



# **CmpE 451 - Milestone 2 Report**

GROUP 1

January 5th, 2021

## **Table of Contents**

<b>Executive Summary</b>	<b>3</b>
Introduction	3
Road Ahead	3
<b>List and Status of Deliverables</b>	<b>4</b>
<b>Evaluation of the Status of Deliverables</b>	<b>5</b>
Backend	5
Android	5
Frontend	5
<b>Summary of Work Done by Each Member</b>	<b>6</b>
<b>Unit Tests Committed by Each Member</b>	<b>9</b>
<b>Challenges We Met</b>	<b>13</b>
6.1. Challenges We Met During Development	13
6.1.1. Challenges in Backend	13
6.1.2. Challenges in Frontend	14
6.1.3. Challenges in Android	14
6.2. Challenges We Met During Deployment	14
6.3. Challenges We Met During Dockerizing and CI/CD Processes	15
<b>API documentation</b>	<b>16</b>
<b>Project Plan</b>	<b>37</b>
<b>User scenarios</b>	<b>39</b>
9.1. Android	39
9.2. Frontend	39
<b>Code Structure</b>	<b>40</b>
When to open a new branch?	40
How to open a new branch?	40
When to commit?	41
How to commit?	41
Pull Requests	42
<b>Evaluation of Tools and Managing the Project</b>	<b>42</b>
General Tools	42
Backend Tools	42
Frontend Tools	43
Android Tools	43
Managing the Project	43
<b>Assessment of the Customer Presentation</b>	<b>44</b>
Issues Pointed out by the Team Members:	44

# 1. Executive Summary

## 1.1. Introduction

Tursu is our e-commerce platform which provides a lot more than just selling and buying products. Customers can make and view comments on the products to ease their decision making process. Our platform provides filtering and searching engines to make it very easy to find products. Customers can use their shopping lists to keep track of products they want, their shopping cart to add products they want to buy, notifications to set up alarms to keep track of the product prices. Our recommendation engine provides different products for each customer which makes the experience in Tursu individual.

For the second milestone, we decided to wrap up most of the functionalities on our platform to get feedback on them before the final milestone. Vendors now can add and edit products using our mobile or web application. We added shopping lists and the shopping cart to our interface where customers can keep the products they want to buy. We introduced new filtering and sorting functionalities which provides better experience than before. Most importantly now customers can actually buy products, vendors can verify the orders. We are now proposing more individual experiences, where users can edit their information after registration and customers can see recommended products we prepared just for them. Not but not least, we introduced the admin concept for this milestone, admin can verify products and vendors using the admin panel.

## 1.2. Road Ahead

Our customers' feedback is very important to us. Firstly, we will work on the feedback we got in our demo - which is shared in our repository. We already started fixing those issues. In parallel we are working on broadening the admin panel and wrapping up the commenting functionality. After that we will be implementing our notification engine and messaging interface. We will also set up the logging functionality. To make sure we are providing the best experience, we will be focusing more on the user experience and designs. We will continue using the technologies we are currently using and we will be implementing more broad tests for each part of the project.

## 2. List and Status of Deliverables

Deliverable	Status
<b>Backend</b>	
Shopping cart endpoints	Completed
Profile editing endpoints	Completed
Order endpoints	Completed
Login functionalities	Completed
Filtering & sorting functionalities	Completed
Semantic search	Completed
Vendor search	Completed
Product photo related endpoints	Completed
Shopping list endpoints	Completed
Recommendation endpoints	Completed
Google login and sign up endpoints	Completed
Delete product for admin	Completed
Product verification for admin	Completed
Customer page endpoint	Completed
Vendor page endpoint	Completed
Add comment functionalities	Completed
Ban user for admin	Completed
<b>Android</b>	
Sign in page	Completed
Sign up page	Completed
Order page	Completed
Filtering and Sorting page	Completed
Shopping cart page	Completed

Shopping lists page	Completed
Payment page	Completed
Product Display page	Completed
User Profile page	Completed
<b>Frontend</b>	
Sign In Page - Customer/Vendor	Completed
Sign Up Page - Customer/Vendor	Completed
Search	Completed
Filtering	Completed
Shopping Cart/List Page	Completed
Shopping Lists Page	Completed
User Profile Page - Edit Info	Completed
User Profile Page - My Orders	Completed
User Profile Page - My Lists	Completed
Admin Panel Basic Functionality (Page layouts, delete operations)	Completed
Advanced Admin Panel (With full listings and a proper UX/UI.)	Will be Completed in Milestone 3.
Payment Page	Completed
Order Page	Completed

### 3. Evaluation of the Status of Deliverables

#### 3.1. Backend

As the backend team, we have implemented most of the functionalities as planned for milestone 2. Also, we received some feedbacks during the customer

milestone presentation, and we will complete all planned tasks and suggested changes before the last milestone.

### 3.2. Android

As the android team, we have implemented most of the functionality as planned for the milestone 2. Some functionalities are still missing and being worked on, but the vast majority of the requirements are now fulfilled. Recommendation system is not implemented yet, but we will complete all of what we had planned to finish before the last milestone.

### 3.3. Frontend

As the frontend team, we have implemented most of the functionalities as planned for the milestone 2. However, we were not able to implement an advanced admin panel. It is going to be completed in the milestone 3. Also, there are some problems with UI/UX design of existing pages, and we will continue to polish them during the preparation of the milestone 3.

## 4. Summary of Work Done by Each Member

TEAM MEMBER	CONTRIBUTION
<b>Ali Batır</b>	All the following are implemented for the Android <ul style="list-style-type: none"><li>• Implemented adding a shopping list and removing a shopping list functionalities</li><li>• Implemented adding a product to the shopping list and removing a product from the shopping list functionalities</li><li>• Implemented displaying products that belong to a list functionality</li><li>• Implemented adding/deleting/updating a product functionalities</li><li>• Implemented displaying vendor products on sale functionality</li><li>• Implemented displaying comments functionality</li><li>• Refactored product detail page</li></ul>
<b>Asena Karolin Özdemir</b>	All the following are implemented for the Frontend <ul style="list-style-type: none"><li>• The sign in and sign up functionalities have been adjusted to match the changes in the endpoints.</li><li>• The sidebar for the customer and vendor profile pages is created and linked to the corresponding sections.</li><li>• The “My Information” section is created for both customer and vendor users. The functionalities that enable the users to view their information and change some of their information is created.</li><li>• The “My Orders” section is created for both customer and vendor users. The functionalities</li></ul>

	<p>that enable the users to view their orders and to change the status of their order is created.</p> <ul style="list-style-type: none"> <li>• The “My Products” page is created for the vendor users. The functionality that enables the users to view their products is created.</li> <li>• Created a dropdown menu in the navigation bar for the user to reach her profile page together with Barış Alhan.</li> <li>• Displayed the recommended products on the homepage together with Barış Alhan.</li> </ul>
<b>Baran Deniz Korkmaz</b>	
<b>Barış Alhan</b>	<p>All the followings are implemented for the frontend:</p> <ul style="list-style-type: none"> <li>• Created shopping cart and shopping list pages. Its generic components are widely used by other team members with small modifications, such as in order page, my products page, payment page and profile page.</li> <li>• Create a dropdown which enables adding to a shopping cart or a shopping list, or creating a new shopping list, by just clicking on a product in the homepage.</li> <li>• Implemented “My Lists” page in the profile page.</li> <li>• Contributed to “My Orders” page of vendors and customers, which is mainly implemented by Asena.</li> <li>• Made the backend connection of the payment page, which is mainly implemented by Yağız.</li> <li>• Created a dropdown menu in the navigation bar for the user to reach her profile page together with Asena.</li> <li>• Displayed the recommended products on the homepage together with Asena.</li> </ul>
<b>Barış Mutlu</b>	<p>All the following are implemented for the Android</p> <ul style="list-style-type: none"> <li>• Shopping cart interface. Listing products in there by dummy products. Buse added its Api Integration.</li> <li>• Payment Page and its input checks, when user tries to buy her products in shopping cart.</li> <li>• Payment Successful page when payment is done.</li> <li>• Transition between these 3 pages.</li> <li>• Customer Profile page API Integration.</li> </ul>
<b>Buse Kabakoğlu</b>	<p>All the following are implemented for the Android</p> <ul style="list-style-type: none"> <li>• Filtering and sorting interface design and API integration</li> <li>• Changing status of an order in customer (cancelled or delivered) interface and API are implemented</li> <li>• Customer giving an order only API integration</li> <li>• An order page interface is designed for vendors to see his or her orders and changing status to “in delivery” is implemented and integrated with API.</li> <li>• Adding to the shopping cart, removing from the shopping cart is integrated with APIs.</li> <li>• Refactoring the code after authentication was implemented by another teammate.</li> <li>• Searching for vendors interface and API integration.</li> </ul>
<b>Eray Sezgin</b>	<p>All the following are implemented for the Android</p> <ul style="list-style-type: none"> <li>• Fixes in login page</li> <li>• Signup pages for customer and vendors</li> <li>• Google map integration for location selection</li> </ul>

	<ul style="list-style-type: none"> <li>Entering the app without logging in/signing up as a guest</li> </ul> <p>I also did bug fixes for signup in the backend before being transferred to the Android team.</p>
<b>Mehmet Çelimli</b>	<p>All the following are implemented for the Frontend</p> <ul style="list-style-type: none"> <li>Filtering and sorting interfaces for search and category pages and API integrations.</li> <li>Filtering option includes filter by vendor, filter by category and price range. Sorting considers price, bestseller, rating, and commented features.</li> <li>Admin panel integration including ban user,, delete comment, verify or remove product and API integrations.</li> <li>Product not found page for non-existing products.</li> <li>Product detail page is refactored and comments are displayed.</li> <li>Sending multiple API request bug is fixed in the search page.</li> </ul>
<b>Murat Ekici</b>	<p>All the following things are implemented for the backend</p> <ul style="list-style-type: none"> <li>Implemented shopping cart functionalities.</li> <li>Implemented profile editing endpoints.</li> <li>Implemented new login functionalities.</li> <li>Implemented order endpoints including changing the order status and creating orders.</li> <li>Implemented helper endpoints for getting categories, brand names etc.</li> <li>Implemented unit tests for my endpoints.</li> <li>Worked on refactoring the database and added necessary columns when adding new endpoints.</li> <li>Communicated and worked with other teams to make sure everything works as expected.</li> <li>Reviewed 11 pull requests created by the backend team members.</li> </ul>
<b>Onur Kılıçoğlu</b>	<p>All the following things are implemented for the backend</p> <ul style="list-style-type: none"> <li>Refactored the database models and implemented them in the repo</li> <li>Implemented filtering and sorting functionalities for category and search endpoints</li> <li>Implemented product searching with filtering, sorting and semantic search</li> <li>Implemented vendor searching and its tests</li> <li>Implemented add, edit, delete product endpoints for vendors and its tests</li> <li>Implemented product photo related endpoints</li> <li>Implemented all the shopping list endpoints and its tests</li> <li>Implemented recommendation endpoints</li> <li>Implemented Google login and Google sign up endpoints</li> <li>Implemented delete product endpoint for admin and product verification</li> <li>Reviewed 8 pull requests created by the backend team members.</li> </ul>



