CMPE 451

Milestone 2 Report

Fall 2019

Sercan Ersoy

Alper Çakan

Barış Zöngür

Ali Özden

İsmet Dağlı

Volkan Yılmaz

Furkan Aydar

Efe Önal

Ceren Tahtasız

Table of contents

1. Executive Summary

This is the second milestone report that describes the work, the deliverables and the status of work accomplished by group 8 so far. We are working on a project called “Traders’ Platform” as a social platform to inspect, share, and perform actions for trading purposes.

We planned that we design articles, events, forex and digital equipment, comments on articles and trading equipments, follow and unfollow for user, search user, trading equipment and several minor functionalities. Until now we accomplished what we planned and make a good presentation to explain our application with real time scenarios.

We describe what we have done detailed in this report. We write overviewed project requirements. Backend team explain API documentation. Frontend and mobile teams explain scenarios that is created for presentation.

2. List and Status of Deliverables

Deliverable Name	Description	Status
Backend Project	The API for our frontend and the mobile client to use. Tech stack: Postgre, Django	In progress
Mobile project	Android client for Mercatus. Tech stack: Kotlin	In progress
Frontend Project	Browser client for Mercatus. Tech stack: React JS	In progress
API Documentation	The document describing the authentication and the usage of the endpoints	In progress
Deployment	Published version of the mentioned projects.	Done. Changes may be needed in the future.
Group Meetings Project Requirements	Meeting notes and agendas are in Wiki Description of what the	In progress Done. No requirement

	final product should be able to serve	will be removed, new requirements might be specified in the future.
Diagrams	Use case diagrams, class diagrams and sequence diagrams of the project.	Completed
Personas and User Stories	Various use cases with imaginary users of the app.	Completed
Project Plan		Completed
Issues	Issues can be viewed to realize how the three project was distributed into smaller problems and how the assignments were given.	In progress
Pull requests	Milestones of the commits.	In progress

3. Evaluation of Deliverables and Impacts on Plan

So far, we have completed all of what we had planned to complete before the first milestone. After several discussions with our customer Alper, we had planned to accomplish most of the functionalities within the application. Our application is built on mostly feed screens of articles, events and trading equipment. Therefore, we planned to implement those and leave minor functionalities at the end.

We are planning to finish, again just like we have discussed and concluded together with our customer, almost all of the features before the final milestone. Since we have succeeded in keeping up with our first and second milestone plan; our plan for the final milestone has not been delayed or adversely affected in anyway. So, we believe we will be able to keep up with the plan again.

4. Summary of Coding Work Done

Member	**Work Done**

Sercan Ersoy	1. Deleted unused framework Kronos from the backend project. 2. Cleaned up the code (organized imports, line breaks, indentations in order to stick with the Python coding conventions) 3. Added Kronos cron job decoration but then we decided not to use it. (I then added cron jobs manually in the server) 4. Fixed bug in the email verification part and cleaned up the code. 5. Added installation command for 'requests' library in backend project's readme file. 6. Added basic serializer for user to prevent getting recursive responses (because followers are also user objects). This change was made in the server, I only committed it. 7. Other than that, I was controlling all the deployment (e.g. deployment when new commits are merged, server is down), database management (e.g. truncating database, checking whether some endpoints work safe and sound), and job scheduling (e.g. adding cron jobs for fetching trading equipment parities, checking whether it is working fine) parts of the project.
Alper Çakan	1. Reformatted some of the backend code to comply with PEP rules 2. Fixed the bug that caused the passwords to be hashed twice 3. Implemented a feature for reading the email server config from the environment instead of hardcoding them 4. Implemented a feature for skipping the email activation (ie, activate by default) while in debug mode to speed up testing 5. Implemented a feature for saving configuring the axios to automatically send user token and headers to prevent code repetition 6. Removed the content length restriction imposed on the articles and the comments to make the models more flexible 7. Implemented the Events API for querying the financial events that happened/will happen on a particular day, along with their country of origin and importance 8. Created an event model for saving the events into the database 9. Implemented a serializer for the event model 10. Implemented a regex based parser that fetches the events from investing.com, parses them and saves them in the database 11. Implemented an internal endpoint that runs the events parser 12. Implemented a public authorization class (ie, all allowed) that was need for some of the new endpoints 13. Implemented Google Maps based location selection (with marker) in the frontend sign up form 14. Implemented a utility function that creates a meaningful, detailed human-friendly error message using the error object returned by the backend

