# CMPE-451 Milestone 2 Report
# Group 5

**Members**
Kayacan Vesek
Emre Hoşer
Volkan Bulca
M.Zeynep Çayırçimen
Mısra Yavuz
Sertay Akpınar
İsmet Sarı
Yaşar Selçuk Çalışkan
Muhammed Halas
Yusuf Yüksel
Algı Kanar
Ramiz Dündar

# Contents

# 1 Executive Summary

## 1.1 Project Status

This year, new members joined to our group. Before starting implementing the ecommerce platform, we arranged a meeting and explained them what we have done in our project so far on CmpE352 course. In our first group meeting, we determined our development teams: frontend, backend and mobile (Android). Also, although we have finalized our project requirements and project plan during CmpE352 course, we decided to go through and update them. We have given some Customer milestones and we set some internal milestones for each team, and make some updates on project plan accordingly.

Until our first customer milestone, we have implemented the user register and login functionalities in both backend and android. There is also a password change functionality on backend as well. Backend team has also implemented the feature for users to change their personal information. The design of the homepage, search bar and category bar, login and register pages are done by the frontend team. User profile page is also implemented, from which the users can see their personal information and change their password.

Until our second customer milestone, we first complete the missing or inadequate features that were stated in the first customer milestone. Backend team added the social login functionality and also implemented verification mail send functionality. Android team implemented the profile pages, and both frontend and android teams added the pages for personal information change pages. The personal info update functionality has been implemented until the first milestone.

After completing the missing parts of the application, we started working on the tasks that are given in our project plan. Backend team created a product model, and both frontend and mobile teams implemented the product pages that shows the information about the products and some product specific functionalities such as making comment and rating, adding the product to a list and adding the product to the shopping cart. The comment and rating functionality, product lists and adding the product to shopping cart features are implemented by all teams. Lists can be created by customers, new products can be added or removed from the lists. Customers can also add and remove products to the shopping cart for purchase. Only the customers who purchased a product can make comments and give ratings to that product. All ratings and comments are displayed on the product page.

The homepage is also updated with the introduction of proper product models. We have implemented categories and subcategories to the homepage to classify products according to the given category. From those category pages, the products can be sorted and filtered with given features. Also, the search functionality is implemented so products can be searched by given keywords. Search also supports sorting and filtering.

Vendor page is updated in the frontend. Vendors can add products from their profile pages. New products are added to the new arrivals list in the homepage. Customers can send messages to vendors, so chat system is supported. Both functionalities are implemented by the backend team. However, vendors can add products only in frontend, and messaging can be done only in mobile.

Customers can also purchase products that are in their shopping carts. After seeing the total cost, they are directed to the page where they provide an address and a valid credit card. Customers can add new credit cards if they have no cards provided, or choose a credit card that are already listed. The credit cards can also be displayed from the profile pages. Both frontend and android supports purchase functionality. The order history(order status display) and order cancellation functionalities is implemented only in the backend.

## 1.2 Moving Forward

The first job that we will do is implementing the missing or inadequate features of the second customer milestone. This includes order status pages and order cancellation functionality. On frontend side, we will implement order history page where customers can trace the statuses of their current and previous orders. Also, messaging functionality will be added. On mobile side, android team will also implement the order history pages. They will also complete the vendor add product functionality. Those should be added to the project plan, therefore a small update on project plan is needed.

After we are done with the quick changes relating to the second milestone, we will immediately start working on the tasks that are left undone until the final milestone. These tasks include shipment information pages, recommendation and notification systems. The jobs that are planned to be done until the final customer milestone and the task assignments can be found in our project plan on our GitHub page.

## 2 List and Status Deliverables

| **Deliverable** | **Status** |
|---|---|
| Login & Sign up | Done |
| Home Page | Done |
| Profile Page | Done |
| Email Confirmation | Done |
| Product Page | Done |
| Comment Section with Ratings | Done |
| Adding new Comment & Rating | Not Done |
| Search and Sort | Not Done |
| Cart & Wishlist Page | Done |
| Category/Subcategory Page | Partially Done |
| Order Page | Done |
| Payment & Credit Cards Page | Done |
| Add Product Page | Not Done |

Table 1: List and status of the deliverable in Mobile

| **Deliverable** | **Status** |
|---|---|
| Login & Sign up | Done |
| User Profile | Done |
| Database instance for production | Done |
| Database instance for development | Done |
| Online API documentation | Done |
| Email Verification | Done |
| Add Product for Vendor | Done |
| Semantic Search & Sort & Filter | Done |
| Carts & Lists | Done |
| Order & Payment | Done |
| Comment & Rate | Done |
| Order Cancellation | Done |
| Send Message to Vendor | Done |
| Deployment | Done |

Table 2: List and status of the deliverable in Back-end

| **Deliverable** | **Status** |
|---|---|
| Login & Sign up | Done |
| Email Confirmation | Done |
| Home Page Design | Done |
| Profile Page | Done |
| Product Page | Done |
| Comment Section | Done |
| Favorites and lists | Done |
| Cart Page | Done |
| Search and Sort | Done |
| Payment Page | Done |
| Chat | Partially Done |
| Category/Subcategory Page | Done |
| Add Product Page for Vendor | Done |
| Deployment | Done |

Table 3: List and status of the deliverable in Front-end

## 3 Evaluation of the Status of Deliverables

**Login & Sign up & Email Verification:**

Login and Sign up functionalities were already completed in milestone 1. What's new is that now we require customers to validate their email addresses by clicking on the verification link on the email sent by our system, after they create a new account. This specification is a must in real-life social platforms for security purposes, so having this implemented is a plus for bupazar.

**Home Page:**

In Milestone 1, home pages were static and filled with mockup data since we haven't implemented the product components then. Now, home pages of both frontend and mobile are connected to backend, and on front-end products are classified as bestsellers, new arrivals and etc. with responses gotten from backend. Product grids are also dynamic and by clicking on the images, one can visit the corresponding product page.

**Category & Subcategory Pages:**

Category tab have been made functional and users now can view products of different categories or even subcategories. Users can also do filtering or sorting on products viewed in a category/subcategory page. On frontend, this functionality is fully implemented. Mobile team also made progress about it, even if it is not completely done.

**Product Page:**

Product pages are functional on frontend and mobile. Backend team provided all necessary endpoints for different sections of the product page. On frontend, viewing product details(name, vendor name, brand name, description, image, rating), viewing old comments, adding to cart, adding review(rating & commenting) if the product has been already bought by the customer, adding to favorites, adding to an existing list, adding to a new list after creating are implemented. Product pages are implemented using URL parameters so that the links can be shared between users of bupazar. On mobile, viewing product details(name, vendor name, brand name, description, image, rating), viewing old comments, adding to cart, adding to wishlist are completed. Mobile team have not implemented adding new comments or adding to lists yet, these are planned for the next milestone.

**Comments/Reviews:**

Backend team implemented necessary endpoints for getting and posting comments/reviews, and on both mobile and frontend old comments and ratings can be seen. Also, on frontend, a customer who have bought a product can rate and leave a comment on that product. Comments are publicly visible to everyone, so a customer also has the option to post his/her review as anonymous. Averages of rating scores is shown as a read-only property on product pages. On frontend, when a customer posts a review without selecting a rating score, it is considered 0 by default, we plan to change this to give a warning when it is 0 and therefore, only allow ratings between 1 and 5.

**Profile Page:**

Even if the profile page is not fully completed considering the menu items, it is functional and it has some updates compared to milestone 1. Users can update his/her personal information. My lists section is connected to lists page of the customer, on frontend. Also vendor profile has the add product section. Other sections such as tracking orders for customers or vendors have not been implemented, however these are planned for the next milestone.

