CmpE451

Dec 2019

Group10

Final Milestone Report

Contents

1 Executive Summary **5**
- 1.1 Project Information . . . 5
- 1.2 Summary of Project Status . . . 5
  - 1.2.1 Milestone1 . . . 5
  - 1.2.2 Milestone2 . . . 5
  - 1.2.3 Final Milestone . . . 6
- 1.3 What is next? . . . 6

2 List and Status of Deliverables **7**

3 Evaluation of the Status of Deliverables and Their Impact on Plan **7**

4 All Design Documents **8**
- 4.1 Class diagram . . . 8
- 4.2 Use case diagram . . . 9
- 4.3 Sequence diagram . . . 9
- 4.4 Mockup . . . 11

5 Work-Done by Team Members **12**

6 Annotation implementation & W3C standard compliance **13**
- 6.1 Annotation on a Textual Object . . . 14
- 6.2 Annotation on an Image Object . . . 15

7 Requirements **16**
- 7.1 Glossary . . . 16
- 7.2 Functional Requirements . . . 16
  - 7.2.1 User Requirements . . . 16
  - 7.2.2 System Requirements . . . 18
- 7.3 Non-Functional Requirements . . . 18

8 API Documentation **19**
- 8.1 Annotation . . . 19
  - 8.1.1 /annotation/id . . . 19
  - 8.1.2 /annotation/all . . . 19
  - 8.1.3 /annotation/all/writingId . . . 20
  - 8.1.4 /annotation/create . . . 21
  - 8.1.5 /annotation/delete . . . 22
  - 8.1.6 /annotation/image/id . . . 22
  - 8.1.7 /annotation/image/all . . . 23
  - 8.1.8 /annotation/image/create . . . 24
  - 8.1.9 /annotation/image/delete . . . 25
  - 8.1.10 /annotation/update . . . 26
- 8.2 Authentication and Registration . . . 27
  - 8.2.1 /authenticate . . . 27
  - 8.2.2 /register . . . 27
- 8.3 Member Operations . . . 28
  - 8.3.1 /member/profile . . . 28
  - 8.3.2 /member/addlang . . . 29
  - 8.3.3 /member/removelang . . . 30
  - 8.3.4 /member/memberId . . . 31
  - 8.3.5 /member/update . . . 32
  - 8.3.6 /member/profileImage . . . 33
- 8.4 Language Operations . . . 34

8.4.1 /lang . . . 34
8.4.2 /lang/unsubs . . . 34
8.5 Quiz Operations . . . 35
8.5.1 /quiz/quizId . . . 35
8.5.2 /quiz/level . . . 36
8.5.3 /quiz/quizId/submit . . . 37
8.5.4 /quiz . . . 38
8.5.5 /quiz/language/languageId . . . 39
8.5.6 /quiz/level/level/language/languageId . . . 40
8.5.7 /quiz/level/levelId . . . 41
8.5.8 /quiz/levelorlower/level/language/languageId . . . 42
8.6 Search Operations . . . 43
8.6.1 /search/quiz/languageId/term . . . 43
8.6.2 /search/writing/languageId/term . . . 44
8.7 Writing Operations . . . 45
8.7.1 /writing/writingId . . . 45
8.7.2 /writing/writingId/submit . . . 46
8.7.3 /writing/completedAssignments . . . 47
8.7.4 /writing/nonCompletedAssignments . . . 48
8.7.5 /writing/getJson/languageId . . . 49
8.7.6 /writing/read/writingId . . . 50
8.7.7 /writing/scores . . . 51
8.7.8 /writing/score/writingResultId . . . 52
8.7.9 /writing/uploadWritingImage . . . 53
8.7.10 /writing/writingId/submitWithImageURL . . . 53
8.8 Comment . . . 54
8.8.1 /comment . . . 54
8.8.2 /comment/memberId . . . 55
8.8.3 /comment/delete . . . 56
8.8.4 /comment/make . . . 56
8.8.5 /comment/rating . . . 57
8.8.6 /comment/rating/memberId . . . 57
8.8.7 /comment/update . . . 58
8.9 Image . . . 59
8.9.1 /image/deleteFile . . . 59
8.9.2 /image/upload . . . 59
8.10 Message . . . 60
8.10.1 /message/conversationId . . . 60
8.10.2 /message/conversations . . . 61
8.10.3 /message/send . . . 62
8.11 Notifications . . . 63
8.11.1 /notification/make/read . . . 63
8.11.2 /notification/make/not/read . . . 64
8.11.3 /notification/read . . . 64
8.12 Reporting . . . 65
8.12.1 /report/causes . . . 65
8.12.2 /report/send . . . 65

9 Project Plan **66**

10 User Scenarios **67**
10.1 Android Scenario . . . 67
10.2 Frontend Scenario . . . 67

11 Code Structure and Group Process **68**
11.1 Backend 68
11.2 Frontend 69
11.3 Android 69

12 Evaluation of Tools and Processes **72**

1 Executive Summary

1.1 Project Information

Our app, "YALLP", is a platform where people can improve their language skills via learning from the materials which are carefully selected from the creators of the app. All the materials are analyzed and inspected by site admins. The solely goal of this app is transforming learning of a language into something as easy as playing candy crush. Users will be able to learn language while using the metro, waiting a friend, spending time in the bathroom, anywhere you can imagine. Any user can contact with an expert to talk or discuss about anything. Since, daily speech is a part of learning, messaging between users is also another benefit "YALLP" provides to its users. We all know the famous lines from Cem Yilmaz "Speak English? I live in English". That's exactly what we are aiming for.

1.2 Summary of Project Status

At the start of this project, we began from the bottom. What we do first is arranging teams. Afterwards, each week task by task we developed our application in every aspects. On the other hand, we discussed what we should not develop. This was also crucial since we have limited time, requirements had to be renewed. Early weeks we could not decide what to implement what to discard. Once we got developed it appeared clearer. Each team worked hard and did their parts so far. In this section, the topic will be what we have done so far.

1.2.1 Milestone1

At first, we all decided on platforms that we are going to write our code. Since most of us are new to these platforms, first week was the learning week. We analysed some examples on what we are about to develop. Afterwards our first task was to implement login and register functions on both front-end and android which are connected to back-end. We also implemented authentication and token mechanism so that users will be able to stay logged in as long as their tokens are valid.

Next, we implemented quiz feature, which is our main instrument to teach users a language. After a user enters the application, in order to determine their levels they enter quizzes. Users are able to choose any quiz that is allowed at their level. Currently, we are delivering quizzes and after a quiz, we are showing the score they get. If the exam is a placement test then we also show which level they are in.

There is a profile page also where users can see their levels in each language. Right now there is only English and the level is determined by the placement test. They also can take level up exams later. They have a bio in their profile page where they can write something about themselves.

We successfully deployed both backend and frontend. We user Amazon for that purpose. We have a fully running Postgres database which is also deployed to Amazon. We have not merged our different branches however each branch is developing steadily.

1.2.2 Milestone2

In the second part of the project, we have focused on more advanced features such as semantic search and writing assignments. In addition to these big features, we did not forget to improve our older features to have better user interaction. I will describe what are those features and how we implemented these features in the below.

To improve our older features, we first detected which fields are unnecessary such that returning those fields in the response causes extra time delay. We created specific response models for main endpoints such as Language and Member. Afterwards, we changed each endpoint to filter level and language. Now we can display contents in a categorized manner which eases user's moves in the app and website.

We also created update member profile endpoint and add/remove language endpoints. Now, users have more control over their profiles and personalization in the app is brought to fore. While we are creating more and more features, we also updated our database, made it to have richer contents. In the frontend and android part, all these features are shown with better style. In addition to these, now they can see their past solved quizzes with thier

scores when they try to find a quiz to solve.

We created writing assignments to our users. They can choose any topic they want from what we offer to them. Afterwards, when they finished writing they can choose a user who is expert in that language. After they assigned a user, on that assigned user's profile page, s/he can see pending writings which are assigned to him/her and waiting to be evaluated. After the evaluation writer and evaluator can see the writing from their profile pages.

We added search functionality in this milestone. Our search mechanism works semantically. By semantically, I mean that when you type some text in the search bar, it returns similar tagged quizzes or writings according to your choice. It returns sorted such that most similar material is shown on the top. We used a third party API for finding similarity scores.

1.2.3 Final Milestone

In the last milestone, we added annotation feature to the system. Annotations will be explained in detail in the below annotation section. However it is necessary to say that, they are compatible with W3C standards and end users are annotating without dealing with W3C standards. We have 2 different annotation types implemented currently. One is annotation on a text. This feature works as selecting some interval in the text then highlighting and commenting it. In the annotation on an image, we implemented it to enable selecting rectangular shape to annotate. After selecting the area on the image they can comment on it.

We have implemented report system. Currently, what it does is enables admins to see who is reported. When a user harass another user, they can report the scammer to admins. When a user is reported, the user is added to reported users table. Afterwards, system admins judge whether the user is a scammer or just a victim of report.

We also added comment section to the profile page so that users can comment on and rate each others profile. This way interaction between users is increased. Also we can derive credibility of a user from the comments and rates. There is also one extra help of these comments. Since there is a reporting feature, system admins, us, can judge users based on their comments.

We implemented messaging feature also. Users can message each other to communicate between them. When a message comes, the receiver gets a notification saying that you have a message. We have implemented notification feature also while implementing messaging. Messages and writing assignments have notification.

We have added several progress properties to the profile. Now users can see how many questions they have solved, what is their true/false ratio. They can also see what percentage of a course has finished. Users can also see other users progress too.

We added user search functionality to the system. Currently this feature is used when searching someone to text. They can easilty find users, they search. Oneself isn't appearing on the results.

We have also enhanced our recommendation algorithm to discover more users as a reviewer since review also a part of the learning. We haven't assigning anybody who have already assigned in that day. Also if there are a lot of reviews waiting for someone we also do not show that user.

1.3 What is next?

We have implemented every requirement that we have. From now on what we are looking is improvements rather than new features. First thing in mind is improving data quality. Currently questions and writing topics are for there as examples. We want to improve our content quality so that users from different language level can benefit from the app equally.

We also want to improve our CSS design so that frontend and android can be compatible in meaning of design and coloring. Also we need to search some psychological effects of coloring to make a effective design.

Currently our annotation server isn't apart from the system itself. We want to make it separate so that anyone can access our annotations. Also we want to implement image as an annotation in the application. This way learning can be done via visual contents.

We have a messaging system but there is no real time messaging. We want to implement a real time messaging. We want this platfrom to be a social platform. We want people connect each other and learn from each other. Also we want to add follow feature. This way each user can see their followed friends progress immediately.