	15. Implemented some improvements for the sign up form (such as using a calendar based date of birth selection and showing meaningful error messages) 16. Co-implemented the feature that shows a selection of news articles on the home page in the frontend 17. Fixed some of the styling problems in the frontend 18. Reviewed pull requests
Barış Zöngür	1. Done the restructuring of the front-end as first implementation was not suitable for bigger projects. 2. Implemented main big picture structure for routing and component placement and placed routing for all components accordingly. 3. Implemented front-end follow profile functionality and design 4. Implemented search functionality for users. 5. Added followings and followers pages reached from own profile page 6. Added own articles to the Navbar's dropdown menu and profile. 7. Implemented and designed like, dislike and number of likes for articles 8. Implemented and designed like dislike and number of likes for comments. 9. Restructured and designed the trading equipment page to final functionality as selecting parities and returning the corresponding page. 10. A lot of bug fixing of all pages for more than two full days. 11. Implemented and designed other users profile pages as seen when a logged in user goes for another person's profile 12. General functionality of profile pages. 13. Ordered the folder structure for main code for readability. 14. Some design choices as what to render when for example showing person's profile after log-in.
Ali Özden	
İsmet Dağlı	1. Since the mobile team was 2 people and the increasing necessity for mobile, I transferred the mobile team and started to learn Kotlin 2. I added follow and unfollow functionality for users. 3. I added logout functionality. 4. I added the fix when you click the back button on Android, it logouts and exists the app. So, the app remembers the login credentials. 5. I enhanced register page screen visuality. 6. I added go events shortcut in the navigation bar and connection activity 7. I created the page of the details of an event. 8. I changed event fragment in the bar 9. I created an adapter in order list events, and also created a connection

	with the back-end so that we can call all existing events. 10. I created floating action buttons and those actions through each trading equipment fragments'. 11. I created a search screen for trading equipments. 12. I connected these fragments with the same search screen via click event. 13. I created a trading item to print search results. 14. I connected with the backend those equipments. 15. I created necessary back-up database info for mobile demo and helped for demos. 16. Since Volkan implemented lots of functionality and created lots of pull-request, it took quite a long time to test them in a piece by piece. I helped many of them.
Volkan Yılmaz	1. I determined and then removed Android deprecated function calls and signatures. 2. I organized user profile page and created a more efficient layout. Moreover I added edit profile page and connected with backend to make changes in email and birthday. (Name and surname cannot be changed.) 3. I added animations in app opening activity and smooth transition between screens. Moreover, I designed an app logo in order to show more pleasant. 4. I reorganize choose user type menu to show users options in a more pleasant way. 5. I added if user login credentials are not match warning and international iban number in trader register screen . 6. I designed a navigator menu and added submenus in order to give user easy motions and to find functions in the app easily. 7. I created articles feed page, an article show page, and connected with backend to fetch results. I designed an article item to supply functionalities within an object for this particular article. I added make like-dislike or revoke like-dislike and comment features in articles. Also I implemented edit-delete options for articles. I also organize view options because only creator is able to edit or delete an article. 8. I designed a comment item to supply functionalities within an object for this particular comment. I implemented edit-delete a comment options and feed screen for comments. Moreover, I added like dislike comments below articles, forex and digital trading equipment. I also organize view options because only creator is able to edit or delete a comment. 9. I created forex feed page, a forex show page, and connected with backend to fetch results. I added line chart in show forex item screen to view its values over time. I also added zoom screen horizontally. I designed a forex item to supply functionalities within an object for this particular forex and added increase or decrease situation with respect to last closing and opening data. I added make like-dislike or