<b>Ömer Ak</b>	<p>All the following things are implemented for the backend</p> <ul style="list-style-type: none"><li>• Implemented customer profile page endpoint</li><li>• Implemented vendor profile page endpoint</li><li>• Implemented comment addition endpoint</li><li>• Customer comments were added to the product pages</li><li>• Implemented ban user for admin</li><li>• Implemented delete product for admin</li><li>• Reviewed some of the codes of the other backend team members</li></ul>
<b>Yağız Can Çolak</b>	<p>All the following are implemented for the Frontend</p> <ul style="list-style-type: none"><li>• Adding a product page for vendors.</li><li>• Putting together the vendor profile page that Asena has implemented with product adding page.</li><li>• Checkout pages including address form page, credit card page, review of the ordered items page, and final review page.</li><li>• First draft of the Google Sign In functionality, further to be improved in milestone 3.</li><li>• Editing a product page for vendors.</li><li>• Deleting a product functionality for vendors.</li><li>• Assembling the editing and deleting a product features with the vendor profile page.</li><li>• Adding distant sales agreement to the payment page.</li></ul>

## 5. Unit Tests Committed by Each Member

### 5.1. Ali Batır

Since I was working in the android team and the functionalities I have implemented were mostly visual functionalities, I have been testing them by hand. Thus, I have not committed any unit tests to the repository.

### 5.2. Asena Karolin Özdemir

Since the functionalities I have implemented were mostly visual functionalities, I have been testing them by hand. Thus, I have not committed any unit tests to the repository.

### 5.3. Baran Deniz Korkmaz

### 5.4. Barış Alhan

Since I was working in the frontend team, I did user experience testing to see how first-time users respond to the UI. However, I did not write any unit tests.

### 5.5. Barış Mutlu

I tested my implementations manually because they are mostly about UI things. I tested them by different devices and different inputs but I have not committed any unit tests to the repository.

### 5.6. Buse Kabakoğlu

I was developing in the Android team. We tested our interfaces manually for different use cases and conditions. Therefore I have not committed any unit tests yet.

### 5.7. Eray Sezgin

I got transferred to the Android team from the backend team. I tested the Android app manually for different scenarios and screen sizes. But I didn't create any unit tests. My unit tests for the backend are below.

## Login Tests

<b>test_correct_username</b>	Tests logging in with correct username and password.
<b>test_correct_email</b>	Tests logging in with correct email and password.
<b>test_wrong_username</b>	Tests logging in with wrong username.
<b>test_wrong_email</b>	Tests logging in with wrong email.
<b>test_wrong_password</b>	Tests logging in with wrong password.

## Login View Tests

<b>test_correct_username</b>	Tests login request with correct username and password.
<b>test_correct_email</b>	Tests login request with correct email and password.
<b>test_wrong_username</b>	Tests login request with wrong username.
<b>test_wrong_email</b>	Tests login request with wrong email.
<b>test_wrong_password</b>	Tests login request with wrong password.
<b>test_empty_username</b>	Tests login request with empty username field.
<b>test_empty_password</b>	Tests login request with empty password field.

## Sign Up Tests

<b>test_customer_all_fields_provided</b>	Tests creating(signing up) a user of type customer with all fields provided.
<b>test_customer_missing_field</b>	Tests creating(signing up) a user of type customer with the "first_name" field missing.
<b>test_vendor_all_fields_provided</b>	Tests creating(signing up) a user of type vendor with all fields provided.
<b>test_vendor_missing_first_name</b>	Tests creating(signing up) a user of type vendor with the "first_name" field missing.

<b>test_vendor_missing_city</b>	Tests creating(signing up) a user of type vendor with the “city” field missing.
---------------------------------	---

## 5.8. Mehmet Çelimli

I was working in the frontend team and all the features I’ve implemented are tested via console or by using user scenarios I haven’t write any unit tests.

## 5.9. Murat Ekici

Tests for shopping cart endpoints in backend:

Commit: **"e1e6f940dc8c90ff237ad2054c095544d98bb563"**

<b>test_add_product</b>	Tests adding a product.
<b>test_delete_product</b>	Tests deleting a product.
<b>test_get_products_from_cart</b>	Tests adding a product and then getting it from the cart.
<b>test_missing_params</b>	Tests different endpoints with missing parameters.

Tests for order endpoints in backend:

Commit: **"ece5941faad381692df92eb69b6a76a585bd2802"**

<b>test_create_orders</b>	Tests creating orders from shopping cart.
<b>test_get_set_cancel_orders</b>	Tests creating orders, setting it's status to different status and getting created orders.

## 5.10. Onur Kılıçoğlu

Tests for search endpoint in backend:

Commit **"5ad4e2b07f58ed8d3bea840ce80c76c551f4887c"**

<b>test_searching_product_with_vendor_nam</b>	Tests searching for products with the
---	---------------------------------------

<b>e</b>	exact name of its vendor and checks if the products of the vendor returns in the response
<b>test_searching_product_with_category</b>	Tests searching for products with the exact name of its category and checks if the products of the category returns in the response
<b>test_searching_product_with_string</b>	Tests searching for products with the a query that is related with different products and checks if those related products returns in the response
<b>test_searching_product_with_vendor_name_case_insensitive</b>	Tests searching for products with the case insensitively written name of its vendor and checks if the products of the vendor returns in the response
<b>test_searching_product_with_category_case_insensitive</b>	Tests searching for products with the case insensitively written name of its category and checks if the products of the category returns in the response
<b>test_searching_nonexisting_product</b>	Tests searching for nonexistent product with a nonsense query and checks if any product is returned with this nonsense query searching for nonexistent product

Commit “**0d50ac674281265f1871218d988d2b1d5143de3c**”

<b>test_searching_vendor</b>	Tests searching for a vendor with the exact name and checks if the vendor returns in the response
<b>test_searching_vendor_case_insensitive</b>	Tests searching for a vendor with the case insensitively written name and checks if the vendor returns in the response
<b>test_searching_multiple_vendors</b>	Tests searching for multiple vendors with the same query and checks if those vendors are returned in the response
<b>test_searching_nonexisting_vendors</b>	Tests searching for nonexistent vendor with a nonsense query and checks if any vendor is returned with this nonsense query searching for nonexistent vendor

Tests for product/add endpoints

Commit “11938a7a5b597f488f2239e287bf83c5756a6de0”

<b>test_add_product_verified_vendor</b>	Tests if a verified vendor can add a valid product using the endpoint
<b>test_add_product_non_verified_vendor</b>	Tests if a nonverified vendor cannot add a valid product using the endpoint
<b>test_add_product_nonexisting_category</b>	Tests if a verified vendor cannot add an invalid product with a nonexisting category using the endpoint
<b>test_add_product_invalid_stock</b>	Tests if a verified vendor cannot add an invalid product with an invalid stock using the endpoint
<b>test_add_product_invalid_price</b>	Tests if a verified vendor cannot add an invalid product with an invalid price using the endpoint

Tests for product delete and edit endpoints:

Commit “338271bbbd47fb35dd370a23530e328e9d35e0b1”

<b>test_edit_product_verified_vendor</b>	Tests if a verified vendor can edit a product using the endpoint
<b>test_edit_product_non_verified_vendor</b>	Tests if a nonverified vendor cannot edit a product using the endpoint
<b>test_edit_product_invalid_product_id</b>	Tests if the edit endpoint can handle invalid product id.
<b>test_delete_product_invalid_product_id</b>	Tests if the delete endpoint can handle invalid product id.
<b>test_delete_product_valid_product_id</b>	Tests if the delete endpoint can delete a product with valid product id

Tests for shopping list endpoints:

Commit “a16491d6119e02f36fbe31991f62145bca0510ce”

<b>test_create_list</b>	Tests if a customer can create a list using the /shoppinglist/createlist/ endpoint
<b>test_delete_list</b>	Tests if a customer can delete a list using the /shoppinglist/deletelist/ endpoint
<b>test_add_product_to_list</b>	Tests if a customer can add valid product to a list of its own using the /shoppinglist/addtolist/ endpoint
<b>test_delete_product_from_list</b>	Tests if a customer can delete a product from a list of its own using the /shoppinglist/addtolist/ endpoint
<b>test_get_lists</b>	Tests if a customer can retrieve the lists of its own, using /shoppinglist/getlists/ endpoint
<b>test_get_products_from_list</b>	Tests if a customer can retrieve the products from a list of its own, using /shoppinglist/products/ endpoint
<b>test_missing_params</b>	Tests if all the shoppinglist endpoints can handle the missing parameter.

### 5.11. Ömer Ak

All the features that I have implemented were tested by hand. I have not committed any unit tests.

### 5.12. Yağız Can Çolak

All the features that I have implemented was tested by hand. I have not committed any unit tests.

## 6. Challenges We Met

### 6.1. Challenges We Met During Development

#### 6.1.1. Challenges in Backend

During the development of the backend application we encountered the following challenges.

1. The code that is developed in the local environment and the code that is deployed in AWS had different behaviour. We fixed the unexpected behaviour by user testing the API and by utilizing the feedback from Android and Frontend teams.
2. Using Postgresql in Windows created many significant problems that couldn't be solved.
3. Unverified data added before the features are implemented caused some errors in the functionalities which took some time to fix, however we corrected the data and made sure that this would not happen again.
4. The implemented tests were not working properly because of some errors caused by Postgre, so we individually fixed the problem using the Postgre command line.

