CMPE 451 - MILESTONE 1

1. Executive Summary

EyeTrade is an application program which aims to create a useful and interactive environment for people who want to follow recent trends of major investment tools as well as those investors who want to make an investment in certain trading equipment. This platform supports various functionalities including following the instant progress of various trading equipment, following important economic events and sharing comments on articles which are written by expert investors and many more.

Until now, we have completed the design process of our project including requirements, user scenarios, class diagram etc. during last semester. In the beginning of this semester, we have made a complete review of our requirements and mock-ups to clarify the questions on them as well as to introduce the project to new team members. As a result of this review, we made necessary changes on the requirements and mock-ups and we make sure that all the members in the team is on the same line.

After finishing the review part, we had a discussion on which tools should be used in the development phase of the project including backend, frontend and mobile sections by considering the requirements that are set by the customer. After the determination of the tools, the team is splited into three parts including frontend, backend and mobile teams. As of 27th of October, we have completed the planned tasks for the first milestone. We have a working API for users to login which checks the credentials and a sign up which registers new users by the required inputs. In terms of frontend and android, we have also adequate pages and designs for these functionalities. In addition to these, we have implemented a mock page for events section and a draft for the profile page of users. The overall summary of the implemented parts for each sub-team can be found in the following sections.

1.1. Summary Subpart for Backend

- **Learned Spring Boot:** For the backend application we decided to use Spring Boot. However, some of our teammates did not know about Spring Framework so we set a tutorial meeting. Then, we talked about the most critical libraries of Spring, how to use the annotations and beans in Spring, and we discussed the best practices of designing a proper backend architecture. After the tutorial meeting, everyone in the team implemented simple example APIs using Spring Boot.
- **Deployment of Backend Application and Database:** We decided to use AWS educate accounts to deploy our backend and databases. Then, we created an EC2 instance and deployed the backend application to it where the application could be accessed on the port 8080. When we needed to deploy also our PostgreSQL database, we first decided to deploy it to the same EC2 machine but after we saw that the

performance of EC2 machine is very limited, due to student account, we decided to open an RDS instance and deploy the database server to that instance.

- **Developing the Most Critical Features:** Before the milestone we, as a whole team, decided to prioritize some features to develop until the deadline. Therefore, we selected the Registration, Login, Password Reset, Currency Convert, and Authentication features. Finally, we implemented those features by sharing the tasks among us.
- **Developing the Authentication Mechanism:** For the authentication, we first thought that Spring Security Framework is a proper choice. However, because of the different user types in our project (where each type has different authorizations for the features) integration of Spring Security was hard. Nonetheless, we were still thinking about using Spring Security but when we realize the need for user dependency of our functionalities (For example, if a user wants to see her friends then we need the know the user's username for retrieving that information) we decided to use JWT authentication without Spring Security. Then, we implemented two authentication sub-mechanism for both trader and basic user types.

1.2. Summary Subpart for Android

- **Login - Register:** We implemented Login and registration features for our Android app. Basic users can register with their basic info while trading users need to provide their ID and IBAN number additionally.
- **Homepage:** We enriched homepage with mock data consisting some economic events.
- **Navigation Drawer:** We created a navigation drawer to be able to navigate through different pages in the app.
- **Events and Article Page:** We used some mock data to present articles and events about economics. Users can go to those pages via navigation drawer.
- **Currency Converter:** We implemented currency converter which can convert one of the four different currencies to another.

1.3. Summary Subpart for Frontend

- **Login - Register:** Implementation of login and register pages by checking the validity of input fields.
- **Homepage:** Implementation of a welcome page on web frontend.
- **Navigation Drawer:** Implementation of a navigator to navigate through various pages.
- **Events Page:** We have implemented a mock events page to show the functionality to our customer.

- <u>**Currency Converter:**</u> Implementation of a currency converter page which is connected to backend.

1.4. Changes Planned

- Completion of home page for web frontend and android.
- Completion of profile page for android/web frontend and connection with backend.
- Implementation of following and unfollowing users functionality.
- Implementation of Google location functionality.
- Implementation of Articles page and connection with backend.

2. List Status of Deliverables

Web:

- Homepage: Works fine.
- Login/Sign-up: Works fine except login/sign-up with Google option is not functional for now.
- Currency Page: Works fine.
- Events Page: It is not connected to backend, only a mock page for now.

Mobile(Android):

- Homepage: Works fine.
- Login/Sign-up: Works fine except login/sign-up with Google option is not functional for now.
- Currency Page: Works fine.
- Events Page: It is not connected to backend, only a mock page for now and it's content should change.
- Articles Page: It is not connected to backend, only a mock page for now.

Backend:

- Registration: Works fine.
- Authentication: Works fine.
- Login: Works fine.
- Password reset: Works fine.
- Currency: Works fine.
- Email confirmation: Implemented but confirmation links has some problems, can be fixed in a small amount of time.

3. Evaluation of Status of Deliverables and Its Impact on Plan

- **Homepage**

We have coded the basis of our homepages and navigation throughout the application. New pages can be built and easily integrated into the system. A simple navigation system is important for an application with many pages, so we opted for a vertical list of all pages.

- **Login/Sign Up**

Users can sign up either as a Basic User or a Trader User. They can login with their account or choose to continue as a guest. Accounts allow users to follow each other's portfolios and write articles -though not implemented yet-, and letting the user continue as a guest instead of signing up lets them experience the application without creating an account, potentially increasing their likeliness to sign up.

- **Currency**

Users can see the equivalent of any amount of money in another currency. This is an essential feature for an application that deals with currency trading, as users are constantly buying and selling.

- **Events/Articles**

We have created events (Web/Android) and articles (Android) pages as mock pages. This let us show the customer how they will look once we have implemented those features. Once they are done, events will inform the users and articles will let the users share their own ideas and opinions.

- **Email Confirmation**

We have coded the email confirmation, with a few problems remaining. Clicking them lets us confirm that the email the user entered while registering is a real email. This is important in case we want to send the user updates on occasion, or when they forget their passwords and need a password reset link.

- **Password Reset**

Backend implementation is complete. We just need to add a new page asking for the user's email. It is important in the case that the user loses access to their account, either by forgetting their password or having their password changed by some third-party who managed to access their account.

