CMPE 451 - Milestone 2 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

January 5, 2021

Contents

1 Executive Summary **2**

2 List and Status of the Deliverables **5**

3 Evaluation of the Deliverables **5**

4 Coding Work Done by Each Team Member **6**

5 Challenges Met During the Deployment, Dockerizing, and CI/CD Processes **10**

6 API Documentation **10**

7 Project Plan **14**

8 User Scenarios Presented **16**

9 Code Process **16**

10 Evaluation of tools and managing the project **18**
10.1 Swagger 18
10.2 Typora 18
10.3 ReactJS 18
10.4 GitKraken 19
10.5 Visual Studio Code 19
10.6 Android Studio 19
10.7 Ant Design 19
10.8 Figma 20
10.9 Postman 20
10.10Github Projects 20
10.11PgAdmin 20
10.12Android Studio 21

11 Assessment of the customer presentation **21**

12 Unit Test **21**
12.1 Mehmet Umut Öksüz 21
12.2 Burak Çetin 22
12.3 Bekir Yıldırım 22
12.4 Ömer Faruk Deniz 23
12.5 Furkan Nane 23

1 Executive Summary

For milestone 2, we planned our scenario after milestone 1 and implemented our functionalities according to our scenario. We focused on completing major functioanlities for customers and vendors (Purchasing a product, reviews, managing orders and products for vendor, etc). And planned to implement remaining functionalities we specified in requirements at milestone 3. Backend team have completed majority of coding one week earlier than deadline to buy other teams time for integrating to endpoints. Thanks to our effective collaboration and feedbacks at last week, frontend and Android teams have successfully integrated their code with backend endpoints. So, as Getflix team, we have provided all features we planned to have in milestone 2 demo.

Implemented Front-end Features

- manage credits cards
- manage addresses
- full checkout process
- make search products work
- remove trending products and replace it with horizontal list of best sellers in each category
- move search bar in header bar, move category under header bar
- add profile page with tabs for (edit, cards, messages, addresses)
- fix all kinds of bugs (routing, race conditions, product card...)
- add product detail page
- add user reviews
- review a product when delivered
- change signup form to take full addresses
- add email verification
- add loading page when doing init
- better token management and injection into http headers
- add mock proxy server for faster development
- add list functionality (only mock)

Planned Frontend Changes

- vendor and admin pages
- ability of vendor to manage products and manage orders
- add messaging functionality

- allow reviews during at cargo order status
- implement notification feature for price changes
- integrate lists with backend

Implemented Android Features

- manage credit cards (add, update, delete)
- manage addresses (add, update, delete)
- list of orders for customer
- full checkout process
- add different dialog types
- add fragment animations
- add subcategory page
- fixed all the bugs related to toolbar and navigationfixed all the bugs related to toolbar and navigation
- add product page with detailed info and reviews
- add loading animation
- add filter options page
- add necessary viewmodels and adapters for customer option
- add four vendor pages with necessary viewmodels and adapters(vendor profile, vendor orders, vendor messages and vendor home)
- list and update products for vendor
- redesigned the profile page with more options on it
- redesigned the home page for a better UX with many realistic adapters on it
- redesigned the minimal product layouts on cart, subcategory and home pages

Planned Android Changes

- add admin pages
- add more endpoint connections to vendor pages
- add message functionality
- make all the filter options work
- make search products work
- post review

- add email verification
- change signup form to take full addresses
- implement notification feature for price changes

Implemented Backend Features

- Vendor Functionalities
  - Vendor login and signup
  - Add update and delete products
  - See orders with details
  - Manage status of orders
- Customer Functionalities
  - Add, update, delete and see credit cards
  - Add, update, delete and see credit addresses
  - Searching and filtering products
  - Reviewing a product and seeing reviews of other users
  - See checkout details (price, delivery price, discount)
  - Purchase a products
  - Get past orders
  - Cancel an order
- Email verification

Planned Backend Changes

- Customer
  - Creating lists for products
  - Adding deleting products for lists
  - Account management (updating profile and changing password)
- Vendor
  - Account management (updating profile and changing password)
- Messaging system
- Notification system
- Completing Google sign up and sign in functionalities and integrating with frontend and Android

