# CONTENTS

# Executive Summary

## Summary of project status

First, we took Project Plan as our guide, we separated each task between sub-teams. Each sub-team makes their task division on their own. For the first milestone, the predefined tasks were sign up, validation via email, login, gathering personal info, create project, basic search functionalities on both mobile(Android) and frontend, also related tables in the database created for these at the backend side. Necessary post and get requests implementation held by backend team for these functionalities. In addition to those, deployment of backend and frontend is done with dockerized fashion.

## Changes that are planned for moving forward

Users will be able to edit and delete their projects. Users will be able to invite other users to collaborate their projects, also users can request to collaborate other users' projects, users will be able to accept or reject the invitation and collaboration requests. Search will be improved and support semantic search. Research tags and the Affiliation system will be improved. Following functionality will be implemented and users will be able to see who follows them and who they are following through their profile. Upvoting a user and commenting on a user functionality will become present, also a number of upvotes and comments will be shown on the profile of the user. Implementation of the events page will be done, and upcoming events and deadlines will be shown in this page. Notification for upvotes, comments, invitations and upcoming events will be implemented. Homepage (user feed) will be implemented and the projects and events related to that user will be shown in their homepage. Relation will be decided via previously collaborated projects and followed users. Creation of a project will be improved according to customers comments at Milestone 1. Google Scholar and ResearchGate integration will be done to our system to be able to view more projects of the users. Users will be able to delete their accounts if they do not want to benefit plenty of functionalities of the Akademise.

# Challenges About Dockerizing, Deployment, Ci/Cd

## CI/CD

First of all, we still don't have a CI/CD pipeline implemented to our github repository yet. We aimed to create a deployment pipeline before milestone 1 but we had some challenges implementing it.

Firstly, since we were using AWS to deploy our code, we also tried to create a deployment pipeline with the CodePipeline service AWS provides. It took a lot effort, a lot of rules, connections between services, creation of tags etc. In the end when it was about to be done, service didn't let us create a connection to github. At first we thought this was because of some needed configuration on Github repository and we talked with our teaching asistant to help us with that but still that didn't work. So, we decided to move onto the next CI/CD tool.

Travis was our next option. We connected our github repository with travis, travis.yaml was ready everything was fine. But then travis.yaml was asking for aws secret keys and ids to deploy our code. We had to create a user in AWS for that purpose and use its credentials in the travis.yaml but since we were using AWS Educate, AWS didn't let us create a user since we aren't the root user. We tried our existing student user's credentials but these credentials are not volatile. They were changing periodically with a session and also travis was not able to identify our credentials. Solution was using a non student AWS account but since we had our code deployed and every other configurations set up in the current account, that was the last solution if we didn't have any other.

We actually didn't have enough time to come up with a running CI/CD pipeline under those circumstances before the milestone 1. But we are currently planning to use github actions with either ssh or some http request method to deploy our code when a push or pull request is made on the repository.

## Dockerizing

Docker was a complete new technology for every one of us. But we learned it through the online courses and walkthroughs etc. We found that technology amazing and it was actually pretty easy to set up and use. Without docker it would be actually way harder to make a running served application in the EC2 instance. Docker helped that nicely where all we had to do was create the image and run it.

## Deployment

Deployment to AWS EC2 is done by CodeDeploy service which is provided by AWS. We needed to create an appspec.yml file in the repository to configure the deployment. Connected the CodeDeploy and EC2 instances together, created an application, deployment group and a deployment for that process. After all those complicated document guided processes, all we had to do was give the github commit where we want to take the code from to the CodeDeploy service and it does the rest.

The only challenge was going through all the documents of AWS and other articles about this process where they explain this process. It actually took a considerable amount of time.

# Requirements

**REQUIREMENTS**

**Requirement Changes Report:**

*1-There was an unnecessary registered corporate user - individual user duality, it’s removed. Also we removed Guest User, to keep things simple and flawless. Now, we have one type of registered user.*

*2-There was a duality of Project and Paper. It was creating confusion and redundancy. After talking with the customer, Paper is removed. Now, all the posts created are considered as Projects.*

*3-Password reset feature has been added, since User may forget or want to change his/her password.*

*4-Text editor feature has been added to the system.Thus, made the system more handy.*

*5-Users can make some information on their profile as private, since they may want more privacy*

**Glossary**

1. **User**:A person capable of contributing to or collaborating with others Projects, and has their own profile page. Authentication is required.
2. **Collaborator**: A user that participates in a project.
3. **Profile Page**: Page that users shall be able to share information about themselves.
4. **Events**: The news about upcoming conferences, meetings and important dates.
5. **Profile**: A visual display of personal data associated with a specific user.
6. **Search**: Find semantically similar users and Projects on the context information provided in the semantic tags.
7. **Password**: Consists of at least six characters and at most sixteen characters that are a combination of letters, numbers and special characters.
8. **Projects**: Long-term works that involve milestones, business packages, deadlines.
9. **Basic Search**: Only text matching based search.
10. **Tag**: Small labels that indicate related aspects of a project.
11. **Notification**: Message that informs the user about a specific event.
12. **Recommendation System**: System that recommends contents and projects that users might be interested in.
13. **Conference**: An event where researchers in a field meet, present and discuss their researches.

14. **Academician**: Users that teach or do research in university.
15. **Public**: Projects that can be seen by all users and can be requested to join by all users.
16. **Private**: Projects that can be seen by only collaborators and the authors of them can invite other users.
17. **Author**: Owner of the project.

## Project Requirements

### ***1. Functional Requirements***

.

#### **1.1 *User Requirements***

- 1.1.1 *Registration*
    - 1.1.1.1 Users shall be able to register by providing name, surname, unique email address and password.
    - 1.1.1.2 Users shall validate the email address for completing the registration phase.
    - 1.1.1.3 Users shall specify the research interests by tags.
    - 1.1.1.4 Users shall provide the information about affiliation.
- 1.1.2 Users shall login by email and password provided.
- 1.1.3 Users shall be able to reset their passwords through their associated email.
- 1.1.4 Users shall be able to search and view users, public projects.
- 1.1.5 Users shall be able to view profile pages of authors, and collaborators of projects.
- 1.1.6 Users shall be able to search related information about the upcoming conferences or journal special issues.
- 1.1.7 Users shall be able to send request to join the public projects as collaborators.
- 1.1.8 Users, as the owner of the projects, shall be able to accept or reject requests.
- 1.1.9 Users shall be able to invite any user to their projects.
- 1.1.10 Users shall be able to accept or reject invitations.
- 1.1.11 User shall be able to disable join requests to end call for collaboration phase.
- 1.1.12 Users shall be able to rate and comment other users that their collaborated with during the collaboration and after the project is completed.
- 1.1.13 Users shall be able to collaborate on more than one project.
- 1.1.14 Users shall be able to have more than one project topic posted.
- 1.1.15 Users shall be able to specify a deadline for the project.
- 1.1.16 Users shall not be able to delete a project that is in progress.
- 1.1.17 Users shall be able to log out from his account.
- 1.1.18 Users shall be able to delete his account.
- 1.1.19 Users shall be able to delete or update a comment they made before.
- 1.1.20 Users shall be able to delete or update a rate they gave before.
- 1.1.21 Users shall be able to add additional files to the collaborated projects.
- 1.1.22 Users shall be able to follow other users on the platform.

*-1.1.23 Profile Page*

- 1.1.23.1 Users shall be able to provide information about the research area, recent publications and affiliation.
- 1.1.23.2 Users shall be able to link their Google Scholar or ResearchGate accounts.
- 1.1.23.3 Users shall be able to edit their own profile page and make some personal information private.
- 1.1.23.4 Users shall be able to see invitations that are sent from the other users.
- 1.1.23.5 Users shall be able to see ratings and comments that are made by other users.
- 1.1.23.6 Users shall be able to see their followers.
- 1.1.23.7 Users shall be able to see people who they follow.

*-1.1.24 Creating / Editing Page*

- 1.1.24.1 Users shall be able to provide information about: topic of the research, deadline of submission, milestones, codes, documents, result plots / figures, required skills in order to apply and if the project is funded or not.
- 1.1.24.2 Users shall be able to add a summary(abstract) part.
- 1.1.24.3 Users shall be able to state the type of the content (Project).
- 1.1.24.4 Users shall be able to state whether project is private or public.
- 1.1.24.5 Users shall be able to add collaborators during creation of a project.
- 1.1.24.6 Users shall be able to add tags related to project.
- 1.1.24.7 Collaborators shall be able to track the information shared by other collaborating users.
- 1.1.24.8 Collaborators shall be able to prepare document or article for submission on the platform.

*- 1.1.25 Project Page*

- 1.1.25.1 Users shall be able to create a new project.
- 1.1.25.2 Users shall be able to see the status of their project(s).
- 1.1.25.3 Users shall be able to edit the specifications of their project(s).
- 1.1.25.4 Users shall be able to see their project(s) in progress.
- 1.1.25.5 Users shall be able to edit the specifications of their project(s) in progress.
- 1.1.25.6 Users shall be able to invite another user to collaborate on their own project(s) in progress.
- 1.1.25.7 Users shall be able to accept or reject other users' request to collaborate on their own project

*- 1.1.26 Search Page*

- 1.1.26.1 Users shall be able to search; other users, projects, conferences and journals within the system.
- 1.1.26.2 Users shall not be able to see private projects unless being a contributor of that project.
- 1.1.26.3 Users shall be able to filter search results with regards to the tag, research area, topic, scope and difficulty.
- 1.1.26.4 Users shall be able to find related posts searching related keywords.
- 1.1.26.5 Users shall be able to view abstract and description of journals/projects/conferences related to their activities, profile and interests.

- *1.1.27 Events Page*
  - 1.1.27.1 Users shall be able to see upcoming conferences.
  - 1.1.27.2 Users shall be able to see closing deadlines of joined projects.
  - 1.1.27.3 Users shall be able to travel in the timeline to see past and future events.
  - 1.1.27.4 Users shall be able to view particular event more detailed when clicked/tapped.

**1.2 System Requirements**

- *1.2.1 Search & Recommendation*
  - 1.2.1.1 System shall support basic search of the available content.
  - 1.2.1.2 System shall support semantic search.
  - 1.2.1.3 System shall provide a tagging system.
  - 1.2.1.4 System shall provide a filtering system. This system filters the content such as projects/profiles shown to the user.
  - 1.2.1.5 System shall provide a recommendation mechanism that recommends journals/projects/conferences to the user based on the user activities and profile.

- *1.2.2 Homepage*
  - 1.2.2.1 System shall provide a homepage for each user, showcasing recent and followed users' project(s).
  - 1.2.2.2 System shall provide ongoing activities of the collaborators for each user.

- *1.2.3 Profile Page*
  - 1.2.3.1 System shall provide a profile page for each user, showcasing their interests, past contributions to projects, and their related links (example: Google Scholar).
  - 1.2.3.2 System shall retrieve information of a user from their Google Scholar/ResearchGate pages.

- *1.2.4 Notifications*
  - 1.2.4.1 System shall provide a notification mechanism for informing users about the incoming invitations.
  - 1.2.4.2 System shall provide a notification mechanism for informing users about whether their requests to join a project accepted or rejected.
  - 1.2.4.3 System shall provide a notification mechanism for informing users about whether someone has commented/rated on their profile.

- *1.2.5 Events*
  - 1.2.5.1 System shall provide a page of events for every user.
  - 1.2.5.2 System shall fill this page with the events related to user.

- 1.2.6 System shall provide a text editor for publishing documents and articles.

## 2. Nonfunctional Requirements

*- 2.1 Activity Streams*

- 2.1.1 System shall obey [W3C Standards]
- 2.1.2 System shall follow [W3C Activity Streams Model]

*- 2.2 Availability & Accessibility*

- 2.2.1 The system shall be available as a web application and an Android application.
- 2.2.2 Applications shall be available in English.
- 2.2.3 The system shall notify users in case of a failure.
- 2.2.4 The system shall support Safari, Chrome and Firefox as a browser and shall support Android 6.0 and later versions.

- 2.3 Performance & Scalability

- 2.3.1 The system shall use caching mechanism to reduce response time.
- 2.3.2 The system shall respond to a request at most 15 seconds.
- 2.3.3 The system shall be able to respond up to 100 requests per second.

- 2.4 Security

- 2.4.1 The system shall be protected from unauthorized accesses and attacks to the system and its stored data according to [KVKK], [GDPR].
- 2.4.2 The system shall force users to use strong passwords that contain a lowercase letter, an uppercase letter, a number and a special character and contain 6-16 characters in total.
- 2.4.3 The system shall store only the hashed version of passwords.
- 2.4.4 The system shall provide a validation code for users during registration.
- 2.4.5 The system shall be protected against SQL injection and DDOS attacks.

# Design Documents

## Use Case Diagrams

You can visit and see the wiki page to take a look at new Use Case Diagram.

Use case diagram has updated with following changes:

### Removed

1. Reporting funtionality is removed.
2. Following functionality case is removed.
3. Profile visibility (public/private) is removed.
4. Calendar sync removed.

5. State whether users are currently seeking for new collaborations case is removed.

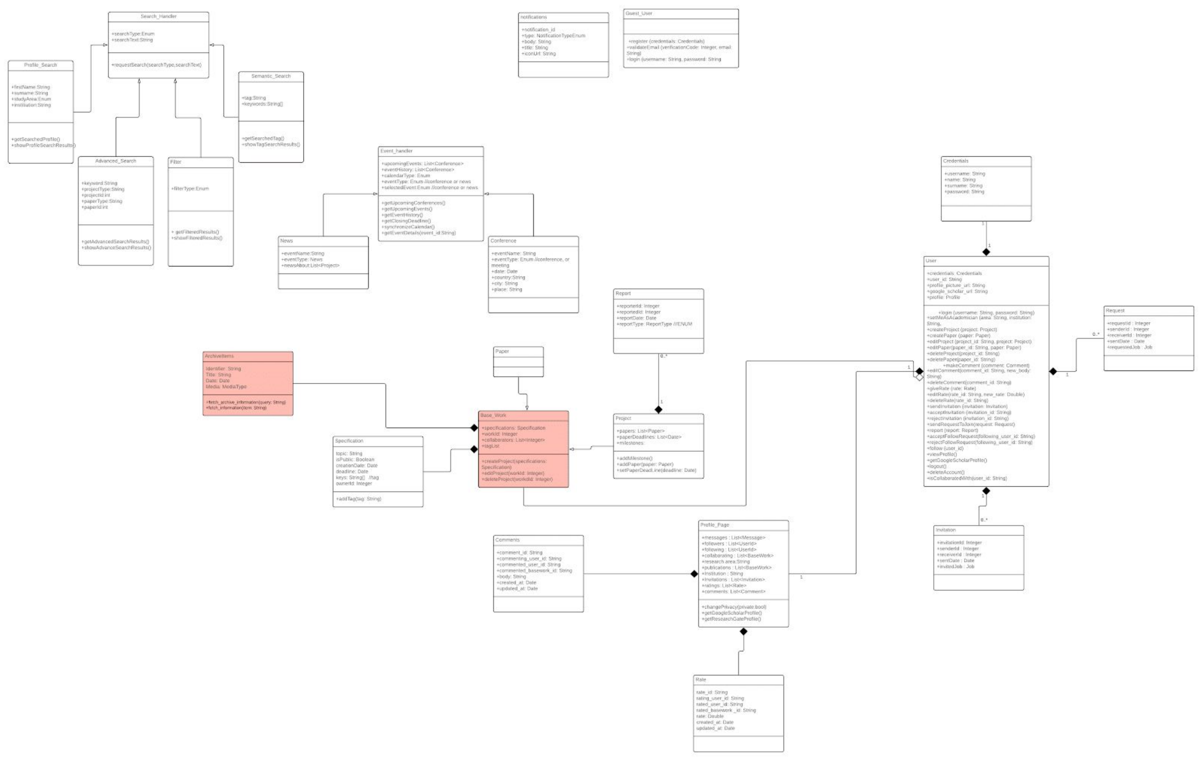
Created

1. Guest user is removed and unregistered user is added.
2. Rate functionality cases are added.

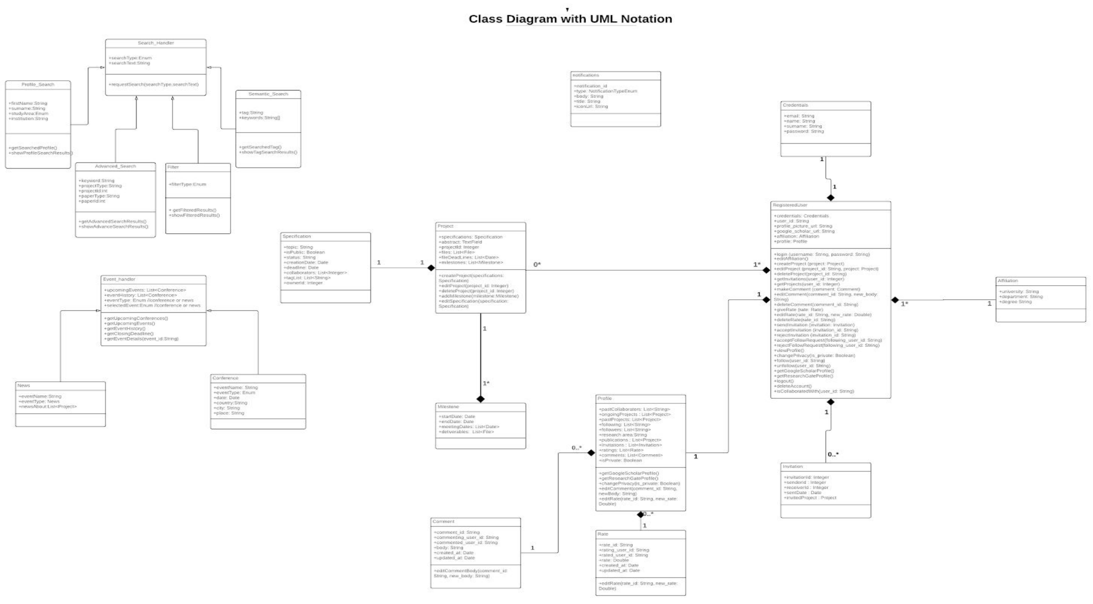
## Class Diagram:

You can see the two class diagrams below, although it cannot be seen since it is quite big, you can see the whole picture more clearly in our Wiki Page.

