Milestone 2 Report

2020 Fall CmpE451 Group 9

5 January 2021

Contents

1 Executive Summary

1.1 Project Description

Our task is to develop an e-commerce website from scratch. The site should have a complete backend with functioning connections to the frontend part. Also, we are supposed to develop a mobile application connecting to the backend part. We choose to develop an iOS application for the mobile part, since our members have more experience developing applications for iOS.

1.2 Work Done So Far

We have passed the halfway mark of the development process. We have provided most of the core functionalities to our projects for frontend, backend and mobile.

On frontend, customers can now browse products and add them to their shopping cart or custom lists. They can view product details and comments/reviews. Vendors can add products to their inventory as well as edit/delete them. Profile page view/edit functionalities are added for both customers and vendors. A new customer signup page is implemented, along with a complete email confirmation process.

On iOS, we handled all previous cases above as well as additionally filtering and sorting. Additionally, backend supplies the required JSON requests as needed.

1.3 Road Ahead

For the future, our initial aim is to implement a commenting/reviewing structure, sorting/filtering mechanism along with an improved search experience for the users, messaging system, vendor/customer addresses(frontend part), order structure until the following milestone. We plan to accustom these new functionalities according to the requirements and other design documents.

2 List and Status of Deliverables

Deliverable	**Status**
Revised Requirements	Delivered
Revised Project Plan	Delivered
Frontend	
Home page design and backend connection	Delivered (1st Milestone)
Customer signup screen	Delivered (1st Milestone)
Vendor signup screen	Delivered
Customer/vendor email confirmation and forgot password screens	Delivered
User Login screen	Delivered (1st Milestone)
Customer/vendor profile screens and change password functionality	Delivered
Filter with category functionality	Delivered
Basic search functionality	Delivered
Product details page	Delivered
Add to cart/list functionalities and view cart/list pages	Delivered
Product comment and review section	Delivered
Vendor add/delete/edit product functionality	Delivered
Backend	
Sign-in Sign-up Endpoints	Delivered
Order Endpoints	Delivered
Customer Lists Endpoints	Delivered
Shopping Cart Endpoint	Delivered
Product Adding Endpoint	Delivered
Comment Endpoints	Delivered
Search and Sort Functionality and Endpoints	Delivered
Messaging endpoints	**Postponed**
Mobile	
Searching View	Delivered
Vendor Sign-Up View	Delivered
Filtering View for Search	Delivered
User Lists View	Delivered
Profile Editing View	Delivered
Categories View	Delivered
Product Detail View	Delivered
Cart View	Delivered
Google Sign-In View	Delivered
Backend Connections for All Existing Views	Delivered
Comments View	Delivered
Messaging Functionality	**Postponed**

3 Evaluation of the Status of Tools and Deliverables

1. **Milestone 2 - Group Deliverable:**
2. **Backend:**
   - User Sign-in/Sign-up: All requirements (1.1.1.*. - 1.1.2.*.) in this context are completed. Users can sign-up sign-in. Additionally, we decided that only customers can use Google API to sign in, and updated the requirements as needed.
   - User Profile: Any user is able to get his/her and other users' profile data and change his/her password and profile information. User can also reset his/her password by using his/her e-mail account.
   - Product: Products can be added by vendors and viewed by everyone.
   - Product Categories: Requirements 1.1.5.*. are completed. Vendor decides of the category and subcategory of the product that he/she adds.
   - Product Order: Requirements 1.1.9.*. , 1.1.11.*. , 1.1.12.1. , 1.1.12.2. , 1.1.12.3., 1.1.12.4. , 1.1.13.1. are completed.
   - Shopping Cart: Requirements 1.1.8.*. are completed
   - Product Comments: Requirements 1.1.4.2. and 1.1.4.3. are completed. User can post only 1 comment for each product. User must rate the product while posting the comment.
   - Product Search: Requirements 1.1.6.*. , 1.2.1.*. are completed. Search has semantic search, filtering and sorting features.
   - Customer Lists: Requirements 1.1.7.*. are completed. Public lists, private lists, alert list, and shopping cart list works properly. Customers can add product to lists and delete product from them.
   - Vendor Profile: Requirements 1.1.10.*. are completed.
3. **Frontend:**
   - Customer sign up: All related requirements (in 1.1.1. Sign Up part of the requirements document) for the Customer sign up has been completed except for 1.1.1.3,1.1.1.4, 1.1.1.5, 1.1.1.6. The backend connection for user sign up is also completed. (1st Milestone)
   - Vendor sign up: In addition to the requirements specified in 1.1.1.1, the vendors should now specify their addresses when they sign up, as specified in 1.1.1.3(instead of using Google maps, they manually give their full address.)
   - User profile: User profile is done for the customer and vendors, but not for admins yet. The design and basic backend connectivity of the user profile is done, however this view needs more work in order to be done for now (1st Milestone)

