CMPE 451 - Milestone 1 Report
Group 2

Abdullah Yıldız
A. Furkan Varol
Aleyna Kara
Aslı Aykan
Bekir Yıldırım
Burak Çetin
Furkan Nane
Hande Şirikçi
Mehdi Saffar (Communicator)
Mehmet Umut Öksüz
Murat Can Bayraktar
Ömer Faruk Deniz
Özdeniz Dolu

November 29, 2020

Contents

1 Executive Summary

We focused on the requirements we had for this milestone and kept additional functionalities as sketches as we worked to our goal. The project is up and running on a remote server on back-end side. Our front-end and android applications are also running and properly connecting to the server in the scope of our requirements.

Implemented Front-end Features

- Customer Shopping Cart
- Customer sign in (without Google login)
- Customer sign up (without Google sign up)
- Customer profile page
- Product page
- Product search page with filters (without search functionality)
- Trending products in homepage
- Add to card functionality
- Fully dockerized continuous integration

Planned Frontend Changes

- Improving overall styling of the site
- Clarifying product to picture matching on trending products page
- Redesigning customer profile and product pages

Implemented Android Features

- Post requests to the login endpoint
- Active initial screen
- Navigation between different layouts, e.g. From home to category page where the the products are horizontally listed in which subcategory and all subcategories of the category are laid horizontally
- Profile page adjusting according to the user type
- Shopping cart and the special design for the products in it
- Favorites and the unique design for the products in it
- Quick buy and add to favorites features
- Bottom toolbar showing fundamental scheme of the app
- Recycler view that provides the efficient scrolling and listing for products and subcategories
- Customized icons for not only representing categories in the home page but also for the symbolization for the structure of the app at tool bar
- Design of the login page that is similar to the trending apps

Planned Android Changes

- Changing login page to integrate google sign in
- Completing the product page

Implemented Backend Features

- Database deployment on amazon servers
- Login and Signup endpoints
- Endpoints for requesting product data
- Endpoints for requesting and changing shopping cart data

Planned Backend Changes

- Finishing the google login and sign up endpoints that were not implemented in time for the milestone
- Adding filtering functionality to product listing requests
- Search functionality and recommendation functionality

2 List and Status of the Deliverables

Deliverable	Delivery Date	Status
Web interface of the demonstration	24.11.2020	Complete ✓
Android interface of the demonstration	24.11.2020	Complete ✓
Database and endpoints of the demonstration	24.11.2020	Complete ✓
Requirements	24.11.2020	Complete ✓
API documentation	24.11.2020	Complete ✓
Project plan	24.11.2020	Complete ✓
User scenarios presented during the demonstration	24.11.2020	Complete ✓

3 Evaluation of the Deliverables

- Web Interface: Web interface was ready during the presentation and link to the web interface can be found at the wiki page of our group. Web interface is a very crucial deliverable in terms of demonstrating and having a healthy communication with the customer.
- Android Interface: Android interface was ready during the presentation. Just like the web interface, it serves a very important role in terms of having a healthy communication with the customer.
- Database and endpoints: Database and endpoints functioned properly during the presentation and they served their purpose very well. They are crucial for user interfaces to function properly as user interfaces rely on the back-end to get the data.
- Requirements: Requirements are updated according to the latest meeting with the customer and they are up to date. They can be found at the wiki page. Requirements are very important because they track the agreement between the software engineers and the customer. Also any time people are unsure as how to implement something, the requirements page is the place to visit.
- API documentation: API documentation is ready and link can be found at the wiki page. API documentation plays a crucial role in terms of communication between front-end and back-end teams.
- Project plan: Project plan is ready and can be found at the wiki page. Project plan is a general guideline that determines the dates and the tasks of the teams.
- User scenarios: User scenarios presented during the demonstration are ready and they can be found in this report. They are important to mimic the behaviour of the customers.