### The Old Class Diagram



## The New Class Diagram


Class Diagram with UML Notation
Search_Handler
+searchType:Enum
+searchText:String
+requestSearch(searchType,searchText)
Profile_Search
+firstName:String
+surname:String
+studyArea:Enum
+institution:String
+getSearchedProfile()
+showProfileSearchResults()
Semantic_Search
+tag:String
+keywords:String[]
+getSearchedTag()
+showTagSearchResults()
Advanced_Search
+keyword:String
+projectType:String
+projectId:int
+paperType:String
+paperId:int
+getAdvancedSearchResults()
+showAdvanceSearchResults()
Filter
+filterType:Enum
+ getFilteredResults()
+showFilteredResults()
notifications
+notification_id
+type: NotificationTypeEnum
+body: String
+title: String
+iconUrl: String
Credentials
+email: String
+name: String
+surname: String
+password: String
Specification
+topic: String
+isPublic: Boolean
+status: String
+creationDate: Date
+deadline: Date
+collaborators: List<Integer>
+tagList: List<String>
+ownerId: Integer
Project
+specifications: Specification
+abstract: TextField
+projectId: Integer
+files: List<File>
+fileDeadLines: List<Date>
+milestones: List<Milestone>
+createProject(specifications: Specification)
+editProject(project_id: Integer)
+deleteProject(project_id: Integer)
+addMilestone(milestone:Milestone)
+editSpecification(specification: Specification)
RegisteredUser
+credentials: Credentials
+user_id: String
+profile_picture_url: String
+google_scholar_url: String
+affiliation: Affiliation
+profile: Profile
+login (username: String, password: String)
+editAffiliation()
+createProject (project: Project)
+editProject (project_id: String, project: Project)
+deleteProject(project_id: String)
+getInvitations(user_id: Integer)
+getProjects(user_id: Integer)
+makeComment (comment: Comment)
+editComment(comment_id: String, new_body: String)
+deleteComment(comment_id: String)
+giveRate (rate: Rate)
+editRate(rate_id: String, new_rate: Double)
+deleteRate(rate_id: String)
+sendInvitation (invitation: Invitation)
+acceptInvitation (invitation_id: String)
+rejectInvitation (invitation_id: String)
+acceptFollowRequest(following_user_id: String)
+rejectFollowRequest(following_user_id: String)
+viewProfile()
+changePrivacy(is_private: Boolean)
+follow(user_id: String)
+unfollow(user_id: String)
+getGoogleScholarProfile()
+getResearchGateProfile()
+logout()
+deleteAccount()
+isCollaboratedWith(user_id: String)
Affiliation
+university: String
+department: String
+degree String
Event_handler
+upcomingEvents: List<Conference>
+eventHistory: List<Conference>
+eventType: Enum //conference or news
+selectedEvent:Enum //conference or news
+getUpcomingConferences()
+getUpcomingEvents()
+getEventHistory()
+getClosingDeadline()
+getEventDetails(event_id:String)
News
+eventName:String
+eventType: News
+newsAbout:List<Project>
Conference
+eventName: String
+eventType: Enum
+date: Date
+country:String
+city: String
+place: String
Milestone
+startDate: Date
+endDate: Date
+meetingDates: List<Date>
+deliverables: List<File>
Profile
+pastCollaborators: List<String>
+ongoingProjects : List<Project>
+pastProjects: List<Project>
+following: List<String>
+followers: List<String>
+research area:String
+publications : List<Project>
+invitations : List<Invitation>
+ratings: List<Rate>
+comments: List<Comment>
+isPrivate: Boolean
+getGoogleScholarProfile()
+getResearchGateProfile()
+changePrivacy(is_private: Boolean)
+editComment(comment_id: String, newBody: String)
+editRate(rate_id: String, new_rate: Double)
Invitation
+invitationId: Integer
+senderId : Integer
+receiverId : Integer
+sentDate : Date
+invitedProject : Project
Comment
+comment_id: String
+commenting_user_id: String
+commented_user_id: String
+body: String
+created_at: Date
+updated_at: Date
+editCommentBody(comment_id: String, new_body: String)
Rate
+rate_id: String
+rating_user_id: String
+rated_user_id: String
+rate: Double
+created_at: Date
+updated_at: Date
+editRate(rate_id: String, new_rate: Double)
1
1
0*
1*
1
1*
1
1
1
1*
1
1
0..*
0..*
1
0..*
1
1

### Changes

1. The class ‘Guest User’ is removed: Guest User requirement was removed from the Requirements.
2. The fields ‘Collaborators’ and ‘tagList’ have been added to the class ‘Specification’: It is now easier to manipulate, and more modular.
3. The class ‘Report’ has been removed: As it was decided by the team, the system no longer supports a report functionality.
4. The classes ‘BaseWork’ and ‘Paper’ have been removed. As it was suggested by the Instructor and TAs, we have only one class, ‘Project’.
5. The class ‘Project’ now has the following fields: specifications, abstract, projectId, files, fileDeadlines,
6. The class ‘Milestone’ has been created: It was missing.

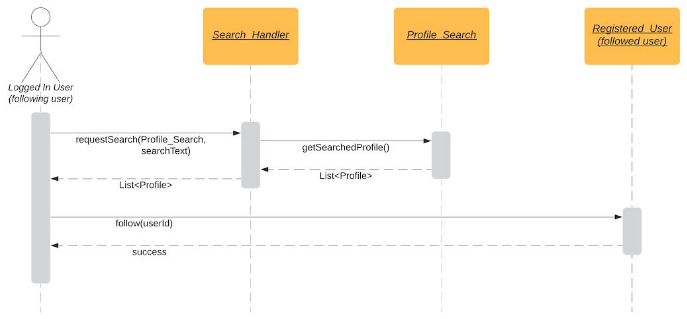
7. The field ‘messages’ is removed from the class ‘Profile’: The system no longer supports such a system, as it was decided by the team.
8. The class ‘request’ has been removed: It was the same as the class ‘Invitation’.
9. The class ‘Affiliation’ has been added: It was our design choice.
10. The class ‘ArchiveItems’ has been removed: It was unnecessary.
11. The functions ‘addMilestone’ and ‘editSpecification’ have been added: The reason for the addition of the first class is that we now have a class named ‘Milestone’, hence the function. The reason for the second one is that the function was missing.
12. The class ‘Milestone’ has no functions now: It was unnecessary to have a function in this class.
13. The function ‘editCommentBody’ is added to the class ‘Comment’: It was missing.
14. The field ‘commented_basework_id’ has been removed from the class ‘Comment’: We have no class named ‘BaseWork’ and also users now can comment on profiles, not on projects.
15. The function ‘editRate’ is added to the class ‘Rate’. It was missing.
16. The field ‘commented_basework_id’ has been removed from the class Rate: We have no class named ‘BaseWork’ and also users now can give rates to profiles, not to projects.
17. The function ‘setMeAsAcademician’ has been removed from the class ‘User’, which was previously named ‘RegisteredUser’: Users now can set their degree in the Affiliation part.
18. All the paper-related functions have been removed from the class ‘User: The system has no such thing as paper.
19. All the request-related functions have been removed from the class ‘User’: As explained above, it was redundant to have a class ‘Request’ while we have ‘Invitation’.
20. The function report has been removed from the class ‘User: We have no such functionality now.
21. The function ‘changePrivacy’ is added to the class ‘User’: It was missing.
22. The function ‘getResearchGateProfile’ has been added to the class ‘User’: It was missing.

## Sequence Diagrams

All sequence diagrams are revised in order to match the same visual standards (size, spacing, fontsize, etc.). Technical specifications are mentioned under the related sections.

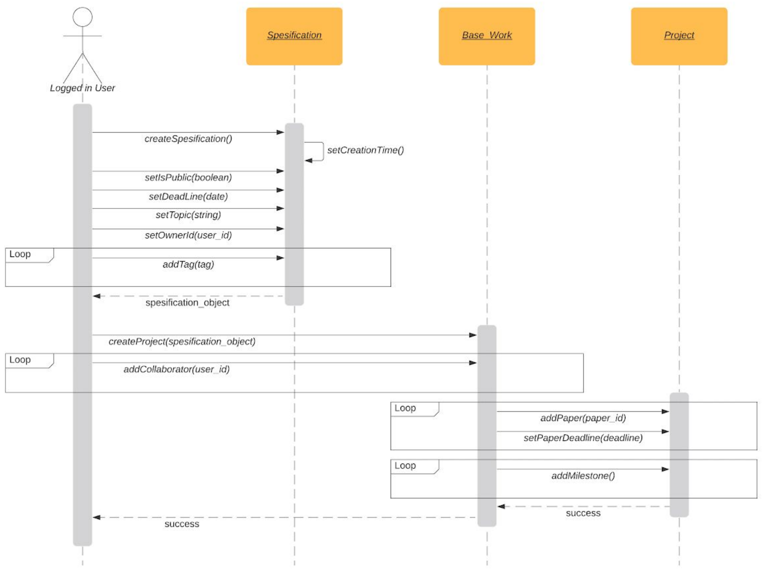
### Unchanged

1. *Follow a User*
   User first searches for a user profile and follows it.

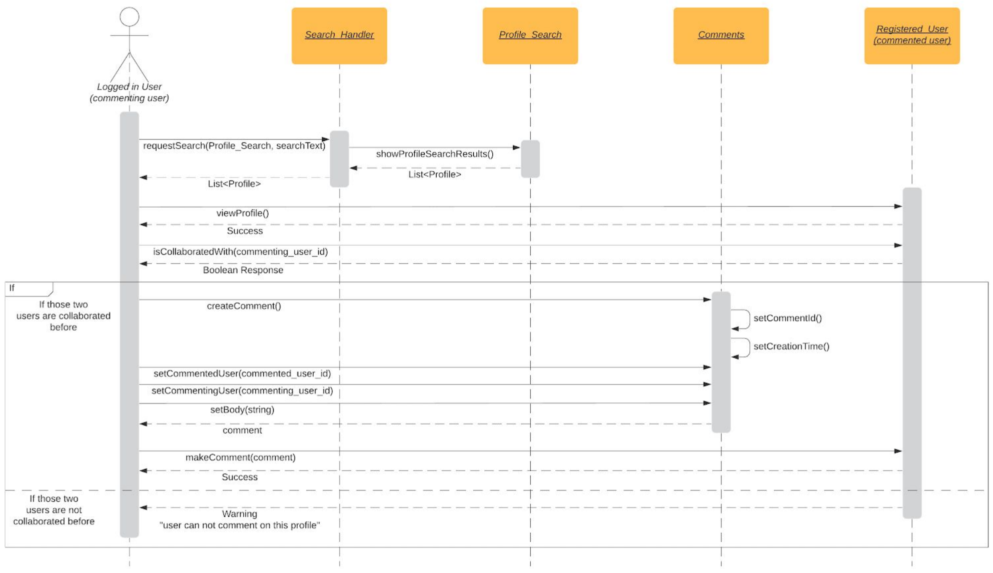


Logged In User (following user)
Search_Handler
Profile_Search
Registered_User (followed user)
requestSearch(Profile_Search, searchText)
getSearchedProfile()
List<Profile>
List<Profile>
follow(userId)
success

2. *Create Project*
   User first determines the specifications of the project and adds collaborators to it. And determines milestones, after that project creation is complete.



3. *Make Comment*
   User searches for another user and views their profile. If they are collaborated before, it creates a comment object and successfully creates its comment. Otherwise, there is a warning occurs



Logged in User (commenting user)
Search_Handler
Profile_Search
Comments
Registered_User (commented user)
requestSearch(Profile_Search, searchText)
showProfileSearchResults()
List<Profile>
List<Profile>
viewProfile()
Success
isCollaboratedWith(commenting_user_id)
Boolean Response
If
If those two users are collaborated before
createComment()
setCommentId()
setCreationTime()
setCommentedUser(commented_user_id)
setCommentingUser(commenting_user_id)
setBody(string)
comment
makeComment(comment)
Success
If those two users are not collaborated before
Warning "user can not comment on this profile"

## Removed

1. *Be Private* diagram is removed.

   Since the private profile option is removed from the requirements, there is no such sequence anymore.

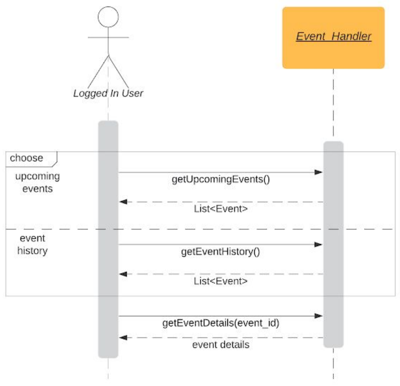
2. *File Complaints* diagram is removed.

   Filing complaints feature is removed from requirement because it would make the project unnecessarily complex. So this sequence diagram is removed as well.

## Revised

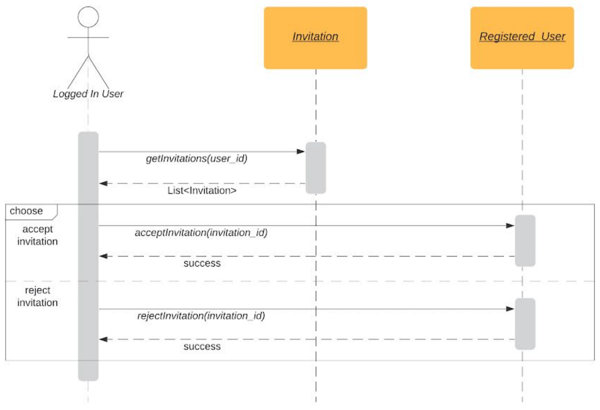
1. *See Upcoming Events in Detail* diagram is changed into *See Upcoming/Former Events in Detail* diagram.

   Former version of this diagram was very naive, therefore it is reimplemented with a bit more sophisticated approach. In the former implementation users can only display upcoming events, however the new version allows users to display former events as well.



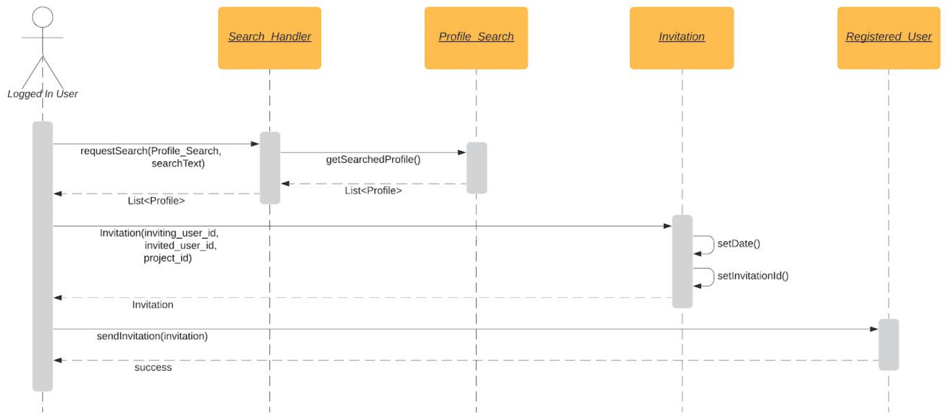
2. *Reject Collaboration Invitation* diagram changed into *Accept/Reject Collaboration Invitation* diagram.

   Accept and reject collaboration invitation sequence diagrams are demonstrated on the same diagram. In the former version, rejecting collaboration invitations was a diagram by itself.



3. *Send Collaboration Invitation* diagram is recreated.

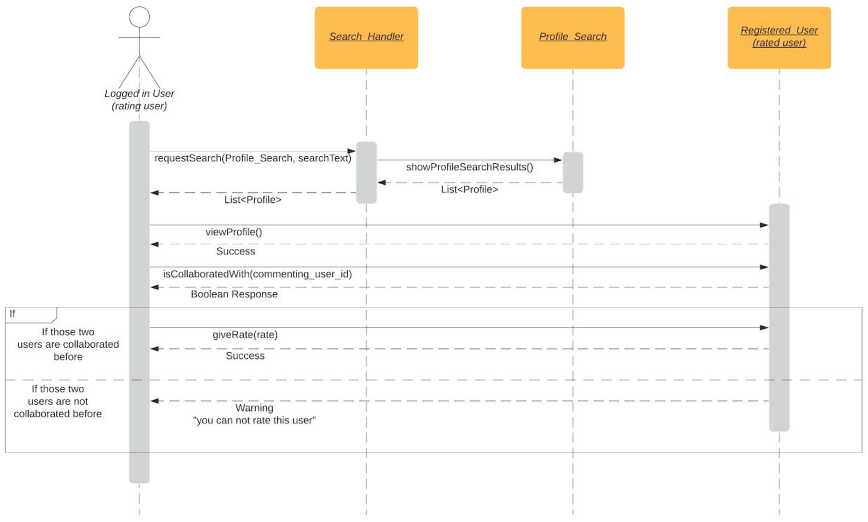
   Former implementation was naive. So it is changed to become more sophisticated and coherent with other sequence diagram implementations. There were two functions (Requests and Invitation) present in the class diagram after this issue was resolved. This diagram is corrected accordingly.