	revoke like-dislike and comment features below forex items. 10. I created digital feed page, a digital show page, and connected with backend to fetch results. I added line chart in show digital item screen to view its values over time. I also added zoom screen horizontally. I designed a digital item to supply functionalities within an object for this particular digital and added increase or decrease situation with respect to last closing and opening data. I added make like-dislike or revoke like-dislike and comment features below digital items. 11. I added make prediction layouts in forex and digital items. 12. I created a home page to show 10 forex items and 3 articles to be used more efficiently in the future 13. I created an example profit screen. 14. I helped to get Google maps api key. 15. I helped to design user follow and unfollow operations. 16. I made contributions in events feed screen with visual improvements.
Furkan Aydar	1. I arranged the alphavantage keys in our settings. 2. I created the Article model. 3. I created the Trading Equipment and the Parity models. 4. I added base Comment, TradingEquipmentComment and Article Comment models. 5. I added base LikeDislike, ArticleLikeDislike and CommentLikeDislike models. 6. I added the Current Price model in relation with parities. 7. I added Prediction model for Trading Equipments. 8. I added article, trading equipment, parity and comment views including all of their functionalities (Create, Retrieve, Update, Delete) 9. I added like / dislike functionality to the comments and articles. 10. I added prediction functionality to the trading equipments 11. I arranged endpoint urls for mentioned functionalities. 12. I created some authentication classes in permissions. 13. I created trading equipment, comment, article, like-dislike, parity, current price serializers for our models. 14. I implemented av.py which includes functions for traversing all trading equipments. This file connects to alphavantage to retrieve Forex and Digital currency data and update our equipment objects. 15. I implemented email authentication, and then fixed it to make it redirect users to just our front-end page. 16. I redesigned our navbar, changed the buttons.. 17. I added icons to our dropdown menu in navbar 18. I changed the look of the profile page image holder. 19. I redesigned our ‘Write New Article’ and ‘View Article’ pages. I added new article summary cards. Tried to make these pages more responsive. I added comment box designs to the article pages, like/dislike button designs. These comment boxes are later used on Trading Equipment pages too. 20. I redesigned our Signup and Login pages to give it more organized look.

	21. I tried to organize new layouts in the pages lacking the organization. 22. I pretty much tested all of the functionalities or features above.
Efe Önal	1. I adapted the code so that it would make good use of the local storage and life cycles. 2. I added trader sign-up and first Google Geocode api which was then changed by other group members. The previous maps functionality took the address of the user as an address and posted the coordinates to backend. 3. I helped restructuring the code. 4. I implemented the main article page which renders all articles, single article page on which one can read the full article, and write article page where one writes the article. 5. I implemented dynamic trading equipment graphs for frontend which are fed historical data that was gotten from backend and single trading equipment page with the comments. 6. I implemented the events page in which upcoming events for the next three days are rendered. 7. I implemented commenting functionalities for trading equipment and articles. 8. I implemented email verification. If the user has not verified her account, she is routed to “please verify” page, otherwise she logs in successfully. 9. I implemented update credentials functionality however all fields have to be filled, which I am going to fix. 10. Our first implementations were all based on trader users. I adapted the code so that it would work for basic users as well. 11. I fixed small bugs at all parts of the frontend. Bu makaleyi efe bey mi yazdı⚡⚡⚡⚡Yes sir yes sir
Ceren Tahtasız	1. I fixed the search and edit buttons in profile page to prevent app crash. 2. I connected the register page with profile page so that when a user signs up, they don’t have to go back and login: they’re directly logged in and in their Home page. 3. I implemented Google Maps functionality to choose location. 4. I decided visual standards with mobile and web team together. 5. I deleted unused layouts after Volkan revised the structure. 6. I created user scenarios with mobile team to show in demo.

5. Project Requirements

1. Functional Requirements

1.1. User Requirements

1.1.1. Guests

- 1.1.1.1. The system shall allow the guests to view the price of any trading equipment.
- 1.1.1.2. The system shall allow the guests to read user comments about trading equipment.
- 1.1.1.3. The system shall allow the guests to view articles and annotations in articles.

