CmpE 451
Fall 2020
Milestone 2
Report

Group 8
Afra Arslan
Ahmet Yiğit Gedik
Burak Berk Özer
Göksu Başer
Muhammet Yazıcı
Nazım Berke Metin
Nursima Çelik
Onur Enginer
Öncel Keleş
Özgürcan Öztaş
Yasin Kaya

January 5, 2021

Contents

1 Executive Summary of Milestone 1:

1.1 Project Summary

Our project is an e-commerce application, where people can sign up as a customer or vendor and buy or sell stuff online. Apart from that, people can also use the app as a guest user.
Customers and guest users can search products, add them to their shopping cart and buy them with credit card information given that they provided an address. Then they can see the details about the order and may return the product or request missing/flawed parts. Customers also can put products on their watchlist, save their payment and address information.
Given that the provided a company name, location and contact information Vendors can put products on their pages and sell them to other users. They can also message a customer back, if that customer purchased a product from that vendor.
Apart from that, the system includes an admin profile. The admin users can see, edit and confirm changes about product and also message with both the Customers and the Vendors.

1.2 Project Status

For this milestone, we finished most of the crucial functionalities for our product as indicated in our milestone. These consist of: product/product page, admin panel for web, vendor page, shopping cart, shopping lists, order/purchase. We have also revised some of our previous functionalities such as login and sign-up.

1.3 Planned Changes

We moved the implementation of notifications to our next milestone, where we will also polish our shopping lists implementation and integrate the notifications functionality with it. We also moved the implementation of activity streams to our final milestone.

2 Status of the deliverable:

Deliverable	**Status**
Profile page	DONE
Admin Panel	DONE
Product	DONE
Search	DONE
Activity Stream	NOT YET
List	HALF-DONE
Notification	NOT YET
Shopping cart - order - purchase	HALF-DONE
Comment and Rate	HALF-DONE

Table 1: Status of Deliverable for Front-end

Deliverable	**Status**
Profile page	DONE
Admin Panel	DONE
Product	DONE
Search	DONE
Activity Stream	NOT YET
List	DONE
Notification	NOT YET
Shopping cart - order - purchase	DONE
Comment and Rate	HALF-DONE

Table 2: Status of Deliverable for Backend

Deliverable	**Status**
Profile page (Customer)	DONE
Admin Panel	REMOVED
Product	DONE
Search	DONE
Activity Stream	NOT YET
List	HALF-DONE
Notification	NOT YET
Shopping cart - order - purchase	HALF-DONE
Comment and Rate	HALF-DONE

Table 3: Status of Deliverable for Mobile

3 Evaluation of the deliverable:

3.1 Web

3.1.1 Login-Signup

At the beginning of this milestone, we've redesigned our login and signup pages. We came up with more user-friendly and consistent design with the rest of our app. Also, we've added Google Maps for the Vendor Signup Page where they can select their locations.

3.1.2 Home Page

We've redesigned our homepage as well. Firstly we've decided to use carousels in our homepage instead of putting all the products in a matrix form. We also added banner carousel to have more realistic design. On the other hand, we were using mock/static data in the homepage. With the implementation of the products, the homepage currently is connected to the backend side and shows the added products.

3.1.3 Profile Page

We've implemented the both vendor and customers' profile page. They both able change their information other than their email addresses. In addition to that, vendors are able to change their locations using the Google Maps, and customers are able to add credit cards or addresses which they may use during the purchase.

3.1.4 Admin Panel

We've implemented the Admin Panel, and it will be updated with future features. Admins are now able to suspend/delete users, see the product requests and accept/decline them. Also, we will add functionality to remove comments if they found to be inappropriate, or messaging systems to be able to communicate with users.

3.1.5 Product

We've implemented the Product Page, Add Product Pages. First of all, in the product page, one able to see product's photos, details, vendor, etc. Also, one may able to buy from another vendor, or navigate to details of same product with different attributes in the same page (this different product does not necessarily should be from same vendor). Lastly, one can read the comments made to this product.

In the vendor side, vendors are able to create Main Product giving the Main Product's title, attributes, tags, category, etc. Then, they could either create a new Product (giving its photos, price, etc.) using one of the created Main Products, or add their name to the already existing product giving the price

they want. To create a new product, they have to make a request to the admins though. Only if their request is accepted by the admins, their product will be shown to the users.

3.1.6 Search

We've implemented the search functionality as well as the filtering functionality. One can filter the search results according to the vendors, product specific parameters (color, size, etc.), price. Also, one can sort the results according to price.

3.1.7 List

We are currently polishing the list functionality of our application. It will be available in a short time.

3.1.8 Shopping Cart - Order - Purchase

We've designed and added the pages using mock data. But we are still working on to connect the pages to the backend. When it finishes, users will be able to add products to their cart, purchase them using their credit cards/addresses and see their order history in their page. We will also inform the vendors about the order.

3.1.9 Comment and Rate

We've implemented the comment functionality, but currently it does not check whether the user that wants to comment has ordered the product or not. It only checks whether the customer has signed in or not. We will add the check. Also, we are not able to rate right now.

3.2 Mobile

3.2.1 Login-Signup

We revised the signup according to the updated endpoints. Everything about login and signup is done.

3.2.2 Home Page

We had a home page in the first milestone, but since then we made some changes. We made the backend connection so that products that are shown on the home page are retrieved from the backend. This also enabled us to show the same products from different vendors (possibly with different prices, etc.) as different entries on the home. Users are able to see different deals at once. The appearance of the home page is also improved. We added a banner to welcome users. Products are chosen more carefully, to spread good vibes and energy. In conclusion, the status of the homepage is aligned with our project plan.

However, our homepage is still open to improvement. We discussed adding numerous things: more banners in a sliding fashion, add to cart option for a product in the home (without having to go to the page of the product), "like" button on products. Furthermore, for the final milestone, we will implement a recommendation system. We consider showing recommended products for customers in the homepage.

3.2.3 Profile Page

In the profile page, we created different pages: User Information page for editing user information (name, surname, number and birthday). Change Password page for changing password after old password confirmation. Addresses and Credit Cards page for Addresses and Credit Card CRUD operations. Logout option to be logged out from backend and frontend together. Also About and Contact page for hard coded text pages for our project, group and contact information. Also there is My Lists option to access user's List CRUD operations explained in detail below. Last but not the least, Legals page to be formed in the Milestone 3 for KVKK and GDPR explanations to the user.

3.2.4 Admin Panel

After a discussion with our customer, we concluded that there is no need for an admin panel in the mobile app.

3.2.5 Product

We added the product page feature. Users can go to a product's page by clicking on it. In the product page, various properties of products can be seen: title, photos, rating, price, vendor name, description, and user comments. Also, users are able to add the product to cart/list; specify quantity while adding to cart; and add comment/rating.

Currently, different vendors of the same product are not shown on the product page, we plan to add it as soon as possible.

3.2.6 Search

Search, sort, and filter functionalities are implemented and the status is aligned with the plan. We have the minimum necessary UI elements: search bar, a place to show products, sort, and filter buttons. We connected all to the backend.

3.2.7 List - Notification

Shopping lists are initially implemented as in requirements, with two default lists named "Favourites" and "Watchlist". All the functionalities are completed. Customers are able to create/delete them, and add/remove products to/from their lists. Furthermore, when a user wants to add a product to a list, they have

the option to create a new list. As soon as they enter the name and click OK, the product is added to the new list automatically.

