CMPE451

GROUP 11

Milestone 2 Report

1- Executive Summary

As we have given our Milestone 2 presentation to the customer, we are going to deliver the details of our progress since the Milestone 1 and current situation in the project with this Milestone 2 report.

1.1 Introduction

To start, our project is a social platform for people who consider themselves as traders. People can sign up and log in the platform to take advantage of more advanced features such as buying and selling trading equipment. However, users can use searching functionality without login and signup. Guest users also can view articles, events, trading equipment, and comments.

Fundamentally, there are two types of users: basic and trader. Basic users can create, comment and rank on the articles. In addition, they can create and share portfolios. Not only that, they can add trading equipment to the created portfolios. They can also chase economic events and filter them by country and significance levels. Our platform has more advanced features such as setting an alert, making a prediction about trading equipment and following trading equipment.

Trader users have all the functionality that basic users have and they have extra features such as trading indices, stocks, ETFs, commodities, currencies, funds, bonds, and cryptocurrencies. They have an extra ”My Investments” section that enables them to set buy and sell order. Another cool feature of the platform is that people can follow other people. In a nutshell, our platform creates a connection between people who are interested in trading and trading world.

1.2 Project Status

We have created the basis and the social part of our projects. Backend services are built, initial design of front-end and mobile services is implemented.

Based on our initial meetings to construct a general view on the progress of our project, a big portion of deliverables planned for Milestone 1 are delivered to the customer. **Sign-up and login** functionalities are added. Users can sign up and **verify** their **emails** in the registration process.

After a successful login, users can create **portfolios,** add **trading equipment** to their portfolio and **follow the progress of the equipment** via a handy front-end system. Same functionalities are added to the **Mobile** app.

Users can also check **Events** section to follow **impactful news and events.** Besides that, they can also **write, read and rate** articles both by other users and themselves.

In milestone 2, we further improved our **trading equipments** and **articles** section as well as implemented **Google sign-in/sign-up** and user **profile**. We generally focused on the social requirements of our platform to help users interact with each other.

All users now have a profile page where others can see their bio, articles and portfolios. Users can change their privacy settings to set their profile to **private or public**. They can **send follow requests** to private users or directly **follow** other users. Users receive a notification when there is a follow request. Followers and followed users are also visible in the profile page. Users can unfollow a followed user or cancel a follow request.

In the articles section, we added **comments and voting**. User can **add, edit and delete a comment** to an article. Also, can vote the article by upvoting or downvoting.

In the first milestone, we added stocks to our system and showed their graphs and price in terms of dollars. In the second milestone, we added **currency** parity with their graphs and values. Users can search for parities in trading equipment page and add them to their portfolio with our new **search** functionality in the portfolio page. Besides those new features, there is a comment section in trading equipment. Comments can be added, deleted and edited. Also, one can make **predictions** about the trading equipment as increase or decrease and those predictions affect the prediction rate of the user. The current prediction and the total number of votes can be viewed in the related trading equipment's page.

Finally, Google **sign-up and login** is implemented in Mobile application and home page of our website is redesigned. In home page, top parities, articles and events can be seen.

1.3 Changes planned

The first thing we want to do is polish the platform we have right now. For example, we now have caching of currency rates but this is not the case for stocks. We have comments for currencies but not for stocks and so on.

Then, we are planning to implement searching for users, articles and trading equipments. Also, there are missing pieces on the trading part of the platform such as the ability to buy(done in backend), sell or trade equipments. Auto buy/sell orders for equipments also planned to be added. Notification and alert system is not done yet so some work is needed there too. And finally, we will do our best to add annotation support to our platform.

2- List and Status of Deliverables

Task	Status
Review of project requirements	Done
Project plan update	Done
Register	Done
Register/Login via Gmail	Mobile : Done Frontend : In progress
Login / Logout	Done
Email verification	Done
Google Maps API	Done
Article Page	Done
Profile Page	In progress
Portfolio	In progress : Adding stocks and currencies is done
Trading Equipments	In progress : Stocks and currencies are added

Events	Done
User Follow	Done
Notification System	In progress

3- Evaluation of the Status of Deliverables

Backend Structure

In the backend, we were able to implement lots of social and trading related endpoints such as comments, votes, predictions, profiles, profile privacy, ability to follow other users enhanced portfolios, investments and so on. We are happy to deliver these. For the final milestone, we need to polish some of the existing parts and implement crucial parts of the system such as searching, buy-sell orders, alerts and annotations.

Frontend

In the frontend part there are various developments since the Milestone 1. These are the changed pages with the new features on them :

Homepage :

In Milestone 1 homepage existed but it was empty. Now there are three components on this page. There is a section of the newest 5 Articles and Events can be previewed. Users can go to the Articles and Events pages by using these section to see the rest of the articles and events. There are also two other sections that user can see stocks and currencies. History over the span of previous 1-2 weeks for top currencies and stocks are shown in charts. These two sections are not fully implemented currently.

Login/Register Page :

There is a minor change in these pages. In Milestone 1 there were two different links on the navigation bar which direct Login and Register pages but later we prefered to put just one link for Login page and there is a small link for non-registered users below the login form which directs to the register page. We chose it this way to keep navigation bar simple.

Articles Page :

Articles page is the page where we preview the articles in the database. We added a button for registered users to add articles. This button directs user to another page. Voting functionality is added with up- and downvoting buttons and total vote count.

Add Article Page :

This page is a whole new page, it was not implemented in Milestone 1. In this page there are two text areas, one for the article title and another for the article body.

Article Page :

In Milestone 1 this page was where we can view the whole article with its title, author and body. It had no functionality other than viewing text from the database. Now by clicking the author name user is directed to the user author profile. Registered users can like or dislike the article. The comments for the article are listed below the article. If user is logged he/she sees a trash icon to delete the comment and also can add a comment by using comment editor.

Currencies Page :

In Milestone 1 there were no features related to currencies. We added currencies to the system and users can view all types USD/Currency parities on this page. Also each parity is shown with a graph covering one week period.

Other Users Profile Page :

For now users can visit other users' profile pages by clicking on their names on articles. On other user's profile one can see the articles belonging to that user.

Android

In Android, we successfully completed all the deliverables. The user can see their and others profile pages, in profile pages they can the followers, follows, articles and portfolios.They can see the followers, follows and articles if the profile is public.We add comments to articles and trading equipments.We also add voting for the articles.Now, users can add currencies to portfolios.We also add search functionality since it was one of the feedback from the Milestone 1.For the final milestone, we need some small fixes such as refreshing profile pages after accepting or denying request.

Coding work done by each member

Alkım Ece Toprak	**Milestone 1:** Implemented the backend part for the articles including routes and services. Added some functions' documentation to the Swagger json. **Milestone 2:** ımplemented the backend part of investments feature of our project. This implementation includes route, service, model and documentation scripts. I also added the backend api of portfolio to add currencies.
Alperen Değirmenci	I am in the mobile team. For the milestone 1, I have added the following: Implemented the Login and Sign-Up pages of the mobile app. Created the design and the functionalities. For the Milestone 2: Implemented the connections to the profile page so that users can reach other users profile by clicking on the names of users on various pages.
Aysu Sayın	I am in mobile team so all the work done by me is about android application. Milestone 1:Implemented Events and article pages of the mobile app. On the article page, all articles are shown as a custom list I designed. There is a button to create article which directs user to "Add article" page. Also, I implemented the page where user can see the contents of the article. There is also another page for viewing events. Milestone 2: Designed and implemented comments and voting for articles. Plus, designed and implemented trading equipments main page with the search functionality. Improved trading equipment detail page by adding comments and predictions.
Burak Enes Çakıcı	Implemented Registration Form fields (including choosing location) checker. Added a pop-up in case of bad entries. Added location attributes to user type. Was part of the Front End team and contributed team mates in creating general design and login, signup and profile pages.

	Milestone 2: Worked on Front-end Article page and home page: Added vote up and vote down functionality of articles communicating with backend to update vote count, contributed to the home page design choice and added a chart showing the history of the top stocks.
Cumhur Kılıç	**Milestone 1:** Implemented event part of the backend. Added data model, route and service for event. Added event functions' documentation to the Swagger json. **Milestone 2:** Added data models for comments on stock and currency. Changed routes and services on stock and currency for comments endpoints.
Doğa Yüksel	Booted up the frontend project with React, implemented Sign-up form and modals for successful/failed sign-ups/logins, implemented Navbar navigation and Navbar changes according to login state. Implemented global logged in state and integrated changes according to state into other pages (can't access profile if not logged in etc.) Implemented Portfolio lists and Stock lists (view & add items). Implemented Articles page and individual article pages and previews. Helped with overall design choices. Milestone 2: I implemented profile pages for other users and follow user feature for frontend. I also implemented notifications for follow request. I have done a lot of small fixes, search for trading equipment and fixed the geolocation on profile pages.
Hasan Yaman	Implemented Portfolio, Portfolio Detail and Trading Equipment Detail pages of the mobile app.I also add creating portfolio functionality and adding trading equipments to portfolio functionality. I created Profile page.I created the design for the pages. Milestone 2: I implemented Google login and sign up page in Android.I also implemented profile page.I added adding currency to portfolio functionality.
İbrahim Kamacı	Milestone 1: Implemented article data model and wrote database initializer for the push

