CmpE451

Nov 2019

Group10

Milestone 2: Group Deliverables

Contents

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.

1.3 What is next?

We have deployed a lot of impactful features in this milestone. As we get close to final milestone, we are aiming to improve those features like fixing bugs, creating better UI pages and correcting some response models. In addition to these features, we have to feed them with rich contents, we aim to create quality contents for quizzes and writings.

For new features, we have left not much, however new features are going to be big. We will implement annotation feature in forthcoming weeks. We are planning to use annotations in the writing section because of rich contents. Another big feature will be messaging. We will add messaging between users so that they learn via communication. This messaging will allow users to communicate with native people.

We are going to implement admin functionalities so that any illegal actions in the system will be punished. Admin users will be delete users. Also we will give some options to users. Users will be able to delete their own profiles whenever they want. In addition to that, users will have option to hide their information.

2 List and Status of Deliverables

Name	Delivery Date	Delivery Status
Backend/Frontend/Android Codes	Nov 25, 2019	Delivered
User Scenarios (Demo)	Nov 26, 2019	Delivered
Project Plan	Dec 1, 2019	Delivered
API Documentation	Dec 1, 2019	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 completed a writing task and saw a previous writing task's evaluation and the second one was Burak Tepedelen who is a teacher of English and wants to help the students on YALLP. He evaluated Süreyya's writing task and added new languages.
- **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 writing exercises and search mechanism. Currently we are up do date on code parts, and we are ready to go further.
- **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 update our plan to keep track of the progression. It is a high level plan. In our project plan we have 3 major tasks to deliver. Annotations, messaging feature and admin user functionalities. We shared our project plan with the customer also we are planning to finish them by the end of the semester and in the final milestone we will demo them to the customer approriately.
- **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

4 Work-Done by Team Members

Team Member	Contributions
Yavuz Demir	I created an endpoint for member to add and remove languages. I enabled authentication while using swagger.io . I created semantic search functionality for both quizzes and writings. I added native language field to user I created github issues and opened branches related to these issues.
Murat Buldu	After milestone 1, I implemented language selection part in frontend, so that users can start learning new languages and drop a language they have already started. The language of the quizes and writing exercises shown in the home page can be changed in the header. After these language options, I implemented search functionality for quiz and writing exercises. I also updated the user interface of quiz list and writing list in the home page.
Fahri Can Şanlı	I attended all the meetings (both with the customer and with the team) actively. I implemented add language by myself and I also added the delete language feature to the profile page. I created language and writing retro clients and apis for linking up our app with backend language endpoints. I also implemented the first version of quiz list and the three dots custom view in the profile page where user can reach edit profile page, writings that he/she completed,writings that are waiting to be evaluated by the user and log out functionality. Besides, I implemented the page where user can see the writings that he/she completed before as a list with the status of the writing and score if it is evaluated and when user clicks on one of them, detailed version of the writing appears. I also implemented the page where user can see the writings that are waiting to be evaluated by himself/herself. Upon clicking on one of them, detailed version appears and user can give a score to it.
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.
Sümeyra Yılmaz	After milesotone 1, we changed and added some many things in out activity pages. I edited the profile page,in order to complete it and add new facilities. In the previous page, language list was static, I changed it as it shows user's added languages, and its levels. Also the add new language property is depending on the users current languages. The bio part is added, if it is too long, it only shows a small part of it with a see full button. Also, in previous design, if user clicks on a language a quiz is suddenly pop up to solve, and there was only one quiz. I changed this as, if a user clicks on a language, a tabbed activity is shown, a tab for quizzes and a tab for writings. In quiz tab it shows all quizzes with level or lower as user's level for this language, and in writings tab, all writings for this language. For all quizzes and writings its status also shown, as solved with its score, not solved, and pending for writings.These lists also have search property, with a search bar on top. I implemented this search property and dynamic lists for quizzes and writings.