However, lists are not yet connected to the backend.

About notifications, we wanted to prioritize making other features fully functional first. We plan to complete notifications by the next milestone.

3.2.8 Shopping Cart - Order - Purchase

Shopping cart, order and purchase functionalities are implemented as planned. Users are able to add products to their cart, along with the quantity information. In the cart page, they can change the product's quantity, remove products from the cart. Also there is a heart-shaped button that will enable users to add products to their "Favourite" list. With the purchase button in the cart page, users can proceed to the payment where they choose an address and credit card (these information can be added via profile page).

Currently, address and credit card cannot be added in the process of payment, because the "purchase" endpoint expects an already created address and credit card. Also Agree to Terms & Conditions button is to be added. We will resolve this issues by the final milestone.

3.2.9 Comment and Rate

Comment and rate functionality were initially implemented as displaying a comment array belonging to the product, where a comment has fields of rating, user name, and comment body. So, UI is done. However, the backend implementation of comment did not have a rating field and it returned user id's instead of user names, so, it has not been integrated to Android fully yet.

3.3 Backend

3.3.1 Login-Signup

The former signup pages didn't have all the fields written in the requirements, we added those fields in the endpoints.

3.3.2 Profile Page

We created endpoints for profile pages of vendors and customers. Users can update their profiles.

3.3.3 Admin Panel

We created endpoints for customer and vendor operations for admin. They can suspend/delete users. We also added endpoints for product requests, admins can see the requests and either accept or decline them. For the final milestone there will be other functionalities such as messaging and removing inappropriate comments.

3.3.4 Product

We created three models for product operations. They are: main product, product and product request. Main products have parameters. Example parameters could be "size": ["XL","L"], "color": ["blue","red"]. For each parameter combination a product can be created. Each product can have multiple vendors and vendor specific information. Vendors are able to create main product giving its title, parameters, tags and category. They can also create a new product or they can add themselves to an existing product. They can try to update or delete their products. All these actions create a product request and goes under the supervision of admins.

3.3.5 Search

We created two endpoints for searching: search and search filters. Search filters endpoint takes a query string in the body and returns the filterable fields for that specific query. Users can use those fields to filter their search. Search endpoint also takes a query string in the body and filtering, sorting, paginating and limiting options in the query parameter.

3.3.6 List - Notification

We added endpoints for manipulating lists. Users can create new lists, add and remove products from their lists. They can also export their lists to shopping cart. We preferred not to implement the notification in this milestone to focus on other features.

3.3.7 Shopping Cart - Purchase - Order

We added endpoints for adding and removing products from the shopping cart, purchasing of the shopping cart and executing the order. Users can increase or decrease the number of products in the shopping cart along with the option of removing the product from the shopping cart. Then, users can buy the products in their shopping cart and proceed to payment stage.

3.3.8 Comment and Rate

We added an endpoint adding comments and rate to the products. Customers can add comments and rate the products they bought. They can also change their comment and ratings. Admins can delete the comments they find inappropriate.

4 Summary of the Coding Work Done:

Name	Coding Work Done
İbrahim Özgürcan Öztaş	In Milestone 2, I've implemented commenting functionality in backend side of our project, which is a crucial part of our product. While integrating products in our system, comments should not be outside of the scope. In commenting operation, GET, POST, PATCH and DELETE functionalities are implemented and merged with our products. After commenting, rating should not be forgotten, so I've implemented the rating functionality as a generic approach, not unique to each comment, therefore users rating are not shown in our project right now. After a discussion, we've decided to change it to user-specific, therefore all guest or customer users can see the unique comments and rating of any customer in our environment. Furthermore, I've implemented lists in our project which is a functionality of collecting different products into one collection to refer or import into cart in a future time. It was a crucial task to accomplish, and I've done it with minor glitches. I've been assigned to implement activity stream and notification in Milestone 2 right after Milestone 1, but we've changed our internal planning to show them at the final milestone.
Afra Arslan	I've made improvements to Google Sign in and integrate it to our new home page. Before we started our new goal Profile page, I've built an Account page structure. I placed a sidebar into the account page so that users can navigate through to user account pages. Yasin and I took the job of the user profile page in the account. The profile info section contains two forms, a user information form, and a password change form. A user can change his/her personal information with this form. In the reset password form, I used the one that I used in the password change. For the search feature, I added a filter bar that users can filter their search. When the API request was ready, I integrated it to the front-end side so that if users search for a product, it navigates to the search page, and search results appear on the content side. I split the vendor account pages from the customer account and built the vendor account separately, as I did earlier for the customer. Öncel and I have added the feature of adding products to the system by vendors. I did the main product part of product adding. For the list and cart features, I designed the UI part. However, the API requests were ready the day before the Milestone. Therefore, we could not test and deploy them. Other than these, the backend time had to change the API base URL. On the web, I deleted the URL from all requests in our project and added the new URL to just one page. With this, we can use it from that page and edit it easily.

Name	Coding Work Done
Yasin Kaya	Firstly, I reimplemented the homepage as it was not consistent with the rest of our application and was not so user friendly. Then, I started to implement the customer profile with Afra. I implemented the pages to add/remove/update addresses and credit cards and made sure that they are working with our backend. Then, I've integrated the Google Maps to our system so that vendors are able to select their locations during the registrations/from their profile page. Also, while adding Google Maps integration to the vendor's profile page, I integrated the backend connections for the vendor profile page. Then, I've added photo carousel to the product page and made sure that all product info is retrieved from the backend in the remaning parts of the product page. Then, Afra and I worked on the Search Page. She retrieved the filter parameters, and constructed the filter according to the parameters. Then, I've connected the filter to the backend to be able to filter. Also, I've added the products that will be shown in the search and added functionality to sort them according to their price. Lastly, I've created an S3 bucket. Then I implemented functionality that whenever a vendor uploads a photo of the product, uploads the photo to the bucket and a publicly visible URL is obtained.
Nazım Berke Metin	I have implemented a global api caller class in order make api calls easier for the mobile team. This class handles json serialization and deserialization. Also handles the error and success messages returned from backend. Developer only have to give response and request types to apiInterface. Last, it also handles the authorization. Second mission was revision of the vendor and customer signup pages. There was a negative customer feedback on the first milestone for the vendor signup so, I revised those pages after backend added parameters for vendor signup. Made some graphic updates also so I revised the customer pages to made them same with the vendor. Last, I designed vendor dashboard that will automatically open after vendor login and return to customer dashboard when logout. I implemented the vendor homepage that shows vendor products and show product page on click to see comments and product itself. Added stock update button but did not made backend connection for stock update. Also made a last minute merge for a PR that has lots of conflicts.