2 List and Status of the Deliverables

Deliverable	Delivery Date	Status
Web interface of the demonstration	05.01.2021	Complete ✓
Android interface of the demonstration	05.01.2021	Complete ✓
Database and endpoints of the demonstration	05.01.2021	Complete ✓
Requirements	05.01.2021	Complete ✓
API documentation	05.01.2021	Complete ✓
Project plan	05.01.2021	Complete ✓
User scenarios presented during the demonstration	05.01.2021	Complete ✓

3 Evaluation of the Deliverables

- Web Interface: The frontend application of our project was flawless in the presentation. We have showcased multiple components without any bugs and problems. Planning and implementation are designed by taking user experience into account as discussed in the lectures and lab hours. The functionalities that are presented were in line with our customer's needs.
- Android application of our project was ready and fully functional in the presentation. Previous technical difficulties with the emulator are solved in this milestone. Components in the Android application were in sync with the Web Interface in terms of design and functionality.
- Database and endpoints: Database and endpoints worked as intended during the presentation. We have created data to be used in the presentation specifically for our scenario. We have also formed a test database in case things go out of control with the original database. For once during the presentation, the backend team manually changed the status of an order from the database such that the frontend can present the 'Orders' page and can post a review of a product to complement our scenario.
- Requirements: Requirements were already cleared and updated in Milestone 1 so there was no need for any other change for this milestone. Here is the link for our requirements wiki page : Requirements
- API documentation: API documentation is being updated by the backend team regularly as they add new endpoints or change models. This way, Android and Frontend teams could work without constant need for the Backend team.
- Project plan: Our project plan is ready and can be found on our wiki page. The project plan has been a directive in planning our work, distributing tasks, and completing the aforementioned deliverables.
- User scenarios: User scenarios that we used in the presentation can be found in the report. The scenarios for Frontend and Android teams are arranged by the team leaders and got approved by all team members. Scenarios are arranged to be in line with each other and to complement the flow of the presentation.

4 Coding Work Done by Each Team Member

A. Furkan Varol	At the start of Milestone 2, I refactored the Profile Page. Transformed the simple design of this page into a tabs pane. In addition to updating the profile, several functionalities are either relocated into the profile page or implemented for the profile page such as displaying order history, cards, addresses, and messages by teammates. Then I created the Best Sellers component which is used on the Home Page and the Product Page. This is a horizontally scrollable list that displays multiple products. I used the search endpoint of our backend with the best sellers query parameter to retrieve the products displayed here. Product Card component is slightly modified and used to show the best selling products. At last, I refactored and updated the Trending Products component on the Home Page. It was containing only product images. I implemented a foreground that shows the title, price, and rating information while hovering.
Aleyna Kara	I have pushed Google sign in forward until the first milestone. However, there have been an issue regarding to the fact that a user could not authenticated as well as can be expected. After the exchange of ideas with Aslı Aykan and search through various resources, we have settled on the completion of task by Aslı Aykan while I have left the A-google-signin branch, which currently exists, and worked with other tasks that needed to be fulfilled. Parallel to the topics that have been discussed during the class hours and PSs, I have researched UI and UX design from the rudimentary knowledge to the underlying ideas for the mobile apps, e.g. the how to create a mobile app having better UI and UX design. In order to put what I have learnt into practice, I have tried to add Flutter into our existing project. However, it has been onerous compared to building the mobile app using Flutter from scratch and needed to be made enormous changes in the code and studied by each team member, which might have seen as an extra workload. Despite Flutter, I have added the dependency of the libraries-namely DiscreteScrollView, CircularImageView, CircularIndicator and Picasso- to drive benefit from them in UI and UX design. Furthermore, I have designed my own ui component by implementing the recycler view adapter with multi view holders -HomeCategoriesAdapter.kt. In fact, there have ben considerable amount of open source project in GitHub but I have gone for customizing my own component from scratch due to fact that they have not only a few number of forks, stars,contributors and so and so forth but also a small community. Starting with the design of the product page, I have implemented three recycler view adapters, namely RecommenderAdapter.kt, CommentAdapter.kt and ImageAdapter.kt, the fragment class with its view model and the layouts both for the fragment and the view holders of the adapters. In spite of the existence of the libraries for the expandable recycler view, I have written the expandable recycler view myself for the best practice. When checking almost all mobile apps out, finding the corresponding place where we search for the information that is intended to be found out can be really cumbersome. With the aim of better UX, I have implemented both info and comment button which are able to scroll directly to the details section and review section of the page respectively. The fact that images of a product stays at the top while a user is able to scroll, read reviews and details within the section lying under the images is one more additional nuance that bear fruits in UX design and the uniqueness of the project because a user has not scroll to the top of the product while comparing the either details and reviews of the product with its images in these use cases. I have also design the home page from scratch by creating 5 recycler view adapters including the layouts for their view holders. I have created space generator and examined post at the website of the material design in order to design cardeditorpick.xml whose design is commonly used in the today's trending apps. For the consistency in the design, I have changed some details, e.g button style, colors, which have been previously implemented while I have redesigned the profile page and cart page. During the design of the profile page, I have used the circular image view component and updated the necessary places in ProfileFragment.kt. For the cart page, I have redesigned fragmentcart.xml and productcard.xml and also I have updated CartAdapter.kt, CartViewModel.kt and CartFragment.kt according to the new layouts. As a result of the navigation, same addresses and credit cards of the user have been listed according to how many times a user had navigated. To handle the this bug, I have implemented fragmentorder.xml from scratch and added two recycler views, namely OrderAddressAdapter.kt and CreditCardAdapter.kt with their implementation of view holders, namely cardcreditcard.xml and and cardaddress.xml.