**Favorites & Lists:**

Favorites list is considered as a part of the user account, and backend team implemented it as the default list for any customer who has an account. Therefore, favorites list alreay exists when a customer wants to add a product to it, and correspondingly the favorites list can not be deleted by the customer. On frontend and mobile, adding to favorites list and viewing products in it are completed. Something to note is that frontend team referred to liked products as Favorites and mobile team as Wishlist on their platform, these are synonyms pointing to the same list.

A customer can also have other lists. Backend implemented the create a new list, add to list, remove from list, delete list, view products in list endpoints. Frontend implemented adding to a list, and creating new list functionalities as a part of the product page when corresponding icons are clicked. Removing from lists, deleting lists, viewing products in lists are implemented as a section of the user profile. Mobile team have not implemented lists other than wishlist yet.

**Shopping Cart, Credit Cards, Payment:**

Shopping cart functionalities are mainly covered by all teams. Products can be add to cart, amounts can be increased and decreased from cart pages. Also payment follows the cart page. Here, customers can select one of their saved credit cards or save a new one. Credit card information is gathered and customer can complete ordering, however payment does not happen in real-life. Also customers have the opportunity to edit their addresses while giving orders. As a technical note, our platform sometimes show buggy behaviors on Safari, we are not sure what is causing those, but it works smoothly on Google Chrome. So we would appreciate if it is tested on Chrome.

**Orders Page:**

Backend team implemented necessary endpoints about canceling orders by customer or vendor, updating statuses of orders, and viewing old orders. Frontend and mobile teams have not implemented the pages for these endpoints but they will be completed within few weeks, according to our project plan.

**Search, Sort, Filter:**

Backend implemented semantic search functionality. On frontend, search page is implemented and results can be filtered based on brand, vendor, price range, rating, discount rate. Products can also be sorted according to prices, best sellers, new arrivals, ratings and comments. Mobile team have not completed search, sort and filter functionalities yet. Since filtering and sorting is a little hard to debug, sometimes we encounter bugs and resolve them on any side that needs fixing.

**Chat/Send Message to Vendor:**

Mobile team has completed and also presented chat functionality on our milestone presentation. Frontend team has also started implementing it, but have not completed it yet. Besides, after milestone presentation, it's been decided ,between the members responsible for Chat functionality, that the relevant endpoints should be updated to give responses with necessary data efficiently. All teams have made some modifications accordingly, after the presentation.

**Add Product Page:**

Add product functionality is implemented on backend and frontend. Mobile team have not implemented this yet. Backend also provided an endpoint for a vendor to get all of his/her products, but corresponding pages have not been implemented on frontend and mobile yet. We plan to enhance vendors' bupazar experiences in following weeks as well, since our platform, as it's developed until Milestone 2, might crush when a vendor wishes to visit different pages on our platform.

**Deployment, CI/CD:**

We have started working with GitHub actions to achieve CI/CD pipeline. Since most of us had no experience using this functionality before, sometimes our deployed versions might reflect the previous commits of our code but we are trying our best to get the most of this automated pipeline. Also, deployment links are updated.

## 4 A Summary of Coding Work Done by Each Team Member

| Member Name | Contributions |
|---|---|
| Kayacan Vesek | - I have implemented messages page which can be accessed from profile page.<br>- I have implemented chatting page, user can chat with vendors.<br>- I add a button to the product page to navigates to chat with vendor of the product<br>- I have implemented navigation when clicking the product in the cart and wishlist page.<br>- I have helped to solve a bug in adding a product to a list.<br>- I have add a class that helps to keep log-in information.<br>- I have reviewed 297, 294, 292 290 283 263.<br>- I created pull request 296, 289. |
| Emre Hoşer | - I have implemented logout button functinality on profile page.<br>- I hava created the proper function and method that passsed user data from login response to user profile page.<br>- I have contributed to the implementation of user page view components.<br>- I have implemented the page for basket, profile and favorites page according to guest user.<br>- I have contributed to the implementation of user page backend integration.<br>- I have changed to functionality of login and register responses according to changes in backend part.<br>- I have modified user warning messages shown to user and make them clear and more user friendly.<br>- I have implemented the email verification page and neccessary method to verificate user email.<br>- I have implemented the forgot my password button and page.<br>- I have implemented the reset password functionality on forgot password page.<br>- I have implemented comment and rating view component and backend connection of comment, rating part on product page. |
| Volkan Bulca | - I have created Comment, Purchase and CreditCard models.<br>- I have created necessary serializers used in the api endpoints.<br>- I have implemented the customer comment and rating endpoints such as add comment, get all comments. (Under products endpoint)<br>- I have implemented customer credit card endpoints such as add, delete, get all credit cards.<br>- I have implemented purchase endpoints.<br>- I have worked on customer orders and implemented endpoints such as make purchase, get customer orders, vendor update status and customer purchased.<br>- I have implemented 3 unit tests: AddCreditCardTest, CommentRatingTest, UpdateStatusTest. |

| Member Name | Contributions |
|---|---|
| Sertay Akpınar | - I have implemented the wish list feature.<br>- I have implemented the design of the user profile page and editing personal info feature.<br>- I have implemented the design of the categories.<br>- I have worked on mobile-categories and mobile-wishlist branches generally.<br>- I have documented the code structure and group process of mobile team.<br>- I have taken the notes of meeting #7.<br>- I have supported the design of payment page xml and resolved the bugs in order process with Kayacan and Yaşar.<br>- I have opened issues #302, #235, #293, #261, #183, #180<br>- I reviewed pull requests #296, #294, #262, #255, #245, #238, 234<br>- I created pull requests #297, #292, #291, #239 |
| Mısra Yavuz | - I implemented the comment and rating sections of product pages and connected adding reviews to backend.<br>- I worked on URL parameters for product pages so that each product page can be accessed with different URLs.<br>- I connected add to cart button to backend and also made some visual modifications on product page.<br>- I implemented My Lists page and a page to view products in any list, with list id given as URL parameter.<br>- I added the create new list, add to list/favorites and remove from favorites functionalities to corresponding buttons on the product page.<br>- I added removing products from lists and deleting lists functionalities to My Lists page and products in list pages.<br>- I reviewed pull requests of search, category pages, cart/payment and etc. |
| İsmet Sarı | - I have implemented register with email verification endpoints,<br>- I have implemented password reset endpoints<br>- I have implemented creating, sending getting and deleting chat and message endpoints.<br>- I have also written GitHub action workflows with sh scripts and made the CI/CD deployment |
| Yaşar Selçuk Çalışkan | - Implemented the grid recycleview to show products in homepage in grid view and made the backend connection to show products.<br>- Implemented end-to-end the whole navigation bar - including xmls, adapters and fragments.<br>- Implemented MyAccount page xml file.<br>- Implemented "Add To Cart" functionality.<br>- Implemented the cart page including the xml and fragment.<br>- Implemented the order placing end-to-end that includes xmls for the order and order completed pages, changing credit cards,<br>- Opened the issues #182, #218, #220, #221, #236, #251, #252, #253.<br>- Created the pull requests #217, #222, #234, #238, #240, #255, #262, #294.<br>- I have reviewed requests for the chat functionality and the product page. |
| Muhammed Halas | - I implemented the api endpoints related to customer shopping carts such as get, edit, clear cart.<br>- I implemented the api endpoints related to customer's favorites list such as getting the favorites list, adding product and removing product.<br>- I implemented the api endpoints related to customer's product lists such as, creating a list, getting the list, deleting a list, adding products to a and removing products from a list. |