Table 1: Work-done by Each Team member

Team Member	Contributions
Selim Karaduman	Implemented the endpoint for member update functionality Then, added documentation interface by using swagger library. Just after that, have written the corresponing documentations. After those, implemented the endpoints for the writing requirements. It includes all the requirements however, it needs to be updated according to the feedback given in Milestone. For all of the above endpoints: implemented the corresponding models,repositories, services and stored procedures. Also, imported necessary writings into the database. Have written the documentation of the endpoints for the milestone. Opened issues on github, and used branches to develop different functionalities.
Melih Demir	I have been developing the Android part of the project. In particular, I have been working on the profile page part of the application. After Milestone I the profile page started looking a lot different than before. The user's language dashboard is a lot more interactive now. Users now can change what they display on their profile. Also, writing exercises are now a part of the app and user can fill and submit a writing assignment.
Hatice Melike Ecevit	Created the new tables for each feature. I maintain the database relations between the entities and decide for which is best. Fixed the level name issue between the clients and backend. and implemented the member model-repository-service and controller for each feature I developed. I compiled and deployed our application using AWS Elastic Beanstalk I implemented the quiz result feature and also I dealed with internal server errors. Also I created the DTO classes and DTO Converter Services for clients and 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. Made frontend portion of the milestone scenario.

Table 2: Work-done by Each Team member

5 Requirements

5.1 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

5.2 Functional Requirements

5.2.1 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 tutorial on each language.
    - Guest users shall be able to see options to sync with Google.
    - 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 that are interacted with through a writing assignment or communication mechanism(message).
    - Users shall be found on the search bar.
    - Users shall be able to learn as much new languages as they want.

        * Users shall post comments to writing materials that they receive from other users.
        * Users shall be able to annotate writing exercises.
        * 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 on a topic.
        * Users shall be able to annotate images and texts.
        * Users shall be able to post comments to other users.
        * Users shall be able learn and evaluate others' writing exercises simultaneously.
        * User can select one of the materials he prefers to do.
        * User can select the user who will review his writing exercise.
        * User can 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.
        * The users shall be able to sign up via Google.
    - Sign In
        * The users shall be able to sign in anytime by using username-password pair or email-password pair.
        * The users shall be able to sign in via Google if they signed up via Google.

- 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 notifications preferences.
        * 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 each other if the receiver user accepts the sender's message.
    - The users shall get notifications when they get a message from other users if they have chosen to get notifications.

- Recommendation and Contribution
    - The users shall be able to recommend new learning materials to the system.

5.2.2 System Requirements

- Search
  - System shall provide a search algorithm through keywords and show the related content.
  - System shall provide semantic search by an algorithm that selects materials by topic, type, difficulty and scope, and show the related content.
- Recommendation
  - System shall recommend users to send review.
- Annotation
  - System shall support the W3C Web Annotation Data Model.
- Account
  - System shall have a hide option to allow users to disable sharing functionality of their profile pages.
- 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.

5.3 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.
- 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 database shall be backed up daily and weekly.
  - The system should have monthly maintenance period of 3 hours.
  - The system shall recover fully from a crash in at most 24 hours.

6 API Documentation

6.1 Authentication and Registration

6.2 /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...."
}

6.3 /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...."
}

6.4 Member Operations

6.5 /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
}
```

6.6 /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
}
]
```

6.7 /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
}
]
```

6.8 /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,
"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"
}
```

6.9 /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"
}

6.10 Language Operations

6.11 /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”
}
]

6.12 /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”
}
]

6.13 Quiz Operations

6.14 /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
},
}
```

6.15 /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
},
}
```

6.16 /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
}
```

6.17 /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
}
]
```

6.18 /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
}
]
```

6.19 /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
}
```

6.20 /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
}
]
```

6.21 /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
}
]
```

6.22 Search Operations

6.23 /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
}
]
```

6.24 /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"
}
}
]
```

6.25 Writing Operations

6.26 /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”
}
}

6.27 /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"
}

6.28 /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"
}
]
```

6.29 /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"
}
]
```

6.30 /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"
}
}
]
```

6.31 /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"
}
}
```