2 List and Status of Deliverables

Name	Delivery Date	Delivery Status
Backend/Frontend/Android Codes	Dec 24, 2019	Delivered
User Scenarios (Demo)	Dec 24, 2019	Delivered
Project Plan	Dec 24, 2019	Delivered
Issues	Dec 24, 2019	Delivered
API Documentation	Jan 1, 2020	Delivered
Wiki Pages	Jan 1, 2020	Delivered
User & System Manuals	Jan 1, 2020	Delivered

3 Evaluation of the Status of Deliverables and Their Impact on Plan

- **User Scenarios (Demo):** As Group 10, we know how important user scenarios for our project. Because, user scenarios remind us why we are struggling to develop this project and how our project can be important for the users' lives. Therefore, we prepared two different user scenarios which were realistic for our presentation. First one was Süreyya Yıldı who checked her notifications and then controlled one of her evaluated image writing and saw the annotations on it. And then, she evaluated a text writing and added some annotations on it. She also uploaded a new image for a writing exercise. She made a comment for Murat Buldu and rated him. She saw a inappropriate message from Yavuz and reported him. Second character was Murat Buldu who checked his notifications firstly and controlled his evaluated text writing and saw annotations on it. After that, he evaluated an image writing and add some annotations on it. He sent a message to another user and completed a text writing.
- **Backend/Frontend/Android Codes:** Each week, we have delivered tasks which have required mostly coding on every platform. At first, we are failed to deliver fully working codes since platforms we are using are new to most of us, we struggled in the beginning. Later, we have successfully delivered our coding tasks. Each week, we implemented a new feature which will play key role in our app such as making comments to other users, rating other users, messaging with other users, uploading an image as a writing exercise and being able to annotate writing exercises. Currently, we completed the development process and our code base contains the necessary and sufficient code for our requirements.
- **Project Plan:** Planning is important when we talk about group projects. So, we created a project plan that outlines the major activities and milestones of our project. We updated our plan to keep track of the progression. It was a little bit hard to be in harmony with project plan because as we all know, some unexpected and blocking bugs could come up during implementation. Luckily, we could catch our deadline to implement all the requirements.
- **Issues:** As group 10, we know how important using issues actively is and the key impacts of using issues. Therefore, during the whole development process, we always tried to use issues actively to keep our team members updated about other parts of the project and struggled to improve the way we use issues.
- **API Documentation:** We implemented backend, frontend and android parts as much as we can for this milestone. We used swagger for backend api documentation which helped us a lot while communicating with

frontend and android about endpoints. Link to our api documentation:
http://cmpe451group10-env.mw3xz6vhgv.eu-central-1.elasticbeanstalk.com/swagger-ui.html

- **Wiki Pages:** From the beginning of our journey in this project, we updated our wiki pages weekly. We did this because we know that wiki pages reflect out project and our dedication to our project.Our meeting notes, user scenarios and mock ups are stored in wiki pages and after each team meeting and customer meeting, topics that are discussed and the decisions that are taken are added to the corresponding files under our wiki page.

- **User & System Manuals:** Throughout one semester, we struggled to implement a language learning platform 'YALLP' and we achieved it. Our primary and unique goal was to provide people a good platform where they can improve their language skills. Therefore, we know how important user and system manuals are for our project because they gives a chance to users to have a good experience with our platform. They can understand the way our platform works by using manuals provided by us.