	mock articles to the database. Implemented 'Stock' data model as a trading equipment. Used third party API to fetch stock data for real time experience and added some stocks to database. Stock service and proper endpoints were implemented. Implemented 'Portfolio' data model. Corresponding services and endpoints were implemented. All corresponding endpoint documentations were constructed by using Swagger API Documentation. Deployed frontend application on AWS. Milestone 2 : Modified user data model to implement social networking features.Implemented user follow,unfollow,accept follow request and decline follow request methods. Implemented profile related functions and added public private profile settings to user data model. Implemented profile endpoints to ensure client-side implementing social networking and profile page. All corresponding endpoint documentations were constructed by using Swagger API Documentation. Deployed frontend application on AWS.
Metin Dumandağ	**Milestone 1:** Constructed Swagger documentation mechanism for the backend. Prepared the database on Mongo Atlas. Implemented and added Swagger documentation for the login, signup, email verification, user and lost password endpoints. Added related models and services for these endpoints. Added authentication check to the article endpoint. Deployed backend service on AWS. **Milestone 2:** Implemented and added Swagger documentation for Google sign up and sign in, adding, editing and deleting comments to articles, up-voting, down voting and clear vote for articles, current, intraday, weekly, monthly and last 100 days rates for currencies and prediction for currencies. Also, implemented cron jobs to periodically fetch currency rates from Alpha Vantage and check

	the predictions made by users and update the related fields.
Semih Akgül	**Milestone 1 :** Implemented profile card and profile page in frontend. Implemented events and event/id pages. Designed the logo of the app and general view of the website. Added pin feature to the map. Added ionicon open source icon set to the project. **Milestone 2 :** Up to the Milestone 2 I implemented : ● USD/EUR Section on Homepage ● Articles/Events Section on Homepage ● "Article Add" button on /articles page ● /AddArticle page ● Viewing comments below an article ● Comment editor below an article to add comments ● Deleting comments users own comments ● On /currencies page ○ Listing currencies ○ Showing graphs of currencies ● Added icons to the navbar, comments, articles etc. to enhance the visuality ● Rating of the events were random. Now they are drawn from backend

4- Requirements

(Until the end of Requirements section, numbering system discards no. of this section '5')

1. Functional Requirements

- **1.1 User Requirements**
  - **1.1.1. Common User Requirements**
    - **1.1.1.1. Sign Up Requirements**
      - 1.1.1.1.1. Guest users shall be able to sign up to the platform providing necessary information, i.e. Name, surname, e-mail address and location. Also IBAN and Turkish Identification Number(TC) to sign in as trading user.
      - 1.1.1.1.2. Guest users shall be able to provide their location information via Google Maps.

        - 1.1.1.1.3. After guest users completing the processes 1.1.1.1.1 and 1.1.1.1.2 they shall be able to validate their e-mail addresses by visiting the link that is sent to their e-mail address.
        - 1.1.1.1.4. Guest users shall be able to sign up to the platform using their Google account. Location shall be given by the user after signing up. Also IBAN and Turkish Identification Number(TC) to sign in as trading user.
        - 1.1.1.1.5. Traders are required to provide an ID number and an IBAN while signing up.
    - **1.1.1.2. Sign In Requirements**
        - 1.1.1.2.1. Registered users shall be able to sign in with their e-mail address and password.
        - 1.1.1.2.2. Registered users should be able to sign in with their Google account.
    - **1.1.1.3. Profile Requirements**
        - 1.1.1.3.1. Registered users profiles shall be either public or private to the other registered users.
        - 1.1.1.3.2. Registered users shall follow the registered users having private profiles to see private contents in a profile.
        - 1.1.1.3.3. Name, surname, location, prediction success rate and profile biography shall be public to every registered user and guest.
    - **1.1.1.4. Portfolio Requirements**
        - 1.1.1.4.1. Users shall have at least one portfolio.
        - 1.1.1.4.2. Users shall be able to create portfolios.
        - 1.1.1.4.3. Users shall be able to rename portfolios.
        - 1.1.1.4.4. Users shall be able to delete portfolios.
        - 1.1.1.4.5. Users shall have private and public portfolios and be able to share their public portfolio in their profile page and follow other shared portfolios.
        - 1.1.1.4.6. Users shall be able to add trading equipments to their portfolio.
        - 1.1.1.4.7. Users shall be able to remove trading equipments from their portfolio.
    - **1.1.1.5. User Specific Events Requirements**
        - 1.1.1.5.1. Users shall be able to have an Events section. In this section, users can chase economic events with different significance levels from 1 to 3.

        - 1.1.1.5.2. Users should be able to filter economic events by considering their significance level, date and country base.
    - **1.1.1.6. Profit/Loss Requirements**
        - 1.1.1.6.1. Users shall have a Profit/Loss section which is private to the user.
    - **1.1.1.7. Article Requirements**
        - 1.1.1.7.1. Registered users shall be able to share their ideas as articles.
        - 1.1.1.7.2. Registered users shall be able to comment on the articles.
        - 1.1.1.7.3. Registered users shall be able to like or dislike the articles.
        - 1.1.1.7.4. All articles regardless of author's privacy shall be able to be seen by other users.
        - 1.1.1.7.5. All comments regardless of author's privacy shall be able to be seen by other users.
        - 1.1.1.7.6. All article scorings regardless of author's privacy shall be able to be seen by other users.
        - 1.1.1.7.7. Articles can only contain text.
    - **1.1.1.8. Alert Requirements**
        - 1.1.1.8.1. Registered users shall be able to set alerts for trading equipments.
        - 1.1.1.8.2. Registered users shall be able to set alerts for events.
    - **1.1.1.9. Prediction Requirements**
        - 1.1.1.9.1. Predictions shall be in the form of multiple choice question with two options: value will decrease and increase.
        - 1.1.1.9.2. Registered users shall be able to make predictions about the trading equipments.
        - 1.1.1.9.3. Registered user's prediction success rate for all trading equipments shall be available on their profile.
    - **1.1.1.10. Searching Requirements**
        - 1.1.1.10.1. Registered users shall be able to search for other users.
        - 1.1.1.10.2. Registered users shall be able to search for other users by location.
        - 1.1.1.10.3. Registered users shall be able to search for trading equipments.
        - 1.1.1.10.4. Registered users shall be able to search for events.
    - **1.1.1.11. Social Requirements**

- 1.1.1.11.1. Registered users shall be able send follow requests to other users.
- 1.1.1.11.2. Registered users with private profiles shall be able accept or decline follow requests.
- 1.1.1.11.3. Registered users shall be able follow trading equipments.
- 1.1.1.11.4. Registered users shall be able comment on trading equipments.

    - **1.1.2. Basic User Requirements**
        - 1.1.2.1. Basic users shall be able to see their profit/loss amount in terms of the currency they choose by manually entering their investments.
    - **1.1.3. Trader User Requirements**
        - 1.1.3.1. Traders shall be able to see their profit/loss amount in terms of the currency they choose by both manually entering their investments and using the investments they made in the Traders Platform.
        - 1.1.3.2. Traders shall be able to trade indices, stocks, ETFs, commodities, currencies, funds, bonds, and cryptocurrencies.
        - 1.1.3.3. Traders shall have a My Investments section.
    - **1.1.4. Guest Requirements**
        - 1.1.4.1. Guests shall be able to view prices of trading equipments, user comments,articles and events.
        - 1.1.4.1. Guests shall be able to use searching functionality.
- **1.2. System Requirements**
    - **1.2.1. Searching Requirements**
        - 1.2.1.1. System shall consider all the information available in user profiles, trading equipments, events and articles while performing the search.
        - 1.2.1.1. System shall allow semantic search. System shall also display the users, trading equipments and articles that have semantic relationship with the search tags in the search results.
    - **1.2.2. Events Requirements**
        - 1.2.2.1. Events shall have different significance levels from 0 stars to 3 stars.
        - 1.2.2.2. Events shall be filtered by date, significance levels and country base.
    - **1.2.3. Investments Requirements**
        - 1.2.3.1. Investments shall be made with any trading equipment.
        - 1.2.3.2. Investments section shall be private to user.

        - 1.2.3.3. Investments should contain some functionality such as buy/sell order for a specified rate, set a stop/loss limit.
    - **1.2.4. Trading Equipment Requirements**
        - **1.2.4.1 Trading Equipment Examples**
            - 1.2.4.1.1. Indicies
                - 1.2.4.1.1.1. S&P 500
                - 1.2.4.1.1.2. Dow 30
                - 1.2.4.1.1.3. Nasdaq 100
            - 1.2.4.1.2. Stocks
                - 1.2.4.1.2.1. General Electric Company (GE)
                - 1.2.4.1.2.2. Microsoft Corporation (MSFT)
                - 1.2.4.1.2.3. HP Inc. (HPQ)
            - 1.2.4.1.3. ETFs
                - 1.2.4.1.3.1. SPDR S&P 500 ETF (SPY)
                - 1.2.4.1.3.2. Invesco QQQ Trust (QQQ)
                - 1.2.4.1.3.3. iShares Gold Trust (IAU)
            - 1.2.4.1.4. Commodities
                - 1.2.4.1.4.1. Gold
                - 1.2.4.1.4.2. Silver
                - 1.2.4.1.4.3. Crude Oil
            - 1.2.4.1.5. Currencies
                - 1.2.4.1.5.1. USD
                - 1.2.4.1.5.2. EUR
                - 1.2.4.1.5.3. TRY
            - 1.2.4.1.6. Funds
                - 1.2.4.1.6.1. Pimco Total Return (PTTAX)
                - 1.2.4.1.6.2. Vanguard Total Stock Market Index Fund (VTSMX)
                - 1.2.4.1.6.3. American Funds Growth Fund of America (AGTHX)
            - 1.2.4.1.7. Bonds
                - 1.2.4.1.7.1. Treasury Bonds
                - 1.2.4.1.7.2. Municipal Bonds
                - 1.2.4.1.7.3. Corporate Bonds
            - 1.2.4.1.8. Cryptocurrencies
                - 1.2.4.1.8.1. Bitcoin
                - 1.2.4.1.8.2. Ethereum
                - 1.2.4.1.8.3. Monero
        - **1.2.4.2. Trading Equipment Functionality**

            - 1.2.4.2.1. Trade equipments should have some functionality such as the previous close, the percentage change with the previous close, amount change with the previous close, day's range, and moving averages.
    - **1.2.5. Alert Requirements**
        - 1.2.5.1. System shall be able to notify the users in accordance to their alerts.
    - **1.2.6. Recommendation Requirements**
        - 1.2.6.1. System should recommend articles or trading equipments to the users based on their reading and search histories.