4. Work Done by Each Member

Oğuzhan Kırlar	● Create API for backend ● Development of authentication feature of the backend application ● Development of login-logout feature on the backend ● Research on Docker containers ● Development of changing password feature on the backend
Alican Şafak Özdek	● Implemented basic frontend for the first week's mock API. ● Helped backend team while deploying API on AWS. ● Deployed frontend on AWS. ● Designed/reorganized navigation bar, header, footer, homepage, login-page of CoreUI template for web. ● Implemented/edited index.js for directing from one page to another. ● Designed and implemented currency page for web. ● Wrote meeting report #13 and #17 ● Wrote "List and Status of Deliverables" part for Milestone report.
Anıl Can Kara	● Designed login page on Android. ● Implemented login features and related network requests. ● Designed currency converter page on Android. ● Implemented currency converter by sending necessary network requests. ● Implemented network requests for our mock API.
Merve İlik	● Designed Events and Article pages on Android app. ● Implemented related classes to visit those pages. ● Integrated mock data.
M.Emir Gökdemir	● Controller layer, Service Layer and Data Layer are initialized. ● Constants, Mappers are initialized

	● User Controller,Service and Model classes are implemented. ● Registration property is implemented. ● Mail address is taken. ● Reset password property is implemented. ● Send Token property is implemented. ● Authorization and authentication properties are implemented. These features are edited laterly by me and Kutay. ● Reorganized initial codes with Kutay. ● Login property implemented with Kutay. ● Currency Converting property implemented. ● Swagger and Cors Configurations are implemented. (These are updated by Kutay later)
Alihan Çelikcan	● Designed sign up page on Android ● Implemented signing up as a Basic User or a Trader User ● Added “continue as a guest” button to the login page ● Implemented the navigation drawer ● Helped with articles and events mock page on Android ● Helped with login and sign up on web ● Wrote the “Evaluation of Status of Deliverables and Its Impact on Plan” part for the Milestone Report

Seyfi Kutay Kılıç	● Designed a general architecture for the backend application. ● Deployed the backend application to AWS EC2, every week. ● Created an initial backend application with it's dependencies, which is re-implemented by Emir later. ● Integrated a security feature to the backend application. ● Developed a better authentication mechanism, with Emir. ● Refactored the backend code to make it more maintainable, with Emir. ● Developed the login feature, with Emir. ● Developed a communicator class, which sends http request to an external API, to retrieve the currency exchange rates. ● Updated the Swagger integration and fixed Cors in the backend application, which were configured by Emir before. ● Prepared an API documentation using Swagger. ● Wrote the "Backend Summary", "Api Documentation" and "Evaluation of the Tools" parts of the milestone document.
Kemal Duru	● I have learned Vue.js by implementing a simple task. ● I have implemented a mock events page on web frontend and routed it through navigator.. ● I have drafted a mock profile page on web frontend for the next milestone. ● Wrote meeting report #16. ● Wrote "Executive Summary" and "Project Management" parts for Milestone report.
M. Hakan Kurtoğlu	● Developed a dummy endpoint in backend for testing purposes ● Approval of pull requests for backend features and merging them with master ● Writing code and branch structure part in milestone report
Esra Yılmaz	● Checking and fixing the mockups ● Searching a template for Vue.js ● Designing welcome page with Vue.js ● Implementing forms for login and sign-up page

	● Implementing a code in Vue.js for post the data of login and sign-up page ● Implementing a code for getting response and showing errors on the pages, if any ● Implementing a code in Vue.js for showing or hiding a div according to basic or trading user. ● Adding a logo for our Platform ● Arranging routing of pages in Vue.js ● Preparing the Milestone1 and converting to PDF format ● Implementing a validation code for the forms

5. Requirements

GLOSSARY

- **All Users:** It can be a basic user or a trading user.
- **Annotation:** A term referring to documentation and comments that may be found on code logic.
- **Basic Users:** A person who is registered and able to access the basic functionalities of the platform.
- **Economic Event:** Events having an impact on trading equipments. (e.g., Meetings of the Central Bank of the Republic of Turkey or a speech of the president of Federal Reserve Bank of New York.)
- **My Investments:** The section in which users are able to invest on any trading equipment, make a buy order for a specified rate, and set stop/loss limits.
- **Prediction Success Rate:** The success rate of the predictions made by users on trading equipment's values which can be seen in the user profile.
- **Portfolio:** A grouping of financial assets such as stocks, bonds, commodities, currencies etc.
- **Profile:** The page which contains user information such as personal information, prediction success rate and portfolios.
- **Profit/Loss:** Amount whether the user lose or gain in any trading equipment.
- **Profit/Loss Section:** The section provided by the system for all users to view profit/loss value of their investments.
- **Recommendation System:** It recommends articles or trading equipments to the users based on their histories.

- **Semantic Search:** It tries to find semantically similar users and trading equipments based on the context information provided in the semantic tags.
- **Traders Platform:** Lets people follow and trade indices, stocks, ETFs, commodities, currencies, funds, bonds, and cryptocurrencies.
- **Trading Equipment:** Indices, stocks, ETFs, commodities, currencies, funds, bonds, cryptocurrencies etc.
- **Trading Users (Registered Users):** A person who is registered and, in addition to basic user, is able to invest in any trading equipment.

1.FUNCTIONAL REQUIREMENTS

1.1. User Requirements

1.1.1. User Types

1.1.1.1. All Users (except guests)

- 1.1.1.1.1. All users shall have a “Profit/Loss” and an "Events" section.
- 1.1.1.1.2. All users shall have at least one Portfolio.
- 1.1.1.1.3. All users shall be able to follow other users.
- 1.1.1.1.4. All users shall be able to share ideas as an article.
- 1.1.1.1.5. All users shall be able to comment about trading equipments and articles.
- 1.1.1.1.6. All users shall be able to follow the values related to any trading equipment that is available.
- 1.1.1.1.7. The followers of a user and the followed users by that user shall be seen on the profile of that user.
- 1.1.1.1.8. A private user profile shall be seen only by its followers.

1.1.1.2. Only Trading Users

- 1.1.1.2.1. Trading users shall have a “My Investments” section.
- 1.1.1.2.2. Trading users shall be able to invest on any trading equipment, make a buy order for a specified rate, and set stop-loss limits.
- 1.1.1.2.3. Trading users shall have different success rates for each trading equipment on which they made prediction.

1.1.1.3. Guests

- 1.1.1.3.1. Guests shall be able to view the price of a trading equipment and view the economic events, but shall not be able to follow them.

- 1.1.1.3.2. Guests shall be able to read user comments about trading equipment and articles.

1.1.2. Authentication

1.1.2.1. Sign Up

- 1.1.2.1.1. Basic users shall provide their name, surname, e-mail address, and location (by specifying it on Google Maps).
- 1.1.2.1.2. Trading users shall provide their name, surname, e-mail address, location (by specifying it on Google Maps), IBAN and TR Identity Number.
- 1.1.2.1.3. Users should be able to use Google account to retrieve necessary information for signing up.
- 1.1.2.1.4. All users shall choose a password.(see 2.1.2)

1.1.2.2. Sign In

- 1.1.2.2.1. All users shall be able to sign in by providing their e-mail and password.
- 1.1.2.2.2. All users should be able to use Google account to retrieve necessary information for signing in.

1.1.3. Profile

- 1.1.3.1. Profile can be either public or private to other users.
- 1.1.3.2. In order to see the contents in a private user profile, user should be followed.
- 1.1.3.3. Basic users and trading users' prediction success rate shall be visible in their profile page.

1.1.4. Portfolio

- 1.1.4.1. Basic users and trading users shall be able to rename their portfolio.
- 1.1.4.2. Basic users and trading users shall be able to add any trading equipment to their portfolio.
- 1.1.4.3. Basic users and trading users shall be able to share their portfolio in their profile page and follow other users' shared portfolios.

1.1.5. Profit/Loss Section

- 1.1.5.1. Profit/Loss section shall be private to user.
- 1.1.5.2. All users shall be able to use the Profit/Loss Section by providing the type and the amount of their investment, the exact time at when that investment made by the user and the currency which they eventually want to see the profit/loss value in terms of.

