executive-summary.md

Our Mission

Kereviz offers superior on-site language learning meterials, providing the intercultural and communicative environment for the customers who love to interact with each other whilst learning and exercising with our unique learning meterials. Our team will ensure that every one of our customers will get the necessary information on the languages that they like to learn and support with any problemd on the site.

Our Services

Our clients are people who like to learn languages, teach languages and interact and communicate with each other while doing that. For our clients, we have language learning opportunuties including:

- Self assesment exams
- Reading exercises
- Writing exercises
- Listening exercises
- Speaking exercises with natives or learners
- Contacting, following and keeping in touch with other learners
- On-site online chat and data sharing
- Self progress management
- On-site learning plans

Our Team

We are 9 senior students in the department of Computer Engineering in Bogazici University. We are very eager to learn cutting-edge technologies and solutuions to clients' problems when we are developing our product. We think that developing a global language learning platform for everyone and connecting people under this purpose is valuable and act according to that.

2.md

List Of Deliverables

Team	Deliverables
Android	- Register Page - Login Page - Language selection page - Proficiency exam page - Exam result page - Profil page - Viewing someone else's profile - Sending message request to other users - Chats list page - Messaging page - User search - Content search (semantic search) - Exercise types list page - Solving exercise page
Frontend	- Register Page - Login Page - Forgot Password Page - Language selection page - Proficiency exam page - Profile page (self or other user) - Language Page (statistics & summary) - Exercise Pages (Reading/Listening/Grammar/Vocabulary) - Exercise search (semantic) - User Search
Backend	- Register Endpoint - Login Endpoint - Language endpoint - Proficiency exam endpoint - Grade endpoint - Profil Info endpoint - Sending message request endpoint - Chats list endpoint - Messaging endpoints - User search endpoint - Content search (semantic search) endpoint - Solving exercise endpoint - Tagging endpoint - Conversation endpoint

Status and Evaluation of Deliverables

Android

We have implemented register, login, language selection, proficiency exam, exam result page in the previous milestone and made an evaluation for these pages.

Profil Page

User can view their profile page when they clicked the profile icon on the bottom navigation bar and when they login to the app. In the user's own profile page, users can see their languages, progress on these languages, their levels in languages and the ratings(unimplemented now) as stated in the requirement item 1.1.1.1.2.6. and 1.1.1.1.2.10. In future, we are planning to add a button so that user's can see their message requests. Users also can take proficiency exam from their profile page.

Viewing Someone Else's Profile

User can view other user's profile page when they clicked the user's name on the user search result. In other user's profile page, users can see their languages, progress on these languages, their levels in languages and the ratings(unimplemented now) as stated in the requirement item 1.1.1.1.2.6. and 1.1.1.1.2.10. Users also can send a message request on another user's profile page.

Sending Message Request to Other Users

We have implemented the feature where user's can send a message request to each other as stated in the requirement 1.1.2.1.1. Users can see in other user's profile page if their request is approved or not, and send a request if they haven't already. The only unimplemented part of this requirement is to see the list of the message request that has sent to the user and approving/rejecting them as told in requirement 1.1.2.1.2. We will implement that part in the profile page of the user.

Chats List Page

Users can see the list of users that they are having a chat within the chat list page. The chat list page is available when the user clicks on the message icon on the bottom navigation bar. This is not a feature that is stated in the requirements but is an essential part of the implementation of the requirement 1.1.2.1.

Messaging Page

In the messaging page, users can send each other messages and see their previous messages. This page is also a part of the implementation of the requirement 1.1.2.1. Currently, if a user sends a message through the messaging page directly, the user can see their own message popping up in the messaging page directly. However, the user would need to open the messaging page again to see the message that they received as the messaging doesn't work instantaneously. And it wasn't given as a requirement.

User Search Page

Users can reach the user search page by clicking the search icon on the navigation bar. By writing another user's name on the search part, users would be able to list all the users with that name. Users can click on the result and go to that user's profile page. This implementation is the part of requirement item 1.2.8.3. But we don't have other functionalities except for the user search by the user name.

Content Search Page(Semantic Search)

When the user clicks on the search icon in the bottom navigation bar, the user will be able to go to the content search page. In the content search page, the user can do a semantic search with the tags of the exercises. For example, users can search for a tag that they wanted. In the result, user would see the exercises that have that tag or a tag that is related to the tag that the user was searching for. Currently, users can only do a semantic search with the tag of the given exercise. This feature is written to implement the requirements 1.2.8.2. and 1.2.8.1.

Exercise types list page

The exercise type list in Android works as expected. The item 1.2.1.2. (The learning materials should be mapped into 5 different categories: listening, reading, grammar, vocabulary and writing.) of project requirements list has been implemented in this page. We have found related icons for these categories and added to this page. Users can click on them and go to related exercises as in our use case diagram.

Solving exercise page

Users can see their languages, levels, and progresses before the start exercise. The item 1.1.1.1.2.6. (Users shall be able to see their learning process such as completed exercises, grade achievements for each language.) of project requirements list has also been implemented in this page. Questions are retrieved from the backend and displayed with no problem. Unlike the proficiency exam page, the correct answers are shown after each question by making the correct answer dark green, selected answer yellow and wrong answers red. Thus, the item 1.1.1.1.2.3. (After every question, the users shall be able to see the correct answer) of project requirements list has been fulfilled. Users can improve their progress in a language as solving exercises.

Frontend

Previous Pages

We have implemented register, login, forgot password, language selection, proficiency exam in the previous milestone. Until now, we've made several updates.

- We've added antd form validation in register, login, forgot password, proficiency exam pages and updated their design to ant design.
- In proficiency exam, we've made it so that correct answers aren't displayed after each question finishing the exam and selected answers are displayed at each question.
- We've fixed bugs related to sending and getting latest grade in a language.

Profile Page (self or other's)

User can view other user's profile page when they click the user's name on the user search result. User can view self profile page when they click profile button from topbar. In profile page; languages, progress on these languages, their levels in languages are shown as stated in the requirement item 1.1.1.1.2.6. and 1.1.1.1.2.10. Users also can request to chat with a user on another user's profile page.

Sending Message Request to Other Users

We have implemented the feature where user's can send a message request to each other as stated in the requirement 1.1.2.1.1. Users can see in other user's profile page if their request is approved or not, and send a request if they haven't already. The only unimplemented part of this requirement is to see the list of the message request that has sent to the user and approving/rejecting them as told in requirement 1.1.2.1.2. We will implement that part in the profile page of the user.

User Search Page

Users can reach the user search page by going to users page through topbar and find out results by filling search form values. By writing another user's name on the search part, users would be able to list all the users with that name. Users can click on the result and go to that user's profile page. This implementation is the part of requirement item 1.2.8.3.

Content Search Page(Semantic Search)

Users can reach the content search page by going to exercise search page through sidebar. In the content search page, by filling search form values the user can do a semantic search with the tags of the exercises. For example, users can search for a tag that they wanted. In the result, user would see the exercises that have that tag or a tag that is related to the tag that the user was searching for. Currently, users can only do a semantic search with the tag of the given exercise. This feature is written to implement the requirements 1.2.8.2. and 1.2.8.1.

Solving exercise page

Questions are retrieved from the backend and displayed with no problem. Unlike the proficiency exam page, the correct answers are shown after each question by making the correct answer green, selected answer highlighted and wrong answers red. Thus, the item 1.1.1.1.2.3. (After every question, the users shall be able to see the correct answer) of project requirements list has been fulfilled. Users can solve exercises with audio/image in listening/reading/grammar/vocabulary exercises.

Backend

All of our endpoint are available for testing on Swagger. If you can not see the page, please change your DNS configuration, we have trouble with some internet suppliers. We have an API Documentation given below, which is a really good guide to use our API.

API Documentation

Our API documentation can be found in our master branch under Rest API Documentation section and the documentation is updated regularly to contain the latest changes. There are examples of successful and unsuccessfull requests.

