# CMPE451 - Milestone 2 Report

**Group Members**

Burak Çuhadar

Mehmet Erdinç Oğuz

Koray Çetin

Eylül Yalçınkaya

Emre Girgin

Berkay Alkan

M. Olcayto Türker

Çağrı Çiftçi

Alperen Bağ

Veli Can Ünal

Meriç Üngör

Berke Can Gürer

*Fall, 2020*

# Table of Contents

# Executive Summary

## Introduction

Buyo is an e-commerce platform where customers can find a variety of products from different vendors. The platform provides a functional interface for customers to search for and buy products as well as vendors to sell their products. Intuitive user interactions and global standards are important for the platform to be successful.

## Work Done So Far

In the first milestone, we had kicked off the project and implemented basic functionalities for customer users. During the second milestone, we focused on completing the feature set for customer users and started the development of admin and vendor features.

Backend team focused on preparing our data from scratch and necessary endpoints. Frontend and android teams prepared UI's and implemented the backend connections for the whole customer user journey from start to finish: sign in/up, changing profile information, searching/filtering/sorting products, adding products to wishlist/cart and making payment for the cart.

During this milestone, we communicated very efficiently as a team. Request and response structures for the endpoints were decided with all teams included. Bug reports and suggestions were made and taken into consideration. Thanks to hardwork and good cooperation within the team, we were able to deliver our product to the second customer meeting.

## Road Ahead

We are almost done with customer user features. We will be focusing on vendor and admin functionalities and plan to deliver an MVP product with three users and their functionalities for the last milestone.

# Deliverable Status and Evaluation

## Backend

| | |
|---|---|
| Checkout Implementation | Done |
| Order Information | Done |
| Profile Information for Users | Done |
| Update Profile for Users | Done |
| Filtering Mechanism | Done |
| Shopping Cart Logic | Done |
| Vendor Logic | Partially Done |
| CI/CD for Backend | Partially Done |
| Change My Password | Partially Done |
| Credit Card Verification | Done |
| Comment and Rate | Done |

**Checkout Implementation - DONE:**
The checkout service is able to add the products in the cart for a customer and keep them as ordered products, therefore a customer can order products.

**Order Information - DONE:**
Users can look at their orders and see their status in their order page. Backend sends the products, their shipping status, their prices.

**Profile Information for Users - DONE:**
Backend provides the information that the user provided before to users back. Users can see their information in the related profile page in frontend or android.

**Update Profile for Users - DONE:**
Users can change their profile details such as their address, name, surname and phone number. We take the fields

**Filtering Mechanism - DONE:**

Users can provide the filters that they want to apply to the products in the products page, and get the filtered products after the backend applies filtering to productInfo fields to all products.

**Shopping Cart Logic - DONE:**
Users can add products to cart, and the sum of the product prices and the discounted prices can be provided to the users, in addition to the products added to the cart.

**Vendor Logic - PARTIALLY DONE:**
Vendor service currently has adding products feature, and uploading product pictures to the backend server. More features for vendors will be added, such as communicating with customers and changing the order status, in the future.

**CI/CD for Backend - PARTIALLY DONE:**
We tried to implement CI/CD for the backend, but failed for several reasons. We didn't understand how to use the Github's secrets feature, and after we realized, we still didn't find how to use the .pem key file for connecting to EC2 servers by using Github Actions. However, we made progress and until milestone-3, we will implement CI/CD.

**Change My Password - PARTIALLY DONE:**
Changing passwords is an available feature right now. However, there is no email verification service and it should be implemented until the next milestone.

**Credit Card Verification - DONE:**
Static credit card verification is implemented, and customers cannot make an order without providing valid credit card information.

**Comment and Rate - DONE:**
Customers can rate products, along with their comments if they want to. We keep those comment-rating pairs in a separate DB on the server-side.

## Mobile

| | |
|---|---|
| Customer Profile Page | Done |
| Vendor SignUp | Done |
| Address Page | Done |
| Cart Page | Done |
| Checkout Page | Done |

| | |
|---|---|
| Order Page | Done |
| Filter & Sort Page | Done |
| Add to Cart Page | Done |

**Customer Profile Page - DONE:**
The user is directed to this page after the login. On this page, the users can edit their credentials, add new addresses, and take an action about the orders. The customers can reach this page any time s/he wants by clicking on the profile tab. The profile page for vendor user is going to be added for the final milestone.

**Vendor SignUp Page - DONE:**
On this page, we ask the vendor to enter more credentials than we wanted from the customer user. For example, the vendor should enter his shop's location from the map using Google Maps  API. In addition to that, the domain of the e-mail of the vendor must match with the domain of her/his website.

**Address Page - DONE:**
The address page is a subpage of both checkout and profile pages. The customer user can edit and see her/his address using this page. This page is accessible from the checkout page as well. This property allows users to have a more flexible experience during their purchase process.

**Cart Page - DONE:**
On this page, the customer sees all the products s/he added to her/his cart. Note that, if a customer adds the same product more than one but with different features, (like different sizes or colors) those orders are listed separately. Therefore, the user can have a more clear insight into her/his orders. In addition, the user can remove a product from the cart and see the final amount that will be paid. (including discounts and shipping fees.) The user is directed to the checkout page from this page.

**Checkout Page - DONE:**
This page is the place where the user enters the payment options like credit card id. Besides, the customer can add a new address or edit the existing ones by being directed to the address page. After the payment process is completed. The products in the cart drop to the orders page.

**Order Page - DONE:**
On the order page, the customer can check all the orders s/he gave. Orders are divided into three categories: Pending, Shipped, and Delivered. Once an order is delivered, the customer can make a comment about the product and rate it. Until that, the customer can cancel her/his orders. The canceling, commenting, and rating process will be added to the final milestone.

**Filter & Sort Page - DONE:**
On this page, either customer or guest user can sort or filter the products in the product list page. The filter options are fetched according to the query that fetches the products. To be concrete, search for any keyword, and any category will have its own filters on that page. However, the sorting criteria are fixed.

Note that the user does not have to enter all the options when s/he is filtering or sorting. S/he may leave empty the features s/he does not want to filter for.

**Add to Cart Page - DONE:**
On the add to the cart page, the available options for that product are listed. By default, a configuration is selected. After the customer starts to change the default ones, the current stock value of that configuration is dynamically updated and shown to the user, without needing any additional refresh action. Note that, to be able to add a product to the cart, the user must log in.

## Frontend

| | |
|---|---|
| Product List Page | Done |
| Cart Page | Done |
| Checkout Page | Done |
| Vendor Sign In/Up Page | Done |
| Customer Profile Page | Done |
| Customer Orders Page | Partially Done |
| Dockerization, CI/CD | Done |
| Customer Addresses Page | Done |

**Product List Page - DONE:**
This page is one of the most important pages as users spend most of their time here. Therefore we decided to implement this for this milestone. On this page products are listed according to the selected categories or the given search query. Users can order the products by rate, price and name in ascending or descending order. Filtering functionality is also implemented but users cannot choose more than one filter for an attribute as it is not yet supported by backend.
**Cart Page - DONE:**
In this milestone we wanted to implement all the necessary functionalities to enable the customers experience each step of shopping flow. Therefore implementing the cart page was a must. Users can add a product to their cart at the product page. The shopping cart of a user is stored in the backend so that the products added to the cart are saved and can be displayed even after the user logs out and logs in again. Users can also delete products from their cart. There is a checkout button to buy the products in the cart. Clicking that button redirects to the checkout page.
**Checkout Page - DONE:**
This is where the user enters the address and payment information to buy the products in the cart. The user can either choose an address already saved in her profile or add a new address here. Addresses are stored in the backend. For the payment information users enter the name on the card, expiration month and year, card number and finally the CVV number. These fields are validated to ensure the user has

entered the information in the correct format. If there is a problem the corresponding field is highlighted with red color. There is also a terms and conditions checkbox. The user has to read and accept those forms but unfortunately we could not add the forms to the page for this milestone. Those documents are ready and will surely be delivered for the next milestone.

**Vendor Sign In/Up Page - DONE:**

For this milestone we also started to think about other user types. We implemented the vendor sign in/up page and connected it to the backend. Sign in page is very similar to the customer sign in page. However at the vendor sign up page additional information such as company name, business website and the location of the distribution center are required.

**Customer Profile Page - DONE:**

On this page name, surname, email, phone number and gender of the user are displayed and can be edited. There is also a section where the user can change the password. All the information is obtained from the backend and edited information is updated at the backend accordingly.

**Customer Orders Page - PARTIALLY DONE:**

After the user makes the payment, a corresponding order is created. Those orders can be displayed at the orders page. Ordered products are listed for each order and there are additional options for each product like cancelling the order, messaging the vendor of the product and giving feedback after the product is delivered. Cancelling the order and messaging the vendor buttons are nonfunctional for this milestone as the backend api for them is not implemented yet. However the users can give feedback after the status of the product is delivered. After clicking the button a dialog opens where the user can rate the product and add comments.

**Dockerization, CI/CD - DONE:**

We had a problem with dockerization for the first milestone and therefore it was postponed for this milestone. Finally we successfully dockerized the app and deployed it to an AWS machine. We also automated our workflow using Github Actions. Our app is dockerized and deployed after each push and pull request to the master branch.

**Customer Addresses Page - DONE:**

On this page customers can add and edit their addresses. The addresses are saved to the backend so that the same addresses appear at the checkout page.

## Summary of Coding Work

| | |
|---|---|
| Eylül Yalçınkaya (Frontend) | I implemented cart page, vendor sign in/up, profile and change password page UI. I also |

| | |
|---|---|
| | implemented these pages' backend connections: remove from cart, get cart info, vendor sign in/up, change profile info, change password. I contributed to bug fixes and testing as well as code review. |
| Meriç Üngör (Frontend) | I implemented continuous integration and delivery for our frontend system using Github Actions. Also, I implemented the checkout flow with credit card validation, a customer addresses page where the customer can see/add/edit their addresses, a google maps functionality on vendor sign up where vendors can choose their distribution location. Finally I have dealt with the initial project setup and the login page of our Admin project and setup Eslint to achieve a cleaner, scalable and more consistent codebase. |
| Mehmet Erdinc Oguz (Backend) | I implemented the shopping cart logic in the backend where users can add products to the cart, get the products in their cart, and empty their cart. I also implemented the checkout service from scratch where users can order the products in their cart, get the current status for ordered products and change the status of the orders. I changed the product structure so that it includes generic attributes for each product to help filtering mechanism and adding product mechanism for vendors. I helped CI/CD implementation for both frontend and backend by using Github Actions. Also, I created mock data for the products in backend DB(furniture mock data).<br>Finally, I wrote unit tests for the cart logic |
| Alperen Bağ (Android) | I implemented the vendor sign-up/login features and their UIs. In addition to this, I changed the UI of customer login/sign-up pages. I implemented the validation process of the credentials (vendor email, password validity etc.) I did some visual changes for the homepage. Finally, I implemented the order page feature with the UI. We also did many bug solving sessions on zoom. |

| | |
|---|---|
| Koray Cetin (Backend) | In this milestone, I’ve configured the microservice architecture using docker compose and nginx. This way we divided our backend code to six sub-services. Other than this, I completed several endpoint tasks in various sides of our code. I’ve implemented the update account info, vendor add product, upload file, and comment endpoints. Upload file endpoint was tough because it requires us to use the AWS bucket library. I’ve also created mock products for the clothing category and wrote the first tests by configuring the test structure. |
| Emre Girgin (Android) | I had fixed some of the bugs that inherited from the first milestone. I made some visual updates on the product detail and product list page. Besides, I have implemented the filter and sort page and added to cart pages and their backend connections. I've also implemented android comments page and its backend connection but we didn't include it to the 2. milestone, since we did not test it enough. |
| Olcayto Türker (Backend) | I implemented get profile info and change password features for account service. I made some changes in vendor and customer schemas for these implementations. I also implemented address mechanism such as adding an address, updating an address and deleting an address. I created cosmetics products in our products database as well. Lastly, I implemented unit tests for the views in interaction service, like and comment. |
| Burak Çuhadar (Frontend) | I implemented the UI of orders page, category products(product list) page, and added UI for choosing attributes of a product at the product page. I connected the add-to-cart button at the |

| | |
|---|---|
| | product page to the backend so that the product is saved to the cart. Data for the orders page is also obtained from the backend. I also implemented giving feedback functionality at the orders page which also is connected to the backend. At the product list page all the data is obtained from the backend and filtering/sorting functionalities are implemented using the api provided by the backend. I also connected the comments section of the product page to the backend, so those comments are not mock-up anymore. Finally, I also contributed to bug fixes and reviewed code of my team members. |
| Berkay Alkan (Android) | I have designed the UI and implemented the back-end connections of the customer profile page, account information page, change password page and address information page with their functionalities. |
| Veli Can Ünal (Backend) | I designed a sorting mechanism according to rating, name and price in ascending or descending order. I coded the filter mechanism for every attribute of different categories. I edit the product database several times and add electronics categories specifically. I wrote the preliminary information form. I tried to create the CI/CD structure for a backend server. |
| Çağrı Çiftçi(Android) | I fixed some minor bugs about dissimilarity between android default and our application back button and changing tab. I implemented the cart, checkout, add/update address pages features and their UIs. In addition to these, I implemented the necessary server connection(end-points). I attend all android group meetings. After opening pull requests, we reviewed our code and we merged them together in those meetings. Also we solve many bugs in the meetings. Before the customer meeting, we had enough rehearsal. I also had a few meeting with my friends from backend team(Erdinç-Veli) to discuss about some data models and future plans |

| | |
|---|---|
| Berke Can Gürer (Frontend) | I got more involved in the project and figured out more of what we're doing on the frontend side as a team. I attend the meetings and provide support. I also didn't forget to write this section, this time around. |

# Unit Tests

### Koray Cetin

I've initialized the test logistics with this pull request BKND-#344-Test.

Commit: 66c0539577553561756490d4347d9135f7c97031

I've written tests for login and signup endpoints with this commit.

- Login tests cover missing parameter error, user not found error and vendor, customer successful logins.
- Signup tests cover missing parameter error, email already exists error and vendor, customer successful signups.