| Member Name | Contributions |
|---|---|
| Algı Kanar | - I have revied password change,add product to cart.<br>- I helped the merging process of category sections.<br>- I did product page pull request.<br>I implemented design and initial UI features of the product page.<br>- I did all the functionalities of search.<br>Implemented the connection between home page, search page<br>and product pages.<br>- I did the filtering and sorting parts of the search too.<br>While implementing these I created components such as<br>filterlist and gridlistforsearch.(Search page pull request)<br>-I also did the home page final design.<br>Connected it with backend. Modified simplegridlist for better display<br>- Also partially worked for upcoming messaging feature. |
| Ramiz Dündar | - Deployment of separate frontend AWS instance with AWS Education<br>- Changing Dockerfile in order to reduce RAM consumption in AWS<br>(it was failing)<br>- Email verification on frontend side<br>- For vendors add product page<br>- Cart page that users can add products<br>- Integration of cart page into order system with Zeynep<br>- Frotend continuous integration with necessary shell scripts |
| Yusuf Yüksel | - I have configured back-end repository to store images at AWS S3.<br>- I have implemented custom permissions for customer and vendor users.<br>- I have created Product, Category, Subcategory, Document and Order models.<br>- I have implemented listing all products, getting product details via giving<br>id to path, listing all products which are in same subcategory and listing<br>all products which are in same category api endpoints.<br>- I have implemented listing all products of vendor and listing home page<br>products api endpoints.<br>- I have implemented adding, updating and deleting product for a vendor<br>api endpoints.<br>- I have implemented semantic search with Porter stemming and Datamuse API,<br>filter and sort products api endpoints.<br>- I have worked on payment and order features and implemented cancelling<br>order for customer, cancelling purchase for vendor and listing all purchases<br>for a vendor api endpoints.<br>- I have implemented 6 unit tests: SearchProductTest, SortProductTest,<br>FilterProductTest, ProductDetailTest, UpdateProductTest and CancelOrderTest. |
| M.Zeynep Çayırçimen | - I have implemented the categories and subcategories pages<br>and nativage from category bar<br>- I have completed profile page. Now user can update their profile<br>except for the email.<br>- I have implemented payment page with a visual credit card. User can<br>add a new credit card or use one of the saved credit cards.<br>- I have integrated the cart to payment process with Ramiz.<br>- I changed the search icon colors on the search-sort page<br>- I have connected vendor's add product page to vendor's profile page<br>- I reviewed pull requests #275, #248, #209 #207<br>- I created pull requests #210, #214, #278, #282 |

## 5 Unit Tests

### 5.1 Yusuf Yüksel

- Commit SHA - **a63b23342d8f5762b4a5afd963d36a731075d21a** (This commit includes 3 unit tests.)

  **-Unit Test for Search Products:**
  It can be found in 'test_search_product.py' file. In this unit test, I defined 'Notebook' as query. In the semantic search which is implemented using Porter stemmer and Datamuse API, It basically searches products which contain 'notebook' or 'laptop' keywords in their name or description, when we search 'Notebook' keyword. Thus, I filtered all products which contain notebook' or 'laptop' keywords in their name or description. Then, I compared these products with returning products in response when we post proper json which includes 'query' as key and 'Notebook' as value to 'search_products' function. Also, I compared status code of response with http 200 and check length of product list in response is greater than or equal to 0 or not. I compared status code of response with http 200 because 'search_products' function returns with http status code 200 for a successful search. It passed all the conditions.

  **-Unit Test for Sort Products:**
  It can be found in 'test_sort_product.py' file. In this unit test, I defined list of product ids as [67, 68, 69, 70, 71, 72, 73] in order to sort them. Also, I chose sort option as 'price' and sort order as 'descending'. I filtered all the products which have these product ids and sort them according to price in descending order. Then, I compared these products with returning products in response when we post proper json which includes 'product_ids', 'sort_by' and 'order' keys to 'sort_products' function. Also, I compared status code of response with http 200 to check if it is successful sort or not. It passed all the conditions.

  **-Unit Test for Filter Products:**
  It can be found in 'test_filter_product.py' file. In this unit test, I defined list of product ids as [67, 68, 69, 70, 71, 72, 73] in order to filter them. Also, I chose filter option as 'price_range' and lower limit of price as 500 and upper limit of price as 10000. I filtered all the products which have these product ids. Then, I filtered these products via setting price range as (500, 10000). I compared these products with returning products in response when we post a proper json which includes 'product_ids', 'filter_by', 'lower_limit', 'upper_limit' keys to 'filter_products' function. Also, I compared status code of response with http 200 to check if it is successful filter or not. It passed all the conditions.

- Commit SHA - **f11b595fb5c185165c1377f5642518642dc6a97b** (This commit includes 2 unit tests.)

  **-Unit Test for Product Details:**
  It can be found in 'test_product_detail.py' file. In this unit test, I created category, subcategory and vendor instances to create a product instance. I got attributes of this product instance via id. Then, I compared these product attributes with returning attributes of product in response when we get attributes of product via giving id to path of product_detail function.

Also, I compared status code of response with http 200 to check if it is successful get operation or not. It passed all conditions.

**-Unit Test for Update Product for Vendor:**
It can be found in 'test_update_product.py' file. In this unit test, I created category, subcategory and vendor instances to create a product instance. Also, I defined price as 200 and stock as 7 to update price and stock of this product. I got token of vendor via AuthUserSerializer and put it to header of request object. I post a proper json which includes 'product_id', 'price' and 'stock' as keys to 'update_product' function. Then, I compared status code of response with http 200 to check if it is successful update product operation or not. Also, I compared 'success' message of response with 'Successfully updated product' because 'update_product' function returns this message when vendor successfully updates product. It passed all conditions.

- Commit SHA - **4e5aa65e59d4f38cca369950bd7b373cb5db634b** (This commit includes 1 unit test.)

**-Unit Test for Vendor Cancel Purchase:**
It can be found in 'test_cancel_order.py' file. In this unit test, I created category, subcategory and vendor instances to create a product instance. I created customer and order instances to use them in purchase instance. I got token of vendor via AuthUserSerializer and put it to header of request object. I post proper json which includes 'purchase_id' as key to 'vendor_cancel_purchase'. Then, I compared status code of response with http 200 to check if it is successful cancel purchase for vendor or not. Also, I compared 'success' message of response with 'Purchase is successfully canceled.' because 'vendor_cancel_purchase' function returns this message when vendor successfully cancels a purchase. It passed all conditions.

## 5.2 İsmet Sarı

- SHA code - c967f36c6b609be9717bead287e6476d5f2be2a4

In this part I test the endpoint called "register". When we send the request the endpoint require us to enter user properties. The request is by self.client.post and get the reponse then the response status needs to be http 201 and I check it by assertEqual and the response message must be 'user_info is created'. I check it again by assertEqual again. Then a TempUser object is expected to be created and I try to get it by filter. If it is created then the if statement is going to be executed and is_presented variable is set to True and self.assertTrue will be passed.

- SHA code - a4a7ca986b4b0cc2a2e16a4a317cfadff1df3acb

In this part I test the register_activate endpoint. I establish the body part and send a rest request. When I get the response I check the status code of it then it needs to be 201 to pass the test. After that the TempUser object with the email that we sent in body part is expected to be deleted. If it is so then is_present is set to False and pass the test. At the last part I check if the User object is created and if it is then is_present_User_List is set to True and pass the last test.

- SHA code - 23193e3d0d206ca357305307b231d316ad90eb8e

In this part I test the google login endpoint. I take the G_CLIENT_ID from our settings file and create a body dictionary. I send the request and get the response and check if the status code is 201. Then I get the user information from verify_oauth2_token and check. At the last part I check the response if there is a SocialDocs object with the email that we got from verify_oauth2_token. If it is present then the test is passed

- SHA code - 985a08a0eb3f1ccf9af6ae3a64a86207ee672e35

  In this part I test the create chat endpoint. First I assume that there is a user with email "mrvyldm2@mailpoof.com" and got the authorization response properties. After all create a body for request and set the client authorization token. First I check if there is a chat and I expect there is no chat with the fields in body part. If it is then first test pass. Then I send the request and check if the status code of the response is http 201 or not. After that test I check this time our chat is created or not. If it is created the is_found is true and last two assert test are passed.

