user can follow a public profile and send a follow request to a private profile. Also it provides a mechanism to reply to a follow request. Also our follow system provides a list of followers, following, and follow requests of a user.

In the future, we planned to implement the unfollow mechanism of follow system.

Covered Requirements

1.1.3.1, 1.1.3.2, 1.1.3.3, 1.2.6.4

Profile Page Endpoint

Explanation

Our backend provides name, surname, e-mail, an affinity, research information, Research Gate and Google Scholar accounts information of the users. Also by using our backend, any user can edit or delete any of the information other than e-mail. Also by using our backend it is possible to set the privacy of the user account and related profile page.

Our backend server provides a privacy mechanism for the profile page information of private users. It only gives information about a private user account, if the user who requests the information follows the private account.

Also It provides profile information of public users to the all logged in users of our platform.

In the future, we will add profile photo and skills information to the user profile data. And also a functionality to add multiple affinities will be added to the backend server.

Covered Requirements

1.1.5.1, 1.1.5.2, 1.1.5.4, 1.1.5.5, 1.2.6.1, 1.2.6.5, 1.2.6.6, 1.2.6.7

FRONTEND

Login Page

Explanation

On the website's Login Page, users are prompted to give their credentials. This page is completed and all of the relevant requirements are met.

Covered Requirements

1.1.2.1, 1.1.2.3

Home Page

Explanation

This page contains the Navigation Bar and an Activity Stream.

Navigation Bar is partially completed. There are connections to currently active pages (Profile Page, Home Page and notifications.) . Rest of the links will be added as we complete the pages.

Activity Stream is only a bare bone structure. We added it to gain an insight on how the page would look like in the future.

Notification and Activity Stream API's are functional and are ready for use when the necessary content generation is active.

Covered Requirements

1.2.1.1

Profile Page

Explanation

This page contains information about the user such as their research data, followers, their profile photo. There is also an option to edit profiles on this page.

We integrated Google Scholar & Research Gate system. The user can enter their Google Scholar & Research Gate link in the edit Profile section. We then automatically match their research information and add it to their profile. We will implement an authentication for this link in the future.

User's followers and the accounts they follow are displayed on their profile page. Users can check their followers or accounts they follow by clicking on their name. Users can also follow them back or report them to our support team directly on this page.

Covered Requirements

1.1.3.1 , 1.1.5.1 ,1.2.6.4

Register Page

Explanation

This page allows guest users to create an account on our platform.

The required information for an account is email, password, name, surname terms & conditions and their job.

After registering a verification email is sent to the given address. User should verify his account before she has access to our platform.

In the future we are going to add Google Scholar & Research Gate connections to this page. We also will add more skills and job options.

Covered Requirements

1.1.1.1 ,1.1.1.5 ,1.1.1.6

Landing Page

Explanation

This is the page displayed when a guest user enters our website.

In this page users can see Trending Projects and Upcoming Events.

Covered Requirements

1.1.1.5 ,1.1.1.6

ANDROID

Register Page

Explanation

Guest users can provide their credentials and register into the system by accepting our terms and conditions prepared online. According to requirement 1.1.1.1, register should provide some interface that takes affinities, profile photo, GoogleScholar&ResearchGate profile, but we currently do not have support for them.

Covered Requirements

1.1.1.1, 1.1.1.3

Login Page

Explanation

Users can enter the system by providing their emails and passwords. Requirements are fully satisfied

Covered Requirements

1.1.2.1

Profile Page

Explanation

Logged users can view their profiles. Profile has some contents as profile information, researches, etc. They can edit their profile to add Research Gate & Google Scholar accounts, to change publicity of account, etc. Therefore, requirements 1.1.5.1 , 1.1.5.2, 1.1.5.3 are fully satisfied. However, requirement 1.1.5.7 is not satisfied because it is not possible to see comments, ratings and photo currently. Therefore, the profile page as a whole is partially delivered.

Covered Requirements

1.1.5.1, 1.1.5.2, 1.1.5.3, 1.1.5.7

Reset Password Page

Explanation

Users can reset their password by providing their e-mail to the system and then providing the key code that has been sent to their emails with a new password. Requirements are fully satisfied.

Covered Requirements

1.1.2.3

Home Page

Explanation

Logged users can see the home page special for them. They can see trending projects, activity stream and upcoming events there. However, requirements 1.1.1.5,1.2.1.1 and 1.1.1.6 are not fully satisfied. We can say home page as a whole is partially delivered.

Covered Requirements

1.1.1.5, 1.2.1.1, 1.1.16

Landing Page

Explanation

Guest users can see the landing page. In there, they can see trending projects and upcoming events. However, requirements 1.1.1.5 and 1.1.1.6 are not fully satisfied. We can say home page as a whole is partially delivered.

Covered Requirements

1.1.1.5, 1.1.16

GENERAL EVALUATION

We completed the initial phase of our project. Users can now successfully create accounts, verify their accounts, login to the site, reset their password, follow other users and see their followers. This features are the first step in completing our product. We started by these steps because they are essential for all the other parts that are to come. With only these features we were able to make a demonstration of our product to our customer. In the demonstration we showed what we had accomplished so far. We also showed the vision we had for the future of our product. On top of these completed features there are some features we only partially completed due to the time constraint or other blocks on the way. These partially completed features are: - Profile syncing with Google Scholar & Research Gate: A user can currently add her account to their profile with only providing the link to their researcher profile. We will add an identification authentication system. - Profile editing is missing some fields like Skills and Profile Photo editing . Currently these features are implemented with dummy structures.

WORK DONE

SUMMARY OF WORK DONE

BACKEND

TEAM MEMBER	CONTRIBUTION
HALİL UMUT ÖZDEMİR	• Initial Database Design was Done with the Backend Team • Made Research about Flask and Create Initial Project Structure • Created a Base Unit Test Class which Provides a Capability of Unit Tests which uses Database • Implemented Login Mechanism (POST) • Implemented Reset Password Mechanism (GET,POST) • Implemented E-Mail Mechanism of Backend Server • Implemented Notification Mechansim (GET,DELETE) • Implemented a Mechanism that Fetches Reserach Information from Google Scholar(API) and ResearchGate(HTML Parsing) • Implemented a Background Process that will Update GS and RG information of all users in the system • Implemented Ednpoints of Research Information (GET,POST,PUT,DELETE) • Implemented Unit Tests for Each Funtionality that I implemeted • Created a CI/CD Pipeline using Travis-CI (Prepared some scripts to make deplomент easy) • Creaed Dockerfiles of the project with Alperen Divriklioğlu and Ahmet Dadak
HÜSEYİN CAN BÖLÜKBAŞ	• Implemented FollowRequestsAPI(GET, POST, DELETE) • Implemented FollowersAPI(GET) • Implemented FollowingsAPI(GET) • Reviewed merge requests of the backend team.
HİLAL DEMİR	• Implemented, get/post/put/delete endpoints for jobs and skills tables for the application. Initial thought was to fill the database tables from existing apis on the internet but then implementing these endpoints by ourselves so that a user can add, update or delete a job or skill from the database is decided. This way the database updates itself continuously and holds no unnecessary data. • These features are implemented with the Flask framework by Python. MySQL is used for the database. • Implemented unit tests for each of the endpoints. • Kept track of the issues and code reviews from the teammates on the GitHub repository.
ALPEREN DİVRİKLİOĞLU	• created the Dockerfile for the backend app, • created the Docker Composefile for the backend and the database services combined, which later served as the base of the Composefile for the whole project (frontend - backend - database)

	• implemented a RESTful API for the "User" resource (namely, the endpoints for creating, reading, updating and deleting a user) • implemented the endpoint for the user activity stream filler. (we had not implemented the "activity stream" yet but needed an endpoint to return static content for the milestone presentation)

FRONTEND

TEAM MEMBER	CONTRIBUTION
RAMAZAN YURT	• Implemented the “profile page” and the “profile editing page”. Was also responsible for reaching these pages using routing. • Made a color palette selection before the pages were designed with the frontend team. Designed the front pages using a tool called “figma”. Mockups we made last term for inspiration. • Implemented the page that came out during the design phase using JSX because React was decided to be used. MaterialUI components to make the page more beautiful and responsive are used. • Used the documentation prepared by the backend team to access the user information to be displayed on the profile page. HTTP requests are sent by using Axios. Routing part was done with the “link to” feature of react. • Took part in the team coding of the authorization part in react.
AHMET DADAK	• Created a detailed color palette after selecting the color codes with the team. • Created initial designs of web page in “Figma” together with the frontend team. • Implemented “landing page for guest user”, “login page”, “register page”, “forgot password page”, “reset password page”, “account verification page” using ReactJS and MaterialUI design kit. • Implemented the follow mechanism in the profile page and connected the edit profile page to the backend. • Implemented authentication mechanism of the website with frontend team. • Implemented the routing mechanism of the website using React-Router. • Implemented test cases for the components used in the pages using Jest. • Created docker file for React App. • Integrated frontend with TravisCI so that all team members are able to see frontend implementation.
BURHAN AKKUS	• Implemented HomePage InfiniteScroll • Implemented HomePageNotifications