Commit: 43499f20c92eecf343f35f93fdf244c49797059a

I've written tests for get account info, update account info, change password, add address, update address, delete address,

- Update account info tests cover missing parameters and user not found errors. Success functionality is checked with getAccountInfo function.
- Change password tests cover missing parameter error, user not found error and successful password change. To check that the password has changed properly, I've used the login function.
- Get account info tests cover missing parameters error, user not found error and successful result cases.
- Add address tests cover missing parameters, user not found and address already exists errors. Successful result is tested with the getAccountInfo function.
- Update address and delete address tests cover missing parameters, user not found and success results.

### Veli Can Ünal

I wrote the tests for checking product list and categories
Issue: https://github.com/bounswe/bounswe2020group4/pull/392

- getProducts test checks the returning type of the array and the size of the array
- Since we are returning array again , getProductCategories test checks the returning type of the array and the size of the array again

Pull request: https://github.com/bounswe/bounswe2020group4/pull/392

### Mehmet Erdinc Oguz

I wrote the tests for some of the cart logic functions with this PR: 393

Commit: b8b5e535f1f95e3dfa43b52b816763268ebbdec8

I've added unit tests for the following:

- updateCart function tests cover missing arguments case for the function.
- getCartProducts function tests cover missing arguments and successful result cases.
- emptyCart functions tests cover missing arguments and successful result cases

### M.Olcayto Türker

I implemented unit tests for the functions in interaction service such as comment and like. PR: 394
commit: 877147d00f75284f5925de8939d46fe466a8b40b

I've added unit tests for the following:

- Tests controlling if request is sent correctly for comment and like endpoints.
- Expected returning values are written for both correct and incorrect requests.
- For incorrect endpoint cases like missing parameter or product not found, expected returning value is defined in corresponding unit tests.

# Challenges Met During Deployment, Dockerizing and CI/CD

## Frontend

We have already dealt with the dockerization and the deployment of our frontend project before the first customer meeting and provided a detailed explanation in our first milestone report. Until the second milestone, we managed to set up CI/CD for our project as well. Our CI/CD process runs as follows. When a commit is pushed or a pull request is merged to the master branch, Github actions workflow is automatically run. In this workflow, we build our docker image, decrypt our encrypted key pair which is used to connect to our machine via ssh(located in the repository), connect to our machine using this key pair, clear the machine from any past docker images and containers, transfer the docker image built earlier to the machine, and run the docker image.

Our first decision was to use GitHub Actions to achieve this task since it was easy to get adapted to, had tremendous community support, and was just one of the most popular choices for this task. After some research we decided to try deploying our docker images to the elastic container repository service of AWS where our servers are running, since it was the first path we encountered online. But after playing with it for some time, we couldn't seem to make it work and since we weren't really competent on the steps that were taken and the services used, we couldn't really debug what was wrong and got frustrated. That was the first challenge we faced. So we decided to go in a different direction. We were wondering if we can't just connect to our machine via ssh and just run some scripts. So we found a public action which is perfect for this. After that we realized pulling the repository and building the docker image every time was not the best option in terms of maintainability and saw that the same author who wrote the ssh action also had an scp action. So we decided to use this to build the docker image inside the GitHub actions and then transfer the image to our machine.

The final challenge we faced was no matter what variation we tried, we couldn't seem to successfully make the ssh connection. And we couldn't really find any support about it online. But after some research, we came across another project that mostly followed the same path as us but with one big difference. We included our key pair which is used for the ssh connection in the repository's secrets, and then accessed it using 'key: ${{ secrets.KeyPair }}'. But this project just uploaded their key pair to their repository, encrypted it for security, decrypted it in actions and provided a file path to it using 'key_path: PATH'. And then we realized that since these custom actions we were using were third party software, it is quite possible that the 'key' parameter may not even work properly even though we may have been doing everything right. So we decided to try this similar approach. We encrypted our key pair file using gpg, provided the decryption passphrase in our repository's secrets and used it in our actions. After that everything worked like a charm.

## Backend

In the previous milestone we successfully deployed a docker container to AWS EC2 server and ran it. But what we have in our minds was building a microservice structure because it has numerous advantages such as fault isolation, independent deployment and independent scaling. We divided our backend to six microservices: account service, checkout service, product service, interaction service, vendor service and db service. Account service handles the user functions such as signup and login. Product service consists of product and cart functionalities, the payment flow process after the cart phase is handled by checkout service. Interaction service includes the user interactions such as comment, rate and like whereas vendor service handles the vendor functionalities such as add or update product. And finally db service is only for initializing a mock database.

Six microservices are placed into different docker containers and they all depend on a mongodb container. All these containers are orchestrated by docker-compose. In docker-compose we can run multiple dockerfiles and these containers are served in different ports. This way when a container crashes, other containers are able to continue to work properly. Since all these containers are served in different ports, we had to proxy all the requests to related containers with another technology.

Nginx is a proxy tool that allows us to redirect the requests received on port 8080 to microservice containers' ports: 5000, 5001, 5002… This functionality is provided with a nginx.conf file that maps the request urls that start with a text to provided ports.]

Building this microservice structure was hard because it's a whole new area for us and it's not easy to find resources that match with our criteria. Also there are not many proper instructions to create a docker-compose.yml and nginx.conf file. And when an attempt fails, we don't get much of an idea why it didn't work. Because of this we couldn't make a cumulative progress, instead we always started from the beginning in each try. Also making a mongo container that every container depends on, necessitated a good research.

We created a AWS EC2 workflow called "Backend Deploy" in github actions. Github already provides a template for continuous deployment. We modified this template to fulfill our purposes. A hook is added to configure the deployment after a push occurs in branch "backend-staging" and this hook triggers the job "deploy-backend". This job has the following steps. The machine clears the existing docker containers, pushes docker image to aws, deploys an image to the machine and runs docker container, loads the image and runs it. However because of a problem with our server, we couldn't deliver a successful CI/CD feature to our application. Since we don't have enough experience with CI/CD, we couldn't deliver this part quickly and we couldn't focus completely on this part because we tried to deliver our endpoints.

# API Documentation

Base URL: http://localhost:8080/

Base URL on Amazon server: http://3.138.113.101:8080/

## About the App

This is the backend of an e-commerce product which serves web and android applications.

## How to Run?

Once you are inside the backend/product_service folder, install docker and run

```
docker-compose build
docker-compose up
```

Web app is available on the base URL, you can query the app with any request tool.

## Endpoints

All of the endpoints return responses in JSON format.

### Delete Comment

**Deletes a comment from a product.**

**Endpoint: /comment**

Method: **DELETE**

Authorization: None

Parameters: id=5fcf71186e8db70012a8e2f3

Response: **{ "status": { "code": 200, "message": "OK" } }**

### Add Address to Customer User

Adds an address to a customer. Note: addressTitle must be unique.

**Endpoint: /account/address**

Method: **POST**

Authorization: None

Parameters: id=123&address={"addressTitle": "home", "city": "istanbul", "province": "sarıyer", "street": "elm", "address": "dksdssd"}

Response: { "status": { "code": 200, "message": "OK" }}

## Update Address of Customer User

Updates adress of a customer with given addressTitle.

**Endpoint: /account/address**

Method: **PATCH**

Authorization: None

Parameters: id=123&address={"addressTitle": "home", "city": "denizli", "province": "merkez", "street": "elm", "address": "dksdssd"}

Response: { "status": { "code": 200, "message": "OK" }}

## Delete Address of Customer User

Deletes address of a customer with given addressTitle.

Endpoint: **/account/address**

Method: **DELETE**

Authorization: None

Parameters: id=123&address={"addressTitle": "home"}

Response: { "status": { "code": 200, "message": "OK" }}

## Add Comment

Adds a comment to a product.

Endpoint: **/comment**

Method: **POST**

Authorization: None

Parameters: userId=123&productId=123&comment=asdadsdsadas&rating=4.23

Response: { "status": { "code": 200, "message": "OK" }, "data": {"commentId": 324}}

## Sort products

Sorts products according to rating, name and price in ascending or descending order. sortingFactor parameter can be rating, name and price. sortingType can be ascending or descending.

Endpoint: **/products**

Method: **POST**

Authorization: None

Parameters:
categories=["Kids"]&sortingFactor=rating&subcategory=T-shirt&size=XS&brand=Adidas&color=Blue&sortingType=descending

Response: { status: { code: 200, message: "OK", }, data: { products: []}}

## Get Categories

Returns an inspirational quote from a famous, historical person.

Endpoint: **/categories**

Method: **GET**

Authorization: None

Parameters: None

Response: { status: { success: true, code: 200 }, data: { categories: [{ name: "Woman", path: "Woman", subcategories: [{ name: "Shoes", path: "Woman,Shoes", subcategories: []}] } ]}}

**Get Products of a Category**

Gets the products of a category

Endpoint**: /products**

Method: **GET**

Authorization: None

Parameters(Query):?categories=["Phones"] Warning: "filterCriterias" shows that which attributes can we use the current category.

Response:

```
{
   "status": {
      "code": 200,
      "message": "OK"
   },
   "data": {
      "products": {
         "productList": [
            {
               "category": [
                  "Electronics",
                  "Phones"
               ],
               "productInfos": [
                  {
                     "attributes": [
                        {
                           "name": "screenSize",
                           "value": "5.5 Inch"
                        },
                        {
                           "name": "RAM",
                           "value": "4 GB"
                        },
                        {
```

```
                    "name": "diskSize",
                    "value": "64 GB"
                  }
                ],
                "stockValue": 21
              },
              {
                "attributes": [
                  {
                    "name": "screenSize",
                    "value": "5.5 Inch"
                  },
                  {
                    "name": "RAM",
                    "value": "4 GB"
                  },
                  {
                    "name": "diskSize",
                    "value": "128 GB"
                  }
                ],
                "stockValue": 18
              }
            ],
            "description": "Uncover the Ultimate Design. Beauty Beyond the Visual",
            "name": "Samsung S8",
            "price": 2800,
            "originalPrice": 3000,
            "imageUrl":
"https://elasticbeanstalk-us-east-2-334058266782.s3.amazonaws.com/images/16089907472046",
            "rating": 4,
            "brand": "Samsung",
            "vendor": {
              "name": "JohnsShop",
              "rating": 4.23
            },
            "id": "5fe86281a17abd0012d601e6"
          },
          {
            "category": [
    "Electronics",
    "Phones"
],
"productInfos": [
    {
        "attributes": [
            {
                "name": "screenSize",
                "value": "5.5 Inch"
            },
            {
                "name": "RAM",
                "value": "8 GB"
            },
            {
                "name": "diskSize",
                "value": "64 GB"
            }
        ],
        "stockValue": 54
    },
    {
        "attributes": [
            {
                "name": "screenSize",
                "value": "5.5 Inch"
            },
            {
                "name": "RAM",
                "value": "8 GB"
            },
            {
                "name": "diskSize",
                "value": "128 GB"
            }
        ],
        "stockValue": 74
    }
],
"description": "Uncover the Ultimate Design. Beauty Beyond the Visual",
            "name": "Samsung S9",
            "price": 4000,
            "originalPrice": 4200,
            "imageUrl":
"https://elasticbeanstalk-us-east-2-334058266782.s3.amazonaws.com/images/16089907472057",
            "rating": 4.4,
            "brand": "Samsung",
            "vendor": {
              "name": "Ahmet",
              "rating": 3.22
            },
            "id": "5fe86281a17abd0012d601e7"
          },
          {
            "category": [
              "Electronics",
              "Phones"
            ],
            "productInfos": [
              {
                "attributes": [
                  {
                    "name": "screenSize",
                    "value": "6 Inch"
                  },
                  {
                    "name": "RAM",
                    "value": "8 GB"
                  },
                  {
                    "name": "diskSize",
                    "value": "64 GB"
                  }
                ],
                "stockValue": 95
              },
              {
                "attributes": [
                  {
                    "name": "screenSize",
                    "value": "6 Inch"
                },
                {
                    "name": "RAM",
                    "value": "8 GB"
                },
                {
                    "name": "diskSize",
                    "value": "128 GB"
                }
            ],
            "stockValue": 95
        }
    ],
    "description": "Uncover the Ultimate Design. Beauty Beyond the Visual",
    "name": "Samsung S10",
    "price": 4500,
    "originalPrice": 5000,
    "imageUrl":
"https://elasticbeanstalk-us-east-2-334058266782.s3.amazonaws.com/images/16089907472078",
    "rating": 4.4,
    "brand": "Samsung",
    "vendor": {
        "name": "AyseTeyze",
        "rating": 3.21
    },
    "id": "5fe86281a17abd0012d601e8"
},
{
    "category": [
        "Electronics",
        "Phones"
    ],
    "productInfos": [
        {
            "attributes": [
                {
                    "name": "screenSize",
                    "value": "5 Inch"
                },
                {
                    "name": "RAM",
                    "value": "4 GB"
                },
                {
                    "name": "diskSize",
                    "value": "64 GB"
                }
            ],
            "stockValue": 47
        },
        {
            "attributes": [
                {
                    "name": "screenSize",
                    "value": "5 Inch"
                },
                {
                    "name": "RAM",
                    "value": "4 GB"
                },
                {
                    "name": "diskSize",
                    "value": "128 GB"
                }
            ],
            "stockValue": 94
        }
    ],
    "description": "Uncover the Ultimate Design. Beauty Beyond the Visual",
    "name": "Samsung S7",
    "price": 4500,
    "originalPrice": 5000,
    "imageUrl":
"https://elasticbeanstalk-us-east-2-334058266782.s3.amazonaws.com/images/16089907472099",
    "rating": 3.5,
    "brand": "Samsung",
    "vendor": {
        "name": "JohnsShop",
        "rating": 4.23
    },
    "id": "5fe86281a17abd0012d601e9"
},
{
    "category": [
        "Electronics",
        "Phones"
    ],
    "productInfos": [
        {
            "attributes": [
                {
                    "name": "screenSize",
                    "value": "5 Inch"
                },
                {
                    "name": "RAM",
                    "value": "8 GB"
                },
                {
                    "name": "diskSize",
                    "value": "64 GB"
                }
            ],
            "stockValue": 39
        },
        {
            "attributes": [
                {
                    "name": "screenSize",
                    "value": "5 Inch"
                },
                {
                    "name": "RAM",
                    "value": "8 GB"
                },
                {
                    "name": "diskSize",
                    "value": "128 GB"
                }
                ],
                "stockValue": 23
            }
        ],
        "description": "Uncover the Ultimate Design. Beauty Beyond the Visual",
        "name": "Iphone 8",
        "price": 4700,
        "originalPrice": 5000,
        "imageUrl":
"https://elasticbeanstalk-us-east-2-334058266782.s3.amazonaws.com/images/160899074721010
",
        "rating": 4.8,
        "brand": "Apple",
        "vendor": {
            "name": "Pablos",
            "rating": 2.43
        },
        "id": "5fe86281a17abd0012d601ea"
    },
    {
        "category": [
            "Electronics",
            "Phones"
        ],
        "productInfos": [
            {
                "attributes": [
                    {
                        "name": "screenSize",
                        "value": "5 Inch"
                    },
                    {
                        "name": "RAM",
                        "value": "4 GB"
                    },
                    {
                        "name": "diskSize",
                        "value": "64 GB"
                    }
                ],
                    "stockValue": 42
                },
                {
                    "attributes": [
                        {
                            "name": "screenSize",
                            "value": "5 Inch"
                        },
                        {
                            "name": "RAM",
                            "value": "4 GB"
                        },
                        {
                            "name": "diskSize",
                            "value": "128 GB"
                        }
                    ],
                    "stockValue": 45
                }
            ],
            "description": "Uncover the Ultimate Design. Beauty Beyond the Visual",
            "name": "Iphone 6",
            "price": 5200,
            "originalPrice": 5500,
            "imageUrl":
"https://elasticbeanstalk-us-east-2-334058266782.s3.amazonaws.com/images/160899074721211
",
            "rating": 4,
            "brand": "Apple",
            "vendor": {
                "name": "Pablos",
                "rating": 2.43
            },
            "id": "5fe86281a17abd0012d601eb"
        }
    ],
    "filterCriterias": [
        {
            "name": "screenSize",
            "displayName": "Screen Size",
                    "possibleValues": [
                        "5.5 Inch",
                        "6 Inch",
                        "5 Inch"
                    ]
                },
                {
                    "name": "RAM",
                    "displayName": "RAM",
                    "possibleValues": [
                        "4 GB",
                        "8 GB"
                    ]
                },
                {
                    "name": "diskSize",
                    "displayName": "Disk Size",
                    "possibleValues": [
                        "64 GB",
                        "128 GB"
                    ]
                }
            ]
        }
    }
}
```