### 6.1.2. Challenges in Frontend

During the implementation of the Frontend application, we encountered the following challenges.

1. We added the Google Sign in functionality which worked well while we were developing our project on our local host. However, the Google Sign in API only accepts requests that are sent from 'actual' urls. The localhost is considered an actual url however our deployment link is not. Since the functionality was working on our local, we did not realize this problem until our application was deployed. Thus, our deployed version of the project does not support the Google Sign in functionality.
2. We have used Material ui in our code. However material ui components do not recognize stylings that are written in separate css files. Thus, we had to include all stylings in the same Javascript file as the code of the components. This resulted in duplicate codes and a bad structural design of the code.
3. We have experienced some non-deterministic behaviour in our application. Some functionalities worked at one instance but did not work at another instance even though all observable variables were the same. Some of this behaviour could not be resolved.
4. We had a hard time dealing with life cycle functions. We did not have a lot of experience with such methods, thus we had to invest a lot of time to understand the working principle of these functions.

### 6.1.3. Challenges in Android

During the implementation of Android, we encountered the following challenges.

1. Integrating Google maps into the application was challenging due to the complication of the API.
2. While making transitions between fragments and activities, they are added into a stack and when the back button on the phone is pressed, it brings a fragment from the stack to the screen. We firstly did not consider this as a problem. But the more fragments added, the more problems surfaced. Therefore, we needed to fix these transition problems.
3. The layouts that we designed appeared with different portions in different phones, we needed to make it flexible.



4. The emulator in Android Studio is not so stable. It may not connect the internet and crashes in some situations. This is a serious problem for us. The emulator needs to be downloaded again, even though this may not solve the problem

## 6.2. Challenges We Met During Deployment

During the deployment of the backend part of the project we encountered one significant challenge which is performing migrations of the database after the models are changed. The Django migration did not work properly so to solve this problem we manually edited the database from the command line. During the deployment of the frontend we did not encounter any issues. The deployment part was significantly easier than the first milestone since our dockerized system from the first milestone can be deployed with only 3 commands.

## 6.3. Challenges We Met During Dockerizing and CI/CD Processes

Since we completed the dockerization in the first milestone and the CI/CD pipeline will be done in the last milestone we did not meet any challenges for these parts.

## 7. API documentation

### ["/search/"](#)

Search endpoint is used to search the products or vendors in the system.

Method: **GET**

Parameter	Description	Required/ Optional
<b>search_string</b>	string: String to be used in searching the products or vendors	Required
<b>search_type</b>	string: "vendor" for searching vendors, "product" for searching products in the system (Note: In the scope of this milestone you can only use product searching)	Required
<b>frating_gte</b>	int: Filter parameter for filtering products with ratings greater than or equal to the given value	Optional
<b>fprice_lower</b>	float: Filter parameter for filtering products with price greater than or equal to the given value, lower limit to the price	Optional
<b>fprice_upper</b>	float: Filter parameter for filtering products with price less than or equal to the given value, upper limit to the price	Optional
<b>fvendor_name</b>	string: Filter parameter for filtering products by the vendors	Optional
<b>fbrand</b>	string: Filter parameter for filtering products by the brands	Optional
<b>fcategory</b>	string: Filter parameter for filtering products by the categories	Optional
<b>sort_by</b>	=bestseller: sorting by best seller =newest: sorting by newest arrival =priceAsc: sorting by price in ascending order =priceDesc: sorting by price in descending order =numComments: sorting by number of comments	Optional

**Example query:**

[http://3.232.20.250/search?search\\_string=apple&search\\_type=product](http://3.232.20.250/search?search_string=apple&search_type=product)

**Response:**

**JSON object:** List of products with product\_info if search\_type is product  
List of vendors with vendor\_info if search\_type is vendor

**Notes:**

- If you don't use the filtering type, you should not use the parameter in the request

## "/product/"

Product page endpoint is used to see the product with a given ID.

Method: GET

Parameter	Description	Required/Optional
id	int: ID of the product	Required

Example query:

<http://3.232.20.250/product?id=1>

Response:

JSON object: detailed\_product\_info

Notes:

- If you don't use the filtering type, you should not use the parameter in the request

---

## "/product/category/"

Category endpoint is used to see the products in the category.

Method: GET

Parameter	Description	Required/Optional
name	string: Name of the category	Required
frating_gte	int: Filter parameter for filtering products with ratings greater than or equal to the given value	Optional
fprice_lower	float: Filter parameter for filtering products with price greater than or equal to the given value, lower limit to the price	Optional
fprice_upper	float: Filter parameter for filtering products with price less than or equal to the given value, upper limit to the price	Optional
fvendor_name	string: Filter parameter for filtering products by the vendors	Optional
fbrand	string: Filter parameter for filtering products by the brands	Optional
fcategory	string: Filter parameter for filtering products by the categories	Optional
sort_by	=bestseller: sorting by best seller =newest: sorting by newest arrival =priceAsc: sorting by price in ascending order =priceDesc: sorting by price in descending order	Optional

	=numComments: sorting by number of comments	
--	---	--

**Example query:**

<http://3.232.20.250/product/category?name=Home>

**Response:**

**JSON object:** List of products with product\_info

**Notes:**

- If you don't use the filtering type, you should not use the parameter in the request

---

## "/product/add/"

Product adding endpoint is used by vendors to add a new product.

**Method:** POST

Parameter	Description	Required/Optional
name	string: Name of the product to be added	Required
category	string: Name of the category of the product to be added	Required
description	string: Description of the product to be added	Required
brand	string: Brand of the product to be added	Required
stock	int: Stock of the product to be added	Required
price	float: Price of the product to be added	Required
photo	file: Primary photo of the product to	Optional

**Example query:**

<http://3.232.20.250/product/add/>

**Response:**

**HTTP Response:** Success or fail messages

**Notes:**

- You should give the parameters in the form data.
  - Auth token should be added to the header of the request as:  
"Authorization: 'Token <auth\_token>'" where key is Authorization and value is string with the word "Token" and the actual auth\_token separated with a whitespace
-

## "/product/edit/"

Product editing endpoint is used by vendors to edit a product of their own.

Method: **POST**

Parameter	Description	Required/Optional
id	int: ID of the product to be edited	Required
name	string: New name of the product to be edited	Optional
category	string: New name of the category of the product to be edited	Optional
description	string: New description of the product to be edited	Optional
brand	string: New brand of the product to be edited	Optional
stock	int: New stock of the product to be edited	Optional
price	float: New price of the product to be edited	Optional
photo	file: Optional photo of the product to be edited	Optional

Example query:

<http://3.232.20.250/product/edit/>

Response:

**HTTP Response:** Success or fail messages

Notes:

- You should give the parameters in the form data.
- Auth token should be added to the header of the request as:  
"Authorization: 'Token <auth\_token>'" where key is Authorization and value is string with the word "Token" and the actual auth\_token separated with a whitespace

---

## "/product/delete/"

Product deleting endpoint is used by vendors to delete a product of their own by its id.

Method: **POST / DELETE**

Parameter	Description	Required/Optional
id	int: ID of the product to be deleted	Required

Example query:

<http://3.232.20.250/product/delete/>

**Response:**

**HTTP Response:** Success or fail messages

**Notes:**

- Auth token should be added to the header of the request as:  
"Authorization: 'Token <auth\_token>'" where key is Authorization and value is string with the word "Token" and the actual auth\_token separated with a whitespace

---

## **"/product/addphoto/"**

Product photo adding endpoint is used by vendors to add a photo to a product of their own by its id.

**Method:** POST

Parameter	Description	Required/ Optional
id	int: ID of the product to be edited	Required
photo	file: Photo to be added to the product	Required

**Example query:**

<http://3.232.20.250/product/addphoto/>

**Response:**

**HTTP Response:** Success or fail messages

**Notes:**

- You should give the parameters in the form data.
- Auth token should be added to the header of the request as:  
"Authorization: 'Token <auth\_token>'" where key is Authorization and value is string with the word "Token" and the actual auth\_token separated with a whitespace

---

## **"/product/deleteallphotos/"**

Product delete all photos endpoint is used by vendors to delete all photos of a product of their own by its id.

**Method:** POST / DELETE

Parameter	Description	Required/ Optional
id	int: ID of the product to be edited	Required

**Example query:**

<http://3.232.20.250/product/deleteallphotos/>

**Response:**

**HTTP Response:** Success or fail messages

**Notes:**

- Auth token should be added to the header of the request as:  
"Authorization: 'Token <auth\_token>'" where key is Authorization and value is string with the word "Token" and the actual auth\_token separated with a whitespace

---

## **"/product/deletephoto/"**

Product delete photos endpoint is used by vendors to delete photos of a product of their own by its id and the url of the photo to be deleted.

**Method:** POST / DELETE

Parameter	Description	Required/ Optional
id	int: ID of the product to be edited	Required
photo_url	string: Url of the photo to be deleted.	Required

**Example query:**

<http://3.232.20.250/product/deletephoto/>

**Response:**

**HTTP Response:** Success or fail messages

**Notes:**

- Auth token should be added to the header of the request as:  
"Authorization: 'Token <auth\_token>'" where key is Authorization and value is string with the word "Token" and the actual auth\_token separated with a whitespace

---

## **"/shoppingcart/add/"**

Adds given product to the shopping cart with given quantity for the authenticated customer. If quantity is not given, then it is set to 1.

**Method:** POST

Parameter	Description	Required/ Optional
product_id	int: id of product to add	Required

quantity	int: quantity to be added	Optional
----------	---------------------------	----------

**Example query:**

<http://3.232.20.250/shoppingcart/add/>

**Response:**

**HTTP Response:** Success or fail messages

**Notes:**

- Auth token should be added to the header of the request as:  
"Authorization: 'Token <auth\_token>'" where key is Authorization and value is string with the word "Token" and the actual auth\_token separated with a whitespace

---

## **["/shoppingcart/increase/"](#)**

Adds given product to the shopping cart if it doesn't exist, else increases its quantity.