Name	Coding Work Done
Onur Enginer	I have created the product class and reimplemented home-page using dynamic product objects instead of static ones in Android. Then I designed a simple product page for products which consists of a photo, price, vendor name, product overall rating; which takes the average rating from all the reviews from review section in product page, where a user enters a rating 1-5, adds a comment and their names are shown near their comments. Then I implemented list functionality, where users have 2 default lists named favourites and watchlist, and they can create new lists and name them, delete them except for the default ones, add all items in list to cart and remove items from them. Then I have created shopping cart, where a view of products with title, price, photos and amount of each item in cart is shown, where user can collapse this view and just see the total cost of products too. Users can delete products from this page and each item has a heart outline button which is supposed to add the item to favourites list. I implemented purchase page where it is directed from purchase button in shopping cart page, where a user can select a previously saved address or credit card and also enter new values for these fields. I have not shown the full credit card number on saved cards there but only last 4 digits. I also added add to cart and add to list buttons in product page and an amount counter which indicates how many of those items will be added to cart if the add to cart button is pressed. After the backend counterparts of these features were available, I went on to reimplement most of them. Product, shopping cart, purchase were mostly working but comments did not have ratings and lists did not have backend connection in the end.
Ahmet Yiğit Gedik	Firstly, I have moved to implement the category section of the admin panel. In the category section, the admin of the web application can create new categories, or delete a category. After this task, I have begun to implement the information section of the product page. The information section consists of three components. These are features, comments, and other sellers' section. The feature section is to show the characteristics of the product. Other sellers section of the product to indicate other sellers of the same main product and the last component is the comments section to show comments of the main product. Besides these coding tasks, I have also got a domain name carousel.ml and deployed our web application on to this domain. The last task that I have tried to do is adding SSL to our server to provide a secure connection. I have done this task successfully but we could not use this feature because our backend server had not an SSL certificate.

Name	Coding Work Done
Nursima Çelik	Created and added banner image for the home. Added icons for search and categories in the bottom menu. Created the search fragment and search layout. For UI of the search page, added a searchview, sort button, and filter button. Added a recyclerview so that retrieved products can be seen as the content, in scroll-able manner. For the sort button, I researched and learnt how to create a pop up in Android. Then, added a pop up in the oncliklistener of sort button. When user clicked on sort, a window would pop up. But then I changed it to a spinner as adding onclicklistener to an item in a spinner looked easier. Added an array adapter for the spinner. Also, to make the sorting functional, I added a mapping from each button's text to the corresponding field by which we like to sort. While constructing filter, researched and learnt how to implement a right drawer. Placed the content of filter page in the drawer as a scroll view and made sure it opens when filter button is clicked. Filled the drawer with filtering options. Added various topics like Price, Rating, Color, Brand, Category, etc. For each topic, made its content to be expandable/collapsible. Made content of Price and Rating is hard coded, since they are independent from the query. Added two edittexts to Price, one for min price and one for max, also check boxes with price ranges. When user clicks one, the range appears in the edittexts. For rating, I added 5 check boxes. Other filters are retrieved from the backend. For each topic retrieved, I added its content programmatically in the Kotlin file. We had /product/search and /product/searchFilters endpoints. I made the backend connection for these endpoints. For the category page, I created category fragment and layout. Got the categories from backend and showed them with related copyright free photos that I found on the net. For milestone, we used a different port with clean database. When I switched to that port, our home crashed. So, I fix this bug by changing structure of Product.

Name	Coding Work Done
Muhammet Tayyip Yazıcı	Our client model held both the customers and vendors, we thought it would be confusing for the coming features. So, I split the client model into customer and vendor models. I created the related dataAccess, router, controller and service files. I also transformed the client model into a simplified version of the old one. All the fields that are common in users are kept in the new client model and new models inherit from it using Mongoose's discriminator functionality. Using Express' middlewares, I added a protection controller to check user authorization. The scripts for automatic deployment for backend were finished last milestone but they were not merged. After doing some tests I finally merged the workflow with the backend, the workflow gets triggered after any pushes to backend branch. I created the admin model and necessary endpoints for admin panel operations, namely CRUD operations for vendors and customers. I created mainProduct, product and productRequest models. I created necessary files in dataAccess, controller, service and router folders for these models. In this structure, vendors' actions on products are supervised through usage of productRequests by admins. This structure also supports multiple vendors per product and multiple options for mainProducts, this way vendors will have a lot of flexibility on their products. After the products, I added two endpoints for searching, they are: /search and /searchFilters. The searchFilters endpoint returns the filterable fields dynamically for each query. Search endpoint can filter fields returned by searchFilters for the same query string. The endpoint also can sort, paginate and limit the number of results. Along the way I dealt with the maintenance of the backend app and fixed some of the bugs. I also wrote a similar script for web auto deployment using Ansible and Github Actions, however the workflow doesn't show up in the Actions page, it is yet to be solved.

Name	Coding Work Done
Göksu Başer	I was responsible for profile page implementation. I have designed and written code of profile page and pages accessible pages from profile page except My Lists tab. I was not really good at fragment usage in Android. It was a challenge for me. Also retrieving information from back-end and displaying it was complicated for me because code was not working exactly in a linear manner but making leaps according to the back-end returns.
Öncel Keleş	First, I redesigned our login and signup page in order to present it with a better user experience and fit it to our app's overall design. After that, I started working on admin's user account panel where the admins could delete or suspend accounts. Then I also made adjustments to the design kit we use - antd, so that each component use our selected colors. I also added a 404 not found page so that we cover incorrect urls any user enter. I designed the product page for the app without the endpoints, so that we have the skeleton of the page where we can build upon when we got the endpoints and developed other necessary components regarding the product page. Then, I together with Afra, changed the header and routings so that vendors can reach vendor functionalities only. Then, I worked on vendor accounts' add product part with Afra. I covered the main and child product tables and adding new child product and adding vendor to an existing product functionalities. Then I added "My Products" page for vendors where they could delete or edit their products. After that, I implemented "My Product Requests" for vendors, where they can see their pending requests and delete them if they like to. I finished the admin's product panels after the vendor parts, where admins could list products and delete them, or list the product requests and confirm on decline them when necessary. Then I implemented only the design part of the customer order pages, and fixed some little design bugs related to visual aspects overall.

5 Integration Tests:

- **Current Status:** We did not created our complete test suite for now. We prefer to test with ourselves, but until the final milestone, we will have our integration tests as automated.

6 Challenges on the road of Milestone 2:

Name	Challenges encountered while coding
İbrahim Özgürcan Öztaş	In Milestone 2, I've encountered with several issues regarding with the MongoDB mainly. For my list implementation, I need to learn how to use MongoDB database features and functions to retrieve or search any document with different specifications one after another or separately. Therefore, I've spent plenty of time to understand the dynamics and documentation of MongoDB and adapt them into our project. For me, cascading retrieved documents one after another was harder than a SQL database in MongoDB, since I was taught SQL based databases in previous years and therefore I didn't need any NoSQL database approach up until this course. Apart from these, time management and the ratio of workload/time interval was so great that I've felt unfairness to be honest. Our project is almost a complete replica of any global e-commerce app around the world, yet the gain from the project is so little compared to the time and effort I provide, therefore I do not feel a great amount of motivation.
Afra Arslan	Since we had several challenges during Milestone-1, we started to develop new features as earlier as possible. But there are some review needed features like Google and Log-in/sign-up pages. The time issue is always on the stage since it is two and a half months school project. The last day was rough again. Unfortunately, there were some features we could not make it until Milestone-2.
Yasin Kaya	I was more confident about my implementations skills during this milestone, but time management was an issue for me. First of all, as a web team, we started to this milestone by re-implementing the most of the features that we did before the milestone-1. Therefore, we spent some time on the features that we are not planned at the beginning (it was necessity). Also, while some of the features were taking more time than we expected, some of the features were taking less time than we expected. Therefore, I struggled to allocate right amount of time to the project. While some days I needed to work up to 10-12 hours, there were period of times that I didn't have work to do.
Nazım Berke Metin	I didn't have any difficulties during coding. Finding motivation to give effort to this project was the real challenge, I don't why but doing anything for this project is makes me feel bad. Anytime I start coding something, I have to focus on quantity rather then quantity in order to finish project in narrow timeline. This takes any fun out of this project and kills the motivation because you know that you are not doing great.