## Get Products with Keyword

Returns the products whose title is the keyword.

Endpoint**: /products**

Method: **GET**

Authorization: None

Parameters(Query): search=cocuk

Response:
{ status: { code: 200, message: "OK", }, data: { products: [ { category: ["Cocuk", "Erkek Cocuk", "Ic Giyim"], sizes: null, colors: ["Red", "Blue", "White", "Purple", "Orange", "Black"], name: "Erkek Cocuk Pamuklu Atlet 2'li", id: 10001, imageUrl: "https://img-lcwaikiki.mncdn.com/mnresize/230/-/pim/productimages/20201/4041406/l_20201-0w1007z4-jyx_a.jpg", rating: 0.78, price: 22.99, originalPrice: 22.99, brand: "Adidas", stockValue: { Red: 17, Blue: 69, White: 72, Purple: 57, Orange: 71, Black: 44, }, vendor: { name: "Ahmet", rating: 3.22, }, }, { category: ["Cocuk", "Kiz Cocuk", "Kazak"], sizes: null, colors: ["Red", "Blue", "White", "Purple", "Orange", "Black", "Grey", "Green"], name: "Kiz Cocuk Kalin Triko Kazak", id: 10002, imageUrl: "https://img-lcwaikiki.mncdn.com/mnresize/230/-/productimages/20192/4/3870373/l_20192-9wp531z4-gzn_a.jpg", rating: 4.92, price: 19.99, originalPrice: 39.99, brand: "Nike", stockValue: { Red: 82, Blue: 37, White: 35, Purple: 15, Orange: 53, Black: 26, Grey: 49, Green: 17, }, vendor: { name: "Ahmet", rating: 3.22, }, }, ], }, };

## Get Product Detail

Returns the detailed information of a product.

Endpoint: **/product**

Method: **GET**

Authorization: None

Parameters(Query): id=5fe86281a17abd0012d601eb

Response:

```
{
  "status": {
    "code": 200,
    "message": "OK"
  },
  "data": {
    "result": {
      "category": [
        "Electronics",
        "Phones"
      ],
      "productInfos": [
        {
          "attributes": [
            {
              "name": "screenSize",
              "value": "5 Inch"
            },
            {
              "name": "RAM",
              "value": "4 GB"
            },
            {
              "name": "diskSize",
              "value": "64 GB"
            }
          ],
          "stockValue": 42
        },
        {
          "attributes": [
            {
              "name": "screenSize",
              "value": "5 Inch"
            },
            {
              "name": "RAM",
              "value": "4 GB"
            },
            {
                    "name": "diskSize",
                    "value": "128 GB"
                  }
                ],
                "stockValue": 45
              }
            ],
            "description": "Uncover the Ultimate Design. Beauty Beyond the Visual",
            "name": "Iphone 6",
            "price": 5200,
            "originalPrice": 5500,
            "imageUrl":
"https://elasticbeanstalk-us-east-2-334058266782.s3.amazonaws.com/images/160899074721211
",
            "rating": 4,
            "brand": "Apple",
            "vendor": {
              "name": "Pablos",
              "rating": 2.43
            },
            "id": "5fe86281a17abd0012d601eb",
            "comments": [],
            "filterCriterias": [
              {
                "name": "screenSize",
                "displayName": "Screen Size",
                "possibleValues": [
                  "5 Inch"
                ]
              },
              {
                "name": "RAM",
                "displayName": "RAM",
                "possibleValues": [
                  "4 GB"
                ]
              },
              {
                "name": "diskSize",
                "displayName": "Disk Size",
                    "possibleValues": [
                        "64 GB",
                        "128 GB"
                    ]
                }
            ]
        }
    }
}
```

## Like / Dislike

Triggers the like or dislike event from a user to a product.

Endpoint**: /like**

Method: **POST**

Authorization: None

Parameters(Query): customerId=12341&productId=10003

Response: { "status": { "code": 200, "message": "OK" } }

## Get Wishlist

Gets the wishlist of a user.

Endpoint**: /wishlist**

Method: **GET**

Authorization: None

Parameters: customerId=12341

Response: { "status": { "code": 200, "message": "OK" }, "data": { "products": [ { "category": [ "Cilt", "Yüz Bakımı", "Yüz Maskesi" ], "sizes": null, "colors": null, "name": "Saf Kil Detoks Maskesi 50 ml", "id": 12312, "imageUrl": "https://img-watsons.mncdn.com/Content/Images/Thumbs/0320616_saf-kil-detoks-maskesi-50-ml.png", "rating": 1.2, "price": 69.95, "originalPrice": 69.95, "brand": "Nike", "stockValue": { "self": 45 }, "vendor": { "name": "AyseTeyze", "rating": 3.21 } } ] } }

## Get Customer User Information

Gets a customer user's information.

Endpoint: /**account**

Method: **GET**

Authorization: None

Parameters(Query): id=1024&userType=customer

Response: {"status": {"code": 200, "message": "OK"}, "data": {"result": {"address": [],"email": "customer@mail.com", "id": 1024, "password" = "1234", "gender" = "male"}}}

## Get Vendor User Information

Gets a vendor user's information.

Endpoint: /**account**

Method: **GET**

Authorization: None

Parameters(Query): id=1024&userType=vendor

Response: {"status": {"code": 200, "message": "OK"}, "data": {"result": {"longitude": "long", "longitude": "lang", "email": "customer@mail.com", "id": 1024, "password" = "1234", "company" = "vendor's company", "website" = "www.vendor.com",}}}

## Login

Performs the login event of a user.

Endpoint: /**login**

Method: **POST**

Authorization: None

Parameters(Query): userType=customer&email=mark@zucker.org&password=1234

Response: { "status": { "code": 200, "message": "OK" }, "data": { "userId": 10 } }

## Vendor Sign-up

Performs the sign-up event for a vendor user.

Endpoint: /**signup**

Method: **POST**

Authorization: API key from .env file

**Parameters(Query):**userType=vendor&email=mark@zucker.org&password=1234&longitude=long&latitude=lang&website=website.com&company=company

Response: { "status": { "code": 200, "message": "OK" }, "data": { "userId": 10 } }

## Customer Sign-up

Performs the sign-up event for a customer user.

Endpoint: /**signup**

Method: **POST**

Authorization: API key from .env file

Parameters(Query): userType=customer&email=mark@zucker.org&password=1234

Response: { "status": { "code": 200, "message": "OK" }, "data": { "userId": 10 } }

## Reset the Mock Database

This endpoint is created for internal purposes, it can be used when the database is needed to initialized or reset.

Endpoint: **/db/init**

Method: **POST**

Authorization: API key from .env file

Parameters: None

Response: { "status": { "code": 200, "message": "OK" } }

## Add Product to Cart

Add a product to customer's cart. Product Info can be created as below:

PRODUCT_INFO = {"attributes": [{"name": "size","value": "L"},{"name": "color","value": "Blue"}], "quantity": 1}

Endpoint: /**cart**

Method: **POST**

Authorization: None

Parameters(Query): customerId=1234567890abc&productId=0987654321abc&productInfo=<PRODUCT_INFO>

Response: { "status": { "code": 200, "message": "OK" } }

## Remove Product from Cart

Remove a product from customer's cart. Same endpoint as above is used, except the "quantity" field in ProductInfo should be missing when deletion is made. created as below:

PRODUCT_INFO = {"attributes": [{"name": "size","value": "L"},{"name": "color","value": "Blue"}]}

Endpoint: /**cart**

Method: **POST**

Authorization: None

Parameters(Query): customerId=1234567890abc&productId=0987654321abc&productInfo=<PRODUCT_INFO>

Response: { "status": { "code": 200, "message": "OK" } }

## Get Products in the Customer Cart

Gets all products in a customer's cart. Product Info should be created as below:

Endpoint: /**cart**

Method: **GET**

Authorization: None

Parameters(Query): customerId=1234567890abc

Response: { "status": { "code": 200, "message": "OK"}, data: { products } }

## Empty the Cart

Remove all products in a customer's cart.

Endpoint: /**cart**

Method: **DELETE**

Authorization: None

Parameters(Query): customerId=1234567890abc

Response: { "status": { "code": 200, "message": "OK"} }

## Checkout Order

Create an order from the products in the cart.

Endpoint: /**order**

Method: **POST**

Authorization: None

Parameters(Query): customerId=1234567890abc&creditCard={"name": "712837123","number": 1234666666667777,"expirationMonth": 10,"expirationYear": 2024,"cvc": 999}&address=ADDLATER

Response: { "status": { "code": 200, "message": "OK" }, data: { cartId: 7821478372, orderedProducts: orderedProducts, unavailableProducts: unavailableProducts, customerId: params.customerId, } }

### Get Orders

Get the orders of a vendor, or customer.

Endpoint: /**order**

Method: **GET**

Authorization: None

Parameters(Query): id=721837123&userType=customer

Response: { "status": { "code": 200, "message": "OK" }, "data": { orders } }

### Update Order Status

Update the status of the given order.

Endpoint: /**order**

Method: **PATCH**

Authorization: None

Parameters(Query):
userType=customer&userId=721837123&status=Approved&orderId=6as7dasdh

Response: { "status": { "code": 200, "message": "OK" } }

## Filter Data

Update the status of the given order.

Endpoint: /**products**

Method: **GET**

Authorization: None

Parameters(Query): /products?brand=LC Waikiki&color=Green&categories=["Bag"] Warning: "filterCriterias" shows that which attributes can we use the current category.

Response:

```
{
    "status": {
        "code": 200,
        "message": "OK"
    },
    "data": {
        "products": {
            "productList": [
                {
                    "category": [
                        "Men Clothing",
                        "Bag",
                        "Backpack"
                    ],
                    "productInfos": [
                        {
                            "attributes": [
                                {
                                    "name": "size",
                                    "value": "20L"
                                },
                                {
```

```
                    "name": "color",
                    "value": "Green"
                }
            ],
            "stockValue": 68
        }
    ],
    "description": "Men casual 20L backpack. It's smart and very useful.",
    "name": "Men Casual Grey Backpack",
    "price": 89.99,
    "originalPrice": 111.99,
    "imageUrl":
"https://elasticbeanstalk-us-east-2-334058266782.s3.amazonaws.com/images/16089146597058",
    "rating": 3.97,
    "brand": "LC Waikiki",
    "vendor": {
        "name": "Ahmet",
        "rating": 3.22
    },
    "id": "5fe86281a17abd0012d601bb"
  }
],
"filterCriterias": [
  {
```

```
                "name": "size",
                "displayName": "Size",
                "possibleValues": [
                    "70L",
                    "20L"
                ]
            },
            {
                "name": "color",
                "displayName": "Color",
                "possibleValues": [
                    "Green"
                ]
            }
        ]
    }
  }
}
```

## Sort Data

Update the status of the given order.