## Created

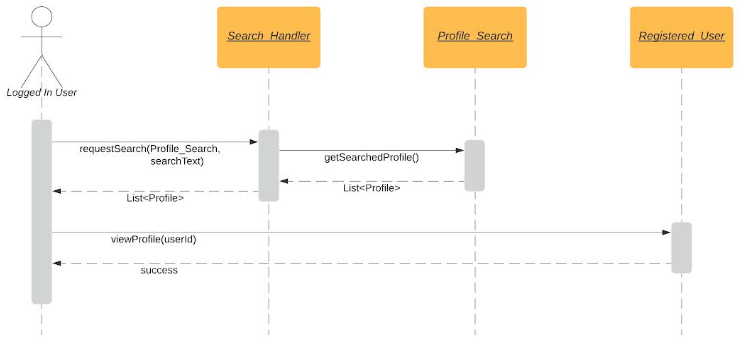
1. *Rate User* diagram is created.

   In this sequence diagram, a user navigates to a profile and rates the related user. Users first search for a profile and view it. If a user has collaborated with this user before they can rate the user. If that is the case, the rating is successful.


Logged in User (rating user)
Search_Handler
Profile_Search
Registered_User (rated user)
requestSearch(Profile_Search, searchText)
showProfileSearchResults()
List<Profile>
List<Profile>
viewProfile()
Success
isCollaboratedWith(commenting_user_id)
Boolean Response
If
If those two users are collaborated before
giveRate(rate)
Success
If those two users are not collaborated before
Warning "you can not rate this user"

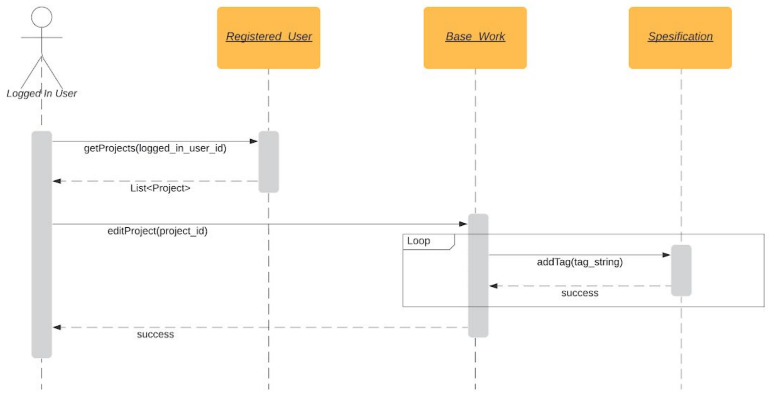
2. *Search and View Profile* diagram is created.

   In this sequence diagram, how a user can search and display a user's profile is demonstrated. Users first search for a profile. Then basically display the desired one among the search results.



3. *Add Tags to a Project* diagram is created.

   In this sequence diagram, how a user can search and display a user's profile is demonstrated. Users first select the project that they want to add tags and add the tags they want.



# API Documentation

**Allowed HTTPs requests:**

**POST:** To create a resource
**PATCH:** To update a resource
**GET:** To get a resource or list of resources
**DELETE:** To delete a resource

**Description of Usual Server Responses:**

**200:** OK => the request successfully handled
**201:** CREATED => the request successfully handled and a resource created
**204:** NO CONTENT => the request successfully handled and there is nothing to return
**400:** BAD REQUEST => the request could not be understood. (missing parameters)

**401:** UNAUTHORIZED => authentication failed. user does not have permissions.
**404:** NOT FOUND => resource was not found.
**500:** INTERNAL SERVER ERROR => something went wrong in the server.

**Endpoints:**

**/auth/login**
**Request type:** POST
**Required parameters:** email type of String, password type of String.
**Responses:**
**1-)** code: 200, message: “Success”, body: {authorization token, userId}.
**2-)** code: 401, message: “Wrong password”
**3-)** code: 404, message: “User not found”

**Description:** /auth/login endpoint is for registered users who want to gain authorization using their credentials. The user sends their credentials in the body of this post request. First, their email is checked for validity, then the hash of their password is compared with the one with the database. If the password matches, response 1 is returned. If the email address is valid but the password does not match, response 2 returned. If the email address has no user registered via itself, response 3 returned.

**/auth/signup**
**Request type:** POST
**Required parameters:** email type of String, password type of String, name type of String, surname type of String.
**Responses:**
**1-)** code: 201, message: “User created”, body: {authorization token, userId}.
**2-)** code: 400, message: “This email address is already used”
**3-)** code: 400, message: “Something went wrong”

**Description:**/auth /signup endpoint is for those who want to register to the application. The user sends their email address, password, name and surname in the body of this post request. First, their email is checked to know if the email address is in use or not, then their password is hashed and an authorization token is created. After that the user is saved to the database and a validation code is sent to their email address. If no errors are thrown in the processes above, response 1 is returned. If the given email address is in use, response 2 is returned. For any other errors that can be thrown at the server side, response 3 is returned.

**/auth/jwt**
**Request type:** POST
**Required parameters:** token type of String
**Responses:**
**1-)** code: 200, message: “Token is valid”, body: authorization token.
**2-)** code: 400, message: “Token is invalid”

**Description:** /auth/jwt endpoint is for the users who have an authorization token in their hand and want to check if they can continue being authorized with that token. The user sends their current authorization token in the body of this post request. The given token is tried to be verified. If the token has not expired and is still the valid token for the user, response 1 is returned. If the token has expired or does not match with the valid one, response 2 is returned.

**/validate**
**Request type:** POST
**Required parameters:** validation code type of String
**Header:** Bearer token for authorization
**Responses:**
**1-)** code: 200, message: “Validation is successful”.
**2-)** code: 400, message: “Validation code is wrong”

**Description:** /validate endpoint is for the users who have registered to the app but are yet to validate their email addresses. The user sends the validation code sent to their email address in the body of this post request. The given code is compared with the one in the database. If the code matches, response 1 is returned. If the code does not match, response 2 is returned.

**/search?query&type**
**Request type:** GET
**Required parameters:** query type of String, type type of Integer
**Header:** Bearer token for authorization
**Responses:**
**1-)** code: 200, message: “Users are fetched”, body: list of users.
**2-)** code: 200, message: “Projects are fetched”, body: list of projects.
**2-)** code: 500, message: “Something went wrong”

**Description:** /search endpoint is for the authorized users who want to search existing users or projects with the keyword they provide. The user sends the search keyword and search type the user wants to execute in the url parameters of this get request. If the type is 0 which means that the user wants to search for users, and if the type is 1 which means that the user wants to search for projects. If users are searched, their first name and last name are concatenated and the execution of search is based on this query: **%query& isLike fullname** If projects are searched, the execution of search is based on this query: **%query& isLike title.** If no errors are thrown in the process above, response 1 or 2 is returned based on the search type. If any error is thrown, response 3 is returned.

**/profile/update**
**Request type:** PATCH
**Required parameters:** interests type of list of strings, or affiliations type of list of strings, or both.
**Header:** Bearer token for authorization
**Responses:**
**1-)** code: 200, message: “Successful”
**2-)** code: 500, message: “Something went wrong”

**Description:** /profile/update endpoint is for the authorized users who want to update a portion of their profile information. The portion is not necessarily the entire profile information. The user sends a new set of their interests or affiliations or both in the body of this patch request. The old data is removed from the database and new data is saved to the database. If no errors are thrown in the process above, response 1 is returned. If any error is thrown, response 2 is returned.

**/profile/update**
**Request type:** POST
**Required parameters:** interests type of list of strings, affiliations type of list of strings
**Header:** Bearer token for authorization
**Responses:**
**1-)** code: 200, message: “Successful”
**2-)** code: 500, message: “Something went wrong”

**Description:** /profile/update endpoint is for the authorized users who want to add information to their profile. The user sends a new set of their interests and affiliations in the body of this post request. The data is saved to the database. If no errors are thrown in the process above, response 1 is returned. If any error is thrown, response 2 is returned.

**/profile/{userId}**
**Request type:** GET
**Required parameters:** userId type of String
**Header:** Bearer token for authorization
**Responses:**
**1-)** code: 200, message: “Successful”, body: user profile.
**2-)** code: 500, message: “User not found”

**Description:** /profile/{userId} endpoint is for the authorized users who want to fetch the profile information of the user whose id is given by themselves. The user sends the userId of the user whose profile the user wants to fetch in the url parameters of this get request. First check is to find whether the userId matches with any user. If there is a match, response 1 with the profile information of the matched user in its body is returned. If there is no match, response 2 is returned.

**/post/add**
**Request type:** POST
**Required parameters:** title type of String, abstract type of String, privacy type of Boolean, deadline type of String, requirements type of String, tags type of list of strings, collaborators type of integers, uploaded files.
**Header:** Bearer token for authorization
**Responses:**
**1-)** code: 201, message: “Post is created” body: postId.
**2-)** code: 500, message: “Something went wrong”

**Description:** /post/add endpoint is for the authorized users who want to create a new project with the fields given by themselves. The user sends all required parameters in the body of this post request. All information is saved to the relevant database tables. If no errors are thrown in the process above, response 1 is returned. If any error is thrown, response 2 is returned.

**/post/update/{postId}**
**Request type:** PATCH
**Required parameters:** postId type of Integer, or title type of String, or abstract type of String, or privacy type of Boolean, or deadline type of String, or requirements type of String, or tags type of list of strings, or collaborators type of integers or uploaded files, or any combination of previous fields.
**Header:** Bearer token for authorization
**Responses:**
**1-)** code: 200, message: “Post is updated” body: postId.
**2-)** code: 500, message: “Something went wrong”

**Description:** /post/update/{postId} endpoint is for the authorized users who want to update their projects with the fields given by themselves. The user sends all fields they want to update in the body of this post request. The id of the post to be updated is sent in the url parameter. All information is saved to the relevant database tables. If no errors are thrown in the process above, response 1 is returned. If any error is thrown, response 2 is returned.

**/post/delete/{postId}**
**Request type:** DELETE
**Required parameters:** postId type of Integer
**Header:** Bearer token for authorization
**Responses:**
**1-)** code: 200, message: “Post is deleted”.
**2-)** code: 500, message: “Something went wrong”

**Description:** /post/delete/{postId} endpoint is for the authorized users who want to delete their own project whose id is given by themselves. The user sends the id of the post to be deleted in the url parameter of this delete request. The post is found and deleted from the database. If no errors are thrown in the process above, response 1 is returned. If any error is thrown, response 2 is returned.

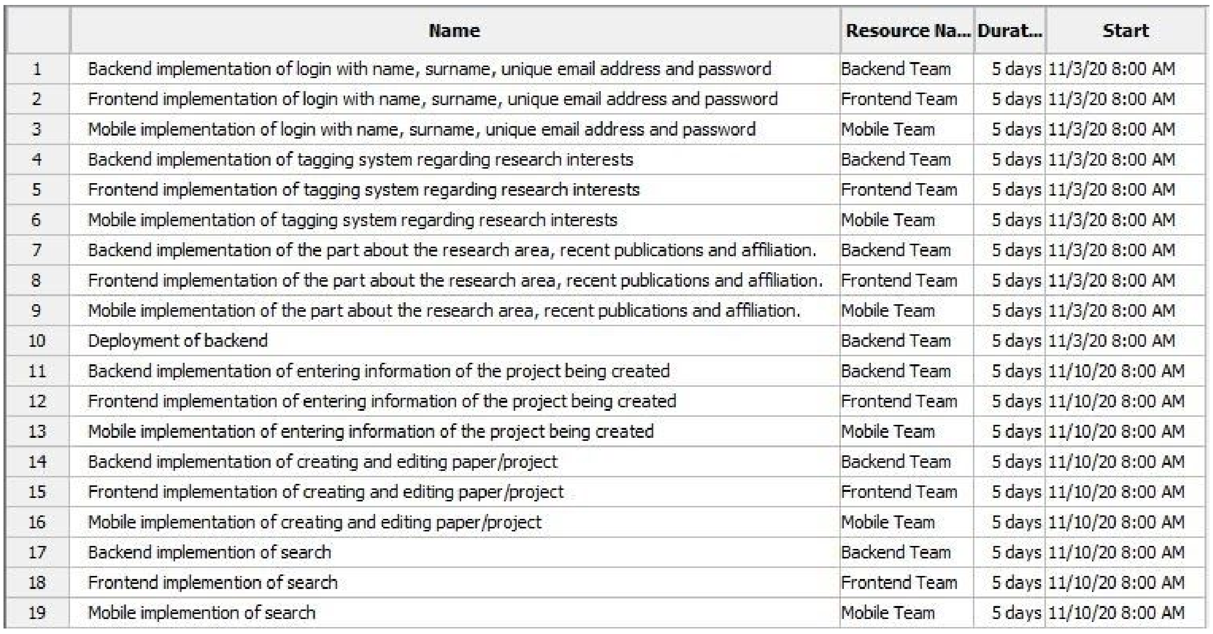
**/post/get/{userId}**
**Request type:** GET
**Required parameters:** userId type of Integer
**Header:** Bearer token for authorization
**Responses:**
**1-)** code: 200, message: “Posts are fetched”, body: list of posts.
**2-)** code: 500, message: “Something went wrong”

**Description:** /post/get/{userId} endpoint is for the authorized users who want to fetch all posts of a specific user whose id is given by themselves. The user sends the id of the user whose posts are gonna be fetched in the url parameter of this get request. The posts are found fetched from the database. If no errors are thrown in the process above, response 1 is returned. If any error is thrown, response 2 is returned.

## Project Plan

Firstly, tasks were defined according to the Requirements. Then the tasks were grouped and sorted according to their importance, and the milestones. Because the project plan was prepared before we got the milestone dates, we again decided which task to do before the Milestone 1 and opened an issue . We discussed with the assistant and informed him about which tasks are planned before the Milestone 1. He confirmed our plan. The plan was to accomplish **the first 19 tasks before the Milestone 1.**

We also decided to assign tasks to groups instead of a specific member. We gave freedom to each group to decide how to do a given task. It helped us to gain a reasonable time during meetings and implementation because we did not discuss as a whole team which tool to use for backend related task.

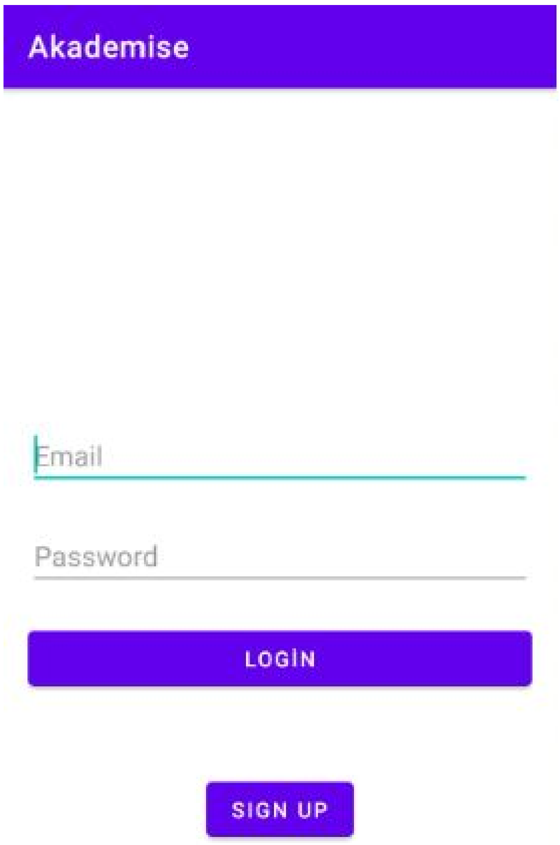


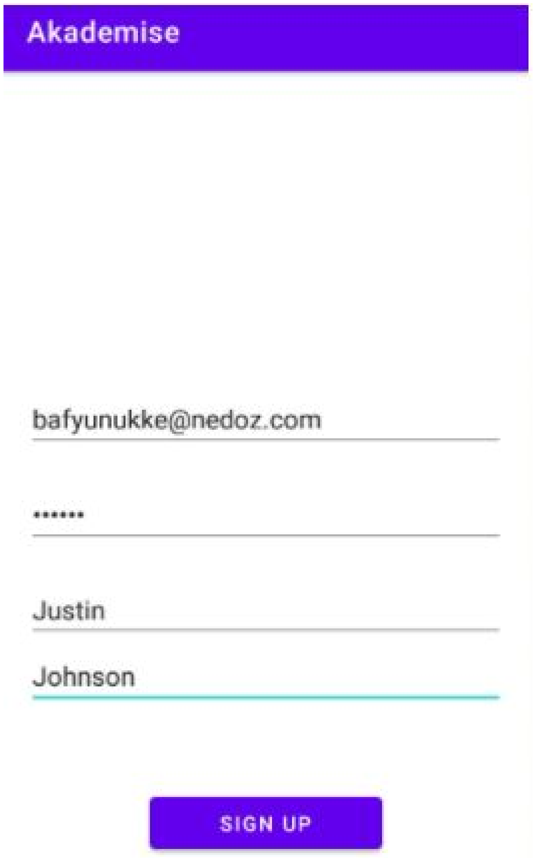
| | Name | Resource Na... | Durat... | Start |
|---|---|---|---|---|
| 1 | Backend implementation of login with name, surname, unique email address and password | Backend Team | 5 days | 11/3/20 8:00 AM |
| 2 | Frontend implementation of login with name, surname, unique email address and password | Frontend Team | 5 days | 11/3/20 8:00 AM |
| 3 | Mobile implementation of login with name, surname, unique email address and password | Mobile Team | 5 days | 11/3/20 8:00 AM |
| 4 | Backend implementation of tagging system regarding research interests | Backend Team | 5 days | 11/3/20 8:00 AM |
| 5 | Frontend implementation of tagging system regarding research interests | Frontend Team | 5 days | 11/3/20 8:00 AM |
| 6 | Mobile implementation of tagging system regarding research interests | Mobile Team | 5 days | 11/3/20 8:00 AM |
| 7 | Backend implementation of the part about the research area, recent publications and affiliation. | Backend Team | 5 days | 11/3/20 8:00 AM |
| 8 | Frontend implementation of the part about the research area, recent publications and affiliation. | Frontend Team | 5 days | 11/3/20 8:00 AM |
| 9 | Mobile implementation of the part about the research area, recent publications and affiliation. | Mobile Team | 5 days | 11/3/20 8:00 AM |
| 10 | Deployment of backend | Backend Team | 5 days | 11/3/20 8:00 AM |
| 11 | Backend implementation of entering information of the project being created | Backend Team | 5 days | 11/10/20 8:00 AM |
| 12 | Frontend implementation of entering information of the project being created | Frontend Team | 5 days | 11/10/20 8:00 AM |
| 13 | Mobile implementation of entering information of the project being created | Mobile Team | 5 days | 11/10/20 8:00 AM |
| 14 | Backend implementation of creating and editing paper/project | Backend Team | 5 days | 11/10/20 8:00 AM |
| 15 | Frontend implementation of creating and editing paper/project | Frontend Team | 5 days | 11/10/20 8:00 AM |
| 16 | Mobile implementation of creating and editing paper/project | Mobile Team | 5 days | 11/10/20 8:00 AM |
| 17 | Backend implemention of search | Backend Team | 5 days | 11/10/20 8:00 AM |
| 18 | Frontend implemention of search | Frontend Team | 5 days | 11/10/20 8:00 AM |
| 19 | Mobile implemention of search | Mobile Team | 5 days | 11/10/20 8:00 AM |