📖 **3.md**

Member	Work
İbrahim	Worked on Android application. - Created a simple profile page at first. - Created a login session,after user logged in, its info stored in a cache. - Implemented user search with SearchView. - Added target user page, which is a page users can send message request/message to its friends. - Created ExerciseViews , so users can display and solve exercises. - Created a start exercise page, which users can see their progress before starting the exercise. - Connected exercise solving to back-end. Now after exercise solved, users progress is increased and can be seen from profile page.
Gamze	Worked on Android application. - Created an exercise type selection page. - Created a chats list page and connected it with the navbar message button on all pages. - Created a conversation page, made different colors to messages according to sender and reciver id, created a send button for sending message and connected with the API with Irem.
Ahmet	Worked on Backend. - Implemented profile info endpoint and services. - Implemented the tagging system with all endpoints and services. - Implemented solved exercises services and progress level feature. - Implemented the whole search system with all endpoints and services, both user and content, with the integration of 3rd party API and semantic search. -Implemented messaging, conversation and chat endpoints and services. - Helped Arda creating invitation mechanism.
Irem	Worked on Android application. - Created bottom navigation bar and added it to the existing pages. - Polished user profile page. Added the progress, language level and rating of the languages to the profile page. Connected it to the backed so the information there is updated with the database. - Added the tab view for user search and content search. - Implemented content search page and connected it to exercise view page. - Implemented message page, sending and receiving message feature with Gamze. - Implemented send message request button in user profiles so the users can send and receive message requests to each other.
Arda	Worked on Backend. - Implemented the endpoints and services for chat invitation functionality.
Halit	Worked on web application. - Moved frontend structure and current pages to ant.design removing reactstrap & bootstrap. - Remade language pages layout with a better structure, created navbar & sidebar. - Created Reading, Writing, Vocabulary, Listening Exercises Pages - Created Exercise search page - Added logo and made color customizations. - Bug fixes in frontend.
Emirhan	Worked on web application. - Learned the new structure Halit created, fixed bugs - Added some exercises to frontend. - Learned React Redux

requirements.md

Glossary for Project Requirements

- **User:** A person using the app/web platform of the project.
  - **Guest:** A user who is not registered yet.
  - **Registered:** A user who has signed up to the platform.
  - **Admin:** A user with special privileges.
- **Language:** One of English/Turkish/Chinese or other provided language in the platform.
- **Learning Material:** Content in one of the categories below, in one of the types below, in a given language used to teach users that language.
  - **Categories:** Categories of a learning material, one of Listening, Reading, Grammar, Vocabulary or Writing.
    - **Listening:** Materials relating to improving listening skills in a given language.
    - **Reading:** Materials relating to improving reading skills in a given language.
    - **Grammar:** Materials relating to improving grammar skills in a given language.
    - **Vocabulary:** Materials relating to improving vocabulary skills in a given language.
    - **Writing:** Materials relating to improving writing skills in a given language.
  - **Types:** Types of a learning material, one of Notes, Assignment, Exercise or Exam.
    - **Notes:** Materials that present notes about the content.
    - **Assignments:** Materials that require user to write a long answer(paragraph, essay, etc.) and another user to review & evaluate.
    - **Exercises:** Materials that require user to answer questions which can be automatically graded.
- **Proficiency Exam:** A special exam consisting of questions used to evaluate the expertise of a user in a given language if the user wants to start directly from any level higher than `A1`.
- **Achievement:** A user can get special labels according to accomplishing some tasks with given conditions, such as in 2 minutes.
- **Progress of Learning:** The statistics about the users' learning history of a given language i.e. accomplished exercises, assignments, duration.
- **Interaction:** Users can interact with each other in either one of the ways below.
  - **Communication:** A user can send a request to another user to chat privately.
    - **Request:** When a user sends a chat request to another user, chatting only starts after receiver user accepts the request.
  - **Review:** The process of a user grading another user's writing assignment, and providing feedback to that user related to the assignment.
    - **Feedback:** An explanation of how the user can do better or an error found in one's writing assignment.
    - **Annotation:** A user can add annotation to the writing assignment he/she is reviewing to give feedback to the reviewed.
- **Rating:** A user can rate other users he/she interacted based on the related interaction.
- **Comment:** A user can comment about other users he/she interacted based on the related interaction.
- **Level of Expertise:**: The users' level of proficiency shown as either `A1`, `A2`, `B1`, `B2`, `C1` or `C2` in a given language.
- **Search System:** A system allowing users to search for contents in a given language.
  - **Basic Search System:** A search system based on keywords and semantic search.
  - **Advanced Search System:** A search system that allows search and filter features by type, difficulty and tags.
- **Contribution System:** Users can upload new learning materials and suggest them to be added to the system, or they can suggest new tags to existing material.
  - **Verification:** Admins can verify suggested tags to existing material to add them to the system. Also, for suggested new user uploaded materials, either a specified amount of users can support the suggestion to approval or an admin user can verify the material by himself/herself.

Requirements

1. Functional Requirements

1.1. User Requirements

- **1.1.1. Users**
  - 1.1.1.1. There will be three types of users.
  - **1.1.1.1.1. Guests**
    - 1.1.1.1.1.1. Guest users can only access 5 exercises for each type of learning materials(except writing) before being prompted for registering.
  - **1.1.1.1.2. Registered Users**
    - 1.1.1.1.2.1. Registered users should access to materials anytime and anywhere.
    - 1.1.1.1.2.2. **Proficiency exam**
      - 1.1.1.1.2.2.1. The users who want to start from higher level, shall take a proficiency exam.
      - 1.1.1.1.2.2.2. Users shall be able to see their exam results and see the content of corresponding level of their scores.
    - 1.1.1.1.2.3. After every question, the users shall be able to see the correct answer.
    - 1.1.1.1.2.4. Users shall be able to send their essays to other users for grading which includes feedback as well as the grade.
    - 1.1.1.1.2.5. If one user has graded other's assignment, or they have conversed via messaging, they shall be able to rate or comment on each other.
    - 1.1.1.1.2.6. Users shall be able to see their learning process such as completed exercises, grade achievements for each language.
    - 1.1.1.1.2.7. Users shall be able to upload their suggestion of some learning materials.
    - 1.1.1.1.2.8. Users shall be able to declare whether they want to review essays or not.
    - 1.1.1.1.2.9. Users shall be able to report inappropriate behavior and sensitive content.
    - 1.1.1.1.2.10. Users' profile pages should include the review count, rating, achievements, uploaded contents and comments about them.
  - **1.1.1.1.3. Administrators**
    - 1.1.1.1.3.1. Administrators shall handle reports and be able to suspend or ban user accounts.
    - 1.1.1.1.3.2. Administrators shall be able to accept or reject a suggested content.
- **1.1.2. Communication**
  - 1.1.2.1. Users shall be able to communicate over a messaging channel.
    - 1.1.2.1.1. Two users can use the messaging service if and only if one sends a request for communication and the other one accepts.
    - 1.1.2.1.2. Users shall be able to see their message requests and accept or reject them.
- **1.1.3. Login and Sign-up**
  - 1.1.3.1. Unregistered users shall be able to register after giving the necessary information.
    - 1.1.3.1.1. Users shall provide their email address while registering.
    - 1.1.3.1.2. Users shall provide their names while registering.
    - 1.1.3.1.3. Users shall set up a password while registering to the system.
    - 1.1.3.1.4. Users shall provide their native languages whlie registering.
  - 1.1.3.2. Registered users shall be able to login to the system.

        - 1.1.3.2.1. Users shall be able to login to the system using their email address and their passwords.
        - 1.1.3.2.2. The users that had forgotten their passwords shall be able to set up a new password with a verification e-mail.

1.2. System Requirements

- **1.2.1. Learning Materials**
    - 1.2.1.1. There should be learning materials in different languages.
    - 1.2.1.2. The learning materials should be mapped into 5 different categories: listening, reading, grammar, vocabulary and writing.
    - 1.2.1.3. The materials uploaded by users exist in the system after verification.
    - 1.2.1.4. The materials can have one or more images or sounds.
- **1.2.2. Assignments and Exams**
    - 1.2.2.1. Listening, reading, grammar and vocabulary categories of assignments or exams will be graded automatically by the system.
    - 1.2.2.2. A user should be able to pick the person who shall grade the writing assignment.
    - 1.2.2.3. System provides the answers after every question in exercises.
    - 1.2.2.4. Writing assignments should be uploaded by tags in order for the system to suggest reviewers.
- **1.2.3. Languages**
    - 1.2.3.1. The system shall support multiple languages.
- **1.2.4. Recommendation**
    - 1.2.4.1. For grading writing assignments, the system shall recommend a selection of users.
        - 1.2.4.1.1. the system shall provide a recommendation mechanism that recommends a set of users as potential evaluators of the writing exercises.
- **1.2.5. Annotation**
    - 1.2.5.1. The system shall provide an annotation mechanism for enabling users to annotate text and images.
- **1.2.6. Rating**
    - 1.2.6.1. The system shall support rating and comments only between interacting users.
- **1.2.7. Contribution**
    - 1.2.7.1. New learning materials should be suggested by the users who upload their suggested materials.
    - 1.2.7.2. The admin users shall be able to accept or reject the uploaded learning materials by the users.
- **1.2.8. Searching**
    - 1.2.8.1. The system shall provide a basic search mechanism based on the keywords entered by the users.
    - 1.2.8.2. The system shall provide an advanced search mechanism to filter the content by difficulty, tag, type and like count.
    - 1.2.8.3. The system shall provide an user search mechanism by their name, ratings' value and count, previously reviewed tags and language expertise.
- **1.2.9. Tagging**
    - 1.2.9.1 The system should support a set of tags with every writing assignment.
    - 1.2.9.2 The system should suggest reviewers for writing assignments considering tags of the assignment and the reviewer.
- **1.2.10. Liking and Reporting**
    - 1.2.10.1. Users should be able to report any content for inappropriate, offensive or other reasons.