Endpoint: /**products**

Method: **GET**

Authorization: None

Parameters(Query): ?sortingFactor=name&sortingType=ascending&categories=["Phones"] ** ⚠️ Warning:** "sortingType" can be "ascending" or "descending". "sortingFactor" can be "name", "rating","originalPrice" & "price"

Response:

```
{
    "status": {
        "code": 200,
        "message": "OK"
    },
    "data": {
        "products": {
            "productList": [
                {
                    "category": [
                        "Electronics",
                        "Phones"
                    ],
                    "productInfos": [
                        {
                            "attributes": [
                                {
                                    "name": "screenSize",
                                    "value": "5 Inch"
                                },
            {
                "name": "RAM",
                "value": "4 GB"
            },
            {
                "name": "diskSize",
                "value": "64 GB"
            }
        ],
        "stockValue": 42
    },
    {
        "attributes": [
            {
                "name": "screenSize",
                "value": "5 Inch"
            },
            {
                "name": "RAM",
                "value": "4 GB"
            },
            {
                "name": "diskSize",
                    "value": "128 GB"
                  }
                ],
                "stockValue": 45
              }
            ],
            "description": "Uncover the Ultimate Design. Beauty Beyond the Visual",
            "name": "Iphone 6",
            "price": 5200,
            "originalPrice": 5500,
            "imageUrl":
"https://elasticbeanstalk-us-east-2-334058266782.s3.amazonaws.com/images/160899074721211
",
            "rating": 4,
            "brand": "Apple",
            "vendor": {
              "name": "Pablos",
              "rating": 2.43
            },
            "id": "5fe86281a17abd0012d601eb"
          },
          {
            "category": [
              "Electronics",
        "Phones"
    ],
    "productInfos": [
        {
            "attributes": [
                {
                    "name": "screenSize",
                    "value": "5 Inch"
                },
                {
                    "name": "RAM",
                    "value": "8 GB"
                },
                {
                    "name": "diskSize",
                    "value": "64 GB"
                }
            ],
            "stockValue": 39
        },
        {
            "attributes": [
                {
                    "name": "screenSize",

                    "value": "5 Inch"

                },

                {

                    "name": "RAM",

                    "value": "8 GB"

                },

                {

                    "name": "diskSize",

                    "value": "128 GB"

                }

            ],

            "stockValue": 23

        }

    ],

    "description": "Uncover the Ultimate Design. Beauty Beyond the Visual",

    "name": "Iphone 8",

    "price": 4700,

    "originalPrice": 5000,

    "imageUrl":
"https://elasticbeanstalk-us-east-2-334058266782.s3.amazonaws.com/images/160899074721010
",

    "rating": 4.8,

    "brand": "Apple",
    "vendor": {
        "name": "Pablos",
        "rating": 2.43
    },
    "id": "5fe86281a17abd0012d601ea"
},
{
    "category": [
        "Electronics",
        "Phones"
    ],
    "productInfos": [
        {
            "attributes": [
                {
                    "name": "screenSize",
                    "value": "6 Inch"
                },
                {
                    "name": "RAM",
                    "value": "8 GB"
                },
                {
```

```
            "name": "diskSize",
            "value": "64 GB"
        }
    ],
    "stockValue": 95
},
{
    "attributes": [
        {
            "name": "screenSize",
            "value": "6 Inch"
        },
        {
            "name": "RAM",
            "value": "8 GB"
        },
        {
            "name": "diskSize",
            "value": "128 GB"
        }
    ],
    "stockValue": 95
}
```

```
          ],
          "description": "Uncover the Ultimate Design. Beauty Beyond the Visual",
          "name": "Samsung S10",
          "price": 4500,
          "originalPrice": 5000,
          "imageUrl":
"https://elasticbeanstalk-us-east-2-334058266782.s3.amazonaws.com/images/16089907472078",
          "rating": 4.4,
          "brand": "Samsung",
          "vendor": {
            "name": "AyseTeyze",
            "rating": 3.21
          },
          "id": "5fe86281a17abd0012d601e8"
        },
        {
          "category": [
            "Electronics",
            "Phones"
          ],
          "productInfos": [
            {
              "attributes": [
                {
```

```
            "name": "screenSize",
            "value": "5 Inch"
        },
        {
            "name": "RAM",
            "value": "4 GB"
        },
        {
            "name": "diskSize",
            "value": "64 GB"
        }
    ],
    "stockValue": 47
},
{
    "attributes": [
        {
            "name": "screenSize",
            "value": "5 Inch"
        },
        {
            "name": "RAM",
            "value": "4 GB"
```

```
                },
                {
                    "name": "diskSize",
                    "value": "128 GB"
                }
            ],
            "stockValue": 94
        }
    ],
    "description": "Uncover the Ultimate Design. Beauty Beyond the Visual",
    "name": "Samsung S7",
    "price": 4500,
    "originalPrice": 5000,
    "imageUrl":
"https://elasticbeanstalk-us-east-2-334058266782.s3.amazonaws.com/images/16089907472099",
    "rating": 3.5,
    "brand": "Samsung",
    "vendor": {
        "name": "JohnsShop",
        "rating": 4.23
    },
    "id": "5fe86281a17abd0012d601e9"
},
{
```

```
"category": [
   "Electronics",
   "Phones"
],
"productInfos": [
   {
      "attributes": [
         {
            "name": "screenSize",
            "value": "5.5 Inch"
         },
         {
            "name": "RAM",
            "value": "4 GB"
         },
         {
            "name": "diskSize",
            "value": "64 GB"
         }
      ],
      "stockValue": 21
   },
   {
```

```
            "attributes": [
              {
                "name": "screenSize",
                "value": "5.5 Inch"
              },
              {
                "name": "RAM",
                "value": "4 GB"
              },
              {
                "name": "diskSize",
                "value": "128 GB"
              }
            ],
            "stockValue": 18
          }
        ],
        "description": "Uncover the Ultimate Design. Beauty Beyond the Visual",
        "name": "Samsung S8",
        "price": 2800,
        "originalPrice": 3000,
        "imageUrl": "https://elasticbeanstalk-us-east-2-334058266782.s3.amazonaws.com/images/16089907472046",
        "rating": 4,
        "brand": "Samsung",
        "vendor": {
            "name": "JohnsShop",
            "rating": 4.23
        },
        "id": "5fe86281a17abd0012d601e6"
    },
    {
        "category": [
            "Electronics",
            "Phones"
        ],
        "productInfos": [
            {
                "attributes": [
                    {
                        "name": "screenSize",
                        "value": "5.5 Inch"
                    },
                    {
                        "name": "RAM",
                        "value": "8 GB"
                    },
        {
            "name": "diskSize",
            "value": "64 GB"
        }
    ],
    "stockValue": 54
},
{
    "attributes": [
        {
            "name": "screenSize",
            "value": "5.5 Inch"
        },
        {
            "name": "RAM",
            "value": "8 GB"
        },
        {
            "name": "diskSize",
            "value": "128 GB"
        }
    ],
    "stockValue": 74
```

```
                }
            ],
            "description": "Uncover the Ultimate Design. Beauty Beyond the Visual",
            "name": "Samsung S9",
            "price": 4000,
            "originalPrice": 4200,
            "imageUrl":
"https://elasticbeanstalk-us-east-2-334058266782.s3.amazonaws.com/images/16089907472057",
            "rating": 4.4,
            "brand": "Samsung",
            "vendor": {
                "name": "Ahmet",
                "rating": 3.22
            },
            "id": "5fe86281a17abd0012d601e7"
        }
    ],
    "filterCriterias": [
        {
            "name": "screenSize",
            "displayName": "Screen Size",
            "possibleValues": [
                "5.5 Inch",
                "6 Inch",
                "5 Inch"
              ]
            },
            {
              "name": "RAM",
              "displayName": "RAM",
              "possibleValues": [
                "4 GB",
                "8 GB"
              ]
            },
            {
              "name": "diskSize",
              "displayName": "Disk Size",
              "possibleValues": [
                "64 GB",
                "128 GB"
              ]
            }
          ]
        }
      }
    }
```

## Change Password

Changes password of a user.

Endpoint: **/account-change-password**

Method: **POST**

Authorization: None

Parameters: userType=customer&id=5fe79b54500d4000191358c5&password=3244892

Response: { "status": { "code": 200, "message": "OK" } }

## Update Profile Info

Updates information of a user.

Endpoint: /**account**

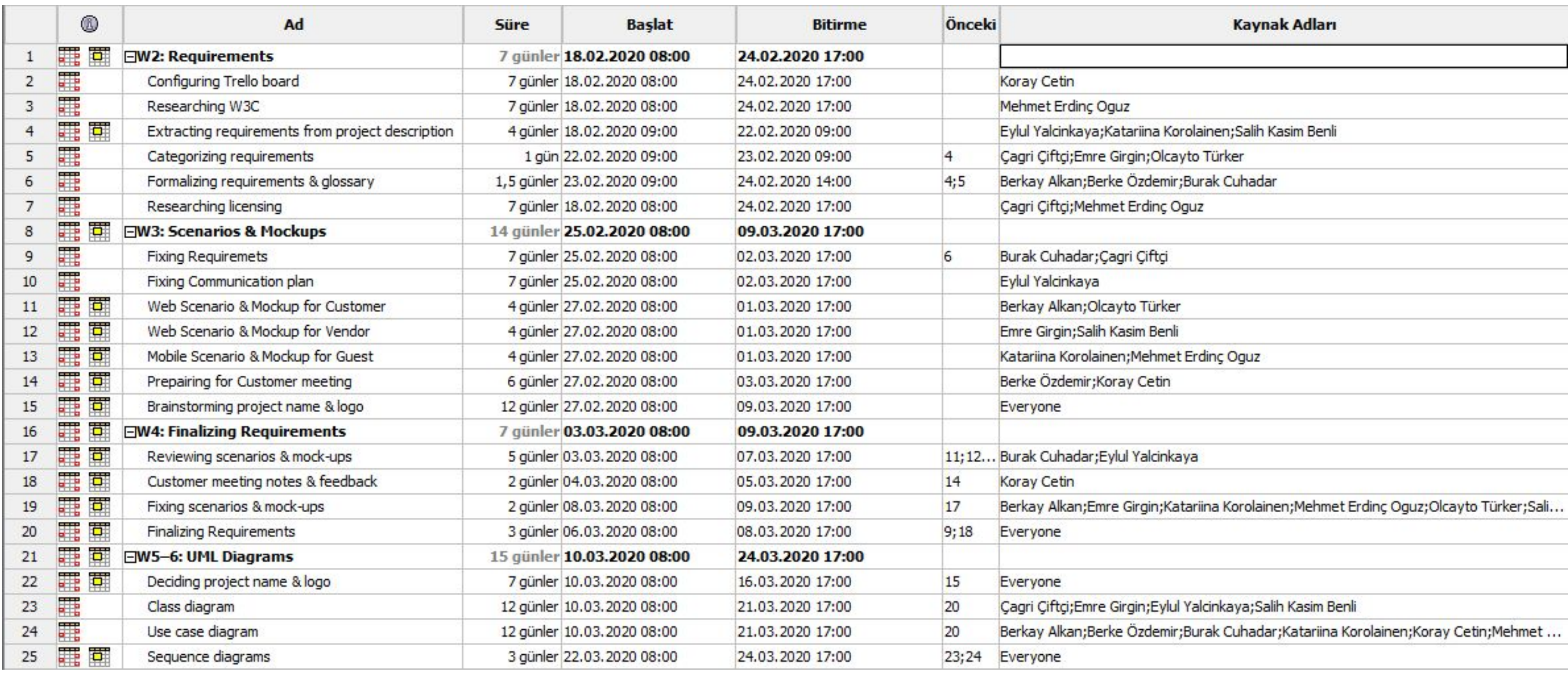
Method: **POST**

Authorization: None

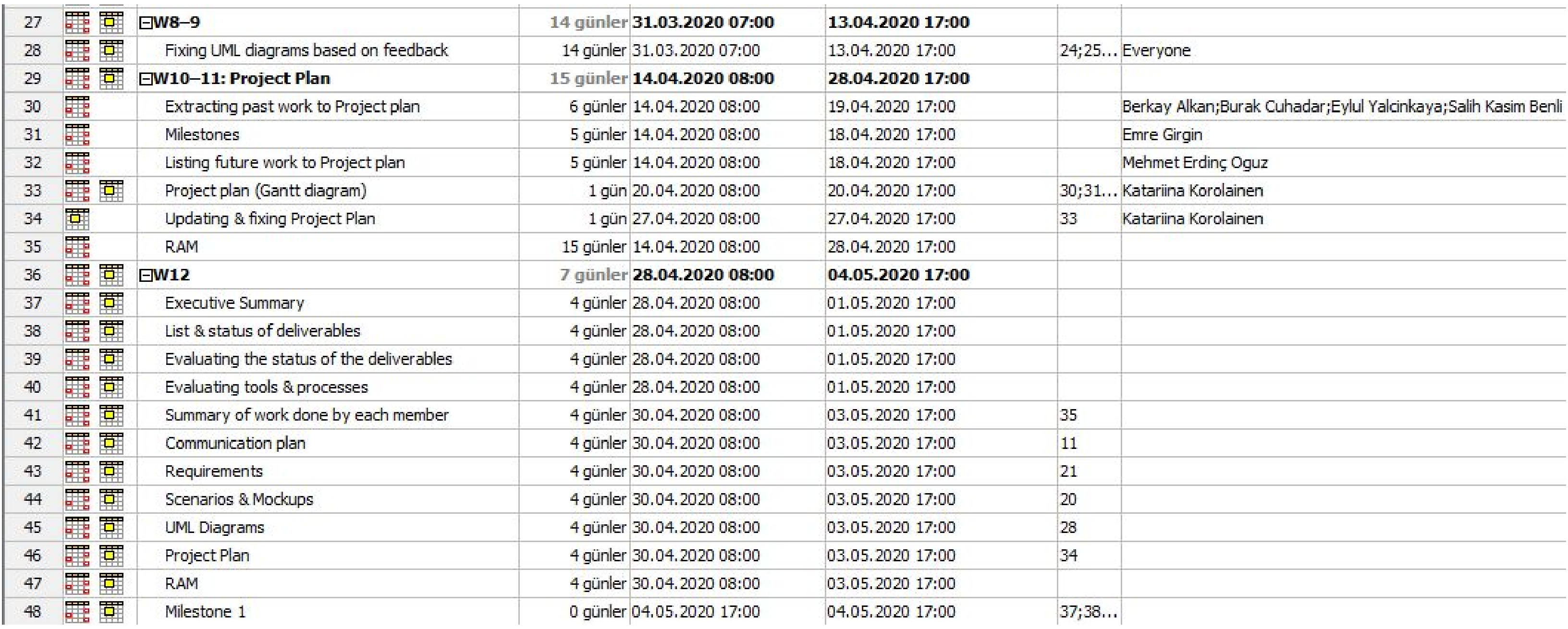
Parameters:
userType=customer&email=jim.gmail@hotmail.com&id=5fe79b54500d4000191358c5&name=Jim&surname=Morrison&gender=male&phoneNumber=3244892