	• Implemented General Layout of Website • Reviewed merge requests of the frontend team.
UMUTCAN UVUT	• Inspected the requirements, diagrams and plan documents for the integration to group and project • Contributed to prototype design • Attended to code review and debug sessions of profile page and landing page with other members • Reviewed merge requests of the fronted team

ANDROID

TEAM MEMBER	**CONTRIBUTION**
BURAK ÖMÜR	• Implement some pages up to the first milestone which are “home page for guests”, “login page”, “register page”, “reset password page” and also their connection and persistence in the android application using kotlin language. • As the Android team, designed pages together to have some understanding between teams using “fluid.ui” website. Pages have been designed using the color gamut which we have decided beforehand with the rest of the team using “coolors.co”. • Arranged the code using the MVP architecture which has been decided by the team. • Implemented the REST API interface using Retrofit2 with a bit of research. • To make the app work smoothly, Implemented RX-Kotlin to use multithreading while connecting to the server and combining it with the current app’s state. • Debugged other pages
ERTUĞRUL BÜLBÜL	• Researched about MVP architecture. • Implemented initial project using MVP architecture. • Design of the pages are done with the Android team. • Researched about Retrofit and its practices. • Implemented the REST API interface using Retrofit. • Implemented home page. • Implemented an interface to provide connection between the repository and presenter class. • Debugged other pages
ÖYKÜ YILMAZ	• Researched about Android apps, MVP structure, and KOTLIN. • Attended a meeting in which initial layouts of the pages, colors that will be used are decided. • Implemented and designed profile page. • Implemented and designed edit profile page. • Implemented and designed following-follower pages using KOTLIN language. • Using MVP architecture, connection with backend is established. • Presented the Android and Frontend scenarios.

CHANGES

CHANGES IN REQUIREMENTS

Since Cmpe 352, there have been small changes in our requirements. They are referred in the table below.

Glossary needed for the table is as follows:

1. Account Information: Name, surname, e-mail and affinities are necessary information. Profile photo, research, ResearchGate, and Google Scholar accounts are optional information.

12. Ongoing State: It is the second of the three states of a project. This is the development phase of the project. After finding enough collaborators, the project will go to the second state which is the Ongoing State.

15. Search for Collaborators State: It is the first of the three states of a project. In this state, the founder of the project waits for requests from other Users and sends invitations to Users.

OLD REQUIREMENT	NEW REQUIREMENT	REASON FOR ALTERING
1.1.1.2 Guests shall be able to register by using their Google accounts.	1.1.1.2 Guests *should* be able to register by using their Google accounts.	This is not an essential requirement since the users are already able to register by providing their account information. It can be implemented if the time allows.
1.1.1.6 Guests shall be able to see the trending projects.	1.1.1.6 Guests *should* be able to see the trending projects.	Trending projects are primary for registered users. So this may be implemented later.
1.1.2.6 Guests shall be able to login by using their Google accounts.	1.1.1.6 Guests *should* be able to login by using their Google accounts.	Since users should be able to register by using their Google accounts, this requirement must be also changed.

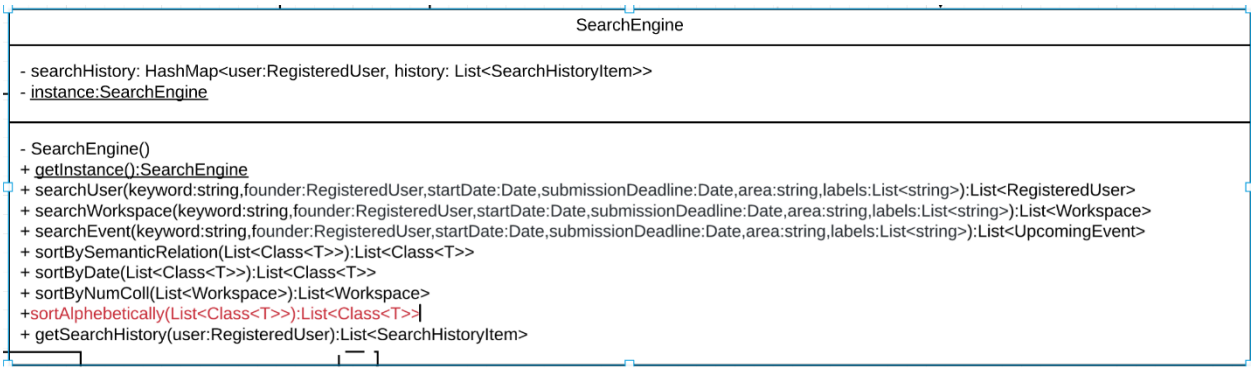
OLD REQUIREMENT	NEW REQUIREMENT	REASON FOR ALTERING
1.1.4.3 Users shall be able to send a request for collaboration to the public workspaces at Search for Collaborators State.	1.1.4.3 Users shall be able to send a request for collaboration to the public workspaces at Search for Collaborators State *or Ongoing State.*	"Users should be able to send a collaboration request for a workspace that they are interested even if it is in the Ongoing States", was the feedback from the customer.
1.1.6.3 Users and guests shall be able to sort search results according to the sorting criteria as date, and number of collaborators needed.	1.1.6.3 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. *1.1.6.4 Users and guests should be able to sort search results of upcoming events according to the sorting criteria as date, alphabetical order.* *1.1.6.5 Users and guests should be able to sort search results of users according to the sorting criteria as date, alphabetical order.*	Sorting is not the priority of our search mechanism. It shall first support semantic search, also there are labels to ease the search of users. Sorting may be implemented at the end. Also since users, workspaces, and upcoming events are of different kind, there should be separate requirements for them.
1.2.1.1 System shall provide a site map to navigate users.	*Removed.*	This is about how, not what. Hence, it must have been removed.
1.2.1.5 System shall provide profile recommendations for new people to follow	1.2.1.3 System *should* provide profile recommendations for new people to follow	The platforms initial aim is to find suitable workspaces, not users. So this may be omitted while implementation. Also there was a typo in the initial requirement, it should have been 1.2.1.4, not 5.
1.2.1.6 The home page shall provide workspace recommendations for users which are at Search for Collaborators according to the requirements explained in 1.2.4	1.2.1.6 *System* shall provide workspace recommendations for users which are at Search for Collaborators according to the requirements explained in 1.2.4	The home page is considered as the how part, system is more convenient for this requirement.
1.2.4.4 The system should send notifications to the user when he/she gets accepted for collaboration from a project.	1.2.4.4 The system *shall* send notifications to the user when he/she gets accepted for collaboration from a project.	Customer specifically asked the requirement to be on the platform.

OLD REQUIREMENT	NEW REQUIREMENT	REASON FOR ALTERING
1.2.4.5 The system should send notifications to the collaborators of the workspace when any application for collaboration is sent to that workspace.	1.2.4.5 The system *shall* send notifications to the collaborators of the workspace when any application for collaboration is sent to that workspace.	Customer specifically asked the requirement to be on the platform.
1.2.4.6 The system should send notifications to the collaborators of the workspace when a deadline for any assigned issue is closed.	*Removed.*	This requirement is considered to be too specific to send notification for by the customer.
2.2.3 System objects shall be encrypted with MD5.	2.2.3 System objects shall be encrypted.	Encryption detailed considered enough for this requirement by the customer.
2.2.4 System shall be backed up to at the end of each day.	2.2.4 System shall be backed up to at the end of each day *according to GMT +3*.	Time zone added to this requirement makes it more clear.
2.5.4 The color designs of the platform will be made to improve the experience of color-blind people.	*Removed*	There are too many versions of color-blindness, so this requirement is removed.

CHANGES IN THE DESIGN DOCUMENTS

CLASS DIAGRAM

Since our requirements have not change drastically, class diagram has not been updated much. The difference can be seen in the picture below, the red-colored function is added to the Search Engine Class.