2. Non-Functional Requirements

2.1. Security

- 2.1.1. User data shall be protected and used according to LAW ON THE PROTECTION OF PERSONAL DATA
- 2.1.2. The personal information, contact information, copyrighted contents, license issues and everything related to these paradigms should be respected and considered.
- 2.1.3. System shall be protected against SQL injection.
- 2.1.4. System should use encryption for personal messages between users.
- 2.1.5. Storing of passwords should conform to security standards such as hashing.

2.2. Reliability

- 2.2.1. The system shall serve to 300 users without breaking.

2.3. Availability

- 2.3.1. Project shall be available on both Android and Web platforms.
- 2.3.2. The application should be deployable on a manually configurable remote server.

2.4. Protocols & Standards

- 2.4.1. The system shall support the W3C Web Annotation Data Model.

2.5. Performance

- 2.5.1. The system shall respond to 100 requests per second.
- 2.5.2. Maximum response time shall be at most 500 ms.

Change Log

26 February 2019

- Added Glossary
- Fixed requirements according to feedback
- Updated requirements to match with the answers from customer meeting

27 February 2019

- Added the requirement part 1.1.3. Further changes will be made to this section after determining the details.

3 March 2019

- Updated non-functional requirements.

5 March 2019

- Proof-read and made minor corrections.

14 March 2019

- Guest user's access details are edited.
- Added a part about approval of learning materials. (1.2.7 edited)
- Updated registered users part about reviewing essays.

28 September 2019

- Removed accessibility
- Revised tags and reviews.
- Every user can suggest content, admins accept or reject them.
- Minor corrections.
- User search added.
- Liking and reporting content is added.

RestApiGuide.md

Language Learning Platform REST API Documentation

Test Endpoint

Greeting

This is created for testing purpose, say hi

```
GET /greeting?name=group9
```


```
{
  "id": 3,
  "content": "Hello, group9!"
}
```


User Endpoints

Get User By Id

Request Content: id

Response Content: The user with corresponding id if it exists, null otherwise.

Example Request

```
GET /users/get?id=2
```

Example Response


```
{
  "id": 2,
  "email": "ahmettest@testtest.coam",
  "password": "9f86d081884c7d659a2feaa0c55ad015a3bf4f1b2b0b822cd15d6c15b0f00a08",
  "firstName": "ahmet",
  "lastName": "test"
}
```


Register User

Request Content: e-mail, password, firstName, lastName

Response Content: The created user with 200 status code if successful. The reason with 400(Bad Request) if unsuccessful.

Example Request-1

```
POST /users/register
```

```
{
  "email": "test@test.com",
  "password": "123456",
  "firstName": "testname",
  "lastName": "testsurname"
}
```

Example Response-1

```
{
  "status": 200,
  "explanation": null,
  "data": {
    "email": "test@test.com",
    "password": "123456",
    "firstName": "testname",
    "lastName": "testsurname"
  }
}
```

Example Request-2

```
POST /users/register
```

```
{
  "email": "testtest.com",
  "password": "123456",
  "firstName": "testname",
  "lastName": "testsurname"
}
```

Example Response-2

```
{
  "status": 400,
  "explanation": "Invalid email",
  "data": null
}
```

Example Request-3

```
POST /users/register
```

```
{
  "email": "somebodyelsesemailaddress@test.com",
  "password": "123456",
  "firstName": "testname",
  "lastName": "testsurname"
}
```

Example Response-3

```
{
  "status": 400,
```

```
  "explanation": "This e-mail has already been registered",
  "data": null
}
```

Login User

Request Content: e-mail, password

Response Content: The user with 200 status code if successful. The reason with 400(Bad Request) if unsuccessful.

Example Request-1

```
POST /users/login
```

```
{
  "email": "test@fortest.com",
  "password": "123456"
}
```

Example Response-1

```
{
  "status": 200,
  "explanation": null,
  "data": {
      "id": 5,
      "email": "test@fortest.com",
      "password": "123456",
      "firstName": "testname",
      "lastName": "testsurname"
  }
}
```

Example Request-2

```
POST /users/login
```

```
{
  "email": "wrongemailaddress@wrong.email",
  "password": "wrongpassword"
}
```

Example Response-2

```
{
  "status": 400,
  "explanation": "Wrong credentials",
  "data": null
}
```

Get Profile Info by User Id

Request Content: id

Response Content: The profile info with 200 status code if successful. The reason with 400(Bad Request) if unsuccessful.

Example Request-1

```
GET /users/profile?id=6
```

Example Response-1

```
{
  "status": 200,
  "explanation": null,
  "data": {
    "userId": 6,
    "firstName": "testname",
    "lastName": "testsurname",
    "email": "test@test.com",
    "languages": [
      "English"
    ],
    "grades": [
      5
    ],
    "progressLevels": [
      0
    ]
  }
}
```

Example Request-2

```
GET /users/profile?id=66
```

Example Response-2

```
{
  "status": 404,
  "explanation": "User not found with id: 66",
  "data": null
}
```

Solved Exercise

Request Content: exercise id, user id

Response Content: The user with 200 status code if successful. The reason with 400(Bad Request) if unsuccessful.

Example Request-1

```
POST /users/solved


{
  "userId": 6,
  "exerciseId": "6"
}
```

Example Response-1

```
{
  "status": 200,
  "explanation": "Successful",
  "data": null
}
```


Content Endpoints

Get Exercise By Id

Request Content: exercise id

Response Content: Exercise with tags if successful, 404 not found if not.

Example Request

```
GET /contents?id=3
```

Example Response


```
{
  "status": 404,
  "explanation": "No exercise with this id.",
  "data": null
}
```


Example Request

```
GET /contents?id=13
```

Example Response


```
{
  "status": 200,
  "explanation": null,
  "data": {
    "id": 13,
    "languageId": 1,
    "grade": 2,
    "typeId": 2,
    "imageUrl": "",
    "soundUrl": "",
    "questionBody": "_______ he should have spent all the weekend preparing for his test, he in fact just lay in bed
watching videos.",
    "optionA": "however",
    "optionB": "whereas",
    "optionC": "despite",
    "optionD": "nevertheless",
    "correctAnswer": 2,
    "tags": [
      {
        "id": 4,
        "exerciseId": 13,
        "tagText": "lazy"
      },
      {
        "id": 3,
```

```
        "exerciseId": 13,
        "tagText": "video"
      },
      {
        "id": 6,
        "exerciseId": 13,
        "tagText": "tv"
      },
      {
        "id": 7,
        "exerciseId": 13,
        "tagText": "television"
      }
    ]
  }
}
```


Get All Available Languages List

Request Content: none

Response Content: The list of available languages

Example Request

```
GET /contents/languages
```

Example Response


```
{
  "status": 200,
  "explanation": null,
  "data": [
    "English",
    "Turkish",
    "Italian"
  ]
}
```


Get All Exercises List

Request Content: none

Response Content: The list of all exercises of all languages

Example Request

```
GET /contents/all
```

Example Response


```
{
    "status": 200,
    "explanation": null,
    "data": [
        {
            "languageId": 1,
            "typeId": 1,
```

```
            "imageUrl": null,
            "soundUrl": null,
            "question": "what is your name?",
            "optionA": "my name is..",
            "optionB": "your name is..",
            "optionC": "his name is...",
            "optionD": "her name is...",
            "correctAnswer": 1
        },
        {
          ...
        },

        ...
    ]
}
```


Get Proficiency Exam

Request Content: Language name as request parameter. Language names should be in English and capitalized.