1.1.2. User Authentication and Account Management

- 1.1.2.1. Sign Up
- 1.1.2.1.1. The system shall allow the user accounts to be either "basic" type or "trader" type.
- 1.1.2.1.2. The system shall require the user to provide their name, surname, email address and location to be able to create a **basic** account.
- 1.1.2.1.3. The system shall require the user to provide their name, surname, email address, location, id number and IBAN to be able to create a **trader** account.
- 1.1.2.1.4. The system, when it requires the user to provide their location, shall require the user to provide it using Google Maps.
- 1.1.2.1.5. The system should allow the user to use their Google account to provide the necessary sign up information which are available in their Google account and require the rest of the necessary information to be input using text.
- 1.1.2.1.6. The system shall obtain the necessary sign up information using text inputs, unless otherwise specified.
- 1.1.2.1.7. The system shall allow users to change their account type when necessary information is provided. Trader user can downgrade to Basic user and Basic user can upgrade to Trader user.
- 1.1.2.2. Sign In
- 1.1.2.2.1. The system shall allow users to sign in only using their Google accounts or by providing the necessary sign in information, username and password.

1.1.3. Trading

- 1.1.3.1. The system shall, for the trading users, have a "My Investments" section which shall have the functionalities specified here (1.1.3).
- 1.1.3.2. The system shall allow the trading users to invest on any trading equipment.

- 1.1.3.3. The system shall allow the trading users to make a buy order for a specified rate.
- 1.1.3.4. The system shall allow the trading users to set stop/loss limits.

1.1.4. Tracking

- 1.1.4.1. The system shall provide each user with a "Profit/Loss" section that is private to the respective users, and this section shall have the functionalities specified here (1.1.4).
- 1.1.4.2. The system shall show both basic and trading users their profit/loss amount, in terms of the currency they choose, by the user manually entering their investments.
- 1.1.4.3. The system shall show trading users their profit/loss amount, in terms of the currency they choose, by the user's trading history on the system.
- 1.1.5. Comments
- 1.1.5.1. The system shall allow users to comment and annotate on articles.
- 1.1.5.2. The system shall allow users to comment on trading equipment.

1.2. System Requirements

1.2.1. Search

- 1.2.1.1. System shall support searching for users and trading equipments considering all the information available and filtering the search results.
- 1.2.1.2. System shall support semantic search.
- 1.2.1.3. System shall support location based search.

1.2.2. Recommendation

- System shall have a recommendation system that recommends articles or trading equipment to users based on their history (what they already follow).

1.2.3. Ranking

- 1.2.3.1. System shall allow traders to rank others idea.

1.2.4. Trading Equipment

- 1.2.4.1. System shall allow traders to trade indices, stocks, ETFs, commodities, currencies, funds, bonds, and cryptocurrencies.
- 1.2.4.2. Each trading equipment shall include many functionalities, including but not limited to: the previous close, percentage change with the previous close, amount change with the previous close, day’s range, and moving averages.

1.2.5. Communication

- 1.2.5.1. System shall allow users to follow other users and trading equipment.
- 1.2.5.2. System shall allow users to share their ideas as articles.

- 1.2.5.3. System shall allow users to comment and rate ideas of other users.
- 1.2.5.4. System shall allow users to comment about trading equipment.

1.2.6. Profit

- 1.2.6.1. System shall have a Profit/Loss section for each user.
- 1.2.6.2. This section should be private for each user.
- 1.2.6.3. System shall allow basic users to see their profit/loss amount in terms of the currency they
choose by manually entering their investments.
- 1.2.6.4. System shall allow trading users 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.

1.2.7. Portfolio

- 1.2.7.1. System shall ensure each user has at least one portfolio.
- 1.2.7.2. Portfolios shall be able to renamed, edited, shared and followed (if already shared).
- 1.2.7.3. System shall allow users to add any trading equipment to their portfolios.

1.2.8. Notifications

- 1.2.8.1. System shall allow users to set alerts for certain levels of trading equipment.
- 1.2.8.2. System shall notify users in accordance with their alerts.