4 Coding Work Done by Each Team Member

A. Furkan Varol	As a member of the frontend team, I have learned ReactJS frontend framework and Antd UI framework. I also got familiar with a design tool named Figma. I have contributed to the design of multiple pages of our web page. I implemented the login page and login form. First, I implemented them according to our designs and made them work on mock data. Then, I integrated backend communication. I also implemented the trending grid component on the home page to show trending products to users. I reviewed pull requests of my teammates in the frontend team.
Aleyna Kara	Customized the icons of the app and designed the splash screen of the app. Created the layout for the home page consisting of categories. Tested its UX design with people out of our team. Provided necessary live data variables and functions considering object oriented principles, e.g encapsulation, after creating two fragments and their view models namely CategoryViewModel and HomeViewModel in order to follow MVVM pattern. To enhance the MVVM pattern in our app, implemented data binding into the layouts of our app to handle the click events immediately in their corresponding layouts. Implemented two classes, namely ProductViewAdapter and SubcategoryAdapter. The former is for arranging products horizontally and the latter is to list subcategories vertically. Then designed the layout for the use of by their view holders, e.g. product-card.xml which aims at serving the name, description, price and image of the product in addition to the quick buy and add to favorites button. Designed the login page after examining the current trending mobile apps and completed its functionalities for responsible user interface. Furthermore, implemented Google sign in procedure to be handy for Milestone 2. Then, created the interface namely GetflixApiService and implemented its function that sends post request to our endpoint responsible for the procedure of signing in. Also provided its necessary functions and object instance to propose smooth login process to our customers. Created mock data after gathering their images. Not only reviewed almost all pull requests except that for quick bug fixes but also provided necessary codes to be used instead of previous ones.
Aslı Aykan	Initialized the Android subrepo as the Android subteam leader with the splash screen activity, bottom navigation bar, and five empty fragments which are for the login, home, categories, cart and profile pages by also adding the necessary navigations and dependencies. Initialized the localization of our app with English and Turkish options. Created ProductModel, CategoryModel and SubcategoryModel class. Joined the team lead meetings, and created a poll to decide on a color palette for mobile and web. Shared Android and coroutine tutorials in Kotlin with our subteam, and tried to answer their questions about git and opening the subrepo on Android Studio via Zoom. Attended the database meeting of the backend team to have an idea about our database. Studied the MVVM pattern to use it on our app. Reviewed all the Android pull requests and also the first frontend pull request. Created an expandable recycler view for the categories page, and also a recycler view for favorites fragment, which is created from scratch. Added the toolbar. Tried to make our code to follow the MVVM pattern, so created the necessary view models for the favorites and cart page. Edited Hande's code for the cart page to use data binding and the necessary view model. Created a horizontal product layout for the cart and favorites page. Tried to resolve all the navigation bugs on the app and my friends' conflicts while merging. Added the register page with different options for the vendor and customer. Added the "continue as a guest" option to the login page and added the check function for empty edit texts on the register page, made the necessary changes on the profile page(different visibilities and button names for the guest and logged user) and added the necessary navigation to login/logout on the profile page. Tried to help our subteam while connecting to the backend. Added the product endpoint but postponed it to Milestone 2 to integrate it to our app more accurately instead of adding it at the last minute.

Burak Çetin	Setting up the Google API on google development console for backend usage. Coding the google login endpoint, token parsing and confirmation and google login system for backend. Helping frontend implement the google login system. Helping out with the milestone report.
Furkan Nane	Coded couple of model classes. Coded a serializer in order to send data in JSON format. Coded the /products/homepage/numberOfProducts end point which returns "numberOfProducts" products. Created 20 products and 15 brands in the database to use them during the presentation. Also attended the backend team meetings and helped the creation of the database structure.
Hande Şirikçi	As a member of Android team, I have learned Kotlin basics by following tutorials and improved my Kotlin skills each week as required. I studied MVVM pattern to use in our app. Attended all meetings of android and also one of the backend meetings which is for understanding the database structure. Designed and implemented three classes and two xml pages: the horizontal product cart view, first version of the login page, cart page with its adapter and profile page. Created structural variables in order to make code adaptable for different cases. Designed and implemented necessary static variables to control the app according to the user type. After connecting android version to the end points, tested the code and detected errors by debugging. Designed the user scenario for demo.
Mehdi Saffar	Initialize frontend subrepo. Initiate multiple team lead meetings to agree on a few important things (git branching scheme, pull request templates, color palette for mobile and web, where to have the api doc, folder structure...). Setup husky pipeline to enforce code style automatically with a git pre-commit hook. Write Dockerfile for frontend. Open figma document to work on the design of the website collaboratively in real time. Have a tutorial/q-a zoom meeting to explain how react works, and how we will distribute the coding tasks. Explain how to use git command line and why we want to squash and merge instead of regular merge for the PR in order to have a cleaner git history and avoid pitfalls of last semester. Create the EC2 instances of backend frontend and database and setup docker, security groups etc. Work on CI/CD from beginning to end with testing and creating the related scripts (for example to encrypt secrets). PRs on initial layout (header content and footer) in react, global state management, page routing, show customer info when logged in, attaching request interceptor to add authorization header and storing token in localStorage, implement search page with multiple filters, sort by, pagination which use query params in url and refreshes properly, improve home page design, show cart button in header, and various last minute fixes and making sure everything works on the website before the demo. Reviewed all frontend PRs and a couple ones from backend. Rewrote the Dockerfile of backend to make it work and accept db params from environment. Helped members of my team multiple time by going through their code in Zoom sessions and debugging. Worked on fixing CORS problem between frontend and backend.
Mehmet Umut Öksüz	Initializing backend subrepo and setting up environment (packages, db connection etc.). Creating endpoint documentation at Swagger. Creating folder structure and base classes for backend. Setting up JWT Authentication mechanism and permission classes. As backend team lead, I set up two meetings for database design and creation of model classes and as the team we designed db structure and implemented required model classes. I also worked on distributing tasks and following statuses of tasks. I have implemented regular login that is login via username and password no Google login, regular signup and token validation endpoints. I was a reviewer, for each pull request of backend team.