Response Content: 10 questions from given language

Example Request

```
GET /contents/prof?language=English
```

Example Response


```
{
    "status": 200,
    "explanation": null,
    "data": [
        {
            "languageId": 1,
            "typeId": 1,
            "imageUrl": null,
            "soundUrl": null,
            "question": "what is your name?",
            "optionA": "my name is..",
            "optionB": "your name is..",
            "optionC": "his name is...",
            "optionD": "her name is...",
            "correctAnswer": 1
        },
        {
          ...
        },

        ...
    ]
}
```


Example Request

```
GET /contents/prof?language=english
```

Example Response

```
{
    "status": 400,
    "explanation": "Language not found.",
    "data": null
}
```


Create exercise

Request Content:

languageId(1 for English, 2 for Turkish, 3 for Italian)

typeId(1 for Grammar, 2 for vocabulary, 3 for reading, 4 for listening)

imageUrl(not necessary for proficiency, just delete)

soundUrl(not necessary for proficiency, just delete)

questionBody(As it sounds)

optionA, optionB, optionC, optionD(as it sounds)

correctAnswer(1 for A, 2 for B etc)

Only image url and sound url can be null

Response Content: Exercise itself

Example Request

```
POST /contents/add
```


```
{
  "correctAnswer": 4,
  "languageId": 1,
  "optionA": "Bean",
  "optionB": "Potato",
  "optionC": "Bread",
  "optionD": "Apple",
  "questionBody": "Which one of these is a fruit?",
  "typeId": 2
}
```


Example Response


```
{
    "status": 200,
    "explanation": null,
    "data": {
        "correctAnswer": 4,
        "imageUrl": null,
        "languageId": 1,
        "optionA": "Bean",
        "optionB": "Potato",
        "optionC": "Bread",
        "optionD": "Apple",
        "questionBody": "Which one of these is a fruit?",
        "soundUrl": null,
        "typeId": 2
```

```
        }
}
```

Delete exercise

Request Content:

Exercise id

Response Content: none

Example Request

```
GET /contents/delete?id=3
```

Example Response

```
{
    "status": 200,
    "explanation": null,
    "data": null
}
```

Grade Endpoints

Get Grade by UserId and LanguageId

Request Content: User id and language id

Response Content: Grade

Example Request

```
GET /grades/get?userId=3&languageId=1
```

Example Response

```
{
  "status": 200,
  "explanation": null,
  "data": {
    "id": 10,
    "userId": 12,
    "languageId": 1,
    "grade": 1
  }
}
```

Add Grade

Request Content: UserId, languageId, grade

Response Content: Grade itself

Example Request

```
POST /grades/add
```

```
{
  "grade": 4,
  "languageId": 1,
  "userId": 4
}
```

Example Response

```
{
  "status": 200,
  "explanation": null,
  "data": {
    "userId": 4,
    "languageId": 1,
    "grade": 4
  }
}
```

Search Endpoints

Search User

Request Content: First name, last name, grade and language

Response Content: List of users

Example Request

```
POST /search/users
```

```
{
  "firstName": "t",
  "grade": 3,
  "languageId": 1,
  "lastName": "te"
}
```

Example Response

```
{
  "status": 200,
  "explanation": null,
  "data": [
    {
      "userId": 3,
      "firstName": "ahmet",
      "lastName": "test",
      "email": "ahmettestttttt@testtest.coam",
      "languages": [
        "English"
      ],
      "grades": [
```

```
        3
      ],
      "progressLevels": [
        0
      ]
    },
    {
      "userId": 6,
      "firstName": "testname",
      "lastName": "testsurname",
      "email": "test@test.com",
      "languages": [
        "English"
      ],
      "grades": [
        5
      ],
      "progressLevels": [
        0
      ]
    }
  ]
}
```


Search Exercises

Request Content: UserId, languageId, grade, typeId, tag

Response Content: List of exercises which fits in given parameters; grade, type and language. Also if tag is given, returns only the exercises with tags semantically similar to the given one. Also if user id is given, returns the exercises which that user haven't solved yet.

Example Request

```
POST /search/exercises
```


```
{
  "grade": 2,
  "languageId": 1,
  "tag": "television",
  "typeId": 2,
  "userId": 7
}
```


Example Response


```
{
  "status": 200,
  "explanation": null,
  "data": [
    {
      "id": 13,
      "languageId": 1,
      "grade": 2,
      "typeId": 2,
      "imageUrl": "",
      "soundUrl": "",
      "questionBody": "_______ he should have spent all the weekend preparing for his test, he in fact just lay in bed
      "optionA": "however",
      "optionB": "whereas",
      "optionC": "despite",
      "optionD": "nevertheless",
```

```
      "correctAnswer": 2,
      "tags": [
        {
          "id": 4,
          "exerciseId": 13,
          "tagText": "lazy"
        },
        {
          "id": 3,
          "exerciseId": 13,
          "tagText": "video"
        },
        {
          "id": 6,
          "exerciseId": 13,
          "tagText": "tv"
        },
        {
          "id": 7,
          "exerciseId": 13,
          "tagText": "television"
        }
      ]
    }
  ]
}
```


Tag Endpoints

Add Tag

Request Content: Exercise id, tag text

Response Content: Tag itself

Example Request

```
POST /tags
```


```
{
  "exerciseId": 12,
  "tagText": "test"
}
```


Example Response


```
{
  "status": 200,
  "explanation": null,
  "data": {
    "id": 8,
    "exerciseId": 12,
    "tagText": "test"
  }
}
```


Invitation Endpoints

Add Invitation

Request Content: receiver id, source id

Response Content: Invitation itself

Example Request

```
POST /invitations
```

```
{
  "receiverId": 1,
  "sourceId": 5
}
```

Example Response

```
{
  "status": 200,
  "explanation": null,
  "data": {
    "id": 15,
    "sourceId": 5,
    "receiverId": 1,
    "createdAt": "2019-12-01T15:06:22.626+0000"
  }
}
```

Answer Invitation

Request Content: receiver id, source id, approved(boolean)

Response Content: Conversation itself

Example Request

```
POST /invitations/answer
```

```
{
  "approved": true,
  "receiverId": 1,
  "sourceId": 5
}
```

Example Response

```
{
  "status": 200,
  "explanation": null,
  "data": {
    "id": 11,
    "userIdOne": 5,
    "userIdTwo": 1,
    "lastUpdatedAt": "2019-12-01T15:08:05.896+0000"
```

```
  }
}
```

Example Response(if conversation already exists)

```
{
  "status": 400,
  "explanation": "Conversation already exists",
  "data": null
}
```

Get Inviter Profile Infos

Request Content: receiver user id

Response Content: List of profile infos

Example Request

```
GET /invitations/byReceiverId?userId=1
```

Example Response

```
{
  "status": 200,
  "explanation": null,
  "data": [
    {
      "userId": 6,
      "firstName": "testname",
      "lastName": "testsurname",
      "email": "test@test.com",
      "languages": [
        "English"
      ],
      "grades": [
        5
      ],
      "progressLevels": [
        0
      ]
    }
  ]
}
```

Get Invitation State

Request Content: userId1, userId2

Response Content: Invitation state

Example Request

```
GET /invitations/state?userId1=1&userId2=6
```

Example Response

```
{
  "status": 200,
  "explanation": null,
  "data": {
    "userId1": 1,
    "userId2": 6,
    "pendingRequestFromOneToTwo": false,
    "pendingRequestFromTwoToOne": true,
    "startedConversation": false
  }
}
```


Conversation Endpoints

Get Conversation Profile Info

Request Content: user id

Response Content: List of profile infos of given user id

Example Request

```
GET /conversations?id=1
```

Example Response


```
{
  "status": 200,
  "explanation": null,
  "data": [
    {
      "userId": 5,
      "firstName": "testname",
      "lastName": "testtest",
      "email": "test@fortest.com",
      "languages": [],
      "grades": [],
      "progressLevels": []
    },
    {
      "userId": 2,
      "firstName": "ahmet",
      "lastName": "test",
      "email": "ahmettest@testtest.coam",
      "languages": [],
      "grades": [],
      "progressLevels": []
    },
    {
      "userId": 3,
      "firstName": "ahmet",
      "lastName": "test",
      "email": "ahmettesttttt@testtest.coam",
      "languages": [
        "English"
      ],
      "grades": [
        3
      ],
      "progressLevels": [
```

```
        0
      ]
    },
    {
      "userId": 4,
      "firstName": "ahmet",
      "lastName": "test",
      "email": "ahmettestttttttt@testtest.coam",
      "languages": [
        "English"
      ],
      "grades": [
        1
      ],
      "progressLevels": [
        25
      ]
    }
  ]
}
```