| | | | | |
|---|---|---|---|---|
| 20 | Backend implementation of inviting a user, accepting and rejecting requests | Backend Team | 5 days | 11/17/20 8:00 AM |
| 21 | Frontend implementation of inviting a user, accepting and rejecting requests | Frontend Team | 5 days | 11/17/20 8:00 AM |
| 22 | Mobile implementation of inviting a user, accepting and rejecting requests | Mobile Team | 5 days | 11/17/20 8:00 AM |
| 23 | Backend implementation of seeing, accepting and rejecting invitations | Backend Team | 5 days | 11/17/20 8:00 AM |
| 24 | Frontend iimplementation of seeing, accepting and rejecting invitations | Frontend Team | 5 days | 11/17/20 8:00 AM |
| 25 | Mobile implementation of seeing, accepting and rejecting invitations | Mobile Team | 5 days | 11/17/20 8:00 AM |
| 26 | Backend implemention of entering affiliation | Backend Team | 5 days | 11/17/20 8:00 AM |
| 27 | Frontend implemention of entering affiliation | Frontend Team | 5 days | 11/17/20 8:00 AM |
| 28 | Mobile implemention of entering affiliation | Mobile Team | 5 days | 11/17/20 8:00 AM |
| 29 | Backend implementation of log out | Backend Team | 5 days | 11/24/20 8:00 AM |
| 30 | Frontend implementation of log out | Frontend Team | 5 days | 11/24/20 8:00 AM |
| 31 | Mobile implementation of log out | Mobile Team | 5 days | 11/24/20 8:00 AM |
| 32 | Backend implementation of seeing the status of paper/project | Backend Team | 5 days | 11/24/20 8:00 AM |
| 33 | Frontend implementation of seeing the status of paper/project | Frontend Team | 5 days | 11/24/20 8:00 AM |
| 34 | Mobile implementation of seeing the status of paper/project | Mobile Team | 5 days | 11/24/20 8:00 AM |
| 35 | Backend implementation of home page | Backend Team | 5 days | 11/24/20 8:00 AM |
| 36 | Frontend implemention of home page | Frontend Team | 5 days | 11/24/20 8:00 AM |
| 37 | Mobile implemention of home page | Mobile Team | 5 days | 11/24/20 8:00 AM |
| 38 | Preparation of Milestone document | Backend Team... | 5 days | 12/1/20 8:00 AM |
| 39 | Backend implementation of e-mail validation | Backend Team | 5 days | 12/8/20 8:00 AM |
| 40 | Frontend implementation of e-mail validation | Frontend Team | 5 days | 12/8/20 8:00 AM |
| 41 | Mobile implementation of e-mail validation | Mobile Team | 5 days | 12/8/20 8:00 AM |
| 42 | Backend implementation of linking Google Scholar and ResearchGate accounts. | Backend Team | 5 days | 12/8/20 8:00 AM |
| 43 | Frontend implementation of linking Google Scholar and ResearchGate accounts. | Frontend Team | 5 days | 12/8/20 8:00 AM |
| 44 | Mobile implementation of linking Google Scholar and ResearchGate accounts. | Mobile Team | 5 days | 12/8/20 8:00 AM |
| 45 | Backend implementation of simultaneous document preparation | Backend Team | 5 days | 12/8/20 8:00 AM |
| 46 | Frontend implementation of simultaneous document preparation | Frontend Team | 5 days | 12/8/20 8:00 AM |
| 47 | Mobile implementation of simultaneous document preparation | Mobile Team | 5 days | 12/8/20 8:00 AM |
| 48 | Backend impementation of filtering system | Backend Team | 5 days | 12/15/20 8:00 AM |
| 49 | Frontend impementation of filtering system | Frontend Team | 5 days | 12/15/20 8:00 AM |
| 50 | Mobile impementation of filtering system | Mobile Team | 5 days | 12/15/20 8:00 AM |
| 51 | Backend Implementation of events page | Backend Team | 5 days | 12/15/20 8:00 AM |
| 52 | Frontend Implementation of events page | Frontend Team | 5 days | 12/15/20 8:00 AM |
| 53 | Mobile Implementation of events page | Mobile Team | 5 days | 12/15/20 8:00 AM |
| 54 | Backend implementation of seeing events detailed | Backend Team | 5 days | 12/22/20 8:00 AM |
| 55 | Frontend implementation of seeing events detailed | Frontend Team | 5 days | 12/22/20 8:00 AM |
| 56 | Mobile implementation of seeing events detailed | Mobile Team | 5 days | 12/22/20 8:00 AM |
| 57 | Backend implementation of notification system | Backend Team | 5 days | 12/22/20 8:00 AM |
| 58 | Frontend implementation of notification system | Frontend Team | 5 days | 12/22/20 8:00 AM |
| 59 | Mobile implementation of notification system | Mobile Team | 5 days | 12/22/20 8:00 AM |
| 60 | Preparation of Milestone 2 document | Backend Team... | 5 days | 12/29/20 8:00 AM |
| 61 | Backend implementation of editing all information entered by the user | Backend Team | 5 days | 1/5/21 8:00 AM |
| 62 | Frontend implementation of editing all information entered by the user | Frontend Team | 5 days | 1/5/21 8:00 AM |
| 63 | Mobile implementation of editing all information entered by the user | Mobile Team | 5 days | 1/5/21 8:00 AM |
| 64 | Backend implementation of tracking information shared by other collaborating users | Backend Team | 5 days | 1/5/21 8:00 AM |
| 65 | Frontend implementation of tracking information shared by other collaborating users | Frontend Team | 5 days | 1/5/21 8:00 AM |
| 66 | Mobile implementation of tracking information shared by other collaborating users | Mobile Team | 5 days | 1/5/21 8:00 AM |
| 67 | Backend implementation semantic search | Backend Team | 5 days | 1/5/21 8:00 AM |
| 68 | Frontend implementation semantic search | Frontend Team | 5 days | 1/5/21 8:00 AM |
| 69 | Mobile implementation semantic search | Mobile Team | 5 days | 1/5/21 8:00 AM |
| 70 | Backend implementation of account delete | Backend Team | 5 days | 1/12/21 8:00 AM |
| 71 | Frontend implementation of account delete | Frontend Team | 5 days | 1/12/21 8:00 AM |
| 72 | Mobile implementation of account delete | Mobile Team | 5 days | 1/12/21 8:00 AM |
| 73 | Backend implementation of ratings and comments | Backend Team | 5 days | 1/19/21 8:00 AM |
| 74 | Frontend implementation of ratings and comments | Frontend Team | 5 days | 1/19/21 8:00 AM |
| 75 | Mobile implementation of ratings and comments | Mobile Team | 5 days | 1/19/21 8:00 AM |

## User Scenarios

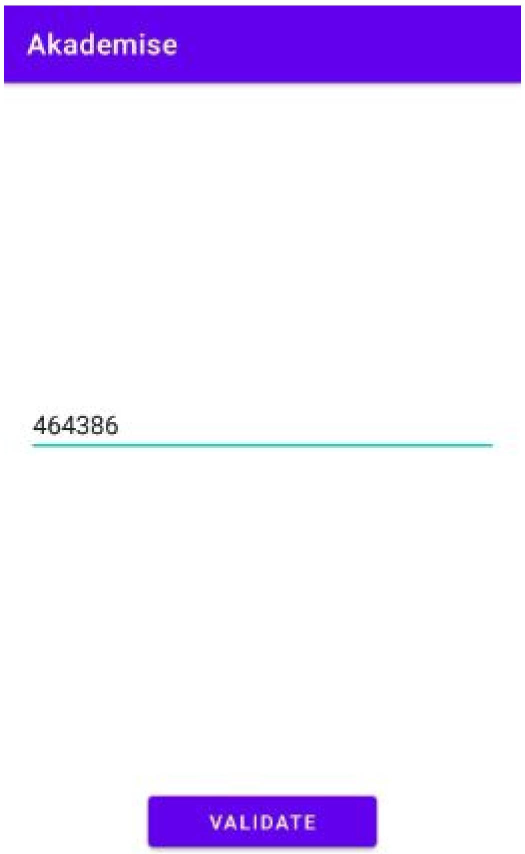
### Justin Johnson getting familiar with Android app

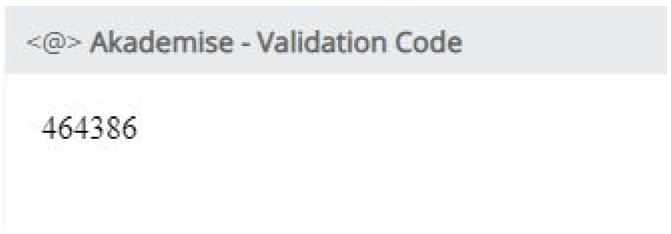




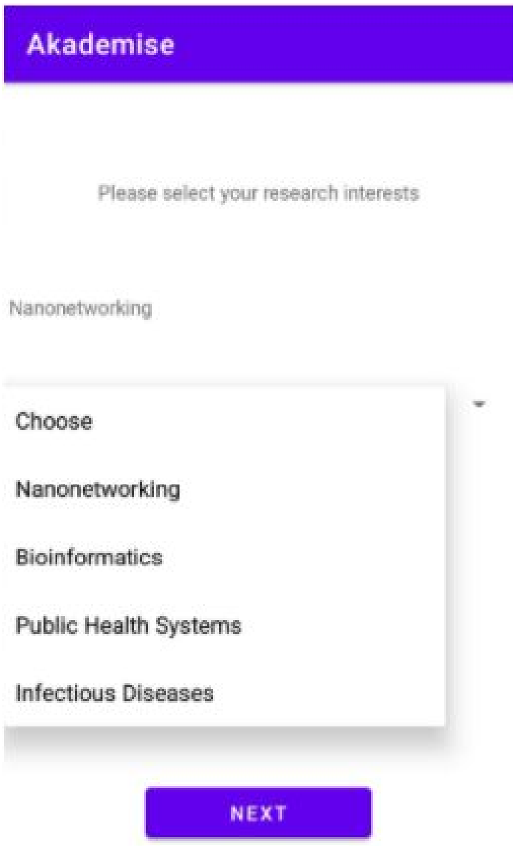
bafyunukke@nedoz.com
••••••
Justin
Johnson

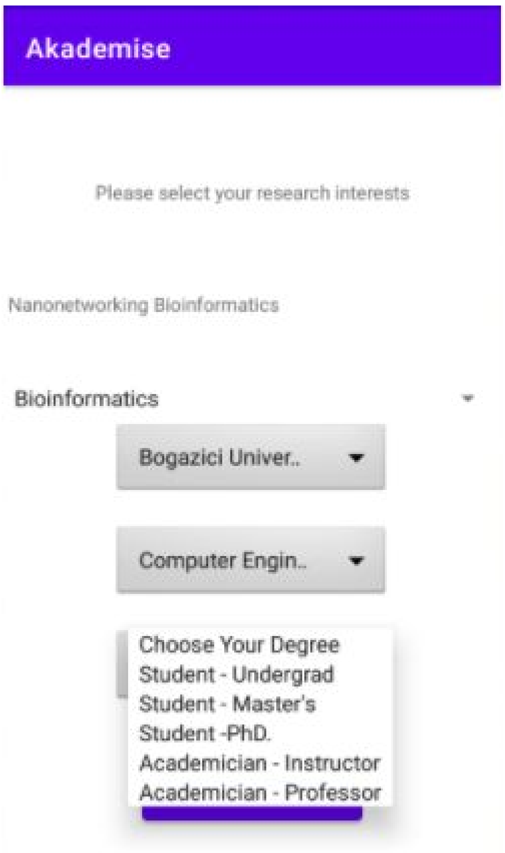
Justin Johnson is a good friend of George. George is using our app already and suggests Justin to use our app. Justin decides to download the android app of Akademise. After that he needs to sign up first, he proceeds to sign up from the login page. In the sign up page he fills the necessary information to create his account. These necessary information are email, password, name, and surname. He will need this email and password to login later.



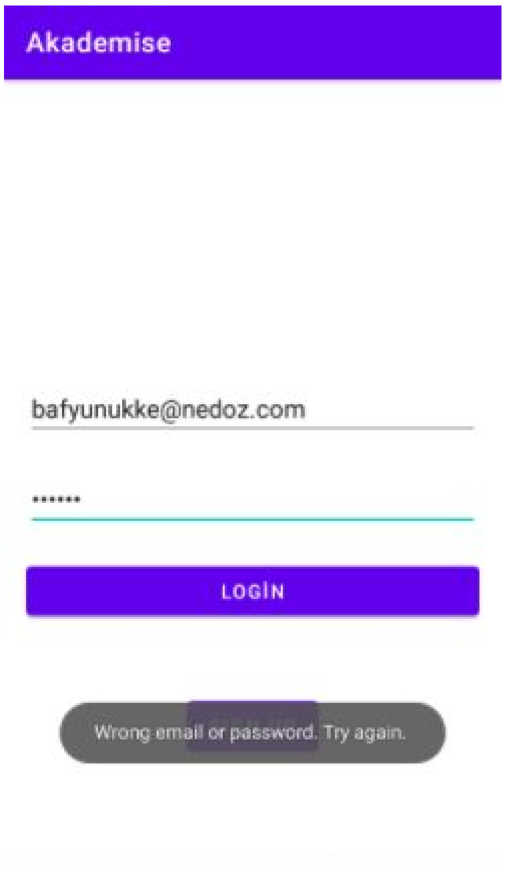


When he presses the sign-up button, validation code is sent to his provided e-mail, then he enters the validation code to proceed to the personal info page.





In the personal info page, Justin enters his affiliation related information which are his university, department, and degree. Justin should enter research areas that he is interested in, because we will use this data to suggest related projects/users in later steps of our development.


Akademise

bafyunukke@nedoz.com

••••••

Wrong email or password. Try again.

Akademise

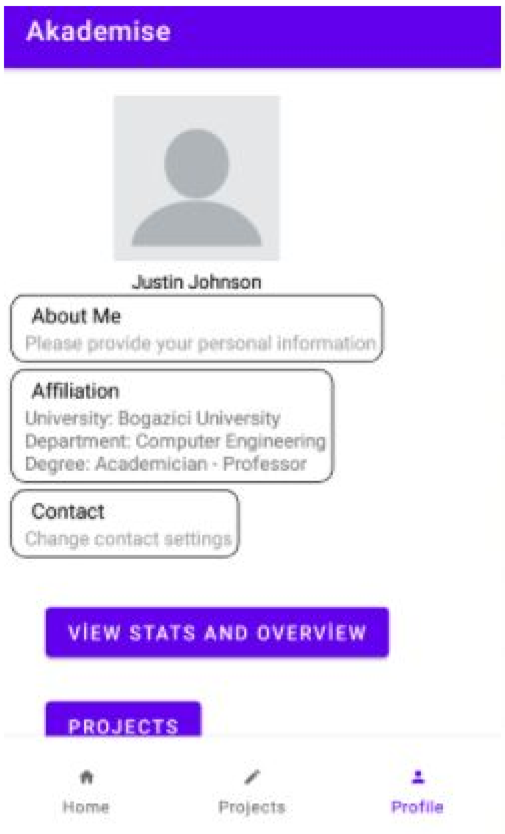
Successful.

Home
Projects
Profile

Normally, we proceed to the homepage after the personal info page and Justin does not need to login again, but we want to show login functionality too in this demo. Justin enters his email and password. If he enters his email or password incorrectly, there is a warning message pop up and he can try again to enter his email and password. After a successful login, we get a response token as validator of the user and we store the token of the user in local storage of the app. We use this token to send posts and get requests.

Akademise
Home
Projects
Profile

After successful login Justin proceeds to homepage. At homepage, we are planning to display feeds of the user. By feed, I mean related projects according to his previous collaborated projects and research interests, but this was not a task of the first milestone.