**Method:** POST

Parameter	Description	Required/Optional
product_id	int: id of product to increase	Required

**Example query:**

<http://3.232.20.250/shoppingcart/increase>

**Response:**

**HTTP Response:** Success or fail messages

**Notes:**

- Auth token should be added to the header of the request as:  
"Authorization: 'Token <auth\_token>'" where key is Authorization and value is string with the word "Token" and the actual auth\_token separated with a whitespace

---

## **["/shoppingcart/decrease/"](#)**

Deletes the given product to the shopping cart if it's quantity is 1, decreases the quantity otherwise.

**Method:** POST

Parameter	Description	Required/Optional
product_id	int: id of product to decrease	Required

**Example query:**

<http://3.232.20.250/shoppingcart/decrease>



**Response:**

**HTTP Response:** Success or fail messages

**Notes:**

- Auth token should be added to the header of the request as:  
"Authorization: 'Token <auth\_token>'" where key is Authorization and value is string with the word "Token" and the actual auth\_token separated with a whitespace

---

## **"/shoppingcart/delete/"**

Deletes the given product (all of them with the given product\_id) from the shopping cart for the authenticated customer.

**Method:** POST / DELETE

Parameter	Description	Required/Optional
product_id	int: id of product to delete	Required

**Example query:**

<http://3.232.20.250/shoppingcart/delete>

**Response:**

**HTTP Response:** Success or fail messages

**Notes:**

- Auth token should be added to the header of the request as:  
"Authorization: 'Token <auth\_token>'" where key is Authorization and value is string with the word "Token" and the actual auth\_token separated with a whitespace

---

## **"/shoppingcart/all/"**

Returns the shopping cart for the authenticated customer.

**Method:** GET

Parameter	Description	Required/Optional
-----------	-------------	-------------------

**Example query:**

<http://3.232.20.250/shoppingcart/all>

**Response:**

**JSON Response:** List of products in the cart with their quantities.

**Notes:**

- Auth token should be added to the header of the request as:  
"Authorization: 'Token <auth\_token>'" where key is Authorization and value is

string with the word "Token" and the actual auth\_token separated with a whitespace

---

### **"/shoppinglist/createlist/"**

Shopping list create list endpoint is used by customers to create a new shopping list with the given name

**Method:** POST

Parameter	Description	Required/Optional
list_name	string: Name of the new shopping list	Required

**Example query:**

<http://3.232.20.250/shoppinglist/createlist/>

**Response:**

**HTTP Response:** Success or fail messages

**Notes:**

- Auth token should be added to the header of the request as:  
"Authorization: 'Token <auth\_token>'" where key is Authorization and value is string with the word "Token" and the actual auth\_token separated with a whitespace
- 

### **"/shoppinglist/deletelist/"**

Shopping list delete list endpoint is used by customers to delete a shopping list of their own with the given name

**Method:** POST / DELETE

Parameter	Description	Required/Optional
list_name	string: Name of the shopping list	Required

**Example query:**

<http://3.232.20.250/shoppinglist/deletelist/>

**Response:**

**HTTP Response:** Success or fail messages

**Notes:**

- Auth token should be added to the header of the request as:  
"Authorization: 'Token <auth\_token>'" where key is Authorization and value is

string with the word "Token" and the actual auth\_token separated with a whitespace

---

### **"/shoppinglist/addtolist/"**

Shopping list add to list endpoint is used by customers to add a product to a shopping list with the given product id and list name

**Method:** POST

Parameter	Description	Required/Optional
list_name	string: Name of the shopping list	Required
product_id	int: ID of the product to be added to the list	Required

**Example query:**

<http://3.232.20.250/shoppinglist/addtolist/>

**Response:**

**HTTP Response:** Success or fail messages

**Notes:**

- Auth token should be added to the header of the request as:  
"Authorization: 'Token <auth\_token>'" where key is Authorization and value is string with the word "Token" and the actual auth\_token separated with a whitespace

---

### **"/shoppinglist/deletefromlist/"**

Shopping list delete from list endpoint is used by customers to delete a product from a shopping list with the given product id and list name

**Method:** POST / DELETE

Parameter	Description	Required/Optional
list_name	string: Name of the shopping list	Required
product_id	int: ID of the product to be deleted from the list	Required

**Example query:**

<http://3.232.20.250/shoppinglist/deletefromlist/>

**Response:**

**HTTP Response:** Success or fail messages

**Notes:**

- Auth token should be added to the header of the request as:  
"Authorization: 'Token <auth\_token>'" where key is Authorization and value is

string with the word "Token" and the actual auth\_token separated with a whitespace

---

### **"/shoppinglist/getlists/"**

Shopping list get lists endpoint is used by customers to retrieve all the lists they created before

**Method:** GET

Parameter	Description	Required/ Optional
-----------	-------------	-----------------------

**Example query:**

<http://3.232.20.250/shoppinglist/getlists/>

**Response:**

**JSON Response:** List of strings with the list names i.e. ["mylist1", "mylist2"]

**Notes:**

- Auth token should be added to the header of the request as:  
"Authorization: 'Token <auth\_token>'" where key is Authorization and value is string with the word "Token" and the actual auth\_token separated with a whitespace
- 

### **"/shoppinglist/products/"**

Shopping list get lists endpoint is used by customers to get all the products from a shopping list they created.

**Method:** GET

Parameter	Description	Required/ Optional
list_name	string: Name of the shopping list	Required

**Example query:**

<http://3.232.20.250/shoppinglist/products/>

**Response:**

**JSON Response:** List of products with product\_info

**Notes:**

- Auth token should be added to the header of the request as:  
"Authorization: 'Token <auth\_token>'" where key is Authorization and value is string with the word "Token" and the actual auth\_token separated with a whitespace
- 

### **"/order/create\_orders/"**

Creates the order from the customer's shopping list.

Method: **POST**

Parameter	Description	Required/ Optional
-----------	-------------	-----------------------

Example query:

[http://3.232.20.250/order/create\\_orders](http://3.232.20.250/order/create_orders)

Response:

**HTTP Response:** Success or fail messages

Notes:

- Auth token should be added to the header of the request as:  
"Authorization: 'Token <auth\_token>'" where key is Authorization and value is string with the word "Token" and the actual auth\_token separated with a whitespace

---

## **["/order/set\\_delivered/"](#)**

Set the given order as delivered. Authentication must belong to corresponding customer.

Method: **POST**

Parameter	Description	Required/ Optional
order_id	int: id of the corresponding order	Required

Example query:

[http://3.232.20.250/order/set\\_delivered](http://3.232.20.250/order/set_delivered)

Response:

**HTTP Response:** Success or fail messages

Notes:

- Auth token should be added to the header of the request as:  
"Authorization: 'Token <auth\_token>'" where key is Authorization and value is string with the word "Token" and the actual auth\_token separated with a whitespace

---

## **["/order/set\\_delivery/"](#)**

Sets the given orders as in delivery with estimated arrival time and cargo id, authentication must belong to the corresponding vendor.

Method: **POST**

Parameter	Description	Required/Optional
order_id	int: id of the corresponding order	Required
cargo_id	string: id of the cargo	Required
days	int: number of days till arrival	Required

**Example query:**

[http://3.232.20.250/order/set\\_delivery](http://3.232.20.250/order/set_delivery)

**Response:**

**HTTP Response:** Success or fail messages

**Notes:**

- Auth token should be added to the header of the request as:  
"Authorization: 'Token <auth\_token>'" where key is Authorization and value is string with the word "Token" and the actual auth\_token separated with a whitespace

## **"/order/cancel\_order/"**

Sets the given orders as cancelled.

**Method:** POST

Parameter	Description	Required/Optional
order_id	int: id of the corresponding order	Required

**Example query:**

[http://3.232.20.250/order/cancel\\_order](http://3.232.20.250/order/cancel_order)

**Response:**

**HTTP Response:** Success or fail messages

**Notes:**

- Auth token should be added to the header of the request as:  
"Authorization: 'Token <auth\_token>'" where key is Authorization and value is string with the word "Token" and the actual auth\_token separated with a whitespace

## **"/order/get\_orders/"**

Returns the orders of given user, inner lists are the orders in the same shopping cart.

Method: GET

Parameter	Description	Required/ Optional
-----------	-------------	-----------------------

Example query:

[http://3.232.20.250/order/get\\_orders/](http://3.232.20.250/order/get_orders/)

Response:

**JSON Response:** List of orders nested into the corresponding carts at the time.

Notes:

- Auth token should be added to the header of the request as:  
"Authorization: 'Token <auth\_token>'" where key is Authorization and value is string with the word "Token" and the actual auth\_token separated with a whitespace
- 

## **"/recommendation/recommended"**

Recommended endpoint is used by customers to customized recommendations

Method: GET

Parameter	Description	Required/ Optional
-----------	-------------	-----------------------

Example query:

<http://3.232.20.250/recommendation/recommended>

Response:

**JSON Response:** List of products with product\_info

Notes:

- Auth token should be added to the header of the request as:  
"Authorization: 'Token <auth\_token>'" where key is Authorization and value is string with the word "Token" and the actual auth\_token separated with a whitespace
- 

## **"/recommendation/bestseller"**

Bestseller endpoint returns 10 bestsellers of Tursu

Method: GET

Parameter	Description	Required/ Optional
-----------	-------------	-----------------------

**Example query:**

<http://3.232.20.250/recommendation/bestseller>

**Response:**

**JSON Response:** List of products with product\_info

**Notes:**

## **"/recommendation/toprated"**

Top rated endpoint returns 10 top rated products of Tursu

**Method:** GET

Parameter	Description	Required/ Optional
-----------	-------------	-----------------------

**Example query:**

<http://3.232.20.250/recommendation/toprated>

**Response:**

**HTTP Response:** List of products with product\_info

**Notes:**

## **"/recommendation/newest"**

Newesr endpoint returns 10 newest added products of Tursu

**Method:** GET

Parameter	Description	Required/ Optional
-----------	-------------	-----------------------

**Example query:**

<http://3.232.20.250/recommendation/newest>

**Response:**

**JSON Response:** List of products with product\_info

**Notes:**



---

## "/recommendation/recommendation\_pack"

Recommendation pack endpoint returns all recommendation types at the same request

Method: **GET**

Parameter	Description	Required/ Optional
-----------	-------------	-----------------------

Example query:

[http://3.232.20.250/recommendation/recommendation\\_pack](http://3.232.20.250/recommendation/recommendation_pack)

Response:

**JSON Response:** recommendation\_pack:

- **recommended** → List of products with product\_info
- **bestseller** → List of products with product\_info
- **top\_rated** → List of products with product\_info
- **newest\_arrival** → List of products with product\_info

Notes:

- Auth token should be added to the header of the request as:  
"Authorization: 'Token <auth\_token>'" where key is Authorization and value is string with the word "Token" and the actual auth\_token separated with a whitespace

---

## "/admin/verificationpending/products/"

Verification pending products endpoint is used by admin to get verification pending products

Method: **GET**

Parameter	Description	Required/ Optional
-----------	-------------	-----------------------

Example query:

<http://3.232.20.250/admin/verificationpending/products/>

Response:

**JSON Response:** List of products with product\_info

Notes:

- Auth token should be added to the header of the request as:  
"Authorization: 'Token <auth\_token>'" where key is Authorization and value is string with the word "Token" and the actual auth\_token separated with a whitespace
-

## "/admin/removeproduct/"

Product removing endpoint is used by admin to delete a product by its id.