- User login: All related requirements (1.1.2. Sign In part of the requirements document) for the user login has been completed. The backend connection of user sign up view is done. (1st Milestone)
- Home page: Home page design is completed. Products and their details are coming from the backend properly. Users can filter the products according to their category, as specified in 1.1.5.1. Filter/sort functionality(1.1.5.2) is not complete yet.
- Product Page: Users can view the details of a product, as specified in 1.1.4.1. Though the comment/review section is visible, users can not comment on/review products as of yet.(1.1.4.2, 1.1.4.3)
- Customer lists/carts: Customers can create, edit, delete custom lists as specified in 1.1.7.1, 1.1.7.2, 1.1.7.3. They can add/remove items from their cart as specified in 1.1.8.1, 1.1.8.2. The functionalities specified in 1.1.7.4 and 1.1.8.3 are not implemented yet.
- Add product for vendor: Vendors can add/edit/delete products as specified in 1.1.11.1.

4. **iOS:**

- Searching View: Searching view is completed with semantic search and its backend connections. Since search history feature in the backend is not functional right now, we implemented it locally for the mobile application.
- Vendor Sign-up View: Vendor Sign-Up view is completed with the company name and address field additions as requested by the customer. Also, Google Maps integration is done for choosing the address in order to provide better user experience.
- Filtering View for Search: Filtering view for search is completed with filtering by brand, price and rating specifications. Sorting search results by price, sell count and ratings are also available in filtering view. Filtering by vendor and stock status in 1.1.6.2 are omitted since it is not found as needed by the team and the customers. 1.1.6.4 and "sorting by comments" in 1.1.6.3 are also omitted because of similar reasons.
- User Lists and List Detail View: User lists view and list detail view are completed with their backend connections. Editing and deleting lists are done in this view, and adding products to a list is done in the product detail view. 1.1.7.4 is omitted since it is not very useful or needed for the customer.
- Profile Editing View: Currently, profile editing view has the functionalities for editing the name-surname and password of the user. Editing the address field will be implemented later along with its backend counterpart.

- Categories View: Categories view is completed with search bar integration and all categories are shown in this view.
- Product Detail View: Product detail view currently has adding to list, adding to cart, choosing an amount, showing product information functionalities. Seeing the vendor details functionality will be added later along with its backend counterpart for the next milestone.
- Cart View: Cart view is completed with its backend connections. User can delete a product from cart or edit the amount of that product in cart in cart view. Buying functionality is not supported for now.
- Comments View: Comments view is done but still needs further improvement.
- Google Sign-In View: Google Sign-In view is completed along with its backend connections."if they have signed up via their Google account" part in 1.1.2.2 is omitted since this would ensure a better user experience.
- Backend Connections for All Existing Views: Backend connections are done as stated above.

As can be seen, we have mostly been conformant to our project plan except for some exhaustive requirements and functionalities for some views, except for messaging. Messaging is not completed right now since its backend counterpart is not available for now. The work that has been done on the mobile part except for those who are stated above as "not completed" don't need further improvement, which is a sign that mobile project is proceeding as expected. Requirements numbered as 1.1.1.5, 1.1.1.6, 1.1.3, 1.1.4.2, 1.1.4.3, 1.1.5.3, 1.1.9, 1.1.10, 1.1.11, 1.1.12, 1.1.13, 1.1.14, 1.1.15 are not implemented yet and will be implemented for the next milestone.

5. **API documentation:** The API documentation was essential and it has been quite helpful for everyone. Our initial API documentation is growing with each feature, and our plan is to expand it so that all backend requirements are displayed.

   We started with a Postman collection; however, it is no longer needed to be updated frequently. Even if we still use it now and then, we can say that the well-written API documentation has replaced it.

6. **GitHub Page:**

   GitHub page is very essential for a development team consisting of 12 people, because without sufficient communication between the developers, a project easily can go wrong in any way possible. Therefore, our team always made our GitHub page stay up-to-date, used issues very effectively to make decisions, especially about design, meeting times, tools and processes.

GitHub "Discussions" feature is also used effectively for making announcements for the API updates.