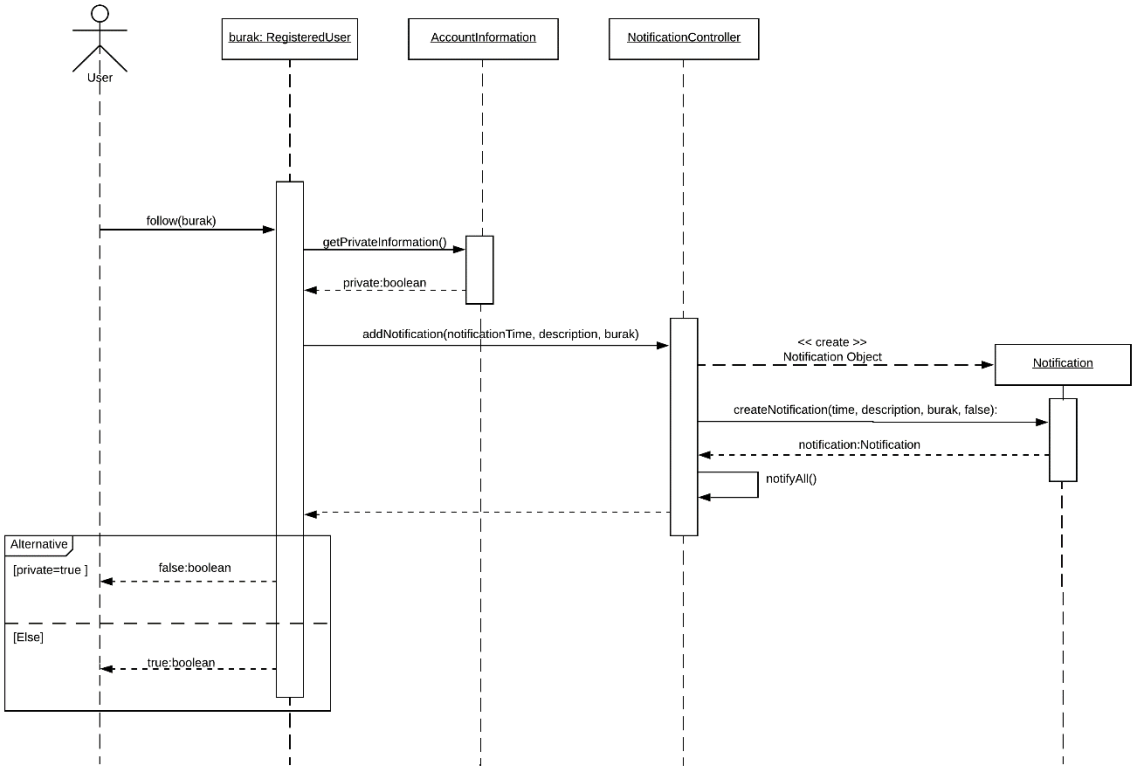


USE CASE DIAGRAM

Because of the same reason mentioned in class diagram, no changes have been made on the Use Case Diagram.

SEQUENCE DIAGRAMS

It has been decided that the sequence diagrams created in April are convenient to keep. Only one of them is removed, due to the consideration that it does not fully represent the notification creation sequence, rather it is specific to the follow request.



REMOVED DIAGRAM

PROJECT PLAN

		Ad	Süre	Balat	Bitirme	Önceki	Kaynak Adlar
1		**Orientation**	**6 günler**	**10.02.2020 17:00**	**18.02.2020 17:00**		
2		Researching about version control system	6 günler	10.02.2020 17:00	18.02.2020 17:00		Burak Ömür;Hasan Ramazan Yurt;Halil Umut Özdemir;Öykü Ylmaz...
3		Researching about GitHub repositories	6 günler	10.02.2020 17:00	18.02.2020 17:00		Ahmet Dadak;Alperen Divrikliolu;Burak Ömür;Erturul Bülbül;Halil...
4		Researching about markdown	6 günler	10.02.2020 17:00	18.02.2020 17:00		Ahmet Dadak;Alperen Divrikliolu;Burak Ömür;Erturul Bülbül;Halil...
5		First team meeting	1 gün	13.02.2020 17:00	14.02.2020 17:00		Ahmet Dadak;Alperen Divrikliolu;Burak Ömür;Erturul Bülbül;Halil...
6		Creating a workspace on Slack	3 günler	13.02.2020 17:00	18.02.2020 17:00		Erturul Bülbül
7		**Documentation**	**5 günler**	**13.02.2020 08:00**	**19.02.2020 17:00**		
8		Creating the wiki page of the team	3 günler	13.02.2020 08:00	17.02.2020 17:00		Burak Ömür
9		Creating the readme.md for the repository	4 günler	13.02.2020 08:00	18.02.2020 17:00		Hasan Ramazan Yurt
10		Preparing communication plan	4 günler	13.02.2020 08:00	18.02.2020 17:00		Ahmet Dadak
11		Preparing favourite github repositories page	5 günler	13.02.2020 08:00	19.02.2020 17:00		Meltem Arslan
12		Preparing own personal wiki pages	5 günler	13.02.2020 08:00	19.02.2020 17:00		Ahmet Dadak;Alperen Divrikliolu;Burak Ömür;Erturul Bülbül;Halil...
13		Preparing version control system research page	5 günler	13.02.2020 08:00	19.02.2020 17:00		Öykü Ylmaz
14		**Requirements**	**13 günler**	**20.02.2020 08:00**	**09.03.2020 17:00**		
15		Requirement engineering	5 günler	20.02.2020 08:00	26.02.2020 17:00		Ahmet Dadak;Alperen Divrikliolu;Burak Ömür;Erturul Bülbül;Halil...
16		Reviewing the requirements before first version	1 gün	28.02.2020 08:00	28.02.2020 17:00		Halil Umut Özdemir;Meltem Arslan;Burak Ömür;Öykü Ylmaz
17		Adding/altering system requirements	3 günler	28.02.2020 08:00	03.03.2020 17:00		Meltem Arslan;Burak Ömür;Ahmet Dadak;Kerem Uslular;Hasan Ra...
18		Adding/altering user requirements and non-functi...	3 günler	28.02.2020 08:00	03.03.2020 17:00		Erturul Bülbül;Öykü Ylmaz;Alperen Divrikliolu;Mehmet Temizel;...
19		Customer meeting #1 for specifying requirements	3 günler	05.03.2020 08:00	09.03.2020 17:00		Halil Umut Özdemir;Kerem Uslular;Ahmet Dadak
20		**Logo and Name**	**4 günler**	**28.02.2020 08:00**	**04.03.2020 17:00**		
21		Preparing logo and name	4 günler	28.02.2020 08:00	04.03.2020 17:00		Ahmet Dadak;Alperen Divrikliolu;Burak Ömür;Erturul Bülbül;Halil...
22		**User Scenarios**	**11 günler**	**28.02.2020 08:00**	**13.03.2020 17:00**		
23		Creating scenario #1	4 günler	10.03.2020 08:00	13.03.2020 17:00	14	Öykü Ylmaz;Hasan Ramazan Yurt;Halil Umut Özdemir
24		Creating scenario #2	4 günler	28.02.2020 08:00	04.03.2020 17:00		Burak Ömür;Alperen Divrikliolu;Erturul Bülbül;Kerem Uslular
25		Creating scenario #3	4 günler	28.02.2020 08:00	04.03.2020 17:00		
26		**Mockups**	**4 günler**	**16.03.2020 08:00**	**19.03.2020 17:00**	**22**	
27		Creating mockup #1	4 günler	16.03.2020 08:00	19.03.2020 17:00		Halil Umut Özdemir;Hasan Ramazan Yurt;Öykü Ylmaz
28		Creating mockup #2	4 günler	16.03.2020 08:00	19.03.2020 17:00		Alperen Divrikliolu;Burak Ömür;Erturul Bülbül;Kerem Uslular
29		Creating mockup #3	4 günler	16.03.2020 08:00	19.03.2020 17:00		
30		**Diagrams**	**11 günler**	**12.03.2020 08:00**	**26.03.2020 17:00**	**14**	
31		Preparing Use Case Diagram	4 günler	12.03.2020 08:00	17.03.2020 17:00		Ahmet Dadak;Burak Ömür;Meltem Arslan;Erturul Bülbül;Alperen ...
32		Preparing Class Diagram	4 günler	12.03.2020 08:00	17.03.2020 17:00		Halil Umut Özdemir;Öykü Ylmaz;Kerem Uslular;Mehmet Temizel;H...
33		Preparing Sequence Diagrams	3 günler	24.03.2020 08:00	26.03.2020 17:00	31;32	Burak Ömür;Hasan Ramazan Yurt;Halil Umut Özdemir;Öykü Ylmaz...
34		**Planning**	**5 günler**	**13.04.2020 13:00**	**20.04.2020 13:00**		
35		Until Planning	4 günler	13.04.2020 13:00	17.04.2020 13:00		Ahmet Dadak;Alperen Divrikliolu