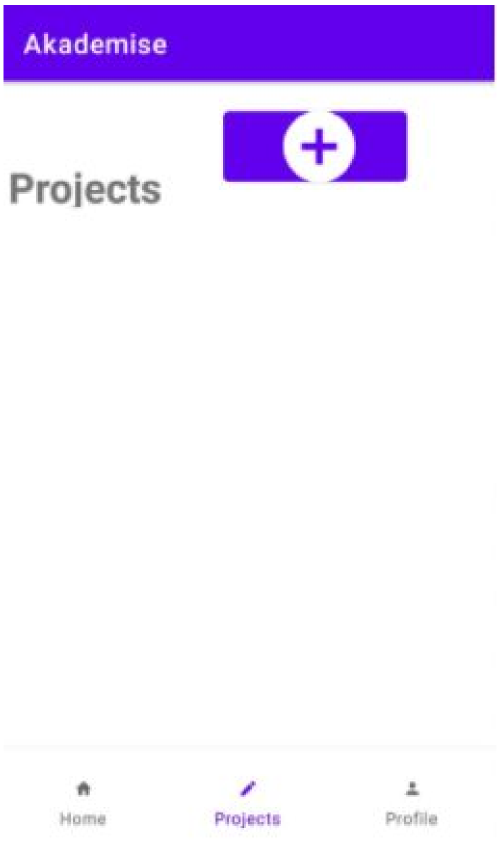


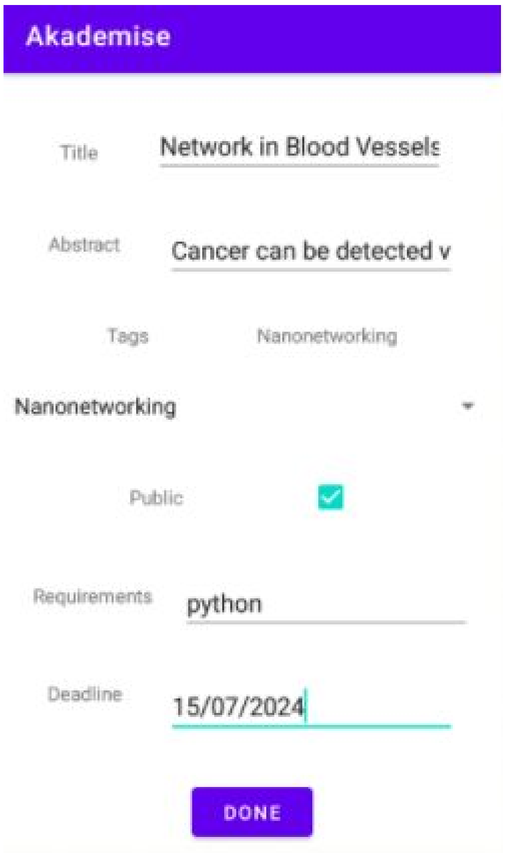
About Me
Please provide your personal information

Affiliation
University: Bogazici University
Department: Computer Engineering
Degree: Academician - Professor

Contact
Change contact settings

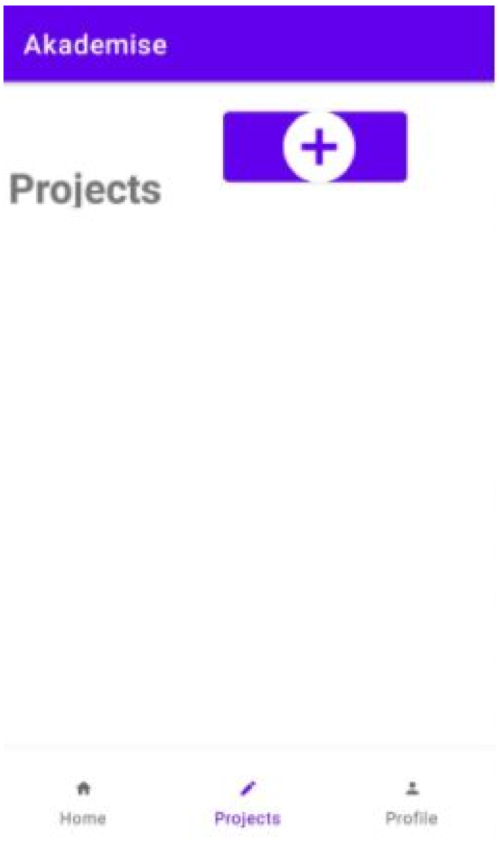
Justin wants to check his profile first. He wants to be sure that his profile is created correctly, so he will not get unwanted invites from other users because of the wrong information on his profile page. There is a Stats & Overview page which shows the stats of the user, these values are not assigned yet, because we do not have stats functionalities for the first milestone.

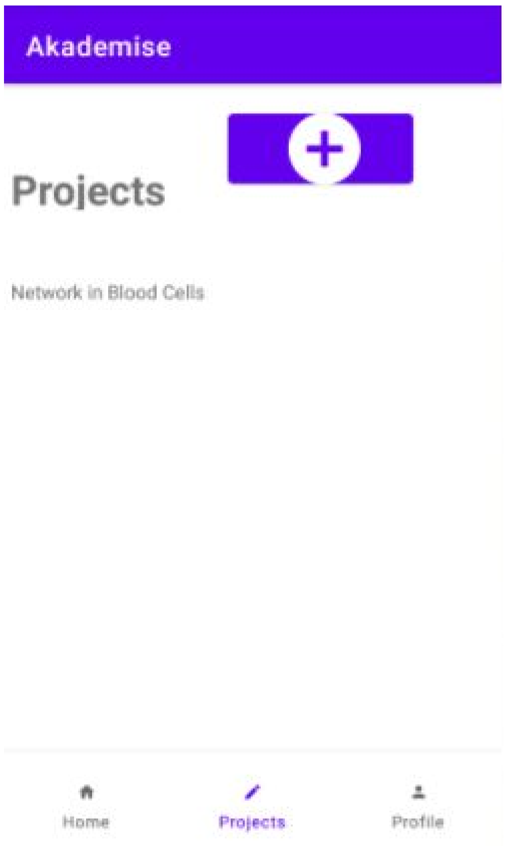




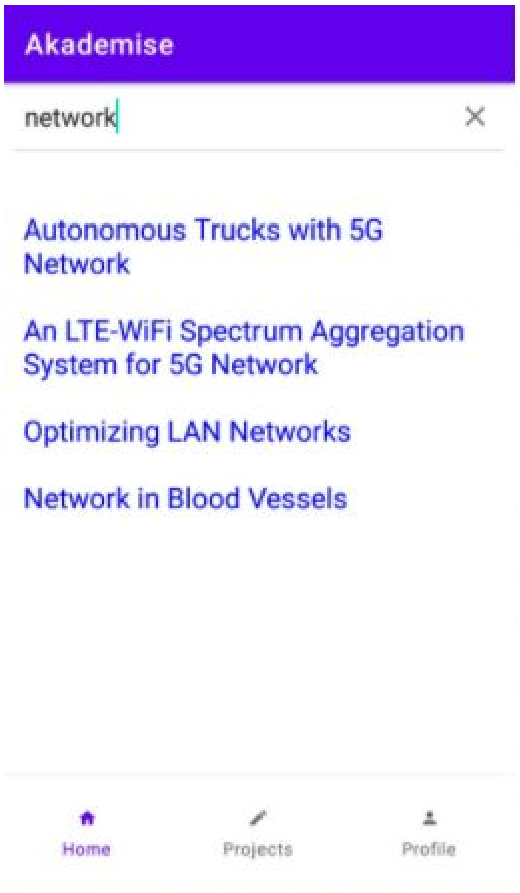
Title: Network in Blood Vessels
Abstract: Cancer can be detected v
Tags: Nanonetworking
Nanonetworking
Public: ✓
Requirements: python
Deadline: 15/07/2024

After checking his profile, he wants to create a project called Network in blood vessels’. This project will be public and related with Nano Networking. Its abstract will provide the basic description of the project.





After filling the project form, he presses the done button to go back to the projects page, he cannot see his project. After refreshing the page, he can see his recently created project, too.


Akademise

network

Autonomous Trucks with 5G Network

An LTE-WiFi Spectrum Aggregation System for 5G Network

Optimizing LAN Networks

Network in Blood Vessels

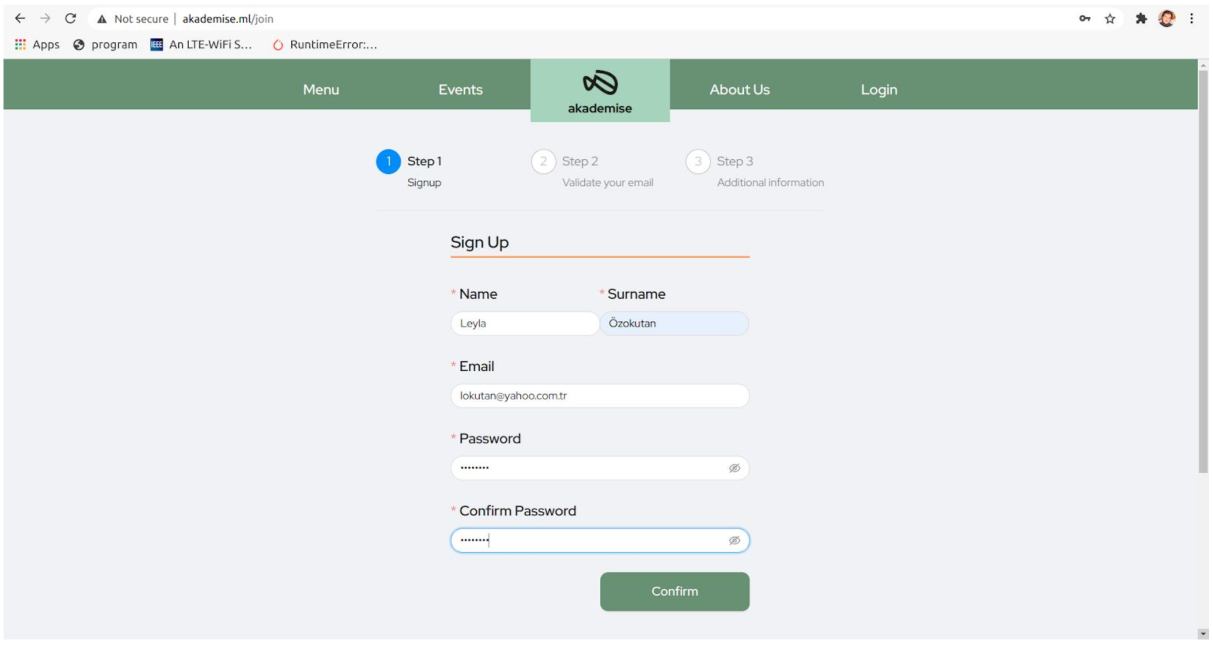
Also, he heard about the searching mechanism of the Akademise that he knows he can search projects with respect to title. Justin wants to search for ‘Network' related projects. He just enters the word 'network' to get the projects. For the first milestone, we just display the title of the project to show the search functioning well. The search functionality will be improved later but we want to provide basic search functionality to the customer.

## Leyla Özokutan needs a collaborator for her project

Leyla Özokutan wants to work on a project about the usage of computer science in education for immaterial formal logic. But she has no one to work on the project with. She stumbles upon akademise and thinks it is just what she is looking for She proceeds to sign up, clicking join. She enters her information.

Step1

| |
|---|
| name:Leyla<br>surname: Özokutan<br>e-mail: lokutan@yahoo.com.tr<br>password: Lokutan1 |


Not secure | akademise.ml/join

Apps program An LTE-WiFi S... RuntimeError:...

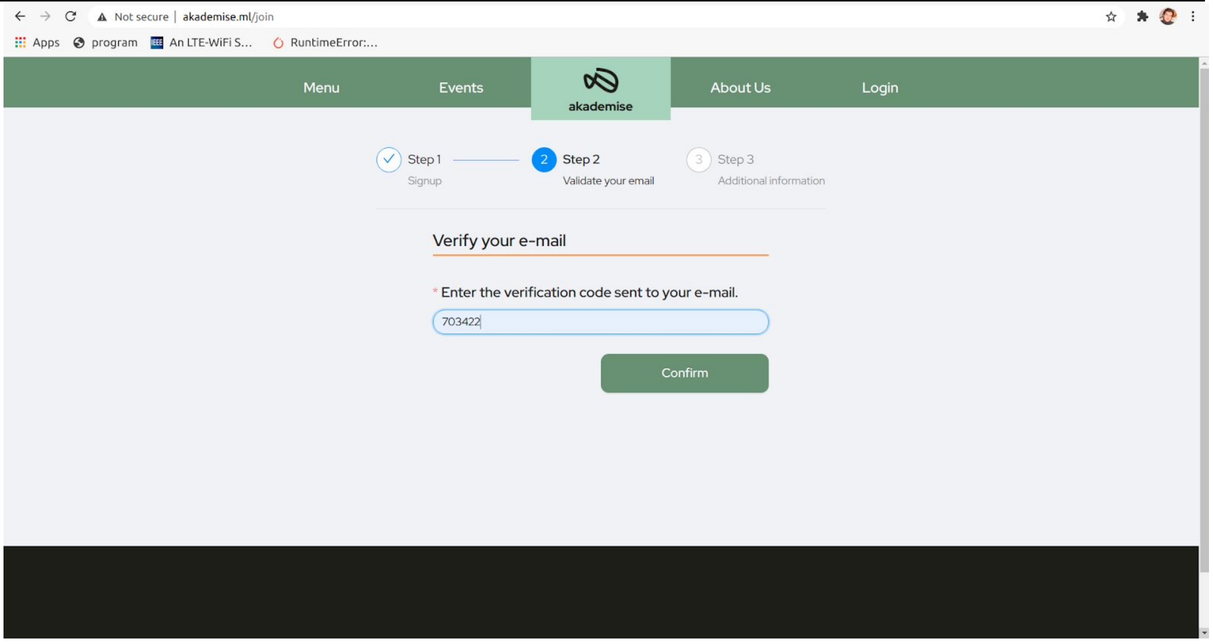
Sign Up

* Name: Leyla
* Surname: Özokutan
* Email: lokutan@yahoo.com.tr
* Password: ••••••••
* Confirm Password: ••••••••

After step 1, Leyla receives a validation code as an email.

Step2

| |
|---|
| Validation code: the code sent to her email address |


Not secure | akademise.ml/join

Apps program An LTE-WiFi S... RuntimeError:...

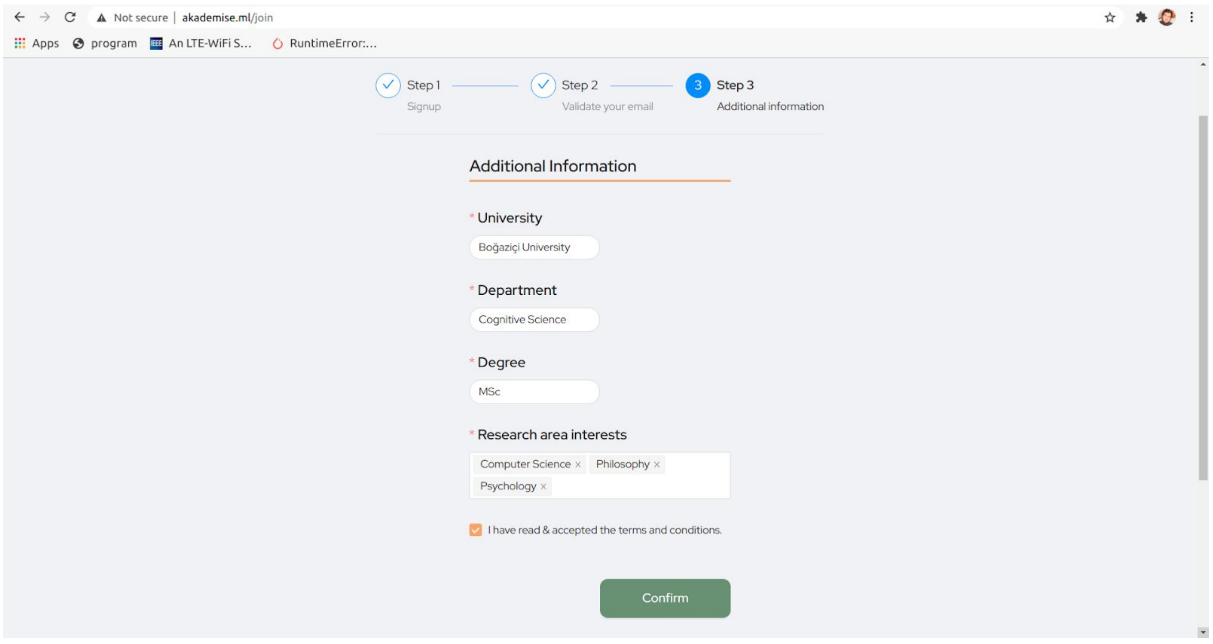
Verify your e-mail

* Enter the verification code sent to your e-mail.: 703422

She validates her email by entering the code sent to her by akademise.

Step3

| |
|---|
| university: Boğaziçi University<br>department: Cognitive Science<br>degree: MSc<br>research area interests: Computer Science, Philosophy, Psychology |



Not secure | akademise.ml/join

Apps program An LTE-WiFi S... RuntimeError:...