- SHA code - bf466ab3d138cf771ea7444d9638d783e8df4bef

  In this test I expect to fail such that I send a body with a chat_id field that does not exist. I assume that there is a user with "mrvyldm2@mailpoof.com" email and get the authorization properties and add them into client and send the request and get the response message. The response message should have the error field. If it is so the assertEqual is going to be passed.

## 5.3 Volkan Bulca

- SHA code - bf3be2feb7f00806e9f41d199479f21f9e3d5aff

  In this commit, I wrote a unit test for **adding comment and rating to a product**. The test is done for add_comment endpoint. First, I set up the required features for the unit test. Here, I added a test product and a test customer, and make that customer purchase that product. This should be done because only the customers who have purchased a product can give comment and rating. After everything is set up, I make the login authorization and receive the generated token. I give the required fields in the request body. These are product_id, comment_text, is_anonymous and rating_score fields. After sending the request, there should be a response message received. If the response status is HTTP_201, then the comment and rating score should be given to the product and in response message, "Comment added, Rating is given" should be displayed.

- SHA code - 53c66e718fa4953ef2a73bee9c78b343d3320ec2

  In this commit, I wrote a unit test for **adding a credit card for a customer**. The test is done for credit-cards/opts-add endpoint. Only the customers can add credit cards to their account, so in set up function of the unit test, I created a test customer. Then I make the login with that customer and received the authorization token. After receiving the token, I filled the request body with the necessary fields. Those fields are name, card_owner, card_number, expiration_date and cvc_security_number. If everything goes fine, the unit test should give a response message saying "Credit card is successfully added" with a HTTP_201 status.

- SHA code - 844f579348f7abb9fb766cb47fd67c732299f6a7

  In this commit, I wrote a unit test for **updating the status of an order by vendor**. This test is done for vendor_update_status endpoint. At the beginning, I need to create a vendor, a customer and a product because the status update is done by a vendor whose product is purchased by a customer. After creation of those, I make that customer purchase the product. This way, that vendor would receive that product as a valid order. After setting everything up, I make login with that vendor and get the authorization token. Then, I prepared the request body and give order_id and the status as the fields. In this test, I choose to give a "Preparing" status, but the vendor can also choose "Ship" and "Delivered" for that product. After making the request, if nothing goes wrong, the response should have a message "Order status is successfully updated." and a status code 200.

- Other than those unit tests, I also tested the API endpoints using postman during the development constantly, giving a valid request body as json, and receive the response.

## 6 Deployment, Dockerizing, and CI/CD processes

We have accomplished to deploy both the front-end and the back-end platforms on the Amazon EC2 instances without CI/CD. In this milestone we tried to deploy our app automatically. The workflow files had been updated but in the final version we decided that the workflow is trigger when there is a change in backend directories for backend deployment or frontend directories for frontend deployment. We used workflow_dispatch field to trigger the workflow manually. After all we set the jobs step by step.

Before setting the first step we defined the operation system of our machine that runs the workflow steps as ubuntu-latest then used a actions/checkout@v2 to clone our repository. In the next step we login our Dockerhub accountç The username and password of our account are kept in the secrets part of our repository. Then we build our docker and push it to Dockerhub with tag django_app for backend react_app for frontend.

Until this step everything went well but we had some difficulties to connect our ec2 instance. However we used the codes of the person with appleboy nickname and succeeded to connect and run the bash commands. These commands were written to pull docker image and run it after stopping and removing the old version. We run the "docker pull bupazar/django_app" and said that it is okay but it was not since we realized that our backend was not updated so we forced to pull and run by appending latest tag. After pulling and running the image we removed the stopped unnecessary containers with "docker container prune -f". The old image's tag was set to none and we took its image id with some basic linux configurations and remove the images by docker rmi. After this step we run the updated image and got rid of the old version image and stopped container. By doing this the free space of our ec2 instance was not decreased.

# 7 API Documentation

Swagger url: bupazar API

## 7.1 Authentication

| Functionality | URL | Method | Parameters | Responses |
|---|---|---|---|---|
| Sign up | /api/auth/register | POST | email*, username*, first_name*, last_name*, password*, address is_customer, is_vendor | success |
| Sign Up Activate | /api/auth/register_activate | POST | email*, number* | email, username, first_name, last_name, is_customer, is_vendor, is_active, is staff, address, auth_token |
| Login | /api/auth/login | POST | email*, password* | id, email, username, first_name, last_name, is_customer, is_vendor, is_active, is_staff, address, auth_token |
| Logout | /api/auth/logout | POST | auth_token (inside headers) | success |
| Password Change | /api/auth/password_change | POST | auth_token (inside headers) current_password*, new_password* | success |
| User Information | /api/auth/user_info | POST | auth_token (inside headers) | id, email, username, first_name, last_name, is_customer, is_vendor, is_active, is_staff, address |
| User Profile Update | /api/auth/profile_update | POST | auth_token (inside headers) email, username, first_name, last_name, address | success |
| Google Login | /api/auth/google_login | POST | auth_token (that is taken from Google API) | id, email, username, first_name, last_name, is_customer, is_vendor, is_active, is_staff, address, auth_token |
| Google Login | /api/auth/google_login | POST | auth_token (that is taken from Facebook API) | id, email, username, first_name, last_name, is_customer, is_vendor, is_active, is_staff, address, auth_token |

Table 4: Authentication Endpoints

## 7.2 Users

| Functionality | URL | Method | Parameters | Responses |
|---|---|---|---|---|
| List all users | /api/users | GET | - | id, email, username, first_name, last_name, is_vendor, is_customer, is_active, is_staff, address |
| Retrieve user details by user id | /api/users/{id} | GET | - | id, email, username, first_name, last_name, is_vendor, is_customer, is_active, is_staff, address |

Table 5: User Endpoints

## 7.3 Products

| Functionality | URL | Method | Parameters | Responses |
|---|---|---|---|---|
| Retrieve product details by id | api//products/{id}/ | GET | id* (path) | id, name, price, stock, description, date_added, number_of_sales, image_url, category, subcategory, vendor, total_rating_score, rating_count, brand, discount, rating |
| List all products | api/products/ | GET | - | [Products] |
| List all products which are in same subcategory | api/products/subcategory/ | POST | subcategory_name* | [Products] |
| List all products which are in same category | api/products/category/ | POST | category_name* | [Products] |
| List all products of vendor which sends request | api/products/vendor-products/ | GET | - | [Products] |
| List home page products | api/products/homepage/ | POST | number_of_products* | 'newest_arrivals' : [Products], 'best_sellers' : [Products], 'trends' : [Products] |
| Add product for vendor | api/products/opts/add/ | POST | name*, price*, stock*, description*, image_file*, subcategory_name*, brand*, discount* | success |
| Delete product for vendor | api/products/opts/delete/ | POST | product_id* | success |
| Update product for vendor | api/products/opts/update_product/ | POST | product_id*, name, price, stock, description, discount | success |
| Add comments and give rating | api/products/opts/add_comment/ | POST | product_id*, comment_text*, is_anonymous*, rating_score* | success |
| Retrieve all comments of a given product | api/products/opts/get_all_comments/ | POST | product_id* | [Comments] |
| Search products | api/products/search/ | POST | query* | [Products] |
| Sort products | api/products/sort/ | POST | [product_ids*], sort_by*, order* | [Products] |
| Filter products | api/products/filter/ | POST | [product_ids*], [filter_data*] | [Products] |