		Ad	Süre	Balat	Bitirme	Önceki	Kaynak Adlar
36		After Planning	4 günler	13.04.2020 13:00	17.04.2020 13:00		Halil Umut Özdemir;Öykü Yılmaz;Meltem Arslan;Kerem Uslular
37		Merge of Plans	1 gün	17.04.2020 13:00	20.04.2020 13:00	35;36	Burak Ömür;Hasan Ramazan Yurt;Mehmet Temizel;Erturul Bülbül
38		Milestone1	0 günler	04.05.2020 17:00	04.05.2020 17:00	26;7;20;22;14;1	Ahmet Dadak;Alperen Divrikliolu;Burak Ömür;Erturul Bülbül;Halil...
39		**API Test & Study**	**12 günler?**	**23.04.2020 08:00**	**08.05.2020 17:00**		
40		API Meeting	1 gün?	23.04.2020 08:00	23.04.2020 17:00		
41		API Usage	5 günler?	23.04.2020 08:00	29.04.2020 17:00		
42		Design of New API	8 günler?	29.04.2020 08:00	08.05.2020 17:00		
43		Milestone2	0 günler	18.05.2020 17:00	18.05.2020 17:00	26;30;38;14;22...	Ahmet Dadak;Alperen Divrikliolu;Burak Ömür;Erturul Bülbül;Halil...
44		Arrange teams	1 gün	27.10.2020 08:00	27.10.2020 17:00		Burak Ömür;Hasan Ramazan Yurt;Halil Umut Özdemir;Öykü Yılmaz...
45		Revise Requirements, Design, Plans	6 günler	27.10.2020 08:00	03.11.2020 17:00		Ahmet Dadak;Alperen Divrikliolu;Burak Ömür;Erturul Bülbül;Halil...
46		**Group Meetings**	**1,833 günler**	**04.11.2020 08:00**	**05.11.2020 15:39**		
47		All members	0,833 günler	04.11.2020 08:00	04.11.2020 15:39		Ahmet Dadak;Alperen Divrikliolu;Burak Ömür;Erturul Bülbül;Halil...
48		Backend	1 gün	04.11.2020 15:39	05.11.2020 15:39	47	Alperen Divrikliolu;Halil Umut Özdemir;Hilal Demir;Hüseyin Can B...
49		Frontend	0,75 günler	04.11.2020 15:39	05.11.2020 13:39	47	Ahmet Dadak;Burhan Can Akku;Hasan Ramazan Yurt;Umutcan ...
50		Android	0,75 günler	04.11.2020 15:39	05.11.2020 13:39	47	Burak Ömür;Erturul Bülbül;Orkan Akisu;Öykü Yılmaz
51		**Pre Implementation**	**2,25 günler**	**05.11.2020 13:39**	**09.11.2020 15:39**		
52		**Backend**	**2 günler**	**05.11.2020 15:39**	**09.11.2020 15:39**		
53		Create Test Server	2 günler	05.11.2020 15:39	09.11.2020 15:39	48	Halil Umut Özdemir;Alperen Divrikliolu;Hüseyin Can Bölükba;Hila...
54		**Frontend**	**1,5 günler**	**05.11.2020 13:39**	**09.11.2020 08:39**		
55		Create Initial Design	1,5 günler	05.11.2020 13:39	09.11.2020 08:39	49	Ahmet Dadak;Hasan Ramazan Yurt;Burhan Can Akku;Umutcan ...
56		**Android**	**0,25 günler**	**05.11.2020 13:39**	**05.11.2020 15:39**		
57		Create Initial Design	0,25 günler	05.11.2020 13:39	05.11.2020 15:39	50	Erturul Bülbül;Burak Ömür;Öykü Yılmaz;Orkan Akisu
58		**Test Case**	**5 günler**	**09.11.2020 15:39**	**16.11.2020 15:39**		
59		Implement Test Cases of Log in-Register and Profi...	5 günler	09.11.2020 15:39	16.11.2020 15:39	53;55;57	Ahmet Dadak;Alperen Divrikliolu;Burak Ömür;Erturul Bülbül;Halil...
60		**Log In - Register**	**5 günler**	**10.11.2020 08:00**	**16.11.2020 17:00**		
61		**Backend**	**5 günler**	**10.11.2020 08:00**	**16.11.2020 17:00**		
62		Implementation of Register System	1 gün	10.11.2020 08:00	10.11.2020 17:00		Alperen Divrikliolu;Halil Umut Özdemir;Hilal Demir;Hüseyin Can B...
63		Implementation of Log In System	2 günler	11.11.2020 08:00	12.11.2020 17:00	62	Alperen Divrikliolu;Halil Umut Özdemir;Hilal Demir;Hüseyin Can B...
64		Reset-Change Password	2 günler	13.11.2020 08:00	16.11.2020 17:00	62;63	Alperen Divrikliolu;Halil Umut Özdemir;Hilal Demir;Hüseyin Can B...
65		**Frontend**	**3 günler**	**10.11.2020 08:00**	**12.11.2020 17:00**		
66		Register Page	3 günler	10.11.2020 08:00	12.11.2020 17:00		Ahmet Dadak;Hasan Ramazan Yurt;Umutcan Uvut;Burhan Can A...
67		Log In Page	3 günler	10.11.2020 08:00	12.11.2020 17:00		Ahmet Dadak;Burhan Can Akku;Hasan Ramazan Yurt;Umutcan ...
68		Reset-Change Password Page	1,5 günler	10.11.2020 08:00	11.11.2020 13:00		Ahmet Dadak;Burhan Can Akku;Hasan Ramazan Yurt;Umutcan ...
69		**Android**	**3 günler**	**10.11.2020 08:00**	**12.11.2020 17:00**		
70		Register Page	3 günler	10.11.2020 08:00	12.11.2020 17:00		Erturul Bülbül;Burak Ömür;Öykü Yılmaz;Orkan Akisu