Step 1 Signup — Step 2 Validate your email — Step 3 Additional information

## Additional Information

* University: Boğaziçi University

* Department: Cognitive Science

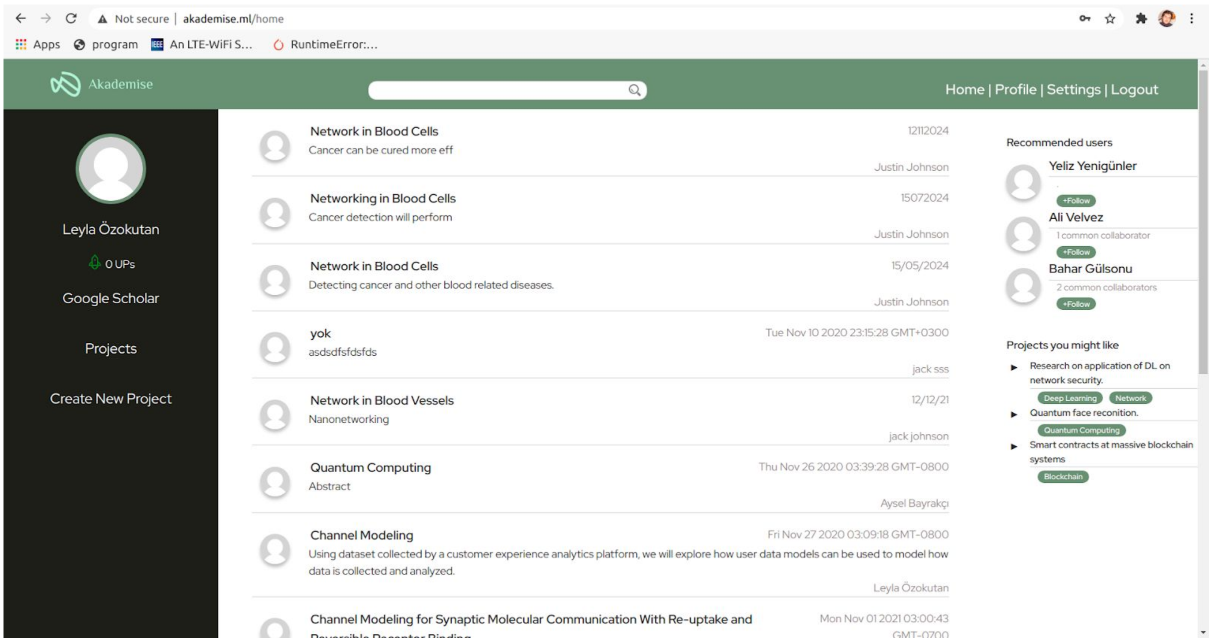
* Degree: MSc

* Research area interests: Computer Science × Philosophy × Psychology ×

☑ I have read & accepted the terms and conditions.

Confirm

She is now an Akademise member! She is taken to her home page, which is out of this milestone's scope so let me just skip that and proceed. We can see that the information she entered can be seen on the sidebar, with her profile picture and name.



Not secure | akademise.ml/home

Apps program An LTE-WiFi S... RuntimeError:...

Akademise

Home | Profile | Settings | Logout

Leyla Özokutan

0 UPs

Google Scholar

Projects

Create New Project

Network in Blood Cells — 12112024
Cancer can be cured more eff
Justin Johnson

Networking in Blood Cells — 15072024
Cancer detection will perform
Justin Johnson

Network in Blood Cells — 15/05/2024
Detecting cancer and other blood related diseases.
Justin Johnson

yok — Tue Nov 10 2020 23:15:28 GMT+0300
asdsdfsfdsfds
jack sss

Network in Blood Vessels — 12/12/21
Nanonetworking
jack johnson

Quantum Computing — Thu Nov 26 2020 03:39:28 GMT-0800
Abstract
Aysel Bayrakçı

Channel Modeling — Fri Nov 27 2020 03:09:18 GMT-0800
Using dataset collected by a customer experience analytics platform, we will explore how user data models can be used to model how data is collected and analyzed.
Leyla Özokutan

Channel Modeling for Synaptic Molecular Communication With Re-uptake and — Mon Nov 01 2021 03:00:43 GMT-0700

Recommended users

Yeliz Yenigünler
+Follow

Ali Velvez
1 common collaborator
+Follow

Bahar Gülsonu
2 common collaborators
+Follow

Projects you might like

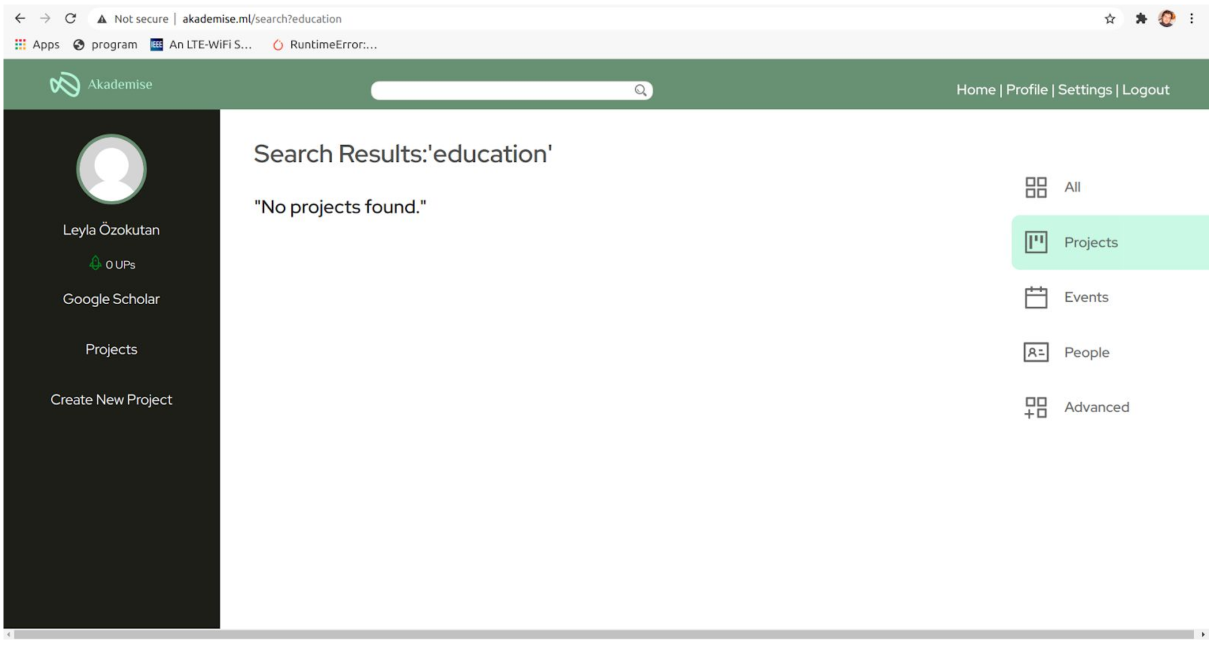
▶ Research on application of DL on network security.
Deep Learning Network

▶ Quantum face reconition.
Quantum Computing

▶ Smart contracts at massive blockchain systems
Blockchain

Now, Leyla is continuing her search. She wants to know if any project like what's on her mind is on akademise. So she performs a search.

| Query: "education" |
|---|

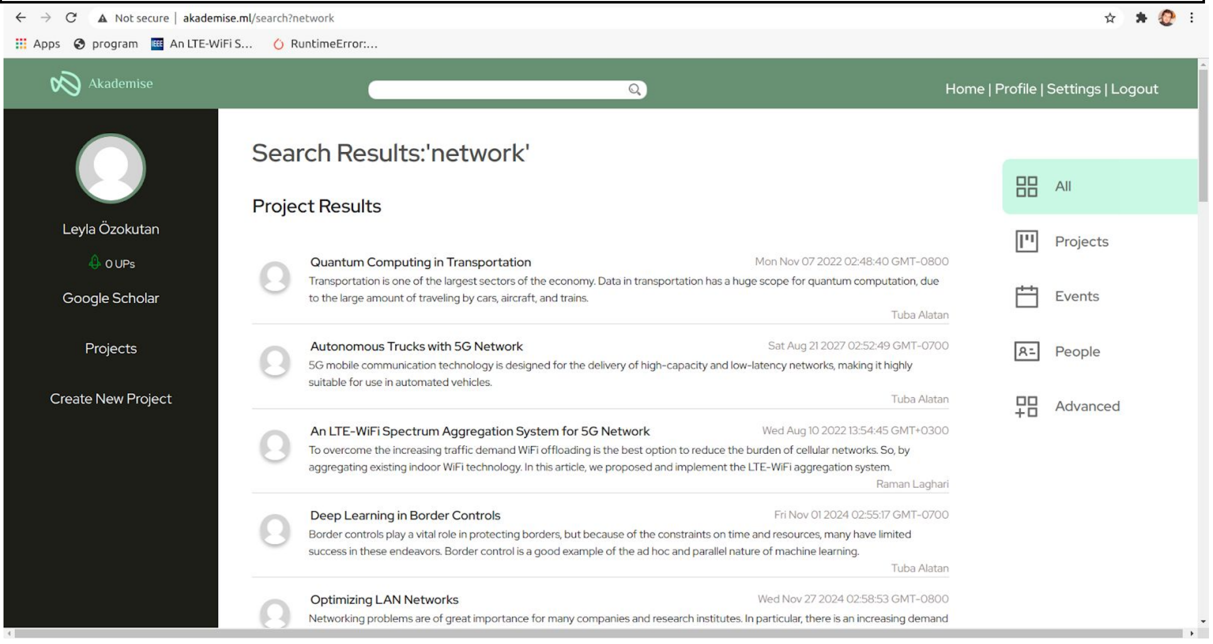


Search Results:'education'

"No projects found."

She sees that no such project exists, she looks for some topics of interest to learn about the different types of projects on Akademise.

Query: “network”



Search Results:'network'

Project Results

Quantum Computing in Transportation — Mon Nov 07 2022 02:48:40 GMT-0800
Transportation is one of the largest sectors of the economy. Data in transportation has a huge scope for quantum computation, due to the large amount of traveling by cars, aircraft, and trains.
Tuba Alatan

Autonomous Trucks with 5G Network — Sat Aug 21 2027 02:52:49 GMT-0700
5G mobile communication technology is designed for the delivery of high-capacity and low-latency networks, making it highly suitable for use in automated vehicles.
Tuba Alatan

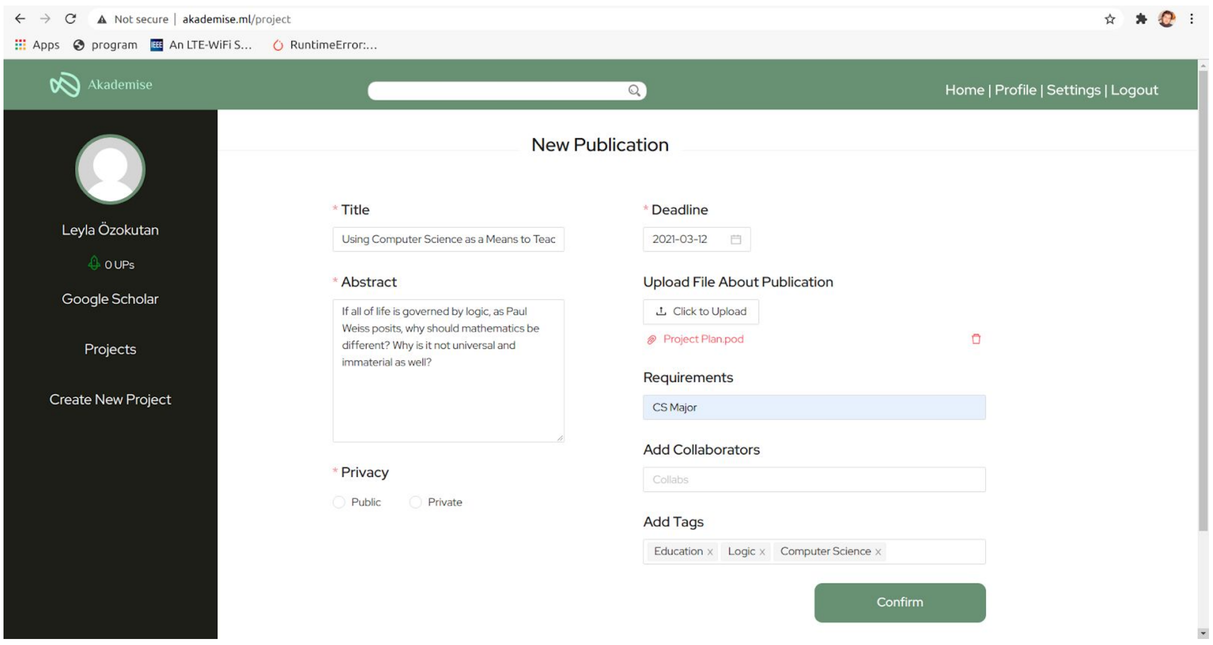
An LTE-WiFi Spectrum Aggregation System for 5G Network — Wed Aug 10 2022 13:54:45 GMT+0300
To overcome the increasing traffic demand WiFi offloading is the best option to reduce the burden of cellular networks. So, by aggregating existing indoor WiFi technology. In this article, we proposed and implement the LTE-WiFi aggregation system.
Raman Laghari

Deep Learning in Border Controls — Fri Nov 01 2024 02:55:17 GMT-0700
Border controls play a vital role in protecting borders, but because of the constraints on time and resources, many have limited success in these endeavors. Border control is a good example of the ad hoc and parallel nature of machine learning.
Tuba Alatan

Optimizing LAN Networks — Wed Nov 27 2024 02:58:53 GMT-0800
Networking problems are of great importance for many companies and research institutes. In particular, there is an increasing demand

Later on, she proceeds to create a new project so people can apply to collaborate. She enters the information as follows:

*Using Computer Science as a Means to Teach Immaterial Formal Logic
*If all of life is governed by logic, as Paul Weiss posits, why should mathematics be different? Why is it not universal and immaterial as well? I will show how the use of other forms of logic, especially fiction and fantasy literature, can help students to come to terms with immateriality.
*Education, Logic, Computer Science


Not secure | akademise.ml/project

Apps program An LTE-WiFi S... RuntimeError:...

Akademise

Home | Profile | Settings | Logout

Leyla Özokutan

0 UPs

Google Scholar

Projects

Create New Project

New Publication

* Title: Using Computer Science as a Means to Teac

* Deadline: 2021-03-12

* Abstract: If all of life is governed by logic, as Paul Weiss posits, why should mathematics be different? Why is it not universal and immaterial as well?

Upload File About Publication: Click to Upload

Project Plan.pod

Requirements: CS Major

Add Collaborators: Collabs

* Privacy: Public Private

Add Tags: Education x Logic x Computer Science x

Confirm

She clicks confirm.

Now her project is created and open for collaboration requests!

# The Code Structure and Group Process

## Frontend

Team:

- Cemre Efe Karakaş
- Ali Ramazan Mert
- Emilcan Arıcan
- Ömer Faruk Doğru

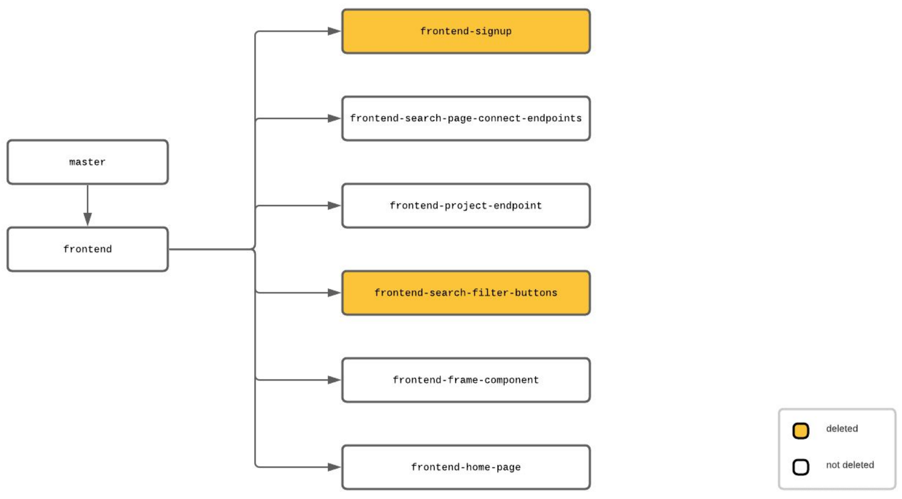
Tools Used:
- Figma (collaborative design tool)
- Yarn (package manager)
- React (frontend library)
- VSCode (ide)
- Ant Design (react UI library)

As the frontend team, we opted for using React framework and making use of Antd's readily available components. As our friend Ali had quite a bit of experience with both, we organized three workshops. One for react, one for antd and one for redux and async calls.

We used the existing branch under the name `frontend`. During the workshops there was not really a lot of simultaneous work involved so we pushed directly into the frontend branch.

Later on, we made small feature additions and typo corrections on the frontend branch and created our own branches for specific tasks (like separate pages).