6.32 /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"
}
]
```

6.33 /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"
}

7 Project Plan

	Name	Duration	Start	Finish	Predecessors	Resource Names
1	**Revision of The Project**	**5 days**	**9/25/19 8:00 AM**	**10/1/19 5:00 PM**		
2	Review of the Github project and creating own wiki page	5 days	9/25/19 8:00 AM	10/1/19 5:00 PM		Hatice Melike Ecevit
3	Review of the Github project and creating own wiki page	5 days	9/25/19 8:00 AM	10/1/19 5:00 PM		Selim Karaduman
4	Review of the Github project and creating own wiki page	5 days	9/25/19 8:00 AM	10/1/19 5:00 PM		Burak Tepedelen
5	Refining questions to be asked to the customer	5 days	9/25/19 8:00 AM	10/1/19 5:00 PM		Melih Demir
6	Refining questions to be asked to the customer	5 days	9/25/19 8:00 AM	10/1/19 5:00 PM		Sümeyra Yilmaz
7	Revision of the class diagrams and the project plan	5 days	9/25/19 8:00 AM	10/1/19 5:00 PM		Hasan Basri Balaban
8	Revision of the requirements	5 days	9/25/19 8:00 AM	10/1/19 5:00 PM		Fahri Can Sanli
9	Updating the project plan	5 days	9/25/19 8:00 AM	10/1/19 5:00 PM		Murat Buldu
10	Revision of the mockups	5 days	9/25/19 8:00 AM	10/1/19 5:00 PM		Yavuz Demir
11	**Backend**	**60 days**	**10/2/19 8:00 AM**	**12/24/19 5:00 PM**	**1**	**Hatice Melike Ecevit;Selim Karaduman;Yavuz Demir**
12	Authentication for backend	5 days	10/2/19 8:00 AM	10/8/19 5:00 PM		Selim Karaduman
13	Necessary tables for login and sign up will be created	5 days	10/2/19 8:00 AM	10/8/19 5:00 PM		Hatice Melike Ecevit
14	Login and sign up functions for backend	5 days	10/9/19 8:00 AM	10/15/19 5:00 PM	13	Yavuz Demir
15	Authentication bugfix	5 days	10/9/19 8:00 AM	10/15/19 5:00 PM	12	Yavuz Demir
16	Creation of necessary quiz and question classes for backend	5 days	10/9/19 8:00 AM	10/15/19 5:00 PM		Selim Karaduman
17	Creation of quiz related tables such as questions and quizes	5 days	10/16/19 8:00 AM	10/22/19 5:00 PM	16	Hatice Melike Ecevit
18	Start of Unit Testing in backend	5 days	10/16/19 8:00 AM	10/22/19 5:00 PM		Yavuz Demir
19	Quiz score calculations	5 days	10/16/19 8:00 AM	10/22/19 5:00 PM	16	Selim Karaduman
20	Addition of language type	5 days	10/16/19 8:00 AM	10/22/19 5:00 PM	16	Hatice Melike Ecevit
21	Update profile	5 days	10/30/19 8:00 AM	11/5/19 5:00 PM		Yavuz Demir
22	Update profile	5 days	10/30/19 8:00 AM	11/5/19 5:00 PM		Selim Karaduman
23	Update profile	5 days	10/30/19 8:00 AM	11/5/19 5:00 PM		Hatice Melike Ecevit
24	Rearranging quizes in database	5 days	11/6/19 8:00 AM	11/12/19 5:00 PM		Yavuz Demir
25	Updating quiz endpoints	5 days	11/6/19 8:00 AM	11/12/19 5:00 PM		Selim Karaduman
26	Updating profile endpoint	5 days	11/6/19 8:00 AM	11/12/19 5:00 PM		Hatice Melike Ecevit
27	Implementing search function	10 days	11/13/19 8:00 AM	11/26/19 5:00 PM		Yavuz Demir
28	Updating endpoints for tests	5 days	11/13/19 8:00 AM	11/19/19 5:00 PM		Selim Karaduman