		Ad	Süre	Balat	Bitirme	Önceki	Kaynak Adlar
71		Log In Page	3 günler	10.11.2020 08:00	12.11.2020 17:00		Burak Ömür;Erturul Bülbül;Orkan Akisu;Öykü Yılmaz
72		Reset- Change Password Page	1,5 günler	10.11.2020 08:00	11.11.2020 13:00		Burak Ömür;Erturul Bülbül;Orkan Akisu;Öykü Yılmaz
73		**Profile Page - User Actions**	**4 günler**	**16.11.2020 08:00**	**19.11.2020 17:00**		
74		**Backend**	**4 günler**	**16.11.2020 08:00**	**19.11.2020 17:00**		
75		Account Information System	3 günler	17.11.2020 08:00	19.11.2020 17:00	62;63;64	Alperen Divrikliolu;Halil Umut Özdemir;Hilal Demir;Hüseyin Can B...
76		User Action - Following System	2 günler	16.11.2020 08:00	17.11.2020 17:00		Alperen Divrikliolu;Halil Umut Özdemir;Hilal Demir;Hüseyin Can B...
77		User Action - Reporting System	3 günler	16.11.2020 08:00	18.11.2020 17:00		Alperen Divrikliolu;Halil Umut Özdemir;Hilal Demir;Hüseyin Can B...
78		Profile Privacy	1 gün	16.11.2020 08:00	16.11.2020 17:00		Alperen Divrikliolu;Halil Umut Özdemir;Hilal Demir;Hüseyin Can B...
79		Linking Google Scholar-ResearchGate Account	3 günler	16.11.2020 08:00	18.11.2020 17:00		Alperen Divrikliolu;Halil Umut Özdemir;Hilal Demir;Hüseyin Can B...
80		**Frontend**	**3 günler**	**16.11.2020 08:00**	**18.11.2020 17:00**		
81		User Profile Page	3 günler	16.11.2020 08:00	18.11.2020 17:00		Ahmet Dadak;Burhan Can Akku;Hasan Ramazan Yurt;Umutcan ...
82		Follower and Following Accounts Pages	1,5 günler	16.11.2020 08:00	17.11.2020 13:00		Ahmet Dadak;Burhan Can Akku;Hasan Ramazan Yurt;Umutcan ...
83		Follow Requests Pages	1,5 günler	16.11.2020 08:00	17.11.2020 13:00		Ahmet Dadak;Burhan Can Akku;Hasan Ramazan Yurt;Umutcan ...
84		Reporting Page	0,75 günler	16.11.2020 08:00	16.11.2020 15:00		Ahmet Dadak;Burhan Can Akku;Hasan Ramazan Yurt;Umutcan ...
85		**Android**	**3 günler**	**16.11.2020 08:00**	**18.11.2020 17:00**		
86		User Profile Page	3 günler	16.11.2020 08:00	18.11.2020 17:00		Burak Ömür;Erturul Bülbül;Orkan Akisu;Öykü Yılmaz
87		Follower and Following Accounts Pages	2,25 günler	16.11.2020 08:00	18.11.2020 10:00		Burak Ömür;Erturul Bülbül;Orkan Akisu;Öykü Yılmaz
88		Follow Requests Pages	0,75 günler	16.11.2020 08:00	16.11.2020 15:00		Burak Ömür;Erturul Bülbül;Orkan Akisu;Öykü Yılmaz
89		Reporting Page	0,75 günler	16.11.2020 08:00	16.11.2020 15:00		Burak Ömür;Erturul Bülbül;Orkan Akisu;Öykü Yılmaz
90		**Testing**	**1 gün**	**16.11.2020 08:00**	**16.11.2020 17:00**		
91		Test Log in-Register and Profile Page	1 gün	16.11.2020 08:00	16.11.2020 17:00	62	Ahmet Dadak;Alperen Divrikliolu;Burak Ömür;Erturul Bülbül;Halil...
92		Internal Milestone 1	0 günler	20.11.2020 17:00	20.11.2020 17:00	58;51;73;60	Ahmet Dadak;Alperen Divrikliolu;Burak Ömür;Erturul Bülbül;Halil...
93		Milestone3	0 günler	24.11.2020 17:00	24.11.2020 17:00	51;58;60;73	Ahmet Dadak;Alperen Divrikliolu;Burak Ömür;Erturul Bülbül;Halil...
94		**Search Engine**	**5,833 günler**	**30.11.2020 08:00**	**07.12.2020 15:39**		
95		**Backend**	**4 günler**	**30.11.2020 08:00**	**03.12.2020 17:00**		
96		Basic User- Workspace- Event Search (Semantic)	2 günler	30.11.2020 08:00	01.12.2020 17:00		Alperen Divrikliolu;Halil Umut Özdemir;Hilal Demir;Hüseyin Can B...
97		Advanced User- Workspace- Event Search (Sem...	3 günler	30.11.2020 08:00	02.12.2020 17:00		Alperen Divrikliolu;Halil Umut Özdemir;Hilal Demir;Hüseyin Can B...
98		Search History	1 gün	03.12.2020 08:00	03.12.2020 17:00	96;97	Alperen Divrikliolu;Halil Umut Özdemir;Hilal Demir;Hüseyin Can B...
99		**Frontend**	**3,75 günler**	**30.11.2020 08:00**	**03.12.2020 15:00**		
100		Basic User- Workspace- Event Search Section	1,5 günler	30.11.2020 08:00	01.12.2020 13:00		Ahmet Dadak;Burhan Can Akku;Hasan Ramazan Yurt;Umutcan ...
101		Advanced User- Workspace- Event Search Section	2,25 günler	30.11.2020 08:00	02.12.2020 10:00		Ahmet Dadak;Burhan Can Akku;Hasan Ramazan Yurt;Umutcan ...
102		Search History Section	1,5 günler	01.12.2020 08:00	02.12.2020 13:00		Ahmet Dadak;Burhan Can Akku;Hasan Ramazan Yurt;Umutcan ...
103		Search Results Page	0,75 günler	03.12.2020 08:00	03.12.2020 15:00		Ahmet Dadak;Burhan Can Akku;Hasan Ramazan Yurt;Umutcan ...
104		**Android**	**4,5 günler**	**30.11.2020 08:00**	**04.12.2020 13:00**		
105		Basic User- Workspace- Event Search Section	1,5 günler	30.11.2020 08:00	01.12.2020 13:00		Burak Ömür;Erturul Bülbül;Orkan Akisu;Öykü Yılmaz

		Ad	Süre	Balat	Bitirme	Önceki	Kaynak Adlar
106		Advanced User- Workspace- Event Search Section	2,25 günler	30.11.2020 08:00	02.12.2020 10:00		Burak Ömür;Erturul Bülbül;Orkan Akisu;Öykü Yılmaz
107		Search History Section	0,75 günler	02.12.2020 08:00	02.12.2020 15:00		Burak Ömür;Erturul Bülbül;Orkan Akisu;Öykü Yılmaz
108		Search Results Page	1,5 günler	03.12.2020 08:00	04.12.2020 13:00		Burak Ömür;Erturul Bülbül;Orkan Akisu;Öykü Yılmaz
109		**Testing**	**0,833 günler**	**07.12.2020 08:00**	**07.12.2020 15:39**		
110		Implement Test Cases of Search Engine	0,833 günler	07.12.2020 08:00	07.12.2020 15:39		Ahmet Dadak;Alperen Divrikliolu;Burak Ömür;Burhan Can Akku;..
111		**Test Case**	**4 günler**	**07.12.2020 08:00**	**10.12.2020 17:00**		
112		Implement Test Cases of Search Engine	4 günler	07.12.2020 08:00	10.12.2020 17:00		Ahmet Dadak;Alperen Divrikliolu;Burak Ömür;Erturul Bülbül;Halil..
113		**Workspace**	**8 günler**	**07.12.2020 08:00**	**16.12.2020 17:00**		
114		**Backend**	**5 günler**	**07.12.2020 08:00**	**11.12.2020 17:00**		
115		Workspace Information System	3 günler	07.12.2020 08:00	09.12.2020 17:00		Alperen Divrikliolu;Halil Umut Özdemir;Hilal Demir;Hüseyin Can B..
116		File Storage System	2 günler	10.12.2020 08:00	11.12.2020 17:00	115	Alperen Divrikliolu;Halil Umut Özdemir;Hilal Demir;Hüseyin Can B..
117		Issue - Milestone System	2 günler	10.12.2020 08:00	11.12.2020 17:00	115	Alperen Divrikliolu;Halil Umut Özdemir;Hilal Demir;Hüseyin Can B..
118		Comment-Rate System	2 günler	10.12.2020 08:00	11.12.2020 17:00	115	Alperen Divrikliolu;Halil Umut Özdemir;Hilal Demir;Hüseyin Can B..
119		Invitation Mechanism	1 gün	10.12.2020 08:00	10.12.2020 17:00	115	Alperen Divrikliolu;Halil Umut Özdemir;Hilal Demir;Hüseyin Can B..
120		**Frontend**	**3 günler**	**07.12.2020 08:00**	**09.12.2020 17:00**		
121		Workspace Pages for all States of Workspaces	3 günler	07.12.2020 08:00	09.12.2020 17:00		Ahmet Dadak;Burhan Can Akku;Hasan Ramazan Yurt;Umutcan ...
122		Edit File Page	1,5 günler	07.12.2020 08:00	08.12.2020 13:00		Ahmet Dadak;Burhan Can Akku;Hasan Ramazan Yurt;Umutcan ...
123		Issue - Milestone Pages	2 günler	07.12.2020 08:00	08.12.2020 17:00		Ahmet Dadak;Burhan Can Akku;Hasan Ramazan Yurt;Umutcan ...
124		Add Comment Rate Sections to Profile Pages	1 gün	07.12.2020 08:00	07.12.2020 17:00		Ahmet Dadak;Burhan Can Akku;Hasan Ramazan Yurt;Umutcan ...
125		Invitation Section of Workspace	2 günler	07.12.2020 08:00	08.12.2020 17:00		Ahmet Dadak;Burhan Can Akku;Hasan Ramazan Yurt;Umutcan ...
126		Add Invitation Section to User Profiles	1,5 günler	07.12.2020 08:00	08.12.2020 13:00		Ahmet Dadak;Burhan Can Akku;Hasan Ramazan Yurt;Umutcan ...
127		**Android**	**3 günler**	**07.12.2020 08:00**	**09.12.2020 17:00**		
128		Workspace Pages for all States of Workspaces	3 günler	07.12.2020 08:00	09.12.2020 17:00		Burak Ömür;Erturul Bülbül;Orkan Akisu;Öykü Yılmaz
129		Edit File Page	1,5 günler	07.12.2020 08:00	08.12.2020 13:00		Burak Ömür;Erturul Bülbül;Orkan Akisu;Öykü Yılmaz
130		Issue - Milestone Pages	0,75 günler	07.12.2020 08:00	07.12.2020 15:00		Burak Ömür;Erturul Bülbül;Orkan Akisu;Öykü Yılmaz
131		Add Comment Rate Sections to Profile Pages	0,75 günler	07.12.2020 08:00	07.12.2020 15:00		Burak Ömür;Erturul Bülbül;Orkan Akisu;Öykü Yılmaz
132		Invitation Section of Workspace	0,75 günler	07.12.2020 08:00	07.12.2020 15:00		Burak Ömür;Erturul Bülbül;Orkan Akisu;Öykü Yılmaz
133		Add Invitation Section to User Profiles	0,75 günler	07.12.2020 08:00	07.12.2020 15:00		Burak Ömür;Erturul Bülbül;Orkan Akisu;Öykü Yılmaz
134		**Testing**	**2 günler**	**15.12.2020 08:00**	**16.12.2020 17:00**		
135		Test Upcoming Events-Workspace	2 günler	15.12.2020 08:00	16.12.2020 17:00		Ahmet Dadak;Alperen Divrikliolu;Burak Ömür;Erturul Bülbül;Halil..
136		**Upcoming Events**	**4 günler**	**21.12.2020 08:00**	**24.12.2020 17:00**		
137		**Backend**	**4 günler**	**21.12.2020 08:00**	**24.12.2020 17:00**		
138		Upcoming Events API	4 günler	21.12.2020 08:00	24.12.2020 17:00		Alperen Divrikliolu;Halil Umut Özdemir;Hilal Demir;Hüseyin Can B..
139		**Frontend**	**1,5 günler**	**21.12.2020 08:00**	**22.12.2020 13:00**		
140		Upcoming Event Page	1,5 günler	21.12.2020 08:00	22.12.2020 13:00		Ahmet Dadak;Burhan Can Akku;Hasan Ramazan Yurt;Umutcan ...