Ömer Faruk Deniz	Learned Django and PostgreSQL and provided necessary resources for other backend team members to learn the fundamentals of Django. Created two endpoints: listing the shopping cart items of a customer, and providing the details of a product. Created the database schema and model classes with backend team. Created a sample database that will help us to test our endpoints, it was also enriched by Furkan Nane. Helped frontend and android team members to setup our backend and database to their computers in the early stage, then dockerized and deployed our database to AWS. Reviewed all of the endpoints before merge, and collaborated with other team members.
Özdeniz Dolu	I am a member of the frontend team. Firstly, I have learned React, AntD and how to setup my local environment to work on the project. We had a team-only meeting after the casual meeting to create a design for the first pages that we are going to deliver on Milestone 1. We used Figma, which has a simultaneous edit feature. After we designed the pages, we assigned everyone in the team some of the parts. Since then, I have created the signup form, signup page, improvements on category bar, left half of the shopping cart page. I also have written the code to make signup page communicate with the backend. I have also done many minor adjustments and corrections on the components above. Other than the frontend code itself, I also had many meetings with Mehdi and others on how to make CI work properly. I did the frontend presentation in Milestone 1 customer meeting. I have taken ordinary meeting notes twice and frontend meeting notes twice. I have made PR reviews for many PR's on Frontend.
Abdullah Yıldız	I am a member of the frontend team. I have learned React and Antd UI framework until Milestone. I have contributed to the design process of our application in Figma. I took part in the shopping cart design, functionality and integration of multiple components with each other and also with the backend. I took part in creating product page. I created a component for url search bar. I reviewed most of the Pull requests.
Bekir Yıldırım	As a member of the Backend team, I have learned Django by following tutorials and completed my shortcomings about PostgreSql that I was familiar with before. We gathered as a backend team to create the database schema and implemented the schema we created as a model. On the backend side, I implemented the 'user / int: id / shoppingCart' endpoint that allows us to add items to the shopping cart. To test my implementation, I created a test class. I initialized Docker and made progress with other teammates over the docker file I created and dockerize it. I reviewed the pull requests opened for the backend side.

5 Challenges Met During the Deployment, Dockerizing, and CI/CD Processes

Author: Mehdi Saffar

In order to facilitate deployments, and following the advice given in the course, we decided to work on a CI/CD solution. In the past practice app we didn't have time to work on CI/CD so deploying was a hassle. We wanted to do CI/CD to be able to ship features and fix bugs quickly, keeping both the team and the client happy. It reduces deployments to simple git pushes to the remote repo, and doesn't make anyone dependent on a single person to build the apps and deploy them. Also, Github offers a free CI/CD solution called Github Actions, so why not benefit from it?

According to the requirements, we had to use EC2 instances and dockerize all of backend, frontend and database using Docker containers. Amazon also offers managed ECR instance specifically made to handle deployments with Docker but as far as we know that is also not allowed.

Since I am leading the frontend team, I thought the first priority after initializing our subrepo would be to prepare the Dockerfile. I know from experience that is easier to deal with deployment-related issue from the beginning when the project is still new and relatively uncomplicated, rather than towards the end near the milestone. I researched on how to create Dockerfiles for React apps and followed tutorials online. I had a first solution but it wasn't perfect. Since react apps require node modules, the final size of the Docker image would be in the 1.4GBs. This was too big to store and upload to the EC2 instance. It turns out there is a way to shrink the final image by an extreme amount using multistage build. The idea is that building the react app takes a lot of memory and a lot of disk storage, but once the build/ folder is created, it only requires a simple HTTP server to serve those files. The most used HTTP server for such a case is nginx. Once the second stage of the Dockerfile build is written, the final image shrinks down to 7.5MB which is a huge gain.

Now that we have the Dockerfile ready, we have to somehow build it and run it on the EC2 instance. First idea I tried was to clone the repo onto the EC2 instance and build the docker image there. The problem is that building uses too much resources and the build would just run out of memory or the EC2 would stop responding until it restarted. At first I thought I would have to build the Dockerfile somewhere else (like on local machine) then publish the image to Dockerhub and then pull it from inside the instance, but that would pose problems since the Docker image might contain secrets in the future and we can only use Dockerhub for public things, so I looked in the docs of Docker and found exactly what I needed: docker save gzip image.tar.gz to save a gzipped Docker image and docker load -i image.tar.gz to load the image on the EC2 instance.

I learned how to run commands remotely using SSH and copy files using SCP and I got used to running them through practice.

Now that we know approximately what needed to be done, I forked the repo to my account and started writing a workflow file in github actions. The reason for the fork is that I needed to try Github Secrets which we cannot experiment with in the real repo due to lack of admin permissions. I didn't know how many Github Secrets I would need to keep, and having to ask a TA everytime would make the feedback loop too slow. I created the workflow and after many hours of trial I was able to deploy the frontend.