Name	Challenges encountered while coding
Öncel Keleş	The main challenge for me was implementing the panels for admin and vendor, as they were much more complex than I expected. Because of the way we move on with our product structure, I needed to implement nested get requests in the panels, which made it harder for me to trace correct data I needed and pass them on different methods and components. Also due to the limited time we had, the code I wrote was redundant in some parts and I did not have the time to reduce these because I needed to move on with other parts of the project. The design parts were much easier compared to the first milestone as I felt more experienced. Other than these issues, receiving some of the necessary apis very late caused some of the features on our(web) side to be incomplete because we needed to do the deployment and bug fixes before completing the half-done features. I think losing 1 member from the backend team did hurt us after all.
Nursima Çelik	First challenge I encountered was imagining how home page should look like. Certain things were for sure, like a search bar, preview of some products (with image, title, and price). But I had to decide whether there should be a banner at the beginning, under which titles we should present the products, which products I should use this milestone, whether there should be some brands at the beginning to get the attention of the user, etc. Also for a good impression we would need to come up with nice colors. After deciding what should be in the home page, I tried to realize it, but next challenge was creating such a page in Android Studio. Sometimes I spent too much time on easy tasks because of my lack of experience. Lastly, using git was not so easy, but I managed to push my code to correct place thanks to its user friendliness.
Ahmet Yiğit Gedik	In milestone 2, the hardest challenge that I have encountered is that communication with the backend team. In order to implement the frontend feature of some basic parts, I should have waited for the completion of the API part but unfortunately, there was a delay in the completion of some APIs. Additionally, there was some misunderstanding in terms of design issues hence we have lost time to implement some features. Moreover, I have encountered an unexpected situation. This was an SSL certificate that I have deployed on to the front-end server has not worked as I expected because the back-end side has not an SSL certificate.

Name	Challenges encountered while coding
Onur Enginer	First few weeks into Milestone 2 things were going smoothly and I feel like I was progressing well. However, as the product endpoints were deployed, it was quite more complicated than I implemented according to class diagram. This was a week before the milestone, which could have been enough time but all the features I had implemented before(product page, shopping lists, cart, purchase, comment/rate) were all tightly connected to my initial product class and it wasn't very possible to implement them all from scratch. So I had to integrate them to my previously created product model very inelegantly. For the rest of the endpoints which all came the last 2-3 days, it was just a terrible experience to make the connections last minute and change a lot of the things, removing features such as rating in comment as it wasn't implemented in backend that way etc. There were also very few pull requests for Android for the most part of the milestone, which was also disheartening and imposed difficulties on merging as it had too many conflicts with the main branch. Overall I don't have any of the enthusiasm about this project I had before due to those challenges, because I just don't think there can be a decent end product under these circumstances.

Name	Challenges encountered while coding
Muhammet Tayyip YAZICI	My first challenge was splitting the client model into two parts. I had to create and change quite a lot of files before I could test the endpoints. Following all the changes was quite a problem. However, my main problem was with the endpoints related to products. Initially we thought of having only one model for it. However after discussing with the customer we decided we would create the products with a greater flexibility. This caused me to redesign everything about products. I also ended up creating a lot of endpoints, it was hard for me to keep track of the interaction between them. My second biggest challenge was with MongoDB, I had only used very limited features of it and I had go deep into it in this milestone to implement a lot of functionality. Although the documentation is very well written, there are some discrepencies between Mongoose and MongoDB. This has caused a number of problems for me. The aggreation pipeline I used for searching products required a lot of knowledge about MongoDB, finding the exact operations I needed was a big challenge. On the deployment side I had already implemented the deployment scripts using Ansible and Github Actions. At the start of the second milestone we merged the commit and everything has worked out without a problem until now. However, there were some problems with our AWS EC2 instances but they were merely about our free tier usage limits. We switched to an education account and we are now freely able to create instances to our needs. I also tried to write the automatic deployment script for the web app using same tools. When I tested the script on my fork it worked without a problem but the in the main repo, Github Actions didn't seem to detect the the yaml file as a workflow, I couldn't find a solution for it yet.

Name	Challenges encountered while coding
Göksu Başer	My coding challange was learning new coding schemes in Android because in the first Milestone I used Activity pages; however, in this one I needed to learn how to code on Fragment and Adapter files. Also back-end connections was hard because the code was not exactly working linearly and making leaps depending on the back-end returns. In addition all of these in general, needed time to accomplish the task neatly was nearly impossible. I was responsible for the profile and I estimated this page would be finished in a week and planned to continue with another feature following weeks. Despite coding and studying sleeplessly 24-30 hours a week it took me 3 weeks to finish it. I want to mention that not just to whine but explain how hard was the process. I did not sleep for solid 50 hours (not exaggerated) before milestone and after all I must have make a quick scenario for the android presentation. After milestone it took solid 3-4 days for me to get back to myself and gain back my sleeping schedule.

7 API Documentation:

Our domain of API Documentation.: http://54.165.207.44:8080/api-docs/
P.S.: Our documentation is not complete yet, but it will
be when it is converted to multifile documents, since
its current situation is not manageable.
You can also import the endpoints in our POSTMAN documentation:
https://www.getpostman.com/collections/bf3f8631ccf940b001a5

7.1 Client: (/client)

Functionality	**URI**	**Method**	**Parameters**	**Return**
Login	/login	POST	password, email	tokenCode: String
Logout	/logout	POST	tokenCode	None
Verify Email	/verifyEmail	GET	verifyToken	None
Change Password	/changePassword	POST	tokenCode, newPassword, newPasswordRepeat	None
Forgot Password	/forgotPassword	POST	email, type	Check your mailbox
Reset Password	/resetPassword	POST	resetPasswordToken, newPassword, newPasswordCheck	None

7.2 Admin: (/admin)

Functionality	**URI**	**Method**	**Parameters**	**Return**
Login	/loginAdmin	POST	email, password	tokenCode
Logout	/logoutAdmin	POST	None	None

7.3 Customer: (/customer)

Functionality	URI	Method	Parameters	Return
Signup as customer	/signup	POST	email, password, passwordConfirm, name, lastName	None
Login with Google	/loginWithGoogle	POST	googleToken	None
Get all customers	/	GET	None	results: number of customers, data: Customer list
Get one customer	/:id	GET	None	data: Customer
Update one customer	/:id	PATCH	customer	data: updated customer
Delete one customer	/:id	DELETE	None	data: null
Get this customer	/me	GET	None	data: Customer
Update this customer	/me	PATCH	customer	data: updated customer
Deactivate this customer	/me	DELETE	None	data: null