		Ad	Süre	Balat	Bitirme	Önceki	Kaynak Adlar
141		Upcoming Events Calendar Widget	1,5 günler	21.12.2020 08:00	22.12.2020 13:00		Ahmet Dadak;Burhan Can Akku;Hasan Ramazan Yurt;Umutcan ...
142		**Android**	**1,5 günler**	**21.12.2020 08:00**	**22.12.2020 13:00**		
143		Upcoming Event Page	1,5 günler	21.12.2020 08:00	22.12.2020 13:00		Burak Ömür;Erturul Bülbül;Orkan Akisu;Öykü Yılmaz
144		Upcoming Events Calendar Widget	1,5 günler	21.12.2020 08:00	22.12.2020 13:00		Burak Ömür;Erturul Bülbül;Orkan Akisu;Öykü Yılmaz
145		Internal Milestone 2	0 günler	22.12.2020 08:00	22.12.2020 08:00		
146		**Test Case**	**3 günler**	**24.11.2020 08:00**	**26.11.2020 17:00**		
147		Implement Test Cases of Upcoming Events-Works...	3 günler	24.11.2020 08:00	26.11.2020 17:00		Ahmet Dadak;Alperen Divrikliolu;Burak Ömür;Erturul Bülbül;Halil...
148		Milestone4	0 günler	29.12.2020 17:00	29.12.2020 17:00	113;146;136	Ahmet Dadak;Alperen Divrikliolu;Burak Ömür;Erturul Bülbül;Halil...
149		**Test Case**	**3,333 günler**	**31.12.2020 08:00**	**05.01.2021 10:39**		
150		Implement Test Cases of Recommendation Syste...	3,333 günler	31.12.2020 08:00	05.01.2021 10:39		Ahmet Dadak;Alperen Divrikliolu;Burak Ömür;Burhan Can Akku;...
151		**Recommendation System**	**2,25 günler**	**05.01.2021 08:00**	**07.01.2021 10:00**		
152		**Backend**	**0,75 günler**	**05.01.2021 08:00**	**05.01.2021 15:00**		
153		Recommendation System	0,75 günler	05.01.2021 08:00	05.01.2021 15:00		Alperen Divrikliolu;Halil Umut Özdemir;Hilal Demir;Hüseyin Can B...
154		**Frontend**	**0,75 günler**	**05.01.2021 08:00**	**05.01.2021 15:00**		
155		Add Recommendation Section to Profile Pages	0,5 günler	05.01.2021 08:00	05.01.2021 13:00		Ahmet Dadak;Burhan Can Akku;Hasan Ramazan Yurt;Umutcan ...
156		Add Recommendation Section to Workspace Pages	0,75 günler	05.01.2021 08:00	05.01.2021 15:00		Ahmet Dadak;Burhan Can Akku;Hasan Ramazan Yurt;Umutcan ...
157		**Android**	**2,25 günler**	**05.01.2021 08:00**	**07.01.2021 10:00**		
158		Add Recommendation Section to Profile Pages	2,25 günler	05.01.2021 08:00	07.01.2021 10:00		Burak Ömür;Erturul Bülbül;Orkan Akisu;Öykü Yılmaz
159		Add Recommendation Section to Workspace Pages	1,5 günler	05.01.2021 08:00	06.01.2021 13:00		Burak Ömür;Erturul Bülbül;Orkan Akisu;Öykü Yılmaz
160		**Home Page**	**2,25 günler**	**07.01.2021 08:00**	**11.01.2021 10:00**		
161		**Backend**	**0,75 günler**	**07.01.2021 08:00**	**07.01.2021 15:00**		
162		Email Notification System	0,5 günler	07.01.2021 08:00	07.01.2021 13:00		Alperen Divrikliolu;Halil Umut Özdemir;Hilal Demir;Hüseyin Can B...
163		Activity Stream System	0,75 günler	07.01.2021 08:00	07.01.2021 15:00		Alperen Divrikliolu;Halil Umut Özdemir;Hilal Demir;Hüseyin Can B...
164		**Frontend**	**2,25 günler**	**07.01.2021 08:00**	**11.01.2021 10:00**		
165		Home Page	2,25 günler	07.01.2021 08:00	11.01.2021 10:00		Ahmet Dadak;Burhan Can Akku;Hasan Ramazan Yurt;Umutcan ...
166		Add Upcoming Event Calendar Widget	0,75 günler	07.01.2021 08:00	07.01.2021 15:00		Ahmet Dadak;Burhan Can Akku;Hasan Ramazan Yurt;Umutcan ...
167		**Android**	**2,25 günler**	**07.01.2021 08:00**	**11.01.2021 10:00**		
168		Home Page	2,25 günler	07.01.2021 08:00	11.01.2021 10:00		Burak Ömür;Erturul Bülbül;Orkan Akisu;Öykü Yılmaz
169		Add Upcoming Event Calendar Widget	0,75 günler	07.01.2021 08:00	07.01.2021 15:00		Burak Ömür;Erturul Bülbül;Orkan Akisu;Öykü Yılmaz
170		**Testing**	**1 gün**	**07.01.2021 08:00**	**07.01.2021 17:00**		
171		Test Recommendation System and Home Page	1 gün	07.01.2021 08:00	07.01.2021 17:00		Ahmet Dadak;Alperen Divrikliolu;Burak Ömür;Erturul Bülbül;Halil...
172		System Testing	6 günler?	10.01.2021 08:00	18.01.2021 17:00		
173		Milestone5 (Final Milestone)	0 günler	19.01.2021 15:39	19.01.2021 15:39	151;160;94	Ahmet Dadak;Alperen Divrikliolu;Burak Ömür;Burhan Can Akku;...

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, "Login", "Reset Password" and "Create User" functionalities are all located in the "auth_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-registration" from "backend-dev" was created for the implementation of the RESTful API for "User" resource. As another example, "backend-dev-login" was created for the implementation of the login system.