Table 6: Product Endpoints

## 7.4 Comments

| Functionality | URL | Method | Parameters | Responses |
|---|---|---|---|---|
| List all comments | /api/comments | GET | - | id, customer, product* comment_text*, rating_score*, is_anonymous* |
| Retrieve comment details by comment id | /api/comments/{id} | GET | - | id, customer, product* comment_text*, rating_score*, is_anonymous* |

Table 7: Comment Endpoints

## 7.5 Credit Cards

| Functionality | URL | Method | Parameters | Responses |
|---|---|---|---|---|
| List all credit cards | /api/credit-cards | GET | - | id, name, customer*, card_owner* card_number*, expiration_date*, cvc_security_number* |
| Retrieve credit card details by comment id | /api/credit-cards/{id} | GET | - | id, name, customer*, card_owner* card_number*, expiration_date*, cvc_security_number* |
| Add new credit card | /api/credit-cards/opts/add | POST | name, card_owner* card_number* expiration_date*, cvc_security_number* | success |
| Delete a credit card | /api/credit-cards/opts/delete | POST | creditcard_id* | success |
| Retreive all credit cards of a user | /api/credit-cards/opts/get_all_credit_cards | POST | - | [Credit Cards] |

Table 8: Credit Card Endpoints

## 7.6 Purchases

| Functionality | URL | Method | Parameters | Responses |
|---|---|---|---|---|
| List all purchases | /api/purchases | GET | - | id, customer*, vendor*, product amount*, unit_price*, order* status* |
| Retrieve purchase details by purchase id | /api/purchases/{id} | GET | - | id, customer*, vendor*, product amount*, unit_price*, order* status* |

Table 9: Purchase Endpoints

## 7.7 Orders

| Functionality | URL | Method | Parameters | Responses |
|---|---|---|---|---|
| Make purchase | /api/orders/make_purchase | POST | - | success |
| Get a customer's orders | /api/orders/customer-orders | GET | - | order_id*, [Purchases] |
| Look whether a product is purchased by customer | /api/orders/customer-purchased | POST | product_id* | true/false |
| Vendor update status of an order | /api/orders/update-status | POST | order_id*, status* | success |
| Retrieve all purchases which contain requesting vendor' products | api/orders/vendor-orders/ | GET | - | [Purchases] |
| Cancel purchase for vendor which sends request | api/orders/vendor-cancel/ | POST | purchase_id | success |
| Cancel order for customer which sends request | api/orders/customer-cancel/ | POST | order_id* | success |

Table 10: Order Endpoints

## 7.8 Chats

| **Functionality** | **URL** | **Method** | **Parameters** | **Responses** |
|---|---|---|---|---|
| Create Chat | /api/chats/create_chat | POST | vendor_username*, product_id* | success, [Chat] |
| Get all chats | /api/chats/get_all_chats | GET | - | success, [Chat,[Message]] |
| Get chat history | /api/chats/get_chat_history/ | POST | chat_id* | success, [Message] |
| Get last message | /api/chats/get_last_message/ | POST | chat_id* | success, Message |
| Send message | /api/chats/send_message/ | POST | chat_id* , content* | success, [Message] |
| Get the number of unread messages | /api/chats/get_unread_messages_number/ | GET | - | number |
| Delete message | /api/chats/delete_message/ | DELETE | message_id | success |
| Get the number of unread messages | /api/chats/delete_chat/ | DELETE | chat_id , message_id | success |

Table 11: Chat Endpoints

## 7.9 Cart

| **Functionality** | **URL** | **Method** | **Parameters** | **Responses** |
|---|---|---|---|---|
| Get Cart | /api/cart/get | GET | None | Cart |
| Edit cart | /api/cart/edit | POST | product_id, count | CartResponse |
| Clear Cart | /api/cart/clear | DELETE | None | CartResponse |

Table 12: Cart Endpoints

## 7.10 Favorites

| **Functionality** | **URL** | **Method** | **Parameters** | **Responses** |
|---|---|---|---|---|
| Get Favorites | /api/favorites/get | GET | None | FavoriteList |
| Add product to favorites | /api/favorites/add | POST | product_id | FavoritesResponse |
| Remove product from favorites | /api/favorites/add | POST | product_id | FavoritesResponse |

Table 13: Favorites Endpoints

## 7.11 Product Lists

| **Functionality** | **URL** | **Method** | **Parameters** | **Responses** |
|---|---|---|---|---|
| Get my product lists | /api/product-lists/opts/my | GET | None | [ProductList] |
| Create A product list | /api/product-lists/opts/add | POST | name | ProductListResponse |
| Delete one of my lists | /api/product-lists/opts/delete | POST | list_id | ProductListResponse |
| Add a product to one of my lists | /api/product-lists/opts/add_product | POST | list_id , product_id | ProductListResponse |
| Remove a product from one of my lists | /api/product-lists/opts/remove_product | POST | list_id , product_id | ProductListResponse |

Table 14: Product Lists Endpoints

## 8 Project Plan

| | Name | Duration | Start | Finish | Resource Name | Pre |
|---|---|---|---|---|---|---|
| - | Backend | 14 days | 11/03/20 5:00 | 11/23/20 5:00 | - | - |
| 2 | Initialize backend server | 4 days | 03.11.2020 17:00 | 09.11.2020 17:00 | Yusuf Yuksel | - |
| 3 | Creating MongoDB | 3 days | 09.11.2020 17:00 | 12.11.2020 17:00 | Yusuf Yuksel | 2,3 |
| 4 | Login/Sign up | 4 days | 13.11.2020 08:00 | 18.11.2020 17:00 | Ismet Sari;Volkan Bulca;Yusuf Yuksel | 2,3 |
| 5 | Backend Milestone 1 | 0 days | 18.11.2020 17:00 | 18.11.2020 17:00 | Ismet Sari;Muhammed Halas;Volkan Bulca;Yusuf Yuksel | 4 |
| 6 | Edit profile | 3 days | 19.11.2020 08:00 | 23.11.2020 17:00 | Volkan Bulca;Yusuf Yuksel | 5 |
| 7 | Dockerizing | 7 days | 13.11.2020 08:00 | 23.11.2020 17:00 | Ismet Sari;Muhammed Halas;Volkan Bulca | 2,3 |
| - | Frontend | 14 days | 03.11.2020 17:00 | 23.11.2020 17:00 | - | - |
| 9 | Initializing project | 4 days | 03.11.2020 08:00 | 09.11.2020 17:00 | Ramiz Dündar | - |
| 10 | Design Template | 2 days | 03.11.2020 08:00 | 05.11.2020 17:00 | Müslüme Zeynep Çayırçimen;Algı Kanar;Mısra Yavuz | - |
| 11 | Login/Sign up pages | 10 days | 10.11.2020 08:00 | 23.11.2020 17:00 | Ramiz Dündar | 9;10 |
| 12 | Home page | 2 days | 10.11.2020 08:00 | 11.11.2020 17:00 | Müslüme Zeynep Çayırçimen;Algı Kanar;Mısra Yavuz | 9;10 |