- 1.1.5.3. Trading users shall also be able to use the Profit/Loss Section using the investments that they made in the Traders Platform.

1.2. System Requirements

1.2.1. Notifications

- 1.2.1.1. The system shall notify users in accordance with alerts.

1.2.2. Predictions

- 1.2.2.1. Each trading equipment shall be available for the prediction.

1.2.3. Searching

- 1.2.3.1. The application shall support location-based search.
- 1.2.3.2. The application shall allow semantic search.
- 1.2.3.3. Searching process shall consider all the information available in user profiles and trading equipment.

1.2.4. Trading Equipment

- 1.2.4.1. The trading equipments shall be available on the site in the form of trade indices, stocks, ETFs, commodities, currencies, funds, bonds, and cryptocurrencies. (e.g., Trade Indices: S&P 500 Futures, S&P 500 and Nasdaq 100. Stocks: Deutsche Bank AG, Apple, Turkish Airlines. ETFs: SPDR S&P 500, iShares Russell 2000, United States Oil. Commodities: Gold, Silver, Brent Oil. Currencies: EUR/USD, TRY/EUR, TRY/USD. Funds: Vanguard 500 Index Admiral, PrivatFonds: Kontrolliert, Fidelity Contrafund. Bonds: U.S. 10Y, U.S. 5Y, U.S. 2Y. Cryptocurrencies: BTC/USD, ETH/USD, XRP/USD.)
- 1.2.4.2. Each trading equipment shall include the information of the previous close, percentage change with the previous close, amount change with the previous close, day's range, and moving averages.

1.2.5. Recommendation System

- 1.2.5.1. The system shall be able to recommend people and trading equipments to follow.
- 1.2.5.2. The system shall be able to recommend articles to read.

1.2.6. Economic Events

- 1.2.6.1. Economic events shall have different significance levels.
- 1.2.6.2. Each economic event shall have a numeric result which shows the actual value, the forecast value and the previous value of the related trading equipment.

- 1.2.6.3. Basic users and trading users shall be able to filter economic events by considering their significance level and country base.
- 1.2.6.4. Basic users and trading users shall be able to search economic events.

2.NON-FUNCTIONAL REQUIREMENTS

2.1. Security and Privacy

- 2.1.1. E-mails for registration shall be valid and unique for each user.
- 2.1.2. Passwords shall be 8 characters long at least and contain at least one uppercase letter, one lower case letter and a digit.
- 2.1.3. All passwords shall be encrypted using PBKDF2 and stored in the database.
- 2.1.4. Data of the users shall be protected according to Law on the Protection of Personal Data.

2.2. Performance

- 2.2.1. Response time for any request shall be 5 seconds at most.

2.3. Availability

- 2.3.1. The application shall have web and mobile (Android) versions.
- 2.3.2. The system shall be compatible with the latest main versions of Google Chrome and Android.
- 2.3.3. The language of the system shall be English but it shall also support Turkish characters.

2.4. Annotations

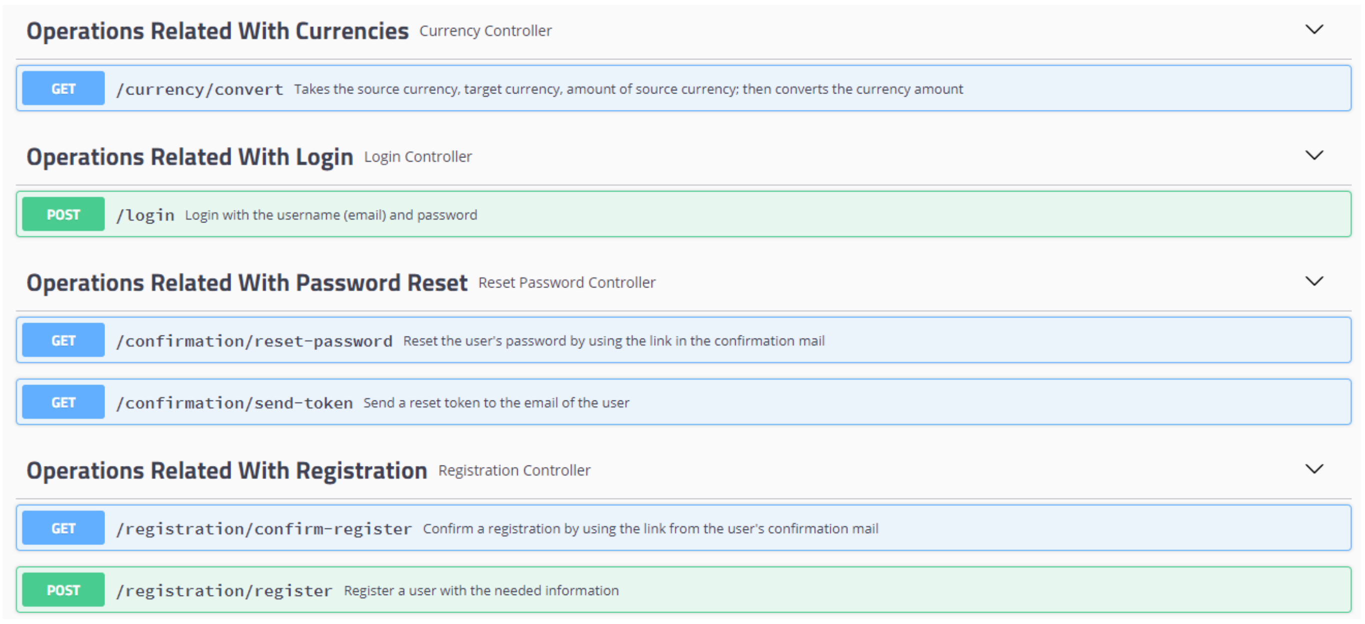
- 2.5.1. The application shall support W3C Web Annotation Data Model and follow W3C Web Annotation Protocol.

2.5. Database

- 2.5.1. System shall store user information, trade equipments, follower relations, portfolios, user histories and user preferences in the database.
- 2.5.2. Information in the database shall be able to be updated.
- 2.5.3. Database structure shall be as efficient as possible, avoiding unnecessary tables and duplicate columns.