29	Implementation of adding and removing languages from the user	5 days	11/13/19 8:00 AM	11/19/19 5:00 PM		Hatice Melike Ecevit
30	Update writing endpoints for backend	5 days	11/20/19 8:00 AM	11/26/19 5:00 PM		Selim Karaduman
31	Missing endpoints for languages for backend	5 days	11/20/19 8:00 AM	11/26/19 5:00 PM		Hatice Melike Ecevit
32	Messaging Functionality	10 days	11/27/19 8:00 AM	12/10/19 5:00 PM		
33	User Writing Topic Recommendation	10 days	11/27/19 8:00 AM	12/10/19 5:00 PM		
34	Notification system	10 days	11/27/19 8:00 AM	12/10/19 5:00 PM		
35	Annotation on Writings	10 days	12/11/19 8:00 AM	12/24/19 5:00 PM		
36	Hide Profile info	10 days	12/11/19 8:00 AM	12/24/19 5:00 PM		
37	Admin User Functionalities	10 days	12/11/19 8:00 AM	12/24/19 5:00 PM		
38	**Mobile**	**40 days**	**10/2/19 8:00 AM**	**11/26/19 5:00 PM**	**1**	**Fahri Can Sanli;Melih Demir;Sümeyra Yilmaz**
39	Sign up/Get Started page for Android	5 days	10/2/19 8:00 AM	10/8/19 5:00 PM		Fahri Can Sanli
40	Profile page for Android	5 days	10/2/19 8:00 AM	10/8/19 5:00 PM		Melih Demir
41	Home and login page for Android	5 days	10/2/19 8:00 AM	10/8/19 5:00 PM		Sümeyra Yilmaz
42	Create quiz activity for Android, bug fix sign up activity	5 days	10/9/19 8:00 AM	10/15/19 5:00 PM		Fahri Can Sanli
43	Bug fix profile page for Android	5 days	10/9/19 8:00 AM	10/15/19 5:00 PM		Melih Demir
44	Bug fix sign in activity for Android, update requirements	5 days	10/9/19 8:00 AM	10/15/19 5:00 PM		Sümeyra Yilmaz
45	Refactoring and developing quiz page, storing token when user sings in or signs up	5 days	10/16/19 8:00 AM	10/22/19 5:00 PM	42;43;44	Fahri Can Sanli
46	Connect profile page with backend	5 days	10/16/19 8:00 AM	10/22/19 5:00 PM	40	Melih Demir
47	Create quiz score page and connect it with backend, logout, update requirements	5 days	10/16/19 8:00 AM	10/22/19 5:00 PM	42;43;44	Sümeyra Yilmaz
48	Update profile	5 days	10/30/19 8:00 AM	11/5/19 5:00 PM		Fahri Can Sanli
49	Update profile	5 days	10/30/19 8:00 AM	11/5/19 5:00 PM		Melih Demir
50	Update profile	5 days	10/30/19 8:00 AM	11/5/19 5:00 PM		Sümeyra Yilmaz
51	Updating quiz requests	5 days	11/6/19 8:00 AM	11/12/19 5:00 PM		Fahri Can Sanli
52	Updating profile request	5 days	11/6/19 8:00 AM	11/12/19 5:00 PM		Melih Demir
53	User interface updates in home page	5 days	11/6/19 8:00 AM	11/12/19 5:00 PM		Sümeyra Yilmaz
54	Implementing adding new languages	5 days	11/13/19 8:00 AM	11/19/19 5:00 PM		Fahri Can Sanli
55	Implementing editing profile page on Android	5 days	11/13/19 8:00 AM	11/19/19 5:00 PM		Melih Demir
56	Updating profile page fields	5 days	11/13/19 8:00 AM	11/19/19 5:00 PM		Sümeyra Yilmaz
57	Writing evaluation page for android	5 days	11/20/19 8:00 AM	11/26/19 5:00 PM		Fahri Can Sanli
58	Writing assignment page for android	5 days	11/20/19 8:00 AM	11/26/19 5:00 PM		Melih Demir