Burak Çetin	Implemented search/products, search/vendors, search/brands endpoints. Helped with changes to database models that were required for these endpoints.
Furkan Nane	Changed the product detail endpoint according to the needs, changed the database according to the needs, implemented the endpoints for getting categories and their subcategories, getting orders of a vendor, and setting order status.
Hande Şirikçi	Implemented the order completion functionality by choosing payment and address options, designed a user friendly credit card form which shows the credit card while user types the fields of it. Designed and implemented adding address page. Implemented the improved version of the profile page by adding user information sections which are: Personal Info, Bank Account Info, Orders Info and Address Info. Implemented all fragments for each info page. For bank accounts and address pages, implemented adding and editing functionalities by adding floating action buttons. Also added functionalities for user to add new cart or address while making payment. Orders page designed and implemented as dynamically updatable according to orders added and delivered to user. Redesigned and implemented the horizontal product cart view and vertical product cart view in order to show elements properly. Implemented the adapter and view model of the subcategory page. Reviewed most of the pull requests and created related issues before creating the pr of a specified task. Attended all meetings both as a whole team and Android team.
Mehdi Saffar	Help design the new homepage, product page and checkout page, change layout of homepage (move search bar to header, move category bar under nav bar), work on product header (image gallery, product details, add to list component), integrate all product page components into one single coherent page, meet with backend to fix inconsistencies in endpoint structure and json results, meet with Burak Çetin to decide on search products/vendor/brands endpoint, meet with Furkan Nane to end the confusion related to static/dynamic categories/subcategories, make search endpoint work with backend, help fix product cart style issues related to overflow, add mock proxy for requests so that we can implement features easily without depending on backend, make some frontend routes only accesible to specifc user types, fix race condition between init and other api calls that depend on it, implement a better token management solution, add loading splash while initing, work with Bekir Yıldırım on the necessary CI/CD changes required to make email verification work, build orders components with Özdeniz Dolu, integrate add review PR, Özdeniz Dolu and I had to take upon ourselves to finish the profile page PR and best sellers PR since we noticed no activity from its author, do frequent meetings with frontend team to review code or help with issues and review pretty much all the frontend PRs and a couple backend PRs until requested changes are made and get merged. Also I wrote the milestone 2 scenario with Aslı Aykan
Mehmet Umut Öksüz	I am a member of backend team, my all code work I explained is regarding backend side of our project. I have implemented reviewing system for both products and vendors. In our application, customers can only review products that they've bought before and vendors that they have bought a product from. To achieve this, helper functions are created. Reviews can be added deleted and filtered by products and vendors. For vendors, I have implemented login and signup endpoints, vendors are required to provide an address additionally when registering, I also needed to interact with address tables. I have implemented vendor product management endpoints. Vendors can add products to their shop, update and delete their products. Since vendor can add images to product, mechanism to store images at backend and provide images to fronend are created. In last days, due to a database change, almost half of our unit tests started crashing, I have fixed all of them. Unit tests are written for all endpoints created by me. And I have reviewed almost all pull requests of the backend team. For all endpoints I have implemented, input validation is created.