The list of the backend branches

backend-dev

The root backend branch

backend-dev-docker-infrastructure

The branch for the implementation of the Docker infrastructure for the backend

backend-dev-follow

The branch for the implementation of the follow system

backend-dev-jobs/skills

The branch for the implementation of the RESTful API for "Job" and "Skill" resources

backend-dev-login

The branch for the implementation of the login system

backend-dev-notification

The branch for the implementation of the notification system

backend-dev-registration

The branch for the implementation of the RESTful API for "User" resource

backend-dev-research_info

The branch for the implementation of the research information collection system

After a team member finishes their task and wants to push the code on their local machine to our GitHub repository, they push their code to a branch carrying the same name with the branch on their local machine. For example, after the implementation in the "backend-dev-follow" has been finished on one's local, they must push the code to "backend-dev-follow" branch of our GitHub repository. After the push is done, the team member goes to GitHub and opens a pull request for the code to be merged into "backend-dev" branch. In the pull request, the team member asks other team members to review the code and approve the request if the code is well written.

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-homepage

This branch is for our homepage users who are logged in. This page contains a sidebar Navigation Menu and an InfiniteScroller for user's feed. This branch is currently depreciated and it's contents are in frontend-dev-burhan branch.

frontend-dev-profilepage

Profile page, followers page, following page, and edit profile pages implemented. Backend connections of these pages still being implemented on this branch.

frontend-dev-milestone-one

This is the branch that shows what we have done for the milestone one. It is up-to-date with frontend-dev branch.

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 functionalty implemented on this branch.

We plan to create new branches for each of our developer's in our next implementation cycle.

ANDROID

As Android team, we have taken care to use the Git version-control system efficiently in order to do that we have two main branches for our android development which are called android-release, android-dev and other task specific branches. 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. In fact, we were setting the names of task specific branches very well in the beginning, but due to the decreasing time and some problems we had with git, meaningless branch names appeared later, but this was a great lesson for us to use it more correctly for our ongoing development process. We tried to use the hotfix branch, but we cannot say that we use it efficiently until this milestone, but we will start using it effectively up to the next milestone.

Task Specific Branches

android-dev-lrpage:

It was the first branch that we merged in android dev. Task of this branch was login page, register page and forget password page implementation. Also the initial design of the bottom navigation bar is implemented on this branch. Also some UI updates for the pre login page are made.

android-dev-profile-page :

Profile page, followers page, following page, and edit profile pages implemented. Also backend connections of these pages are implemented in this branch.

android-dev-lrfpage :

Rx Kotlin is implemented in order to provide data transfer from repository class to presenter class. Also some small bug fixes are done.

android-dev-profile-request:

Research information request and get user request handled in this task. Also home page activity stream recycler view and its adapter is implemented.

android-dev-presentation

This branch was created due to the problem we encountered in merging with android-dev.

API DOCUMENTATION

Authentication System

Authentication System Endpoints

POST	`/auth_system/login`	Takes Login Credentials as argument and returns a valid token
GET	`/auth_system/reset_password`	Sends a reset password email to the user
POST	`/auth_system/reset_password`	Changes password of the user
GET	`/auth_system/self`	
PUT	`/auth_system/user`	
DELETE	`/auth_system/user`	
GET	`/auth_system/user`	
POST	`/auth_system/user`	

Follow System

Follow System Endpoints

DELETE	`/follow/follow_requests`	Takes the follower_id and following_id as inputs and replies the corresponding Follow Request
GET	`/follow/follow_requests`	Returns a list of dictionaries that contains id, name, surname, e_mail, rate and is_private
POST	`/follow/follow_requests`	Creates FollowRequest record if profile is private, creates Follow record if profile is public
GET	`/follow/followers`	Returns a list of dictionaries with id, name, surname, e_mail, rate and is_private informations
GET	`/follow/followings`	Returns a list of dictionaries with id, name, surname, e_mail, rate and is_private informations

Profile Management Profile Management Endpoints

GET	/profile/front_page	
GET	/profile/notifications	Returns all notifications of the logged in user with related user lists
DELETE	/profile/notifications	Deletes the given notification
PUT	/profile/research_information	Updates a Research Information
DELETE	/profile/research_information	Deletes Research Information
GET	/profile/research_information	Returns all research information of a user
POST	/profile/research_information	Adds new Research Information

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/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*

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/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", "e_mail": "string", "rate": 0, "is_private": true }`
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
job **string** *(formData)*	Job of the user
is_valid **boolean** *(formData)*	The flag that shows whether the user account is activated or not.
is_private **boolean** *(formData)*	The flag that shows whether the user's profile is public or private.
profile_photo **string** *(formData)*	URL of the profile picture of the 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
auth_token * required **string** *(header)*	Authentication token

Responses Response content type application/json

Code	Description
200	*Account information has been successfully updated.*
400	*Missing data fields or invalid data.*
404	*The user is not found.*
500	*The server is not connected to the database.*

GET /auth_system/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", "e_mail": "string", "rate": 0, "is_private": true }`
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.*
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.*

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
profile_photo string *(formData)*	URL of the profile picture of the 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

Responses Response content type: application/json

Code	Description
201	*User has been successfully created.*
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.*

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/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.
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


```
{
  "follow request senders": [
    {
      "id": 0,
      "name": "string",
      "surname": "string",
      "e_mail": "string",
      "rate": 0,
      "is_private": true
    }
  ]
}
```


400	*Input Format Error*
401	*Current user is unauthorized to see the Follow Requests*
404	*Follow Requests List is empty*
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/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.
auth_token * required `string` *(header)*	Authentication Token

Responses Response content type application/json

Code	Description
200	*Followers List is successfully returned* Example Value | Model `{ "followers": [ { "id": 0, "name": "string", "surname": "string", "e_mail": "string", "rate": 0, "is_private": true } ] }`
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.
auth_token * required `string` *(header)*	Authentication Token

Responses Response content type application/json

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

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.*

GET /profile/notifications Returns all notifications of the logged in user with related user lists