Message Endpoints

Get Messages By User Id

Request Content: user id

Response Content: List of Messages of the User

Example Request

```
GET /messages?userId=9
```

Example Response


```
{
  "status": 200,
  "explanation": null,
  "data": [
    {
      "id": 10,
      "sourceId": 9,
      "receiverId": 2,
      "content": "selam",
      "createdAt": "2019-11-24T14:49:07.090+0000"
    },
    {
      "id": 11,
      "sourceId": 2,
      "receiverId": 9,
      "content": "selam",
      "createdAt": "2019-11-24T14:49:19.480+0000"
    },
    {
      "id": 12,
      "sourceId": 2,
      "receiverId": 9,
      "content": "nabber",
      "createdAt": "2019-11-24T14:49:28.963+0000"
    },
    {
```

```
  "id": 13,
  "sourceId": 2,
  "receiverId": 9,
  "content": "nabber",
  "createdAt": "2019-11-24T14:49:30.561+0000"
},
{
  "id": 14,
  "sourceId": 2,
  "receiverId": 9,
  "content": "nabber",
  "createdAt": "2019-11-24T14:49:31.344+0000"
},
{
  "id": 15,
  "sourceId": 2,
  "receiverId": 9,
  "content": "nabber",
  "createdAt": "2019-11-24T15:55:51.799+0000"
},
{
  "id": 16,
  "sourceId": 9,
  "receiverId": 2,
  "content": "asffghff",
  "createdAt": "2019-11-24T16:01:23.690+0000"
},
{
  "id": 17,
  "sourceId": 9,
  "receiverId": 2,
  "content": "asdfghjkli",
  "createdAt": "2019-11-24T16:01:34.912+0000"
},
{
  "id": 18,
  "sourceId": 9,
  "receiverId": 2,
  "content": "ibrahm",
  "createdAt": "2019-11-24T16:01:46.860+0000"
},
{
  "id": 19,
  "sourceId": 9,
  "receiverId": 2,
  "content": "iyi",
  "createdAt": "2019-11-24T16:18:16.600+0000"
},
{
  "id": 20,
  "sourceId": 9,
  "receiverId": 2,
  "content": "çok güzel mesajlaşma",
  "createdAt": "2019-11-24T16:56:26.617+0000"
},
{
  "id": 21,
  "sourceId": 9,
  "receiverId": 2,
  "content": "yfgjomö",
  "createdAt": "2019-11-24T17:02:18.924+0000"
},
{
  "id": 22,
  "sourceId": 9,
  "receiverId": 2,
  "content": "irem",
  "createdAt": "2019-11-24T17:09:54.875+0000"
```

```
},
{
  "id": 23,
  "sourceId": 37,
  "receiverId": 9,
  "content": "Selam İremm ",
  "createdAt": "2019-11-25T04:51:11.692+0000"
},
{
  "id": 24,
  "sourceId": 37,
  "receiverId": 9,
  "content": "Deneme yapıyorum :D ",
  "createdAt": "2019-11-25T04:51:39.463+0000"
},
{
  "id": 25,
  "sourceId": 9,
  "receiverId": 37,
  "content": "selam ",
  "createdAt": "2019-11-25T04:53:31.176+0000"
},
{
  "id": 26,
  "sourceId": 37,
  "receiverId": 9,
  "content": "Günaydın irem",
  "createdAt": "2019-11-25T10:39:09.810+0000"
},
{
  "id": 27,
  "sourceId": 37,
  "receiverId": 9,
  "content": "Ben de ibrahim :D",
  "createdAt": "2019-11-25T10:39:16.587+0000"
},
{
  "id": 28,
  "sourceId": 9,
  "receiverId": 37,
  "content": "Ben de game :D\n",
  "createdAt": "2019-11-25T10:39:34.911+0000"
},
{
  "id": 29,
  "sourceId": 37,
  "receiverId": 9,
  "content": "Ama saat sıkıntılı :D",
  "createdAt": "2019-11-25T10:40:12.159+0000"
},
{
  "id": 30,
  "sourceId": 37,
  "receiverId": 9,
  "content": "13 40 yazıyor",
  "createdAt": "2019-11-25T10:40:24.313+0000"
},
{
  "id": 31,
  "sourceId": 9,
  "receiverId": 8,
  "content": "selam gg",
  "createdAt": "2019-11-25T10:40:30.933+0000"
},
{
  "id": 32,
  "sourceId": 37,
  "receiverId": 9,
```

```
      "content": "Halbuki 16 40",
      "createdAt": "2019-11-25T10:40:36.387+0000"
    },
    {
      "id": 33,
      "sourceId": 9,
      "receiverId": 8,
      "content": "deneme",
      "createdAt": "2019-11-25T17:17:43.781+0000"
    }
  ]
}
```


Get Conversation Content

Request Content: user id1, user id2

Response Content: List of Messages of the conversation

Example Request

```
GET /messages/chat?userId1=9&userId2=8
```

Example Response


```
{
  "status": 200,
  "explanation": null,
  "data": [
    {
      "id": 31,
      "sourceId": 9,
      "receiverId": 8,
      "content": "selam gg",
      "createdAt": "2019-11-25T10:40:30.933+0000"
    },
    {
      "id": 33,
      "sourceId": 9,
      "receiverId": 8,
      "content": "deneme",
      "createdAt": "2019-11-25T17:17:43.781+0000"
    }
  ]
}
```


Create Message

Request Content: sourceId, receiverId, content

Response Content: Conversation content between two users

Example Request

```
POST /messages
```


```
{
  "content": "this is a message",
  "receiverId": 9,
```

```
  "sourceId": 8
}
```