4 All Design Documents

4.1 Class diagram

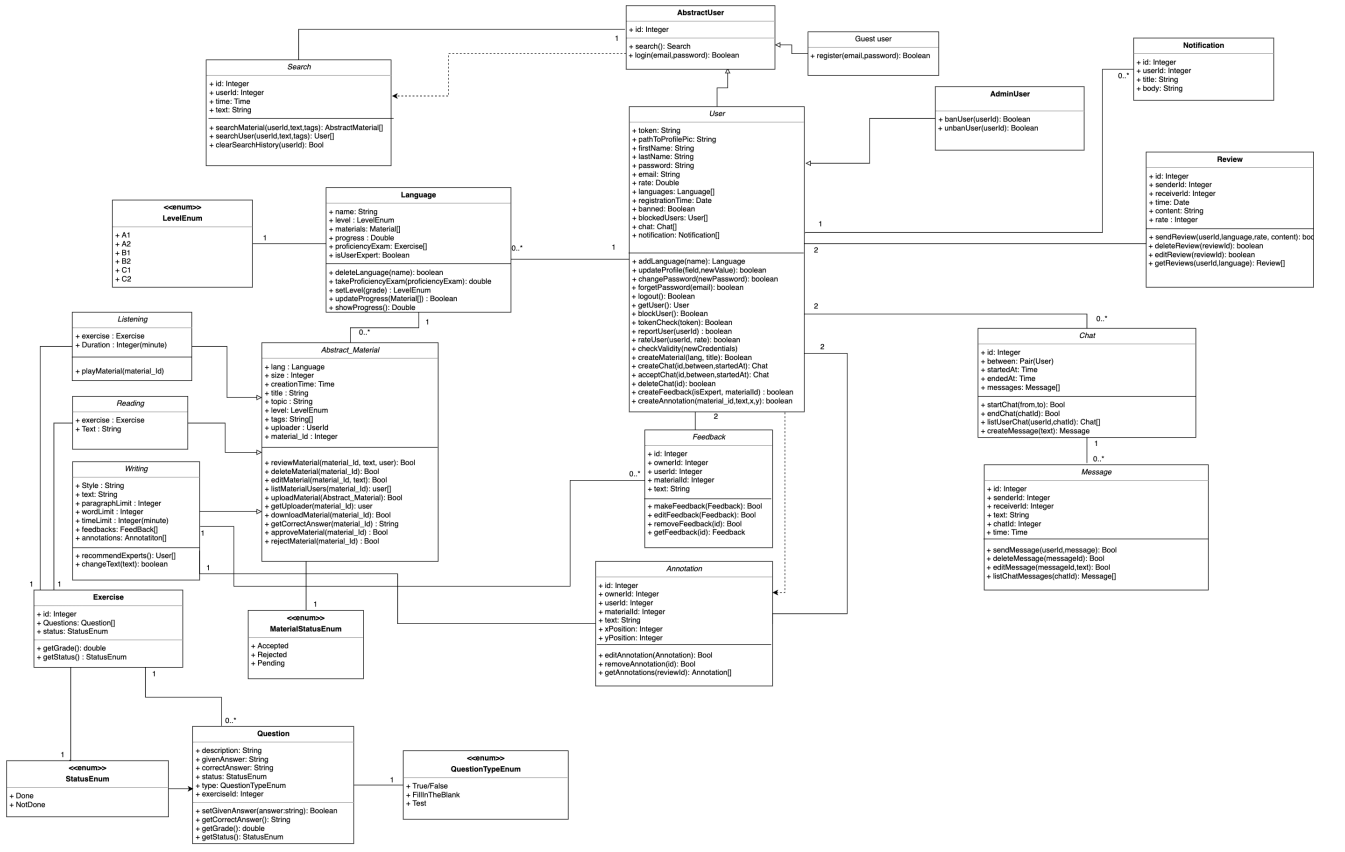


Figure 1: Class diagram of the application

4.2 Use case diagram

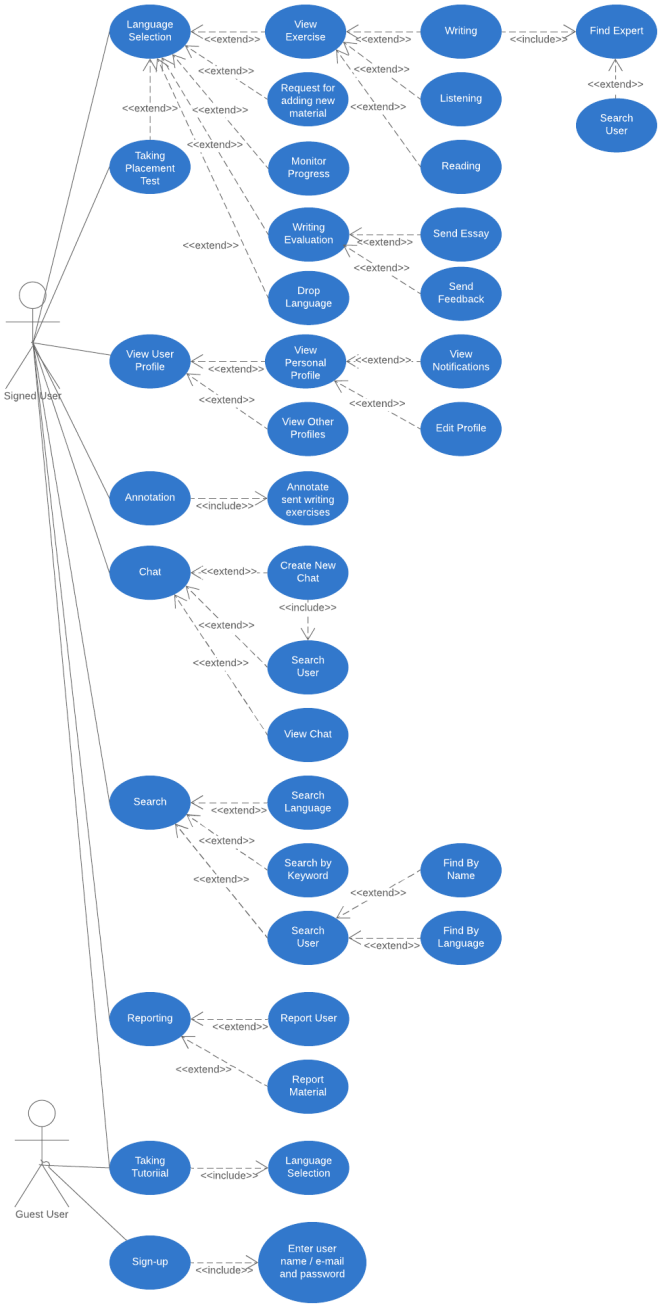


Figure 2: Use case diagram of the application

4.3 Sequence diagram

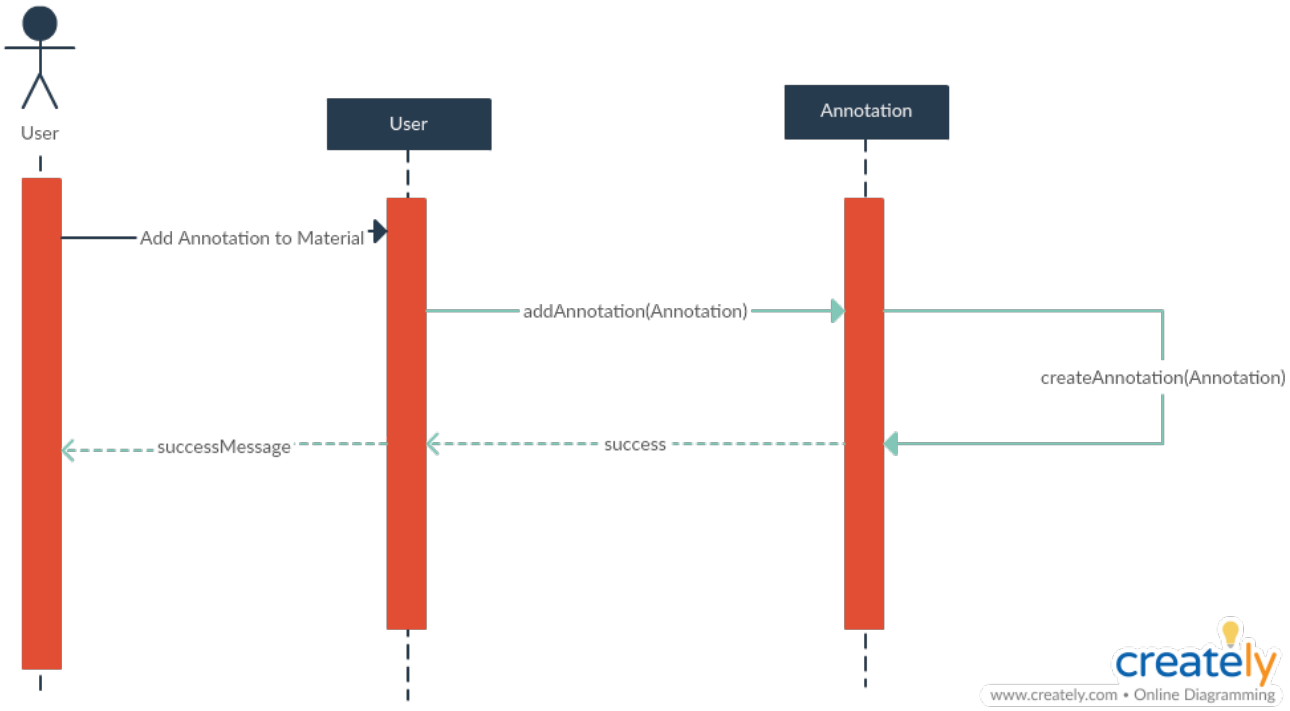


Figure 3: Sequence diagram of making an annotation

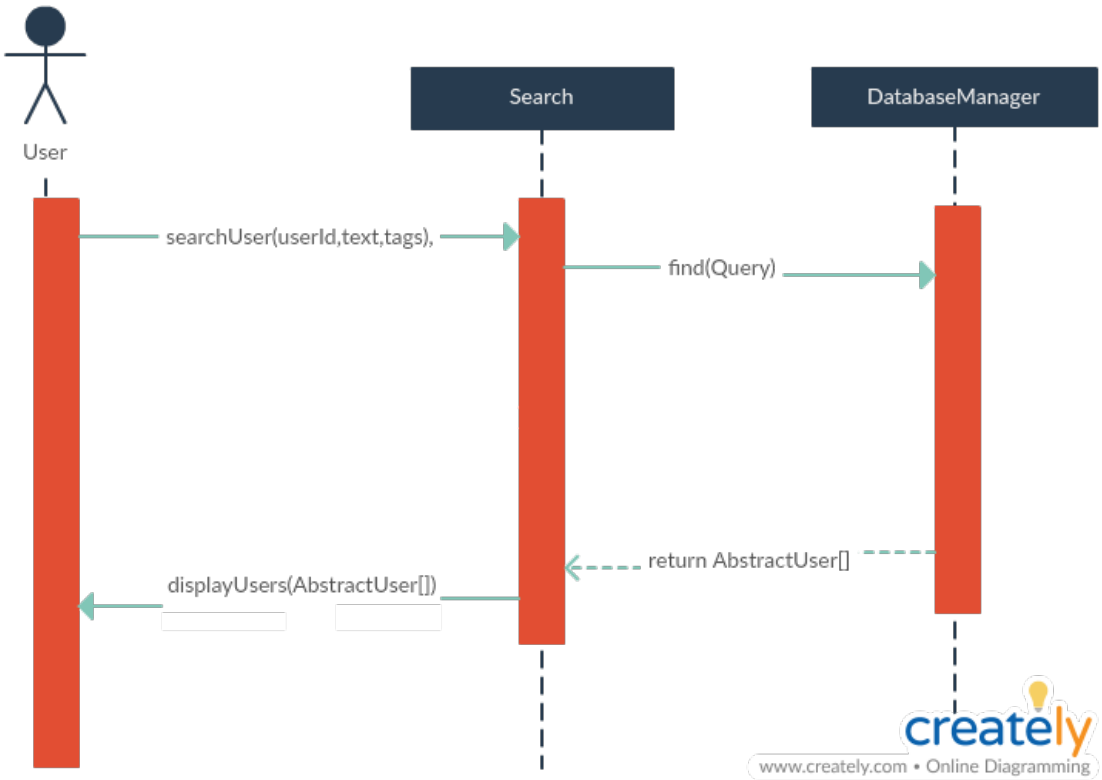


Figure 4: Sequence diagram of making a user search

4.4 Mockup

Figure 5: Mockup of image upload scene 1

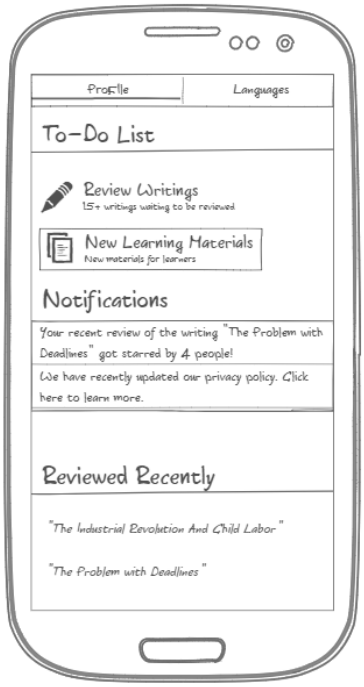


Figure 6: Mockup of image upload scene 2

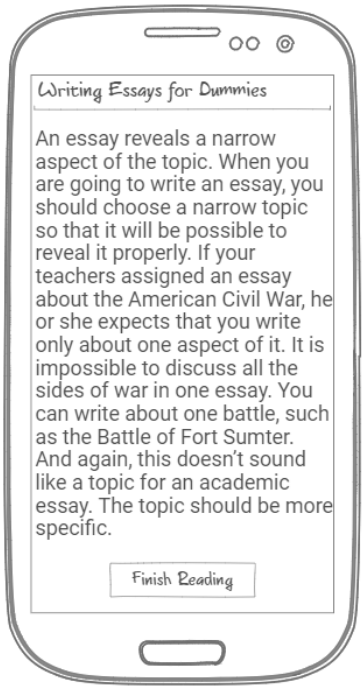


Figure 7: Mockup of image upload scene 3

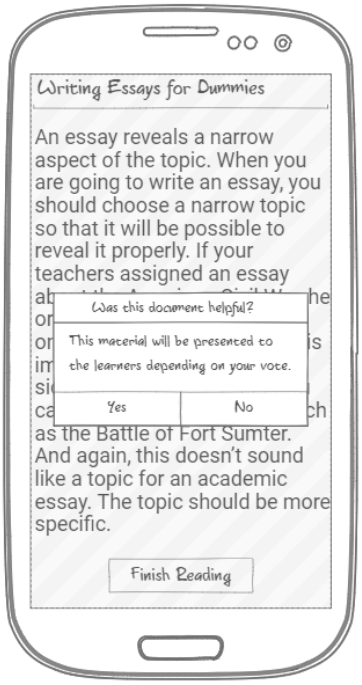


Figure 8: Mockup of image upload scene 4

5 Work-Done by Team Members

Team Member	Contributions
Yavuz Demir	Created our annotation system. Added annotation on a text and annotation on an image. Created comment on a profile feature. Added rate functionality on comments. Calculated number of comments and mean of rate then returned it so that it can be displayed on the profile page. Added user search functionality. Helped creation of upload image functionality. Fixed bugs which are assigned from team members. Opened issues related to my work and closed issues which have assigned on me.
Murat Buldu	After milestone 2, I implemented messaging functionality in frontend, so that users can send messages to each other by searching the other user by his username. I also implemented writing annotations. A user can annotate a text writing and image writing. Also, a user can see the previous annotations, create new annotation and delete an annotation if he created this annotation. I also helped the others in frontend to merge the branches into frontend-dev by resolving conflics and solving errors. In the end of the project, I deployed the frontend to AWS.
Fahri Can Şanlı	For the final milestone, I attended all the meetings (both with the customer and with the team) actively. As the first task, I changed the design and the functionality of our home page. Previously, home page didn't include any tabs but then I added tabs to home page because lots of functionalities are added. This task was a refactoring task. After this task, I implemented adding comments to other users and showing comments of the user himself/herself functionalities Then, I added the feature which enables users to rate other users. This feature is combined with comment feature. After that, I implemented the notification feature. People can see if they have any notification by looking three dots and they can see the list of notifications by clicking it. After finishing this task, I added the user progress feature to the profile page and visiting other users' pages. Users can see their progress for each language. I solved a bug related to quiz activity. I implemented the functionality which is showing average rating of the a user if he/she has any. Then, I made some refactoring on home page to speed it up and solve some bugs related to comment and rating feature. I added the flag images for each langauge and I changed the chat design to provide better experience to the users. And finally, I implemented the writing topic suggestion feature.
Sümeyra Yılmaz	After milestone 2, I added the user search functionality. It stands on top of the messaging tab. The results shown in a list which has only has username of the user. I also added the profile visit page for other users which can be opened by the clicking the elements of the the result of the search or clicking the orange written name in a conversation page. The profile visit page has all information of the profile except password, the languages and user progress, average rating, with see comments, add comment and send messages buttons. I also added the image upload for Android. The user can change the profile picture via clicking on the current one. Profile image is also seen on the visited profile page. Image upload is also used for writings. I edited all writing related pages. While sending writing, user has two options, text and image, listed as a drop down menu. It works with different calls. Evaluating writing and showing results of the old writing page is also edited according to writing type. We changed the whole UI after milestone 2 and made the main activity tabbed. I added the return to the selected tab functionality to other activities. I also did some bug fix related to these new and before created functionalities, and cleaned some unnecessary codes. I edited the search material as the endpoints changed.