6. API Documentations

General View of the API



Operations Related With Currencies
Currency Controller

GET `/currency/convert` Takes the source currency, target currency, amount of source currency; then converts the currency amount

Operations Related With Login
Login Controller

POST `/login` Login with the username (email) and password

Operations Related With Password Reset
Reset Password Controller

GET `/confirmation/reset-password` Reset the user's password by using the link in the confirmation mail

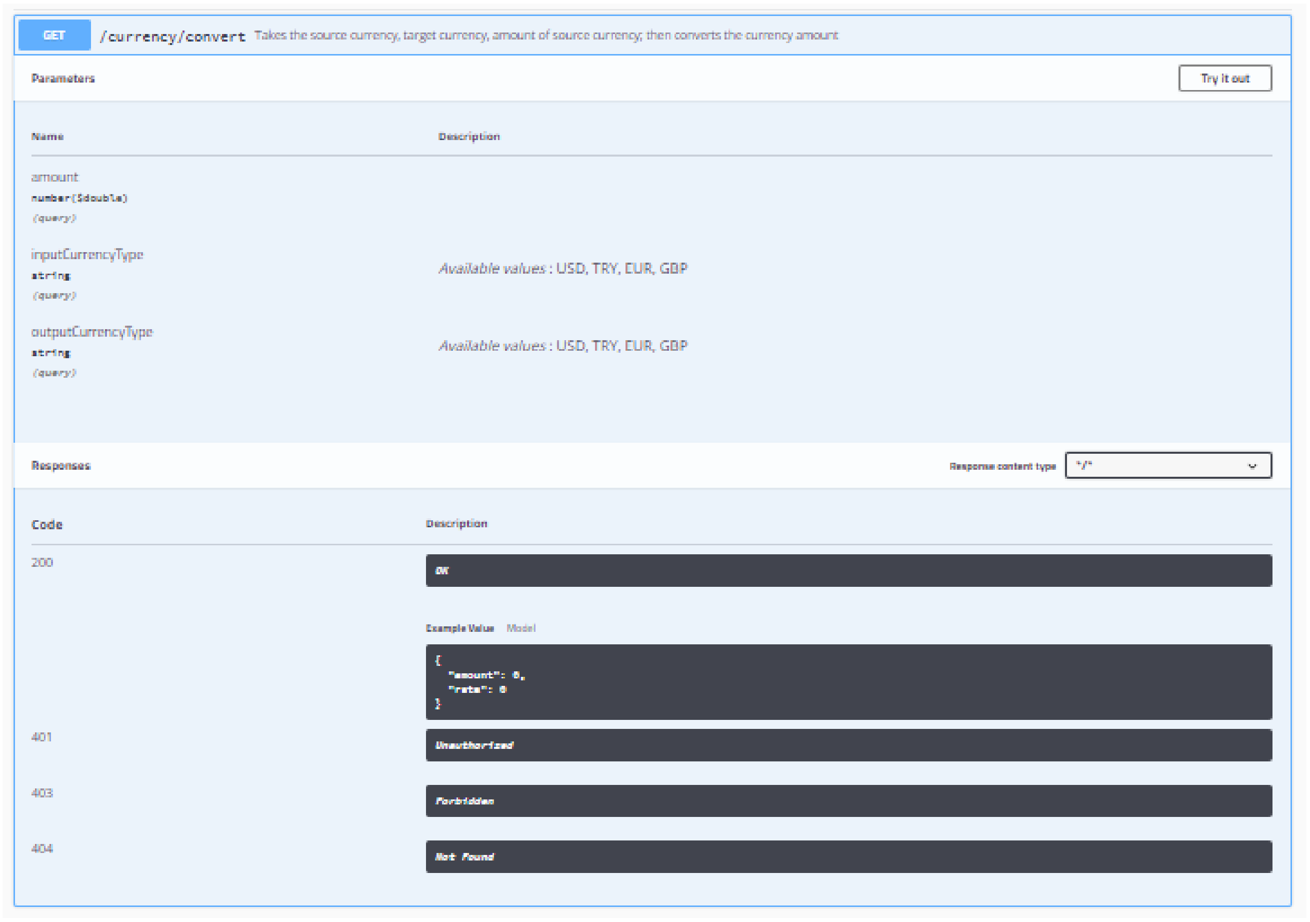
GET `/confirmation/send-token` Send a reset token to the email of the user

Operations Related With Registration
Registration Controller

GET `/registration/confirm-register` Confirm a registration by using the link from the user's confirmation mail

POST `/registration/register` Register a user with the needed information

Currency Conversion:



GET `/currency/convert` Takes the source currency, target currency, amount of source currency; then converts the currency amount

Parameters

Name	Description
amount number($double) *(query)*	
inputCurrencyType string *(query)*	*Available values* : USD, TRY, EUR, GBP
outputCurrencyType string *(query)*	*Available values* : USD, TRY, EUR, GBP

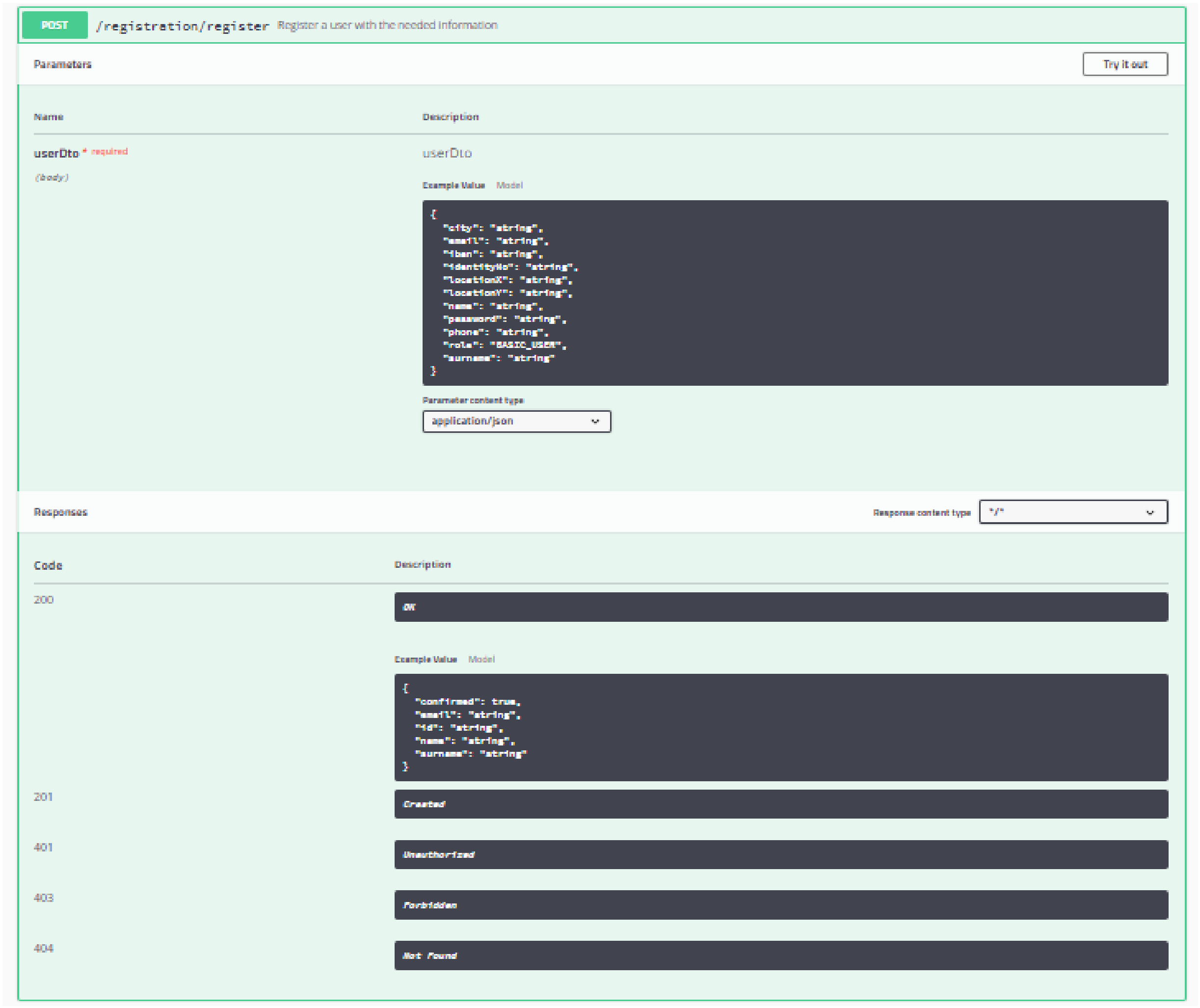
Responses

Response content type: */*

Code	Description
200	*OK* Example Value Model `{ "amount": 0, "rate": 0 }`
401	*Unauthorized*
403	*Forbidden*
404	*Not Found*

```
{
  "amount": 0,
  "rate": 0
}
```