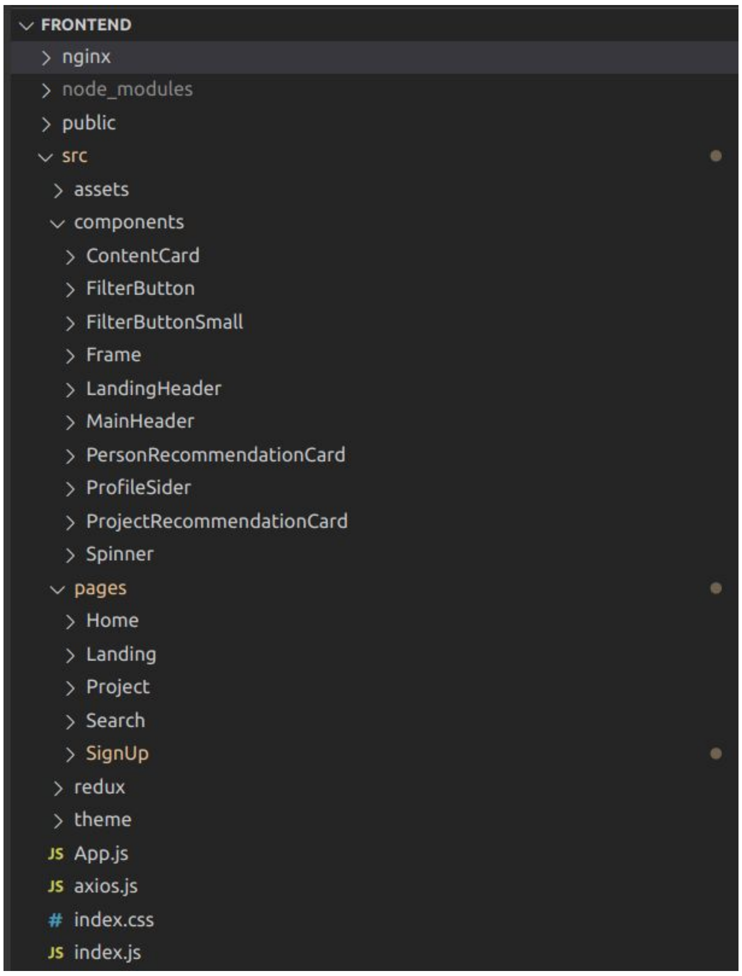
Frontend Related Branches


master
frontend
frontend-signup
frontend-search-page-connect-endpoints
frontend-project-endpoint
frontend-search-filter-buttons
frontend-frame-component
frontend-home-page
deleted
not deleted

Each week we take the responsibilities given to the frontend team and divide them between the team members. We set a deadline before the team's internal deadline to have time for overcoming possible bugs and difficulties.

After the feature being developed in a branch is complete, a pull request is sent. The other frontend members are assigned. At least one team member reviews the changes and makes comments for possible improvements. The user responsible for the request reviews the code and re-commits. After this point, the pull request is approved by a reviewer via comment. And a team member merges the branch. If there is no planned future work for a branch, the branch is deleted.

The directory structure is as follows:



```
FRONTEND
  nginx
  node_modules
  public
  src
    assets
    components
      ContentCard
      FilterButton
      FilterButtonSmall
      Frame
      LandingHeader
      MainHeader
      PersonRecommendationCard
      ProfileSider
      ProjectRecommendationCard
      Spinner
    pages
      Home
      Landing
      Project
      Search
      SignUp
    redux
    theme
    App.js
    axios.js
    index.css
    index.js
```

Each folder in pages and components contains a jsx file for the page/component and a js file for css styles.
The assets folder contains files that are used in pages. The public folder keeps public files like images for the website.

## Backend

Team:

- Muhammed Enes Toptaş
- Hamza Işıktaş
- Hazer Babür
- Ömer Faruk Özdemir

Tools Used:

- ExpressJS (framework)
- PostMAN (api testing)
- VSCode (text editor)
- PostgreSQL (database)
- Docker (containerization)
- Sequelize (orm)

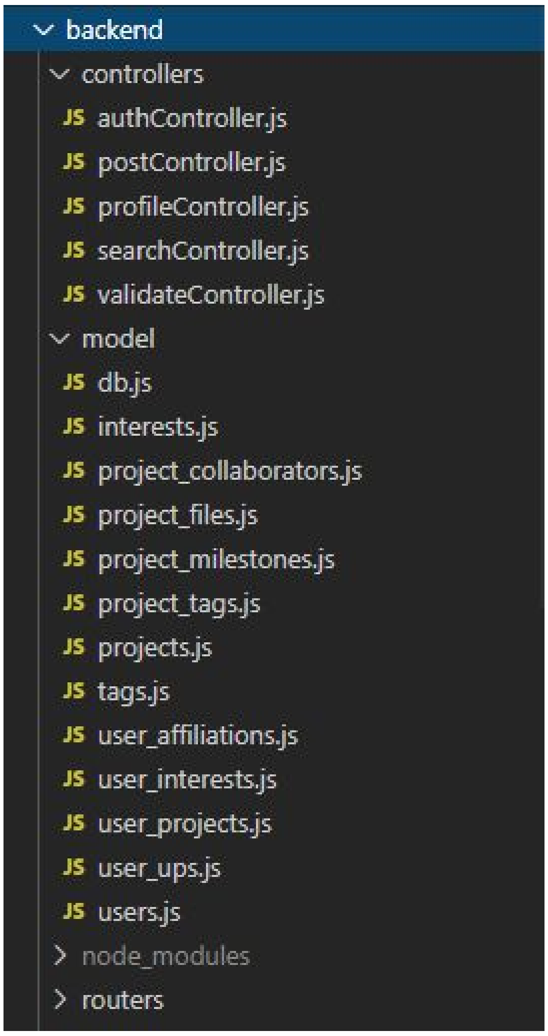
As the backend team, we voted to use NodeJS based ExpressJS for our backend services. Some members of our team, like Ömer and Hazer didn't have prior experience on backend so we made sure we worked and wrote consistent code, to make understanding easier for all of us.

We used Visual Studio Code as our code editor. And utilized Git branches with Visual Studio's integrated Git system to make version control simpler. For technologies we didn't know, such as Docker or Sequelize, we followed online tutorials and worked very closely with each other to not miss any points.

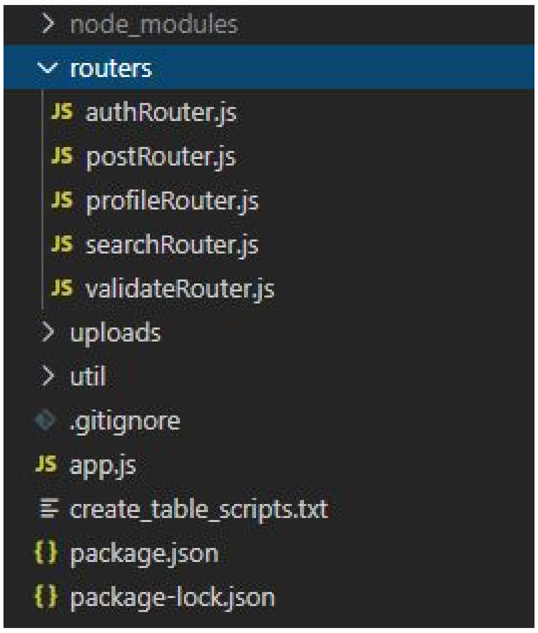
We created a branch under the name "backend" for project implementation.
Every week after our group meetings, we organized a team meeting and discussed what needed to be done and how it could be done. We always tried to open up new issues for newer problems and the most suitable person, both time-wise and skill-wise, would pick it up.

We also had other branches named ORM_DB_Design and profil, as we didn't want to push directly onto the backend branch, we pushed our commits onto those branches then opened Pull Requests, requesting reviews of others.
We tried to follow MVC design pattern as much as we could.

The code directory looks as belows:



```
backend
  controllers
    authController.js
    postController.js
    profileController.js
    searchController.js
    validateController.js
  model
    db.js
    interests.js
    project_collaborators.js
    project_files.js
    project_milestones.js
    project_tags.js
    projects.js
    tags.js
    user_affiliations.js
    user_interests.js
    user_projects.js
    user_ups.js
    users.js
  node_modules
  routers
```


```
> node_modules
v routers
    authRouter.js
    postRouter.js
    profileRouter.js
    searchRouter.js
    validateRouter.js
> uploads
> util
  .gitignore
  app.js
  create_table_scripts.txt
  package.json
  package-lock.json
```

model folder contains the database tables and initialization of the database, routers are about the paths of endpoints, and controllers is where we implement the logic of our server. util is the folder where we implement auxiliary functions to be used throughout the project.

## Mobile

Team:

- Ezgi Gülperi Er
- Cihat Kapusuz
- Doğukan Kalkan
- Gülsüm Tuba Çibuk

Tools Used:

- Android Studio (IDE)

As the mobile team, we decided to use Android Studio as our IDE and Java as our programming language, since all of the team members had previous experience with Java.

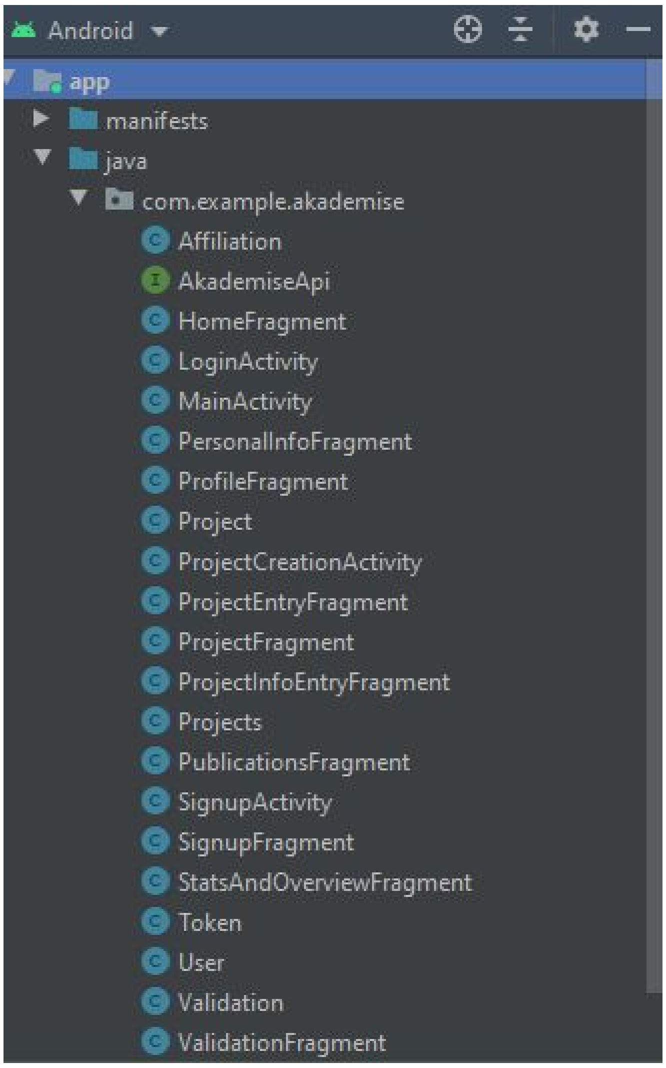
Every member of our team had some previous exposure to android programming but to improve ourselves further and also for the concepts we don't know, we followed the same tutorial and also used Android's official development guide extensively.

We created a branch under the name "android-dev" for project implementation.
Each week we had at least one meeting as the mobile team to divide tasks between members and we set our own internal deadline to iron out any issues and bugs before showing our work to other team members.

When doing work division we assigned independent tasks to each member ( separate pages) so in the first phase we had no possible conflicts.

After every member completes their own task, we had a final meeting where we demonstrate the features we have added and after the meeting we push it to our repository.

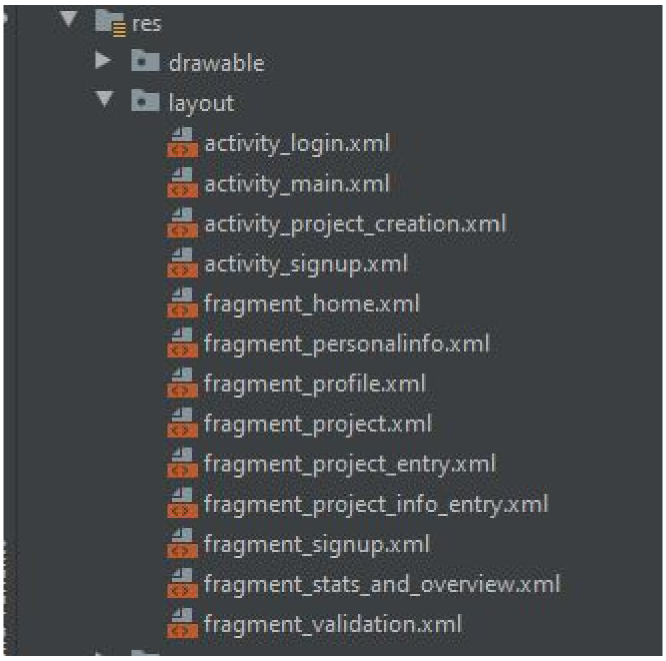
The directory structure is as follows:



```
Android
app
  manifests
  java
    com.example.akademise
      Affiliation
      AkademiseApi
      HomeFragment
      LoginActivity
      MainActivity
      PersonalInfoFragment
      ProfileFragment
      Project
      ProjectCreationActivity
      ProjectEntryFragment
      ProjectFragment
      ProjectInfoEntryFragment
      Projects
      PublicationsFragment
      SignupActivity
      SignupFragment
      StatsAndOverviewFragment
      Token
      User
      Validation
      ValidationFragment
```

We have activities as our main components and fragments for the features we want to implement for that activity. And also we created classes (User,Project) to ease the process while using Refrofit library and also “AkademiseApi” to communicate with backend.

Each Activity and Fragment has an xml file as a layout.



```
res
  drawable
  layout
    activity_login.xml
    activity_main.xml
    activity_project_creation.xml
    activity_signup.xml
    fragment_home.xml
    fragment_personalinfo.xml
    fragment_profile.xml
    fragment_project.xml
    fragment_project_entry.xml
    fragment_project_info_entry.xml
    fragment_signup.xml
    fragment_stats_and_overview.xml
    fragment_validation.xml
```

# Evaluation of Tools and Managing the Project

## Project Management

- I think our project management is quite efficient and works really well this semester. With the experiences we had last semester we can plan and execute tasks better. We set internal deadlines before the course’s deadlines. We set a target point on the project plan. When we plan exactly what should be accomplished in the following week, we start our in-department meetings and divide the work efficiently (speaking on behalf of the frontend team). We set an in-department deadline before the in-team deadline to avoid any unwanted problems. I’m satisfied with the state of the project management.
  -Cemre Efe Karakaş

- We keep our meetings short in a good way and efficient. We specify the tasks to assign to the subgroups and we specify other types of tasks like reporting, note taking and assign to one or more members. Then we meet as subgroups and we define our subtasks. Because of the independence between the teams, we make decisions easily. As a member of the mobile team, we divided the tasks among us fairly and we also helped each other if someone needed help related to their task. We also got help from other subteams especially when we connected the mobile to the backend. We tried to accomplish our tasks before their deadline and I think we

did a good job.
-Gülsüm Tuba Çibuk

- In the beginning of the project, we created a project plan carefully taking the priorities and difficulties of the tasks into account. We did a pretty good job keeping up with the dates that we have set for the tasks. And to avoid last-minute commits and changes we set internal deadlines before the actual deadline (set by the customer). On the day of the internal deadline every team comes together and talks about what is done, what has left to be done and what can be improved. This is useful practice that we plan to stick with throughout the rest of the term .
-Ezgi Gülperi Er

- I think our project management was spectacular. We arranged our project plan according to certain needs and talked about how to shape our plan efficiently. While doing this, we set ourselves realistic goals. We used our weekly meetings effectively and checked where we were on the plan, this helped us keep pace with the project plan. In addition, by putting internal milestones within the team, we had the chance to control what we did just before the real milestone. Thus, we reduced the errors that may occur. Subgroups worked in harmony with each other. All of these took place within the framework of our experiences we gained from last semester's Cmpe 352 course. In general, the whole team did a great job.
-Hazer Babür

- We did a great job while managing the project. At first week, we divided into 3 subgroups and modified requirements, diagrams and the project plan according to our needs. After that we let the teams decide their own task division to catch up the Milestone functionalities. Sub Teams performed excellent jobs about their task division and fulfilling their responsibilities. We also defined a pre-milestone before the real milestone to be sure we are not behind the project plan. Establishing a connection to the backend is left for the last week and that caused a little anxiety. Despite this we completed our tasks in a given time according to the project plan deadlines. I appreciate the work done by my group members.
-Cihat Kapusuz

## Frontend

Tools Used:

- Figma (collaborative design tool)
    - I think figma is a great tool that facilitates the design process. With a ready design we can build our website by following the blueprint provided by the design. Figma gives us a lot of useful information about our design such as readily available CSS for different components, component hierarchy etc.
    -Cemre Efe Karakaş

    - Figma is an easy to learn design tool. It enhanced our design process and allowed us to consider some design ideas before we dive in the code. Having the visual interpretation of the product beforehand helped a lot while implementing the related pages and components.
      -Emilcan Arıcan
- Yarn (package manager)
    - Compared to npm, yarn is more secure, consistent and resource-friendly. I think it is a great tool with no down sides compared to npm.
      -Cemre Efe Karakaş
    - yarn is a package manager alternative to npm. I have used both of those according to my experience; both of those tools are equally powerful and reliable.
      -Emilcan Arıcan
- React (frontend library)
    - I am indifferent to react. It takes care of many functionalities which otherwise require a lot of manual work but still react requires a lot of manual work in other cases. Personally I'd prefer native javascript if it wasn't for Antd's components.
      -Cemre Efe Karakaş
    - React is an excellent tool with a huge community. It eases many cumbersome processes and if there is a problem at any point, you can get help very quickly due to the online community. I think one of the best features of the react is to be able to create reusable components, so that we do not have to write repetitive codes.
      -Emilcan Arıcan
- VSCode (ide)
    - Great ide.
      -Cemre Efe Karakaş
    - VS Code is a great IDE. Users can customize it with extensions. So it can meet many specific needs.
      -Emilcan Arıcan
- Ant Design (react UI library)
    - A really nice ui library. Provides many readily-available components for a lot of cases. I think it is a good addition to the project.
      -Cemre Efe Karakaş
    - Ant Design is one of the most popular UI libraries that is used with react. It has many available components. It eases the process of the creating web interfaces.
      -Emilcan Arıcan

## Backend

- Express.js
  Express.js is an open source and minimalist node.js framework. It is simple and easy to use, and takes care of many functionalities with the help of npm packages.
  -Ömer Faruk Özdemir

- Express.js makes a lot of things easier than it is. Last semester, we used Flask as a framework, I can definitely say that express is better. It has lots of useful modules and usage of those modules makes backend development very efficient and on point.
  -Hazer Babür

- Sequelize ORM
  Sequelize is a promise-based Node.js ORM for Postgres, MySQL, MariaDB, SQLite and Microsoft SQL Server. It makes database access and usage easier, and with its abstraction it makes DB closer to human thinking.
  -Ömer Faruk Özdemir

- At first I wasn't sure if it would be good to use Sequelize. I was used to MySQL and idea of using direct SQL commands for interacting with database was appealing. However, as I used Sequelize ORM I saw that Sequelize makes code more readable and it is as good as using direct SQL commands
  -Hazer Babür

## Mobile

- Android Studio (IDE)
  - Android Studio is complex and huge. It is dedicated to developing android applications. So that it has an emulator to help us test the application without installing apk to our phones. It has an enormous amount of tools and features. It is also integrated with Git. However, there are some downsides of this IDE. One of them is it consumes a lot of RAM, and it is so slow that I waited more than 20 minutes for the emulator to open. Sometimes the IDE and emulator did not respond and I had to close it and open again. Despite all these discouragements, we decided to use this IDE because other methods to implement an android application are way harder.
    - Gülsüm Tuba Çibuk.
  - I can not work in an emulator in Android Studio. It never opens well-functioning and I know times that I have waited half an hour to see it open. Most of the time, I connect my Android device with the Android Studio to study on the project. My computer is a little old and freezes from time to time while I was working on Android Studio. It might be my computer's problem but I think Android Studio has some optimization problems. Also, You can edit layout through design instead of code. That is very good functionality but it has plenty of bugs. You may spend an hour to put a textbox to the desired location in the layout.
    - Cihat Kapusuz

## Assessment of the Customer Presentation

We made Android, Frontend and Backend presentations during the given time. The android demo was presented by Cihat Kapusuz, the frontend demo was presented by Cemre Efe Karakaş and the backend slides presented by Ömer Faruk Özdemir. We used Gülsüm Tuba Çibuk's computer to make the whole presentation so as to not waste time switching between screen shares.