1.2.9. Account

- 1.2.9.1. System shall be provided necessary information for signing up. (Basic users are expected to provide their name, surname, email address, and location.)
- 1.2.9.2. System shall have specific location of users on Google Maps.
- 1.2.9.3. System should use Google account to retrieve necessary information for signing up/in.
- 1.2.9.4. System shall offer basic functionality to users after signing in and validating their email address.

1.2.10. Event

- 1.2.10.1. System shall have events section for users which contains economic events.
- 1.2.10.2. Events shall have significance level and country base properties.
- 1.2.10.3. Events shall be able to searched, filtered and chased.

1.2.11. Deployment

- 1.2.11.1. System shall be able to handle native web and native mobile (Android) clients. (Hybrid applications are not allowed.)
- 1.2.11.2. System shall be deployable on a remote and manually configurable server. (Amazon EC2 or Digital Ocean are strongly recommended.)

1.2.12. Standards

- 1.2.12.1. System shall support W3C Web Annotation Data Model.
- 1.2.12.2. System shall follow W3C Web Annotation Protocol.
- 1.2.12.3. System shall follow the standards introduced by the World Wide Web Consortium (W3C).

1.2.13. Ethics and Legal

- 1.2.13.1. System shall handle personal information, contact information, copyrighted contents and license issues according to legal requirements.

2. Non-Functional Requirements

2.1. Accessibility

- 2.1.1. The system shall be accessible on Android and Web.
- 2.1.2. The system shall be compatible with at least Chrome 70, Firefox 64, IE 11 or higher version.
- 2.1.3. The system shall be compatible with Android 5.1 or higher version.
- 2.1.4. The system shall have responsive UI for different screen resolutions for both Android devices and web browsers.

2.2. Availability

- 2.2.1. The system shall have at least 97% uptime. (Except during maintenance or unexpected errors)
- 2.2.2. The system should be under maintenance every Tuesday between 2.00 am and 2.15 am regularly.
- 2.2.3. If there exists an ongoing maintenance, the system should inform users about the estimated termination of the construction at least one day before maintenance.

2.3. Reliability

- 2.3.1. The system should generate a proper log file which includes server reports, error logs and user activities when an error occurs.
- 2.3.2. The system should create backup file everyday.

2.4. Security

- 2.4.1. The system should remind users to change their password after using it for 1 year.
- 2.4.2. The system should require passwords to contain number of characters between 8 and 20; must contain at least one lowercase letter, one uppercase letter, one numeric digit, but cannot contain whitespace.
- 2.4.3. The system should not allow user to login for 10 minutes after 5 unsuccessful login attempts.
- 2.4.4. The system should session expiration if a user is inactive for 5 minutes.
- 2.4.5. The system should have human verification tool (reCAPTCHA) to prevent bots.

2.5. Privacy

- 2.5.1. The system shall secure the private data of users according to Information Privacy Law.

2.6. Database

- 2.6.1. The system shall encrypt personal data of users with RSA before storing them in the database.
- 2.6.2. The database architecture should have trigger functionality in the case of unexpected network issues in the middle of value exchanges between users.
- 2.6.3. The database architecture should handle 97% of the queries less than 0.1 second.

6. API Documentation

Our API documentation can also be viewed at
https://documenter.getpostman.com/view/8736871/SVzuZgBL?version=latest

7. Project Plan

7.1 Backend Project Plan
7.2 Frontend Project Plan

	30.9.2019-6.10.2019	7.10.2019-13.10.2019	14.10.2019-20.10.2019	21.10.2019-27.10.2019	28.10.2019-3.11.2019	4.11.2019-10.11.2019	11.10.2019-17.10.2019	18.10.2019-24.10.2019
Frontend								

1.General React Study	✓							
2.Initialize Frontend		✓						
3.Create Main Application Layout for Different User Types		✓						
4.Develop Login and SignUp Pages			✓					
5.Develop Profile Page			✓					
6.Establish communication with the API				✓				
7.Develop Following Functionality				✓				
8.Implement Trader SignUp					✓			

9.Restructuring the code using routing library						✓		
10.Implement Article Functionalities							✓	
11.Implement Trading Equipment Functionalities								✓
12.Implement Event Functionalities								✓
13.Implement Commenting Functionality								✓
14.Implement Like/Dislike Functionality								✓
15.Implement email verification								✓