Method: **POST / DELETE**

Parameter	Description	Required/ Optional
id	int: ID of the product to be deleted	Required

Example query:

<http://3.232.20.250/admin/removeproduct/>

Response:

**HTTP Response:** Success or fail messages

Notes:

- Auth token should be added to the header of the request as:  
"Authorization: 'Token <auth\_token>'" where key is Authorization and value is string with the word "Token" and the actual auth\_token seperated with a whitespace

---

## "/admin/verifyproduct/"

Product verification endpoint is used by admin to verify a product by its id.

Method: **POST**

Parameter	Description	Required/ Optional
id	int: ID of the product to be verified	Required

Example query:

<http://3.232.20.250/admin/verifyproduct/>

Response:

**HTTP Response:** Success or fail messages

Notes:

- Auth token should be added to the header of the request as:  
"Authorization: 'Token <auth\_token>'" where key is Authorization and value is string with the word "Token" and the actual auth\_token seperated with a whitespace

---

## "/admin/deletecomment/"

Comment deletion endpoint is used by admin to delete a comment by its id.

**Method:** POST / DELETE

Parameter	Description	Required/ Optional
id	int: ID of the comment to be deleted	Required

**Example query:**

<http://3.232.20.250/admin/deletcomment/>

**Response:**

**HTTP Response:** Success or fail messages

**Notes:**

- Auth token should be added to the header of the request as:  
"Authorization: 'Token <auth\_token>'" where key is Authorization and value is string with the word "Token" and the actual auth\_token seperated with a whitespace

---

## **["/admin/banuser/"](#)**

User banning endpoint is used by admin to ban a user by its username.

**Method:** POST

Parameter	Description	Required/ Optional
username	string: username of the user to be banned	Required

**Example query:**

<http://3.232.20.250/admin/banuser/>

**Response:**

**HTTP Response:** Success or fail messages

**Notes:**

- Auth token should be added to the header of the request as:  
"Authorization: 'Token <auth\_token>'" where key is Authorization and value is string with the word "Token" and the actual auth\_token seperated with a whitespace

---

## **["/user/signup/google"](#)**

Sign up with Google endpoint.

**Method:** POST

Parameter	Description	Required/Optional
token_id	string: Token ID of the Google token obtained from Google	Required
is_vendor	if customer leave it empty, if vendor any non empty value is evaluated as true.	Required
IBAN	string: IBAN of the vendor	Required for Vendor
latitude	float: Latitude of the location of Vendor	Required for Vendor
longitude	float: Longitude of the location of Vendor	Required for Vendor
city	string: City of the vendor	Required for Vendor

**Example query:**

<http://3.232.20.250/user/signup/google>

**Response:**

**JSON Response:** login\_info

**Notes:**

## "/user/login/google"

Login with Google endpoint

**Method:** POST

Parameter	Description	Required/Optional
token_id	string: Token ID of the Google token obtained from Google	Required

**Example query:**

<http://3.232.20.250/user/login/google>

**Response:**

**JSON Response:** login\_info

**Notes:**

## "/user/signup"

Sign up endpoint.

Method: **POST**

Parameter	Description	Required/ Optional
username	string: User's unique username.	Required
password	string: User's password.	Required
email	string: User's email address.	Required
first_name	string: User's first name.	Required
last_name	string: User's last name.	Required
IBAN	string: IBAN of the vendor	Required for Vendor
latitude	float: Latitude of the location of Vendor	Required for Vendor
longitude	float: Longitude of the location of Vendor	Required for Vendor
city	string: City of the vendor	Required for Vendor

Example query:

<http://3.232.20.250/user/signup>

Response:

**JSON Response:** login\_info

Notes:

---

## **"/user/login"**

Login endpoint

Method: **POST**

Parameter	Description	Required/ Optional
email	string: Email or username of the user	Required
password	string: Password of the user	Required

Example query:

<http://3.232.20.250/user/login>

Response:

**JSON Response:** login\_info

Notes:

---

## **"/helper/allbrands"**

Returns all existing brands in a list.

Method: GET

Parameter	Description	Required/ Optional
-----------	-------------	-----------------------

Example query:

<http://3.232.20.250/helper/allbrands>

Response:

**JSON Response:** string list

Notes:

---

## **"/helper/allvendors"**

Returns all verified vendors in a list.

Method: GET

Parameter	Description	Required/ Optional
-----------	-------------	-----------------------

Example query:

<http://3.232.20.250/helper/allvendors>

Response:

**JSON Response:** string list

Notes:

---

## **"/helper/allcategories"**

Returns all existing categories in a list.

Method: GET

Parameter	Description	Required/ Optional
-----------	-------------	-----------------------

Example query:

<http://3.232.20.250/helper/allcategories>

Response:

**JSON Response:** string list

Notes:

---

## "/customerpage"

Returns customer information.

Method: GET

Parameter	Description	Required/ Optional
-----------	-------------	-----------------------

Example query:

<http://3.232.20.250/customerpage>

Response:

JSON Response: customer\_info

Notes:

- Auth token should be added to the header of the request as:  
"Authorization: 'Token <auth\_token>'" where key is Authorization and value is string with the word "Token" and the actual auth\_token seperated with a whitespace

---

## "/vendorpage"

Returns vendor information.

Method: GET

Parameter	Description	Required/ Optional
-----------	-------------	-----------------------

Example query:

<http://3.232.20.250/vendorpage>

Response:

JSON Response: vendor\_info

Notes:

- Auth token should be added to the header of the request as:  
"Authorization: 'Token <auth\_token>'" where key is Authorization and value is string with the word "Token" and the actual auth\_token seperated with a whitespace

---

## DATA STRUCTURES

product\_info:

Field	Description
id	int: ID of the product
name	string: Name of the product

photo_url	string: Link to the photo of the product
vendor_name	string: Name of the vendor of the product
category	string: Name of the category of the product
rating	float: Rating of the product
stock	int: Available stock of the product
price	float: Price of the product
brand	string: Name of the brand of the product

#### detailed\_product\_info:

Field	Description
id	int: ID of the product
name	string: Name of the product
photo_url	string: Link to the photo of the product
vendor_name	string: Name of the vendor of the product
category	string: Name of the category of the product
rating	float: Rating of the product
stock	int: Available stock of the product
price	float: Price of the product
brand	string: Name of the brand of the product
comments	List of <b>comment</b>
all_photos	List of string: List of links to alternative photos of the product

#### login\_info:

Field	Description
auth_token	string: Auth token of the user
first_name	string: First name of the user
last_name	string: Last name of the user
user_type	string: "admin", "customer" or "vendor"

#### customer\_info:



Field	Description
username	string: User name of the customer
email	string: Email of the customer
first_name	string: First name of the customer
last_name	string: Last name of the customer
money_spent	float: Total amount of money spent of the customer
orders	order_info: Orders of the customer
lists	list of product: Product lists of the customer

#### order\_info:

Field	Description
vendor	string: User name of the vendor
product	int: ID of the product
status	string: Status of the order
cargoID	int: ID of the cargo
orderDate	date: Date of the order
estimatedArrivalDate	date: Estimated arrival date of the order
arrivalDate	date: Arrival date of the order
quantity	int: Order quantity
comment	string: Comment of the order

#### vendor\_info:

Field	Description
username	string: User name of the vendor
email	string: Email of the vendor
first_name	string: First name of the vendor
last_name	string: Last name of the vendor
attitude	string: Attitude of the location of the vendor

longitude	string: Longitude of the location of the vendor
iban	string: Iban of the vendor
rating	float: Rating of the vendor
orders	order_info: Orders of the vendor
products	product_info: Products of the vendor

**comment:**

Field	Description
id	int: ID of the comment
customer	string: Username of the customer commented
text	string: Text of the comment
rating	float: Rating given with the comment