In order not to be dependent on TAs to add/remove Github Secrets I made my own 'in-house' solution that works like this:

- Locally I would create an unencypted-secrets/ that is .gitignore'd which would hold any secrets I want to have. (Amazon key pairs, Slack web hooks, API keys)
- A .gitignore'd bash script that would encrypt all the files with gpg and store them in secrets/ folder which can be safely pushed to the repo
- Unencrypt the secrets/ inside the VM of Github Actions with a long password stored in Github Secrets.

The benefit is that all secrets depend on only one GPG PASSWORD. Once set as a Github Secret, any other secrets can be added/removed safely without the need for external intervention.

We met more challenges such as:

- Whenever an instance is stopped or crashes, Amazon assigns a new public IP address to it. This makes our workflow files brittle as anytime the IP address changes, someone has to update the IP and push a new commit just for editing those files. The solution was to associate a name tag to each instance (like backend, frontend, database) and ask the AWS CLI for the IP. For example, frontend need to have the IP of the backend to send REST requests, and backend needs IP of database for CRUD operations.
- Decide on a branching scheme. Ideally we would have master being the deploy branch and have another development branch for frequent commits. Since git is already confusing for many people we decided not to ask people to change the "main" branch they merge their PRs to but instead keep the current branching scheme and just have an extra branch called deploy. Whenever enough fixes/features have been added, it would take a simple merge to trigger the builds.
- Builds can take a long time, and it a stressful wait, so I created a new channel called deployment. Whenever a deployment is done, our Deploy Guy bot would send a notification with a happy message and a link to the frontend or backend.
- Running the docker is not enough, you need to remember that you need to configure security groups properly so that the ports on which the servers listen is exposed to Internet.
- We also needed to learn how to pass environment variables to docker, both at build time for frontend and run time for backend.
- The storage provided to docker containers is ephemeral storage by default, which is terrible to learn about when using dockerized databases. We had to learn what docker volumes were and how to use them to provide temporary disk storage for our PostgreSQL database.
- Since EC2 instance is just a virtual machine with no built-in Docker container management, we had to pay extra attention to "resetting" the state of the machine on every instance. That means deleting previous application workdir, stopping and deleting the previously running Docker container as well as the previous Docker image.

Last thing we had to take care of was the always confusing CORS problems. I had to learn how to properly configure the nginx server to give allow for CORS as well as for the backend API to do the same. Now that we have a working CI/CD, deploying is a breeze and can be relied upon.

I would like to thank Özdeniz Dolu for working closely with me on CI/CD setup and troubleshooting the biggest problems, and Mehmet Umut Öksüz, Ömer Faruk Deniz, Bekir Yıldırım who helped me with setting up the deployment pipeline for the backend as well.

6 Requirements

Here are the changes made to the Requirements and their reasons:

- 1.1.2.2 : "password field will have an show/hide option" is added. This is done because the customer wanted a show/hide option.
- 1.1.8.7 : "Vendors shall accept the returns from customers within the 3 days after the purchase date if a customer wants to return a product. Customers do not need to specify any reason for the return" is added. This is done because we asked customer "can you specify the return policy" and this is the answer we received.
- 1.1.11.1.2 : "Vendor users should be able to customize some features of the online shop to their liking like changing the colour of the background but in general online shops of the different vendors can be generic so how much a vendor can customize their online shop is a design choice" is added. This is done because we asked customer "to what extent vendors will be able to modify their online shops?". Customer told us that it can be generic.
- 1.1.1.2.7 : "Profile Picture (not mandatory)" is added. This is done because we asked customer "Will users be able to upload a profile picture?" and the answer was yes. We added "not mandatory" part to specify that it is not mandatory to sign-up.
- 1.1.9 : This part is edited completely. " Customers shall be able to communicate with the vendors or admins about orders via direct message" and "Vendors shall be able to communicate with the customers or admins about a certain product or order via direct message" are added. This is done according to the customers specification of the direct message system.
- 1.1.11.1.1 : "Vendor users shall be able to put the following information in their online shop (for reference: https://www.n11.com/magaza/trendimbu)" is added. This is done according to the feedback from the customer as to what should online shops supposed to look like.
- 1.2.14.1 : "The system must offer a language switch option" is added because of the feedback during the milestone 1 presentation.

7 Design Documents

Although our Use-Case Diagram was properly designed, I prefered to add some details to it while adhering to the requirements. Some cases are detailed to represent the cases' in-depth functionalities. Changes made to the Use-Case Diagram are given below:

- "Customer - Create List" is extended with "Enter name".
- "Customer - Set Alarm" is extended with "Set alarm for price going down a threshold" instead of "stock availability change" since this was not then wanted to be included in the product and not entered into the requirements.
- "Vendor - Add vendor information" is detailed by including "Enter name" and "Enter description" and extending "Adding picture".
- Vendor's "Cancel Order" case is included "Enter reason of cancellation" since Vendor has to specify the reason, while customer has not.
- Added "Directly Message with Vendors" case for Admins.
- "Punish cheating vendors" is extended with "Warn the vendor" and "Ban the vendor".