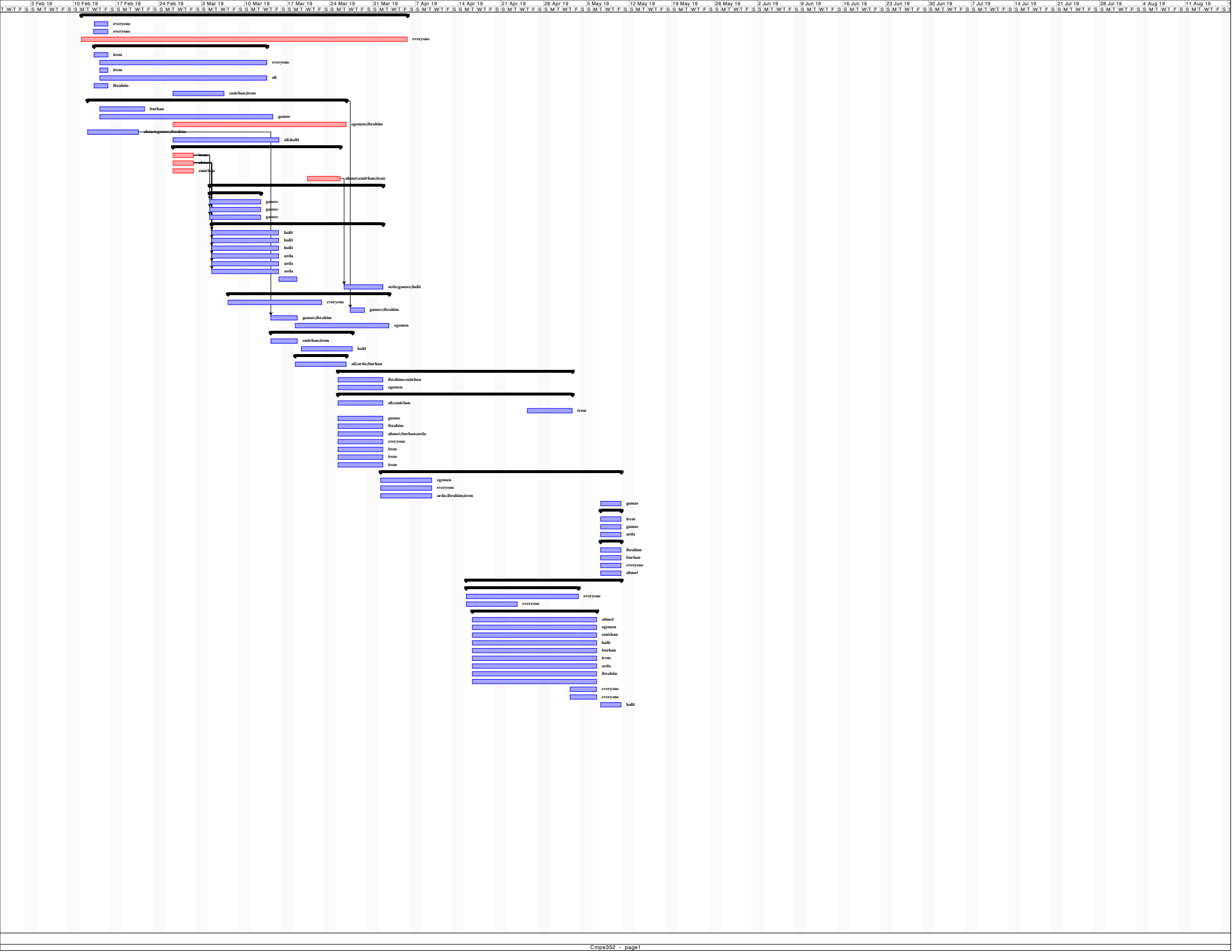
Example Response

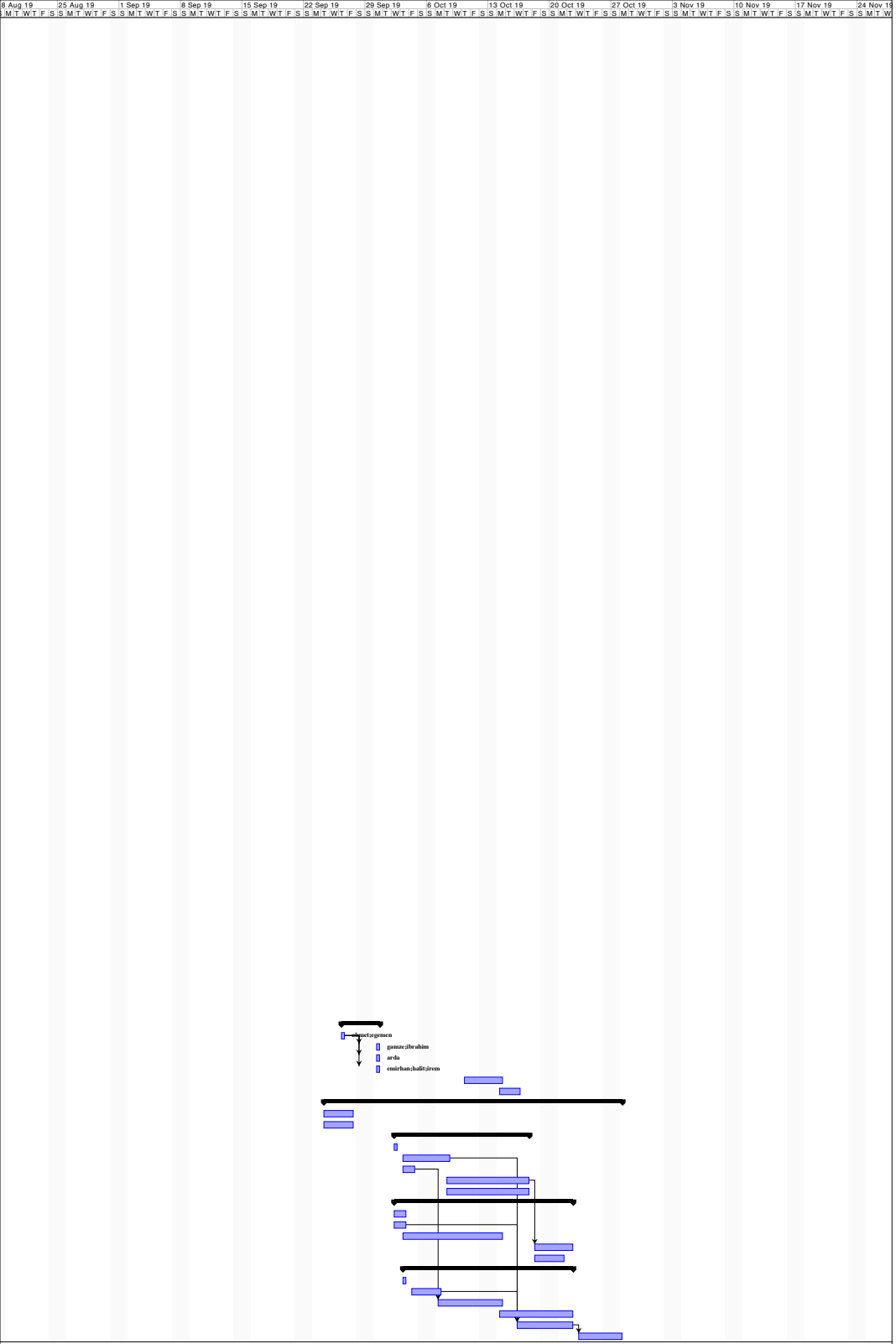
```
{
  "status": 200,
  "explanation": null,
  "data": [
    {
      "id": 31,
      "sourceId": 9,
      "receiverId": 8,
      "content": "selam gg",
      "createdAt": "2019-11-25T10:40:30.933+0000"
    },
    {
      "id": 33,
      "sourceId": 9,
      "receiverId": 8,
      "content": "deneme",
      "createdAt": "2019-11-25T17:17:43.781+0000"
    },
    {
      "id": 42,
      "sourceId": 8,
      "receiverId": 9,
      "content": "this is a message",
      "createdAt": "2019-12-01T16:20:34.554+0000"
    }
  ]
}
```

	Name	Duration	Start	Finish	Predecessors	Resource Names
1	**Preliminary Search**	**40 days?**	**2/11/19 8:00 AM**	**4/5/19 5:00 PM**		
2	Learn Markdown	2.665 days?	2/13/19 10:41 AM	2/15/19 5:00 PM		everyone
3	Learn basic git commands	3 days?	2/13/19 8:00 AM	2/15/19 5:00 PM		everyone
4	Find and document fine git repos	40 days?	2/11/19 8:00 AM	4/5/19 5:00 PM		everyone
5	**Documentation**	**20.835 days?**	**2/13/19 9:19 AM**	**3/13/19 5:00 PM**		
6	Create communication plan	2.835 days?	2/13/19 9:19 AM	2/15/19 5:00 PM		irem
7	Personal wiki pages	20 days?	2/14/19 8:00 AM	3/13/19 5:00 PM		everyone
8	Meeting agenda creation	2 days?	2/14/19 8:00 AM	2/15/19 5:00 PM		irem
9	Create and manage issues every week	20 days?	2/14/19 8:00 AM	3/13/19 5:00 PM		ali
10	Create readme.md	2.835 days?	2/13/19 9:19 AM	2/15/19 5:00 PM		ibrahim
11	Add little icons to sidebar	7 days?	2/26/19 8:00 AM	3/6/19 5:00 PM		emirhan;irem
12	**Requirements**	**31 days?**	**2/12/19 8:00 AM**	**3/26/19 5:00 PM**		
13	Edit the questions to customer to create final draft	6 days?	2/14/19 8:00 AM	2/21/19 5:00 PM		burhan
14	Edit project requirements to creater final draft	21 days?	2/14/19 8:00 AM	3/14/19 5:00 PM		gamze
15	Review and update the requirements	21 days?	2/26/19 8:00 AM	3/26/19 5:00 PM		egemen;ibrahim
16	Update Requirements according to review and Q&A	7 days	2/12/19 8:00 AM	2/20/19 5:00 PM		ahmet;gamze;ibrahim
17	Create Glossary	14 days	2/26/19 8:00 AM	3/15/19 5:00 PM		ali;halit
18	**User Personas and Scenerios**	**20 days?**	**2/26/19 8:00 AM**	**3/25/19 5:00 PM**		
19	create a user persona and a scenerio for a university student	4 days?	2/26/19 8:00 AM	3/1/19 5:00 PM		irem
20	create a user persona and a scenerio for a caretaker	4 days?	2/26/19 8:00 AM	3/1/19 5:00 PM		ahmet
21	create a user persona and a scenerio for a musician	4 days?	2/26/19 8:00 AM	3/1/19 5:00 PM		emirhan
22	Split user scenarios	4 days?	3/20/19 8:00 AM	3/25/19 5:00 PM		ahmet;emirhan;irem
23	**Mock-ups**	**21 days?**	**3/4/19 8:00 AM**	**4/1/19 5:00 PM**		
24	**Web Mockup**	**7 days?**	**3/4/19 8:00 AM**	**3/12/19 5:00 PM**		
25	Login	7 days?	3/4/19 8:00 AM	3/12/19 5:00 PM	19	gamze
26	Take Exercise	7 days?	3/4/19 8:00 AM	3/12/19 5:00 PM	19	gamze
27	Advanced Search	7 days?	3/4/19 8:00 AM	3/12/19 5:00 PM	19	gamze
28	**Android Mockup**	**20.04 days?**	**3/4/19 4:41 PM**	**4/1/19 5:00 PM**		
29	Guest User	9 days	3/4/19 4:41 PM	3/15/19 4:41 PM	20	halit
30	Register	9 days?	3/4/19 4:41 PM	3/15/19 4:41 PM	20	halit
31	Beginner Exercises	9 days?	3/4/19 4:41 PM	3/15/19 4:41 PM	20	halit
32	Login	9 days?	3/4/19 4:41 PM	3/15/19 4:41 PM	20	arda
33	Chat	9 days?	3/4/19 4:41 PM	3/15/19 4:41 PM	20	arda
34	Grading Essays	9 days?	3/4/19 4:41 PM	3/15/19 4:41 PM	20	arda
35	Revise	1 day?	3/15/19 4:41 PM	3/18/19 4:41 PM		
36	Split mock-ups	5 days?	3/26/19 8:00 AM	4/1/19 5:00 PM	22	arda;gamze;halit
37	**Use Case Diagram**	**19 days?**	**3/7/19 8:00 AM**	**4/2/19 5:00 PM**		
38	Learning to create Use Case Diagram	12 days?	3/7/19 8:00 AM	3/22/19 5:00 PM		everyone
39	Use Case Diagram	3 days?	3/27/19 8:00 AM	3/29/19 5:00 PM	12	gamze;ibrahim
40	Update the Use Case Diagram as per updated Requirements	3 days?	3/14/19 8:00 AM	3/18/19 5:00 PM	16	gamze;ibrahim
41	Check and Review the Use Case Diagram	12 days?	3/16/19 8:00 AM	4/2/19 5:00 PM		egemen
42	**Class Diagram**	**10 days?**	**3/14/19 8:00 AM**	**3/27/19 5:00 PM**		
43	Create Class Diagram	3 days?	3/14/19 8:00 AM	3/18/19 5:00 PM		emirhan;irem
44	Check and Review Class Diagram	7 days?	3/19/19 8:00 AM	3/27/19 5:00 PM		halit