7.4 Vendor: (/vendor)

Functionality	URI	Method	Parameters	Return
Signup as vendor	/signup	POST	email, password, passwordConfirm, name, lastName, companyName, companyDomainName, locations	None
Login with Google	/loginWithGoogle	POST	googleToken	None
Get all vendors	/	GET	None	results: number of vendors, data: Vendor list
Get one vendor	/:id	GET	None	data: Vendor
Update one vendor	/:id	PATCH	vendor	data: updated vendor
Delete one vendor	/:id	DELETE	None	data: null
Get this vendor	/me	GET	None	data: Vendor
Update this vendor	/me	PATCH	vendor	data: updated vendor
Deactivate this vendor	/me	DELETE	None	data: null
Get this vendor's products	/me/product	GET	None	results: number of products, data: product list
Get this vendor's product	/me/product/:id	GET	None	data: product
Create an update product request	/me/product/:id	PATCH	vendor	data: product request
Create a delete product request	/me/product/:id	DELETE	None	data: product request
Create a product request for new product	/me/product/new	POST	vendor	data: product request
Create a product request for existing product	/me/product/existing	POST	None	data: product request
Get all main products of this vendor	/me/mainProduct	GET	vendor	results: number of main products, data: main products

Functionality	URI	Method	Parameters	Return
Create a product request for deleting one main product	/me/mainProduct/:mpid	DELETE	None	data: product request
Get this vendor's product requests	/me/productRequest	GET	None	results: number of product requests, data: product request list
Get one product request request	/me/productRequest/:id	GET	None	data: product request
Update a product request	/me/productRequest/:id	PATCH	vendor	data: product request
Delete a product request	/me/productRequest/:id	DELETE	None	data: null

7.5 Category: (/category)

Functionality	URI	Method	Parameters	Return
Get all categories	/	GET	None	results: number of categories, data: category list
Post a category	/	POST	name	data: created category
Get one category	/:id	GET	None	data: required category
Update one category	/:id	PATCH	name	data: updated category
Delete one category	/:id	DELETE	None	data: deleted category

7.6 Product: (/product)

Functionality	URI	Method	Parameters	Return
Get all products	/	GET	None	results: number of products, data: product list
Create one product	/	POST	product	data: product
Get one product	/:id	GET	None	data: product
Add a vendor to the product	/:id	POST	vendorSpecifics	data: product
Update one product	/:id	PATCH	product	data: product
Delete one product	/:id	DELETE	None	data: null
Update a vendor in product	/:pid/vendor/:vid	PATCH	product	data: product
Delete a vendor in product	/:pid/vendor/:vid	DELETE	None	data: null
Search for products	/:pid/vendor/:vid	DELETE	None	results: number of products. data: product list
Get search filters	/:pid/vendor/:vid	DELETE	None	data: filters

For search and searchFilters more detailed explanation can be found in the following PR: https://github.com/bounswe/bounswe2020group8/pull/302

7.7 Main Product: (/mainProduct)

Functionality	URI	Method	Parameters	Return
Get all main products	/	GET	None	results: number of main products, data: main product list
Create one main product	/	POST	main product	data: main product
Get one main product	/:id	GET	None	data: main product
Update one main product	/:id	PATCH	main product	data: main product
Delete one main product	/:id	DELETE	None	data: null
Delete vendor from all products of this main product	/:mpid/vendor/:vid	DELETE	None	data: null

7.8 Product Request: (/productRequest)

Functionality	URI	Method	Parameters	Return
Get all product requests	/	GET	None	results: number of main products, data: main product list
Get one product request	/:id	GET	None	data: product request
Update one product request	/:id	PATCH	main product request	data: product request
Delete one product request	/:id	DELETE	None	data: null

7.9 Comment: (/comment)

Functionality	URI	Method	Parameters	Return
Get all comments from all customers for a product	/:pid/all	GET	None	results: number of comments, data: comment list
Get all comments from a customer for a product	/:pid	GET	None	results: number of comments, data: comment list
Create one comment	/:pid	POST	text	results: 1, data: created comment
Update one comment	/:pid	PATCH	_id, text	results: 1, data: created comment
Delete one comment	/:pid	DELETE	_id	data: deleted comment

7.10 Rate: (/rating)

Functionality	URI	Method	Parameters	Return
Rate a product	/:pid	PATCH	rate	results: 1, data: rated main product

7.11 List: (/list)

Functionality	URI	Method	Parameters	Return
Create a list	/	POST	title, wishedProducts	results: number of lists, data: current lists
Get all lists of a customer	/all	GET	None	results: number of lists, data: lists
Delete all lists of a customer	/all	DELETE	None	None
Export all lists cart	/all/export	POST	None	None
Get one list	/:lid	GET	None	results: 1, data: list
Update one list	/:lid	PATCH	title, wishedProducts	results: 1, data: updated list
Delete one list	/:lid	DELETE	None	data: deleted comment
Export one list to cart	/:lid/export	POST	None	None

8 Our Project Plan:

We've inserted our complete project plan in the next page.