Registration:



POST `/registration/register` Register a user with the needed information

Parameters — Try it out

Name	Description
userDto * required (body)	userDto

Example Value | Model

```
{
  "city": "string",
  "email": "string",
  "iban": "string",
  "identityNo": "string",
  "locationX": "string",
  "locationY": "string",
  "name": "string",
  "password": "string",
  "phone": "string",
  "role": "BASIC_USER",
  "surname": "string"
}
```

Parameter content type: application/json

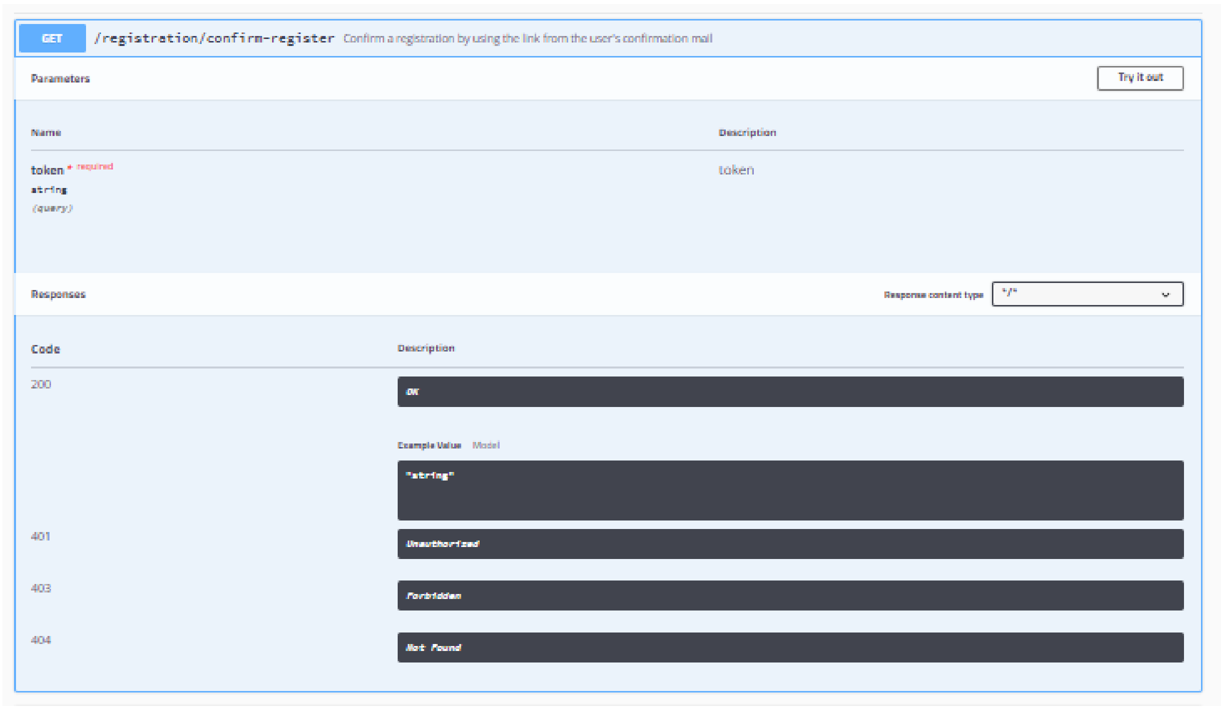
Responses — Response content type: */*

Code	Description
200	OK
201	Created
401	Unauthorized
403	Forbidden
404	Not Found

200 Example Value | Model

```
{
  "confirmed": true,
  "email": "string",
  "id": "string",
  "name": "string",
  "surname": "string"
}
```

Confirm Registration:



GET `/registration/confirm-register` Confirm a registration by using the link from the user's confirmation mail