We made rehearsal sessions before the presentation to get used to the general flow of the presentation. That had a positive effect on our presentation. Also, having a separate presenter and someone that performs the actions was very beneficial for the project presentation. You can easily focus on the things that you are going to express and your friend controls the flow of the demo.

We assigned team members to watchout our customers during presentations, too. Hazer Babur, Doğukan Kalkan, Gülsüm Tuba Çibuk and Ezgi Gülperi Er were assigned to take notes about the feedbacks from Meriç Turan, Suzan Üsküdarlı, Alper Ahmetoğlu and Uğur Acar respectively.

The general feedback on our project status was supportive. They appreciate the work that we have done so far. During Q&A sessions, they were interested in our project creation page mostly, they commented about the possible improvements about the project creation page.

We took notes about the customer presentation and will do the necessary changes.

## Deliverables

### Frontend

- **Signup Page:**
  Complete as planned.
- **Home Page:**
  Requires future work but the planned portion is complete.
- **Landing Page:**
  Complete as planned.
- **Project Creation Page**
  Requires future work, a bit behind the plan.
- **Search Page:**
  Requires future work but the planned portion is complete.
- **Logged Off Header Component**
  As planned.
- **Logged In Header Component**
  As planned.

- **Profile Sider:**
  Requires future work, planned portion is complete.

## Backend

- **Login:**
  Complete as planned.
- **Sign in:**
  Complete as planned.
- **Project Creation**
  Completed the basic scheme. Will be enriched.
- **Search Page:**
  Completed the basic functionality. Will be developed.
- **Validation Code**
  Complete as planned.
- **Profile page**
  Complete as planned.
  .

## Mobile

- **Login Page:**
  As planned.
- **Signup Page:**
  As planned.
- **Validation Page:**
  Complete as planned.
- **Affiliation Page:**
  Requires future work, planned portion is complete.
- **Profile Page**
  Requires future work, planned portion is complete.
- **Project Page**
  Requires future work, planned portion is complete.
- **Project Creation Page**
  Requires future work, planned portion is complete.
- **Search:**
  Requires future work, the planned portion is complete.
- **Home Page:**
  Requires future work but the planned portion is complete.

# Evaluation of the Deliverables

## Frontend

- **Signup Page:**
  - It is implemented as three tabs which makes it aesthetically pleasing. Functionalities of this page is working properly as well.
    -Emilcan Arıcan
- **Home Page:**
  - Implemented as it is designed. Recommend features are not active yet (it was out of the scope of this milestone anyway.).
    -Emilcan Arıcan
- **Landing Page:**
  - The landing page looks complete, and it has a professional feel, especially with the login modal.
    -Cemre Efe Karakaş
  - Landing page design is responsive and easy on the eyes and login modal is a nice touch at this implementation.
    -Emilcan Arıcan
- **Project Creation Page**
  - It needs some work. The page is not responsive, and has some small design issues like adding tags changing the form layout.
    -Cemre Efe Karakaş
  - It seems okay however there should be some revisions. Requirements part should be more sophisticated. And the page is not responsive yet.
    -Emilcan Arıcan
- **Search Page:**
  - The overall design is ok and it captures the essential functionalities.
    -Cemre Efe Karakaş
  - It functions well however retrieval of the search results are messy due to backend implementation. It should be fixed after a revision at the backend side.
    -Emilcan Arıcan
- **Logged Off Header Component**
  - Looks good and works well. One small change could be making sure the logo section stays in the middle in screen sizes. It is slightly left aligned in mobile (xs) mode.
    -Cemre Efe Karakaş
  - It is a good and stylistic implementation. It seems overall responsive but in the range of 500-600 pixels, the drawer button acts strange. It should be fixed.
    -Emilcan Arıcan
- **Logged In Header Component**
  - Simple, effective and faithful to the original design on figma.
    -Cemre Efe Karakaş
  - Responsive and nice implementation as it is designed.
    -Emilcan Arıcan
- **Profile Sider:**

    - Looks good. Although it is not fully implemented yet.
      -Cemre Efe Karakaş
    - It has a simple and easy on the eyes implementation but not fully functional.
      -Emilcan Arıcan

## Backend

- **Login:**
    - It is fully implemented and works as intended. You need to provide your email and password, then the requester will be granted of the userId of loginer and the JWT Access Token
      -Muhammed Enes Toptaş
- **Sign in:**
    - It is fully implemented and works as intended. You provide email, password, name and surname, then if email is unique, the user will be created. Then the requester will get the JWT Access Token and userId of the user. Also a validation code will be sent to their emails.
      -Muhammed Enes Toptaş
- **Project Creation**
    - It is implemented but changes will be made according to what the professor asked us to do. Right now, the requester needs to provide userId, title, abstract, privacy, status, deadline and the requirements of the project. Files can also be included in the request.
      -Muhammed Enes Toptaş
- **Search Page:**
    - It is like a placeholder right now, and will definitely be developed more. Will be a semantic search and will work on multiple fields of the data. Right now when provided a search query, it looks up the database to find similar strings to the given query, also received a type variable to search in the users or in the projects
      -Muhammed Enes Toptaş
- **Validation Code**
    - It is fully implemented and works as intended. Accepts a validation code and userId and checks if it belongs to that user, if it does, then the user gets validated.
      -Muhammed Enes Toptaş

- **Profile page**
    - It is implemented and works as intended but definitely will be developed more. In this endpoint, we return everything related to the user to the requester. This includes emails, their names and surnames, their affiliations and research areas. In future, we plan to also return their projects.
      -Muhammed Enes Toptaş

## Mobile

- **Login Page:**
  - Login page is simple. A user can easily understand what to do. There are simple edit views with explanatory hints. If a user does not have an account, s/he can easily find the sign up button on the screen.
    - Gülsüm Tuba Çibuk
- **Signup Page:**
  - Sign up page is also simple. It requires only 4 pieces of information. Email, password, name, and surname. Easy! We did not fill the page with the full procedure of sign up. It is step by step. Then the user is directed to the validation procedure.
    - Gülsüm Tuba  Çibuk
- **Validation Page:**
  - This page requires only one thing: Validation code! After the user provides his/her email, s/he gets a code as a mail. Then the user should only enter the code. This step is done easily, also.
    - Gülsüm Tuba Çibuk
- **Affiliation Page:**
  - This page comes after validation and it is the final step of the registration. It consists of affiliation and research tags. Affiliation represents the university of the user, department of the user and degree of the user. Research tag represents the users' interests in a tagged fashion to ease the searching process. All of this information can be selected via dropdown combobox.
    - Cihat Kapusuz
- **Profile Page:**
  - This page displays the profile of a user, in which all the information about a user is shown, namely a brief 'About Me' part, an affiliation part, a contact part, the user's name and a profile picture. There are also two buttons that take us to statistics and overview page and projects page. For now, we only display a hardcoded profile. We will be modifying this page so that it could display the profile of the user that is logged in.
    - Doğukan Kalkan
- **Project Page:**
  - Users' own projects are shown here, we acquire the whole projects from the get requests but we show only the titles for now to show our get requests are well functioning. Later, we will show additional features of a project and add a button to show the project's page to see all features of the project.
    - Cihat Kapusuz

- **Project Creation Page**
  - There is a button on the project page with a plus sign. It redirects us to the project creation page. This page is the first prototype of the project creation. It is not fully functional yet, but it shows the general aspects of creating a project. New functionalities like uploading pdf files will be added soon. It has fields for title, abstract, tags, requirements, and deadline. It also has a check view to specify it as public or not. It is a simple version of the project creation but it will be more specific.

    - Gülsüm Tuba Çibuk

- **Search:**
    - Search is working based on titles for now. It returns the whole project as a result but we display only the title for showing the functionality. This will be improved and users will be able to search based on other features of the project. For instance, users will be able to search based on research tags. Also, we will add search user functionality. In addition to those, we will enable semantic search, too.
        - Cihat Kapusuz
- **Home Page:**
    - Homepage was not an issue of the Milestone 1. We provided a basic empty fragment to let us show the general flow of the application. Also there is a search bar on the top of the homepage for the search functionality.
        - Cihat Kapusuz

## Summary of Coding Work

| Name Surname | Summary |
|---|---|
| Cihat Kapusuz | -I implemented the general flow of the sign up on Android and the first draft of the sign up page.<br>-I implemented and tested the view of the projects page on Android.<br>-I followed the Retrofit tutorial which is suggested by Tuba.<br>-I implemented search functionality with respect to title and abstract with the usage of the get request. I used the Retrofit library for this purpose.<br>-I attended all of the Android team meetings and took an active role during the task division.<br>-Tuba checked the functionalities of the Android app before the demo and I assisted her through Zoom. Also,we added final small changes to the code.<br>-In addition to code related work, I prepared the Android demo’s scenario with the help of the Android team and attended rehearsal sessions, and also presented the demo of the Android with the help of Tuba. |

| | |
|---|---|
| Hazer Babur | -searched and found a way to connect Expressjs to database without using Sequelize<br>-implemented project creation, updation, deletion related endpoints and methods, also implemented a method that gathers projects from database according to given parameters<br>-implemented Project related database models and relationships between those models with Sequelize<br>-contributed to general design of backend architecture |
| Ali Ramazan Mert | -Created the React project from scratch and implemented the homepage design and some components on it.<br>-I was the one with the most experience with React, so during the frontend process, I actually prepared some kind of workshops for the team, where each took 1 hour on average.<br>-In general I designed the project structure and chose the necessary npm packages we needed and told the team how to use them properly.<br>-Created a theme file to keep our static theme like colors etc.<br>-Integrated redux to the project from scratch<br>-Created a components folder and put my landing header there in order to make others use it easily<br>-I created a template axios object to make our api calls with it.<br>-I handled async operations on the homepage, like logining.<br>-I implemented the authentication/ authorization logic where we use the local storage to keep our token received from the login request.<br>-I helped Cemre on sign up page in general.<br>-I helped Ömer in the project creation page.<br>-I made the whole deployment/dockerization part on the frontend.<br>-I did a lot of research and testing on CI/CD.<br>-I set up the domain redirection with Cemre. |
| Cemre Efe Karakaş | -I created the signup page from scratch.<br>-I implemented a pagination system for the sign-up page.<br>-I implemented three forms for the signup process.<br>-I used redux to manage the data on the client side. |

| | |
|---|---|
| | -I prepared async action handlers and dispatchers for signup, validation and addinfo actions (called by forms).<br>-I connected the forms to the backend using axios calls to the existing API endpoints.<br>-I used the tokens and other information returned by the endpoints to save them in client-side local storage to implement a real ‘signup’ process.<br>-I obtained the domain ‘akademise.ml’. Ali and I routed it to the frontend amazon instance using nameservers and elastic IP.<br>-I made design changes on the landing page, the header component, the profile sider component, and the recommendation card component.<br>-I made small contributions to the project creation page.<br>-I used history for managing redirects |
| Ömer Faruk Özdemir | -After communicating with the team, receiving information needed by DB, I created the DB design.<br>-I updated the DB design in the following weeks, with respect to changes in the project discussed with the customer and the changes discussed with the team.<br>-I wrote SQL tables according to DB design.<br>-I wrote the Sequelize ORM models according to SQL tables and DB design.<br>-I prepared Backend’s presentation, and presented it. |
| Gülsüm Tuba Çibuk | Firstly, I created the first template for our Android Project. The first template includes templates for Login, Home, Projects, and Profile Pages.<br>After we distributed the jobs among the android team, I implemented ResearchTagFragment.<br>After all team members did their jobs, we again specified new jobs, and discussed the required fields to communicate with the backend.<br>Then I implemented the first prototype of project creation activity according to these fields.<br>After we discussed with our customer, we decided to remove “papers”. We decided that there will be only “projects” that include some papers. Therefore I adjusted the names according to the customer meeting.<br>Then we started to connect our project with the backend. I created the first template for |

| | |
|---|---|
| | API interface, get and post functions using Retrofit library.<br> Then I implemented an example get function for search.<br> After the backend deployment is done, we again specified and distributed the jobs among the android team. Then I implemented the post function for the email validation part.<br> Then I implemented the post function to create a new project.<br> After that, I implemented a get method for existing projects of the user. Then I implemented the jwt validation part and I changed the user class to be able to get user id.<br> Finally, I made some changes on the profile page for the presentation. |
| Ezgi Gülperi Er | -In the first part before Milestone 1, I have implemented the PersonalInfoPage and corresponding layout where the user enters his/her personal information and academic affiliation information during sign up.<br>- After talking with the backend team we decided to change the structure so I merged two fragments and updated the PersonalInfoFragment and xml file accordingly.<br>- I implemented the post function for sending personal information.<br>-I created a new class “Affiliation” to make it easier to communicate with Retrofit. |
| Hamza Işıktaş | -I created the architecture of the backend.<br>-I contributed to creating database<br>-I contributed to connecting the backend to the database.<br>-I deployed the backend to AWS with Enes.<br>-I implemented register and validation related endpoints.<br>-I implemented search endpoint |
| Emilcan Arıcan | -I contributed to the design of the landing, search, and feed pages.<br>-I created the feed page from scratch.<br>-I created the search page from scratch.<br>-I created a profile info sider component that displays a summary of the user information in the sider.<br>-I created a navigation bar component (logged in version).<br>-I created a content card component to display project or user contents.<br>-I created recommendation card |

| | |
|---|---|
| | components to suggest projects and users to follow.<br>-I made backend connections of the search and feed pages.<br>-I used useHistory hook to redirect pages.<br>-I send requests to the backend using Axios to get the search results and feed content. |
| Doğukan Kalkan | -During the process of implementing the requirements of the Milestone 1, I attended all the meetings with the whole team as well as the meetings with the Android Team.<br>-I updated the LoginActivity, and updated the layout for that Login page.<br>-I created the ValidationFragment page and created the layout for the page.<br>-Then I implemented the ProfilePage and created the layout for the page from scratch.<br>-I created the StatisticsAndOverview Page and created the layout for the page.<br>-I updated the ProfilePage so that the buttons would take us to the correct pages.<br>-I created the User class, and updated the SignupFragment to implement the sign up request. As decided earlier, I used retrofit to make requests to the backend server.<br>-I updated the LoginActivity to implement the sign up request.<br>-I added the functionality to save the tokens that help us make the requests for a user. |
| Omer Faruk Dogru | -Contributed to design of pages in the beginning<br>-Created project creation page from scratch<br>-Implemented input form for project creation process<br>-designed page with other components such as profile sider and navigation bar which are implemented by Emilcan<br>-connected form inputs to backend using axios post call to the existing API endpoints<br>-inputs changed json to form-data format before post call to handle uploading a file |
| Muhammed Enes Toptaş | -I contributed to the creation of the architecture of the backend.<br>-I created initial users tables on the database.<br>-I suggested Sequelize ORM to be used on our project.<br>-I implemented JWT Token validation and creation functionalities. |

| | |
|---|---|
| | -I created and deployed the database on AWS EC2 instance, as Dockerized<br>-I created the AWS EC2 instance.<br>-I implemented login related endpoints.<br>-I deployed the backend to AWS with Hamza.<br>-I implemented profile related endpoints, that are about creating, updating and viewing. |