2. Non-functional Requirements

- **2.1. Accessibility and Availability**
    - 2.1.1. The system should work on web browsers that supports Javascript ES6.
        - 2.1.1.1. Chrome 58 and above
        - 2.1.1.2. Firefox 54 and above
        - 2.1.1.3. Edge 14 and above
        - 2.1.1.4. Safari 10 and above
    - 2.1.2. The system should be available %99 of the time.
    - 2.1.3. The application shall be provided for both web browsers and Android mobile devices.
- **2.2. Performance and Response Time**
    - 2.2.1. The system shall work with up to a thousand HTTPS requests without crashing.
    - 2.2.2. The system shall respond to all search requests under 5 seconds.
- **2.3. Security**
    - 2.3.1. The system should use https, so that the traffic between browser (or app) and the web server is encrypted.
    - 2.3.2. System should deny requests if excessive amount of requests has been made.
        - 2.3.2.1. Rate-Limit protocol will be provided by our hosting company.
        - 2.3.2.2. System accepts HTTP requests up to 1000.
    - 2.3.3. All passwords stored in the database must be hashed and encrypted by npm bcryptjs package.
    - 2.3.4. The system shall check that an authorization bearer token is present and correctly formed
- **2.4. Annotations**
    - 2.4.1. The platform shall support W3C Web Annotation Data Model

    - 2.4.2. W3C Web Annotation Protocol shall be followed to store annotations.
- **2.5. Portability**
    - 2.5.1. The application shall be compatible with devices with Android version greater than 7.0 and web browsers that supports Javascript ES6.
- **2.6. Capacity**
    - 2.6.1. The system must meet the agreed capacity requirements.

5- API Documentation

URL for the API documentation is as following:

http://ec2-18-197-152-183.eu-central-1.compute.amazonaws.com:3000/api-docs/

Article

Up

`GET /article`

Article get all endpoint. (**articleGet**)

Return type

array[Article]

Example data

Content-Type: application/json


```
[ {
  "date" : "2019-11-14T21:52:05.381Z",
  "author" : {
    "surname" : "Doe",
    "name" : "John"
  },
  "_id" : "5da07611e407903a5d01b265",
  "voteCount" : 72.0,
  "userVote" : 1.0,
  "title" : "Low-Cost Investing Can’t Get Any Lower Than Free",
  "body" : "The central bank cut rates for the first time since the Great Recession in late July...",
  "authorId" : "5da07611e407903a5d01b265"
}, {
  "date" : "2019-11-14T21:52:05.381Z",
  "author" : {
    "surname" : "Doe",
    "name" : "John"
  },
  "_id" : "5da07611e407903a5d01b265",
  "voteCount" : 72.0,
  "userVote" : 1.0,
  "title" : "Low-Cost Investing Can’t Get Any Lower Than Free",
  "body" : "The central bank cut rates for the first time since the Great Recession in late July...",
  "authorId" : "5da07611e407903a5d01b265"
} ]
```

Produces

This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- `application/json`

Response
s
200
OK. All articles returned succesfully.

Up `POST /article/{id}/clear-vote`

Endpoint to clear vote on an article. (**articleIdClearVotePost**)

Path
parame
ters

id (required)
Path Parameter — Id of the article.

Produces

This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- `application/json`

Response
s
200 400
OK. Vote for the article successfully cleared.
Validation error. Either the article with the given id, the user with the given token or the vote not found. Check causes field for diagnostics.
401
Authorization token was missing,
invalid or expired. **500**
Internal error occurred. Check the cause field for the diagnostics.

`DELETE /article/{id}/comment/{commentId}` Up

Endpoint that deletes the selected comment. (**articleIdCommentCommentIdDelete**)

Path
parame
ters

id (required)
Path Parameter — Id of the

article **commentId (required)**

Path Parameter — Id of the comment

Response
s
200
OK.
Comment
deleted.
400
Article, user or the
comment not found. **500**

Internal error occurred. Check the cause field for the diagnostics.

GET /article/{id}/comment/{commentId} [Up]

Endpoint that returns the selected comment. (**articleIdCommentCommentIdGet**)

Path parameters

id (required)
Path Parameter — Id of the article **commentId (required)**
Path Parameter — Id of the comment

Return type
Comment

Example data
Content-Type: application/json

```
{
  "date" : "2019-11-14T21:52:05.381Z",
  "edited" : true,
  "author" : {
    "surname" : "Doe",
    "name" : "John"
  },
  "lastEditDate" : "2019-11-14T21:52:05.381Z",
  "_id" : "5daedb5af05eb852753ac9c0",
  "authorId" : "5daedb59f05eb852753ac9be",
  "body" : "That's not even remotely funny." }
```

Produces
This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- `application/json`

Responses
200
OK. Comment returned. Comment **400**
Validation error. Either the comment or the article not found. **500**
Internal error occurred. Check the cause field for the diagnostics.

[Up] POST /article/{id}/comment/{commentId}

Endpoint that updates the selected comment. (**articleIdCommentCommentIdPost**)

Path parameters

id (required)
Path Parameter — Id of the article **commentId**
(required)
Path Parameter — Id of the comment

Consumes

This API call consumes the following media types via the Content-Type request header:

Request body

commentBody **commentBody** (optional) *Body*
Parameter — New body of the comment.

Produces

This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- `application/json`

Responses

200
OK. Comment updated. **400**
Validation error. Either the comment or the article not found or the body was empty. **500**
Internal error occurred. Check the cause field for the diagnostics.

Up

POST /article/{id}/comment

Endpoint to add a comment to an article. (**articleIdCommentPost**)

Path parameters

id (required)
Path Parameter — Id of the article.

Consumes

This API call consumes the following media types via the Content-Type request header:

- `application/json`

Request body

comment **comment** (optional)
Body Parameter — Comment to be added.

Produces

This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- `application/json`

Responses

200
OK. Comment successfully added. **400**
Validation error. The body of the comment may be missing or the article with the given id not found. Check causes field for diagnostics. **401**
Authorization token was missing, invalid or expired. **500**
Internal error occurred. Check the cause field for the diagnostics.

POST /article/{id}/down-vote

Up

Endpoint to down vote an article. (**articleIdDownVotePost**)

Path parameters

id (required)
Path Parameter — Id of the article.

Produces

This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- `application/json`

Responses

200

OK. Article successfully down voted. **400**

Validation error. Either the article with the given id or the user with the given token not found. Check causes field for diagnostics.

401

Authorization token was missing, invalid or expired. **500**

Internal error occurred. Check the cause field for the diagnostics.

GET /article/{id}

Up

An endpoint that returns the article with the given id. (**articleIdGet**)

Path parameters

id (required)

Path Parameter — Id of the article.

Return type

ArticleDetail

Example data

Content-Type: application/json

```
""
```

Produces

This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- `application/json`

Responses

200

OK. The article with the given id is returned. ArticleDetail

Up POST /article/{id}/up-vote

Endpoint to up vote an article. (**articleIdUpVotePost**)

Path parameters

id (required)

Path Parameter — Id of the article.

Produces

This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- `application/json`

Responses

200

OK. Article successfully up voted. **400**

Validation error. Either the article with the given id or the user with the given token not found. Check causes field for diagnostics.

401

Authorization token was missing, invalid or expired. **500**

Internal error occurred. Check the cause field for the diagnostics.

POST /article

Up

Endpoint to add an article. (**articlePost**)

Consumes

This API call consumes the following media types via the Content-Type request header:

- application/json

Request body

article **article** **(optional)**
Body Parameter — Article to be added.

Return type
Article

Example data
Content-Type: application/json


```
{
  "date" : "2019-11-14T21:52:05.381Z",
  "author" : {
    "surname" : "Doe",
    "name" : "John"
  },
  "_id" : "5da07611e407903a5d01b265",
  "voteCount" : 72.0,
  "userVote" : 1.0,
  "title" : "Low-Cost Investing Can’t Get Any Lower Than Free",
  "body" : "The central bank cut rates for the first time since the Great Recession in late July...",
  "authorId" : "5da07611e407903a5d01b265"
}
```


Produces
This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

Responses
200 400 401
OK. Article is successfully added. Article
Validation error. Either the title or the body of the article was missing. Check causes field for diagnostics.
Authorization token was missing, invalid or expired. **500** Internal error
occurred. Check the cause field for the diagnostics.

Auth

POST /auth/login/google

Up

Google login endpoint. (**authLoginGooglePost**)

Consumes
This API call consumes the following media types via the Content-Type request header:

- application/json

Request body

idToken **idToken** **(optional)**
Body Parameter — Id token for the login request.

Produces
This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/json

Responses
200
OK. User successfully logged in. User object and JWT token is returned. **401**
Invalid credentials. Either the user with the given email does not exists or the password does not matches the password for the given email.
500
Internal error occurred. Check the cause field for the diagnostics.