|  | Name | Duration | Start | Finish | Resource Name | Pre |
|---|---|---|---|---|---|---|
| 13 | Frontend Milestone 1 | 0 days | 12.11.2020 08:00 | 12.11.2020 17:00 | Müslüme Zeynep Çayırçimen;Algı Kanar;Mısra Yavuz;Ramiz Dündar | 11;12 |
| 14 | Profile page | 3 days | 13.11.2020 08:00 | 15.11.2020 17:00 | Müslüme Zeynep Çayırçimen;Algı Kanar;Mısra Yavuz;Ramiz Dündar | 11 |
| 15 | Search bar | 5 days | 16.11.2020 08:00 | 20.11.2020 17:00 | Müslüme Zeynep Çayırçimen;Algı Kanar;Mısra Yavuz;Ramiz Dündar | 11 |
| 16 | Menu bar(categories) | 4 days | 20.11.2020 08:00 | 23.11.2020 17:00 | Müslüme Zeynep Çayırçimen | 12 |
| - | Android | 14 days | 03.11.2020 17:00 | 23.11.2020 17:00 | - | - |
| 18 | Initializing project | 5 days | 03.11.2020 17:00 | 10.11.2020 17:00 | Emre Hoser;Kayacan Vesek | - |
| 19 | Design template | 2 days | 10.11.2020 17:00 | 12.11.2020 17:00 | Sertay Akpinar;Yasar Selcuk Caliskan | - |
| 20 | Login page | 1 days | 13.11.2020 17:00 | 16.11.2020 17:00 | Emre Hoser;Sertay Akpinar | 18;19 |
| 21 | Sign-up page | 1 days | 13.11.2020 17:00 | 16.11.2020 17:00 | Kayacan Vesek;Yasar Selcuk Caliskan | 18;19 |
| 22 | — Internal Mobile Team Milestone — | 0 days | 16.11.2020 17:00 | 16.11.2020 17:00 | Emre Hoser;Kayacan Vesek;Sertay Akpinar;Yasar Selcuk Caliskan | 20;21 |

|  | Name | Duration | Start | Finish | Resource Name | Pre |
|---|---|---|---|---|---|---|
| 23 | Home page | 4 days | 17.11.2020 08:00 | 20.11.2020 17:00 | Emre Hoser;Kayacan Vesek | 18;19 |
| 24 | Profile page | 4 days | 17.11.2020 08:00 | 20.11.2020 17:00 | Yasar Selcuk Caliskan;Sertay Akpinar | 18;19 |
| 25 | Search bar | 2 days | 20.11.2020 08:00 | 23.11.2020 17:00 | Emre Hoser | - |
| 26 | Menu bar(categories) | 0.5 days | 20.11.2020 08:00 | 23.11.2020 17:00 | Kayacan Vesek;Emre Hoser | - |
| 27 | — CUSTOMER MILESTONE 1 — | 0 days | 24.11.2020 17:00 | 24.11.2020 17:00 | Emre Hoser;Kayacan Vesek;Sertay Akpinar;Yasar Selcuk Caliskan | 1;8;17 |
| 28 | Backend | 23.3 days | 25.11.2020 13:00 | 28.12.2020 17:00 | - | - |
| 29 | Email verification | 4,5 days | 25.11.2020 13:00 | 01.12.2020 17:00 | Ismet Sarı | 4 |
| 30 | Google login | 4,5 days | 25.11.2020 13:00 | 01.12.2020 17:00 | Ismet Sarı | 4 |
| 31 | Add product for vendor | 4,5 days | 25.11.2020 13:00 | 01.12.2020 17:00 | Yusuf Yuksel | 4 |
| 32 | List product | 7 days | 02.12.2020 08:00 | 10.12.2020 17:00 | Muhammed Halas | 31 |
| 33 | Search product | 6 days | 02.12.2020 17:00 | 10.12.2020 17:00 | Yusuf Yuksel | 31 |
| 34 | Comment and Rating product | 7 days | 02.12.2020 08:00 | 10.12.2020 17:00 | Volkan Bulca | 31 |
| 35 | Backend Milestone 2 | 0 days | 10.12.2020 17:00 | 10.12.2020 17:00 | Ismet Sari,Muhammed Halas,Volkan Bulca,Yusuf Yuksel | 32, 33, 34 |
| 36 | Add product to cart | 3 days | 11.12.2020 08:00 | 15.12.2020 17:00 | Muhammed Halas | 35 |

| | Name | Duration | Start | Finish | Resource Name | Pre |
|---|---|---|---|---|---|---|
| 37 | Order product | 4 days | 16.12.2020 08:00 | 21.12.2020 17:00 | Volkan Bulca, Yusuf Yuksel | 36 |
| 38 | Payment and Order Status | 3 days | 22.12.2020 08:00 | 24.12.2020 17:00 | Volkan Bulca, Yusuf Yuksel | 37 |
| 39 | Cancel order | 2 days | 25.12.2020 08:00 | 28.12.2020 17:00 | Volkan Bulca, Yusuf Yuksel | 38 |
| 40 | Send message to vendor | 5 days | 22.12.2020 08:00 | 28.12.2020 17:00 | Ismet Sarı | 4 |
| - | Frontend | 24 days | 25.11.2020 17:00 | 28.12.2020 17:00 | - | - |
| 42 | Email verification | 5 days | 25.11.2020 08:00 | 01.12.2020 17:00 | Ramiz Dündar | 11 |
| 43 | User settings | 4 days | 25.11.2020 17:00 | 30.11.2020 17:00 | Müslüme Zeynep Çayırçimen;Algı Kanar;Mısra Yavuz | 11 |
| 44 | Shopping cart / List | 5 days | 25.11.2020 17:00 | 01.12.2020 17:00 | Ramiz Dündar;Mısra Yavuz | - |
| 45 | Add Product (Vendor) | 4 days | 25.11.2020 08:00 | 30.11.2020 17:00 | Ramiz Dündar | - |
| 46 | Frontend Milestone 2 | 0 days | 02.12.2020 08:00 | 02.12.2020 17:00 | Müslüme Zeynep Çayırçimen;Algı Kanar;Mısra Yavuz;Ramiz Dündar | 27;43;44;46 |
| 47 | Product page | 7 days | 02.12.2020 08:00 | 10.12.2020 17:00 | Algı Kanar;Mısra Yavuz | - |
| 48 | Comment product page | 4 days | 11.12.2020 08:00 | 16.12.2020 17:00 | Mısra Yavuz | 47 |
| 49 | Search/Sort/Filter | 4 days | 17.12.2020 08:00 | 22.12.2020 17:00 | Algı Kanar | - |
| 50 | Payment/Order page | 5 days | 22.12.2020 08:00 | 28.12.2020 17:00 | Müslüme Zeynep Çayırçimen | 44 |

|  | Name | Duration | Start | Finish | Resource Name | Pre |
|---|---|---|---|---|---|---|
| - | Android | 14 days | 03.11.2020 17:00 | 23.11.2020 17:00 | - | - |
| 51 | User settings | 5 days | 25.11.2020 08:00 | 01.12.2020 17:00 | Sertay Akpınar,Emre Hoser | - |
| 52 | Shopping cart / List | 5 days | 25.11.2020 08:00 | 01.12.2020 17:00 | Yasar Selcuk Caliskan | - |
| 53 | Product page | 7 days | 02.12.2020 08:00 | 10.12.2020 17:00 | Emre Hoşer,Kayacan Vesek;Yasar Selcuk Caliskan | - |
| 54 | Comment component | 4 days | 11.12.2020 08:00 | 16.12.2020 17:00 | Emre Hoşer | 53 |
| 55 | Chat/Message page | 4 days | 17.12.2020 08:00 | 22.12.2020 17:00 | Kayacan Vesek | - |
| 56 | Payment/Order page | 5 days | 22.12.2020 08:00 | 28.12.2020 17:00 | Yasar Selcuk Caliskan | 52 |
| 57 | CUSTOMER MILE-STONE 2 | 0 days | 29.12.2020 17:00 | 29.12.2020 17:00 | Emre Hoser;Kayacan Vesek;Sertay Akpinar;Yasar Selcuk Caliskan | 28, 41, 50 |
| - | Backend | 14 days | 30.12.2020 08:00 | 18.01.2021 17:00 | - | - |
| 59 | Recommend-ation | 9 days | 30.12.2020 08:00 | 11.01.2021 17:00 | Ismet Sari,Muhammed Halas | 58 |
| 60 | Notification mechanism | 9 days | 30.12.2020 08:00 | 11.01.2021 17:00 | Volkan Bulca,Yusuf Yuksel | 58 |
| 61 | Shipment -Information | 5 days | 12.01.2021 08:00 | 18.01.2021 17:00 | Muhammed Halas,Volkan Bulca | 38 |
| - | Frontend | 15 days | 30.12.2020 08:00 | 19.01.2021 17:00 | - | - |
| 63 | Chat and Message | 4 days | 30.12.2020 08:00 | 04.01.2021 17:00 | Algı Kanar | - |