Table 1: Work-done by Each Team member

6 Annotation implementation & W3C standard compliance

We implemented **annotation** structures based on **W3C** standards. There are two different ways to annotate in our system. One is text annotation on a text other is text annotation on an image. For both we have created our own **JSON-LD** structure using already defined keys. This way, we shaped the data model according to our needs.

Team Member	Contributions
Hasan Basri Balaban	After milestone-1, I attended most of the meetings with frontend team. I updated and modified our profile page. I reviewed my teammates codes. I edited List and Status of Deliverables and Evaluation of the Status of Deliverables and Their Impact on Plan parts of this report.
Selim Karaduman	Implemented endpoints for messaging functionality, reporting functionality, user writing suggestion functionality, and writing image uploading and submission functionality. Then, implemented user recommendation system for writing evaluators according to the feedback given in milestone 2 in a way that, now it works like a scheduler and prioritizes the good skilled users while making sure no new users will starve by adding some form of randomization. For all of the above endpoints: implemented the corresponding models,repositories, services and other utilities. Then, have written some unit tests for writing and messaging endpoints. Have written the documentation of the endpoints for the milestone. Opened issues on github, and used branches to develop different functionalities separately.
Melih Demir	After milestone 2, I mainly worked on the implementation of messaging and annotation functions. Mes
Hatice Melike Ecevit	Created the new tables for each feature. I maintain the database relations between the entities and decide for which is best. I wrote the user and system manuals. and Implemented the image upload feature to s3. I wrote tests to suggestion service. I implemented the member progress feature on backend Added the profile image feature to member. I fixed the security issue on AWS. Implemented the notifications feature on backend. I reviewed and opened pull requests with the backend team. I also managed the whole database, deleted the test data before the demo and inserted new data to our database.
Burak Tepedelen	Worked with the frontend team. I created page templates. Created loggin/signup system and helped with the backend connection. Assisted with homepage design. Handled error and warning system. Wrote the code structure documentation. Created writing exercise pages and all related writing review system pages including pages to check your own scores. Build the notification age and header. Created frontend portion of image upload and helped with integration of annotions to that pages. Made frontend portion of the milestone scenario and final scenario.

Table 2: Work-done by Each Team member

In this section, these models will be explained in two sub sections.

6.1 Annotation on a Textual Object

In our case, textual objects are writings written by our users. Since their writings are going to be reviewed, best way to elaborate reviews is using annotations. You can see what are fields inside an text annotation from below table.

```
1    {
2        "@context": "http://www.w3.org/ns/anno.jsonld",
3        "type": "Annotation",
4        "id": "http://cmpe451group10-env.mw3xz6vhgv.eu-central-1.
             elasticbeanstalk.com/annotation/95",
```

```
        "creator": {
            "nickname": "SureyyaYildi",
            "id": "http://cmpe451group10-env.mw3xz6vhgv.eu-central-1.
               elasticbeanstalk.com/member/110",
            "type": "Person"
        },
        "created": "2019-12-24T17:20:23Z",
        "modified": "2019-12-24T17:20:23Z",
        "bodyValue": "Is it? ",
        "target": {
            "selector": {
                "start": 180,
                "end": 204,
                "type": "TextPositionSelector"
            },
            "source": "http://cmpe451group10-env.mw3xz6vhgv.eu-central-1.
               elasticbeanstalk.com/writing/resulttext/107"
        }
    }
```

In this data model we store many information such as who is the creator, when it is created/modified, what is the id of the annotation, what is the content inside the body, what is the target and where the target came from.

6.2 Annotation on an Image Object

In our case, image objects are uploaded images by users for writing assignments. Since some of us get used to write on a paper and having difficulty writing with keyboard. They can upload photo of their writing papers and upload it to the system. After they send it to a reviewer reviewer can submit their notes as annotations. You can see what are fields inside an image annotation from below table.

```
    {
        "@context": "http://www.w3.org/ns/anno.jsonld",
        "type": "Annotation",
        "id": "http://cmpe451group10-env.mw3xz6vhgv.eu-central-1.
           elasticbeanstalk.com/annotation/image/45",
        "creator": {
            "nickname": "SureyyaYildi",
            "id": "http://cmpe451group10-env.mw3xz6vhgv.eu-central-1.
               elasticbeanstalk.com/member/110",
            "type": "Person"
        },
        "created": "2019-12-24T19:42:38Z",
        "modified": "2019-12-24T19:42:38Z",
        "bodyValue": "I can't see this",
        "target": {
            "selector": {
                "conformsTo": "http://www.w3.org/TR/media-frags/",
                "type": "FragmentSelector",
                "value": "xywh=17,22,39,23"
            },
```

```
19              "source": "https://s3.eu-central-1.amazonaws.com/group10
                    profilephotos/1577205677862-upload.jpg"
20          }
21      }
```

In this data model we store many information such as who is the creator, when it is created/modified, what is the id of the annotation, what is the content inside the body, what is the target and how the location on the image is selected.

7 Requirements

7.1 Glossary

- **Account:** A set of information related to application and different for each user
- **Admin User:** A person who is responsible for system sustainability and management in general
- **Annotation:** A feature that allows users to attach some explanatory content to items
- **Availability:** Application must be accessible from different platforms
- **Communication:** Interaction between users inside application by sending message
- **Guest User:** A person who does not have an account and have restricted access to the application
- **Language Dashboard:** A part of profile page which contains names of languages being learned by the user and progress bar
- **Learning Material:** A set of exercises related to a language
- **Profile:** A page specific for each user and has user related information
- **Recommendation:** A suggestion system for users to add new learning materials
- **Reliability:** System must have secure the database and be prepared for any type of incident
- **Scalability:** The ability of the application to handle a growing number of users
- **Search:** A tool to help the users find the relevant contents for given input words in the application
- **Settings:** A set of choices specific to each user
- **Sign In:** Entering to the application by providing correct email and password
- **Sign Up:** Creating an account to be a member of application
- **User:** A person that interacts with the application

7.2 Functional Requirements

7.2.1 User Requirements

- User Types
  - Guest User
    - Guest users shall be able to sign up anytime in the site.
    - Guest users shall be able to see a limited part of a quiz, but shall not be able to finish it unless they sign up.
    - Guest users shall not be able to start to learn a language.
  - Admin User

    - Admin users shall be able to block or remove any registered user.
    - Admin users shall be able to view messages between any users.
    - Admin users shall be able to view any student user's progress.
  - Registered User
    - Users shall be able to view the progress of the other users.
    - Users shall be able to learn as much new languages as they want.
    - Users shall be take quizzes or writing exercises for a selected language.
    - Users shall be able to take a writing exercise via writing it on a text-box or upload an image of a writing.
    - Users shall post comments to writing materials that they receive from other users.
    - Users shall be able to select a language to learn.
    - Users shall be able to learn multiple languages simultaneously.
    - Users shall be able to select another user for assigning the writing exercise to be evaluated.
    - Users shall be able to use materials which are presented for their current level and materials from previous levels.
    - Users shall be able to see their completed tasks, number of wrong-correct answers, completed percent of the course.
    - Users shall be able to search the content they are looking. This can be a user or a language specific writing topic.
    - Users shall be able to annotate images and texts.
    - Users shall be able to post comments to other users.
    - Users shall be able to learn and evaluate others' writing exercises simultaneously.
    - Users shall be able to select one of the materials he prefers to do.
    - Users shall be able to select the user who will review his writing exercise.
    - Users shall be able to see the notifications after he logs in.

- Authentication
  - Sign Up
    - The users shall be able to sign up by providing a unique username, an appropriate password and a valid email any time.
  - Sign In
    - The users shall be able to sign in anytime by using username-password pair or email-password pair.

- Profile
  - Personal Information
    - The users shall have a profile page.
    - The users shall have the option to add name and surname to his/her profile.
    - The users shall be able to load a profile picture.
  - Language Dashboard
    - The users shall be able to add new languages to learn.
    - The users shall be able to check his/her progress on each language he/she started to learn.
    - The users shall be able to drop any language course they started.
  - Settings
    - The users shall be able to change their passwords.
    - The users shall be able to delete their accounts permanently.

- Communication
  - The users shall be able to send messages to other users.

  - The users shall get notifications when they get a message from other users.

- Recommendation, Contribution and Report
  - The users shall be able to recommend new learning materials to the system, if it is approved by admins it will be added.
  - The users shall be able to report a user with a necessary reason, admins will be examined this report in detail then take an action.

7.2.2 System Requirements

- Search
  - System shall provide a semantic search mechanism through keywords that selects materials by topic, type, difficulty and scope, and show the related content and users.
- Recommendation
  - The system shall provide a recommendation mechanism to recommend users as potential evaluators of a writing exercise.
- Learning Material
  - System shall have learning materials for English, French, German and Spanish languages and if any new language is added, new materials will come along with the addition.
- Notifications
  - System shall send notifications to the user when the user gets a message, a feedback from an expert or a comment about herself/himself.

7.3 Non-Functional Requirements

- Availability
  - The system shall have a Web application that supports:
    * Chrome browser version 63 and up.
    * Firefox browser version 58 and up.
    * Opera browser version 50 and up.
  - The system shall have an Android application.
    * Min API level for the application is 17(Android 4.2)
- Security
  - The system shall protect user passwords in a database using SHA-256 encryption.
  - The system shall check user queries against SQL injection attacks.
  - User account email shall be unique.
  - User account password shall be valid.
- Scalability
  - The system should handle at least 5000 online registered users at any time.
- Reliability
  - The system should have monthly maintenance period of 3 hours.
  - The system shall recover fully from a crash in at most 24 hours.
- Annotation
  - System shall support the W3C Web Annotation Data Model.

8 API Documentation

8.1 Annotation

8.1.1 /annotation/id

Description

This endpoint is used for a user to get an annotation

Request

GET /annotation/id
id: integer

Response

Content-Type: application/json

```
{
"additionalProp1": {},
"additionalProp2": {},
"additionalProp3": {}
}
```

8.1.2 /annotation/all

Description

This endpoint is used for a user to get all annotations

Request

GET /annotation/all
No parameters

Response

Content-Type: application/json

```
[ null ]
```

8.1.3 /annotation/all/writingId

Description
This endpoint is used for a user to get all annotations of a writing text.
Request
GET /annotation/all/writingId writingId: integer
Response
Content-Type: application/json [null]

8.1.4 /annotation/create

Description

This endpoint is used for a user to create an annotation.