8 API Documentation

Link for API documentation can be found at the wiki page. Here are the API endpoints implemented:

- get /init : Initially checks if token is valid

  Parameters:

  - Authorization

  Responses:

  - token
  - id
  - email
  - firstname
  - lastname

- post /regularsignup: Signup for customers
  Parameters:
  - username
  - email
  - password
  - firstname
  - lastname
  - phonenumber

  Responses:
  - successful
  - message
- post /regularlogin: Login with email and password
  Parameters:
  - email
  - password

  Responses:
  - status
  - user
- get /googlelogin: Login using google apis
  Parameters:
  - id-token

  Responses:
  - status
  - user
- get /products/homepage/numberOfProducts: Get products to be shown in home page
  Parameters:
  - numberOfProducts

  Responses:
  - Product[]

- get /product/productId: Get product details by id

  Parameters:

  - productId

  Responses:

  - id
  - name
  - price
  - creation-date
  - total-rating
  - rating-count
  - stock-amount
  - description
  - image-url
  - subcategory
  - category
  - vendor
  - brand

- get /user/userId/listShoppingCart: get items in shoping cart

  Parameters:

  - Authorization
  - userId

  Responses:

  - id
  - amount
  - product

- post /user/userId/shoppingCart: add a new item into shopping cart

  Parameters:

  - Authorization
  - userId
  - amount
  - productId

  Responses:

  - successful
  - message

9 Project Plan

Name	Start Date	Finish Date	Assignee Group
Creating initial architecture of Django App	11/03/20 8:00	11/17/20 17:00	Backend Team
Setting up JWT authentication	11/03/20 8:00	11/17/20 17:00	Backend Team
Database design	11/03/20 8:00	11/17/20 17:00	Backend Team
Implementation of database model classes in code side.	11/03/20 8:00	11/17/20 17:00	Backend Team
Implementing register, login and token validation for customers	11/17/20 8:00	11/24/20 17:00	Backend Team
Implementing shopping cart get and add items functionalities	11/17/20 8:00	11/24/20 17:00	Backend Team
Implementing Google login	11/17/20 8:00	11/24/20 17:00	Backend Team
Implementing products/homepage and product details	11/17/20 8:00	11/24/20 17:00	Backend Team
Unit test development	11/17/20 8:00	11/24/20 17:00	Backend Team
Dockerizing backend and database	11/17/20 8:00	11/24/20 17:00	Backend Team
Initialize the app	11/03/20 8:00	11/10/20 17:00	Android Team
Implement the bottom navigation bar and necessary navigation	11/03/20 8:00	11/10/20 17:00	Android Team
Implement the splash screen	11/03/20 8:00	11/10/20 17:00	Android Team
Implement the localization of the app	11/03/20 8:00	11/10/20 17:00	Android Team
Implement the categories page	11/10/20 8:00	11/17/20 17:00	Android Team
Implement the favorites page	11/10/20 8:00	11/17/20 17:00	Android Team
Implement the cart page	11/10/20 8:00	11/17/20 17:00	Android Team
Implement the profile page	11/10/20 8:00	11/17/20 17:00	Android Team
Implement the category page	11/10/20 8:00	11/17/20 17:00	Android Team
Implement the sign in form	11/17/20 8:00	11/24/20 17:00	Android Team
Implement the sign up form	11/17/20 8:00	11/24/20 17:00	Android Team
Implement the login endpoint connection	11/17/20 8:00	11/24/20 17:00	Android Team
Initialize frontend folder, setup React	11/03/20 8:00	11/10/20 17:00	Frontend Team
Create initial layout, global state management, routing	11/03/20 8:00	11/10/20 17:00	Frontend Team
Add product card component	11/03/20 8:00	11/10/20 17:00	Frontend Team
Add signup form component	11/03/20 8:00	11/10/20 17:00	Frontend Team
Add search input component	11/03/20 8:00	11/10/20 17:00	Frontend Team
Create login page, login form	11/03/20 8:00	11/10/20 17:00	Frontend Team
Add "Trending Products' grid in home page	11/10/20 8:00	11/17/20 17:00	Frontend Team
Add search product page	11/10/20 8:00	11/17/20 17:00	Frontend Team