## 8. Project Plan

		Name	Duration	Start	Finish	Predecessors	Resource Names
1		<b>Preparing Deliverables for Milestone 1 of 352</b>	<b>54 days?</b>	<b>2/19/20 8:00 AM</b>	<b>5/4/20 5:00 PM</b>		
2		<b>Requirements</b>	<b>35 days?</b>	<b>2/19/20 8:00 AM</b>	<b>4/7/20 5:00 PM</b>		
3		<b>The First Draft</b>	<b>14 days?</b>	<b>2/19/20 8:00 AM</b>	<b>3/9/20 5:00 PM</b>		
4		Distribute the requirements to the team members	1 day?	2/19/20 8:00 AM	2/19/20 4:00 PM		Onur Kılıçolu
5		Preparing the glossary	4 days?	2/27/20 8:00 AM	2/29/20 8:02 AM		Yaz Can Çolak
6		Reviewing the glossary	4 days?	3/4/20 8:00 AM	3/9/20 5:00 PM	5	Bar Alhan,Ömer Ak
7		<b>Functional Requirements</b>	<b>4 days?</b>	<b>2/19/20 8:00 AM</b>	<b>2/24/20 5:00 PM</b>		
8		User Requirements	4 days?	2/19/20 8:00 AM	2/24/20 5:00 PM		Ali Batr;Bar Alhan,Murat Ekici;Yaz ...
9		System Requirements	4 days?	2/19/20 8:00 AM	2/24/20 5:00 PM		Buse Kabakolu;Onur Kılıçolu;Ufuk K...
10		<b>Nonfunctional Requirements</b>	<b>4 days?</b>	<b>2/19/20 8:00 AM</b>	<b>2/24/20 5:00 PM</b>		
11		Availability	4 days?	2/19/20 8:00 AM	2/24/20 5:00 PM		Bar Mutlu
12		Security	4 days?	2/19/20 8:00 AM	2/24/20 5:00 PM		Asena Karolin Özdemir
13		Performance	4 days?	2/19/20 8:00 AM	2/24/20 5:00 PM		Mehmet Çelimli
14		Privacy	4 days?	2/19/20 8:00 AM	2/24/20 5:00 PM		Mehmet Çelimli
15		Prepare the final first draft of Requirements document	4 days?	2/19/20 8:00 AM	2/24/20 5:00 PM	10;7	Ömer Ak
16		Reviewing the Categorization	3 days?	2/27/20 8:00 AM	3/2/20 5:00 PM	10;7	Ömer Ak
17		<b>The second draft</b>	<b>5 days?</b>	<b>4/1/20 8:00 AM</b>	<b>4/7/20 5:00 PM</b>	<b>3</b>	
18		Update the Glossary	5 days?	4/1/20 8:00 AM	4/7/20 5:00 PM	10;7	Bar Alhan
19		Update Requirements	5 days?	4/1/20 8:00 AM	4/7/20 5:00 PM	7;10	Buse Kabakolu;Murat Ekici;Ufuk Kar...
20		Update the Categorization	10 days?	4/1/20 8:00 AM	4/7/20 5:00 PM	19	Buse Kabakolu;Ufuk Karagöz
21		<b>Scenarios &amp; Mockups</b>	<b>8 days?</b>	<b>2/27/20 8:00 AM</b>	<b>3/9/20 5:00 PM</b>	<b>3</b>	
22		<b>Preparing scenario and mockups</b>	<b>4 days?</b>	<b>2/27/20 8:00 AM</b>	<b>3/3/20 5:00 PM</b>		
23		Preparing the mockup designs of the pages shared by more than one scen...	2 days?	2/27/20 8:00 AM	2/28/20 8:01 AM		Onur Kılıçolu
24		Preparing the "Buying a Product" scenario	4 days?	2/27/20 8:00 AM	3/2/20 4:59 PM	23	Asena Karolin Özdemir;Ömer Ak;Ufuk...
25		Creating persona, story and actions for the first scenario	6.998 days?	2/27/20 8:00 AM	3/3/20 5:00 PM	24	Yaz Can Çolak;Asena Karolin Özde...
26		Preparing the "Cancelling an Order" scenario	5.002 days?	2/27/20 8:00 AM	3/3/20 5:00 PM	23	Bar Mutlu;Mehmet Çelimli;Buse Kab...
27		Creating persona, story and actions for the second scenario	8 days?	2/27/20 8:00 AM	3/3/20 5:00 PM	26	Buse Kabakolu;Bar Mutlu;Mehmet ...
28		Preparing the "Make Comment About a Delivered Product" scenario	5.002 days?	2/27/20 8:00 AM	3/3/20 5:00 PM	23	Bar Alhan;Murat Ekici;Ali Batr
29		Creating persona, story and actions for the third scenario	8 days?	2/27/20 8:00 AM	3/3/20 5:00 PM	28	Ali Batr;Bar Alhan;Murat Ekici
30		Reviewing the mockups	4 days?	3/4/20 8:00 AM	3/9/20 5:00 PM	22	Onur Kılıçolu;Yaz Can Çolak;Buse K...
31		<b>UML Designs</b>	<b>21 days?</b>	<b>3/11/20 8:00 AM</b>	<b>4/8/20 5:00 PM</b>	<b>2;21</b>	
32		<b>First Draft of the Design Documents</b>	<b>4 days?</b>	<b>3/11/20 8:00 AM</b>	<b>3/16/20 5:00 PM</b>		

Gantt Chart - page1

		Name	Duration	Start	Finish	Predecessors	Resource Names
33		Preparation of Class Diagram	3 days?	3/11/20 8:00 AM	3/13/20 5:00 PM		Bar Alhan;Buse Kabakolu;Mehmet ...
34		Preparation of Use Case Diagram	4 days?	3/11/20 8:00 AM	3/16/20 5:00 PM		Asena Karolin Özdemir;Yaz Can Çol...
35		Preparation of Sequence Diagram	8 days?	3/11/20 8:00 AM	3/16/20 5:00 PM	34;33	Bar Mutlu;Ali Batr;Onur Kılıçolu;Öm...
36		Revising Design Documents	6 days?	4/1/20 8:00 AM	4/8/20 5:00 PM	32	Ali Batr;Asena Karolin Özdemir;Bar ...
37		<b>Project Plan</b>	<b>5 days?</b>	<b>4/15/20 8:00 AM</b>	<b>4/21/20 5:00 PM</b>	<b>2;31;21</b>	
38		Moving the actions done in the notes of the previous meetings to the project...	3 days?	4/15/20 8:00 AM	4/18/20 5:00 PM		Ali Batr;Bar Mutlu;Buse Kabakolu;...
39		Thinking about how to organize milestones	3 days?	4/15/20 8:00 AM	4/18/20 5:00 PM		Onur Kılıçolu;Yaz Can Çolak;Buse K...
40		Revising the prepared project plan and make sure that they all are consistent	1 day?	4/18/20 8:00 AM	4/20/20 5:00 PM		Murat Ekici
41		Create and fill the Project Plan Document for Milestone 1	2 days?	4/18/20 8:00 AM	4/20/20 5:00 PM	39	Murat Ekici;Ömer Ak;Ufuk Karagöz;Ya...
42		Create and fill the Project Plan Document for Milestone 2	2 days?	4/18/20 8:00 AM	4/20/20 5:00 PM	39	Onur Kılıçolu;Mehmet Çelimli
43		Create and fill the Project Plan Document for Milestone 3	1 day?	4/18/20 8:00 AM	4/20/20 5:00 PM	39	Bar Alhan;Buse Kabakolu
44		Reviewing the project plan document and assuring consistency	1 day?	4/18/20 8:00 AM	4/20/20 5:00 PM	43;42;41	Asena Karolin Özdemir
45		Creating RAM document and making sure it is filled by members	2 days?	4/18/20 8:00 AM	4/21/20 5:00 PM	38	Ali Batr
46		Milestone 1 Report	6 days?	4/26/20 8:00 AM	5/4/20 5:00 PM	37;31;21;2	
47		Milestone 1	0 days?	5/4/20 5:00 PM	5/4/20 5:00 PM	1	Ali Batr;Asena Karolin Özdemir;Bar ...
48		<b>Preparing Deliverables for Milestone 2 of CMPE352</b>	<b>17 days</b>	<b>5/5/20 8:00 AM</b>	<b>5/27/20 5:00 PM</b>	<b>47</b>	
49		<b>Implementation of Practice App</b>	<b>13 days</b>	<b>5/5/20 8:00 AM</b>	<b>5/21/20 5:00 PM</b>		
50		Research about Available APIs on Web	1 day	5/5/20 8:00 AM	5/5/20 5:00 PM		Onur Kılıçolu;Yaz Can Çolak;Buse K...
51		Implementing functions for necessary APIs	4 days	5/6/20 8:00 AM	5/11/20 5:00 PM	50	Onur Kılıçolu;Buse Kabakolu;Ömer Ak
52		Back-End Implementation: Product Page	5 days	5/12/20 8:00 AM	5/18/20 5:00 PM	51	Onur Kılıçolu;Bar Alhan;Murat Ekici
53		Front-End Implementation: Product Page	5 days	5/12/20 8:00 AM	5/18/20 5:00 PM	51	Bar Mutlu;Asena Karolin Özdemir;Ali...
54		Back-End Implementation: Comment Page	5 days	5/12/20 8:00 AM	5/18/20 5:00 PM	51	Ufuk Karagöz;Ömer Ak;Mehmet Çelimli
55		Front-End Implementation: Comment Page	5 days	5/12/20 8:00 AM	5/18/20 5:00 PM	51	Yaz Can Çolak;Bar Alhan;Bar Mutlu
56		Testing the Practice App	3 days	5/19/20 8:00 AM	5/21/20 5:00 PM	52;53;54;55	Onur Kılıçolu;Yaz Can Çolak;Buse K...
57		Milestone 2 Report	4 days	5/22/20 8:00 AM	5/27/20 5:00 PM	56	Onur Kılıçolu;Yaz Can Çolak;Buse K...
58		Milestone 2	0 days?	5/27/20 5:00 PM	5/27/20 5:00 PM	48	Ali Batr;Asena Karolin Özdemir;Bar ...
59		<b>Decisions and Settings</b>	<b>5 days</b>	<b>11/2/20 8:00 AM</b>	<b>11/6/20 5:00 PM</b>		
60		Deciding on Tech Stack	1 day	11/2/20 8:00 AM	11/2/20 5:00 PM		Ali Batr;Asena Karolin Özdemir;Bar ...
61		Setting up the Environment	4 days	11/3/20 8:00 AM	11/6/20 5:00 PM	60	Ali Batr;Asena Karolin Özdemir;Baran...
62		<b>Creation of the User Database</b>	<b>5 days</b>	<b>11/9/20 8:00 AM</b>	<b>11/13/20 5:00 PM</b>		
63		Creating the User Database	5 days	11/9/20 8:00 AM	11/13/20 5:00 PM		Onur Kılıçolu;Murat Ekici;Ömer Ak;Er...
64		Deciding the Account Endpoints	1 day	11/9/20 8:00 AM	11/9/20 6:00 PM		Onur Kılıçolu;Murat Ekici;Ömer Ak;Er...