Response: { "status": { "code": 200, "message": "OK" } }

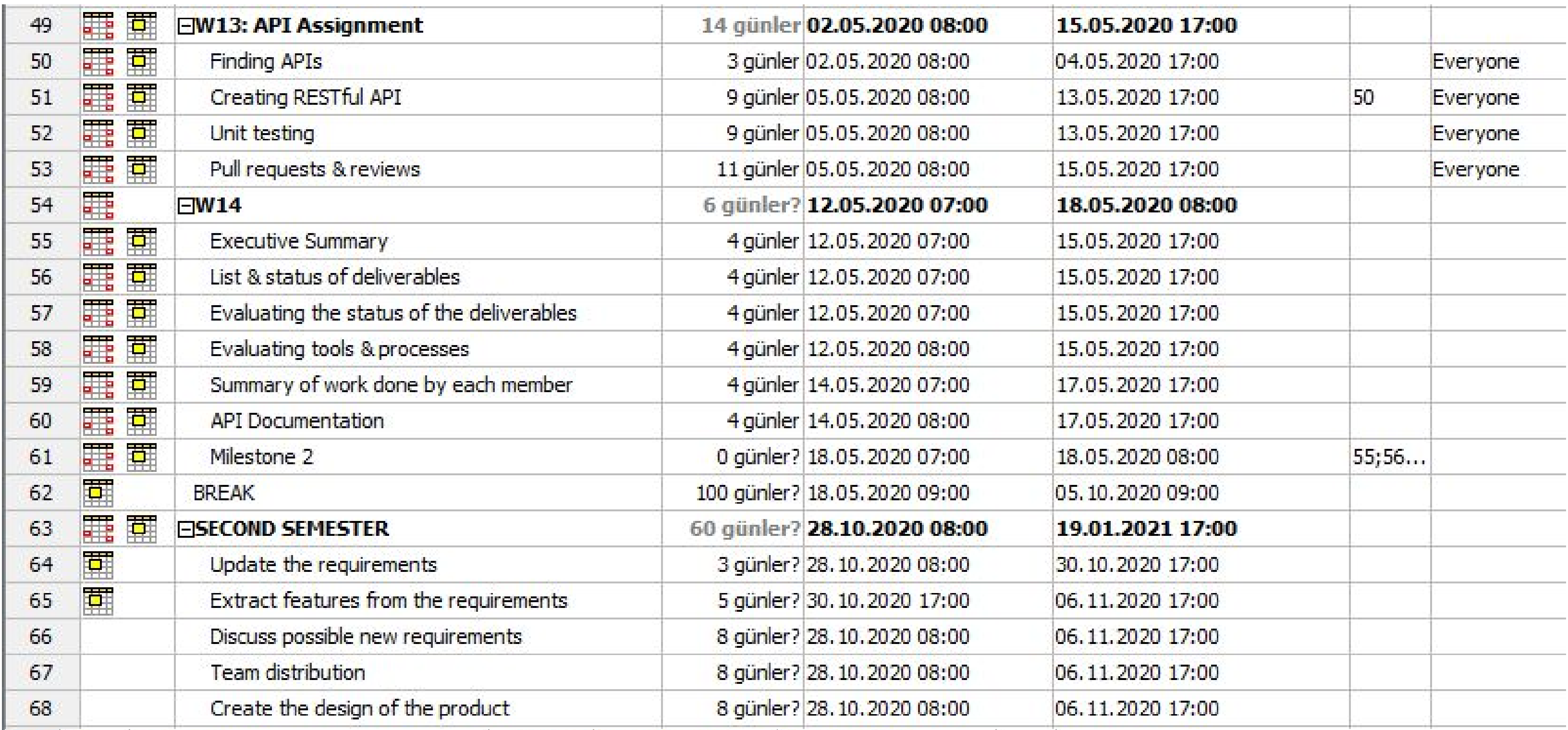
# Project Plan



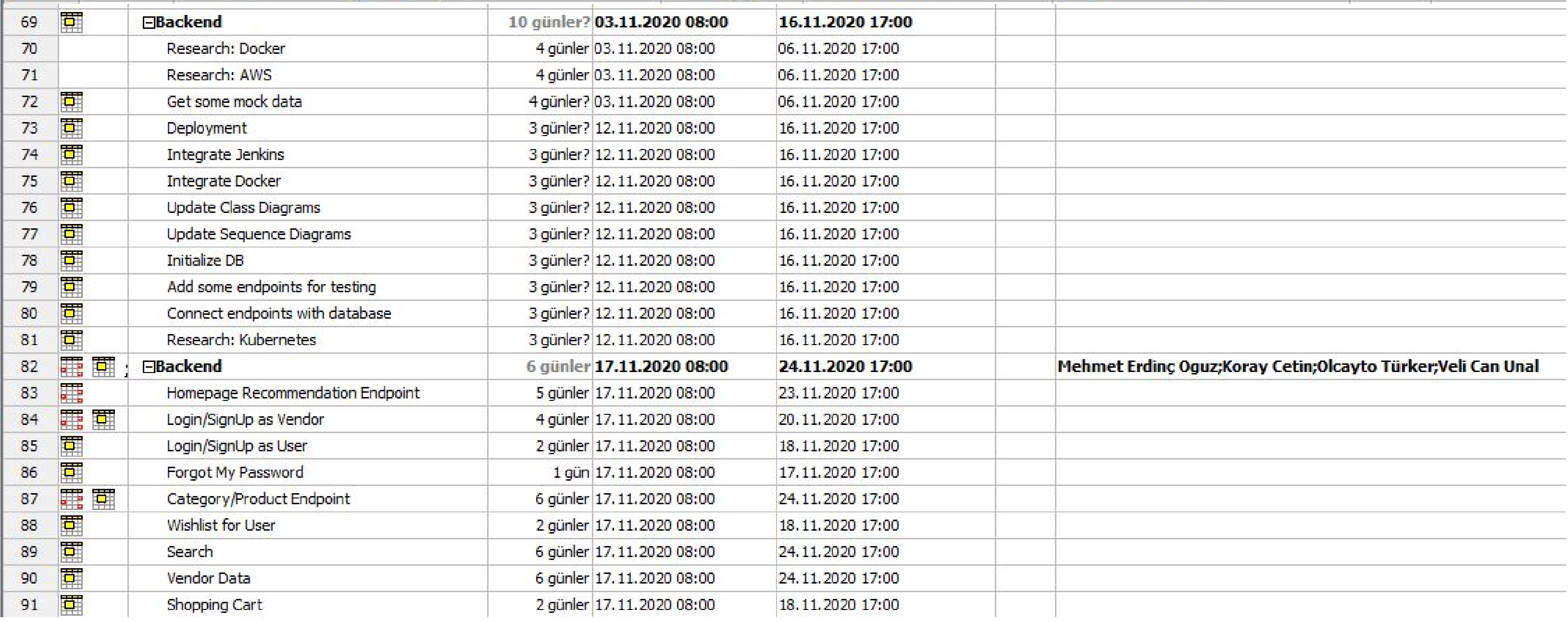
| | Ad | Süre | Başlat | Bitirme | Önceki | Kaynak Adları |
|---|---|---|---|---|---|---|
| 1 | **⊟W2: Requirements** | 7 günler | **18.02.2020 08:00** | **24.02.2020 17:00** | | |
| 2 | Configuring Trello board | 7 günler | 18.02.2020 08:00 | 24.02.2020 17:00 | | Koray Cetin |
| 3 | Researching W3C | 7 günler | 18.02.2020 08:00 | 24.02.2020 17:00 | | Mehmet Erdinç Oguz |
| 4 | Extracting requirements from project description | 4 günler | 18.02.2020 09:00 | 22.02.2020 09:00 | | Eylul Yalcinkaya;Katariina Korolainen;Salih Kasim Benli |
| 5 | Categorizing requirements | 1 gün | 22.02.2020 09:00 | 23.02.2020 09:00 | 4 | Çagri Çiftçi;Emre Girgin;Olcayto Türker |
| 6 | Formalizing requirements & glossary | 1,5 günler | 23.02.2020 09:00 | 24.02.2020 14:00 | 4;5 | Berkay Alkan;Berke Özdemir;Burak Cuhadar |
| 7 | Researching licensing | 7 günler | 18.02.2020 08:00 | 24.02.2020 17:00 | | Çagri Çiftçi;Mehmet Erdinç Oguz |
| 8 | **⊟W3: Scenarios & Mockups** | 14 günler | **25.02.2020 08:00** | **09.03.2020 17:00** | | |
| 9 | Fixing Requiremets | 7 günler | 25.02.2020 08:00 | 02.03.2020 17:00 | 6 | Burak Cuhadar;Çagri Çiftçi |
| 10 | Fixing Communication plan | 7 günler | 25.02.2020 08:00 | 02.03.2020 17:00 | | Eylul Yalcinkaya |
| 11 | Web Scenario & Mockup for Customer | 4 günler | 27.02.2020 08:00 | 01.03.2020 17:00 | | Berkay Alkan;Olcayto Türker |
| 12 | Web Scenario & Mockup for Vendor | 4 günler | 27.02.2020 08:00 | 01.03.2020 17:00 | | Emre Girgin;Salih Kasim Benli |
| 13 | Mobile Scenario & Mockup for Guest | 4 günler | 27.02.2020 08:00 | 01.03.2020 17:00 | | Katariina Korolainen;Mehmet Erdinç Oguz |
| 14 | Prepairing for Customer meeting | 6 günler | 27.02.2020 08:00 | 03.03.2020 17:00 | | Berke Özdemir;Koray Cetin |
| 15 | Brainstorming project name & logo | 12 günler | 27.02.2020 08:00 | 09.03.2020 17:00 | | Everyone |
| 16 | **⊟W4: Finalizing Requirements** | 7 günler | **03.03.2020 08:00** | **09.03.2020 17:00** | | |
| 17 | Reviewing scenarios & mock-ups | 5 günler | 03.03.2020 08:00 | 07.03.2020 17:00 | 11;12... | Burak Cuhadar;Eylul Yalcinkaya |
| 18 | Customer meeting notes & feedback | 2 günler | 04.03.2020 08:00 | 05.03.2020 17:00 | 14 | Koray Cetin |
| 19 | Fixing scenarios & mock-ups | 2 günler | 08.03.2020 08:00 | 09.03.2020 17:00 | 17 | Berkay Alkan;Emre Girgin;Katariina Korolainen;Mehmet Erdinç Oguz;Olcayto Türker;Sali... |
| 20 | Finalizing Requirements | 3 günler | 06.03.2020 08:00 | 08.03.2020 17:00 | 9;18 | Everyone |
| 21 | **⊟W5–6: UML Diagrams** | 15 günler | **10.03.2020 08:00** | **24.03.2020 17:00** | | |
| 22 | Deciding project name & logo | 7 günler | 10.03.2020 08:00 | 16.03.2020 17:00 | 15 | Everyone |
| 23 | Class diagram | 12 günler | 10.03.2020 08:00 | 21.03.2020 17:00 | 20 | Çagri Çiftçi;Emre Girgin;Eylul Yalcinkaya;Salih Kasim Benli |
| 24 | Use case diagram | 12 günler | 10.03.2020 08:00 | 21.03.2020 17:00 | 20 | Berkay Alkan;Berke Özdemir;Burak Cuhadar;Katariina Korolainen;Koray Cetin;Mehmet ... |
| 25 | Sequence diagrams | 3 günler | 22.03.2020 08:00 | 24.03.2020 17:00 | 23;24 | Everyone |