Özdeniz Dolu	I have created the checkout page and all the components in it. In detail, it contains a credit card section, address section, a terms and conditions and a price summary section with a checkout button. In the credit card section, I have implemented list credit cards, edit credit card info, add a new card, delete a card features. All of which have been animated using a third party React library for better user experience. Similarly in the address section, I have implemented list addresses, edit address info, add a new address and delete an address features. Then, I finished the page by adding terms and conditions, price summary box, checkout button and full backend integration. All throughout the checkout page implementation process, I was in communication with Mehdi Saffar, Ömer Faruk Deniz and Bekir Yıldırım. After this page, Mehdi and I have created the component that shows past orders to the customer. It allows features such as show past orders, cancel an order, show order status etc. Other than these, I have reviewed a lot of PRs related to my team. There were many refactoring and bug fixing pull requests in which I did not directly written a lot of code but have commented and reviewed on what they are doing. We have had long online meeting sessions with Mehdi in which we have worked on many different aspects of the project, overdue features, bugs, new components. Out of those long meetings: We have decided to use Mock Service Worker to mock frontend data in case backend is not ready or testing purposes. We have completed two components that was assigned to Furkan Varol but never completed until 1 day left to the milestone 2 presentation. He has some code in those pages but it was far from finished so we had to finish it. Those components are horizontal product list that shows "best sellers in a category" and profile details page for customers which contains my addresses, my cards, my profile details etc. We have fixed a lot of buggy stuff and improved many aspects of the frontend aesthetically. All of the above changes are scattered throughout many issues and PRs.
Abdullah Yıldız	I have refactored the Product Card component to reflect the changes in our database modal fields. Also, redesigned the product card component to solve bugs like overflowing price and spoiling layout, to add new functionality such as discounted price, fixed-sized image and to create more modern-looking component. I have created User Reviews component to display the comment, the date of the comment, rating and the name of commenter. Pagination module is also implemented to show certain size of reviews. Later, I removed the pagination since backend decided to implement pagination in Milestone 3. I have refactored the Shopping Cart page, changed the file structure, fixed the bugs resulting from database changes. I have rewritten the endpoints and backend calls of update, delete and add operations of cart items. I have worked for new shopping cart displaying when hovered on shopping cart icon in header. However, I could not finish it and leave it to Milestone 3. I have worked in Email Verification with Bekir from backend. We have brainstormed about the flow concerning about the UX, created two scenarios and designed the one which suits our requirements. I have created verification page to display the response from BE. I have put a 'Send verification email' button in the profile page and shown appropriate notifications. I have requested backend to add new 'is verified' field in the customer model. Then, arranged 'init' and 'login' endpoints to set this new field. I have created an alert prompting the unverified users to verify their profile, arranged the alert and button in profile to not show when customer is verified. I have solved some bugs in the application like 'not entering the shopping cart after logging in' and 'not entering categories' I have reviewed Pull Requests 312, 298, 257, 239, 227
Bekir Yıldırım	I have implemented checkout endpoints. By this endpoints, customer can see checkout details, give order and also cancel order. In our design, non-verified user cannot pass payment page. With the Furkan Nane, we have changed database table that the result in adding order table and removing some fields rom purchase table. So, I have implemented non-verified user check. To test my endpoints, I have created CheckoutTest TestCase. I have worked in Email Verification with Abdullah Yıldız from frontend. We have brainstormed about the flow concerning about the UX, created two scenarios and designed the one which suits our requirements. According to our design, I have implemented two endpoints to verify email. I researched how to send email in Django. I put the required email backend, email host, ssl, tls etc. values in the settings.py. I have created new Gmail account to send verification mail. Each verification mail includes html templates and url which contains token expired two days. To create the token, I wrote helper functions. With the help of Mehdi Saffar, we put necessary our gmail account information in CI/CD side. Finally, I have implemented customer order. To return all orders of one customer properly, I have wrote serializer and also to make showing price easier for the frontend team, response contains total prices.