	Name	Duration	Start	Finish	Predecessors	Resource Names
45	**Sequence Diagrams**	**7 days?**	**3/18/19 8:00 AM**	**3/26/19 5:00 PM**		
46	Create Sequence Diagrams	7 days?	3/18/19 8:00 AM	3/26/19 5:00 PM		ali;arda;burhan
47	**Milestone1**	**29 days?**	**3/25/19 8:00 AM**	**5/2/19 5:00 PM**		
48	Executive Summary	6 days?	3/25/19 8:00 AM	4/1/19 5:00 PM		ibrahim;emirhan
49	List and Status Deliverables	6 days?	3/25/19 8:00 AM	4/1/19 5:00 PM		egemen
50	**Evaluation**	**29 days?**	**3/25/19 8:00 AM**	**5/2/19 5:00 PM**		
51	Evaluate requirements and mockups	6 days?	3/25/19 8:00 AM	4/1/19 5:00 PM		ali;emirhan
52	Evaluate Use-Case Diagram	6 days?	4/25/19 8:00 AM	5/2/19 5:00 PM		irem
53	Evaluate Class Diagram	6 days?	3/25/19 8:00 AM	4/1/19 5:00 PM		gamze
54	Evaluate Sequence Diagrams	6 days?	3/25/19 8:00 AM	4/1/19 5:00 PM		ibrahim
55	Evaluate project repository	6 days?	3/25/19 8:00 AM	4/1/19 5:00 PM		ahmet;burhan;arda
56	Summary of work	6 days?	3/25/19 8:00 AM	4/1/19 5:00 PM		everyone
57	Communication plan	6 days?	3/25/19 8:00 AM	4/1/19 5:00 PM		irem
58	Requirements and mockups	6 days?	3/25/19 8:00 AM	4/1/19 5:00 PM		irem
59	Design	6 days?	3/25/19 8:00 AM	4/1/19 5:00 PM		irem
60	**Project Plan**	**30 days?**	**4/1/19 8:00 AM**	**5/10/19 5:00 PM**		
61	Create an excel file for project plan	7 days?	4/1/19 8:00 AM	4/9/19 5:00 PM		egemen
62	Add their partitioned weeks	7 days?	4/1/19 8:00 AM	4/9/19 5:00 PM		everyone
63	Merge on Project Libre	7 days?	4/1/19 8:00 AM	4/9/19 5:00 PM		arda;ibrahim;irem
64	Add new works and update according to feedback	4 days?	5/7/19 8:00 AM	5/10/19 5:00 PM		gamze
65	**Milestone 2**	**4 days?**	**5/7/19 8:00 AM**	**5/10/19 5:00 PM**		
66	Executive Summary	4 days?	5/7/19 8:00 AM	5/10/19 5:00 PM		irem
67	Project Plan	4 days?	5/7/19 8:00 AM	5/10/19 5:00 PM		gamze
68	List and status of deliverables	4 days?	5/7/19 8:00 AM	5/10/19 5:00 PM		arda
69	**Evaluation**	**4 days?**	**5/7/19 8:00 AM**	**5/10/19 5:00 PM**		
70	Evaluate the status of deliverables	4 days	5/7/19 8:00 AM	5/10/19 5:00 PM		ibrahim
71	Evaluate the tools and processes	4 days?	5/7/19 8:00 AM	5/10/19 5:00 PM		burhan
72	Summary of personal work	4 days?	5/7/19 8:00 AM	5/10/19 5:00 PM		everyone
73	Merge the whole Milestone report	4 days?	5/7/19 8:00 AM	5/10/19 5:00 PM		ahmet
74	**Sample API Implementation**	**20 days?**	**4/15/19 8:00 AM**	**5/10/19 5:00 PM**		
75	**Preliminary Search**	**15 days?**	**4/15/19 8:00 AM**	**5/3/19 5:00 PM**		
76	Learn Django framework	15 days?	4/15/19 8:00 AM	5/3/19 5:00 PM		everyone
77	Create Amazon Account	7 days?	4/15/19 8:00 AM	4/23/19 5:00 PM		everyone
78	**Funtionality of our Api**	**15 days?**	**4/16/19 8:00 AM**	**5/6/19 5:00 PM**		
79	word of the day	15 days?	4/16/19 8:00 AM	5/6/19 5:00 PM		ahmet
80	inappropriate word detection	15 days?	4/16/19 8:00 AM	5/6/19 5:00 PM		egemen
81	tag detection	15 days?	4/16/19 8:00 AM	5/6/19 5:00 PM		emirhan
82	translate	15 days?	4/16/19 8:00 AM	5/6/19 5:00 PM		halit
83	language detection according to location	15 days?	4/16/19 8:00 AM	5/6/19 5:00 PM		burhan
84	object detection (photo to foreign text)	15 days?	4/16/19 8:00 AM	5/6/19 5:00 PM		irem
85	random multimedia recommendation	15 days?	4/16/19 8:00 AM	5/6/19 5:00 PM		arda
86	language detection	15 days?	4/16/19 8:00 AM	5/6/19 5:00 PM		ibrahim
87	text-to-speech	15 days?	4/16/19 8:00 AM	5/6/19 5:00 PM		
88	Documentation	3 days?	5/2/19 8:00 AM	5/6/19 5:00 PM		everyone

	Name	Duration	Start	Finish	Predecessors	Resource Names
89	Review each other's codes	3 days?	5/2/19 8:00 AM	5/6/19 5:00 PM		everyone
90	Merge all functionalities	4 days?	5/7/19 8:00 AM	5/10/19 5:00 PM		halit
91	**Review Previous Work**	**3 days?**	**9/26/19 8:00 AM**	**9/30/19 5:00 PM**		
92	Review Requirements	1 day?	9/26/19 8:00 AM	9/26/19 5:00 PM		ahmet;egemen
93	Review Diagrams	1 day?	9/28/19 8:00 AM	9/30/19 5:00 PM	92	gamze;ibrahim
94	Review MockUps	1 day?	9/28/19 8:00 AM	9/30/19 5:00 PM	92	arda
95	Review Project Plan	1 day?	9/28/19 8:00 AM	9/30/19 5:00 PM	92	emirhan;halit;irem
96	Decide on a name for the project	3 days	10/10/19 8:00 AM	10/14/19 5:00 PM		
97	Create a project logo	3 days?	10/12/19 8:00 AM	10/16/19 5:00 PM		
98	**Milestone 3**	**24.001 days?**	**9/24/19 8:00 AM**	**10/28/19 8:00 AM**		
99	Research about which deployment methodologies to use	4 days	9/24/19 8:00 AM	9/27/19 5:00 PM		
100	Research technologies to use in implementation	4 days	9/24/19 8:00 AM	9/27/19 5:00 PM		
101	**Backend**	**12 days?**	**10/2/19 8:00 AM**	**10/17/19 5:00 PM**		
102	Develop a simple Spring project	1 day?	10/2/19 8:00 AM	10/2/19 5:00 PM		
103	Deploy it & Automate Deployment	4 days?	10/3/19 8:00 AM	10/8/19 5:00 PM		
104	Create Login/Register API	2 days?	10/3/19 8:00 AM	10/4/19 5:00 PM		
105	Create Language/Proficiency/Exercise API	8 days?	10/8/19 8:00 AM	10/17/19 5:00 PM		
106	Create API Docs & Swagger UI	8 days?	10/8/19 8:00 AM	10/17/19 5:00 PM		
107	**Frontend**	**15 days?**	**10/2/19 8:00 AM**	**10/22/19 5:00 PM**		
108	Develop a simple web interface	2 days?	10/2/19 8:00 AM	10/3/19 5:00 PM		
109	Deploy it & Automate Deployment	2 days?	10/2/19 8:00 AM	10/3/19 5:00 PM		
110	Create Login/Register Pages & Flow	8 days?	10/3/19 8:00 AM	10/14/19 5:00 PM		
111	Create Language Selection, Proficiency Exam Pages & API Connection	3 days?	10/18/19 8:00 AM	10/22/19 5:00 PM	105	
112	Make a second deployment & CI/CD for testing next version	2 days?	10/18/19 8:00 AM	10/21/19 5:00 PM		
113	**Android**	**14 days?**	**10/3/19 8:00 AM**	**10/22/19 5:00 PM**		
114	Develop a simple Android application	1 day?	10/3/19 8:00 AM	10/3/19 5:00 PM		
115	Take APK export and share it within team	2 days?	10/4/19 8:00 AM	10/7/19 5:00 PM		
116	Create Login/Register Screens & Flow & API Conection	6 days?	10/7/19 8:00 AM	10/14/19 5:00 PM	104	
117	Create Laguage Selection, Proficiency Exam, Grade Screens	7 days?	10/12/19 8:00 AM	10/22/19 5:00 PM		
118	Preparing Milestone 3 Demos & Presentation	5 days?	10/16/19 8:00 AM	10/22/19 5:00 PM	103;109;115	
119	Preparing Executive Summary, Deliverables and Evalutation	3.001 days?	10/23/19 8:00 AM	10/28/19 8:00 AM	118	