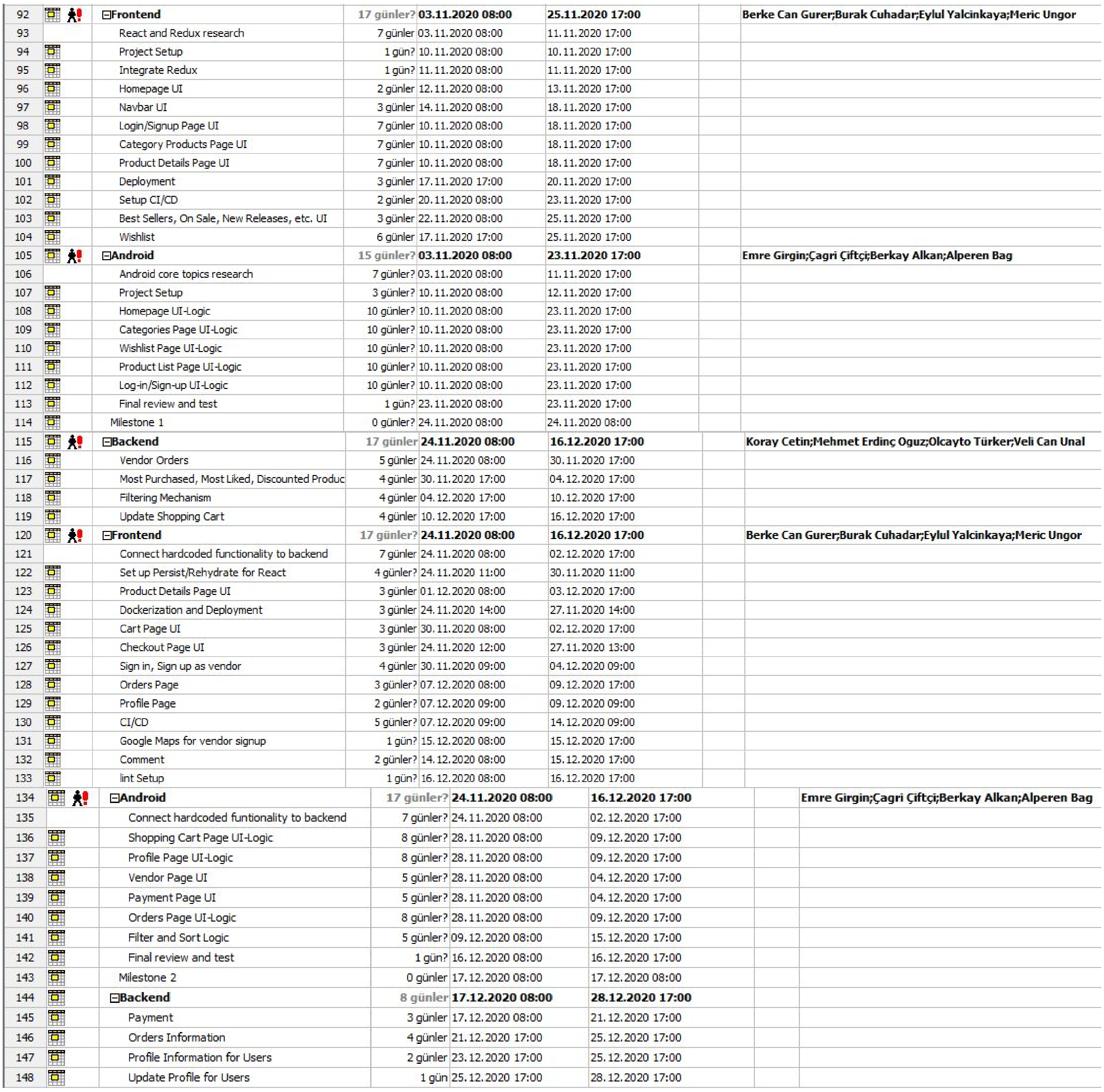
| | | | | | | |
|---|---|---|---|---|---|---|
| 27 | **⊟W8–9** | 14 günler | **31.03.2020 07:00** | **13.04.2020 17:00** | | |
| 28 | Fixing UML diagrams based on feedback | 14 günler | 31.03.2020 07:00 | 13.04.2020 17:00 | 24;25... | Everyone |
| 29 | **⊟W10–11: Project Plan** | 15 günler | **14.04.2020 08:00** | **28.04.2020 17:00** | | |
| 30 | Extracting past work to Project plan | 6 günler | 14.04.2020 08:00 | 19.04.2020 17:00 | | Berkay Alkan;Burak Cuhadar;Eylul Yalcinkaya;Salih Kasim Benli |
| 31 | Milestones | 5 günler | 14.04.2020 08:00 | 18.04.2020 17:00 | | Emre Girgin |
| 32 | Listing future work to Project plan | 5 günler | 14.04.2020 08:00 | 18.04.2020 17:00 | | Mehmet Erdinç Oguz |
| 33 | Project plan (Gantt diagram) | 1 gün | 20.04.2020 08:00 | 20.04.2020 17:00 | 30;31... | Katariina Korolainen |
| 34 | Updating & fixing Project Plan | 1 gün | 27.04.2020 08:00 | 27.04.2020 17:00 | 33 | Katariina Korolainen |
| 35 | RAM | 15 günler | 14.04.2020 08:00 | 28.04.2020 17:00 | | |
| 36 | **⊟W12** | 7 günler | **28.04.2020 08:00** | **04.05.2020 17:00** | | |
| 37 | Executive Summary | 4 günler | 28.04.2020 08:00 | 01.05.2020 17:00 | | |
| 38 | List & status of deliverables | 4 günler | 28.04.2020 08:00 | 01.05.2020 17:00 | | |
| 39 | Evaluating the status of the deliverables | 4 günler | 28.04.2020 08:00 | 01.05.2020 17:00 | | |
| 40 | Evaluating tools & processes | 4 günler | 28.04.2020 08:00 | 01.05.2020 17:00 | | |
| 41 | Summary of work done by each member | 4 günler | 30.04.2020 08:00 | 03.05.2020 17:00 | 35 | |
| 42 | Communication plan | 4 günler | 30.04.2020 08:00 | 03.05.2020 17:00 | 11 | |
| 43 | Requirements | 4 günler | 30.04.2020 08:00 | 03.05.2020 17:00 | 21 | |
| 44 | Scenarios & Mockups | 4 günler | 30.04.2020 08:00 | 03.05.2020 17:00 | 20 | |
| 45 | UML Diagrams | 4 günler | 30.04.2020 08:00 | 03.05.2020 17:00 | 28 | |
| 46 | Project Plan | 4 günler | 30.04.2020 08:00 | 03.05.2020 17:00 | 34 | |
| 47 | RAM | 4 günler | 30.04.2020 08:00 | 03.05.2020 17:00 | | |
| 48 | Milestone 1 | 0 günler | 04.05.2020 17:00 | 04.05.2020 17:00 | 37;38... | |



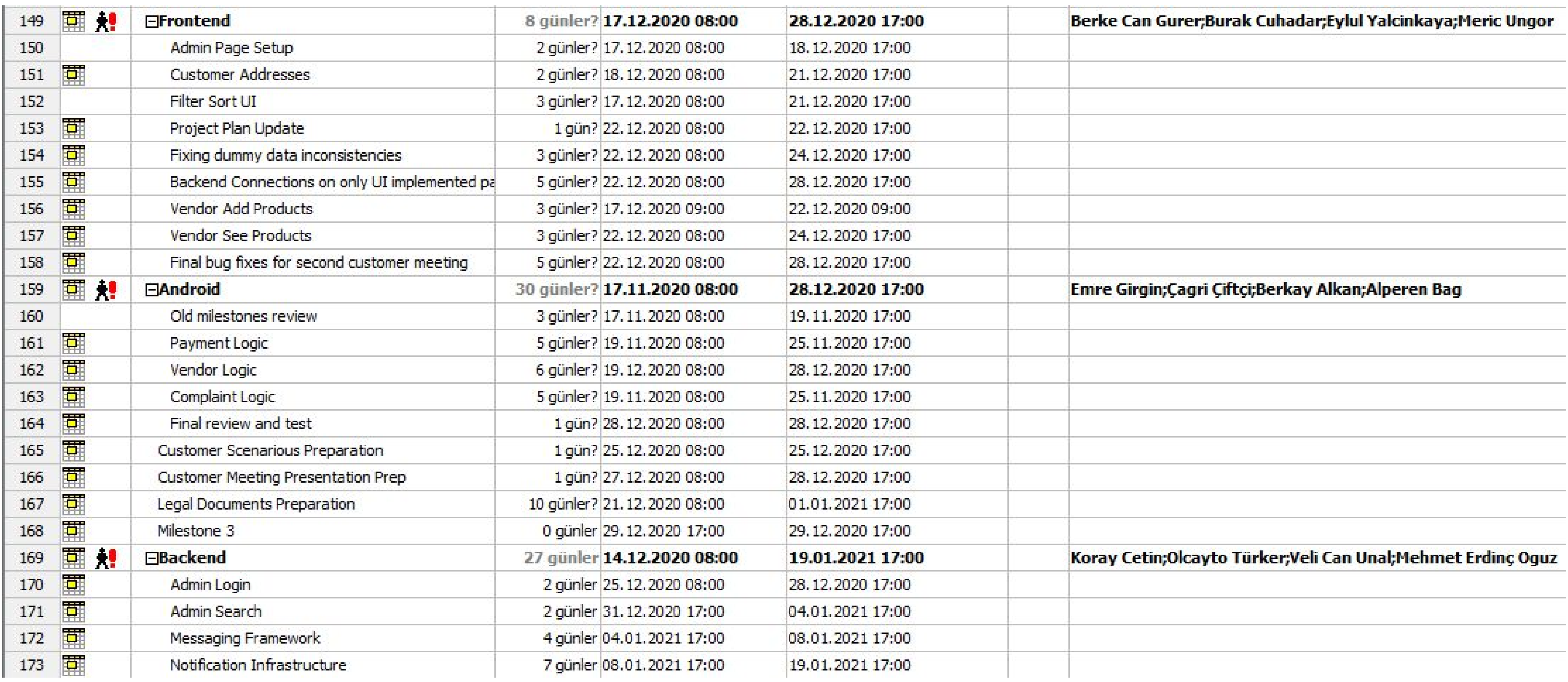
| | | | | | | |
|---|---|---|---|---|---|---|
| 49 | **W13: API Assignment** | **14 günler** | **02.05.2020 08:00** | **15.05.2020 17:00** | | |
| 50 | Finding APIs | 3 günler | 02.05.2020 08:00 | 04.05.2020 17:00 | | Everyone |
| 51 | Creating RESTful API | 9 günler | 05.05.2020 08:00 | 13.05.2020 17:00 | 50 | Everyone |
| 52 | Unit testing | 9 günler | 05.05.2020 08:00 | 13.05.2020 17:00 | | Everyone |
| 53 | Pull requests & reviews | 11 günler | 05.05.2020 08:00 | 15.05.2020 17:00 | | Everyone |
| 54 | **W14** | **6 günler?** | **12.05.2020 07:00** | **18.05.2020 08:00** | | |
| 55 | Executive Summary | 4 günler | 12.05.2020 07:00 | 15.05.2020 17:00 | | |
| 56 | List & status of deliverables | 4 günler | 12.05.2020 07:00 | 15.05.2020 17:00 | | |
| 57 | Evaluating the status of the deliverables | 4 günler | 12.05.2020 07:00 | 15.05.2020 17:00 | | |
| 58 | Evaluating tools & processes | 4 günler | 12.05.2020 08:00 | 15.05.2020 17:00 | | |
| 59 | Summary of work done by each member | 4 günler | 14.05.2020 07:00 | 17.05.2020 17:00 | | |
| 60 | API Documentation | 4 günler | 14.05.2020 08:00 | 17.05.2020 17:00 | | |
| 61 | Milestone 2 | 0 günler? | 18.05.2020 07:00 | 18.05.2020 08:00 | 55;56... | |
| 62 | BREAK | 100 günler? | 18.05.2020 09:00 | 05.10.2020 09:00 | | |
| 63 | **SECOND SEMESTER** | **60 günler?** | **28.10.2020 08:00** | **19.01.2021 17:00** | | |
| 64 | Update the requirements | 3 günler? | 28.10.2020 08:00 | 30.10.2020 17:00 | | |
| 65 | Extract features from the requirements | 5 günler? | 30.10.2020 17:00 | 06.11.2020 17:00 | | |
| 66 | Discuss possible new requirements | 8 günler? | 28.10.2020 08:00 | 06.11.2020 17:00 | | |
| 67 | Team distribution | 8 günler? | 28.10.2020 08:00 | 06.11.2020 17:00 | | |
| 68 | Create the design of the product | 8 günler? | 28.10.2020 08:00 | 06.11.2020 17:00 | | |



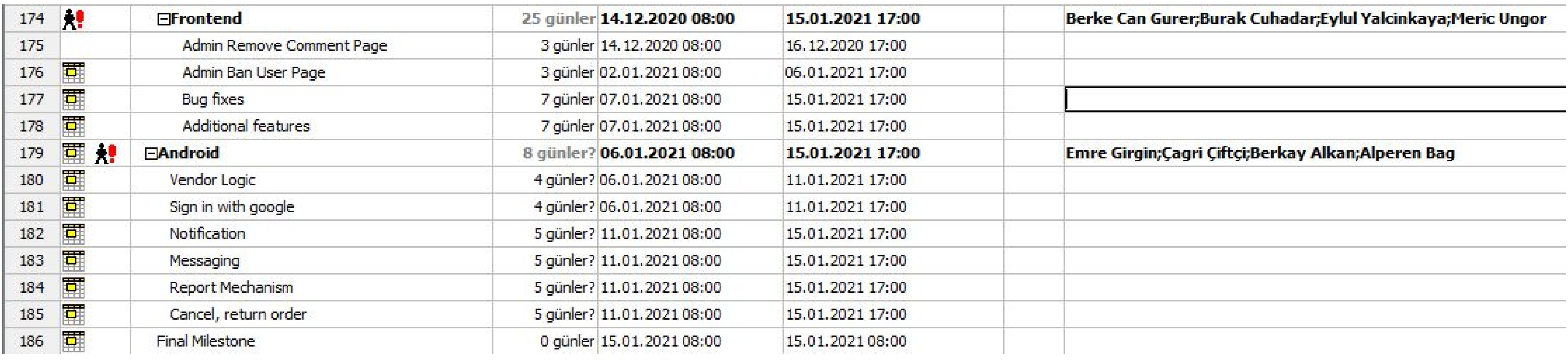
| | | | | | | |
|---|---|---|---|---|---|---|
| 69 | **Backend** | **10 günler?** | **03.11.2020 08:00** | **16.11.2020 17:00** | | |
| 70 | Research: Docker | 4 günler | 03.11.2020 08:00 | 06.11.2020 17:00 | | |
| 71 | Research: AWS | 4 günler | 03.11.2020 08:00 | 06.11.2020 17:00 | | |
| 72 | Get some mock data | 4 günler? | 03.11.2020 08:00 | 06.11.2020 17:00 | | |
| 73 | Deployment | 3 günler? | 12.11.2020 08:00 | 16.11.2020 17:00 | | |
| 74 | Integrate Jenkins | 3 günler? | 12.11.2020 08:00 | 16.11.2020 17:00 | | |
| 75 | Integrate Docker | 3 günler? | 12.11.2020 08:00 | 16.11.2020 17:00 | | |
| 76 | Update Class Diagrams | 3 günler? | 12.11.2020 08:00 | 16.11.2020 17:00 | | |
| 77 | Update Sequence Diagrams | 3 günler? | 12.11.2020 08:00 | 16.11.2020 17:00 | | |
| 78 | Initialize DB | 3 günler? | 12.11.2020 08:00 | 16.11.2020 17:00 | | |
| 79 | Add some endpoints for testing | 3 günler? | 12.11.2020 08:00 | 16.11.2020 17:00 | | |
| 80 | Connect endpoints with database | 3 günler? | 12.11.2020 08:00 | 16.11.2020 17:00 | | |
| 81 | Research: Kubernetes | 3 günler? | 12.11.2020 08:00 | 16.11.2020 17:00 | | |
| 82 | **Backend** | **6 günler** | **17.11.2020 08:00** | **24.11.2020 17:00** | | **Mehmet Erdinç Oguz;Koray Cetin;Olcayto Türker;Veli Can Unal** |
| 83 | Homepage Recommendation Endpoint | 5 günler | 17.11.2020 08:00 | 23.11.2020 17:00 | | |
| 84 | Login/SignUp as Vendor | 4 günler | 17.11.2020 08:00 | 20.11.2020 17:00 | | |
| 85 | Login/SignUp as User | 2 günler | 17.11.2020 08:00 | 18.11.2020 17:00 | | |
| 86 | Forgot My Password | 1 gün | 17.11.2020 08:00 | 17.11.2020 17:00 | | |
| 87 | Category/Product Endpoint | 6 günler | 17.11.2020 08:00 | 24.11.2020 17:00 | | |
| 88 | Wishlist for User | 2 günler | 17.11.2020 08:00 | 18.11.2020 17:00 | | |
| 89 | Search | 6 günler | 17.11.2020 08:00 | 24.11.2020 17:00 | | |
| 90 | Vendor Data | 6 günler | 17.11.2020 08:00 | 24.11.2020 17:00 | | |
| 91 | Shopping Cart | 2 günler | 17.11.2020 08:00 | 18.11.2020 17:00 | | |