59	Search mechanism for android	5 days	11/20/19 8:00 AM	11/26/19 5:00 PM		Sümeyra Yilmaz
60	Messaging Functionality	10 days	11/27/19 8:00 AM	12/10/19 5:00 PM		
61	User Writing Topic Recommendation	10 days	11/27/19 8:00 AM	12/10/19 5:00 PM		
62	Notification system	10 days	11/27/19 8:00 AM	12/10/19 5:00 PM		
63	Annotation on Writings	10 days	12/11/19 8:00 AM	12/24/19 5:00 PM		
64	Hide Profile info	10 days	12/11/19 8:00 AM	12/24/19 5:00 PM		
65	Admin User Functionalities	10 days	12/11/19 8:00 AM	12/24/19 5:00 PM		
66	**Frontend**	**60 days**	**10/2/19 8:00 AM**	**12/24/19 5:00 PM**	**1**	**Burak Tepedelen;Hasan Basri Balaban;Murat Buldu**
67	Login, sign up for frontend	5 days	10/2/19 8:00 AM	10/8/19 5:00 PM		Burak Tepedelen
68	Opening page for frontend	5 days	10/2/19 8:00 AM	10/8/19 5:00 PM		Murat Buldu
69	Home and profile page	5 days	10/2/19 8:00 AM	10/8/19 5:00 PM		Hasan Basri Balaban
70	Connecting backend to frontend	5 days	10/9/19 8:00 AM	10/15/19 5:00 PM	67;68;69	Burak Tepedelen
71	Question page for frontend	5 days	10/9/19 8:00 AM	10/15/19 5:00 PM		Murat Buldu
72	Revisions of the general look and profile page	5 days	10/9/19 8:00 AM	10/15/19 5:00 PM	69	Hasan Basri Balaban
73	Login/signup update in frontend	5 days	10/16/19 8:00 AM	10/22/19 5:00 PM	67	Burak Tepedelen
74	Combining frontend pages	5 days	10/16/19 8:00 AM	10/22/19 5:00 PM	67;68;69;71	Murat Buldu
75	Home page update in frontend	5 days	10/16/19 8:00 AM	10/22/19 5:00 PM	69	Hasan Basri Balaban
76	Update profile	5 days	10/30/19 8:00 AM	11/5/19 5:00 PM		Burak Tepedelen
77	Update profile	5 days	10/30/19 8:00 AM	11/5/19 5:00 PM		Murat Buldu
78	Update profile	5 days	10/30/19 8:00 AM	11/5/19 5:00 PM		Hasan Basri Balaban
79	User interface updates	5 days	11/6/19 8:00 AM	11/12/19 5:00 PM		Burak Tepedelen
80	Updating quiz requests	5 days	11/6/19 8:00 AM	11/12/19 5:00 PM		Murat Buldu
81	Updating profile requst	5 days	11/6/19 8:00 AM	11/12/19 5:00 PM		Hasan Basri Balaban
82	Search Functionality and update quiz	5 days	11/13/19 8:00 AM	11/19/19 5:00 PM		Murat Buldu
83	Update profile and add missing features	5 days	11/13/19 8:00 AM	11/19/19 5:00 PM		Hasan Basri Balaban
84	Writing Functionality	5 days	11/13/19 8:00 AM	11/19/19 5:00 PM		Burak Tepedelen
85	Search mechanism for frontend	5 days	11/20/19 8:00 AM	11/26/19 5:00 PM		Murat Buldu
86	Update profile page for frontend	5 days	11/20/19 8:00 AM	11/26/19 5:00 PM		Hasan Basri Balaban
87	Writing evaluation for frontend	5 days	11/20/19 8:00 AM	11/26/19 5:00 PM		Burak Tepedelen
88	Messaging Functionality	10 days	11/27/19 8:00 AM	12/10/19 5:00 PM		
89	User Writing Topic Recommendation	10 days	11/27/19 8:00 AM	12/10/19 5:00 PM		
90	Notification system	10 days	11/27/19 8:00 AM	12/10/19 5:00 PM		
91	Annotation on Writings	10 days	12/11/19 8:00 AM	12/24/19 5:00 PM		
92	Hide Profile info	10 days	12/11/19 8:00 AM	12/24/19 5:00 PM		
93	Admin User Functionalities	10 days	12/11/19 8:00 AM	12/24/19 5:00 PM		