POST /auth/login

Up

Login endpoint. (**authLoginPost**)

Consumes

This API call consumes the following media types via the Content-Type request header:

- `application/json`

Request body

credentials [credentials](#) **(optional)**
Body Parameter — The user credentials to login.

Produces

This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- `application/json`

Responses

200
OK. User successfully logged in. User object and JWT token is returned. **401**
Invalid credentials. Either the user with the given email does not exists or the password does not matches the password for the given email.
500
Internal error occurred. Check the cause field for the diagnostics.

`POST /auth/sign-up` Up

Sign up endpoint for the Papel. (**authSignUpPost**)

Either the password or the Google user id must be set, not both. When Google user id is set, password will be discarded so if you are not doing sign up with Google, do not set it.

Consumes

This API call consumes the following media types via the Content-Type request header:

- `application/json`

Request body

user [User](#) **(optional)**
Body Parameter — The user to sign up.

Produces

This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- `application/json`

Responses

200 400 500
OK. Successfully signed up. A verification email is sent.
Validation error occurred for one of the parameters. Check the name and message fields for diagnostics.
Internal error occurred. Check the cause field for the diagnostics.

`POST /auth/sign-up/verification/resend` Up

Email verification resend endpoint. (**authSignUpVerificationResendPost**)

Consumes

This API call consumes the following media types via the Content-Type request header:

- `application/json`

Request body

email [email](#) **(optional)**
Body Parameter — Email of the user.

Produces

This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type

response header.

- application/json

Responses
200 400 500
OK. Verification email successfully resent.
Either a user not found with the given email or the user already verified.
Internal error occurred. Check the cause field for the diagnostics.

Up `GET /auth/sign-up/verification/{token}`

Email verification endpoint. (**authSignUpVerificationTokenGet**)

Path parameters

token (required)
Path Parameter — Verification token

Produces

This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/text application/json

Responses
200
OK. User successfully validated. **400**
Validation error occurred for one of the parameters. Check the name and message fields for diagnostics.
500 Internal error occurred. Check the cause field for the diagnostics.

Currency

`POST /currency/{code}/clear-prediction` Up

Endpoint to clear prediction on a currency. (**currencyCodeClearPredictionPost**)

Path parameters

code (required)
Path Parameter — Code of the currency.

Produces
This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.
Responses
200
OK. Prediction for the currency successfully cleared. **400**
Validation error. Either the currency with the given code, the user with the given token or the vote not found. Check causes field for diagnostics.

401
Authorization token was missing, invalid or expired. **500**
Internal error occurred. Check the cause field for the diagnostics.

DELETE /currency/{code}/comment/{commentId}

Endpoint that deletes the selected comment. (**currencyCodeCommentCommentIdDelete**)

Path parameters

code (required)
Path Parameter — Code of the currency **commentId**

(required)

Path Parameter — Id of the comment

Responses

200
OK. Comment deleted.
400
Currency, user or the comment not found.
500
Internal error occurred. Check the cause field for the diagnostics.

GET /currency/{code}/comment/{commentId}

Up
Up

Endpoint that returns the selected comment. (**currencyCodeCommentCommentIdGet**)

Path parameters

code (required)
Path Parameter — Code of the currency **commentId**

(required)

Path Parameter — Id of the comment

Return type Comment

Example data

Content-Type: application/json


```
{
  "date" : "2019-11-14T21:52:05.381Z",
  "edited" : true,
  "author" : {
    "surname" : "Doe",
    "name" : "John"
  },
  "lastEditDate" : "2019-11-14T21:52:05.381Z",
  "_id" : "5daedb5af05eb852753ac9c0",
  "authorId" : "5daedb59f05eb852753ac9be",
  "body" : "That's not even remotely funny." }
```

Produces
This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- `application/json`

Responses
200
OK. Comment returned. Comment **400**
Validation error. Either the comment or the currency not found. **500**
Internal error occurred. Check the cause field for the diagnostics.

POST /currency/{code}/comment/{commentId} Up

Endpoint that updates the selected comment. (**currencyCodeCommentCommentIdPost**)

Path parameters

code (required)
Path Parameter — Code of the currency

commentId (required)
Path Parameter — Id of the comment

Consumes
This API call consumes the following media types via the Content-Type request header:

- `application/json`

Request body

commentBody **commentBody_1** **(optional)** *Body Parameter* — New body of the comment.

Produces
This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type

response header.

- `application/json`

Responses
200
OK. Comment updated. **400**
Validation error. Either the comment or the currency not found or the body was empty.
500
Internal error occurred. Check the cause field for the diagnostics.

Up POST /currency/{code}/comment

Endpoint to add a comment to a currency. (**currencyCodeCommentPost**)

Path parameters

code (required)
Path Parameter — Code of the currency.

Consumes
This API call consumes the following media types via the Content-Type request header:

- `application/json`

Request body

comment **comment_1** **(optional)** *Body Parameter* — Comment to be added.

Produces

This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/json

Responses

200
OK. Comment successfully added. **400**
Validation error. The body of the comment may be missing or the currency with the given code not found. Check causes field for diagnostics. **401**
Authorization token was missing, invalid or expired. **500**
Internal error occurred. Check the cause field for the diagnostics.

GET /currency/{code}/full

Up

Returns the rate, intraday rates with 5 minute intervals and the last 100 daily rates of the given currency. (**currencyCodeFullGet**)

Path parameters

code (required)
Path Parameter — Code of the currency.

Return type CurrencyFull

Example data
Content-Type: application/json

```
""
```

Produces
This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/json

Responses

200

OK. Given currency and its rate, intraday rates with 5 minute intervals and the last 100 daily rates are returned. CurrencyFull **400**

Given code for the currency is not valid.

500

Internal error occurred. Check the cause field for the diagnostics.

GET /currency/{code}

Up

Returns the rate and comments of the given currency. (**currencyCodeGet**)

Path parameters

code (required)

Path Parameter — Code of the currency.

Return type CurrencyGet

Example data

Content-Type: application/json

```
""
```

Produces

This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- `application/json`

Responses

200

OK. Given currency, its rate and comments are returned. CurrencyGet **400**

Given code for the currency is not valid.

500

Internal error occurred. Check the cause field for the diagnostics.

GET /currency/{code}/intraday

Up

Returns the rate and the intraday rates with 5 minutes intervals of the given currency. (**currencyCodeIntradayGet**)

Path parameters

code (required)

Path Parameter — Code of the currency.

Return type

CurrencyIntraday

Example data

Content-Type: application/json

```
""
```

Produces

This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- `application/json`

Responses

200 400 500

OK. Given currency and its rate and intraday rates with 5 minutes intervals are returned. CurrencyIntraday

Given code for the currency is not valid.

Internal error occurred. Check the cause field for the diagnostics.

GET /currency/{code}/last-100

Up

Returns the rate and the last 100 daily rates of the given currency. (**currencyCodeLast100Get**)

Path parameters

code (required)
Path Parameter — Code of the currency.

Return type
CurrencyLast100

Example data
Content-Type: application/json

```
""
```

Produces
This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type

response header.

- `application/json`

Responses
200 400 500
OK. Given currency and its rate and the last 100 daily rates are returned. CurrencyLast100
Given code for the currency is not valid.
Internal error occurred. Check the cause field for the diagnostics.

Up `GET /currency/{code}/last-month`

Returns the rate and the last 30 daily rates of the given currency. (**currencyCodeLastMonthGet**)

Path parameters

code (required)
Path Parameter — Code of the currency.

Return type
CurrencyLastMonth

Example data
Content-Type: application/json

```
""
```

Produces
This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type

response header.

- `application/json`

Responses
200 400 500
OK. Given currency and its rate and the last 30 daily rates are returned. CurrencyLastMonth
Given code for the currency is not valid.
Internal error occurred. Check the cause field for the diagnostics.

Up `GET /currency/{code}/last-week`

Returns the rate and the last 7 daily rates of the given currency. (**currencyCodeLastWeekGet**)

Path parameters

code (required)
Path Parameter — Code of the currency.

Return type
CurrencyLastWeek

Example data
Content-Type: application/json

```
""
```

Produces
This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- `application/json`

Responses
200 400 500
OK. Given currency and its rate and the last 7 daily rates are returned. CurrencyLastWeek
Given code for the currency is not valid.
Internal error occurred. Check the cause field for the diagnostics.

Up POST /currency/{code}/predict-decrease

Endpoint to predict a decrease on a currency. (**currencyCodePredictDecreasePost**)

Path parameters
code (required)
Path Parameter — Code of the currency.

Produces
This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- `application/json`

Responses
200
OK. Prediction successfully made. **400**
Validation error. Either the currency with the given code or the user with the given token not found. Check causes field for diagnostics. **401**
Authorization token was missing, invalid or expired. **500**
Internal error occurred. Check the cause field for the diagnostics.

POST /currency/{code}/predict-increase Up

Endpoint to predict an increase on a currency. (**currencyCodePredictIncreasePost**)

Path parameters
code (required)
Path Parameter — Code of the currency.

Produces
This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- `application/json`

Responses
200
OK. Prediction successfully made. **400**
Validation error. Either the currency with the given code or the user with the given token not found. Check causes field for diagnostics. **401**
Authorization token was missing, invalid or expired. **500**
Internal error occurred. Check the cause field for the diagnostics.

GET /currency Up

Returns the list of currencies with their current rates. (**currencyGet**)

Return type
array[CurrencyMinimal]

Example data
Content-Type: application/json


```
[ {
  "code" : "EUR",
  "rate" : 0.91,
  "name" : "European Euro",
  "_id" : "5dd93e4301df5b4513254756"
}, {
  "code" : "EUR",
  "rate" : 0.91,
  "name" : "European Euro",
  "_id" : "5dd93e4301df5b4513254756" } ]
```


Produces
This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/json

Responses
200
OK. All of the currencies and their rates are returned.
500 Internal error occurred. Check the cause field for the diagnostics.