16.Develop rate functionality for trading equipment								✓
17.CSS improvements								✓
Backend								
1.Initializing the app	✓							
2.User model and serializer		✓						
3.User CRUD endpoints		✓						
4.Authentication			✓					
5.Permissions classes			✓					
6.User follow-unfollow features				✓				

7.3 Mobile Project Plan

	30.09.19 - 06.10.19	07.10.19 - 13.10.19	14.10.19 - 20.10.19	21.10.19 - 27.10.19	28.10.19 - 03.11.19	04.11.19 - 10.11.19	11.10.19 - 17.10.19	18.10.19 - 24.10.19
1.General Kotlin and Android Study	✓							
2.Initialize Mobile	✓							
3.Develop SignIn and SignUp Layouts		✓						
4.Develop basic user and main activities		✓						
5.Establish communication with the API			✓					
6.Sign in and sign up functionality			✓					
7.Develop show profile page activity and functionality				✓				
8.Develop edit profile page activity and functionality				✓				
9.Determine and Remove deprecated functions					✓			

10.Implement Article Functionalities						✓	✓	
11.Implement Commenting Functionality						✓	✓	✓
12.Implement Forex and Digital Functionality							✓	✓
13.Implement Like/Dislike Functionality							✓	
14.Home Feed Screen								✓
15.Implement Event Functionalities								✓
16.Mobile visual improvements								✓

8. User Scenarios Used in Presentation

8.1. Frontend Scenario

Frontend Scenario - 1

Ahmet Kazan is a retired person who wants to make investments with his retirement bonus. But he does not know much about investing but he wants to learn.
He hears about Mercatus App which provides article functionality from his friend Ali Uzun and he decides to create an account.

Actions:
1. Ahmet Kazan signs-up with as a trader user.
2. He tries to login but email has to be verified.
3. He verifies his email and then logs in successfully.
4. He clicks on parities.
5. He selects a USD/TRY.
6. He reads the comments of the parity and likes a comment.
7. He goes to the profile of the commenter by clicking on her name.

8. He follows her and clicks articles to see her articles.
9. He reads one of the articles and he gets really impressed.
10. He likes the article.

Frontend Scenario - 2
Can Uzun is the son of a successful businessman.
He does not work but in time he got very good at investing.
1- Can logins for the preregistered account.
2. Can looks for upcoming events.
3. He sees an event that takes place in Japan.
4. He goes to the trading equipment page to see JPY/TRY.
5. He writes an article about JPY/TRY.
6. He goes to his article and leaves an explanatory comment.
7. Then he searches for his friend Ahmet to whom he told about the app. He wants to see if Ahmet created an account.
8. He follows Ahmet and goes back to his profile to check that the following was successful.
9. Can is happy now.

8.2. Mobile Scenario

Mobile Scenario -1
Mehmet Yerli
Mehmet Yerli works in a banking company and wants to learn more about current economic updates. He has already installed Mercatus. In his leisure time, he reads lots of articles and blogs.

Preconditions:
Mehmet Yerli is a registered trader user.

Actions:
1-Mehmet Yerli opens Mercatus and goes with the articles section.
2-Mehmet Yerli reads several articles and he likes articles from Sevim.
3-Mehmet Yerli opens a particular article and make comments below it. Then he clicks the author to follow Sevim.
4-Mehmet Yerli opens the articles page again and finds a relatively bad article. He disliked this article and shows Taylan profile to unfollow her.
5-Mehmet Yerli log out from Mercatus after reading articles.

Mobile Scenario -2
Sevim Sevgi
Sevim Sevgi works in a stock exchange in Wall Street. She likes writing articles and continuously she views the forex market.

Preconditions:
Sevim Sevgi is registered trader user.

Actions:

1-Sevim Sevgi logins Mercatus and goes with the articles section. She likes Mehmet's comment and follows back to him.
2-Sevim opens home to update. She clicks the GBP USD forex item to investigate. She thinks forex will increase and makes a prediction.
3-Sevim tries to zoom in the chart but cannot reads with his poor eyes. She clicks to the zoom view.
4-Sevim log out from Mercatus.