| | | | | | | |
|---|---|---|---|---|---|---|
| 92 | **⊟Frontend** | **17 günler?** | **03.11.2020 08:00** | **25.11.2020 17:00** | | **Berke Can Gurer;Burak Cuhadar;Eylul Yalcinkaya;Meric Ungor** |
| 93 | React and Redux research | 7 günler | 03.11.2020 08:00 | 11.11.2020 17:00 | | |
| 94 | Project Setup | 1 gün? | 10.11.2020 08:00 | 10.11.2020 17:00 | | |
| 95 | Integrate Redux | 1 gün? | 11.11.2020 08:00 | 11.11.2020 17:00 | | |
| 96 | Homepage UI | 2 günler | 12.11.2020 08:00 | 13.11.2020 17:00 | | |
| 97 | Navbar UI | 3 günler | 14.11.2020 08:00 | 18.11.2020 17:00 | | |
| 98 | Login/Signup Page UI | 7 günler | 10.11.2020 08:00 | 18.11.2020 17:00 | | |
| 99 | Category Products Page UI | 7 günler | 10.11.2020 08:00 | 18.11.2020 17:00 | | |
| 100 | Product Details Page UI | 7 günler | 10.11.2020 08:00 | 18.11.2020 17:00 | | |
| 101 | Deployment | 3 günler | 17.11.2020 17:00 | 20.11.2020 17:00 | | |
| 102 | Setup CI/CD | 2 günler | 20.11.2020 08:00 | 23.11.2020 17:00 | | |
| 103 | Best Sellers, On Sale, New Releases, etc. UI | 3 günler | 22.11.2020 08:00 | 25.11.2020 17:00 | | |
| 104 | Wishlist | 6 günler | 17.11.2020 17:00 | 25.11.2020 17:00 | | |
| 105 | **⊟Android** | **15 günler?** | **03.11.2020 08:00** | **23.11.2020 17:00** | | **Emre Girgin;Çagri Çiftçi;Berkay Alkan;Alperen Bag** |
| 106 | Android core topics research | 7 günler? | 03.11.2020 08:00 | 11.11.2020 17:00 | | |
| 107 | Project Setup | 3 günler? | 10.11.2020 08:00 | 12.11.2020 17:00 | | |
| 108 | Homepage UI-Logic | 10 günler? | 10.11.2020 08:00 | 23.11.2020 17:00 | | |
| 109 | Categories Page UI-Logic | 10 günler? | 10.11.2020 08:00 | 23.11.2020 17:00 | | |
| 110 | Wishlist Page UI-Logic | 10 günler? | 10.11.2020 08:00 | 23.11.2020 17:00 | | |
| 111 | Product List Page UI-Logic | 10 günler? | 10.11.2020 08:00 | 23.11.2020 17:00 | | |
| 112 | Log-in/Sign-up UI-Logic | 10 günler? | 10.11.2020 08:00 | 23.11.2020 17:00 | | |
| 113 | Final review and test | 1 gün? | 23.11.2020 08:00 | 23.11.2020 17:00 | | |
| 114 | Milestone 1 | 0 günler? | 24.11.2020 08:00 | 24.11.2020 08:00 | | |
| 115 | **⊟Backend** | **17 günler** | **24.11.2020 08:00** | **16.12.2020 17:00** | | **Koray Cetin;Mehmet Erdinç Oguz;Olcayto Türker;Veli Can Unal** |
| 116 | Vendor Orders | 5 günler | 24.11.2020 08:00 | 30.11.2020 17:00 | | |
| 117 | Most Purchased, Most Liked, Discounted Produc | 4 günler | 30.11.2020 17:00 | 04.12.2020 17:00 | | |
| 118 | Filtering Mechanism | 4 günler | 04.12.2020 17:00 | 10.12.2020 17:00 | | |
| 119 | Update Shopping Cart | 4 günler | 10.12.2020 17:00 | 16.12.2020 17:00 | | |
| 120 | **⊟Frontend** | **17 günler?** | **24.11.2020 08:00** | **16.12.2020 17:00** | | **Berke Can Gurer;Burak Cuhadar;Eylul Yalcinkaya;Meric Ungor** |
| 121 | Connect hardcoded functionality to backend | 7 günler | 24.11.2020 08:00 | 02.12.2020 17:00 | | |
| 122 | Set up Persist/Rehydrate for React | 4 günler? | 24.11.2020 11:00 | 30.11.2020 11:00 | | |
| 123 | Product Details Page UI | 3 günler | 01.12.2020 08:00 | 03.12.2020 17:00 | | |
| 124 | Dockerization and Deployment | 3 günler | 24.11.2020 14:00 | 27.11.2020 14:00 | | |
| 125 | Cart Page UI | 3 günler | 30.11.2020 08:00 | 02.12.2020 17:00 | | |
| 126 | Checkout Page UI | 3 günler | 24.11.2020 12:00 | 27.11.2020 13:00 | | |
| 127 | Sign in, Sign up as vendor | 4 günler | 30.11.2020 09:00 | 04.12.2020 09:00 | | |
| 128 | Orders Page | 3 günler? | 07.12.2020 08:00 | 09.12.2020 17:00 | | |
| 129 | Profile Page | 2 günler? | 07.12.2020 09:00 | 09.12.2020 09:00 | | |
| 130 | CI/CD | 5 günler? | 07.12.2020 09:00 | 14.12.2020 09:00 | | |
| 131 | Google Maps for vendor signup | 1 gün? | 15.12.2020 08:00 | 15.12.2020 17:00 | | |
| 132 | Comment | 2 günler? | 14.12.2020 08:00 | 15.12.2020 17:00 | | |
| 133 | lint Setup | 1 gün? | 16.12.2020 08:00 | 16.12.2020 17:00 | | |
| 134 | **⊟Android** | **17 günler?** | **24.11.2020 08:00** | **16.12.2020 17:00** | | **Emre Girgin;Çagri Çiftçi;Berkay Alkan;Alperen Bag** |
| 135 | Connect hardcoded funtionality to backend | 7 günler? | 24.11.2020 08:00 | 02.12.2020 17:00 | | |
| 136 | Shopping Cart Page UI-Logic | 8 günler? | 28.11.2020 08:00 | 09.12.2020 17:00 | | |
| 137 | Profile Page UI-Logic | 8 günler? | 28.11.2020 08:00 | 09.12.2020 17:00 | | |
| 138 | Vendor Page UI | 5 günler? | 28.11.2020 08:00 | 04.12.2020 17:00 | | |
| 139 | Payment Page UI | 5 günler? | 28.11.2020 08:00 | 04.12.2020 17:00 | | |
| 140 | Orders Page UI-Logic | 8 günler? | 28.11.2020 08:00 | 09.12.2020 17:00 | | |
| 141 | Filter and Sort Logic | 5 günler? | 09.12.2020 08:00 | 15.12.2020 17:00 | | |
| 142 | Final review and test | 1 gün? | 16.12.2020 08:00 | 16.12.2020 17:00 | | |
| 143 | Milestone 2 | 0 günler | 17.12.2020 08:00 | 17.12.2020 08:00 | | |
| 144 | **⊟Backend** | **8 günler** | **17.12.2020 08:00** | **28.12.2020 17:00** | | |
| 145 | Payment | 3 günler | 17.12.2020 08:00 | 21.12.2020 17:00 | | |
| 146 | Orders Information | 4 günler | 21.12.2020 17:00 | 25.12.2020 17:00 | | |
| 147 | Profile Information for Users | 2 günler | 23.12.2020 17:00 | 25.12.2020 17:00 | | |
| 148 | Update Profile for Users | 1 gün | 25.12.2020 17:00 | 28.12.2020 17:00 | | |



| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 149 | ▦ ❗ | **⊟Frontend** | 8 günler? | **17.12.2020 08:00** | **28.12.2020 17:00** | | **Berke Can Gurer;Burak Cuhadar;Eylul Yalcinkaya;Meric Ungor** |
| 150 | | Admin Page Setup | 2 günler? | 17.12.2020 08:00 | 18.12.2020 17:00 | | |
| 151 | ▦ | Customer Addresses | 2 günler? | 18.12.2020 08:00 | 21.12.2020 17:00 | | |
| 152 | | Filter Sort UI | 3 günler? | 17.12.2020 08:00 | 21.12.2020 17:00 | | |
| 153 | ▦ | Project Plan Update | 1 gün? | 22.12.2020 08:00 | 22.12.2020 17:00 | | |
| 154 | ▦ | Fixing dummy data inconsistencies | 3 günler? | 22.12.2020 08:00 | 24.12.2020 17:00 | | |
| 155 | ▦ | Backend Connections on only UI implemented pa | 5 günler? | 22.12.2020 08:00 | 28.12.2020 17:00 | | |
| 156 | ▦ | Vendor Add Products | 3 günler? | 17.12.2020 09:00 | 22.12.2020 09:00 | | |
| 157 | ▦ | Vendor See Products | 3 günler? | 22.12.2020 08:00 | 24.12.2020 17:00 | | |
| 158 | ▦ | Final bug fixes for second customer meeting | 5 günler? | 22.12.2020 08:00 | 28.12.2020 17:00 | | |
| 159 | ▦ ❗ | **⊟Android** | 30 günler? | **17.11.2020 08:00** | **28.12.2020 17:00** | | **Emre Girgin;Çagri Çiftçi;Berkay Alkan;Alperen Bag** |
| 160 | | Old milestones review | 3 günler? | 17.11.2020 08:00 | 19.11.2020 17:00 | | |
| 161 | ▦ | Payment Logic | 5 günler? | 19.11.2020 08:00 | 25.11.2020 17:00 | | |
| 162 | ▦ | Vendor Logic | 6 günler? | 19.12.2020 08:00 | 28.12.2020 17:00 | | |
| 163 | ▦ | Complaint Logic | 5 günler? | 19.11.2020 08:00 | 25.11.2020 17:00 | | |
| 164 | ▦ | Final review and test | 1 gün? | 28.12.2020 08:00 | 28.12.2020 17:00 | | |
| 165 | ▦ | Customer Scenarious Preparation | 1 gün? | 25.12.2020 08:00 | 25.12.2020 17:00 | | |
| 166 | ▦ | Customer Meeting Presentation Prep | 1 gün? | 27.12.2020 08:00 | 28.12.2020 17:00 | | |
| 167 | ▦ | Legal Documents Preparation | 10 günler? | 21.12.2020 08:00 | 01.01.2021 17:00 | | |
| 168 | ▦ | Milestone 3 | 0 günler | 29.12.2020 17:00 | 29.12.2020 17:00 | | |
| 169 | ▦ ❗ | **⊟Backend** | 27 günler | **14.12.2020 08:00** | **19.01.2021 17:00** | | **Koray Cetin;Olcayto Türker;Veli Can Unal;Mehmet Erdinç Oguz** |
| 170 | ▦ | Admin Login | 2 günler | 25.12.2020 08:00 | 28.12.2020 17:00 | | |
| 171 | ▦ | Admin Search | 2 günler | 31.12.2020 17:00 | 04.01.2021 17:00 | | |
| 172 | ▦ | Messaging Framework | 4 günler | 04.01.2021 17:00 | 08.01.2021 17:00 | | |
| 173 | ▦ | Notification Infrastructure | 7 günler | 08.01.2021 17:00 | 19.01.2021 17:00 | | |



| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 174 | ❗ | **⊟Frontend** | 25 günler | **14.12.2020 08:00** | **15.01.2021 17:00** | | **Berke Can Gurer;Burak Cuhadar;Eylul Yalcinkaya;Meric Ungor** |
| 175 | | Admin Remove Comment Page | 3 günler | 14.12.2020 08:00 | 16.12.2020 17:00 | | |
| 176 | ▦ | Admin Ban User Page | 3 günler | 02.01.2021 08:00 | 06.01.2021 17:00 | | |
| 177 | ▦ | Bug fixes | 7 günler | 07.01.2021 08:00 | 15.01.2021 17:00 | | |
| 178 | ▦ | Additional features | 7 günler | 07.01.2021 08:00 | 15.01.2021 17:00 | | |
| 179 | ▦ ❗ | **⊟Android** | 8 günler? | **06.01.2021 08:00** | **15.01.2021 17:00** | | **Emre Girgin;Çagri Çiftçi;Berkay Alkan;Alperen Bag** |
| 180 | ▦ | Vendor Logic | 4 günler? | 06.01.2021 08:00 | 11.01.2021 17:00 | | |
| 181 | ▦ | Sign in with google | 4 günler? | 06.01.2021 08:00 | 11.01.2021 17:00 | | |
| 182 | ▦ | Notification | 5 günler? | 11.01.2021 08:00 | 15.01.2021 17:00 | | |
| 183 | ▦ | Messaging | 5 günler? | 11.01.2021 08:00 | 15.01.2021 17:00 | | |
| 184 | ▦ | Report Mechanism | 5 günler? | 11.01.2021 08:00 | 15.01.2021 17:00 | | |
| 185 | ▦ | Cancel, return order | 5 günler? | 11.01.2021 08:00 | 15.01.2021 17:00 | | |
| 186 | ▦ | Final Milestone | 0 günler | 15.01.2021 08:00 | 15.01.2021 08:00 | | |

# User Scenarios

## Android Scenario

**Persona:** My name is Lila. I am the daughter-in-law of Battal Dede. I am new in the family. My husband is the most successful son of Battal Dede and I am not as 'conservative' as the rest of the family. So the family's first impression is not great about me. So I decided to bribe my way into the family starting with the elder of the family. I am already a regular customer of BUYO and also downloaded its app to my phone.
**Preconditions:** Lila has an android phone and internet connection.
**Goals:**.Her goal is to buy a nice expensive gift for Battal Dede.
**Actions:**

1. I am starting as logged in.
2. I am going to the cart to check  if there is something left in there from earlier.
3. I realize there is an old product and delete that from the cart.
4. I go to mens clothing category to choose a product.
5. I sort the list by price to buy something expensive and impress the family.
6. I like the first one which is a suit from Altınyıldız and I click the cart icon.
7. Because my dear Battal Dede wears large size, I am choosing large
8. After adding the suit to the cart, I go to the cart page.
9. I want to check the cargo price, so I click the icon next to the total price.
10. Everything looks fine. I go to the checkout page to buy it immediately.
11. I want to add the home address of Battal Dede. I click the add or update button.
12. I can see my address on that page, after that I click the add address button to add a new address.
13. I am filling the needed information then clicking the add address button.
14. I can see the new address then I go back to checkout page.
15. I select the new address via dropdown.
16. I fill the credit card information.
17. I click the pay button.
18. After that I go to the order page to check if everything went ok.