Parameters — Try it out

Name	Description
token * required string (query)	token

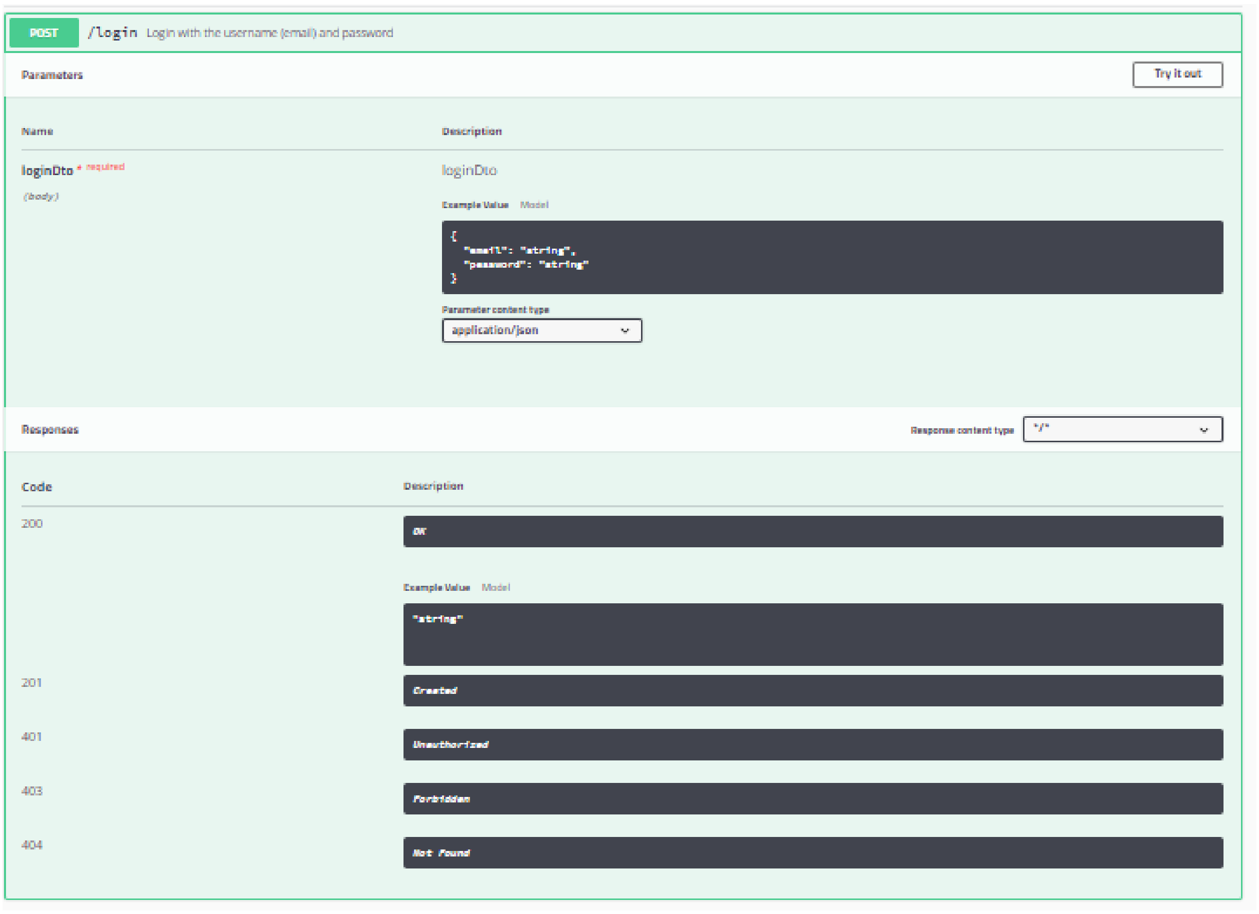
Responses — Response content type: */*

Code	Description
200	OK
401	Unauthorized
403	Forbidden
404	Not Found

200 Example Value | Model

```
"string"
```

Login:



POST `/login` Login with the username (email) and password

Parameters — Try it out

Name	Description
loginDto * required *(body)*	loginDto

Example Value | Model

```
{
  "email": "string",
  "password": "string"
}
```

Parameter content type: application/json

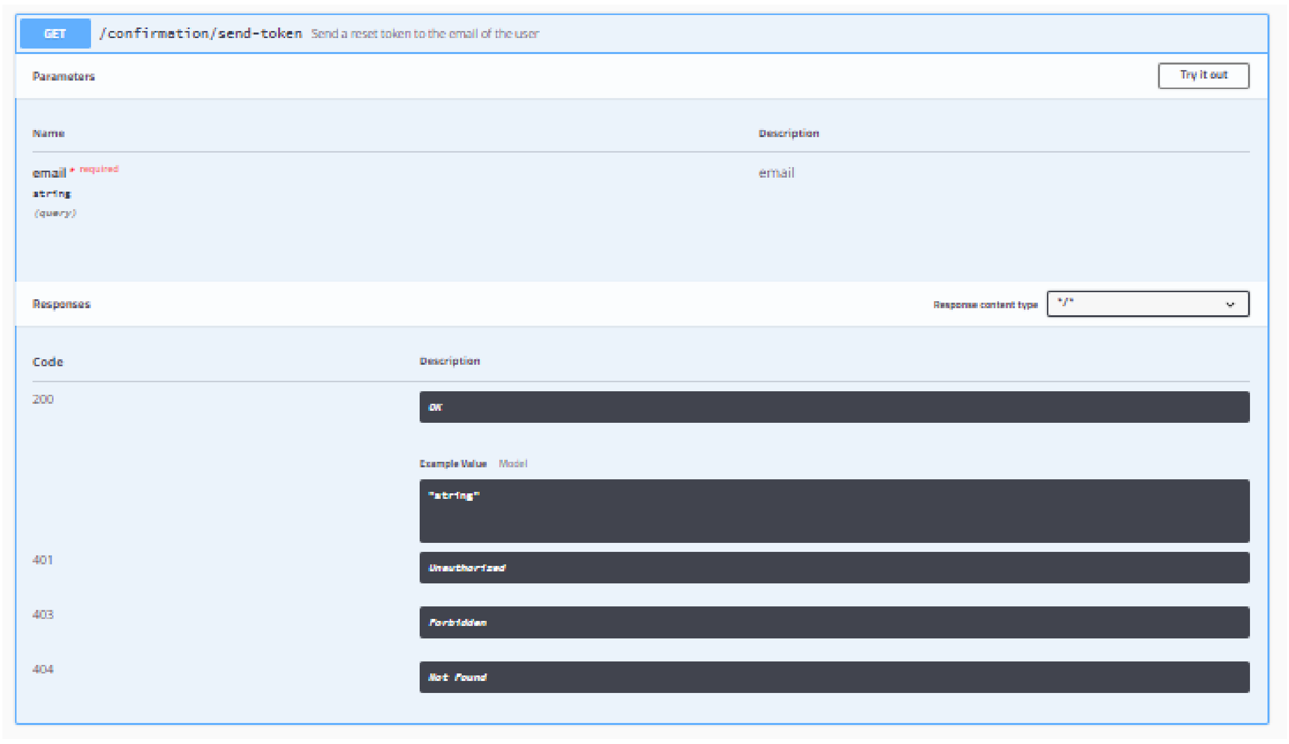
Responses — Response content type: */*

Code	Description
200	OK
201	Created
401	Unauthorized
403	Forbidden
404	Not Found

Example Value | Model (200)

```
"string"
```

Send Password Reset Email:



GET `/confirmation/send-token` Send a reset token to the email of the user

Parameters — Try it out

Name	Description
email * required string *(query)*	email

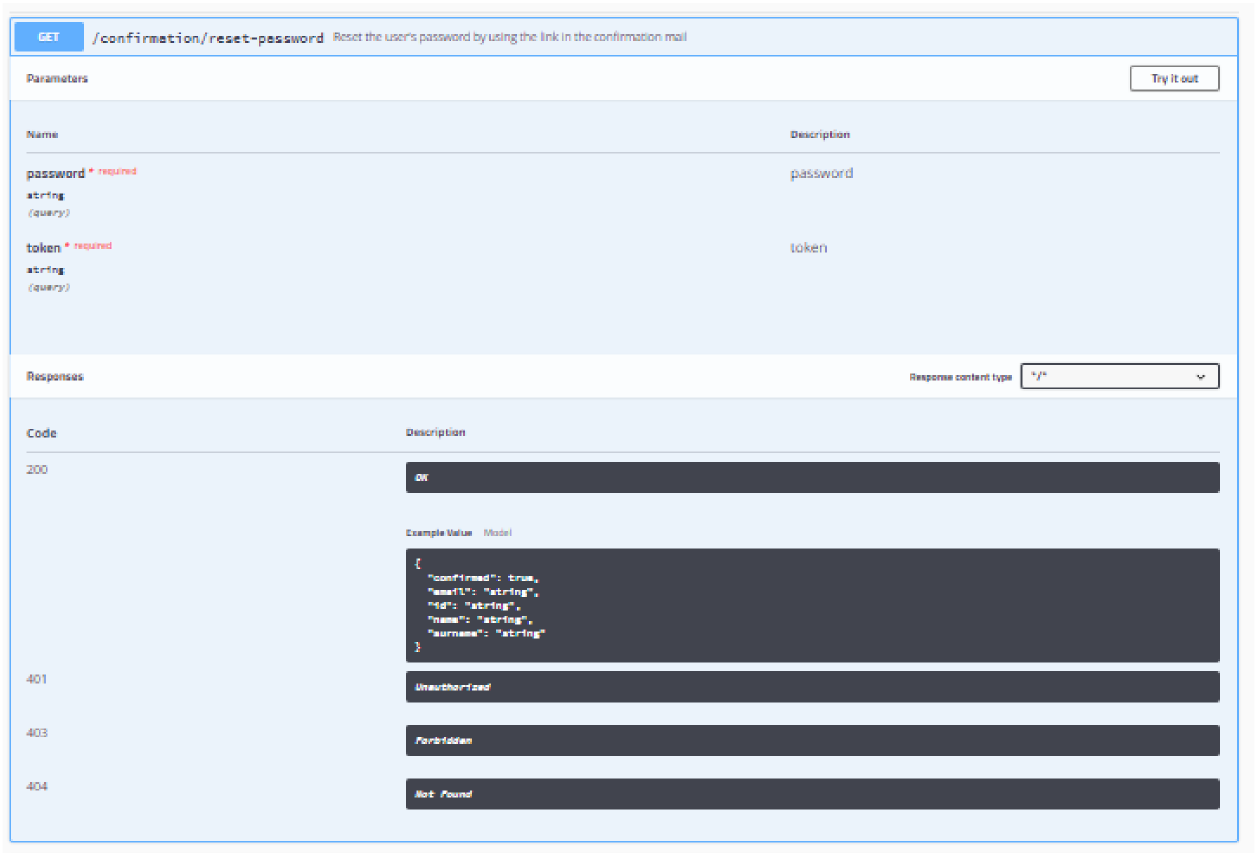
Responses — Response content type: */*

Code	Description
200	OK
401	Unauthorized
403	Forbidden
404	Not Found

Example Value | Model (200)

```
"string"
```

<u>Reset Password by Using the Confirmation Mail:</u>


GET `/confirmation/reset-password` Reset the user's password by using the link in the confirmation mail

Parameters

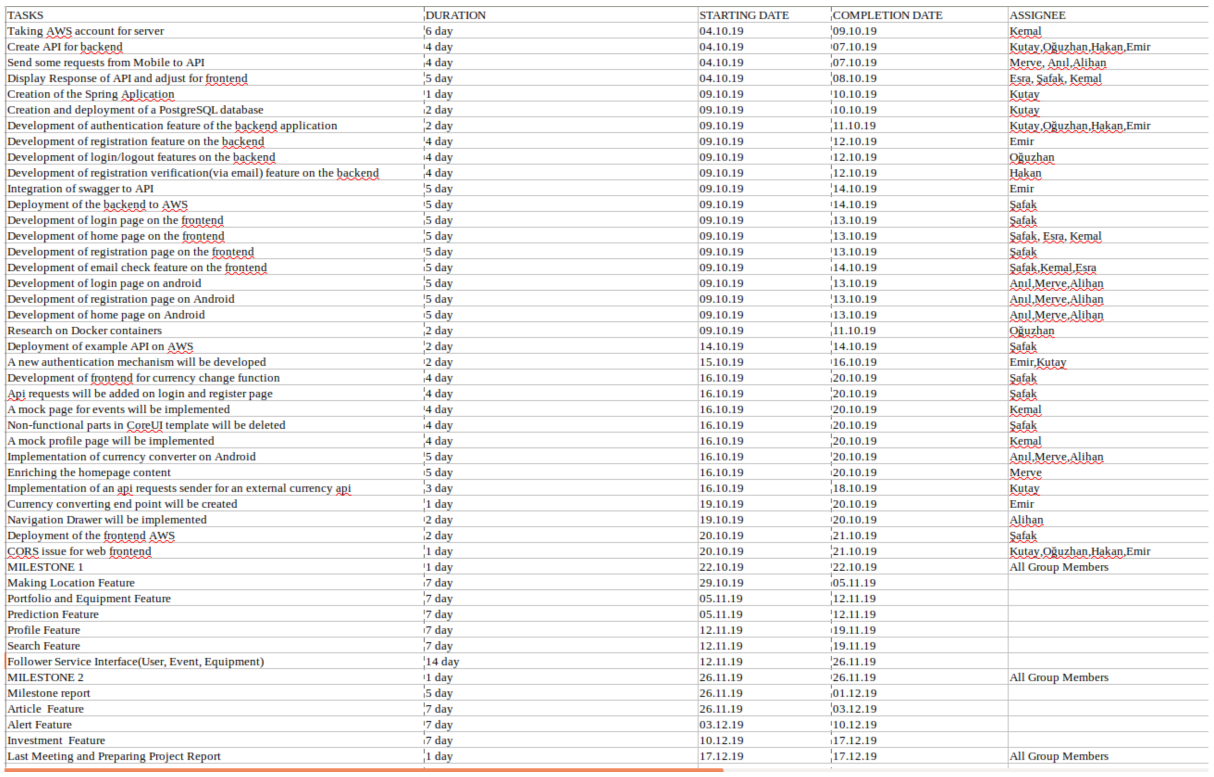
Name	Description
password * required string (query)	password
token * required string (query)	token

Responses — Response content type: */*

Code	Description
200	OK Example Value / Model
401	Unauthorized
403	Forbidden
404	Not Found

```
{
  "confirmed": true,
  "email": "string",
  "id": "string",
  "name": "string",
  "surname": "string"
}
```