Request

POST /annotation/create
annotationDTO:
{
"annotationText": "string",
"annotatorId": 0,
"createdAt": { time
},
"id": 0,
"posEnd": 0,
"posStart": 0,
"updatedAt": { time
},
"writingResultId": 0
}

Response

Content-Type: application/json
{
"annotationText": "string",
"annotatorId": 0,
"createdAt": { time
},
"id": 0,
"posEnd": 0,
"posStart": 0,
"updatedAt": { time
},
"writingResultId": 0
}

8.1.5 /annotation/delete

Description

This endpoint is used for a user to delete an annotation.

Request

POST /annotation/delete
id: integer

Response

Content-Type: application/json
{
"message": "string"
}

8.1.6 /annotation/image/id

Description

This endpoint is used for a user to get an image annotation.

Request

GET /annotation/image/id
id: integer

Response

Content-Type: application/json
{
"additionalProp1": {},
"additionalProp2": {},
"additionalProp3": {}
}

8.1.7 /annotation/image/all

Description
This endpoint is used for a user to get all image annotations.
Request
GET /annotation/image/all No parameters
Response
Content-Type: application/json [null]

8.1.8 /annotation/image/create

Description

This endpoint is used for a user to create image annotations.

Request

POST /annotation/image/create
Image annotation dto: {
"annotationText": "string",
"h": 0,
"imageUrl": "string",
"w": 0,
"x": 0,
"y": 0
}

Response

Content-Type: application/json
{
"annotationText": "string",
"h": 0,
"imageUrl": "string",
"w": 0,
"x": 0,
"y": 0
}
"h":0,
"id":0,
"imageUrl": "string",
"updatedAt": {time},
"w": 0,
"x": 0,
"y": 0
}

8.1.9 /annotation/image/delete

Description
This endpoint is used for a user to delete image annotations.
Request
POST /annotation/image/delete id:integer
Response

Content-Type: application/json
{
”message”: ”string”
}

8.1.10 /annotation/update

Description

This endpoint is used for a user to update annotations.

Request

POST /annotation/update
{
"annotationText": "string",
"annotatorId": 0,
"createdAt": { time
},
"id": 0,
"posEnd": 0,
"posStart": 0,
"updatedAt": { time
},
"writingResultId": 0
}

Response

Content-Type: application/json
{
"annotationText": "string",
"annotatorId": 0,
"createdAt": { time
},
"id": 0,
"posEnd": 0,
"posStart": 0,
"updatedAt": { time
},
"writingResultId": 0
}

8.2 Authentication and Registration

8.2.1 /authenticate

Description

This endpoint is used for a client to sign-in. It is a POST request. The client will get a session token if he/she gives correct credentials. The details are given below.

Request

POST /authenticate
Content-Type: application/json
{
"username":"ali123",
"password":"——-"
}

Response

200
Content-Type: application/json
{
"token": "eyJhbGciOiJIUzUxMiJ9...."
}

8.2.2 /register

Description

This endpoint is used for a client to register. It is a POST request. The details are given below. The user needs to choose a valid username, mail and password to register. In case of success the server responses to the user with the session token.

Request

POST /register
Content-Type: application/json
{
username:"ali123",
password:"——-"
mail:"ali123@gmail.com" }

Response

200
Content-Type: application/json
{
"token": "eyJhbGciOiJIUzUxMiJ9...."
}

8.3 Member Operations

8.3.1 /member/profile

Description

This endpoint is used for an authenticated user to see his/her profile. It is a GET request. The client can get details of his/her profile, such as bio, mail, name, surname and languages.

Request

GET member/profile
Content-Type: application/json

Response

Content-Type: application/json


```
{
"id": 7,
"bio": null,
"password": "—",
"username": "ali",
"mail": "ali@123.com",
"role": "USER",
"name": "ali",
"surname": "kaya",
"memberLanguages": [
{
"id": 4,
"memberId": 7,
"language": {
"id": 1,
"languageName": "ENGLISH",
"hibernateLazyInitializer": {}
},
"languageLevel": 2
}
],
"expert": null,
"enabled": false
}
```

8.3.2 /member/addlang

Description

This endpoint is used for a user to add a language to his/her language list

Request

POST /member/addlang
Content-Type: application/json
["string"]

Response

Content-Type: application/json

```
[
{
"id": 0,
"language": {
"id": 0,
"languageName": "string"
},
"languageLevel": 0,
"levelName": "BEGINNER",
"memberId": 0
}
]
```

8.3.3 /member/removelang

Description

This endpoint is used for a user to remove a language to his/her language list

Request

POST /member/removelang
Content-Type: application/json
["string"]

Response

Content-Type: application/json


```
[
{
"id": 0,
"language": {
"id": 0,
"languageName": "string"
},
"languageLevel": 0,
"levelName": "BEGINNER",
"memberId": 0
}
]
```

8.3.4 /member/memberId

Description

This endpoint is used to get all information of a member from its ID

Request

POST /member/addlang
Content-Type: application/json
memberId: integer

Response

Content-Type: application/json


```
{
"bio": "string",
"enabled": true,
"profileImageUrl": "string",
"expert": true,
"id": 0,
"mail": "string",
"memberLanguages": [
{
"id": 0,
"language": {
"id": 0,
"languageName": "string"
},
"languageLevel": 0,
"levelName": "BEGINNER",
"memberId": 0
}
],
"name": "string",
"nativeLanguage": "string",
"password": "string",
"role": "string",
"surname": "string",
"username": "string"
}
```

8.3.5 /member/update

Description

This endpoint is used to update the information of a member. It is a PUT request

Request

PUT /member/update
Content-Type: application/json

{
"bio": "string",
"id": 0,
"mail": "string",
"name": "string",
"nativeLanguage": "string",
"password": "string",
"surname": "string",
"username": "string"
}

Response

Content-Type: application/json

{
"token": "string"
}

8.3.6 /member/profileImage

Description

This endpoint is used for a user to upload a profile picture.

Request

POST /member/profileImageL
file

Response

Content-Type: application/json
{
"url": "string"
}

8.4 Language Operations

8.4.1 /lang

Description
This endpoint returns all of the languages as a response. It is a GET request.
Request
GET /lang Content-Type: application/json No parameters.
Response

Content-Type: application/json
[
{
”id”: 0,
”languageName”: ”string”
}
]

8.4.2 /lang/unsubs

Description
This endpoint is used to get all languages which are not subscribed already.
Request
GET /lang/unsubs Content-Type: application/json No parameters.
Response

Content-Type: application/json
[
{
”id”: 0,
”languageName”: ”string”
}
]

8.5 Quiz Operations

8.5.1 /quiz/quizId

Description

This endpoint is used for an authenticated user to get a quiz from the server. It is a GET request. The response will contain questions, the choices and the answers. The details are given below

Request

GET /quiz/quizId
Content-Type: application/json

Response

Content-Type: application/json


```
{
"id": 5,
"level": 1,
"quizType": "phrasal",
"questions": [
{
"id": 49,
"questionText": "I ..... could later today to collect the books",
"firstChoice": "check in",
"secondChoice": "broke down",
"thirdChoice": "call back",
"correctChoiceId": 3,
"quizId": 5
},
{
"id": 50,
"questionText": "Mari ..... the wedding at the very last minute!",
"firstChoice": "broke down",
"secondChoice": "called off",
"thirdChoice": "check in",
"correctChoiceId": 2,
"quizId": 5
},
{
"id": 51,
"questionText": "We have to ..... the search.",
"firstChoice": "call off",
"secondChoice": "check in",
"thirdChoice": "broke down",
"correctChoiceId": 1,
"quizId": 5
},
}
```

8.5.2 /quiz/level

Description

This endpoint is used for an authenticated user to get a placement test. It is a GET request. The details are given below.

Request

GET /quiz/level
Content-Type: application/json

Response

Content-Type: application/json


```
{
"id": 66,
"level": 1,
"quizType": "level",
"questions": [
{
"id": 660,
"questionText": "Choose the correct suffix for terr.....",
"firstChoice": "ify",
"secondChoice": "ize",
"thirdChoice": "ly",
"correctChoiceId": 1,
"quizId": 66
},
{
"id": 661,
"questionText": "Choose the correct suffix for ver.....",
"firstChoice": "ly",
"secondChoice": "ify",
"thirdChoice": "ize",
"correctChoiceId": 2,
"quizId": 66
},
}
```

8.5.3 /quiz/quizId/submit

Description

This endpoint is used for an authenticated user to submit his/her quiz. It is a POST request.
The user sends the details of the question in json format. The user includes the answers to each question
In response the server returns the quiz in json format with additional fields. Those fields are: Score of the user, which is calculated using the number correct answers of the user. Level, which is returned only if the quiz is a placement quiz. And finally, for each question the server tells whether or not the question is correct.

Request

POST quiz/quizId/submit
Content-Type: application/json


```
{
"quizId": 66,
"answers":[
{"questionId": 660, "choiceId": 2},
{"questionId": 661, "choiceId": 1},
{"questionId": 662, "choiceId": 3},
{"questionId": 663, "choiceId": 3},
{"questionId": 664, "choiceId": 3},
{"questionId": 665, "choiceId": 3},
{"questionId": 666, "choiceId": 3},
{"questionId": 667, "choiceId": 3},
{"questionId": 668, "choiceId": 3},
]
}
```


Response

Content-Type: application/json


```
{
"quizId":66,
"answers":[
{"questionId":660,"choiceId":2,"correctId":1,"true":false},
{"questionId":661,"choiceId":1,"correctId":2,"true":false},
{"questionId":662,"choiceId":3,"correctId":3,"true":true},
{"questionId":663,"choiceId":3,"correctId":2,"true":false},
{"questionId":664,"choiceId":3,"correctId":1,"true":false},
{"questionId":665,"choiceId":3,"correctId":2,"true":false},
{"questionId":666,"choiceId":3,"correctId":3,"true":true},
{"questionId":667,"choiceId":3,"correctId":2,"true":false},
{"questionId":668,"choiceId":3,"correctId":3,"true":true}
],
"level":3,
"score":3
}
```

8.5.4 /quiz

Description

This endpoint is used to get all quizzed. It returns all the quizzes with additional information: wheher or not they are solved, what is the score

Request

GET /quiz
Content-Type: application/json
No parameters.

Response

Content-Type: application/json


```
[
{
"quiz": {
"id": 0,
"level": 0,
"levelName": "string",
"questions": [
{
"correctChoiceId": 0,
"firstChoice": "string",
"id": 0,
"questionText": "string",
"quizId": 0,
"secondChoice": "string",
"thirdChoice": "string"
}
],
"quizType": "string"
},
"score": 0,
"solved": true
}
]
```

8.5.5 /quiz/language/languageId

Description

This endpoint is used to get all quizzes in a given language.

Request

GET /quiz
Content-Type: application/json
languageId: integer.