Name	Start Date	Finish Date	Assignee Group
Add logged in user info on top bar	11/10/20 8:00	11/17/20 17:00	Frontend Team
Add shopping cart page	11/10/20 8:00	11/17/20 17:00	Frontend Team
Add profile page	11/17/20 8:00	11/24/20 17:00	Frontend Team
Add Axios instance, token management, login, logout	11/17/20 8:00	11/24/20 17:00	Frontend Team
Design improvements	11/17/20 8:00	11/24/20 17:00	Frontend Team
Integrate necessary pages with Backend	11/17/20 8:00	11/24/20 17:00	Frontend Team
Add Product page	11/17/20 8:00	11/24/20 17:00	Frontend Team
Finalize dockerization and deployment pipeline	11/17/20 8:00	11/24/20 17:00	Frontend Team
Milestone 1 - 11/24/20			
Implementing Search-Sort Mechanism	11/30/20 8:00	12/03/20 17:00	Backend Team
Detailed Product and User Profiles	11/30/20 8:00	12/03/20 17:00	Backend Team
Implementing Comment Mechanism	12/03/20 8:00	12/10/20 17:00	Backend Team
Implementing Direct Message Mechanism	12/09/20 8:00	12/10/20 17:00	Backend Team
Implementing Shopping Cart	12/10/20 8:00	12/16/20 17:00	Backend Team
Implementing Lists	12/15/20 8:00	12/17/20 17:00	Backend Team
Implementing Order Mechanism	12/18/20 8:00	12/24/20 17:00	Backend Team
Implementing Notification-Alarm	12/23/20 8:00	12/31/20 17:00	Backend Team
Implementing Search-Sort Mechanism	11/30/20 8:00	12/03/20 17:00	Android Team
Detailed Product and User Profiles	11/30/20 8:00	12/03/20 17:00	Android Team
Implementing Comment Mechanism	12/03/20 8:00	12/10/20 17:00	Android Team
Implementing Direct Message Mechanism	12/09/20 8:00	12/10/20 17:00	Android Team
Implementing Shopping Cart	12/10/20 8:00	12/16/20 17:00	Android Team
Implementing Lists	12/15/20 8:00	12/17/20 17:00	Android Team
Implementing Order Mechanism	12/18/20 8:00	12/24/20 17:00	Android Team
Implementing Notification-Alarm	12/23/20 8:00	12/31/20 17:00	Android Team
Implementing Search-Sort Mechanism	11/30/20 8:00	12/03/20 17:00	Frontend Team
Detailed Product and User Profiles	11/30/20 8:00	12/03/20 17:00	Frontend Team
Implementing Comment Mechanism	12/03/20 8:00	12/10/20 17:00	Frontend Team
Implementing Direct Message Mechanism	12/09/20 8:00	12/10/20 17:00	Frontend Team
Implementing Shopping Cart	12/10/20 8:00	12/16/20 17:00	Frontend Team
Implementing Lists	12/15/20 8:00	12/17/20 17:00	Frontend Team
Implementing Order Mechanism	12/18/20 8:00	12/24/20 17:00	Frontend Team
Implementing Notification-Alarm	12/23/20 8:00	12/31/20 17:00	Frontend Team
Milestone 2 - 12/29/20			

Name	Start Date	Finish Date	Assignee Group
Implementing Recommendation Mechanism	01/04/21 8:00	01/21/21 17:00	Backend Team
Annotation	01/14/21 8:00	02/02/21 17:00	Backend Team
Implementing Recommendation Mechanism	01/04/21 8:00	01/21/21 17:00	Android Team
Annotation	01/14/21 8:00	02/02/21 17:00	Android Team
Implementing Recommendation Mechanism	01/04/21 8:00	01/21/21 17:00	Frontend Team
Annotation	01/14/21 8:00	02/02/21 17:00	Frontend Team
Milestone 3 - 01/19/21			

10 User Scenarios Presented

10.1 Android

- Registered User: User opens the application then splash screen welcomes her. Login page appears and she types her username and password. User looks at the categories in the homepage, clicks on women wear and takes a look at the vertical product carts. Then she passes to the categories page and examines the subcategories. After that, user looks at the favorites page and saw her liked products in cart view which she can adds to her cart. Then she clicks on cart page and scrolls the page to see all of the products in her cart. After all, user comes to the profile page and examines her user information and clicks log out button.

- Guest User: Guest user opens the application then splash screen welcomes her. Login page appears and she continues as a guest. User looks at the categories in the homepage, clicks on women wear and takes a look at the vertical product carts. Then she passes to the categories page and examines the subcategories. After that, user realizes that she can create a favorites list and cart list. After all, user comes to the profile page and by clicking login button she decided to sign up this application and clicks sign up button.

10.2 Frontend

- Ahmet is a white collar male aged 30. He wants to buy a present to his friend for his birthday. He finds Getflix and searches for a "watch". He likes one of the watches in the search results. He clicks on it to see its product page to check on its details. He decides to buy it so he decides to register to the website. Clicks on the signup button on the right top corner of the page. He fills in the form and signs up. Then he gets redirected to the login form. He inputs his credentials and logs in. He gets to his account page through a dropdown list on the top right corner. He sees his account details. He comes back to the search page. He finds the watch he liked and adds it to his shopping cart. He also finds some products that he was thinking of buying and adds them to the shopping cart. Then he clicks on the shopping cart icon on the right top corner of the page and gets redirected to his shopping cart page. At the last minute, he decides that he also wants that watch for himself and he increases the amount of the product by 1 in the shopping cart. Then he proceeds to checkout.