3 Feb 19
10 Feb 19
17 Feb 19
24 Feb 19
3 Mar 19
10 Mar 19
17 Mar 19
24 Mar 19
31 Mar 19
7 Apr 19
14 Apr 19
21 Apr 19
28 Apr 19
5 May 19
12 May 19
19 May 19
26 May 19
2 Jun 19
9 Jun 19
16 Jun 19
23 Jun 19
30 Jun 19
7 Jul 19
14 Jul 19
21 Jul 19
28 Jul 19
4 Aug 19
11 Aug 19
everyone
everyone
everyone
irem
everyone
irem
ali
ibrahim
emirhan;irem
burhan
gamze
egemen;ibrahim
ahmet;gamze;ibrahim
ali;halit
irem
ahmet
emirhan
ahmet;emirhan;irem
gamze
gamze
gamze
halit
halit
halit
arda
arda
arda
arda;gamze;halit
everyone
gamze;ibrahim
gamze;ibrahim
egemen
emirhan;irem
halit
ali;arda;burhan
ibrahim;emirhan
egemen
ali;emirhan
irem
gamze
ibrahim
ahmet;burhan;arda
everyone
irem
irem
irem
egemen
everyone
arda;ibrahim;irem
gamze
irem
gamze
arda
ibrahim
burhan
everyone
ahmet
everyone
everyone
ahmet
egemen
emirhan
halit
burhan
irem
arda
ibrahim
everyone
everyone
halit


8 Aug 19
25 Aug 19
1 Sep 19
8 Sep 19
15 Sep 19
22 Sep 19
29 Sep 19
6 Oct 19
13 Oct 19
20 Oct 19
27 Oct 19
3 Nov 19
10 Nov 19
17 Nov 19
24 Nov 19
ahmet;egemen
gamze;ibrahim
arda
emirhan;halit;irem

User Scenarios

1. Mahmut Kızıloğlu

- Short Bio:

Mahmut Kızıloğlu is a 25 year old waiter working at a doner shop, Hızlı Döner, Taksim, Istanbul. Mahmut never had a decent English education ever since his youth. Recently, with the increasing number of tourists coming to the doner shop, Mahmut's boss decided that all waiters must know English at some level. If Mahmut won't be able to improve his skills in English within 2 months, he will be fired. During a casual chat between waiters, a friend, Doğan Çavdarcı, suggested Mahmut to learn English through Kereviz. Thus, Mahmut decided to try it out.

- Platform: Web
- User Type: Registered
- Language of Interest: English

Scenario

1. Mahmut wants to solve exercises, so he moves to listening exercises page and starts solving listening exercises.
2. After solving listening exercises, he moves to reading exercises page and starts solving reading exercises.
3. After solving reading exercises, he moves to reading exercises page and starts solving grammar exercises.
4. After solving grammar exercises, he moves to reading exercises page and starts solving vocabulary exercises.
5. After solving vocabulary exercises, he moves to search exercises page.
6. He searches sports, since he is interested in sports and checks out the listed exercises.
7. Then, he goes to users page and searches for his friend, Doğan Çavdarcı.
8. He goes to his friend's profile and sends a chat request.

2. Francisco Tárrega

- Short Bio:

Francisco is a guitar player from Mexico who occasionally performs in local bars in Guadalajara. He had a decent English education thanks to his private high school. But due to education system, he knows grammar very well, but he cannot speak with a person in daily life. He wants to improve his listening skills and daily conversation, also due to his nationality, he is a native Spanish speaker. While surfing on the Internet, he finds Kereviz and decided to try it.

- Platform: Android
- User Type: Registered
- Language of Interest: English

Scenario

1. Francisco wants to solve exercises, so he moves to exercise pages and solves vocabulary exercises.
2. After solving exercises, he returns to his profile page and sees his progress in profile page.
3. He is interested in cinema, so he wants to search exercises which are related to cinema.
4. Clicks search button and using content search, he reaches exercises which is related to cinema.
5. He wants to say hi to his friend Alex, so he clicks chats page to send message to his friend.
6. After talking with Alex, he wants to search his friend from college, Mary Jane, so he searches her name and finds her profile.
7. After reaching her profile, he sends a message request to her to start a conversation.

8.md

Code Structure and Group Process

We have the `master` branch and our beloved `frontend` and `android` branches for holding the best code from each user-faced subteam. There are also feature branches and issue branches.

Front-End

Android

We used Github's Issues and Pull Requests effectively in our opinion. Our base branch was `/android` , we developed new features in our new branches like `/android-gradeview-enhancement` , and after completed developing a new feature, we opened pull request and our code reviewed by two peer most of the time and then merged. After completion of Milestone 2, we merged our `/android` branch to master.

We usually managed our group work by talking and dividing the work between each other. We talked about which part to be implemented and divide the part. Then we would together or on our own implement the part and send each other code reviews. We are happy about the communication between our own Android team but we need to improve our communication with other teams as well, because in some cases there exists incompatible practices arises in UI.

Back-End

As the Back-End team, we use Github's issue system and we create a new branch for each issue with the issue id since we want it to be unique. After the development and testing is done, we did pull request, did code review on Github, then merged the branch into the master.

9.md

Evaluation of Tools and Managing the Project

Backend

On backend side we used `Spring Framework` with Java. Throughout the development we used free student version of `IntelliJ`. It has many different tools to provide us a fast development environment. We are glad for this choice. We used `Postman` and `Swagger UI` to test out API. Both of them are easy to use.

For deployment we are using `AWS EC2`. We created our CI/CD pipeline with `Travis`. Our database lives in a database instance on `AWS RDS`. We created an instance of `Postgresql` database and currently using it. All services of Amazon have a good documentation and performance.

Frontend

Android

Android Studio

We are using Android Studio to write our app. After the first milestone, we have learned more about Android Studio and I think we can use it more efficiently now. We were adding log messages for debugging before the previous milestone. Lately, we have learned about the debug tool in Android Studio and we started to use it. It enables us to see the values of the variables and it is faster to use than adding log messages. I believe we started to use Android Studio better now.

Github

I think we are continuously improving on using Github more efficiently. After the previous milestone, we started to write longer commit messages and longer issue descriptions. We also have started to use project feature in Github. We created a project for Android Kereviz and we are adding our issues to there so it is easier to follow the process. One matter that I think we can improve is the code reviews that we are doing. Because we are usually getting together and writing the code together, we ask our questions about the code face to face and solve them together. When someone has a pull request and we are reviewing it, we usually don't make about the code as we have already seen the code when it is being written. Currently, this doesn't create any problem for us but I don't think it is a very good habit to have.

Java

We have chosen Java language to implement our Android project. We are generally happy about Java but we have only one problem with it. We usually have synchronization problems when we are sending a request to the backend. We heard that this is not a problem in Kotlin but we couldn't solve this problem with Java.