All of our meeting notes are not available in our GitHub page and we will try to add them as much as we can remember, both the team and sub-team meetings are available in the GitHub Wiki.

4 Unit Test Commits

4.0.1 Ömer

- Product: 2 Test Cases: SHA 26b36987cb85325fb2fea0e7e4e6bd9dda6b9339
- Product testcase bugfix: SHA a30dfc4658fa64d5acdfac93da08a2e15a021b32

4.0.2 İbrahim

- User Unit tests are added: SHA 3ce204396ec61fec391d9906a55c37d3f3ff5057
- User unit tests fixed: SHA d187c4ee8877085937def600e920410b97b65883

5 Challenges We Met

Our project requires critical effort in order to achive our needs.

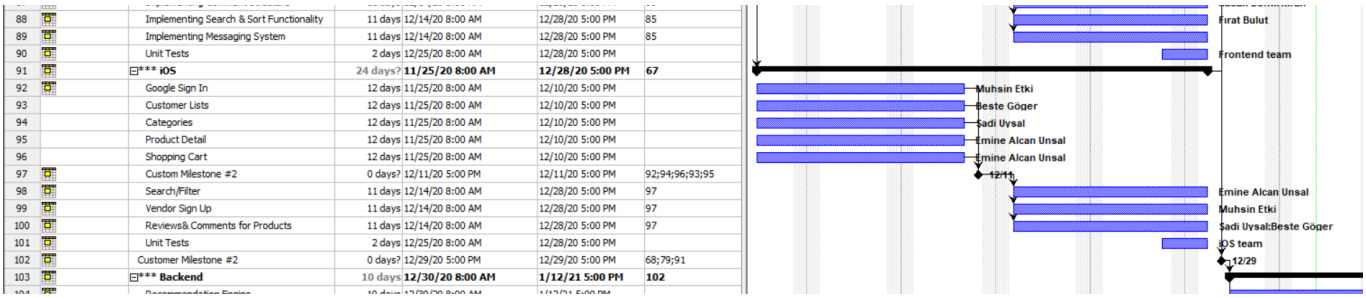
We have faced some challenges in API responses. We had miscommunications within our subgroups about responses of search API. That is why we had to add second search API for Front end. Moreover, in search API we generated words related to the searched word and searched database with respect to those generated words. However, when we tested the results, since we could only add limited number of products, we faced with irrelevant results. That is why we had to make adjustments such as generating less number of words and stop searching in product details. Also sign up system for vendor is changed and that required some changes in sign up API. Vendors had to sign up via adding their locations to database and API is adjusted with respect to change in sign up system.

Another problem was in deployment process. Since we use AWS Free Tier, we have to be efficient using our server, the storage etc. However, there was a problem we came across which required our server to re-deploy, that caused our IP address to change.

As the frontend team, we faced some work distribution issues among the team. This is due to the fact that 1 of our members was not available for unknown reasons and we had to delegate his workload to other members. Toward the end, this morphed into a serious time management issue, since we could not anticipate the extra workload. We addressed this issue by allocating our time to more important parts of the project such as search functionality or cart/list screens rather than trying to fix every little detail of mostly completed features. This is a sub-optimal solution but it was necessary.