Gantt Chart - page2



		Name	Duration	Start	Finish	Predecessors	Resource Names
65		Implementing the Login Pages	4 days	11/10/20 8:00 AM	11/13/20 5:00 PM	64	Onur Kıpçolu,Murat Ekici,Eray Sezgin...
66		Implementation of Sign in / Sign up	5 days?	11/10/20 8:00 AM	11/16/20 5:00 PM		
67		Frontend Sign in page	5 days?	11/10/20 8:00 AM	11/16/20 5:00 PM		Yaz Can Çolak,Bar Alhan,Asena Ka...
68		Frontend Sign up page	5 days?	11/10/20 8:00 AM	11/16/20 5:00 PM		Asena Karolin Ozdemir,Bar Alhan,M...
69		Android Sign in page	5 days?	11/10/20 8:00 AM	11/16/20 5:00 PM		Ali Batr,Baran Deniz Korkmaz,Bar ...
70		Android sign up page	5 days?	11/10/20 8:00 AM	11/16/20 5:00 PM		Ali Batr,Baran Deniz Korkmaz,Bar ...
71		Internal Milestone	1 day?	11/16/20 8:00 AM	11/16/20 5:00 PM		
72		Implementation of Product Database	6 days?	11/16/20 8:00 AM	11/23/20 5:00 PM		
73		Deciding the Homepage Endpoints	1 day	11/16/20 8:00 AM	11/16/20 5:00 PM		Onur Kıpçolu,Ömer Ak,Eray Sezgin,M...
74		Front-End Implementation: Homepage	4 days	11/17/20 8:00 AM	11/20/20 5:00 PM	73	Bar Alhan[150%],Yaz Can Çolak,As...
75		Back-End Implementation: Homepage	4 days	11/17/20 8:00 AM	11/20/20 5:00 PM	73	Onur Kıpçolu,Murat Ekici,Ömer Ak,Er...
76		Android Implementation: Homepage	4 days	11/17/20 8:00 AM	11/20/20 5:00 PM	73	Buse Kabakolu,Bar Mutlu,Baran De...
77		Frontend Implementing Product Page	2 days?	11/20/20 8:00 AM	11/23/20 5:00 PM		Asena Karolin Ozdemir,Bar Alhan,M...
78		Backend Implementing Product Page	2 days?	11/20/20 8:00 AM	11/23/20 5:00 PM		Eray Sezgin,Murat Ekici,Onur Kıpçolu...
79		Android Implementing Product Page	2 days?	11/20/20 8:00 AM	11/23/20 5:00 PM		Ali Batr,Baran Deniz Korkmaz,Bar ...
80		Deciding the Product Endpoints	1 day	11/20/20 8:00 AM	11/20/20 5:00 PM		Onur Kıpçolu[50%],Eray Sezgin,Murat...
81		Customer Milestone 1 for CMPE451	1 day?	11/24/20 8:00 AM	11/24/20 5:00 PM		
82		Implementation of Order Database	5 days	11/24/20 8:00 AM	11/30/20 5:00 PM		
83		Deciding the Order Endpoints	1 day	11/24/20 8:00 AM	11/24/20 6:00 PM		Onur Kıpçolu,Murat Ekici,Eray Sezgin...
84		Front-End Implementation: Order Pages	4 days	11/25/20 8:00 AM	11/30/20 5:00 PM	83	Bar Alhan,Yaz Can Çolak,Asena Ka...
85		Back-End Implementation: Order Pages	4 days	11/25/20 8:00 AM	11/30/20 5:00 PM	83	Onur Kıpçolu,Murat Ekici,Ömer Ak,Er...
86		Android Implementation: Order Pages	4 days	11/25/20 8:00 AM	11/30/20 5:00 PM	83	Bar Mutlu,Ali Batr[150%],Baran Deni...
87		Implementation of Admin Panel and Product Display	5 days	11/30/20 8:00 AM	12/4/20 5:00 PM		
88		Deciding the Admin Endpoints	1 day	11/30/20 8:00 AM	11/30/20 5:00 PM		Onur Kıpçolu,Eray Sezgin,Ömer Ak,M...
89		Front-End Implementation: Admin Page	2 days	12/1/20 8:00 AM	12/2/20 5:00 PM	88	Bar Alhan,Yaz Can Çolak[50%],Me...
90		Back-End Implementation: Admin Page	2 days	12/1/20 8:00 AM	12/2/20 5:00 PM	88	Onur Kıpçolu,Murat Ekici,Ömer Ak,Er...
91		Android Implementation: Admin Page	2 days	12/1/20 8:00 AM	12/2/20 5:00 PM	88	Bar Mutlu,Buse Kabakolu,Baran De...
92		Deciding the Product Display Endpoints	1 day	11/30/20 8:00 AM	11/30/20 5:00 PM		Onur Kıpçolu,Eray Sezgin,Murat Ekici...
93		Deciding the Commenting and Rating Endpoints	1 day	11/30/20 8:00 AM	11/30/20 5:00 PM		Onur Kıpçolu,Eray Sezgin,Ömer Ak,M...
94		Front-End Implementation: Product Display Page	3 days	12/2/20 8:00 AM	12/4/20 5:00 PM	92,93	Yaz Can Çolak,Mehmet Çelimli,Bar...
95		Back-End Implementation: Product Display Page	3 days	12/2/20 8:00 AM	12/4/20 5:00 PM	92,93	Onur Kıpçolu,Murat Ekici,Ömer Ak,Er...
96		Android Implementation: Product Display Page	3 days	12/2/20 8:00 AM	12/4/20 5:00 PM	92,93	Ali Batr,Baran Deniz Korkmaz,Bar ...

Gantt Chart - page3

		Name	Duration	Start	Finish	Predecessors	Resource Names
97		<b>Implementation of Search Engine, filtering and sorting Functions</b>	<b>5 days</b>	<b>12/7/20 8:00 AM</b>	<b>12/11/20 5:00 PM</b>		
98		Deciding the Search Engine Endpoints, Implementing Filtering and Sorting Funct.	1 day	12/7/20 8:00 AM	12/7/20 6:00 PM		Onur Kılçolu;Eray Sezgin;Ömer Ak;M...
99		Front-End Implementation: Search Bar, Filtering and Sorting	4 days	12/8/20 8:00 AM	12/11/20 5:00 PM	98	Yaz Can Çolak; Mehmet Celimli; Asen...
100		Front-End Implementation: Search Bar, Filtering and Sorting	4 days	12/8/20 8:00 AM	12/11/20 5:00 PM	98	Onur Kılçolu; Murat Ekici; Ömer Ak; Er...
101		Front-End Implementation: Search Bar, Filtering and Sorting	4 days	12/8/20 8:00 AM	12/11/20 5:00 PM	98	Bar Mutlu; Buse Kabakolu; Ali Batr; B...
102		Internal Milestone 2	1 day?	12/11/20 8:00 AM	12/11/20 5:00 PM		
103		<b>Implementation of Shopping Carts and Favorite's Lists</b>	<b>5 days</b>	<b>12/14/20 8:00 AM</b>	<b>12/18/20 5:00 PM</b>		
104		Deciding Shopping Cart and Favorite's List Endpoints	1 day	12/14/20 8:00 AM	12/14/20 6:00 PM		Onur Kılçolu;Eray Sezgin;Ömer Ak;M...
105		Frontend Implementation of Shopping Carts and Favorite's Lists	4 days	12/15/20 8:00 AM	12/18/20 5:00 PM	104	Yaz Can Çolak; Mehmet Celimli; Asen...
106		Backend Implementation of Shopping Carts and Favorite's Lists	4 days	12/15/20 8:00 AM	12/18/20 5:00 PM	104	Onur Kılçolu; Murat Ekici; Ömer Ak; Er...
107		Android Implementation of Shopping Carts and Favorite's Lists	4 days	12/15/20 8:00 AM	12/18/20 5:00 PM	104	Bar Mutlu; Buse Kabakolu; Ali Batr; B...
108		<b>Implementation of Payment Page</b>	<b>5 days?</b>	<b>12/15/20 8:00 AM</b>	<b>12/21/20 5:00 PM</b>		
109		Deciding on Payment Page Endpoints	5 days?	12/15/20 8:00 AM	12/21/20 5:00 PM		Eray Sezgin; Murat Ekici; Onur Kılçolu...
110		Frontend Implementation of Payment Page	5 days?	12/15/20 8:00 AM	12/21/20 5:00 PM		Asena Karolin Özdemir; Bar Alhan; M...
111		Backend Implementation of Payment Page	5 days?	12/15/20 8:00 AM	12/21/20 5:00 PM		Eray Sezgin; Murat Ekici; Onur Kılçolu...
112		Android Implementation of Payment Page	5 days?	12/15/20 8:00 AM	12/21/20 5:00 PM		Ali Batr; Baran Deniz Korkmaz; Bar ...
113		<b>Implementation of Recommendation System</b>	<b>4 days</b>	<b>12/21/20 8:00 AM</b>	<b>12/24/20 5:00 PM</b>		
114		Implementing Recommendation System	4 days	12/21/20 8:00 AM	12/24/20 5:00 PM		Onur Kılçolu; Yaz Can Çolak; Buse K...
115		<b>Implementation of User Profiles</b>	<b>3 days?</b>	<b>12/24/20 8:00 AM</b>	<b>12/28/20 5:00 PM</b>		
116		Deciding on User Profile Endpoints	3 days?	12/24/20 8:00 AM	12/28/20 5:00 PM		Eray Sezgin; Murat Ekici; Onur Kılçolu...
117		Frontend Implementation of User Profiles	3 days?	12/24/20 8:00 AM	12/28/20 5:00 PM		Asena Karolin Özdemir; Bar Alhan; M...
118		Backend Implementation of User Profiles	3 days?	12/24/20 8:00 AM	12/28/20 5:00 PM		Eray Sezgin; Murat Ekici; Onur Kılçolu...
119		Android Implementation of User Profiles	3 days?	12/24/20 8:00 AM	12/28/20 5:00 PM		Ali Batr; Baran Deniz Korkmaz; Bar ...
120		Customer Milestone 2 of CMPE451	1 day?	12/29/20 8:00 AM	12/29/20 5:00 PM		
121		<b>Implementation of Messaging System</b>	<b>4 days</b>	<b>12/30/20 8:00 AM</b>	<b>1/4/21 5:00 PM</b>		
122		Implementing Messaging System	4 days	12/30/20 8:00 AM	1/4/21 5:00 PM		Bar Mutlu; Asena Karolin Özdemir; Ali...
123		<b>Implementation of Notification System</b>	<b>5 days</b>	<b>1/4/21 8:00 AM</b>	<b>1/8/21 5:00 PM</b>		
124		Implementing Notification System	5 days	1/4/21 8:00 AM	1/8/21 5:00 PM		Baran Deniz Korkmaz; Eray Sezgin; Me...
125		<b>System Testing</b>	<b>4 days</b>	<b>1/11/21 8:00 AM</b>	<b>1/14/21 5:00 PM</b>	<b>62;72;82;87;...</b>	
126		Testing: Integration Test	4 days	1/11/21 8:00 AM	1/14/21 5:00 PM		Onur Kılçolu; Baran Deniz Korkmaz; ...
127		Testing: Usability Testing	4 days	1/11/21 8:00 AM	1/14/21 5:00 PM		Onur Kılçolu; 150%; Yaz Can Çolak...
128		Performance Testing	4 days	1/11/21 8:00 AM	1/14/21 5:00 PM		Onur Kılçolu; Yaz Can Çolak; Buse K...