Aslı Aykan	After Milestone 1, I firstly dealt with the navigation and toolbar with back button issues which should be handled properly for the project to continue. Additionally, I added all the remaining endpoint functions from Milestone 1 (except register). According to the feedback, I decided to add a switch language button on the login page -for English and Turkish-, which works for all button title names and dialog strings. I refactored the SubCategoryFragment to make it a grid layout, then added the filter and sort options on it. I created the FilterOptionsFragment from scratch (with a price range, rating, brands, and vendors options). I made research about the MaterialDialog library to use it on our app with different dialog types for a better and realistic user experience. After the research, I added three types of dialog builder functions on Utils.kt which are for confirmation, information and success, then used them to confirm log out and give warning on false login attempt. I continued to add them wherever they are necessary as the project progressed, and I also handled the necessary navigations after the dialog is closed. For a better user experience, I also made research about fragment animations and added 4 animation resource files, then used them on some of the navigations, as many e-commerce apps use. I added the app icon, which is our logo. After the new endpoints are created, I created more than 40 data classes from scratch and updated some of the existing classes. For the vendor option, I redesigned the toolbar and bottom navigation bar, and created 5 fragments with their initial design and necessary viewmodels from scratch which are VendorHomeFragment, VendorProfileFragment, MessagesFragment, VendorOrdersFragment and UpdateProductFragment. For these fragments, I created 3 adapters which are VendorOrdersAdapter, VendorHomeBrandsAdapter and VendorHomeProductsAdapter. I created the product layout and brand layout for vendor, but I tried to use a similar product layout as in customer option. I added the remember me checkbox to the login page which saves the previous user data for login. I refactored the login endpoint to fetch the user object and its type. I examined the Aleyna's code for Google sign in and made some improvements on it, but we postponed its backend connection to the Milestone 3. I added a transparent loading animation for a better user experience, which is visible while fetching the API data. I added all the endpoints functions(more than 20) with their necessary mutable live datas that we decided to use for Milestone 2 and moved them to their corresponding view models. Except for the home and product page related endpoints, I also connected all the endpoint functions with the buttons and related layouts. I created UpdateAddressFragment and UpdateCreditCardFragment from scratch. I created 3 SwipeToDelete classes for delete endpoints. I improved the design of AddAddressFragment, AddCreditCardFragment, AddressFragment and BankAccountFragment. I merged all the commits for Android while also resolving all the conflicts. I reviewed all of the pull requests for Android and solved not only my bugs but also some bugs of my friends' while reviewing. I gave importance to the structure and readability of the project, so I created necessary folders, refactored the code for indentation, removed unnecessary code and mock data and renamed many classes and folders when necessary as the project progressed.
Murat Can Bayraktar	I have designed the Product Page as XML file. Later, my design is changed with a modern design.
Ömer Faruk Deniz	For this milestone, I have implemented new endpoints, and refactored the ones from Milestone 1 bringing all of them into the same coding and commenting convention. When it comes to new features, I implemented Card and Address endpoints for 5 request types(get, get all, post, put, delete) and implemented the Shopping Cart Item endpoint from scratch to make it compatible for the new convention so it is expanded from supporting 2 request types(get all, post) to 5 request types(get, get all, post, put, delete). I reimplemented the ProductResponseSerializer to make it compatible for the new requests of the frontend team. I made security improvements to all of the endpoints I have implemented from the beginning of the semester. At the end, I opened new PR's for commenting and refactoring all of my endpoints in great detail. All of my endpoints have unit tests for each request type and they are also detailly commented. I actively participated in the meetings and PR's of the backend team.

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

Author: Mehdi Saffar

Most of the CI/CD pipeline was well setup from the previous milestone. The only change was for the backend code.

Since the email verification endpoint was implemented, around 5 more environment variables needed to be added. Since backend does not use .env files running the server started to be cumbersome since all of the environment variables needed to passed through the command line.

With the help of Bekir Yıldırım, we added two files .env.production and .env.development into the root of backend folder. These files would be .gitignored and passed through Slack.

I adapted the code of settings.py to look for the required environment variables in the following order:

- Attempt to get the value from environment variable
- Attempt to get the value from .env file
- Attempt to get the value from default
- If the environment variable is required throw an exception otherwise return None

I added scripts that would

- Copy .env.production to unencrypted-secrets/ folder
- Encrypts files to secrets/ folder
- During CI/CD, copy .env.production back to backend
- Include the file in the Docker image

This not only made it easier to deploy but also to quickly share environment variables by just storing them in a file.