11 Code Process

In this project we are using the following git branching scheme:

- master is our main development branch. All accepted PRs merge to it.
- deploy is the deployment branch. Whenever we want to deploy our newest changes in master we merge from master to deploy and two Github Actions, one for deploying frontend and one for backend, get triggered automatically. The database is manually deployed.
- All other branches are feature branches. We have agreed that each feature branch should be named as team-feature-description where team can be F for frontend, B for backend and A for android, and feature-description is a short description of the purpose of the branch. Example name: F-add-login-form, B-fix-CORS or A-products-recycler-view. Ideally, each feature branch is to be used by a single developer, and should not be merged-from or merge to other feature branch. It strictly branches from master and merges back to it. Sometimes, intervention by another member is required so commits from multiple people on the same branch can happen.

Last year we had trouble with messy git history when using the regular merge. This year, we decided to try another feature which makes more sense and keeps a very clean git history. In a meeting, I (Mehdi) explained the need for Squash and Merge and showed how it works and explained its benefits. In order to enforce it, I disabled all other options from the repo settings. What Squash and Merge does is it takes all the commits of a single branch and "squashes" them into a single meaningful commit. It preserves all commit messages from those commits and Github interface allows you to edit those message to keep only what is meaningful and throw out the meaningless commits like "fix previous" or "remove typo" which pollute the git history. It makes sense because a feature should be just that, a single "atomic" thing that is added to the project.

The code work flow is like this:

- In a subteam meeting we discuss the current state of the project and check what things need to be done. Then we try to distribute the tasks fairly among members.
- Each person either opens an issue or directly opens a branch and starts coding
- Once the person feels that they have done most of the work, they can open a PR and assign reviewers.
- When opening a PR, a template is automatically used and is filled by the person. The template can be found inside the .github folder in the repository.
- The reviewers take their time to thoroughly review each line of code and run the branch version on their local machine to test if it works as intended.
- The reviewer makes comments on a few code parts and decides if the PR is accepted or requires more changes.
- The person keeps making commits until most reviewers accept the PR.
- The person Squash and Merges the branch to master and deletes their branch since it is no longer needed.

Once enough changes are integrated to master, we deploy the new app by merging master to deploy, and once the app is (hopefully) deployed, everyone in our Slack workspace receives a notification from our Deploy Guy bot!

So far, 55 PRs have been opened since the beginning of the project.

12 Evaluation of tools and managing the project

12.1 Swagger

Author: Furkan Nane

As the back-end group, we used swagger to document API end points. By documenting the endpoints, we were able to establish a healthier communication with the web front-end and android groups. Also it made it possible to share the endpoints since we knew what end-points we wanted to implement. Moving on, we want to improve our documentation skills and keep the swagger page up to date in order to fully take advantage the utilities swagger offers.

12.2 Typora

Author: Mehdi Saffar

Typora is a free minimal markdown editor that we loved using since discovering it. It provides preview as you write (in a single pane, not two panes), supports syntax highlighting of multiple languages, tables and allows you to focus on the content thanks to its design inspired by minimalism and its Zen mode that hides any distraction. It also supports images, diagrams, inline styles and much more. Our meetings have become more focused and typing out the meeting notes has become a breeze. We stsarted sharing the screen so that everyone can see what we are talking about. It has also been helpful to communicate our demands to backend team. For example, after building all of login, signup, search pages etc I created a quick document showing what kind of data we can generate on the page with code blocks and went through it with the backend team, making necessary adjustments as we go.

12.3 ReactJS

Author: Mehdi Saffar

ReactJS is a frontend framework that makes it painless to build modern websites. It provides a way to describe views for each component that is derived from state and efficiently, in the background, updates the HTML DOM for you. It is based on the principles of functional and declarative programming, where the developer simply describes how the view is generated from the state and React does the heavy lifting. It works by giving each component properties as input which combined with its internal state output some React elements. By keeping track of the previous HTML elements and currently rendered HTML elements, and by making optimizations based on the nature of each component, it is able to know exactly which components need to be re-rendered to the HTML DOM. This makes the job of the developer much easier, the code much more predictable and easier to debug. Since everything is neatly compartmentalized into components and since the communication between each components follows a clean hierarchy tree, dividing tasks among team members is easier than with other frameworks or with vanilla js. Usually, we would build the design in Figma collaboratively, try to define "borders" between each component and then assign

those components to ourselves. Later, if needed, someone can take the work of all other team members and "stitches" them together, polishing things along the way and make sure everything is working as intended.