6 Project Plan

68		***** Backend**	24 days?	**11/25/20 8:00 AM**	**12/28/20 5:00 PM**	**67**
69		Implementing Order Structure	12 days	11/25/20 8:00 AM	12/10/20 5:00 PM	
70		Implementing Customer Lists	12 days	11/25/20 8:00 AM	12/10/20 5:00 PM	
71		Implementing Shopping Cart	12 days	11/25/20 8:00 AM	12/10/20 5:00 PM	
72		Implementing Product Adding	12 days	11/25/20 8:00 AM	12/10/20 5:00 PM	
73		Custom Milestone #2	0 days?	12/11/20 5:00 PM	12/11/20 5:00 PM	69;70;71;72
74		Implementing signup for vendor/email confir	11 days	12/14/20 8:00 AM	12/28/20 5:00 PM	
75		Implementing Comment Structure	11 days	12/14/20 8:00 AM	12/28/20 5:00 PM	73
76		Implementing Search & Sort Functionality	11 days	12/14/20 8:00 AM	12/28/20 5:00 PM	73
77		Implementing Messaging System	11 days	12/14/20 8:00 AM	12/28/20 5:00 PM	73
78		Unit Tests	2 days	12/25/20 8:00 AM	12/28/20 5:00 PM	
79		***** Frontend**	24 days?	**11/25/20 8:00 AM**	**12/28/20 5:00 PM**	**67**
80		Fixing issues from the last milestone	12 days	11/25/20 8:00 AM	12/10/20 5:00 PM	
81		Implementing Order Structure	12 days	11/25/20 8:00 AM	12/10/20 5:00 PM	
82		Implementing Customer Lists	12 days	11/25/20 8:00 AM	12/10/20 5:00 PM	
83		Implementing Shopping Cart	12 days	11/25/20 8:00 AM	12/10/20 5:00 PM	
84		Implementing Product Adding	12 days	11/25/20 8:00 AM	12/10/20 5:00 PM	
85		Custom Milestone #2	0 days?	12/11/20 5:00 PM	12/11/20 5:00 PM	80;81;82;83;84
86		Google signin/signup	11 days	12/14/20 8:00 AM	12/28/20 5:00 PM	85
87		Implementing Comment Structure	11 days	12/14/20 8:00 AM	12/28/20 5:00 PM	85
88		Implementing Search & Sort Functionality	11 days	12/14/20 8:00 AM	12/28/20 5:00 PM	85
89		Implementing Messaging System	11 days	12/14/20 8:00 AM	12/28/20 5:00 PM	85
90		Unit Tests	2 days	12/25/20 8:00 AM	12/28/20 5:00 PM	
91		***** iOS**	24 days?	**11/25/20 8:00 AM**	**12/28/20 5:00 PM**	**67**



88		Implementing Search & Sort Functionality	11 days	12/14/20 8:00 AM	12/28/20 5:00 PM	85
89		Implementing Messaging System	11 days	12/14/20 8:00 AM	12/28/20 5:00 PM	85
90		Unit Tests	2 days	12/25/20 8:00 AM	12/28/20 5:00 PM	
91		***** iOS**	24 days?	**11/25/20 8:00 AM**	**12/28/20 5:00 PM**	**67**
92		Google Sign In	12 days	11/25/20 8:00 AM	12/10/20 5:00 PM	
93		Customer Lists	12 days	11/25/20 8:00 AM	12/10/20 5:00 PM	
94		Categories	12 days	11/25/20 8:00 AM	12/10/20 5:00 PM	
95		Product Detail	12 days	11/25/20 8:00 AM	12/10/20 5:00 PM	
96		Shopping Cart	12 days	11/25/20 8:00 AM	12/10/20 5:00 PM	
97		Custom Milestone #2	0 days?	12/11/20 5:00 PM	12/11/20 5:00 PM	92;94;96;93;95
98		Search/Filter	11 days	12/14/20 8:00 AM	12/28/20 5:00 PM	97
99		Vendor Sign Up	11 days	12/14/20 8:00 AM	12/28/20 5:00 PM	97
100		Reviews& Comments for Products	11 days	12/14/20 8:00 AM	12/28/20 5:00 PM	97
101		Unit Tests	2 days	12/25/20 8:00 AM	12/28/20 5:00 PM	
102		Customer Milestone #2	0 days?	12/29/20 5:00 PM	12/29/20 5:00 PM	68;79;91
103		***** Backend**	10 days	**12/30/20 8:00 AM**	**1/12/21 5:00 PM**	**102**

7 User Scenarios from the Presentation

7.0.1 Scenario 1: Mobile

Persona:

- Şahin Doğan
- 42 years old
- Car Dealer

Preconditions:

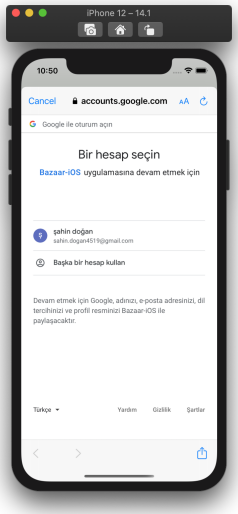
- He discovered Bazaar recently and have been signing up with Google.
- He already has a customer list on Bazaar named "new year gifts".
- He is married with 1 child.

Goals:

- He wants to buy new year gifts for his dear ones.

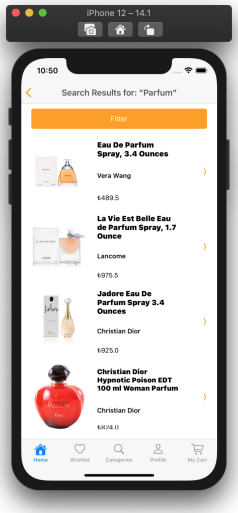
Scenario:

1. Şahin opened the Bazaar app from his iPhone.
2. He confronts the main page but wants to check whether he is signed in or not.
3. He goes to Profile section, clicks on "Sign in with Google" icon.
4. He signs in with his Google account.