6 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

- These are the CRUD operations for the address object
  - get /customer/customerid/addresses
  - post /customer/customerid/addresses
  - get /customer/customerid/addresses/addressid
  - put /customer/customerid/addresses/addressid
  - delete /customer/customerid/addresses/addressid
- These are the CRUD operations for the credit card object
  - get /customer/customerid/cards
  - post /customer/customerid/cards
  - get /customer/customerid/cards/cardid
  - put /customer/customerid/cards/cardid
  - delete /customer/customerid/cards/cardid
- get /categories: Returns the categories and their subcategories
- These are the CRUD operations for the review object
  - get /review : gets all review items
  - post /review : post a new review
  - delete /review : delete a review
- These are the CRUD operations for the product object of a vendor
  - put /vendor/product : updates the product of a vendor
  - post /vendor/product : adds a new product to the shop a vendor
  - delete /vendor/product : deletes a product of a vendor
- get /checkout/details: get prices of a shopping cart
- post /checkout/payment: put shopping cart items into purchase and order table
- post /checkout/cancelorder/id: cancel specific order
- get /customer/orders: get orders of a customer

7 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 Shopping Cart	12/10/20 8:00	12/16/20 17:00	Backend Team
Implementing Order Mechanism	12/18/20 8:00	12/24/20 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 Shopping Cart	12/10/20 8:00	12/16/20 17:00	Android Team
Implementing Order Mechanism	12/18/20 8:00	12/24/20 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 Shopping Cart	12/10/20 8:00	12/16/20 17:00	Frontend Team
Implementing Comment Mechanism	12/03/20 8:00	12/10/20 17:00	Frontend Team
Implementing Order Mechanism	12/18/20 8:00	12/24/20 17:00	Frontend Team
Milestone 2 - 12/29/20			

Name	Start Date	Finish Date	Assignee Group
Implementing Recommendation Mechanism	01/06/21 8:00	01/15/21 17:00	Backend Team
Implementing Notification-Alarm	01/06/21 8:00	01/15/21 17:00	Backend Team
Implementing Lists	01/06/21 8:00	01/15/21 17:00	Backend Team
Implementing Direct Message Mechanism	01/06/21 8:00	01/15/21 17:00	Backend Team
Implementing Recommendation Mechanism	01/06/21 8:00	01/15/21 17:00	Android Team
Implementing Vendor Functionalities	01/06/21 8:00	01/15/21 17:00	Android Team
Implementing Notification-Alarm	01/06/21 8:00	01/15/21 17:00	Android Team
Implementing Lists	01/06/21 8:00	01/15/21 17:00	Android Team
Implementing Direct Message Mechanism	01/06/21 8:00	01/15/21 17:00	Android Team
Implementing Recommendation Mechanism	01/06/21 8:00	01/15/21 17:00	Frontend Team
Implementing Vendor Functionalities	01/06/21 8:00	01/15/21 17:00	Frontend Team
Implementing Notification-Alarm	01/06/21 8:00	01/15/21 17:00	Frontend Team
Implementing Lists	01/06/21 8:00	01/15/21 17:00	Frontend Team
Implementing Direct Message Mechanism	01/06/21 8:00	01/15/21 17:00	Frontend Team
Test Deployment	01/18/21 8:00	01/18/21 17:00	Whole Team
Milestone 3 - 01/19/21			

8 User Scenarios Presented

A university student breaks her phone so she decides to visit Getflix to get herself a new phone. Firstly she verifies her account in order to be able to by things. She has a budget of 12.000 TL. She searches for the phone and finds a suitable one. She looks the description and the reviews, and then she decides to buy the phone. She adds the phone to her shopping cart, and proceeds to the checkout page. She adds a credit card and agrees to the terms. She successfully checks out. Then she realizes she forgot to buy a phone case. From her phone she buys a new phone case. Then a week passes and she receives the phone without a problem. She comments on the product.

9 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, 161 PRs have been opened since the beginning of the project.

PS: Mostly taken from previous milestone report since the workflow didn't change

10 Evaluation of tools and managing the project

10.1 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.

10.2 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.

10.3 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.

10.4 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.

10.5 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.

10.6 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.

10.7 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.

10.8 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.

10.9 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.

10.10 Github Projects

Author: Özdeniz Dolu

As group 2, we have decided to use GitHub Projects to manage our progress/issues/PRs in milestone 2. GitHub projects allows us to create a dashboard with different column labels such as "to do", "in progress", "review in progress", "review approved" and "done". Using these cards, we can track the progress easily. Since it is integrated in the GitHub, it allows us to access everything about our project status in a single page therefore it is convenient. Since it is automated, we also do not have to manually track which PR's and issues are closed as they are automatically put into the "Done" section after they are closed. As a group, we have decided to create 3 GitHub projects called m2-frontend, m2-backend and m2-android in order to separately track each team's progress. As for the frontend team, we have been really satisfied with this tool. We are probably going to use it for milestone 3.