Response

Content-Type: application/json


```
[
{
"quiz": {
"id": 0,
"level": 0,
"levelName": "string",
"questions": [
{
"correctChoiceId": 0,
"firstChoice": "string",
"id": 0,
"questionText": "string",
"quizId": 0,
"secondChoice": "string",
"thirdChoice": "string"
}
],
"quizType": "string"
},
"score": 0,
"solved": true
}
]
```

8.5.6 /quiz/level/level/language/languageId

Description

This endpoint is used to get all quizzes in a given language.

Request

GET /quiz/level/level/language/languageId
Content-Type: application/json
languageId: integer.
level: integer.

Response

Content-Type: application/json


```
{
"quiz": {
"id": 0,
"level": 0,
"levelName": "string",
"questions": [
{
"correctChoiceId": 0,
"firstChoice": "string",
"id": 0,
"questionText": "string",
"quizId": 0,
"secondChoice": "string",
"thirdChoice": "string"
}
],
"quizType": "string"
},
"score": 0,
"solved": true
}
```

8.5.7 /quiz/level/levelId

Description

This endpoint is used to get all quizzes in a given level.

Request

GET /quiz/level/level/language/languageId
Content-Type: application/json
levelId: integer.

Response

Content-Type: application/json


```
[
{
"quiz": {
"id": 0,
"level": 0,
"levelName": "string",
"questions": [
{
"correctChoiceId": 0,
"firstChoice": "string",
"id": 0,
"questionText": "string",
"quizId": 0,
"secondChoice": "string",
"thirdChoice": "string"
}
],
"quizType": "string"
},
"score": 0,
"solved": true
}
]
```

8.5.8 /quiz/levelorlower/level/language/languageId

Description

This endpoint is used to get all quizzes in a given language and level

Request

GET /quiz/levelorlower/level/language/languageId
Content-Type: application/json
languageId: integer.
level: integer.

Response

Content-Type: application/json


```
[ {
"quiz": {
"id": 0,
"level": 0,
"levelName": "string",
"questions": [
{
"correctChoiceId": 0,
"firstChoice": "string",
"id": 0,
"questionText": "string",
"quizId": 0,
"secondChoice": "string",
"thirdChoice": "string"
}
],
"quizType": "string"
},
"score": 0,
"solved": true
}
]
```

8.6 Search Operations

8.6.1 /search/quiz/languageId/term

Description

This endpoint is used for searching the quizzes that are related to a search term.

Request

POST /search/quiz/languageId/term
languageId: integer
term: string

Response

Content-Type: application/json


```
[
{
"quiz": {
"id": 0,
"level": 0,
"levelName": "string",
"questions": [
{
"correctChoiceId": 0,
"firstChoice": "string",
"id": 0,
"questionText": "string",
"quizId": 0,
"secondChoice": "string",
"thirdChoice": "string"
}
],
"quizType": "string"
},
"score": 0,
"solved": true
}
]
```

8.6.2 /search/writing/languageId/term

Description

This endpoint is used for searching the writings that are related to a search term.

Request

GET /search/writing/languageId/term
languageId: integer
term: string

Response

Content-Type: application/json


```
[
{
"solved": true,
"writingDTO": {
"id": 0,
"languageId": 0,
"taskText": "string",
"writingName": "string"
},
"writingResultDTO": {
"answerText": "string",
"assignedMemberId": 0,
"assignedMemberName": "string",
"id": 0,
"memberId": 0,
"memberName": "string",
"score": 0,
"scored": true,
"writingId": 0,
"writingName": "string"
}
}
]
```

8.7 Writing Operations

8.7.1 /writing/writingId

Description

This endpoint is used to get a Writing by ID. Returns writing plus the recommended usernames.

Request

POST /writing/writingId
writingId: integer

Response

Content-Type: application/json


```
{
"usernames": [
"string"
],
"writingDTO": {
"id": 0,
"languageId": 0,
"taskText": "string",
"writingName": "string"
}
}
```

8.7.2 /writing/writingId/submit

Description

This endpoint is used send the answer of a writing. It also needs to send the selected user's username for evaluation.

Request

POST /writing/writingId/submit
writingId: integer

{
"answerText": "string",
"evaluatorUsername": "string",
"writingId": 0
}

Response

Content-Type: application/json

{
"answerText": "string",
"assignedMemberId": 0,
"assignedMemberName": "string",
"id": 0,
"memberId": 0,
"memberName": "string",
"score": 0,
"scored": true,
"writingId": 0,
"writingName": "string"
}

8.7.3 /writing/completedAssignments

Description

A user can see his/her assigned evaluations that had been completed with this endpoint.

Request

GET /writing/completedAssignments
No parameters

Response

Content-Type: application/json


```
[
{
"answerText": "string",
"assignedMemberId": 0,
"assignedMemberName": "string",
"id": 0,
"memberId": 0,
"memberName": "string",
"score": 0,
"scored": true,
"writingId": 0,
"writingName": "string"
}
]
```

8.7.4 /writing/nonCompletedAssignments

Description

A user can see his/her assigned evaluations that had not been completed with this endpoint.

Request

GET /writing/nonCompletedAssignments
No parameters

Response

Content-Type: application/json


```
[
{
"answerText": "string",
"assignedMemberId": 0,
"assignedMemberName": "string",
"id": 0,
"memberId": 0,
"memberName": "string",
"score": 0,
"scored": true,
"writingId": 0,
"writingName": "string"
}
]
```

8.7.5 /writing/getJson/languageId

Description

This endpoint returns all of the writings in a given language.

Request

GET /writing/getJson/languageId languageId: integer

Response

Content-Type: application/json

[
{
"solved": true,
"writingDTO": {
"id": 0,
"languageId": 0,
"taskText": "string",
"writingName": "string"
},
"writingResultDTO": {
"answerText": "string",
"assignedMemberId": 0,
"assignedMemberName": "string",
"id": 0,
"memberId": 0,
"memberName": "string",
"score": 0,
"scored": true,
"writingId": 0,
"writingName": "string"
}
}
]

8.7.6 /writing/read/writingId

Description

This endpoint returns the writing by its ID. Unlike former writing/{writingId} method this method does not return the recommended users as well.

Request

GET /writing/read/writingId
writingId: integer

Response

Content-Type: application/json


```
{
"solved": true,
"writingDTO": {
"id": 0,
"languageId": 0,
"taskText": "string",
"writingName": "string"
},
"writingResultDTO": {
"answerText": "string",
"assignedMemberId": 0,
"assignedMemberName": "string",
"id": 0,
"memberId": 0,
"memberName": "string",
"score": 0,
"scored": true,
"writingId": 0,
"writingName": "string"
}
}
```

8.7.7 /writing/scores

Description

This endpoint returns all of the scores of the user. It gives the information on score or whether or not the writing is scored

Request

GET /writing/scores
No parameters

Response

Content-Type: application/json


```
[
{
"answerText": "string",
"assignedMemberId": 0,
"assignedMemberName": "string",
"id": 0,
"memberId": 0,
"memberName": "string",
"score": 0,
"scored": true,
"writingId": 0,
"writingName": "string"
}
]
```

8.7.8 /writing/score/writingResultId

Description

This endpoint is used for a user to evalueate a given writing.

Request

POST /writing/score/writingResultId
writingResultId: integer

Response

Content-Type: application/json

{
"answerText": "string",
"assignedMemberId": 0,
"assignedMemberName": "string",
"id": 0,
"memberId": 0,
"memberName": "string",
"score": 0,
"scored": true,
"writingId": 0,
"writingName": "string"
}

8.7.9 /writing/uploadWritingImage

Description

This endpoint is used for a user to upload writing as an image

Request

POST /writing/uploadWritingImage
file

Response

Content-Type: application/json

```
{
"url": "string"
}
```

8.7.10 /writing/writingId/submitWithImageURL

Description

This endpoint is used for a user to upload a profile picture.

Request

POST /writing/writingId/submitWithImageURL

```
{
"answerText": "string",
"assignedMemberId": 0,
"assignedMemberName": "string",
"id": 0,
"image": true,
"imageUrl": "string",
"memberId": 0,
"memberName": "string",
"score": 0,
"scored": true,
"writingId": 0,
"writingName": "string"
}
```

Response

Content-Type: application/json

```
{
"url": "string"
}
```

8.8 Comment

8.8.1 /comment

Description
This endpoint is used for users to get comments on theirselves
Request
GET /comment No parameters
Response

Content-Type: application/json
[
{
”comment”: ”string”,
”commentatorId”: 0,
”commentatorName”: ”string”,
”createdAt”: {time
},
”id”: 0,
”memberId”: 0,
”memberName”: ”string”,
”rating”: 0,
”updatedAt”: {time
}
}
]

8.8.2 /comment/memberId

Description
This endpoint is used for users to get comments on a specific user
Request
GET /comment/memberId memberId: integer
Response

Content-Type: application/json
[
{
”comment”: ”string”,
”commentatorId”: 0,
”commentatorName”: ”string”,
”createdAt”: {time
},
”id”: 0,
”memberId”: 0,
”memberName”: ”string”,
”rating”: 0,
”updatedAt”: {time
}
}
]

8.8.3 /comment/delete

Description
This endpoint is used for users to delete a comment
Request
POST /comment/delete id:interger
Response
Content-Type: application/json "string"

8.8.4 /comment/make

Description
This endpoint is used for users make comments
Request

POST /comment/make {
"comment": "string",
"memberId": 0,
"rating": 0
}

Response

Content-Type: application/json
{
"comment": "string",
"commentatorId": 0,
"commentatorName": "string",
"createdAt": {time
},
"id": 0,
"memberId": 0,
"memberName": "string",
"rating": 0,
"updatedAt": {time
}
}

8.8.5 /comment/rating

Description
This endpoint is used for users to get their raing
Request
GET /comment/rating No parameters
Response

Content-Type: application/json
{
”numberOfRatings”: 0,
”rating”: 0
}

8.8.6 /comment/rating/memberId

Description
This endpoint is used for users to get their raing
Request
GET /comment/rating/memberId memberId:interger
Response

Content-Type: application/json
{
”numberOfRatings”: 0,
”rating”: 0
}

8.8.7 /comment/update

Description

This endpoint is used for users to get their raing

Request

POST /comment/update {
"comment": "string",
"commentatorId": 0,
"commentatorName": "string",
"createdAt": {time
},
"id": 0,
"memberId": 0,
"memberName": "string",
"rating": 0,
"updatedAt": {time
}
}

Response

Content-Type: application/json
{
"comment": "string",
"commentatorId": 0,
"commentatorName": "string",
"createdAt": {time
},
"id": 0,
"memberId": 0,
"memberName": "string",
"rating": 0,
"updatedAt": {time
}
}

8.9 Image

8.9.1 /image/deleteFile

Description
This endpoint is used for users to delete a file
Request
DELETE /image/deleteFile url
Response
Content-Type: application/json string

8.9.2 /image/upload

Description
This endpoint is used for users to get a file url
Request
POST /image/upload file
Response
Content-Type: application/json string

8.10 Message

8.10.1 /message/conversationId

Description
This endpoint is used for users to get conversations
Request
GET /message/conversationId conversationId:integer
Response

Content-Type: application/json
{
”id”: 0,
”member1_id”: 0,
”member2_id”: 0,
”messages”: [
{
time
},
”receiverUsername”: ”string”,
”senderUsername”: ”string”
}
],
”otherUsername”: ”string”,
”read”: true
}

8.10.2 /message/conversations

Description
This endpoint is used for users to get all conversations of theirselves
Request
GET /message/conversations No parameters
Response

Content-Type: application/json
[
{
”id”: 0,
”member1_id”: 0,
”member2_id”: 0,
”messages”: [
{
time
},
”receiverUsername”: ”string”,
”senderUsername”: ”string”
}
],
”otherUsername”: ”string”,
”read”: true
}
]

8.10.3 /message/send

Description

This endpoint is used for users to get send a message to another user.