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


```
{
  "notification_list": [
    {
      "id": 0,
      "text": "string",
      "link": "string",
      "timestamp": "2020-11-28T14:16:31.336Z",
      "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*

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*

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*

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
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
	
	{
	"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*

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`

CHALLENGES DURING DEPLOYMENT, DOCKERIZING AND CI/CD PROCESSES

The technologies that we used in deployment, dockerizing and CI/CD processes are Docker, Docker Compose, Amazon AWS, and Travis-CI. At the beginning of the project we took an EC2 instance and a MySQL Database instance on Amazon AWS. We used the MySQL database on Amazon AWS for the development and testing. Whereas the EC2 instance that we took is our production server.

During our deployment process, we decided to use Docker and Docker Compose. Our aim is gathering frontend, backend and database servers on a single machine on Amazon. So, by using the Dockerfiles that we created, for backend and frontend a Docker image is created if there are any changes in the source codes in a deployment stage. Also, by using the Docker Compose file that we created, we create an artificial network between the frontend, backend and the MySQL Database Docker containers.

By using the approach that I explained in the second paragraph, a deployment script is created on our production server. This script, firstly, pulls the release branch of our git repository. Then, by using the Docker Compose file that we created, it creates necessary Docker images, runs Docker containers based on those images and creates a virtual network between those containers. To obtain a CI/CD process, we used a system called Travis-CI. The reason why we choose to use Travis-CI is because it gives a free usage for the public GitHub repositories. By using Travis-CI, we created a pipeline that provides us a mechanism that cıntinously deploys our code to our production server. There are 2 stages of ths pipeline. In the first stage, it runs all the unit tests of the backend by using our test database. Also it concurrently tries to build the frontend code. If there are not any errors on the building phase, it runs all the tests of the frontend. This is the end of the first stage. In the case of an existing error during any of the processes in the first stage, it stops its execution and gives an error message to us.

If there is not any error, the process continues to second stage. Second stage is only available for the pull requests and commits that are done to the "release" branch which is the branch that runs on our production server. In the second stage a machine that is allocated from Travis-CI, connects to our production server using and ssh key. Whenever it connects to our production server, it runs the deployment script of us, which automatically deploys our code to our production server.

During this process when met with various problems. In the building stage of this CI/CD pipeline, because we took an Amazon Linux 2 machine from Amazon, sometimes it got hard to install the necessary tools to our production server. In addition to the Operating System compatibility issues, the most

important issue, which we still can not solve, is the volume problem of the Docker Compose. To protect the data that is stored in our database from deployments, we created a constant volume for our MySQL database. But during the development stage we notice that it did not use the same volume after each deployment, which creates so many errors on our production server. Also another problem that we met is some crash problems during unit tests, initially our CI platform uses the database that is running on our production server. During unit tests which creates some unwanted situations on our database, there can be some errors on our database. As a result of this, we decided to use our Amazon MySQL instance for the tests of CI and tests which are done during the development phase.

SCENARIOS IN THE PRESENTATION

ANDROID

Alper Ahmetoglu is a Teaching Asistant in Computer Engineering Department at Bogaziçi University. He is currently not a member of our platform. He has a problem with remembering the deadlines of Conferences and Journal Special Issues, in an effort to fix this problem he decides to give our platform a try.

First, he looks at our website to decide if it is what he is looking for. After seeing our Landing Page and information about upcoming events he decides to register. He registers using his email account. He checks his emails for an account verification link. After verifying his account, he logs in to the system. In his profile page he sees the Google Scholar option and enters his researcher URL. Then he returns back to profile page and sees his google scholar research there.

FRONTEND

Ningchuan Xiao is a Professor of Geography in the Ohio State University. He is one of the oldest members of our platform.2 years ago he found our system through a recommendation from one of his understudies. He registered to our system to find a new research assistant for his ongoing project about a new modeling of the tectonic movements in Paleogenic era. After his completed his project with his new assistant, he continued to use our platform as he was quite the celebrity and he enjoyed the attention. Right now he uses our platform mostly for checking out projects of his understudies and followers. He thinks that by following the latest developments in his field, he can keep his knowledge up to date on the topics that are interesting for him.

Now we will demonstrate a typical use case of Ningchuan Xiao.

Firstly, he opens our website, and is met by the trending projects and upcoming projects parts.He checks out the Trending Projects section of our main page.

After reading about the projects he logs in using his credentials. In the homepage he sees that he has a notification. He clicks on the notification and sees he gained a new follower. He goes to his profile Page and checks his followers. He returns to the homepage and checks out the latest developments.

EVALUATION OF TOOLS AND MANAGING PROJECT

BACKEND

We 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, and Trello as our planning and issue tracking tool. We will evaluate all the tools one by one. We find Discord as useful tools, because it combines the text channels of Slack and the video and voice communication of Zoom. As a result, we decided to continue to use it as our main communication platform.

We selected Python due to its relatively easy syntax and also it is the most widely used programming language in our team. In the case of Flask, there were lots of discussions about whether it is better to use Flask or Django. During this selection period, we thought that Flask provided less functionality but easier to use than Django. And we selected the easier one Flask. But after the first milestone we met with so many problems because of the lack of some functionalities of Flask. For example it does not provide a testing mechanism for unit tests that use database. So for such lack of functionalities, we had to create our own tools that can manage these problems. But, during the period until Customer Milestone 1, we solved such problems. So we decided to continue to use Flask. In the case of programming language, we did not meet with any problems, which causes our decision to continue to use it.

In the case of MySQL, because we used an ORM tool, which is SQLAlchemy, to use our database, we do not use all of the tools that are supplied by MySQL. However, we used the Workbench tool widely, and we did not encounter any problem. Also during the deployment process, we did not encounter any problems except the volume problem which is explained in the Deployment part of the project. As a result, we will continue to use MySQL database as our database server. Also we used 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.

The tool that we use for project management is Trello. We decided to use it as a team, because it provides an ordered, visual, and easy to understand project management system. By using the cards and tables provided by Trello, we firstly divide all the tasks into some independent subparts, then we assign each independent part to one or two members of our backend team. Also we determine the deadlines of all tasks during the task distribution stage by using the dates provided by the project plan. Initially, the presence of Trello decreased our meeting frequency as a backend team. But after a short period of time, we realized that, even if we divided all tasks as individual parts, it delays our tasks. Because there were some issues that we have to consult with each other, even if our tasks were independent from each other. After realizing this fact we increased our meeting frequency, and solved this problem.

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.

Another problem about the project management is the reduced communication between other sub-teams. Because, as a group, we thought that separation of our tasks will provide us a chance to decrease the communication frequency as a whole group. But after a very short time we also realized that we had to make this project as a whole group. At this stage, Discord was very useful for us as a whole group. During the integration stage of backend with android and frontend, by using the Discord server of our group, we worked on our project as some small groups which includes members from each subgroup. As a result we also solved this problem before it grew and reached an insoluble point.

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 tried the code in the development branch, if it works properly we merged it to the release branch which is the branch that will be deployed by Travis-CI to our production server.

As a result, we are satisfied with most of the tools that we used in this period. We had some problems with the tools that we used, but we solved nearly all of them.

FRONTEND

After the front-end team was established, the first thing we did was to learn the skills of the people in the team and the frameworks they used before. As a result, we decided to write the frontend using React. One of the reasons we chose React was that they had a lot of open source codes.

Before starting to prepare the pages in the front-end section, we chose the dominant colors of our application using "color.io". After the selection process, we started to design the pages to be made with our team. We used a tool called figma to make the drawings professional. After our drawings were finished, we shared the pages among ourselves to implement them.

We used React's routing feature to communicate components. When we could reach our pages with URLs, everyone started to code their own page. We used ready-made CSS sources to make the pages look the same even though they are made by different people. We used two libraries specially made for React: React-Bootstrap and MaterialUI.

We first filled our pages with mock data to examine their general appearance. After verifying the suitability of their appearances among us, we made linking work to the backend part. HTTP requests are needed for this job. After examining the documentation created by the backend team, we realized that some fields are missing on our pages. Therefore, we fixed the missing parts on our page. Then, we started sending requests to fill our pages with the information in the database. For this process, we used Axios library for its easy and common use. Eventually, the pages have reached the look we want.

While doing all these operations, we used git for version control. Except for the first creation of the project, everyone opened a separate branch for their part and developed it there. We used Docker to

run the system we built on the server. For our own control and continuous deployment, we used Travis-CI (in development only).

ANDROID

We have used Android Studio as our IDE; Slack, Github repository and Trello as our main processing tools which are very basic tools that should not have been forgotten while developing an android application.

We have decided to use Model-View-Presenter (MVP) architecture at the beginning of implementation and stick to it. However, while developing we have realized that it is quite old, and there are better architectures. We have decided to change the architecture of our android application from MVP to MVVM. This will take some time but we have it on our to-do list.

We have had to use some REST API interface to communicate with the backend. Therefore, our team separately made research. We have come to the decision of using Retrofit2 library as our HTTP client. We are very delighted to use it because it has a lot of documentation and resources on the internet.

We have had to use some kind of parallel processing in order to manage time and have a responsive UI while sending requests to the server. Our team separately researched for it and we have tried to use 3 separate ways to implement it. First one is the simplest and the worst one, to synchronize threads we have used local storage and a thread that checks whether some change happened in that file. As you can see, we had to change it very fast. Second one is implementing a basic HTTPRequestListener to listen for changes in the responses. It worked really well. However, there are some libraries that implement this feature by itself called RX-Java3. We can use either of the latter two, but using the library could be better in the future.

We have used Git very frequently in our development. We have constructed “android-dev”, “android-release” branches as bases. Then, each task has branched from them. We have faced some merge issues, but we have believed that using MVP made our merge task easier. Using MVVM in future will make it easier in our opinion.

CUSTOMER MEETING

LESSONS FROM PRESENTATION

We learned from our presentation experience that all of the team members should be prepared for presenting in case anything goes bad or any breakdown occurs during presentation. Preparation should be done accordingly and scenarios should be planned carefully so that the features implemented can be shown in a right way. Customers should be able to see what is implemented and find the answers that correspond to his needs.

For our case, displaying the features, pages and applications we implemented went well without any error or breakdown. For the future, we think that can improve our presentation by preparing a slide show or any other visual representation.

LESSONS FROM THE CUSTOMER

From our interaction with the customer, we clarify the differences between notification system and activity stream for our application. We learned that activity stream gives generic events like what is happening, area of interest, associated tags with respect to subscribed individuals whereas notification system is very specific to the user itself. We know now that it is possible that they can overlap.

For our application's direction, we decided to implement a paging mechanism for project listing. We learned that when there are lots of projects, the design should be accordingly. We decided to prevent the users linking another individual's Google Scholar link to their profiles. We learned that it can seem like cheating and it can cause reputation problems.

We also get feedback for the user profile page for displaying papers in the project list section, we should fix it for our next steps. Also, we can add interests part for our user profile management page as a suggestion coming from the customer.

Generally, our current status seems like it satisfies the customer. As an answer to a question that is coming from a customer, we can get data from Google Scholar profile and we can update the Google Scholar link and it updates the user's projects accordingly.