10.11 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.

10.12 Android Studio

Author: Murat Can Bayraktar

Android Team has used Android studio throughout semester. Android Studio is the official IDE developed by Google and later collaborated with JetBrains. Android Studio has a built-in Version Control System(VCS) and this helps the project to be up-to-date for all team members simultaneously. Android Studio also has a built-in emulator which we can choose which APK and Android phone model to run the code and visualize the app. The emulator also makes recording and sharing small feature implementations easier. The records could be attached to the related issue and the issue owner gets opportunity to serve the actual footage to the whole team.

11 Assessment of the customer presentation

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

- We have asked about the direct message functionality. We are planning to implement it for milestone 3
- Our prices were written like 12000 TL but it was a little bit hard to read so we are recommended to write it 12.000 TL
- Vendor pages were not fully ready so we are planning to implement them for milestone 3.
- If a product is at cargo status but it is not delivered for a long time, another status is needed.

Overall, the presentation for the milestone 2 went very well and customers were very happy and satisfied about it. This encouraged us to work harder for milestone 3.

12 Unit Test

12.1 Mehmet Umut Öksüz

1- Commit SHA: 17d4633a0cf52a5d1e0697dcf009203f7583fe08

- 2 tests for vendor signup, one with a valid request and one with a missing address field at the request body
- 4 tests for vendor managing products
  - Adding a valid product to system
  - Updating product with a valid request
  - Trying to update other vendor's product (expected to fail and return 403 FORBIDDEN)

  - Deleting a product

2- Commit SHA: 76f7dae9686bae29856b75c9f9cc1bfa7966d5e7

- Tests for reviewing system
  - Adding a review to a previously purchased product (expected to return successful)
  - Adding a review to a product that is not purchased (expected to return insuccessful)
  - Adding multiple reviews to a previously purchased product (expected to return insuccessful)(Same customer can review a product only once)
  - Adding a review to a vendor purchased from (expected to return successful)
  - Adding a review to a vendor not purchsed from (expected to return insuccessful)
  - Adding multiple reviews to a vendor purchased from (expected to return insuccessful) (Same customer can review a vendor only once)
  - Deleting a review owned by user (expected to return successful)

12.2 Burak Çetin

1- Commit SHA: 40bb9b4eefe78139b60377f948da9fedcfa81fa1
Tests for search endpoints under the tags 'search' and 'search/' for each endpoint

- products
  - Check if all products are returned under no filtering
  - Check subcategory filtering
  - Check brand filtering
  - Check sorting order
- brands
  - Check if all brands are returned under no filtering
  - Check subcategory filtering
- vendors
  - Check if all products are returned under no filtering
  - Check subcategory filtering
  - Check brand filtering

12.3 Bekir Yıldırım

1- Commit SHA: cbde99642a7e6d22ba3d435339f9459e5d5e3862
2- Commit SHA: 6e54a426f5bca60723c4164fb0c659d36da10047
Tests for checkout and customer order

- Check checkout summary prices

- Check non-verified user behaviour for payment
- Check payment endpoint
- Check cancel order endpoint
- Check purchase table not includes deleted product
- Check customer order endpoint

12.4 Ömer Faruk Deniz

1-Commit SHA: 5ca71df06dbd8fc7b537a2bd556fada161f58bc2

- 5 tests for all of the Address operations: GET, GET ALL, POST, PUT, DELETE
- 5 tests for all of the Card operations: GET, GET ALL, POST, PUT, DELETE
- 5 tests for all of the Shopping Cart Item operations: GET, GET ALL, POST, PUT, DELETE

2-Commit SHA: b25d614fe213e480f1adc832953caec2b7cb83ac

- Added detailed comments for the Address endpoint

3-Commit SHA: ba7f715f41e6329f8419925e9fdda30ed7c1e6aa

- Added detailed comments for the Card endpoint

4-Commit SHA: 853a39d3364a8685536d1f108da14f305a89a5a1

- Added detailed comments for the Shopping Cart Item endpoint

12.5 Furkan Nane

1-Commit SHA: ccaae27aaf1b13cb8a237e06b02b0cfe0d88c125

- This test is for checking whether the status of the order is changed after a request to the necessary endpoint