19. I can see my order, everything looks fine.
20. However there is also one more order that I don't remember.
21. I decide to change my password just in case.
22. I go to change the password page by clicking on the profile tab then clicking the change password button.
23. I enter my new password then an error toast appears because they didn't match.
24. I enter it again then click the change password button.

## Frontend Scenario

**Persona:** Battal Dede, the elderly protagonist of our scenarios, is a 70-year-old man with 3 sons and 8 grandchildren. The last time he used BUYO, he was just trying out the system, and didn't get too deep into it. But after seeing how convenient BUYO is, he asks one of his grandchildren, Hakim, to come and help him figure out more things; so that he can shop for more items.
**Preconditions:** Battal Dede has a laptop with internet connection. A web browser is installed in his laptop.
**Goals:** Battal Dede's goal is to learn how to shop for an item. He also wants to buy a thank-you gift for Hakim. Hakim's goal is to help Battal Dede.
**Actions:**

1. Battal Dede and Hakim start as logged in.
2. Battal Dede wishes to search for a thank-you gift for Hakim, since Hakim has been so helpful to him.
3. Hakim shows Battal Dede the search functionality, by using it.
4. Hakim likes one of the products that appears, and asks Battal Dede to buy that.
5. Battal Dede clicks on the product, and looks at various product details on the product page.
6. Battal Dede reads some comments on the product page.
7. Battal Dede adds the product to his cart, choosing a size and color while doing so.
8. Battal Dede clicks on his cart. Instead of going to his cart, he is redirected to his profile, which has an alert and asks for name-surname information.
9. Battal Dede asks Hakim what happened. Knowing English, Hakim reads the alert, and says that they need to fill out name and surname information before buying anything.

10. Battal Dede updates his profile information accordingly.
11. Battal Dede clicks cart again, and proceeds to checkout.
12. Battal Dede selects an address. He has bad eyes, so he inputs wrong credit card information. He also doesn't see the terms and conditions box.
13. When he tries to buy, nothing happens. He asks for Hakim's help again. Hakim sees the red borders around the credit card information, and corrects the mistake by entering the correct credit card information.
14. When they try to buy again, they see an alert. Looking at the alert, Hakim realizes that they need to check the terms and conditions box.
15. Since he is a teenager, Hakim checks the terms and conditions box without reading.
16. They proceed to checkout once again. This time, it succeeds.
17. Since Battal Dede is mistrustful of technology, he wants to make sure that the order is really received.
18. Battal Dede goes to his orders, and looks at the order that he just made. He is now relieved.
19. As a bonus lesson for Battal Dede, Hakim shows that there is also a tab with delivered orders, where Battal dede can rate the products and eventually message the vendor.
20. Battal Dede loves this, and gives feedback for his last order by giving it a low rating.
21. Battal dede also writes a short comment and justifies his rating.

**Acceptance Criteria:**

- **1.1.1.1.5.1** Customers shall be able to add products to their cart.
- **1.1.1.1.5.3** Customers shall be able to purchase all items from their cart at once.
- **1.1.1.1.6.1** Customers shall be able to see all their purchased items in the Orders page.
- **1.2.2.1** Customers shall be able to comment on the products that they have purchased, after the products have been delivered.
- **1.2.2.2** Customers shall be able to rate the products that they have purchased, after the products have been delivered.
- **1.2.2.5** The names of the customers shall be visible only with their initials on top of the comments they make for privacy. ex: M**** U****

- **1.2.8.3** System shall not accept the payment if the customer does not approve an e-commerce shopping agreement.

## Code Structure

**Frontend:** Frontend team's main branch is *web-dev*. This branch is where we merge all the personal branches and solve merge conflicts. Everyone has their own branch: *web-eylul, web-meric, web-burak*. These personal branches are used for developing the assigned features. We use Github's issues system to assign and manage tasks. We try to assign independent features to different people so as to prevent merge conflicts and bugs. Once a feature is done, a pull request is opened from the personal branch to web-dev. The pull request is reviewed by at least one other person from the team. If the code is well written and works fine, it is then merged. Code structure is as follows:

- **web**
    - **public**: public images
    - **src**
        - **components:** components that are used in pages
        - **images**: icons, images that need to be in src folder
        - **pages:** webpages
        - **redux**: redux related - actions and reducers
        - **services:** API calls
        - **util:** utility functions

**Backend:** We used different branches for each purpose, some branches had small changes and some had a lot. We merge all of the branches directly to the master by requiring at least 2 people's review. Here are the branch details

- **#231-Microservice:** Microservice structure is configured.
- **Profile-Info-Endpoint:** Get profile info endpoint is added.
- **Vendor-Signup-Update:** Website and coordinate information is added to vendor signup.
- **BKND-#276-ProfileUpdate:** Update profile endpoint is added.
- **BCKND-CI/CD-Integration-#281:** BackendDeploy.yml file is added.
- **BKND-#311-AddProduct:** Vendor add product and upload image endpoints are implemented.

- **BKND-#279-Comment:** Add comment and delete comment endpoints are implemented, product response is changed accordingly.
- **BKND-#314-AddMockProducts:** Mock clothing products are added.
- **(BKND)-Create-Electronics-Category-Data:** Mock electronics products are added.
- **BKND-#338-Name:** Name field is added to customer signup.
- **(BCKND)-Filter-&-Sort-Problem-#341:** Filter and sort functions are adapted to new attributes structure.
- **Change-Password:** Change password endpoint is implemented.
- **Add-Update-Delete-Address:** Add, update and delete address endpoints are implemented.
- **(BKND)-Provide-Sort-&-Filter-Criterias-According-to-Categories-#320:** Filtering criterias are returned in getProducts endpoint
- **(BKND)-Filter-&-Sort-Improvement-#353:** Extra information is added to filter criterias field
- **erdinc-backend:** Checkout and cart functionalities are implemented.
- **BKND-Comment:** Rate field is added to the comment endpoint.
- **Update-Account-Info:** Surname and phone number fields are added to update account info endpoint.
- **(BKND)-Return-Product-Attributes-in-Another-Field-#360:** Product response structure is changed, attribute fields are restructured for getProduct endpoint.
- **(BKND)-Products-are-not-sorted-by-price-#366:** Product price sort bug is fixed.
- **BKND-#344-Test:** Test structure is initialized and account service tests are written

Code structure is as follows:

- **backend**
  - **nginx.conf:** Proxies the request to relevant services
  - **Docker-compose.yml:** Runs each service's dockerfile
  - **product_service/**
  - **account_service/**
  - **vendor_service/**
  - **checkout_service/**
  - **db_service/**
  - **Interaction_service/**
    - **constants/:** Constants
    - **models/:** Database collection models
    - **routes/:** Endpoints
    - **views/:** Business logic
    - **index.js:** Main class
    - **.env:** Stores key constants
    - **Dockerfile:** Docker config file
    - **package.json:** App dependencies and config

**Android:** Android team's main branch is *android-dev*. This branch is where we merge all the personal branches and solve merge conflicts. Everyone has their own branches such as *android-dev-profilepage, android-dev-cart, android-dev-sortFilter etc*. These personal branches are used for developing the assigned features. We use Github's issues system to assign and manage tasks. We try to assign independent features to different people so as to prevent merge conflicts and bugs. Once a feature is done, a pull request is opened from the personal branch to *android-dev*. The pull requests are reviewed by all team members in a zoom meeting. If the code is well written and works fine, it is then merged.

Code structure is as follows:

- **app -> src -> main**
    - **res**: It is a conventional android folder to store xmls, images, drawables, colors, styles etc.
    - **buyo**
        - **api**: Endpoints Interface and Service wrapper (Api calls 1)
        - **base**: Base classes and fragment transactions helpers
        - **datamanager**
            - **network** api calls helper classes (Retrofit)
            - **repository**: (Api calls 2)
            - **shared_pref**: Storing primitive data in local
        - **dependencyinjection**: Some module classes about di
        - **ui**: Fragments(Pages in the app)
        - **util**: Some utility classes and functions such as base dialog
          **viewmodel**: Storing data coming from api
        - **vo**: Data models according to api
        - **widget**: Custom views such as navigation bar

- **Buyo -> app -> src -> test/AndroidTest:** All tests will be written in this folder
- **Buyo -> Gradle:** Gradle files such as version.gradle, build.gradle

# Evaluation of the Tools and Managing the Project

**Evaluation of the Tools Used by the General Team:**

**Evaluation of the Tools Used by the Frontend Team:**.
We already mentioned most of the tools we used and most aspects of our project management in the first customer milestone report. But there were of course some additions and alterations to these.

We mentioned the use of Redux in our project before. But we made an alteration to its extent. We started using persist/rehydrate which is a library that allows us to save the state of our Redux store into the local storage of the browser. To elaborate, the information we were using on the redux store disappeared when we closed the tab or refreshed the page. This decreased the quality of the user experience. But now with persistence/rehydrate, we can now keep the information in the local storage of our browser and this lets the user for example, not login to their account every time they close the tab or refresh the page.

We configured and started using Eslint in our project to increase maintainability, readability and overall cleanness of the code. From time to time(ideally before any commit), we run eslint using our custom configuration file which contains various rules. Eslint checks everything in our src folder, raises warnings(even automatically fixes most of them) for the rules specified. After getting rid of every warning,  we commit our changes. By using this tool and executing these management steps, we ensure a maintainable, readable, clean and scalable codebase.

We introduced a new CI/CD workflow which uses Github Actions. See “A detailed explanation of the challenges you met during the deployment, dockerizing, and CI/CD processes.” section for details on this. This also somewhat changed our management of the project as well. As the frontend team, we didn’t really use the master branch before. But now, we do bulk deployments of new features from the web-dev branch to the master branch after we make sure everyone’s code is working as they should in the same branch. This provides us an easy to access last version of our project which comes in handy in the development stage. It also provides the Android and Backend teams a reference while working on the same features. Also the fact that all of this is now automatically achieved saves us a ton of time where we can invest somewhere else.

Besides from these, we can say that our planning and work division is effective since we mostly stick to the plan even and our works didn’t cause any major errors yet when merged. One thing we learned and changed from past mistakes was that we finished making commits much earlier than last time (Not considering dummy data and minor bug fixes). Before the last presentation we decided to change something the last day so that it would look nicer on the presentation and it somehow messed up our deployment. So we couldn’t find the problem in time and had to demonstrate from our local machine, which could have gone even worse. So we learned from that and decided to not make any serious changes after 2-3 days left to important dates.

## Evaluation of the Tools Used by the Android Team:

We used the same tools that are indicated in the first milestone report and we have not started to use new tools since the first milestone.

## Evaluation of the Tools Used by Backend Team:

We used Node JS as our main language. We managed our MongoDB database by using Mongoose. We got used to coding our project with them. We didn’t have any problem with learning and using these technologies.

Some members of our team had trouble running docker on their locals and testing endpoints via postman. This was one of the factors that affected us the most. Sometimes the person who wrote the code could not test the code and had to test it on another computer. Since these problems are completely technical problems, we are likely to experience them in the upcoming periods. We will try to solve the problems related to docker and postman and find alternative ways. This will speed up us considerably.

Since the products we added at first were not suitable for the format of the course, we had to add products several times. Adding products one by one manually, writing descriptions, finding images for products was quite a difficult process. In the previous stages, we assigned this task to only one person. However, as it is a tiring task, we found it appropriate to distribute categories to each member of the team.

Although we leave the use of IDE to personal preference, we all preferred to use VS Code. We didn't talk about it. During development, we felt the benefit of using the common IDE. We showed each other shortcuts while doing pair coding.

In the first milestone, we said that we will use Jenkins for CI / CD on the backend side. When we started the second milestone, we decided to try the Github actions section. Github actions was providing us with a default CI / CD config for AWS EC2 servers. We created a branch called "Backend-staging". We were getting our server live on this branch via pull or push requests. However, we could not fully activate this part because there was a problem with our server. As we lacked experience as a team on the CI / CD side, this place did not go well. We could not focus on this part because our priority was to organize the database and develop some features. We definitely plan to solve this part in the next milestone. Most probably, we will focus on this issue all together. As the backend team, we are very fast and solution-oriented when we are trying to deal with a problem all together.

Since we first started coding the backend side, we have tried to use microservice methodology. With the second milestone, we wrote more code for our vendor, product and user apis. We used nginx to manage this place more effectively. Although we didn't get used to this structure at first, we wrote code comfortably over time.

During the process, we did very well with pair programming. In particular, payment flow was a section that had to be carefully considered in detail. We got each other's views in many parts. We checked the places that we could not communicate quickly and solve them together. We have seen that the places where we hang out most of the time are actually very small inconveniences. For example, they were errors in the type of using variable with the wrong name, etc. When a person works for a long time, it is difficult to see his mistakes. When appropriate, we looked at the problems as a team and overcame this difficulty. This was something we didn't do in the first Milestone. We also communicate better with the other teams for the last part of the Milestone 2. We got better in communication over time. Communication has crucial importance in the development process.

# Assessment of the Presentation

We are happy with our presentation. We demonstrated the features we implemented after the first milestone with realistic scenarios. Our customers could follow the presentation and ask their questions which we answered without a problem. After the second milestone, we can say that we are on the same page with our customers and in the right direction. We will continue following our project plan and finish the implementation of all features until the final milestone.

In the next milestone, that is the last milestone, we should prepare a scenario such that our presentation becomes interactive. In the last two milestones, our scenarios were limited to the features we have implemented, however next time our customers should be able to see and experience all the features. They should feel like a real customer using our application and do whatever they want. Therefore, our opinion is that we should make a more interactive presentation next time. In addition, maybe we can emphasize the user-interface of our application more in the last milestone, since the visuality is also an important thing after the functionality.