		Name	Duration	Start	Finish	Predecessors	Resource Names
1		**Revision**	**22 days?**	**10/27/20, 8:00 AM**	**11/17/20, 5:00 PM**		
2		Requirements - revisit	8 days?	10/27/20, 8:00 AM	11/3/20, 5:00 PM		Everyone
3		Scenarios & mockups - revisit	8 days?	10/27/20, 8:00 AM	11/3/20, 5:00 PM		Oncel Keles
4		Class diagram - revisit	5 days	11/4/20, 8:00 AM	11/8/20, 5:00 PM	2	Burak Berk Ozer;Kaan Dura;Muhammet Tayyip Yazici;Yasin Kaya;Ozgurcan Oztas
5		Sequence diagram - revisit	3 days?	11/9/20, 8:00 AM	11/11/20, 5:00 PM	2;4	Afra Arslan;Ahmet Yigit Gedik;Goksu Baser;Nazim Berke Metin;Onur Enginer
6		Project plan - revisit	5 days?	11/3/20, 8:00 AM	11/7/20, 5:00 PM		Nursima Celik
7		W3C Standards Research	8 days	11/10/20, 8:00 AM	11/17/20, 5:00 PM		Everyone
8		Activity Streams 2.0 Research	8 days?	11/10/20, 8:00 AM	11/17/20, 5:00 PM		Everyone
9		**Back end**	**15 days?**	**11/10/20, 8:00 AM**	**11/24/20, 5:00 PM**		
10		Login-Signup	12 days	11/10/20, 8:00 AM	11/21/20, 5:00 PM		Burak Berk Ozer;Kaan Dura;Muhammet Tayyip Yazici;Ozgurcan Oztas
11		Homepage	12 days?	11/10/20, 8:00 AM	11/21/20, 5:00 PM		Burak Berk Ozer;Kaan Dura;Muhammet Tayyip Yazici;Ozgurcan Oztas
12		Dockerize	3 days	11/22/20, 8:00 AM	11/24/20, 5:00 PM		Burak Berk Ozer;Kaan Dura;Muhammet Tayyip Yazici;Ozgurcan Oztas
13		AWS Deployment - R&D	3 days?	11/22/20, 8:00 AM	11/24/20, 5:00 PM	10;11	Burak Berk Ozer;Kaan Dura;Muhammet Tayyip Yazici;Ozgurcan Oztas
14		**Front end**	**15 days?**	**11/10/20, 8:00 AM**	**11/24/20, 5:00 PM**		
15		Login-Signup	12 days?	11/10/20, 8:00 AM	11/21/20, 5:00 PM		Afra Arslan;Ahmet Yigit Gedik;Yasin Kaya;Oncel Keles
16		Homepage	12 days?	11/10/20, 8:00 AM	11/21/20, 5:00 PM		Afra Arslan;Ahmet Yigit Gedik;Yasin Kaya;Oncel Keles
17		Dockerize	3 days	11/22/20, 8:00 AM	11/24/20, 5:00 PM		Afra Arslan;Ahmet Yigit Gedik;Yasin Kaya;Oncel Keles
18		AWS Deployment - R&D	3 days?	11/22/20, 8:00 AM	11/24/20, 5:00 PM	15;16	Afra Arslan;Ahmet Yigit Gedik;Yasin Kaya;Oncel Keles
19		**Mobile**	**15 days?**	**11/10/20, 8:00 AM**	**11/24/20, 5:00 PM**		
20		Login-Signup	12 days?	11/10/20, 8:00 AM	11/21/20, 5:00 PM		Goksu Baser;Nazim Berke Metin;Nursima Celik;Onur Enginer
21		Homepage	12 days?	11/10/20, 8:00 AM	11/21/20, 5:00 PM		Goksu Baser;Nazim Berke Metin;Nursima Celik;Onur Enginer
22		Dockerize	3 days	11/22/20, 8:00 AM	11/24/20, 5:00 PM		Goksu Baser;Nazim Berke Metin;Nursima Celik;Onur Enginer
23		AWS Deployment - R&D	3 days?	11/22/20, 8:00 AM	11/24/20, 5:00 PM	20;21	Goksu Baser;Nazim Berke Metin;Nursima Celik;Onur Enginer
24		Milestone1	0 days?	11/24/20, 5:00 PM	11/24/20, 5:00 PM	9;14;19	Everyone
25		GDPR and KVKK Research	5 days?	11/24/20, 8:00 AM	11/28/20, 5:00 PM		Everyone
26		**Back end**	**34 days?**	**11/24/20, 8:00 AM**	**12/27/20, 5:00 PM**		
27		Profile - Admin Panel	5 days?	11/24/20, 8:00 AM	11/28/20, 5:00 PM		Burak Berk Ozer;Kaan Dura;Muhammet Tayyip Yazici;Ozgurcan Oztas
28		Product	7 days?	11/28/20, 8:00 AM	12/4/20, 5:00 PM		Burak Berk Ozer;Kaan Dura;Muhammet Tayyip Yazici;Ozgurcan Oztas
29		Search	6 days?	12/5/20, 8:00 AM	12/10/20, 5:00 PM	28	Burak Berk Ozer;Kaan Dura;Muhammet Tayyip Yazici;Ozgurcan Oztas
30		Activity Streams 2.0	5 days?	12/10/20, 8:00 AM	12/14/20, 5:00 PM	8;9;28	Burak Berk Ozer;Kaan Dura;Muhammet Tayyip Yazici;Ozgurcan Oztas
31		List-Notification	5 days?	12/10/20, 8:00 AM	12/14/20, 5:00 PM	28	Burak Berk Ozer;Kaan Dura;Muhammet Tayyip Yazici;Ozgurcan Oztas
32		Shopping Cart-Order-Purchase	11 days?	12/14/20, 8:00 AM	12/24/20, 5:00 PM	28	Burak Berk Ozer;Kaan Dura;Muhammet Tayyip Yazici;Ozgurcan Oztas
33		Comment and Rate	4 days?	12/24/20, 8:00 AM	12/27/20, 5:00 PM	28	Burak Berk Ozer;Kaan Dura;Muhammet Tayyip Yazici;Ozgurcan Oztas
34		**Front end**	**34 days?**	**11/24/20, 8:00 AM**	**12/27/20, 5:00 PM**		
35		Profile - Admin Panel	5 days?	11/24/20, 8:00 AM	11/28/20, 5:00 PM		Afra Arslan;Ahmet Yigit Gedik;Yasin Kaya;Oncel Keles
36		Product	7 days?	11/28/20, 8:00 AM	12/4/20, 5:00 PM		Afra Arslan;Ahmet Yigit Gedik;Yasin Kaya;Oncel Keles
37		Search	6 days?	12/5/20, 8:00 AM	12/10/20, 5:00 PM	36	Afra Arslan;Ahmet Yigit Gedik;Yasin Kaya;Oncel Keles
38		Activity Streams 2.0	5 days?	12/10/20, 8:00 AM	12/14/20, 5:00 PM	8;14;36	Afra Arslan;Ahmet Yigit Gedik;Yasin Kaya;Oncel Keles
39		List-Notification	5 days?	12/10/20, 8:00 AM	12/14/20, 5:00 PM	36	Afra Arslan;Ahmet Yigit Gedik;Yasin Kaya;Oncel Keles
40		Shopping Cart-Order-Purchase	11 days?	12/14/20, 8:00 AM	12/24/20, 5:00 PM	36	Afra Arslan;Ahmet Yigit Gedik;Yasin Kaya;Oncel Keles
41		Comment and Rate	4 days?	12/24/20, 8:00 AM	12/27/20, 5:00 PM	36	Afra Arslan;Ahmet Yigit Gedik;Yasin Kaya;Oncel Keles
42		**Mobile**	**34 days?**	**11/24/20, 8:00 AM**	**12/27/20, 5:00 PM**		
43		Profile - Admin Panel	5 days?	11/24/20, 8:00 AM	11/28/20, 5:00 PM		Goksu Baser;Nazim Berke Metin;Nursima Celik;Onur Enginer
44		Product	7 days?	11/28/20, 8:00 AM	12/4/20, 5:00 PM		Goksu Baser;Nazim Berke Metin;Nursima Celik;Onur Enginer
45		Search	6 days?	12/5/20, 8:00 AM	12/10/20, 5:00 PM	44	Goksu Baser;Nazim Berke Metin;Nursima Celik;Onur Enginer
46		Activity Streams 2.0	5 days?	12/10/20, 8:00 AM	12/14/20, 5:00 PM	8;19;44	Goksu Baser;Nazim Berke Metin;Nursima Celik;Onur Enginer
47		List-Notification	5 days?	12/10/20, 8:00 AM	12/14/20, 5:00 PM	44	Goksu Baser;Nazim Berke Metin;Nursima Celik;Onur Enginer
48		Shopping Cart-Order-Purchase	11 days?	12/14/20, 8:00 AM	12/24/20, 5:00 PM	44	Goksu Baser;Nazim Berke Metin;Nursima Celik;Onur Enginer
49		Comment and Rate	4 days?	12/24/20, 8:00 AM	12/27/20, 5:00 PM	44	Goksu Baser;Nazim Berke Metin;Nursima Celik;Onur Enginer
50		Milestone2	0 days	12/29/20, 5:00 PM	12/29/20, 5:00 PM	26;34;42	Everyone
51		**Back end**	**22 days?**	**12/29/20, 8:00 AM**	**1/19/21, 5:00 PM**		
52		Messaging System	8 days?	12/29/20, 8:00 AM	1/5/21, 5:00 PM		Burak Berk Ozer;Kaan Dura;Muhammet Tayyip Yazici;Ozgurcan Oztas
53		Recommendation System	6 days?	1/5/21, 8:00 AM	1/10/21, 5:00 PM		Burak Berk Ozer;Kaan Dura;Muhammet Tayyip Yazici;Ozgurcan Oztas
54		Testing	10 days?	1/10/21, 8:00 AM	1/19/21, 5:00 PM		Burak Berk Ozer;Kaan Dura;Muhammet Tayyip Yazici;Ozgurcan Oztas
55		**Front end**	**22 days?**	**12/29/20, 8:00 AM**	**1/19/21, 5:00 PM**		
56		Messaging System	8 days?	12/29/20, 8:00 AM	1/5/21, 5:00 PM		Afra Arslan;Ahmet Yigit Gedik;Yasin Kaya;Oncel Keles
57		Recommendation System	6 days?	1/5/21, 8:00 AM	1/10/21, 5:00 PM		Afra Arslan;Ahmet Yigit Gedik;Yasin Kaya;Oncel Keles
58		Testing	10 days?	1/10/21, 8:00 AM	1/19/21, 5:00 PM		Afra Arslan;Ahmet Yigit Gedik;Yasin Kaya;Oncel Keles
59		**Mobile**	**22 days?**	**12/29/20, 8:00 AM**	**1/19/21, 5:00 PM**		
60		Messaging System	8 days?	12/29/20, 8:00 AM	1/5/21, 5:00 PM		Goksu Baser;Nazim Berke Metin;Nursima Celik;Onur Enginer
61		Recommendation System	6 days?	1/5/21, 8:00 AM	1/10/21, 5:00 PM		Goksu Baser;Nazim Berke Metin;Nursima Celik;Onur Enginer
62		Testing	10 days?	1/10/21, 8:00 AM	1/19/21, 5:00 PM		Goksu Baser;Nazim Berke Metin;Nursima Celik;Onur Enginer
63		Milestone 3	1 day?	1/20/21, 8:00 AM	1/20/21, 5:00 PM	51;55;59	Everyone