Event

GET /event

Up

Event get all endpoint. (**eventGet**)

Consumes
This API call consumes the following media types via the Content-Type request header:

- application/json

Return type
array[inline_response_200]

Example data
Content-Type: application/json


```
[ {
  "date" : "date",
  "country" : "Turkey",
  "rank" : 3.0,
  "comment" : [ "comment2", "comment2" ],
  "title" : "title",
  "body" : "272.7 example"
}, {
  "date" : "date",
  "country" : "Turkey",
  "rank" : 3.0,
  "comment" : [ "comment2", "comment2" ],
  "title" : "title",
  "body" : "272.7 example" }
]
```


Produces
This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/json

Responses
200
OK. All Events returned succesfully.

GET /event/{_id}

Up

Returns requested event with historical data (**eventIdGet**)

Path parameters

_id (required)
Path Parameter — Unique event ID

Consumes

This API call consumes the following media types via the Content-Type request header:

- `application/json`

Return type inline_response_200

Example data

Content-Type: application/json

```
{
  "date" : "date",
  "country" : "Turkey",
  "rank" : 3.0,
  "comment" : [ "comment2", "comment2" ],
  "title" : "title",
  "body" : "272.7 example" }
```

Produces

This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- `application/json`

Responses

200 OK. Event returned successfully. inline_response_200

Investments

`GET /investments`

Up

Returns the list of investments with trading equipments. (**investmentsGet**)

Return type array[Investments]

Example data

Content-Type: application/json

```
[ {
  "_id" : "_id",
  "userId" : "5ddbeb2296b198e2cf2da98a",
  "stocks" : [ {
    "amount" : 32.0,
    "stock" : {
      "stockSymbol" : "ADBE",
      "stockName" : "Adobe Inc. - Common Stock",
      "price" : 272.7,
      "name" : "ADBE - Adobe Inc. - Common Stock",
      "_id" : "_id"
    }
  }, {
    "amount" : 32.0,
    "stock" : {
      "stockSymbol" : "ADBE",
      "stockName" : "Adobe Inc. - Common Stock",
      "price" : 272.7,
      "name" : "ADBE - Adobe Inc. - Common Stock",
      "_id" : "_id"
    }
  } ],
  "currencies" : [ {
    "amount" : 53.0,
    "stock" : {
      "code" : "EUR",
      "rate" : 0.91,
      "name" : "European Euro",
      "_id" : "5dd93e4301df5b4513254756"
    }
```

```
  }, {
    "amount" : 53.0,
    "stock" : {
      "code" : "EUR",
      "rate" : 0.91,
      "name" : "European Euro",
      "_id" : "5dd93e4301df5b4513254756"
    }
  } ]
}, {
  "_id" : "_id",
  "userId" : "5ddbeb2296b198e2cf2da98a",
  "stocks" : [ {
    "amount" : 32.0,
    "stock" : {
      "stockSymbol" : "ADBE",
      "stockName" : "Adobe Inc. - Common Stock",
      "price" : 272.7,
      "name" : "ADBE - Adobe Inc. - Common Stock",
      "_id" : "_id"
    }
  }, {
    "amount" : 32.0,
    "stock" : {
      "stockSymbol" : "ADBE",
      "stockName" : "Adobe Inc. - Common Stock",
      "price" : 272.7,
      "name" : "ADBE - Adobe Inc. - Common Stock",
      "_id" : "_id"
    }
  } ],
  "currencies" : [ {
    "amount" : 53.0,
    "stock" : {
      "code" : "EUR",
      "rate" : 0.91,
      "name" : "European Euro",
      "_id" : "5dd93e4301df5b4513254756"
    }
  }, {
    "amount" : 53.0,
    "stock" : {
      "code" : "EUR",
      "rate" : 0.91,
      "name" : "European Euro",
      "_id" : "5dd93e4301df5b4513254756"
    }
  } ]
} ]
```


Produces

This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- `application/json`

Responses

200

OK. All of the investments are returned. **500**

Internal error occurred. Check the cause field for the diagnostics.

Up `GET /investments/{_id}`

Returns information about the given "investments". (**investmentsIdGet**)

Path parameters

_id (required)

Path Parameter — Object id of the "investments".

Return type array[Investments]

Example data

Content-Type: application/json

```
[ {
  "_id" : "_id",
  "userId" : "5ddbeb2296b198e2cf2da98a",
  "stocks" : [ {
    "amount" : 32.0,
    "stock" : {
      "stockSymbol" : "ADBE",
      "stockName" : "Adobe Inc. - Common Stock",
      "price" : 272.7,
      "name" : "ADBE - Adobe Inc. - Common Stock",
      "_id" : "_id"
    }
  }, {
    "amount" : 32.0,
    "stock" : {
      "stockSymbol" : "ADBE",
      "stockName" : "Adobe Inc. - Common Stock",
      "price" : 272.7,
      "name" : "ADBE - Adobe Inc. - Common Stock",
      "_id" : "_id"
    }
  } ],
  "currencies" : [ {
    "amount" : 53.0,
    "stock" : {
      "code" : "EUR",
      "rate" : 0.91,
      "name" : "European Euro",
      "_id" : "5dd93e4301df5b4513254756"
    }
  }, {
    "amount" : 53.0,
    "stock" : {
      "code" : "EUR",
      "rate" : 0.91,
      "name" : "European Euro",
      "_id" : "5dd93e4301df5b4513254756"
    }
  } ]
}, {
  "_id" : "_id",
  "userId" : "5ddbeb2296b198e2cf2da98a",
  "stocks" : [ {
    "amount" : 32.0,
    "stock" : {
      "stockSymbol" : "ADBE",
      "stockName" : "Adobe Inc. - Common Stock",
      "price" : 272.7,
      "name" : "ADBE - Adobe Inc. - Common Stock",
      "_id" : "_id"
    }
  }, {
    "amount" : 32.0,
    "stock" : {
      "stockSymbol" : "ADBE",
      "stockName" : "Adobe Inc. - Common Stock",
      "price" : 272.7,
      "name" : "ADBE - Adobe Inc. - Common Stock",
      "_id" : "_id"
    }
  } ],
  "currencies" : [ {
    "amount" : 53.0,
    "stock" : {
      "code" : "EUR",
      "rate" : 0.91,
      "name" : "European Euro",
      "_id" : "5dd93e4301df5b4513254756"
    }
  }, {
    "amount" : 53.0,
    "stock" : {
      "code" : "EUR",
      "rate" : 0.91,
      "name" : "European Euro",
```

```
      "_id" : "5dd93e4301df5b4513254756"
    }
  } ]
} ]
```


Produces

This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- `application/json`

Responses

200

OK. Given investments are returned. **500** Internal error occurred.
Check the cause field for the diagnostics.

Portfolio

Up `GET /portfolio`

Portfolio get all endpoint.(Actually its unnecessary but still it can be useful for development) (**portfolioGet**)

Consumes

This API call consumes the following media types via the Content-Type request header:

- `application/json`

Return type

array[Portfolio]

Example data

Content-Type: application/json


```
[ {
  "name" : "My Portfolio",
  "_id" : "_id",
  "userId" : "5da076d955d9b93a7308f3d7",
  "stocks" : [ {
    "stockSymbol" : "ADBE",
    "stockName" : "Adobe Inc. - Common Stock",
    "price" : 272.7,
    "name" : "ADBE - Adobe Inc. - Common Stock",
    "_id" : "_id"
  }, {
    "stockSymbol" : "ADBE",
    "stockName" : "Adobe Inc. - Common Stock",
    "price" : 272.7,
    "name" : "ADBE - Adobe Inc. - Common Stock",
    "_id" : "_id"
  } ]
}, {
  "name" : "My Portfolio",
  "_id" : "_id",
  "userId" : "5da076d955d9b93a7308f3d7",
  "stocks" : [ {
    "stockSymbol" : "ADBE",
    "stockName" : "Adobe Inc. - Common Stock",
    "price" : 272.7,
    "name" : "ADBE - Adobe Inc. - Common Stock",
    "_id" : "_id"
  }, {
    "stockSymbol" : "ADBE",
    "stockName" : "Adobe Inc. - Common Stock",
    "price" : 272.7,
    "name" : "ADBE - Adobe Inc. - Common Stock",
    "_id" : "_id"
  } ]
} ]
```


Produces

This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- `application/json`

Responses
200 503
OK. All Stocks returned succesfully.
Internal Server Error

DELETE /portfolio/{_id}/currency

Up

Delete the currency to addressed portfolio /portfolio/:id describes the portfolio and /currency describes that request body should be a currency (**portfolioIdCurrencyDelete**)

Path parameters

_id (required)
Path Parameter — Portfolio ID

Consumes
This API call consumes the following media types via the Content-Type request header:

- application/json

Request body

currency CurrencyMinimal **(optional)**
Body Parameter — Delete the specific currency from specific portfolio Note: _id field should be provided it is required

Return type
Portfolio

Example data
Content-Type: application/json


```
{
  "name" : "My Portfolio",
  "_id" : "_id",
  "userId" : "5da076d955d9b93a7308f3d7",
  "stocks" : [ {
    "stockSymbol" : "ADBE",
    "stockName" : "Adobe Inc. - Common Stock",
    "price" : 272.7,
    "name" : "ADBE - Adobe Inc. - Common Stock",
    "_id" : "_id"
  }, {
    "stockSymbol" : "ADBE",
    "stockName" : "Adobe Inc. - Common Stock",
    "price" : 272.7,
    "name" : "ADBE - Adobe Inc. - Common Stock",
    "_id" : "_id"
  } ]
}
```


Produces
This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/json

Responses
200
OK. The currency in portfolio deleted successfully, Actually it should be No Content but for updating the portfolio immediately the endpoint sends the updated portfolio Portfolio **503**
Internal error occurred.