Request

POST /message/send {
"message": "string",
"targetUsername": "string"
}

Response

Content-Type: application/json
{
"id": 0,
"member1_id": 0,
"member2_id": 0,
"messages": [
{
time
},
"receiverUsername": "string",
"senderUsername": "string"
}
],
"otherUsername": "string",
"read": true
}

8.11 Notifications

8.11.1 /notification/make/read

Description

This endpoint is used for users to make their notifications status read.

Request

POST /notification/make/read [
{
"id": 0,
"memberId": 0,
"notificationType": "WRITING_RESULT",
"read": true,
"text": "string"
}
]

Response

Content-Type: application/json
[
{
"id": 0,
"memberId": 0,
"notificationType": "WRITING_RESULT",
"read": true,
"text": "string"
}
]

8.11.2 /notification/make/not/read

Description
This endpoint is used for users to get non-read notifications
Request
GET /notification/make/not/read No parameters
Response

Content-Type: application/json
[
{
”id”: 0,
”memberId”: 0,
”notificationType”: ”WRITING_RESULT”,
”read”: true,
”text”: ”string”
}
]

8.11.3 /notification/read

Description
This endpoint is used for users to get already read notifications
Request
GET /notification/make/read No parameters
Response

Content-Type: application/json
[
{
”id”: 0,
”memberId”: 0,
”notificationType”: ”WRITING_RESULT”,
”read”: true,
”text”: ”string”
}
]

8.12 Reporting

8.12.1 /report/causes

Description
This endpoint is used for users to get possible causes of a report
Request
GET /report/causes No parameters
Response

Content-Type: application/json
[
{
”cause”: ”string”,
”id”: 0
}
]

8.12.2 /report/send

Description
This endpoint is used for users to get possible causes of a report
Request

POST /report/send {
”cause”: ”string”,
”optionalExplanation”: ”string”,
”reported_username”: ”string”
}

Response

Content-Type: application/json
{
”cause”: ”string”,
”optionalExplanation”: ”string”,
”reported_username”: ”string”
}

9 Project Plan

	Backend	Frontend	Android
Login functions	Yavuz Demir	Burak Tepedelen	Sümeyra Yılmaz
Sign up functions	Hatice Melike Ecevit	Burak Tepedelen	Fahri Can Sanli
Basic Profile Functions	Hatice Melike Ecevit	Hasan Basri Balaban	Melih Demir
Quiz Functions	Selim Karaduman	Murat Buldu	Fahri Can Sanli, Sümeyra Yılmaz
Home Page	Yavuz Demir	Murat Buldu	Fahri Can Sanli, Sümeyra Yılmaz, Melih Demir
Milestone1	-	-	-
Language Dashboard (UI)	-	Murat Buldu	Sümeyra Yilmaz
Search Functionality for Quizzes and Writings	Yavuz Demir	Murat Buldu	Sümeyra Yilmaz
Adding and removing languages	Hatice Melike Ecevit	Murat Buldu	Fahri Can Sanli
Android Profile Page UI Changes	-	-	Melih Demir, Sümeyra Yılmaz
Update profile	Hatice Melike Ecevit	Hasan Basri Balaban	Melih Demir
Writing Exercises	Selim Karaduman	Burak Tepedelen	Fahri Can Sanli, Melih Demir
Milestone2	-	-	-
Android Home Page UI Changes	-	-	Fahri Can Sanli
Image Upload for Profile	Hatice Melike Ecevit	Hasan Basri Balaban	Sümeyra Yılmaz
Image Writing Exercises	Hatice Melike Ecevit	Burak Tepedelen	Sümeyra Yilmaz
Language progress in profile	Hatice Melike Ecevit	Hasan Basri Balaban	Fahri Can Sanli
Search Functionality for Users	Yavuz Demir	Murat Buldu	Sümeyra Yılmaz
Messaging Functionality	Selim Karaduman	Murat Buldu	Melih Demir
Android Messaging Chat UI Changes	-	-	Fahri Can Sanli
User Profile Visit	Yavuz Demir	Hasan Basri Balaban	Sümeyra Yılmaz
Notification system	Hatice Melike Ecevit	Burak Tepedelen	Fahri Can Sanli
Comment	Yavuz Demir	Hasan Basri Balaban	Fahri Can Sanli
User Rating	Yavuz Demir	Hasan Basri Balaban	Fahri Can Sanli
Image Annotation on Writings	Yavuz Demir	Murat Buldu	Melih Demir
Text Annotation on Writings	Yavuz Demir	Murat Buldu	Melih Demir
User Writing Topic Recommendation	Selim Karaduman	Burak Tepedelen	Fahri Can Sanli
User Report	Selim Karaduman	Hasan Basri Balaban	Fahri Can Sanli
Final	-	-	-

10 User Scenarios

10.1 Android Scenario

Süreyya Yıldı is a Turkish college student with the aspiring goal of going to a Work and Travel program abroad. She uses YALLP very actively in order to achieve her goal of learning proficient English. Today, she starts using the app by checking her notifications, and sees that she has both a writing to evaluate and one of her writings evaluated by someone. She decides to first read what her evaluator wrote about her writing, so she opens the "See My Writings" tab and sees that her image writing was given a 8 out of 10. She also observes that a couple of annotations were left on her image, mainly addressing the issue that her handwriting is not very readable. Then she moves on to attend to the writing she has to evaluate, so she switches to the "Assigned Writings" tab from her profile. A user named Murat Buldu has sent her a text writing to evaluate. She reads the writing thoroughly, leaves a few annotations where she seems fit, and scores the writing a 7 out of 10. She really likes this user's writing style, so she decides to leave a good comment on his profile page. She opens the user search tab and searches for "MuratBuldu". Then she clicks on his profile, and sees that this user has no comments on their page yet. She leaves a positive comment, and also gives a rating of 4.5. She submits her comment, and immediately sees that the user's rating has been calculated as 4.5. She proceeds to check her messages, and sees that some other user is actively being aggresive towards her. She navigates to this user's page and uses the "Report this user" button, to let admins know that this user's presence is harmful to the app.

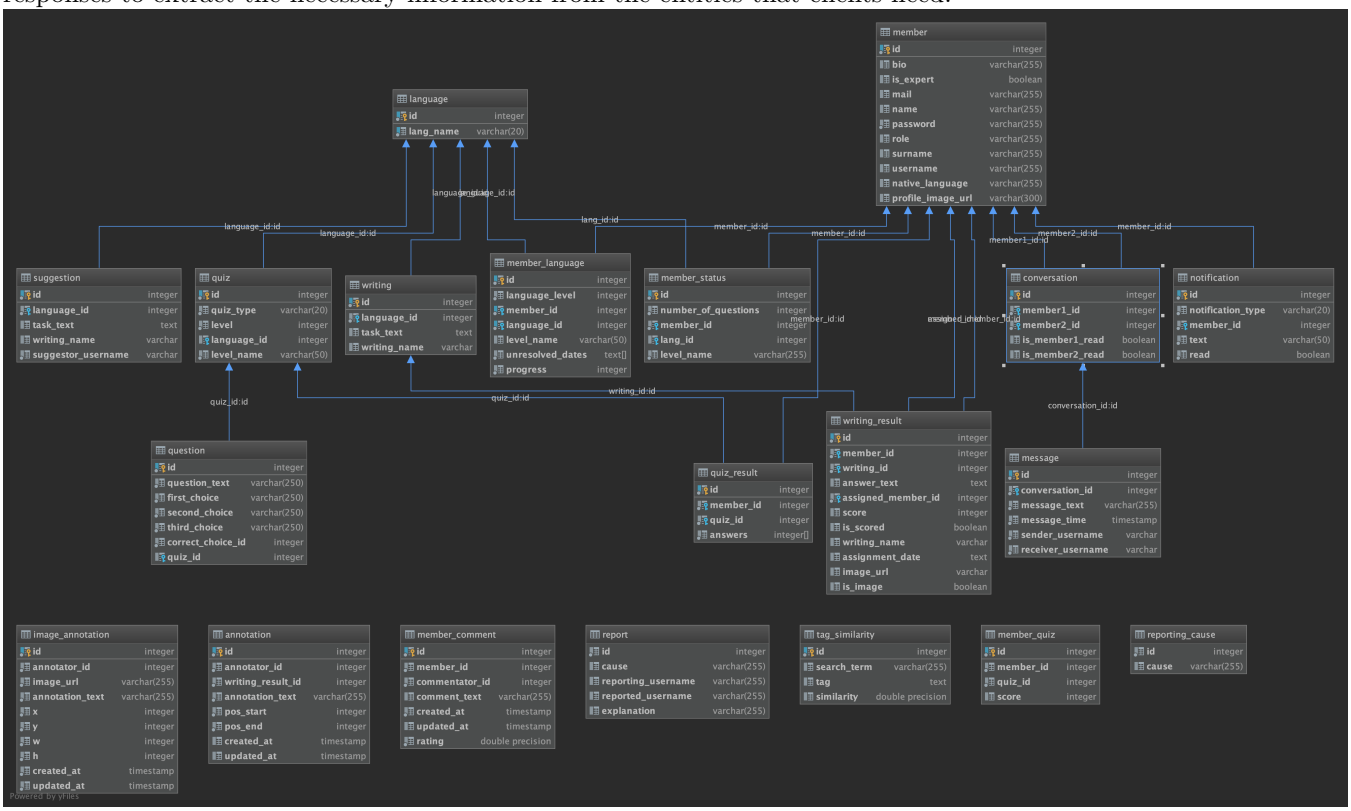
10.2 Frontend Scenario

Murat Buldu is a writer, wishing to use YALLP as a means of educating people with his English grammar and vocabulary knowledge, as well as get second opinions on his short writings. He is on his computer so he decides to use the web client of YALLP. He sees that he has three notifications: a writing of his was evaluated, he has a new writing to evaluate, and someone left a comment on his profile page. He first checks the writing he sent to someone, and sees that his writing was annotated in a few places and given a score of 7. He then proceeds to evaluate the writing he was given, it is an image writing that has really bad quality. He leaves a quick annotation pointing to this issue, gives a rating of 3 and moves on. He goes back to his profile page to check the comment that was left on his page. He realizes that the user who left the comment, Süreyya Yıldı, is the same user that evaluated his text writing. He wants to contact this person for leaving such a positive comment with a good rating, so he navigates to the Messages page and searches for this user. He sends a greeting message to Süreyya.