12.4 GitKraken

Author: Özdeniz Dolu

GitKraken is a free desktop application that provides a git GUI. Some of the group members started using it for managing git commands. It provides a simple interface where the user connect to his/her Github repository. It's great for taking an overall look at the status of the repository. It lists both all the remote branches and local branches. Shows info on local branches if it's behind the remote branch. It makes it easy to stash/stage/commit changes. It provides a text editor to solve merge conflicts and to add commit messages. As many of the group members were inexperienced on git commands, this tool provided a easy to use interface. Therefore, for its users, it provided a less error-prone local development environment.

12.5 Visual Studio Code

Author: Özdeniz Dolu

Visual Studio Code is a free text editor for desktop. It provides a lot of advanced text editing, indenting, formatting features for code writing. It also provides many extensions. It also has a git connection module therefore it can be also used as a git GUI. It has a capacity to open terminals in it and you can access the whole folder structure of your local repo easily. All of these features allows user to do almost all of his/her development in a single application. Due to its convenience and its user friendly interface, it helps with the development process a lot.

12.6 Android Studio

Author: Aslı Aykan

Android Studio is the official Integrated Development Environment (IDE) for Google's Android OS, which is used for Android app development and based on IntelliJ IDEA (a Java IDE for software). It is available for Windows, Linux and Mac desktop platforms. It was firstly announced on May, 2013 at Google conference, and replaced the Eclipse Android Development Tools, which was used for Android app development. For the ones who don't have a chance to run its app on an Android phone, it provides a feature-rich emulator(Android Virtual Device). It has also GitHub integration and a Gradle-based build system. The type of modules that each project on Android Studio may contain are Android app modules, library modules, and Google App Engine modules. It also has a very useful layout editor, in which users can easily create layouts by dragging/dropping UI components and giving fixed sizes/margins without writing code on .xml files. We could easily use the layout editor and ran the project on the Android emulator while implementing our app. GitHub integration also helped us while moving between branches, committing and resolving conflicts. However, sometimes the emulator could cause some problems, especially with the share screen feature on Zoom, it sometimes crashes. With an Android phone, however, everything works well and faster.

12.7 Ant Design

Author: Abdullah Yıldız

Antd is a UI design framework for multiple Javascript Languages. We have used Antd for React framework by importing it as npm package. It helps the developers to create basic HTML elements without worrying about its internal representation or the design of it. Throughout the project, we have used various different ant design components like Button, Card, Rate, InputNumber, etc. For example, with the Card component, we were able to put an image, a paragraph and a title into simple component without doing the CSS. Also, all of the Antd components come with pre-defined functions that developers might use.

12.8 Figma

Author: Abdullah Yıldız

Figma is a collaborative interface design tool that we used as a team. Figma provides a common, intelligent design page for each of the team members. Figma supports tools and components to create a realistic browser and mobile screens. It has very rich components like buttons, search bars, browser screen templates, images, icons etc. It also enables developers to add more library into it. We have imported a color palette, that we decided to use in our project, into Figma. In our first week, we have gathered as a team and created the design templates for frontend. Later on, as we got into the coding, Figma design let us visualize our components' internals. It made the coding part easier.

12.9 Postman

Author: Umut Öksüz

Postman is an application that we used for testing purposes. Postman provides a set of tools by which one can document API, automate testing, send different kinds of requests and get responses. Since our documentation was in Swagger, we used Postman to only test our endpoints by sending different kind of requests before unit testing phase. Postman makes customising a request easy, it allows you to put any kind of authorization headers, body data, etc. That's why we decided to use Postman.

12.10 PgAdmin

Author: Ömer Faruk Deniz

PgAdmin is a management tool for PostgreSQL database. It provided us the interface to create, view, edit and delete the rows, tables and the database itself. It provided a great flexibility when I was creating and filling our sample database. Using its clean interface, we were easily find the bugs related to the database schema. It was simple, clean and fast. Therefore, it made working with database smoother for us.

13 Assessment of the customer presentation

These are the feedback given during the presentation by the customers:

- Our web interface had prices in terms of TL and our android interface had prices in terms of US Dollars. This created some confusion so we will come up with a common currency.
- Some customers were not convinced that if we really had products for each category. We answered these questions and showed the products. Better explanation during the presentation would help to overcome those misunderstandings.
- Search was not ready but during the presentation we used the mock search to get some products as answer. Obviously this created confusion, better explanation would help people to understand the situation.
- Some customers wanted the language to change according to location or from a button. This is clearly a very good feedback since we learned one more requirement.
- We were asked about deployment from git and we explained it to the customers. Customers were happy and satisfied about the deployment.

Overall, the presentation for the milestone 1 went smoothly and we confirmed as a group that we are on the right path. In general, customers looked happy and we felt that they are in agreement with us in terms of the project's direction. Since this was the first milestone, deployment, bugs and getting used to new software tools took a lot of time. That's why we did not have a lot of time to prepare for the presentation. Next time we will make sure that we have enough time to prepare for the presentation because thinking about how the customers will understand and view the presentation is very important and it is crucial to have a healthy communication with the customers.