9 Milestone Scenarios:

9.1 Scenario 6

9.1.1 Kerim Genç (Vendor)

9.1.1.1 Persona Kerim is 25 years old. He is a business manager of a Teknosa store. He had signup to Carousel before but did not place any products before. The store he manages just received new Apple Watch Series 5 products with white and black colors. In the pandemic conditions, he decides to try selling these new products online from the Carousel app.

9.1.1.2 Preconditions

- He is a vendor user, he has already signed up

9.1.1.3 Actions

1. He clicks Login on the Header.
2. He clicks Add Product on the side profile menu.
3. He clicks Create Main Product at the bottom.
4. He enters the form and submits.
5. He finds his newly added Main Product from the table and clicks List Child Products.
6. He clicks Create a New Product From This Main Product.
7. He fills up the form for Apple Watch Series 5 with white color.
8. He clicks submit.
9. He again finds his newly added Main Product from the table and clicks List Child Products.
10. He clicks Create a New Product From This Main Product.
11. He fills up the form for Apple Watch Series 5 with black color.
12. He clicks submit.
13. MEANWHILE, admin checks these products requests and accepts them.
14. He(Kerim) signs out in order to look for his products.
15. He searches for "watch".
16. He clicks on Apple Watch from the search results.
17. He sees both of the products he added on the product page.

9.1.1.4 Acceptance Criteria

1. Registered users shall be able to login with their e-mail and password. (Requirement 1.1.1.5)
2. Vendors shall be able to request to create, update or delete the page of products from admins.(Requirement1.1.3.3)
3. Admins shall be able to confirm if a product information is editted by the vendors(Requirement 1.1.13.6)

4. Users shall be able to choose to search for products or vendors.(Requirement 1.1.4.1)
5. The system shall hold product information on the product's page.(Requirement 1.2.6.1)
6. The system shall have pictures of products. (Requirement 1.1.4.1)

9.2 Scenario 5

9.2.1 Göksu Başer (Customer)

9.2.1.1 Persona Göksu is a 22 years old. He is a software engineer in Bogazici University. He is writing a e-commerce platform for his customer with his Android team. One day one of his teammates mentions that there is a new platform called Corousel and their features worth observing. Therefore, Göksu signups and creates a basic page than decides that to take a look at their features and buy a cheap new watch for new year for his friend.

9.2.1.2 Preconditions

- He is logged in user. He has telephone number and birthday is added before-hand. Also he has 4 addresses and 1 credit card saved

9.2.1.3 Actions

1. He opens the app.
2. He looks at the profile page and sees his name and observes the options.
3. He clicks the User Information button.
4. He tries to edit his informations; however, does not succeeds
5. He clicks Edit button and start editting his information
6. He clicks save and returns back the profile page automatically. Then he check his information changes and sees they are correctly done.
7. He checks his addresses
8. He deletes one of his addresses via scrollable address tab.
9. He clicks "+" button to add address.
10. He enters his new address information and clicks save address.
11. He checks whether the address is added or not and sees it is successful
12. He wants to check what is written in About and Contacts tabs.
13. He clicks these tabs in order and sees there is information is written about project and contact information.
14. He clicks My Lists button and sees his prebuild lists and adds one more list to there.
15. He proceeds to homepage and sees there is several products.
16. He ,therefore, goes to search tab and searches for a watch.
17. He sees several watches is listed and decides to filter them by price.
18. He filters them by 50-100$ price range

19. He sees there is one watch for 77$ and he adds it to his cart.
20. He proceeds to his cart and sees the watch is there.
21. He clicks purchase cart and the scenario finishes here.

9.2.1.4 Acceptance Criteria

1. Customers shall be able to change their name/surname or password after a confirmation of current password. (Requirement 1.1.2.1)
2. Customers shall have a profile settings page where after they confirm their current passwords they can edit their: (Requirement 1.1.2.2)
3. Users shall be able to view the page of products. (Requirement 1.1.3.1)
4. Page of the products shall contain price, expected delivery date, images of the product(if possible), technical features, availability at the moment, comments, rating, and its current vendor. (Requirement 1.1.3.2)
5. Users shall be able to choose to search for products or vendors. (Requirement 1.1.4.1)
6. Users shall be able to filter results according to: The average customer review, Brand name, Vendor name, Minimum and maximum price, Color (Requirement 1.1.4.3)
7. Customers and guests shall be able to add products to their shopping cart. (Requirement 1.1.5.1)
8. Customers and guests shall be able to see the price of each product in the cart. (Requirement 1.1.5.2)
9. Customers and guests shall be able to increase or decrease the quantity of the products. (Requirement 1.1.5.3)
10. Customers and guests shall be able to see the total price of the cart. (Requirement 1.1.5.5)
11. Customers shall perform these operations on the lists: creating, naming, deleting. (Requirement 1.1.6.1)
12. Customers are provided with one list namely "Watchlist" and one list namely "Favlist" by default and Requirements (1.1.6.1) is not applicable to these two lists. (Requirement 1.1.6.3)
13. Customers and guests shall be able to purchase the products in their shopping carts. (Requirement 1.1.8.1)

10 Structure of our codebase:

- **Branches:**
  We've decided to use 5 branches in our repository: master, development, backend, web, android. All these 5 branches have branch protection rules for any contributor to regulate the code pushing and code pulling actions within safety measurements. Also, before pushing our code into the repository, first we pull at our division branch(backend, web, android) and retrieve any changes, then we checkout to a new branch from that divisionary branch, implement our feature or add our bugfix and push to that branch on our remote repository. After pushing, we open a pull request from that branch to our divisionary branch and after an approval of sufficient number of revisions and pass of implemented or added checks into the branch, we merge that branch to our divisionary branch. After completing tasks in our project plan with confirmed changes, we merge our code from all divisionary branches to the 'development' branch to merge all work into one place. After that merging, we plan to check our structure and state of the product and if leader of each division approves the changes, we merge development into master branch to have our production image.
- **Updates in Milestone 2:**
  - **Web Division:**
    Web division has been followed the general naming conventions of our team, therefore there are two types of branches: web-feature and web-bugfix. For this milestone interval, we've planned to integrate admin page and its functionalities, product page, search and filtering, comments and rating functionalities. In the end, our admin page provides us accepting/denying requests related with any product, which are demanded by the vendor of the product. All functionalities for the milestone 2 is composed in the branch '#365' and merged with the previous version.