11 Code Structure and Group Process

11.1 Backend

Backend code is composed of Java, Spring Boot framework also some other internal frameworks of Spring is used such as Spring Security and Spring Test.
It is a Maven project and project structure is as follows,
We have a config module to handle the security and other base configurations of the project also, model module to represent the entities of the project (this part will be explained in detail with the database structure), repository module to use JpaRepository to access the entities without using sql, service module to implement the logic of the platform, service module uses repository module and passes the resulting entities to controller layer, controller module takes the request and returns the response to the clients. Also we used data transfer objects (dtos) for our responses to extract the necessary information from the entities that clients need.



Our database structure as shown is very simple. We have used Postgres Relational Database on AWS and Elastic Beanstalk to deploy our backend application.

Our branch structure is as follows,
Base branch is backend-dev and for each feature we have implemented we created new branches and developed our code then we opened a pull request for each member to approve the new feature/bug/fix and merged it to backend-dev.

11.2 Frontend

Frontend code is composed of Javascript code that uses React libraries, HTML files that mainly uses CSS style and few other configuration files.
Entry to the code is through index.js file in source which initiates redux components and calls app.js in App.
Code uses Redux to handle states which ensures information transfer and changes happen in a safe and predictable manner. Necessary functions called in each page file.
Each folder in the code includes an index for calling correct javascript files inside. HTML files are with the code requires them in this folders. These folders are:

- App file that manages page routing.
- HomePage,OpeningPage,ProfilePage,QuizPage,WritingPage,WritingReviewPage contain respective pages for website. images contain website images.
- _actions contain dispatch calls and alert returns.
- _components contain routing for logged in users.
- _constants contain constants for success and failure returns.
- _reducer contain Redux reducers for state handling.
- _servies contain backend request and other service calls.
- _helpers contain other code that doesn't fit other files.

11.3 Android

Android code is composed of mainly java classes. To make our code more readable and structured, as Android Team, we split our classes into different folders. Currently, we have activity folder where all activities are available, fragments folder in which we wrote reusable code pieces, utilities folder in which we have API folder and retro client folder. API and Retro Client folder is used to send requests to back-end and receive responses from it. We also have models folder in which we indicated the way we send our request and we expect the response to parse it easily. Other than these folders, we also have resources folder where we keep all the things related to design of our pages. In that folder, usually XML files and predefined constant string and style elements are available. Our application starts with the main activity when launched and it uses intents to navigate between activities to show different layouts to the user when needed.
As android team, we use pull requests a lot. The person developing a new feature have to open a pull request before merging his/her development branch to our dev branch and add other teammates as reviewer. The condition for one team member to be able to merge his/her branch, he/she has to get two approval from his/her team members. And if there is any comments on the changes made, the author should make change in the code accordingly and update the pull request.

Branch Name	Author	Brief Explanation
android-add-tab-to-profile	Fahri Can Şanlı	With the every new feature added to the our application, as android team we realized that we had to make some design changes. And in this branch, home page of our application is changed. We added tabs which are 'Profile', 'Languages' and 'Messages'.Lots of refactoring are made. Activities are converted to fragments.
android-comment-list	Fahri Can Şanlı	This is the first branch related to the comment functionality. Therefore, necessary model, API and retro client classes are created.And, comments that are made to a user are shown in the profile page of user himself/herself.
android-add-comment-backup	Fahri Can Şanlı	In this branch, making comments to other users feature is added. Related API class is changed.
android-add-rate	Fahri Can Şanlı	In this branch, comment model is modified. Rating other users feature is added to the our android application. Also, the way we display the comments that are made to a user is also changed. We added 5 stars that are filled according to the rate to demonstrate the received rating in that comment.
android-notification	Fahri Can Şanlı	In this branch, notification feature is added. A new clickable text is added to three dots view and whether a user has new notifications or not is demonstrated with green/red icon. Upon clicking notifications,unread and read notifications are displayed separately.
android-user-progress	Fahri Can Şanlı	In this branch, progress of users for each language is displayed. A progress bar is added to below of ever language and the completed percentage is both shown in progress bar by filling it and with a text above of progress bar.
android-avg-rating-of-a-user	Fahri Can Şanlı	In this branch, the average rating of the ratings received by other users is shown in the profile pages. Also user report feature is added.Upon visiting profile page of another user, 'Report User' button is shown and users can report other users by providing valid excuses.
android-refactor-home-page	Fahri Can Şanlı	The branch is created to increase the speed of our application. Since our home page includes many feature, many data is retrieved during the opening of our application. Performance is increased with the development made in this branch.
android-comment-bug	Fahri Can Şanlı	A minor bug related to comment feature is fixed.
android-rating-update	Fahri Can Şanlı	The average rating is updated automatically when a new rating is made for a user.
android-language-flags	Fahri Can Şanlı	The flag images are added to the application. And a bug deleting a language feature is fixed.
android-message-design	Fahri Can Şanlı	This branch aimed providing a better chat experience to the users. Therefore, layout files are changed related to chat and a new design is applied.
android-writing-suggestion	Fahri Can Şanlı	Writing topic suggestion feature is added to our application with this branch.

Table 3: Android Branches

Branch Name	Author	Brief Explanation
android-search-user	Sümeyra Yılmaz	In this branch, I added search user functionality. I created the related models for user search results. I designed a UI which the search results shown in a list and shows the usernames. List elements are clickable, the user can visit the related user's profile page. The profile page for visit contains the username, name, surname, bio, profile image, visited user's languages and their progress. Also user can see the comments of the visited one, add comment, send messages, and report that user. This four functionality and image upload added in different branches. But last corrections made in this branch.
android-image-upload-profile	Sümeyra Yılmaz	In this branch, I implemented the image related models and backend calls. I first added profile image upload and display. I used Picasso for this purpose. User can click on her profile image and can select an image from gallery. In order to make this, user should allow app permission to read from gallery. Later, this functionality is added to the visited profile page. When I was dealing with the image upload I handled some errors, like hide and show option for long bios.
android-selected-tab	Sümeyra Yılmaz	After milestone 2, we changed the android UI completely. In the former version, there were only one main page that user sees after sing in. We changed this as tabbed activity. When another activity is created and returns to the main screen it was returning to the first tab. I changed this as any activity returns to the tab where it is opened. I did some changes that I can make in different branches but for ease I did there. I connected profile page visit and messaging. I also corrected search for quizzes and writings since I noticed the changes made on backend for search calls.
android-image-upload-writing	Sümeyra Yılmaz	I added the image select and upload as a writing. I changed the writing submit page and created the models and calls for image writing submission. I also changed both the models and the UI for evaluating writings pages and the page that shows the user's writings and scores. I also fixed the bug I noticed in the profile page visit.
android-dev-merge-errors	Sümeyra Yılmaz	I used this branch to resolve the conflicts between android-dev and master branches in order to merge all project to the master.

Table 4: Android Branches

12 Evaluation of Tools and Processes

Tools And Processes	Evaluation
Github	Github is our main and essential tool while developing our application. We put our weekly meeting notes there and create issues frequently to check progress of our tasks. For this purpose, we mainly use projects part of the github. Also, since this is a group project, it is impossible to code it using only one computer. Therefore, we use github to store our actual code and create branches from it to develop multiple features at the same time.
Pull Request	As we stated above, different team member creates different branches to develop different features of our application. For that reason, we benefits from pull request to give other people a chance to review each other's code before merge them with master.
Code Review	Code review is almost one of the most important process when developing our project. Since, different team members takes responsibility for different parts of our application, people may make mistakes when using some functionality provided by their teammates. Hence, we decided to use code review for each feature we develop to prevent mistakes.
Postman	Postman is a tool that we used for developing backend part of our project.As we all know, our backend includes different endpoints and different request types such as get and post. Postman provides us the environment to test our code by sending different request with different contents to our endpoints and see the responses.
Whatsapp and Slack	Since we work as a team, we must be in touch at all times to let everyone be aware of improvements we have achieved and the difficulties we have faced. Sometimes, we also use this communication channels to make some important decisions that shows up after our meetings.
React & Redux	React is "A JavaScript library for building user interfaces". It is component based and so, we can effectively share the work among the team members. Redux is a predictable state container for JavaScript apps. So we can easily create a global state and exploit it in every component. Using react and redux together provides us a good and simple code structure and a good environment to add new features in a easy way.
Visual Studio Code	To develop frontend, we needed a development environment. We haven't decided a common IDE but Visual Studio Code is one of them. Visual Studio Code is a good environment to develop React applications since it has useful extensions for JavaScript, JSX and React itself.
Spring Boot	Spring Boot is an open source Java-based framework used to create a micro Service. We have used Spring Boot since most of our backend team members are familiar with it. It has minimum amount of configuration compared to other frameworks and it is easily deployed and easy to use with Maven as a package manager. All and all it is very efficient to develop using Spring Boot with its inner features.
IntelliJ	IntelliJ IDEA is a special programming environment or integrated development environment (IDE) largely meant for Java. IntelliJ makes it so easy to test, run and compile our Java code and also it works well with maven.
Android Studio	Android Studio provides the fastest tools to build an Android project. As Android team, we worked on it as an IDE. We created both activites, resources and our backend connection models in Android Studio. It provides us a simulator to see the app inside the phone, test its functionalities. We use this simulator at demo session too. Also, it can be connected to a phone, hence one can easily build and create a project in her/his Android phone.
Retrofit	Retrofit is a type-safe HTTP client for Android and Java. Retrofit models REST endpoints as Java interfaces, making them very simple to understand and consume. It provides a convenient builder for constructing our required objects.

Table 5: Evaluation of Tools and Processes