POST /portfolio/{_id}/currency

Up

Add the currency to addressed portfolio /portfolio/:id describes the portfolio and /stock describes that request body should be a stock (**portfolioIdCurrencyPost**)

Path parameters

_id (required)

Path Parameter — Portfolio ID

Consumes
This API call consumes the following media types via the Content-Type request header:

- `application/json`

Request body

currency [CurrencyMinimal](#) **(optional)**
Body Parameter — Add a currency to specific portfolio

Return type
[Portfolio](#)

Example data
Content-Type: application/json

```
{
  "name" : "My Portfolio",
  "_id" : "_id",
  "userId" : "5da076d955d9b93a7308f3d7",
  "stocks" : [ {
    "stockSymbol" : "ADBE",
    "stockName" : "Adobe Inc. - Common Stock",
    "price" : 272.7,
    "name" : "ADBE - Adobe Inc. - Common Stock",
    "_id" : "_id"
  }, {
    "stockSymbol" : "ADBE",
    "stockName" : "Adobe Inc. - Common Stock",
    "price" : 272.7,
    "name" : "ADBE - Adobe Inc. - Common Stock",
    "_id" : "_id"
  } ]
}
```

Produces
This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- `application/json`

Responses
200 **503**
OK. The portfolio created successfully [Portfolio](#)
Internal error occurred.

Up `DELETE /portfolio/{_id}/stock`

Delete the stock to addressed portfolio /portfolio/:id describes the portfolio and /stock describes that request body should be a stock (**portfolioIdStockDelete**)

Path parameters

_id (required)
Path Parameter — Portfolio ID

Consumes
This API call consumes the following media types via the Content-Type request header:

- `application/json`

Request body

stock [Stock](#) **(optional)**
Body Parameter — Delete the specific stock from specific portfolio Note: _id field should be provided it is required

Return type
[Portfolio](#)

Example data
Content-Type: application/json


```
{
  "name" : "My Portfolio",
  "_id" : "_id",
  "userId" : "5da076d955d9b93a7308f3d7",
  "stocks" : [ {
    "stockSymbol" : "ADBE",
    "stockName" : "Adobe Inc. - Common Stock",
    "price" : 272.7,
    "name" : "ADBE - Adobe Inc. - Common Stock",
    "_id" : "_id"
  }, {
    "stockSymbol" : "ADBE",
    "stockName" : "Adobe Inc. - Common Stock",
    "price" : 272.7,
    "name" : "ADBE - Adobe Inc. - Common Stock",
    "_id" : "_id"
  } ]
}
```


Produces
This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- `application/json`

Responses
200
OK. The stock in portfolio deleted successfully, Actually it should be No Content but for updating the portfolio immediately the endpoint sends the updated portfolio Portfolio **503**
Internal error occurred.

`POST /portfolio/{_id}/stock` Up

Add the stock to addressed portfolio /portfolio/:id describes the portfolio and /stock describes that request body should be a stock (**portfolioIdStockPost**)

Path parameters

_id (required)
Path Parameter — Portfolio ID

Consumes
This API call consumes the following media types via the Content-Type request header:

- `application/json`

Request body

stock Stock (optional)
Body Parameter — Add a stock to specific portfolio

Return type
Portfolio

Example data
Content-Type: application/json


```
{
  "name" : "My Portfolio",
  "_id" : "_id",
  "userId" : "5da076d955d9b93a7308f3d7",
  "stocks" : [ {
    "stockSymbol" : "ADBE",
    "stockName" : "Adobe Inc. - Common Stock",
    "price" : 272.7,
    "name" : "ADBE - Adobe Inc. - Common Stock",
    "_id" : "_id"
  }, {
    "stockSymbol" : "ADBE",
    "stockName" : "Adobe Inc. - Common Stock",
    "price" : 272.7,
```

```
    "name" : "ADBE - Adobe Inc. - Common Stock",
    "_id" : "_id"
  } ]
}
```


Produces

This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- `application/json`

Responses

200 503

OK. The portfolio created successfully Portfolio
Internal error occurred.

`GET /portfolio/{portfolioId}`

Up

Portfolio get all endpoint.(Actually its unnecessary but still it can be useful for development) (**portfolioPortfolioIdGet**)

Path parameters

portfolioId (required) *Path*
Parameter — Portfolio ID

Consumes

This API call consumes the following media types via the Content-Type request header:

- `application/json`

Return type Portfolio

Example data

Content-Type: application/json


```
{
  "name" : "My Portfolio",
  "_id" : "_id",
  "userId" : "5da076d955d9b93a7308f3d7",
  "stocks" : [ {
    "stockSymbol" : "ADBE",
    "stockName" : "Adobe Inc. - Common Stock",
    "price" : 272.7,
    "name" : "ADBE - Adobe Inc. - Common Stock",
    "_id" : "_id"
  }, {
    "stockSymbol" : "ADBE",
    "stockName" : "Adobe Inc. - Common Stock",
    "price" : 272.7,
    "name" : "ADBE - Adobe Inc. - Common Stock",
    "_id" : "_id"
  } ]
}
```


Produces

This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- `application/json`

Responses

200 503

OK. All Stocks returned successfully. Portfolio
Internal Server Error

Up `POST /portfolio`

Create portfolio endpoint (**portfolioPost**)

Consumes

This API call consumes the following media types via the Content-Type request header:

- application/json

Request body

portfolio **portfolio** **(optional)**
Body Parameter — The portfolio to create

Return type
Portfolio

Example data
Content-Type: application/json


```
{
  "name" : "My Portfolio",
  "_id" : "_id",
  "userId" : "5da076d955d9b93a7308f3d7",
  "stocks" : [ {
    "stockSymbol" : "ADBE",
    "stockName" : "Adobe Inc. - Common Stock",
    "price" : 272.7,
    "name" : "ADBE - Adobe Inc. - Common Stock",
    "_id" : "_id"
  }, {
    "stockSymbol" : "ADBE",
    "stockName" : "Adobe Inc. - Common Stock",
    "price" : 272.7,
    "name" : "ADBE - Adobe Inc. - Common Stock",
    "_id" : "_id"
  } ]
}
```


Produces
This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/json

Responses
200 **503**
OK. The portfolio created successfully Portfolio
Internal error occurred.

Up `GET /portfolio/user/{userId}`

Returns all portfolios of the specific user. (**portfolioUserUserIdGet**)

Path parameters

userId **(required)** *Path Parameter* — User ID

Consumes
This API call consumes the following media types via the Content-Type request header:

- application/json

Return type
array[Portfolio]

Example data
Content-Type: application/json


```
[ {
  "name" : "My Portfolio",
  "_id" : "_id",
  "userId" : "5da076d955d9b93a7308f3d7",
  "stocks" : [ {
    "stockSymbol" : "ADBE",
    "stockName" : "Adobe Inc. - Common Stock",
    "price" : 272.7,
    "name" : "ADBE - Adobe Inc. - Common Stock",
    "_id" : "_id"
  }, {
    "stockSymbol" : "ADBE",
    "stockName" : "Adobe Inc. - Common Stock",
```

```
    "price" : 272.7,
    "name" : "ADBE - Adobe Inc. - Common Stock",
    "_id" : "_id"
  } ]
}, {
  "name" : "My Portfolio",
  "_id" : "_id",
  "userId" : "5da076d955d9b93a7308f3d7",
  "stocks" : [ {
    "stockSymbol" : "ADBE",
    "stockName" : "Adobe Inc. - Common Stock",
    "price" : 272.7,
    "name" : "ADBE - Adobe Inc. - Common Stock",
    "_id" : "_id"
  }, {
    "stockSymbol" : "ADBE",
    "stockName" : "Adobe Inc. - Common Stock",
    "price" : 272.7,
    "name" : "ADBE - Adobe Inc. - Common Stock",
    "_id" : "_id"
  } ]
} ]
```


Produces

This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- `application/json`

Responses

200

OK. The portfolios sent successfully

503 Internal error occurred.

Profile

Up `GET /profile/myprofile`

Returns user's own profile (**profileMyprofileGet**)

Consumes

This API call consumes the following media types via the Content-Type request header: `application/json`