Gantt Chart - page4

		Name	Duration	Start	Finish	Predecessors	Resource Names
129		Deployment Preparation	6 days	1/15/21 8:00 AM	1/22/21 5:00 PM	125	Onur Kılıçolu[150%];Yaz Can Çolak...
130		Customer Milestone 3 for CMPE451	0 days	11/24/20 5:00 PM	11/24/20 5:00 PM		Onur Kılıçolu;Yaz Can Çolak;Buse K...

## 9. User scenarios

### 9.1. Android

- Eray decides to use Tursu for his new year's shopping and signs up as a customer.
- After signing up he goes to the electronics category to look for a mobile phone.
- He searches for "samsung" and looks at the results.
- He decides to select "Samsung S20 Plus" from the results and he goes to the product detail page.
- There he adds it to his shopping cart and proceeds to the cart.
- At the shopping cart he decides to complete the shopping and goes to the payment page.
- At the payment page he enters his credit card and address infos.
- After successfully completing his payment he decides to look for Apple's new earphones.
- He searches "earphones" and filters the results for the vendor "Apple".
- He finds the AirPods, he goes to the product's detail page.
- Deciding the price is too high for him he decides to just add it to his favourites.
- He clicks the "Add to favourites" button and he is redirected to the lists page.
- There he creates a new list and adds the product to it.
- Finally he proceeds to his profile page to log out before leaving the app.

### 9.2. Frontend

- After Eray completed his order, Samsung signs in to the website as a vendor.
- Samsung checks out the new dropdown menu and clicks *my profile* section.
- From the sidebar selects the *my orders* section to check if any new orders are requested.
- Samsung notices that a new order has requested and sets the status to *in delivery*
- As next, Samsung decides to add a new product and chooses to add *product* section from the sidebar.
- Fills the necessary fields and submits a new product named MNIST IMAGE.
- Samsung logs out.
- Admin logs in and is redirected to the admin panel page.
- Admin chooses to *verify product*, he sees a new non-verified product MNIST IMAGE and verifies it.
- After the verification admin signs out.

- A customer logs in to the system and looks for Samsung's new product.
- Customer filters product and finds MNIST IMAGE product and adds the product to a newly created product list.

## 10. Code Structure

### 10.1. When to open a new branch?

- When you are implementing a new feature
- When you are bugfixing
- When you are writing a test
- When you are improving the documentation part of the already existing code
- When you are refactoring an already existing code.

We do NOT push anything to *develop* branch without a pull request. The code in the develop branch should always be a working and peer-reviewed version of the code.

### 10.2. How to open a new branch?

We follow a name convention for branch names, to make it easier to keep track of changes.

1. We begin naming of branches with one keyword for representing the team.
  - *backend*
  - *frontend*
  - *android*
2. Afterwards, we add a keyword according to the type of the job done within that branch.
  - *-feature*
  - *-bugfix*
  - *-test*
  - *-documentation*
  - *-refactor*
3. And at the end, we add a descriptive words(at most three or four). Such as:  
*/signInPage, /paymentVerification, /forgotPassword*

At the end, we will have a name in the following format:

```
<team>-<type>/<description>
```

Examples:

- I am implementing a new feature, let's say the sign in operation in the frontend. I will create a branch as follows:  
`frontend-feature/signIn`
- We found a bug in the backend side of the payment system, which causes payments to fail with the code 403. Before solving the bug, we create a branch:  
`backend-bugfix/paymentFailed403`

### 10.3. When to commit?

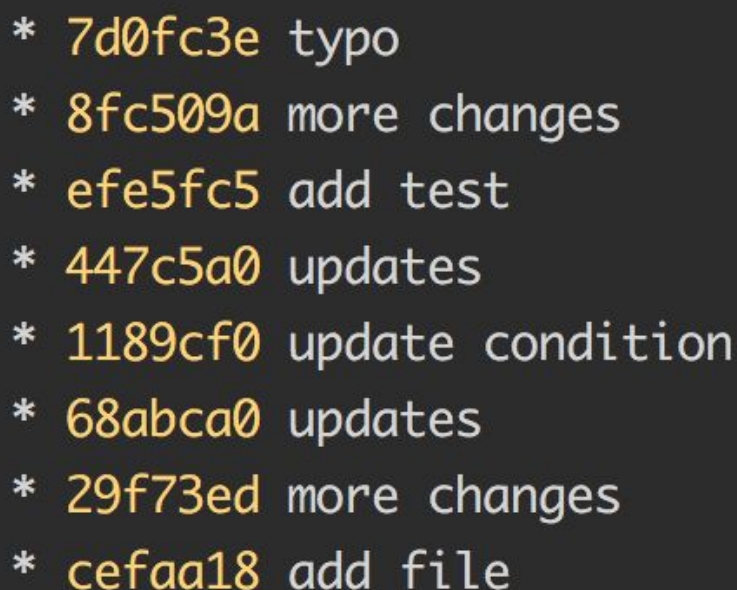
It is mostly a style question and personal. However, best practice is committing for each descriptive substep of the implementation. Because, when the pull request is created, Github creates an automatic changelog list according to the commit history.

### 10.4. How to commit?

Each commit should have a *clear and concise message*, describing the small step done.

We do NOT accept commit messages like these:

- Feature is implemented.
- commit
- minor changes
- oopsie
- wtf
- more changes



```
* 7d0fc3e typo
* 8fc509a more changes
* efe5fc5 add test
* 447c5a0 updates
* 1189cf0 update condition
* 68abca0 updates
* 29f73ed more changes
* cefaa18 add file
```



Some good commit message examples:

- updated test data generation
- fixed parsing of --issue-url option.

## 10.5. Pull Requests

For each branch created we open a pull request with the pull request format available in the wiki of our repository. Then we assign team members to review the code. After the code is reviewed by team members if it requires changes, the changes are implemented by the person who opened the pull request. If pull request is approved then the branch is merged into the respective team's branch.

# 11. Evaluation of Tools and Managing the Project

## 11.1. General Tools

- 11.1.1. Github  
We used GitHub as our project version management system.
- 11.1.2. Slack  
We used slack for communication.
- 11.1.3. Zoom  
During the distance-education, we have used Zoom for our weekly meetings.

## 11.2. Backend Tools

- 11.2.1. Python Django  
We decided to use Python as the main language of our project since all team members knew how to use it and it was easy and simple. Then, we decided to use Python on Django because of the similar reasons.
- 11.2.2. Postgresql  
We had to use a database system. Postgresql was a simple one and compatible for us. So, we decided to use it.
- 11.2.3. Docker  
We used Docker to deliver the project in a container.



## 11.3. Frontend Tools

- 11.3.1.      WebStorm  
As the IDE, we decided to use WebStorm.
- 11.3.2.      React  
We used react for developing our frontend application. React is a JavaScript library for building user interfaces.
- 11.3.3.      Material UI  
We used Material UI for our frontend application design. It is a React UI framework and widely used, so it was convenient to learn and use.
- 11.3.4.      Axios  
We used Axios for connecting our frontend application to the backend. Axios is a JavaScript library used to make HTTP requests.
- 11.3.5.      AWS  
We used AWS for deploying and hosting the application. AWS is widely used and a good choice.
- 11.3.6.      Docker  
We used Docker to deliver the project in a container.

## 11.4. Android Tools

- 11.4.1.      Android Studio  
We used Android Studio for development environment. It has the most crowded community and many sources to learn about it.
- 11.4.2.      Kotlin  
We used Kotlin to implement in Android Studio. It is an easy-learn and understandable language. Also has many sources on the Internet.
- 11.4.3.      Retrofit  
We used Retrofit networking library. We thought it is the best way to use Rest API and also has many sources.

## 11.5. Managing the Project

Managing the project was overall successful. Every team had meetings when it was needed. As communication tools, we used Zoom for meetings and Slack for general communication. Communicating in time and utilization of the communication tools were

effective enough, but we are planning to hold meetings on a more regular basis for the next milestone.

## 12. Assessment of the Customer Presentation

### Issues Pointed out by the Customers:

#	Issue	Teams related
1	Price range shall be visible during filtering.	Frontend
2	Applying multiple filters at once shall be possible.	Backend, Frontend, Android
3	A user should not have to type his mailing address again if an error occurs during the payment.	Frontend, Android
4	Rating should be enabled for different categories such as but not limited to: vendor, cargo, product.	Backend, Frontend, Android
5	Log out option should be available on the sidebar.	Android
6	Only strong passwords should be accepted by the system.	Backend
7	Filtering should be moved to the left side of the page.	Frontend

### Issues Pointed out by the Team Members:

#	Issue	Teams related
1	More realistic data should be used for scenarios.	Next milestone's presenters
2	Typo in the order page shall be fixed.	Frontend
3	Warning messages shall be in English	Android
4	The user shall not be able to enter anything else than positive numbers for stock and price fields.	Frontend, Android

During the presentation, we presented our scenarios and received many important feedbacks about the features we have implemented. Above you can find the feedbacks about the project from the customer and the team members themselves. After discussing these feedbacks in the team meeting we decided to spend more time to prepare and practice the scenario. Further, we decided to put the extra effort to prepare more realistic and structured data to be used during our demo. After the revision of these feedbacks we believe that the quality of our presentation for the last milestone will significantly increase.