Mobile Scenario -3
Furkan Aydar
Furkan Aydar is a retired businessman from a construction company. Since he is old he keeps entering wrong information. (he is that old)

Preconditions:
Furkan Aydar is a registered trader user. He follows Mehmet.

Actions:
1-Furkan Aydar logins Mercatus and goes to the events section and views some events about US dollar sales.
2-Furkan opens forex screen and click search for equipment search. He enters USD and finds GBP USD equipment.
3-Furkan makes a prediction to decrease.
4-Furkan searches one of the friends from the apartment he stays to follow Sevgi through home section.
5-Furkan notices his birthday is wrong from what Sevgi said thus he wants to change his birthday. He opens the Profile section.

9. Git Workflow

9.1. Overall Structure

We used the following Git Workflow in our development process:

Branch Name	What For	Description	Example
master	Development	Used for development, accepts merges from feature or hotfix branches.	-
stable	Stable	Is stable, accepts merges from master or hotfix branches.	-
feature-{f|b|m}{no}	Feature	For new features, must be checked out from master.	feature-b121
hotfix-{f|b|m}{no}	Hotfix	For fixing release bugs, must be checked out from stable.	hotfix-m163

Note that we do not have a stable branch yet because our project is not stable enough right now. As soon as we decide that it is stable, we will create the stable branch from master.

9.2. Details on Branches

1. **master** branch is used for development. It accepts merges only from **feature** or **hotfix** branches (except real urgent situations).
2. **stable** branch is composed of commits that are merged from **master** and are tested/stable. It can also accept merges from **hotfix** branches to fix bugs.
3. **feature** branches stem from **master**. These branches correspond to a new feature issue (e.g. branch **feature-f123** is created for a frontend feature issue with id 123). These branches must be deleted after merging.
4. **hotfix** branches stem from **stable**. These branches correspond to a bug issue (e.g. branch **hotfix-m142** is created for a mobile bug issue with id 142). These branches must be deleted after merging.

10. Evaluation of the Tools and Managing the Project

For the backend application, we had initially decided on using Java Spring along with MongoDB as the database engine. However, later we noticed that the conceptual model structure and the needs of our application was better suited to using PostgreSQL as the database engine. After implementing some features and endpoints using this stack, we observed that Spring was introducing a significant amount of boilerplate code and was more complex, which made it unnecessarily slow-to-develop for a project of this scale. As a result, we migrated the application to Python Django. Now, we are quite satisfied with our choice of technologies and frameworks. Django, along with the REST extension and a couple of more extensions has made the development really easy and fast for us.

For mobile, we decide to use Kotlin and for IDE we use Android Studio. For milestone 1, we were a group of 2 people for mobile application since initialization of backend and work relatively a bit more than other platforms. Even though we don't have any experience with Kotlin, we deliver our product as planned.
Kotlin is rapidly gaining popularity among android developers. It is native mobile language and it has lots of advantages over Java. Java is 20 years old and since each new version should compatible with old versions, it prevents Java to progress more. Kotlin is however very young language and interoperable with Java. Same tasks can be done with less effort and with short code blocks. Kotlin compiler is more advanced and safer. Moreover, a lot of IDEs support Kotlin language now.
Android studio performs well in terms of testing, debugging and run the application. You can connect with github and you can achieve a lot of things from IDE.

For frontend, we used reactJS since it is easier to create a web application with HTML and JS functionalities together. reactJS's component functionality is helped us to manage our project and work cooperatively and also saved us from code reuse. After project initialization we set a meeting to decide our coding standards and make project management plan. Since we set 4 meetings, it was easy to manage project. While deciding which parts we were going to implement before milestone 1, we focused on functionality and skipped static components for being clear to our customer about application's current state. We had a little hard time sending requests to our api since none of frontend team members have experience on this concept. In our meeting about api with backend team we solved that issue. We used axios for sending requests to our api and react-bootstrap (which is a js library) for our styling.