|  | Name | Duration | Start | Finish | Resource Name | Pre |
|---|---|---|---|---|---|---|
| 64 | Recommendation Page | 9 days | 07.01.2020 08:00 | 19.01.2021 17:00 | Müslüme Zeynep Çayırçimen;Mısra Yavuz | - |
| 65 | Notification Mechanism | 9 days | 07.01.2021 08:00 | 19.01.2021 17:00 | Algı Kanar;Ramiz Dündar | - |
| 66 | Shipment Page/Order Progress | 5 days | 12.01.2021 08:00 | 18.01.2021 17:00 | Mısra Yavuz;Ramiz Dündar | - |
| - | Android | 14 days | 30.12.2020 08:00 | 18.01.2021 17:00 | - | - |
| 69 | Recommendation Page | 9 days | 30.12.2020 08:00 | 10.01.2021 17:00 | Emre Hoser;Kayacan Vesek | 57 |
| 70 | Notification Mechanism | 9 days | 30.12.2020 08:00 | 10.01.2021 17:00 | Sertay Akpinar;Yasar Selcuk Caliskan | 57 |
| 71 | Vendor Login, user settings | 2 days | 10.12.2020 08:00 | 12.01.2021 17:00 | Emre Hoşer | - |
| 72 | Vendor product adding | 2 days | 10.12.2020 08:00 | 12.01.2021 17:00 | Kayacan Vesek | - |
| 73 | Sub- category pages | 2 days | 10.12.2020 08:00 | 12.01.2021 17:00 | Yaşar Selçuk Çalışkan | 53 |
| 74 | Search- Sort-Filter product | 5 days | 12.12.2020 08:00 | 18.01.2021 17:00 | Sertay Akpınar,Emre Hoşer | 53 |
| 75 | Shipment Page | 5 days | 12.01.2021 08:00 | 18.01.2021 17:00 | Kayacan Vesek;Yasar Selcuk Caliskan | 56 |
| 76 | FINAL CUSTOMER MILESTONE | 0 days | 19.01.2021 17:00 | 19.01.2021 17:00 | Emre Hoser;Kayacan Vesek;Sertay Akpinar;Yasar Selcuk Caliskan | 66;62 |

# 9 User Scenarios

## 9.1 The First Scenario: Frontend Scenario (Vendor) - Didem Soydan

### 9.1.1 Demographics

- Founder of didem electronic commonly known as Delectronic
- Currently the CEO of the company Delectronic
- 33 years old

### 9.1.2 Goals

- This is Didem Soydan's first time using Bupazar. She is looking for popular and practical e commerce sites to increase her growing company's sales.

### 9.1.3 Scenario

Didem Soydan logins to the site with her company mail as a vendor. There are new Toshiba laptops which she wants to sell using Bupazar site. Therefore, from home page she directly goes to her profile page to add product. In add product page, she fills the necessary information of the product as name,brand,price,stock,category,subcategory and discount etc. Then, she presses submit.After that she directly goes to home page and sees the new product she entered in new-arrivals.She becomes happy because from the front page everybody will be able to see her product.This will definitely garner attention for the product.

## 9.2 The Second Scenario: Frontend Scenario (User) - Furkan Kale

### 9.2.1 Demographics

- Boğaziçi University Graduate Student
- Currently pursuing MBA
- 24 years old

### 9.2.2 Goals

- Previously Furkan gave his feedback about the bupazar. However this milestone Furkan visits bupazar in order to buy a new notebook for himself and buy a new year present for his nephew.

### 9.2.3 Scenario

Furkan Kale, who is pursuing MBA in Boğaziçi University, told by his close friend Ramiz that he is developing a new e-commerce website with his friends. Previously he talked about his opinions about bupazar. However this time he visit bupazar in order to purchase a new notebook for himself. After adding this notebook to the cart, he choses a new year gift for his nephew. He adds a new shoe to the gifts list and another shoe to the favourites list. After this he realizes that the shoe added to the favourites list is too expensive so he removes that shoe. Lastly, he adds the shoe in gifts list to the cart and buys products in the cart.

## 9.3 The Third Scenario: Mobile Scenario - Berkecan Düven

### 9.3.1 Demographics

- Ozyegin University Senior Undergraduate Student
- Currently pursuing BSc in Business Administration and Management
- 23 years old

### 9.3.2 Goals

- He wants to buy gifts for his mom and his girlfriends as the new year's eve is approaching.

### 9.3.3 Scenario

- Berkecan, who is pursuing a Bachelor's degree from Ozyegin University and a public figure due to his high number of followers on social media accounts would like to buy gifts for his mom and his girlfriend, and would like to share this with his followers since bupazar is his favorite e-commerce app.
- After logging in to the app, he scrolls the products and adds the ones he likes to his wishlist. He pays attention to the user reviews and comments as well when he likes the looks of a product. After choosing a gift, he messages the vendor if they can wrap the product in gift packaging. He receives a response message very quickly, and after that he adds the product to the cart and moves on to the order page to buy them. He changes his credit card since he has no limit left on his Garanti card, so he switches to his Akbank card. He then approves to the terms & conditions and GDPR, and places his order.
- After completing the purchase, he communicates to his followers that buying from bupazar is super easy & fast, and he restates that bupazar is his favorite e-commerce app.

# 10 Code Structure and Group Process

## 10.1 Backend - app/backend

### 10.1.1 Folder Structure

All related code is in the app/backend directory. We created two subfolders inside that directory. One is api which is django app and other one is bupazar which is django project. We created migrations, models, serializers, templates, test, utils and views subfolders inside api to keep project structure simple as possible. Thus, we put new file to related folder easily. In order to install dependencies easily, we put all of the dependencies to requirements.txt

### 10.1.2 Branch Structure

We created a common branch is called backend for development. We keep backend branch up to date according to master branch. We don't push master directly. We create pull request when there are remarkable commits. Reviewers pull repository and test it locally. When they get expected results, they merge it to master. If bugs arise, we assign bug fixing to related persons.

### 10.1.3 Workflow

We hold meetings in discord to divide tasks. We create related issues about tasks. Later we communicate over discord and whatsapp. We generally share our screens over discord to communicate easily when we need help. If there is remarkable progress, we create pull request and reviewers review it. We iterate same process until internal milestone dates. When we have internal milestone, we generally talk about what we have done so far and what are the next steps. After customer milestones, we consider customer feedbacks and integrate them via same process.

## 10.2 Frontend - app/frontend

### 10.2.1 Folder Structure

All related code is in the app/frontend directory. We created subfolders inside src folder for tasks and implemented related components inside them, such as login folder. Also there is common folder for code/components used frequently.

### 10.2.2 Branch Structure

All tasks implemented in related feature based branches, after which, pull request is created. Then pull request is reviewed and merged by reviewer. All work is done this way and there is no common branch or direct push to master. Except deployment commits which are pushed with deploy/ prefix. Since merged branches may be deleted by reviewer, there may be less active/merged branches than we as a team actually used. Note that after milestone 1 all frontend related branches start with frontend- prefix.Here is the the 2 example branches from the repository:

- frontend-email-verification: For adding email verification system to signup process.
- frontend-add-product: Used for implementing add product page for vendors.