Return type MyProfile

Example data

Content-Type: application/json


```
{
  "followerPending" : "[{\"_id\": \"5dd7d54b58ec764434728fea\",\"name\": \"Eleman\",\"surname\": \"Takip\",\"email\": \"stel@2
  "followers" : "[{\"_id\": \"5dd7d57758ec764434728fec\",\"name\": \"İbrahim\",\"surname\": \"Kamacı\",\"email\": \"stel@1000m
  "isMe" : true,
  "surname" : "Doe",
  "following" : "[{\"_id\": \"5dd7d54b58ec764434728fea\",\"name\": \"Eleman\",\"surname\": \"Takip\",\"email\": \"mcrawfor@com
  "name" : "John",
  "portfolios" : [ {
    "name" : "My Portfolio",
    "_id" : "_id",
    "userId" : "5da076d955d9b93a7308f3d7",
    "stocks" : [ {
      "stockSymbol" : "ADBE",
      "stockName" : "Adobe Inc. - Common Stock",
      "price" : 272.7,
      "name" : "ADBE - Adobe Inc. - Common Stock",
      "_id" : "_id"
    }, {
      "stockSymbol" : "ADBE",
      "stockName" : "Adobe Inc. - Common Stock",
      "price" : 272.7,
      "name" : "ADBE - Adobe Inc. - Common Stock",
```

```
    "_id" : "_id"
  } ]
}, {
  "name" : "My Portfolio",
  "_id" : "_id",
  "userId" : "5da076d955d9b93a7308f3d7",
  "stocks" : [ {
    "stockSymbol" : "ADBE",
    "stockName" : "Adobe Inc. - Common Stock",
    "price" : 272.7,
    "name" : "ADBE - Adobe Inc. - Common Stock",
    "_id" : "_id"
  }, {
    "stockSymbol" : "ADBE",
    "stockName" : "Adobe Inc. - Common Stock",
    "price" : 272.7,
    "name" : "ADBE - Adobe Inc. - Common Stock",
    "_id" : "_id"
  } ]
} ],
"privacy" : "private",
"location" : {
  "latitude" : 41.085987,
  "longitude" : 29.044008
},
"articles" : [ {
  "date" : "2019-11-14T21:52:05.381Z",
  "author" : {
    "surname" : "Doe",
    "name" : "John"
  },
  "_id" : "5da07611e407903a5d01b265",
  "voteCount" : 72.0,
  "userVote" : 1.0,
  "title" : "Low-Cost Investing Can’t Get Any Lower Than Free",
  "body" : "The central bank cut rates for the first time since the Great Recession in late July...",
  "authorId" : "5da07611e407903a5d01b265"
}, {
  "date" : "2019-11-14T21:52:05.381Z",
  "author" : {
    "surname" : "Doe",
    "name" : "John"
  },
  "_id" : "5da07611e407903a5d01b265",
  "voteCount" : 72.0,
  "userVote" : 1.0,
  "title" : "Low-Cost Investing Can’t Get Any Lower Than Free",
  "body" : "The central bank cut rates for the first time since the Great Recession in late July...",
  "authorId" : "5da07611e407903a5d01b265"
} ],
  "followingPending" : "[{\"_id\": \"5daedb59f05eb852753ac9bc\",\"name\": \"Kâni\",\"surname\": \"Hayal\",\"email\": \"stel@24
}
```


Produces
This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- `application/json`

Responses
200 400 401
OK. The user's profile is returned. MyProfile
User not found
Authorization token was missing, invalid or expired. **500**
Internal error occurred. Check the cause field for the diagnostics.

POST /profile/other/{_id}/accept

Up

Accepts the incoming follow request of user which has given ID (**profileOtherIdAcceptPost**)

Path parameters

_id (required)

Path Parameter — user ID

Consumes
This API call consumes the following media types via the Content-Type request header:

- `application/json`

Return type inline_response_200_3

Example data
Content-Type: application/json

```
{
  "msg" : "Follow request accepted",
  "status" : "accept"
}
```

Produces
This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- `application/json`

Responses
200 **400**
OK. Follow request accepted successfully inline_response_200_3
User not found **401**
Authorization token was missing, invalid or expired. **500**
Internal error occurred. Check the cause field for the diagnostics.

`POST /profile/other/{_id}/decline` Up

Declines the incoming follow request (**profileOtherIdDeclinePost**)

Path parameters

_id (required)
Path Parameter — user ID

Consumes
This API call consumes the following media types via the Content-Type request header:

- `application/json`

Return type inline_response_200_4

Example data
Content-Type: application/json

```
{
  "msg" : "Follow request declined",
  "status" : "decline"
}
```

Produces
This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- `application/json`

Responses
200 **400**
OK. Follow request has been declined successfully inline_response_200_4
User not found **401**
Authorization token was missing, invalid or expired. **500**
Internal error occurred. Check the cause field for the diagnostics.

`POST /profile/other/{_id}/follow`

Performs follow function on the user who is to follow (**profileOtherIdFollowPost**)

Path parameters

_id (required)

Path Parameter — user ID

Consumes

This API call consumes the following media types via the Content-Type request header: `application/json`

Return type inline_response_200_1

Example data
Content-Type: application/json

```
{
  "msg" : "Follow request has been sent to İbrahim Kamacı",
  "status" : "follow"
}
```

Produces

Up

This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- `application/json`

Responses
200 400
OK. The user profile with the given id is returned. inline_response_200_1
The User had been already followed before **401**
Authorization token was missing, invalid or expired. **500**
Internal error occurred. Check the cause field for the diagnostics.

GET /profile/other/{_id}

Up

Returns requested user profile with his/her articles and portfolios (Bearer Token not required because guests may try to see a profile) (**profileOtherIdGet**)

Path parameters

_id (required)
Path Parameter — user ID

Consumes
This API call consumes the following media types via the Content-Type request header:

- `application/json`

Return type
Profile

Example data
Content-Type: application/json

```
{
  "isMe" : true,   "followers" : "[{\"_id\": \"5dd7d57758ec764434728fec\",\"name\": \"İbrahim\",\"surname\":
\"Kamacı\",\"email\": \"stel@1000m
  "surname" : "Doe",
  "following" : "[{\"_id\": \"5dd7d54b58ec764434728fea\",\"name\": \"Eleman\",\"surname\": \"Takip\",\"email\": \"stel@24re
  "name" : "John",
  "portfolios" : [ {
    "name" : "My Portfolio",
    "_id" : "_id",
    "userId" : "5da076d955d9b93a7308f3d7",
    "stocks" : [ {
      "stockSymbol" : "ADBE",
      "stockName" : "Adobe Inc. - Common Stock",
      "price" : 272.7,
      "name" : "ADBE - Adobe Inc. - Common Stock",
      "_id" : "_id"
    }, {
      "stockSymbol" : "ADBE",
      "stockName" : "Adobe Inc. - Common Stock",
      "price" : 272.7,
      "name" : "ADBE - Adobe Inc. - Common Stock",
      "_id" : "_id"
    } ]
  }, {
    "name" : "My Portfolio",
    "_id" : "_id",
    "userId" : "5da076d955d9b93a7308f3d7",
    "stocks" : [ {
      "stockSymbol" : "ADBE",
      "stockName" : "Adobe Inc. - Common Stock",
      "price" : 272.7,
      "name" : "ADBE - Adobe Inc. - Common Stock",
      "_id" : "_id"
    }, {
      "stockSymbol" : "ADBE",
      "stockName" : "Adobe Inc. - Common Stock",
      "price" : 272.7,
      "name" : "ADBE - Adobe Inc. - Common Stock",
      "_id" : "_id"
    } ]
  } ],
  "privacy" : "private",
  "location" : {
    "latitude" : 41.085987,
    "longitude" : 29.044008
  },
  "articles" : [ {
    "date" : "2019-11-14T21:52:05.381Z",
    "author" : {
      "surname" : "Doe",
      "name" : "John"
    },
    "_id" : "5da07611e407903a5d01b265",
    "voteCount" : 72.0,
    "userVote" : 1.0,
    "title" : "Low-Cost Investing Can’t Get Any Lower Than Free",
    "body" : "The central bank cut rates for the first time since the Great Recession in late July...",
    "authorId" : "5da07611e407903a5d01b265"
  }, {
    "date" : "2019-11-14T21:52:05.381Z",
    "author" : {
      "surname" : "Doe",
      "name" : "John"
    },
    "_id" : "5da07611e407903a5d01b265",
    "voteCount" : 72.0,
    "userVote" : 1.0,
    "title" : "Low-Cost Investing Can’t Get Any Lower Than Free",
    "body" : "The central bank cut rates for the first time since the Great Recession in late July...",
    "authorId" : "5da07611e407903a5d01b265"
} ],
  "isInMyNetwork" : "pending" }
```


Produces

This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type

response header.

- application/json

Responses

200 401
OK. The user profile with the given id is returned. Profile
Authorization token was missing, invalid or expired. **500**
Internal error occurred. Check the cause field for the diagnostics.

Up `POST /profile/other/{_id}/unfollow`

Performs unfollow function on the user who is to unfollow (**profileOtherIdUnfollowPost**)

Path parameters

_id (required)
Path Parameter — user ID

Consumes
This API call consumes the following media types via the Content-Type request header:

- application/json

Return type inline_response_200_2

Example data
Content-Type: application/json

```
{
  "msg" : "User has been successfully unfollowed",
  "status" : "unfollow"
}
```

Produces
This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/json

Responses

200 400
OK. The user has ben unfollowed successfully inline_response_200_2
User not found **401**
Authorization token was missing, invalid or expired. **500** Internal error occurred. Check the cause field for the diagnostics.