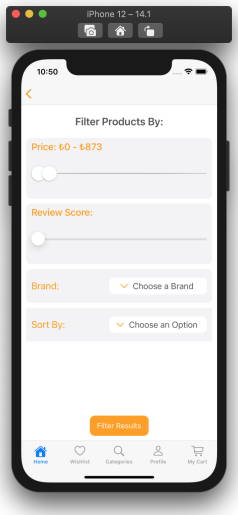


5. Then, he turns back to the main page and looks at the products of the day on the home page to see what he can get as a new year gift for his wife.

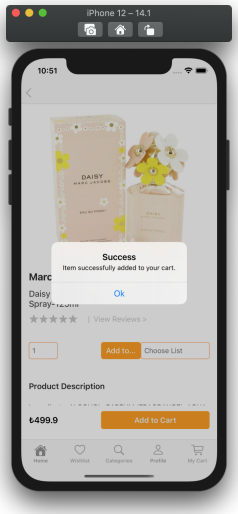
6. While looking, he decides to look for a perfume and searches for the word "parfum" by entering the keyword in the search bar.



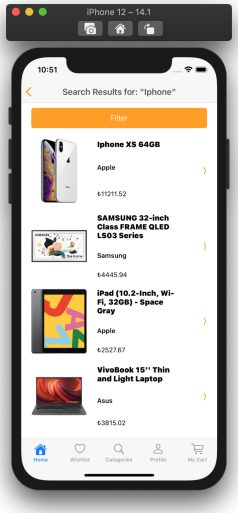
7. Since he doesn't want to purchase a perfume that costs more than 850, entering the price filter as "up to 850".



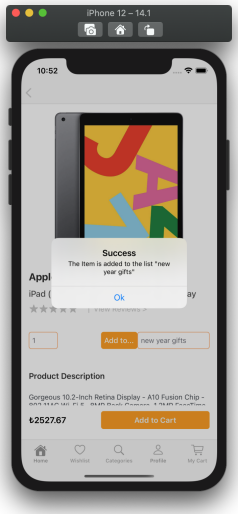
8. From the filtered search results he clicks on the perfume that he knows his wife have been using for years. And he adds it to the cart.



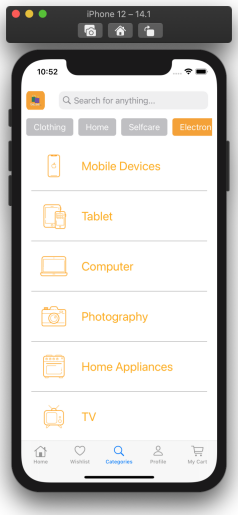
9. Then, remembering that he haven't bought any gifts for his son, he decides to search for "iPhone". Then he sees an iPad in the search results and thinks the tablet will make sense for his son who has been wanting a tablet for a long time.



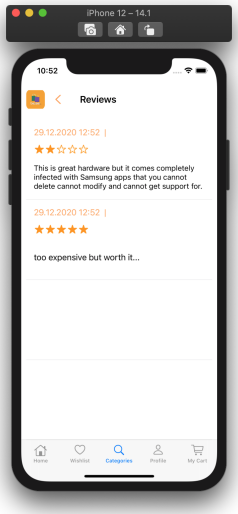
10. He adds the iPad to the "new year gifts" list for later browsing.



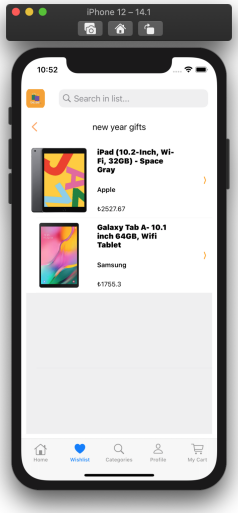
11. To see more tablet options, he enters the category page. In the electronics section, he clicks on the Tablet section.



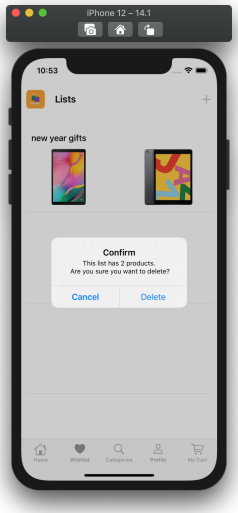
12. He then checks the reviews for Samsung tablets. He adds the tablet with higher rating and better comments to his new year list.