#### 10.2.3 Workflow

We hold meetings in order to confirm divison of tasks, after which related branches and issues are created. Later we inform each other via discord and whatsapp as we progress through tasks. When it's finished, the pull request is created and someone other than assignees reviews the pull request. Then we iterate the same process over the new tasks until internal/customer milestones where we add additional changes over the project according to the in-team/customer feedback.

### 10.3 Mobile - app/mobile/BuPazar

#### 10.3.1 Folder Structure

All related code can be found in the app/mobile/BuPazar directory. While the build files lies within this directory, we have implemented our components in the app/mobile/BuPazar/app/src/main/-java/com/example/bupazar directory. Our models are in bupazar/model directory, restApiService can be found at bupazar/service. In the bupazar/page directory we implemented the fragments and activities. ".xml" files are in app/mobile/BuPazar/app/src/main/res/layout directory and drawables such as icons we used can be found at app/mobile/BuPazar/app/src/main/res/drawable.

#### 10.3.2 Branch Structure

After distributing tasks among the team members that we have pre-defined before starting to implement the project, we have implemented our tasks in the branches that is created with the related tasks in it such as "mobile-addtocart". Since user settings feature is implemented in mobile-categories branch, no other branch is created for this feature. After implementing the features, we have created the corresponding pull requests to merge these branches into the "mobile" branch which we have used as a development branch for the milestone-2. It is then merged into the master branch.

#### 10.3.3 Workflow

Before starting to implement the project, we arranged a meeting and we determined the tasks that we are planning to present for the second milestone we allocated the tasks each other. We also took the project plan into consideration while determining the tasks. Since we have limited time and many features to implement, we worked hard especially for the last 10 days. During the development process, we always have kept ourselves in the loop about where we are and what are the possible challenges that we see ahead. We held regular meetings to stay up-to-date and always helped each other. Whenever a feature is completed, a corresponding pull request is created and team members are assigned as reviewers to review and merge the pull request. We worked collaboratively on this milestone, and each person implemented 2-3 features at least. As a result, we successfully reached our goal that we determined in the project plan. Finally, we presented the features we implemented very well in the customer presentation.

# 11 Evaluation of Tools and Managing the Project

## 11.1 Managing the Project

We decided the backend, frontend and mobile teams before the classes started by arranging a meeting. We tried to ensure that the workload was balanced by giving equal numbers of people to each group. Although we divided the group into 3 subgroups, we decided to use discord to communicate. We created 3 separate channels for frontend, backend and mobile to keep the groups in touch.During the coding, instead of pushing the master, we created branches and continued from these branches. We opened a pull request for the completed branches, after they were approved, we deleted those branches. Each team used the predetermined prefix in order to avoid confusion while commiting and opening issues.

## 11.2 Frontend Part

### 11.2.1 Frontend Framework

As a frontend team we decided to use React JS framework. Since there was no one in the team who was familiar with React before, we had a little difficulty with coding with React but we adapted in a short time. We took advantage of React's component based structure and implement reusable codes.

### 11.2.2 IDE

We did not make any restriction on which IDE to use in the project, so we preferred 3 different IDEs to use as frontend team. These were IntelliJ, WebStorm and Vscode. Although we have used 3 different IDEs, we have not had any problems in the project so far because all 3 had git integration and they worked in harmony with each other.

### 11.2.3 UI Framework

We were between 2 React UI frameworks which are React Bootstrap and Material UI. We decided with the mobile team which one to use in terms of compatibility, and chose Material UI. The main factor in our decision was that there were larger components in Material UI framework.

## 11.3 Mobile Part

### 11.3.1 Mobile Framework

As a mobile team we decided to use Android framework and Kotlin language. Since there was no one in the team who was familiar with Android development before, we start with tutorials. At first, we had a hard time implementing Kotlin and Android application. Even if we could do what we wanted later, we could not find the most accurate and efficient way due to time constraints. We still have shortcomings, but we are getting used to it. We believe that we should learn about Kotlin and Android before writing more code.

### 11.3.2 IDE

In the IDE selection decision, when we did the research, we came to the conclusion that Android Studio was the best. We continued to develop our project on Android Studio. Some parts were easy to use due to its similarity to IntelliJ. On the other hand, there are features and shortcuts that we still cannot discover. We decided to learn these as well, so that we can develop our project more efficiently.

### 11.3.3 UI Framework

We are using the framework of the Android within the scope of the UI framework. We implement our UI file in .xml format. We had to use different kind of layout( linear, relative and constraint) to design our pages.

## 11.4 Backend Part

### 11.4.1 Backend Frameworks

As the backend team, we decided to use Django Rest Framework. Since there was no one in the team who was familiar with Django and Rest Framework before, we had a little difficulty with coding with Django but we adapted in a short time. We took advantage of Django's MVT architecture and ORM to implement reusable codes.

### 11.4.2 IDE

We used Visual Studio Code and Pycharm. Visual Studio Code is open source and free and we already accustomed to using it. It provides many functionalities that the other IDEs provide. We used pylint as coding structure and it also support this extension. Besides, there are super cool themes designed for the python implementation in VS code.

### 11.4.3 Database and Other Tools

We used MongoDB as database and MongoDB Atlas as cloud database service. We integrated it via djongo engine because djongo lets us use Django with MongoDB without changing the Django ORM.

We used Postman and Swagger to test our endpoint while implementing the endpoints.

## 12 Assessment of the Customer Presentation

First of all, for the direction of our project it can be told that it is going according to the plan with a couple of alterations in the schedule. Comparing to first milestone, for the second milestone we had much more functionalities to implement. We implemented email verification, user settings, shopping cart/list, add product by vendor, product page with comment display, search-sort-filter, payment order and messaging. In order to do our presentation according to the limited time we have, we made two change of plans. For the frontend part we decided displaying messaging and for the mobile we decided to display search filtering and sorting functionalities in the final milestone. Other than these two changes we showcased every functionality according to the plan.

In frontend part, it was suggested that in the user settings both for vendor and customer we need to display more structured user information. Especially for the address part of the vendor, our implementation is just writing as a string without much of a validation system for the address part. We need to fix that part according to commonly used global address fields. Our presentation overall progressed smoothly since we try to handle every type of situation while we implemented the functions. Product comment displays, list features along with search, filter and sort algorithms worked as planned. For the ordering part there weren't any issues, but we were suggested to display our orders in a separate page for the next milestone. We will handle this specifically in the shipment/order progress part. In addition, a minor bug was pointed out in our presentation. When we do comment and rating together if one forgets to rate since we initially take it as 0 it automatically lowers the average score of the product. We have to do a check whether some give 0 deliberately or not. Other than this, there weren't any malfunction happened in the presentation and our abundant number of features get praised.

In mobile part, it was suggested that we have to change the display of password taking. We are able to see each individual letter after that it becomes *. This has to get fixed. Overall, there wasn't any major issues either in the mobile part. It went correctly according to the scenario. In the order page our GDPR clicked button was considered as a well touch but we were also advised that we can display the term file for the order. Also, for the product pages, in mobile we need to display the average ratings too. Ratings are only visible for each comment in current implementation. This has to change. There was also a design choice discussion happened for mobile in the question part of the presentation. We were told that it could be better to delete products from Wishlist after we purchase them. However, since this is an open ending suggestion we will discuss as a team and implement accordingly. Moreover, chat messaging feature worked fine too along with the other features which enable us to achieve a good presentation.

When we look at the whole presentation, we can say that we showed what we offered without mistakes for Milestone 2. For the first milestone the most difficult part was the communication between each team member and connection of the development teams since it was the beginning. However, when we look at the second milestone it was evident that we did a good job giving feedback to each other and overcoming the coding issues together. Since we had many features to implement for this milestone, of course we faced with more difficulties if we compare with the first milestone. In the end, we can say that for the second milestone, the presentation and the implementation were a success.