94	**Milestone 1**	**3 days**	**10/22/19 8:00 AM**	**10/24/19 5:00 PM**		
95	Executive summary	3 days	10/22/19 8:00 AM	10/24/19 5:00 PM		Yavuz Demir
96	Project plan	3 days	10/22/19 8:00 AM	10/24/19 5:00 PM		Murat Buldu
97	Work done	3 days	10/22/19 8:00 AM	10/24/19 5:00 PM		Hasan Basri Balaban
98	List and status of deliverables	3 days	10/22/19 8:00 AM	10/24/19 5:00 PM		Fahri Can Sanli
99	User scenarios	3 days	10/22/19 8:00 AM	10/24/19 5:00 PM		Melih Demir
100	Requirements	3 days	10/22/19 8:00 AM	10/24/19 5:00 PM		Sümeyra Yilmaz
101	API Documentation	3 days	10/22/19 8:00 AM	10/24/19 5:00 PM		Selim Karaduman
102	Code structure of backend	3 days	10/22/19 8:00 AM	10/24/19 5:00 PM		Hatice Melike Ecevit
103	Code structure of frontend	3 days	10/22/19 8:00 AM	10/24/19 5:00 PM		Burak Tepedelen

YALLP

8 User Scenarios

8.1 Android Scenario

Süreyya Yıldı is an aspiring high school graduate. She is preparing to study in a college where all lectures are given in Turkish. Due to this fact she has no way of gaining experience on English through her college. However, she wants to attend to a Work and Travel program abroad, so she really needs to find another way to teach herself the language. After doing a quick research she finds YALLP as a very suitable platform for new learners. She had signed up our app before and since her level was beginner, she decides to start taking writing exercises to improve her writing skills. Firstly she checks the status of the writings that she completed before. Afterwards, she opens the writing topics in English and selects one topic about children and completes her essay. Then, she clicks a reviewer and submits her essay to be evaluated.

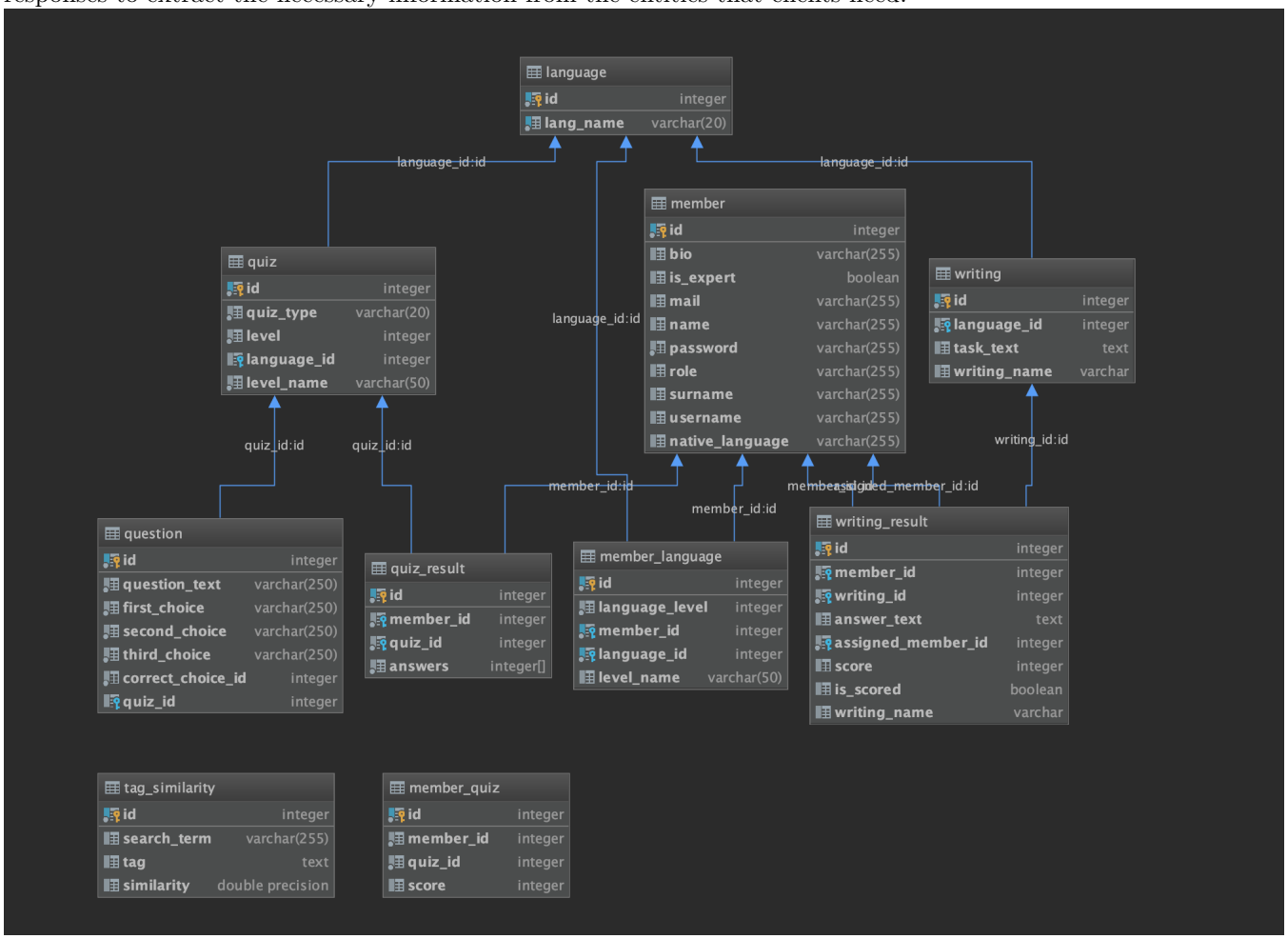
8.2 Frontend Scenario

Burak Tepedelen is a English teacher. He uses YALLP to practice Spanish and help his English students. He especially enjoys reviewing writing exercises and giving constructive feedback. He uses YALLP for a while but haven't updated his profile page. After opening the site he goes to profile page,he fills his name,surname and a short bio then he visit writing review page. He notices a new writing assignment from Sürayya Yıldı,he reads it and liking it very much scores it 10 out of 10 then returns to homepage.

9 Code Structure and Group Process

9.1 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.

9.2 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.

9.3 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-complete-profile-page	Sümeyra Yılmaz	Static language list is removed. A dynamic list is added instead. Add language button is now depends on user's current languages. Bio, name and surname options added. For long bios, show/hide bio option is added.