Stock

`GET /stock` Up

Stock(Trading Equipment) get all endpoint. (**stockGet**)

Consumes
This API call consumes the following media types via the Content-Type request header:

- application/json

Return type array[Stock]

Example data
Content-Type: application/json

```
[ {
  "stockSymbol" : "ADBE",
  "stockName" : "Adobe Inc. - Common Stock",
  "price" : 272.7,
  "name" : "ADBE - Adobe Inc. - Common Stock",
  "_id" : "_id"
}, {
  "stockSymbol" : "ADBE",
  "stockName" : "Adobe Inc. - Common Stock",
  "price" : 272.7,
  "name" : "ADBE - Adobe Inc. - Common Stock",
  "_id" : "_id"
```

```
} ]
```

Produces

This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- `application/json`

Responses

200 **503**

OK. All Stocks returned succesfully.
Internal Server Error

`GET /stock/{_id}` Up

Returns requested stock with historical data (**stockIdGet**)

Path parameters

_id (required)
Path Parameter — Unique Stock ID

Consumes

This API call consumes the following media types via the Content-Type request header:

- `application/json`

Return type StockDetail

Example data

Content-Type: application/json

```
{
  "dailyPrice" : "\"2019-10-16 16:00:00\": {\n            \"1. open\": \"272.7000\",\n            \"2. high\": \"272.8300\"
"stockSymbol" : "ADBE",
  "stockName" : "Adobe Inc. - Common Stock",
  "price" : 272.7,
  "name" : "ADBE - Adobe Inc. - Common Stock",
  "_id" : "_id",
  "monthlyPrice" : "\"2019-10-16\": {\n            \"1. open\": \"270.2300\",\n            \"2. high\": \"272.8100\",\n
}
```

Produces

This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- `application/json`

Responses

200

OK. All Stocks returned succesfully. StockDetail
503 Internal Server Error

User

Up `GET /user`

An endpoint that returns all the users. (**userGet**)

Return type array[User]

Example data


```
[ {
  "password" : "strongPa$.$word",
  "surname" : "Doe",
  "iban" : "TR33 0006 1005 1978 6457 8413 26",
  "googleUserId" : "117027522423426552111",
  "name" : "John",
  "privacy" : "private",
  "location" : {
    "latitude" : 41.085987,
    "longitude" : 29.044008
  },
  "_id" : "_id",
  "idNumber" : "12345678910",
  "totalPredictionCount" : 100.0,
  "successfulPredictionCount" : 10.0,
  "email" : "johndoe@gmail.com"
}, {
  "password" : "strongPa$.$word",
  "surname" : "Doe",
  "iban" : "TR33 0006 1005 1978 6457 8413 26",
  "googleUserId" : "117027522423426552111",
  "name" : "John",
  "privacy" : "private",
  "location" : {
    "latitude" : 41.085987,
    "longitude" : 29.044008
  },
  "_id" : "_id",
  "idNumber" : "12345678910",
  "totalPredictionCount" : 100.0,
  "successfulPredictionCount" : 10.0,
  "email" : "johndoe@gmail.com"
} ]
```


Content-Type: application/json

Produces

This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- `application/json`

Responses

200 500

OK. A list of users are returned.

Internal error occurred. Check the cause field for the diagnostics.

Up `GET /user/{id}`

An endpoint that returns the user with the given id. (**userIdGet**)

Path parameters

id (required)

Path Parameter — Id of the requested user.

Return type User

Example data

Content-Type: application/json

```
{
  "password" : "strongPa$.$word",
  "surname" : "Doe",
  "iban" : "TR33 0006 1005 1978 6457 8413 26",
  "googleUserId" : "117027522423426552111",
  "name" : "John",
  "privacy" : "private",
  "location" : {
    "latitude" : 41.085987,
    "longitude" : 29.044008
  },
  "_id" : "_id",
  "idNumber" : "12345678910",
  "totalPredictionCount" : 100.0,
  "successfulPredictionCount" : 10.0,
  "email" : "johndoe@gmail.com"
}
```


Produces
This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- `application/json`

Responses
200
OK. User with the given id is returned. User **400**
There is no such user with the given id.
500
Internal error occurred. Check the cause field for the diagnostics.

PUT /user/{id}/privacy

Up

An endpoint that updates the user's privacy with the given request body (**userIdPrivacyPut**)

Path parameters

id (required)
Path Parameter — Id of the requested user.

Request body

privacy option privacy option **(optional)** *Body Parameter* — privacy option to update

Return type User
Example data
Content-Type: application/json


```
{
  "password" : "strongPa$.$word",
  "surname" : "Doe",
  "iban" : "TR33 0006 1005 1978 6457 8413 26",
  "googleUserId" : "117027522423426552111",
  "name" : "John",
  "privacy" : "private",
  "location" : {
    "latitude" : 41.085987,
    "longitude" : 29.044008
  },
  "_id" : "_id",
  "idNumber" : "12345678910",
  "totalPredictionCount" : 100.0,
  "successfulPredictionCount" : 10.0,
  "email" : "johndoe@gmail.com"
}
```


Produces
This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- `application/json`

Responses
200
OK. Updated user information is returned for real time frontend experience. User **400**

Invalid privacy option.
404 500
There is no such user with the given id.
Internal error occurred. Check the cause field for the diagnostics.

POST /user/lost-password

Up

Lost password endpoint. (**userLostPasswordPost**)

Consumes
This API call consumes the following media types via the Content-Type request header:

- application/json

Request body

email email_1 **(optional)**
Body Parameter — Email of the password who lost his password.

Produces
This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/json

Responses
200 400
OK. An email that contains instruction about how the user can reset his password is sent.
User with the given email not found. **500**
Internal error occurred. Check the cause field for the diagnostics.

Up POST /user/lost-password/reset

Password reset endpoint. (**userLostPasswordResetPost**)

Consumes
This API call consumes the following media types via the Content-Type request header:

- application/json

Request body

Token and password Token and password **(optional)**
Body Parameter — Lost password verification token and new password of the user.

Produces

This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/json

Responses
200
OK. Credentials are updated with the new password. **400**
Given lost password token is expired, invalid or the password is too short.
500
Internal error occurred. Check the cause field for the diagnostics.

6- User Scenarios

6.1.

Turgay Esen is 34 years old and works in the economy department of a newspaper. He wants to be an active user on Papel by writing articles. He has already written an article. Since he is an active user, he is logged in to the system.

He goes to homepage and sees an important event. After thinking about it for several hours, he proceeds and writes an article about it on the web platform. Then, Ṣadiye Demirci, who is a student interested in economic articles, refreshes the articles section on the android and sees the new article by Turgay Esen. She likes the article so much that she upvotes it and makes a comment. In the comment, she makes a small typo and then she edits the comment. Then, she realizes that Turgay is indeed a good writer and remembers that she made a negative comment in Turgay's other article. She goes to that article by navigating to Turgay's profile. Then, she deletes her negative comment. Then, she goes to Turgay's profile back to look at his portfolios. But, Turgay made his profile private so she is unable to do so. Then, Ṣadiye sends a follow request to Turgay. On the web platform, Turgay refreshes the page and gets a notification about the follow request. He accepts it. Finally, Ṣadiye refreshes the profile page and sees the Turgay's portfolios now.

6.2.

Hasan Yaman is a successful businessman who has little to no time for fuss work. So, when he discovers Papel, he tries to sign up to the platform using his existing Google account on his mobile phone. After successfully signing up, he uses the same Google account to login. Then, he navigates to his profile page and sees that there is a section for portfolios. So, he proceeds and adds a portfolio named Family Savings. Then, he searches the available trading equipments and adds a currency to his portfolio. Then, he goes to the currencies page. In that page, after looking at the graphs, he thinks that the currency will increase in rate and makes a prediction as such. Also, makes a comment related to his prediction. Finally, he thinks that his predictions are valuable so he adds an article about it make his prediction more visible.

7- Code Structure, Branching

We used a fork based branching model. So, each team member forked the main repository under the BounSWE, created branches on their forks as they developed features and send pull requests. The pull requests are reviewed at least by one sub-team member(backend, frontend or mobile). If there are any changes requested, it is addressed and after the approval, the pull request is merged into the master branch.

8- Evaluation of Tools and Project Management

The most important feedback we received in the Milestone 1 was that the scenarios were not that strong. For the first milestone we did not know what kind of an atmosphere

were awaiting us. In the Milestone 2 we were more experienced than the first one and we knew the expectations of the customer for the Milestone. We arranged a meeting, in which all of the team members participated, two days before the Milestone to prepare powerful scenarios which would exhibit the features we implemented. We tried to ensure that the scenarios we wrote were realistic and would not bore the audience. We had rehearsals before the milestone.

Before the Milestone 2 we received another important feedback about the issues we opened. After the feedback we tried reflect and document our work as far as possible via issues. Also we tried to increase the quality of our issues.

8.1 Backend

- **Express:** Express is a fast, unopinionated, minimalist web framework for Nodejs. We used Express to power up our backend endpoints. It was straightforward to write endpoints and the ability to use middlewares that are published by the large Express community (such as bodyParser) made our job a lot easier.
- **Swagger:** Swagger facilitated us to write API documentation for our endpoints. We were able to specify the format of data required or returned by endpoints.
- **MongoDB & Mongoose**: MongoDB is a general purpose, document-based, distributed database for modern applications. We used MongoDB Atlas as our main database. To access the data on MongoDB, we used mongoose. With the help of Mongoose, we were able to create, read, update or delete the data on MongoDB using models instead of queries. Also, the well support for Nodejs promises by Mongoose made development easier.
- **Alpha Vantage:** It is a freemium API endpoint for realtime and historical stock data, FX and cryptocurrency feeds and so on. We used it to periodically fetch currency rates and to get stock prices.

8.2 Frontend

- **ReactJS :** We used ReactJS to build the web part of the project. React JS is an open source JavaScript library. It allows us to create components on a single page which can be updated without reloading the page. React enables us to control the flow of data between the components.
- **Bootstrap :** Bootstrap is an open source CSS framework and a design tool which includes a set of elements that can be useful to build a website such as forms, labels, buttons etc.

- **Ionicons :** Ionicon is an icon package designed for web, Android and iOS applications. It is an open source.
- **Google Maps API :** We used Google Maps API in our registration page to provide location information. Map allows user to select location multiple times.
- **Jquery :** JQuery is JavaScript library that enables writing complex codes in a shorter, faster and simpler manner.

8.3 Mobile

- **Android Studio:** We used Android Studio as an IDE for Android application development since we don't have any other choice. It has lots of built-in features that help development such as code completion, code formatting and creating constructors, getters, and setters automatically for newly created classes.
- **Postman:** We used Postman to test API endpoints, analyze responses from endpoints and find out different error messages.