7. Project Plan



TASKS	DURATION	STARTING DATE	COMPLETION DATE	ASSIGNEE
Taking AWS account for server	6 day	04.10.19	09.10.19	Kemal
Create API for backend	4 day	04.10.19	07.10.19	Kutay,Oğuzhan,Hakan,Emir
Send some requests from Mobile to API	4 day	04.10.19	07.10.19	Merve, Anıl,Alihan
Display Response of API and adjust for frontend	5 day	04.10.19	08.10.19	Esra, Şafak, Kemal
Creation of the Spring Aplication	1 day	09.10.19	10.10.19	Kutay
Creation and deployment of a PostgreSQL database	2 day	09.10.19	10.10.19	Kutay
Development of authentication feature of the backend application	2 day	09.10.19	11.10.19	Kutay,Oğuzhan,Hakan,Emir
Development of registration feature on the backend	4 day	09.10.19	12.10.19	Emir
Development of login/logout features on the backend	4 day	09.10.19	12.10.19	Oğuzhan
Development of registration verification(via email) feature on the backend	4 day	09.10.19	12.10.19	Hakan
Integration of swagger to API	5 day	09.10.19	14.10.19	Emir
Deployment of the backend to AWS	5 day	09.10.19	14.10.19	Şafak
Development of login page on the frontend	5 day	09.10.19	13.10.19	Şafak
Development of home page on the frontend	5 day	09.10.19	13.10.19	Şafak, Esra, Kemal
Development of registration page on the frontend	5 day	09.10.19	13.10.19	Şafak
Development of email check feature on the frontend	5 day	09.10.19	14.10.19	Şafak,Kemal,Esra
Development of login page on android	5 day	09.10.19	13.10.19	Anıl,Merve,Alihan
Development of registration page on Android	5 day	09.10.19	13.10.19	Anıl,Merve,Alihan
Development of home page on Android	5 day	09.10.19	13.10.19	Anıl,Merve,Alihan
Research on Docker containers	2 day	09.10.19	11.10.19	Oğuzhan
Deployment of example API on AWS	2 day	14.10.19	14.10.19	Şafak
A new authentication mechanism will be developed	2 day	15.10.19	16.10.19	Emir,Kutay
Development of frontend for currency change function	4 day	16.10.19	20.10.19	Şafak
Api requests will be added on login and register page	4 day	16.10.19	20.10.19	Şafak
A mock page for events will be implemented	4 day	16.10.19	20.10.19	Kemal
Non-functional parts in CoreUI template will be deleted	4 day	16.10.19	20.10.19	Şafak
A mock profile page will be implemented	4 day	16.10.19	20.10.19	Kemal
Implementation of currency converter on Android	5 day	16.10.19	20.10.19	Anıl,Merve,Alihan
Enriching the homepage content	5 day	16.10.19	20.10.19	Merve
Implementation of an api requests sender for an external currency api	3 day	16.10.19	18.10.19	Kutay
Currency converting end point will be created	1 day	19.10.19	20.10.19	Emir
Navigation Drawer will be implemented	2 day	19.10.19	20.10.19	Alihan
Deployment of the frontend AWS	2 day	20.10.19	21.10.19	Şafak
CORS issue for web frontend	1 day	20.10.19	21.10.19	Kutay,Oğuzhan,Hakan,Emir
MILESTONE 1	1 day	22.10.19	22.10.19	All Group Members
Making Location Feature	7 day	29.10.19	05.11.19	
Portfolio and Equipment Feature	7 day	05.11.19	12.11.19	
Prediction Feature	7 day	05.11.19	12.11.19	
Profile Feature	7 day	12.11.19	19.11.19	
Search Feature	7 day	12.11.19	19.11.19	
Follower Service Interface(User, Event, Equipment)	14 day	12.11.19	26.11.19	
MILESTONE 2	1 day	26.11.19	26.11.19	All Group Members
Milestone report	5 day	26.11.19	01.12.19	
Article Feature	7 day	26.11.19	03.12.19	
Alert Feature	7 day	03.12.19	10.12.19	
Investment Feature	7 day	10.12.19	17.12.19	
Last Meeting and Preparing Project Report	1 day	17.12.19	17.12.19	All Group Members

8. Presented User Scenarios

Mobile:

Mustafa Smith is a dedicated economy student at Boğaziçi University who wants to get to know what's happening in the world of economy. Therefore, he decides to download the beautiful app called eyeTrade to see if he can find what he wants there. He registers as a basic user since he doesn't want to provide his bank account information. After providing necessary information about the registration and login, he finds himself in the homepage which shows some up to date articles regarding economy. Using the navigation bar on the left, he switches to page called Events where he sees the upcoming events related to his interests. Finally, using the same navigation bar again, he opens the currency converter and converts his 1000 Turkish Lira to see if he can manage to have a vacation in Europe during the summer. Unfortunately, he realizes that he only has 156 Euro, so he will need much more money to save.

Web:

Hans Graham is a German journalist who works for the famous newspaper "die Zeit" in Germany. Besides his job, he likes economy and investment, and while he is trying to find a good website for his purposes, he finds eyeTrade. He registers as a trader, and logs in immediately after registration. On the homepage, he sees some events regarding economy, and especially one of them catches his attraction: Turkish government will reveal economic reforms. Since he is extremely jealous of Turkey - like every European - and its economic power, he decides to attend that event. He thinks that he may write a bad article about Turkish economy after the event to satisfy his jealous feelings. After checking the date of the event, he looks for some accommodation options in Turkey from another website, but he wants to know how much money in Turkish Lira he has. That's why he opens the currency converter in eyeTrade and converts his 1000 Euro into Turkish Lira. He sees that he has 6400 Turkish Lira, and once again, he feels impressed with the power of Turkish economy.

9. Branches and Pull Requests

We have put the code for frontend, backend and mobile in different folders. For each of these, responsible teams have their own branches. Sometimes for swift bug fixes or to meet deadlines special branches may be created. However, in general features are developed in separate branches and those are merged after peer review. For instance, backend features are developed in different branches and merged into backend branch. Again, after peer review branches are merged into master.

10. Evaluation of Tools

Github: We used Github to locate our code base. It was helpful for reviewing our codes and commits. Also, it was helped us to manage our project by using issue system.

Git: We used Git as the version control system. It is a really useful tool to manage the different branches of the different teammates.

AWS: We used AWS as the deployment environment; used EC2 to deploy backend and frontend applications and PostgreSQL RDS for the database server. It was a proper choice because of the free credits of AWS Educate accounts.

Java: We used Java as the programming language of the backend application. It was a good choice because of it's powerful and mature libraries as well as it's nice object oriented structure.

Spring Boot: We used Spring Boot as the backend framework. It is a very useful tool to develop maintainable and scalable server-side application with it's proper architecture and powerfull annotations.

Intelij: We used Intelij as the IDE for the backend development. It is a user-friendly IDE for Java applications because of it's auto-complete, auto-import, auto-refactoring, and git integration features.

Swagger: We used Swagger to document our API. It is a tool which is super easy to integrate to the api. Also, it is very user friendly to be used by the client-side application developers.

Kotlin: We used Kotlin as the programming language for the Android development. We had two options for Android development: Java and Kotlin. We choose Kotlin because it is more concise, safer, and bug-free; also it has a better support for functional programming features.

Retrofit: We used Retrofit to send http requests from our Android application. With Retrofit, it is very easy to send requests to the API's as well as parsing JSON responses.

Android Studio: We used Android Studio as the IDE for Android development. Android Studio is a very beneficial tool with it's Android Emulator and it's plugins.

Vue.js: We used Vue as the frontend library. It offers Virtual DOM, component base ui development opportunity, and variety of JavaScript libraries which are very powerful for frontend development.

Visual Studio Code: We used Visual Studio Code as the IDE for frontend development. It is one of the best choices with it's auto-complete, auto-fixing features as well as powershell integration.

11. Managing Project

Project management is an essential factor for a software team to be successful in the long term. As a group of ten people with different backgrounds and experiences with some additional members, we manage the development process during this milestone smoothly. The division of labour in all sub teams including backend, android and frontend was fair and all members had chance to contribute in the development. In terms of communication, weekly meetings were essential to determine the general strategy for each week and additional meetings of all sub teams were important for the division of tasks which are planned in the general meeting into sub tasks for each teams. We also used the github environment in an efficient way in terms of tracking the issues, communication between various teams by the help of these issues and creating branches for each completed development and making pull requests for them. This is a useful way to commit changes in a medium and large scaled projects. Another important thing which makes easy to progress as a team was the interactive communication between the backend team and frontend/android teams. Throughout implementation of some important functionalities such as login and sign up, android and frontend teams have encountered some errors that are stem from backend implementation. However these issues have been resolved quickly by an efficient communication.

In addition to these, as a team we have decided to work harder until the next milestone since we would like to implement most of the core functionalities of our project until the next milestone. In order to accomplish this goal, we are going to follow the deadlines more strictly and all members will take more responsibility until the next milestone.