android-search	Sümeyra Yılmaz	Dashboard for a specific language is completely changed. For a language, quizzes and writings are shown in two tabbed activity. Quiz tab, shows all quizzes with a level equal to or lower than user's current level for this language. Writing tab, shows all writings for this language. Both tabs shows each item's current status and score if it's scored.Status can be solved and not solved for quizzes,and in addition pending for writings. If any writing or quiz is clicked, the corresponding activity to solve it is opened. Each tab has search property. Result of the search is in the same list format.
android-adding-new-language	Fahri Can Şanlı	Some necessary changes are made to make code more readable. Adding new language activity is created. Language api and retro client are created. Language list adapter is created.
android-quiz-list	Fahri Can Şanlı	First version of the quiz list is created. At the first milestone, when user clicked a language, quiz to determine level was appearing automatically. With the development in this branch, a list of quizzes are shown to the user. Each list item was composed of quiz topic, quiz id, status of quiz which shows if it is solved by the user and score.
android-edit-profile-page	Melih Demir	Users can now update the information they have on their profile page. User's name, surname, bio and password can be changed.
android-profile-options	Fahri Can Şanlı	A custom three dots view is created and placed in profile page Since all of our features are accessible from profile page, as android team we decided to make these features accessible from three dots.
android-writing-list-of-user	Fahri Can Şanlı	A page where user can see his/her previously completed writing exercises are shown is created. Writing api, retro client and necessary list adapter are created.
android-details-backup	Fahri Can Şanlı	A page where user can see his/her detailed version of one of the previously completed writing exercises is shown is created. To be able to do that, one activity is created and since score page will be similar to this one, two different fragments are created to write reusable code.
android-score-page-backup	Fahri Can Şanlı	A page where user can see writing assignments that are assigned to him/her. When user selects one of them, a new page appears which is created by using fragments. User can see detailed version of the writing exercise and give score to it.
android-delete-language	Fahri Can Şanlı	A trash icon is added to each language listed in the profile page. When user wants to drop one language, he/she clicks the trash icon and that specific language is removed from the list of languages of the user.
android-writing-activity	Melih Demir	A page is created to let people do writing assignments and submit them to an evaluator they choose from a list of recommended users.

Table 3: Android Branches

10 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 4: Evaluation of Tools and Processes