  - **Back-end Division:**
    Back-end Division has been followed the general naming conventions of our team, therefore there are two types of branches: backend-feature and backend-bugfix. For this interval, we've planned to integrate admin functionalities' endpoints, main product functionalities' endpoints, product functionalities' endpoints, product requests' endpoints, comment functionalities' endpoints, rate functionalities' endpoints, list functionalities' endpoints, shopping cart functionalities' endpoints, purchase functionalities' endpoints, and order functionalities' endpoints. Each functionality is composed in different branch and merged with the current version of our API.

- **Android Division:**
  Android Division has been partially followed the general naming conventions of our team, therefore there are branches, which are named with "android-" prefix. Therefore, each feature is composed into a different branch. For this interval, we've planned to integrate profile functionalities, product page and its related activity pages, lists and its related activity pages, comment activity pages, rate activity pages and home page activity. All of them were shown in the milestone presentations.

- **Pull Requests:** We've added checks and mandatory revision for each pull request to keep our codebase clean and structured. We've implemented a code format checking GitHub Action in our repository and it works fine for backend division. Also, automated unit-testing in GitHub Actions will be implemented in next few days for backend division.

11 Evaluation of the used tools:

11.1 Backend

11.1.1 Express

Express is a back-end web application framework for Node.js. It is free and open source. It has a minimal structure that can be extended using plug-ins allowing us to modify it to our needs. It uses middlewares to create a stack that each request follows. This middleware feature makes it very easy for us to add and delete features for any endpoint we want to change. At the end of Milestone 2, it is observed that choosing Express as our main framework is the greatest decision we've decided on in the planning phase of our product. Its flexibility and modular structure are so facilitative, beyond any discussion. We've composed many endpoints and functionalities during Milestone 2, and it has not failed us. I hope it would not fail us until the final milestone.

11.1.2 Docker

We used Docker to isolate our back-end application from the environment it was running. With Docker we could put all dependencies for our application in the image and run it on any computer with Docker installed. During the milestone 2 implementation, we've automatized our backend deployment with Ansible Playbook and Terraform, therefore we can say our Docker dependency is revoked for now.

11.1.3 Visual Studio Code

As a team we decided we would use Visual Studio Code as our coding environment. We used Prettier plug-in to format our code to follow a coding standard among team. Also, we've realized that dividing our screen into 4 editors and putting router, model/dataAccess, controller and service of a functionality one by one into these 4 editors is quite effective to picture the current status of the currently implementing functionality.

11.1.4 Terraform and Ansible

Terraform offers a declarative configuration language for creating resources on a number of platforms. Ansible is similar to Terraform but it is better suited for automatic configuration of VMs. We used Terraform to deploy our AWS EC2 instances. With Terraform we can easily scale our number of nodes in a more structured manner. Terraform has also integrated with Ansible, we are using them together to easily configure each node we create. We are also planning to use Ansible on its own to configure our VMs if ever need be. In Milestone 2 stage, we've updated our host ips due to insufficient AWS Credits which leads to an account change for our development/deployment machines.

11.1.5 Github Actions

We added a workflow for code quality checking using Prettier. This way we could always have our code at the standard level. We also tried to add a workflow for continuous delivery pipeline and it is under test. When it is finished we will have our backend application updated all the time.

11.2 Android

11.2.1 Android Studio

We have used Android studio IDE for coding. It is easy to learn and use. It has code builtin code formatter and git tools. This specifications makes coding and version control system really easy even for a beginner with Android and Git.

11.3 Frontend

11.3.1 Docker

We have used docker container to run our react web application. Thanks to docker environment, we have isolated our program from the system that the program runs on. No matter on which system the program runs, if the the system has the docker, dependency issues are no longer concerns. Additionally, the docker containers are so lightweight, easy to run and containers are constructed in a fast way.

11.3.2 React

React is an open-source, front end, JavaScript library for building user interfaces or UI components. It is maintained by Facebook and a community of individual developers and companies. React can be used as a base in the development of single-page or mobile applications. However, React is only concerned with rendering data to the DOM, and so creating React applications usually requires the use of additional libraries for state management and routing Redux and React Router are respective examples of such libraries.

11.3.3 Prettier and Visual Studio Code

We have decided to use Visual Studio Code IDE while developing the frontend because it has Prettier code formatter. Thanks to this feature of the Visual Studio Code, our frontend team could work without any conflict about the format of the code and unnecessary code changes in commits.

12 Assessment of the Customer Meeting:

12.1 Experience of Öncel Keleş, our Web presenter:

The presentation of the web site was a bit long, because the process of adding two products from scratch was long itself. So I needed to go over these processes fast enough in order to leave time for the android team. It felt a bit dull to make the presentation online while screen sharing because I was not able to see the expressions of the customer and couldn't see where the customer might felt confused or looked like having questions about the presented parts of the app.

The parts we presented where the ones I was involved heavily, so that made it easier for me to prepare for the presentation. I felt like I could answer any arising question from the customer about the presented features, e.g. if some planned feature was not seen.

After both of the presentations, the questions from the customer about the web part were mainly around some small buggish things which I knew could be solved easily afterwards. However, these things that seemed little details to us might just get the attention from the customer and raise questions in the customer's mind. I think that in order to prevent such complications, it might be better to explain such "details" before hand while also saying that these can be fixed easily in just 1 day, so that we can keep the customer's attention to the parts we really wanted to show and get feedback from.

12.2 Experience of Göksu Başer, our Android presenter:

The presentation of android was created quickly. I used myself as the agent because I thought it suited well with the scenario in my mind a I explained in the presentation section. The final meeting of our team was a little bit late. Therefore I needed improvise for the other features of my other Android teammates. This presentation showed me that even if we work overtime it is impossible to put every single feature with a tidy and good code. At this point every teammate should choose whether to code all features fast and untidy or leave some features and code the ones that most important tidy.

I think I am good at presenting and quick making-up in the hard situations. I even take a course about presentation this semester. Therefore, this presentation is a good practice for me and I am doing it lovingly. It teaches me to create scenarios and present over them. Also, in the problems that occurred in the presentation time, I, now, can make-up solutions faster.

It was very difficult to create a presentation in the last hour. My teammates explained their code and work-done quickly and I am expected to create a scenario and present it in 15 minutes or so. It was overwhelming. I wanted to quit. I wanted to disappear for a day from earth to be honest. However, I must dive headfirst into it so I took a deep breath and used our app like a generic user and make-up solutions fast in the coming bugs.

In general, I did not even expect to present a demo this fluently. Therefore,

I am very proud of myself in this subject. I cannot believe that it was completed this good. Absolutely, it is needed to be completed the day before and I needed more time to write my scenario and demo it over and over. This needs to be improved. I can understand that, if I work 50 hours non-stop before demo. This was a hard thing to accomplish for my teammates. Maybe we needed to compromise our code more and give time to presentation prepare.However, I don't think choosing code over presentation is necessarily bad thing.