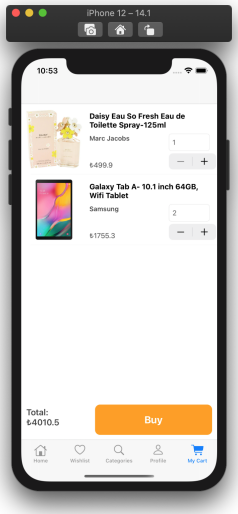
13. After examining the products from the list in detail, he adds the selected tablet to the basket for his son.



14. He then removes the list since he will soon finish his new year shopping.



15. He clicks on the "My Cart" tab and sets the amount of the tablet to 2 since he also wants to buy a gift for his nephew.



16. Then, he leaves the cart to be purchased later.

7.0.2 Scenario 2: Web

Persona:

- Efe Doğramacı
- 36 years old
- Skincare products salesman

Preconditions:

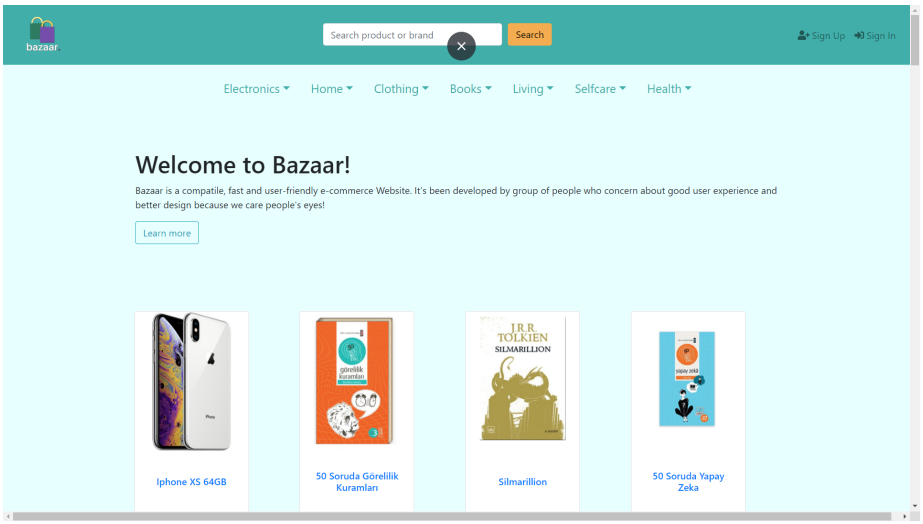
- He uses web browser.
- He is not registered to the Bazaar application.

Goals:

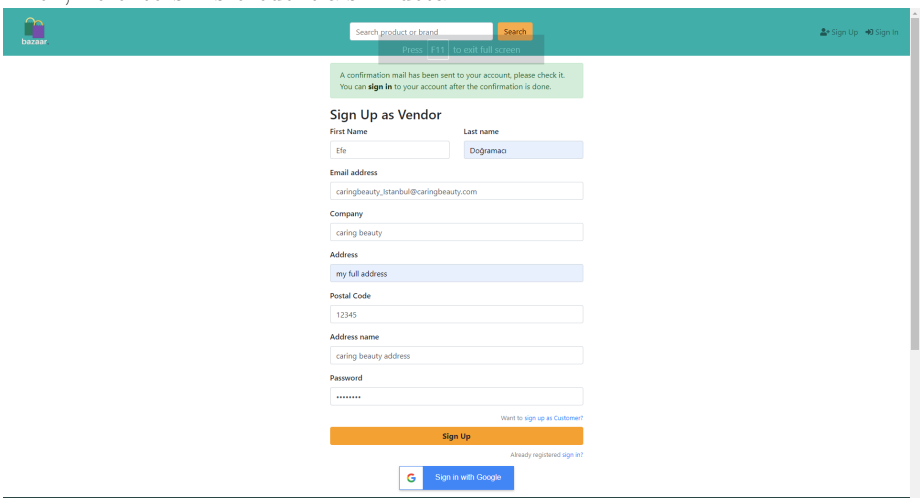
- He has online shops in various e-commerce sites like Hepsiburada and Gittigidiyor. He recently heard about Bazaar and we wants to enter the market in this emerging e-commerce site as well.

Scenario:

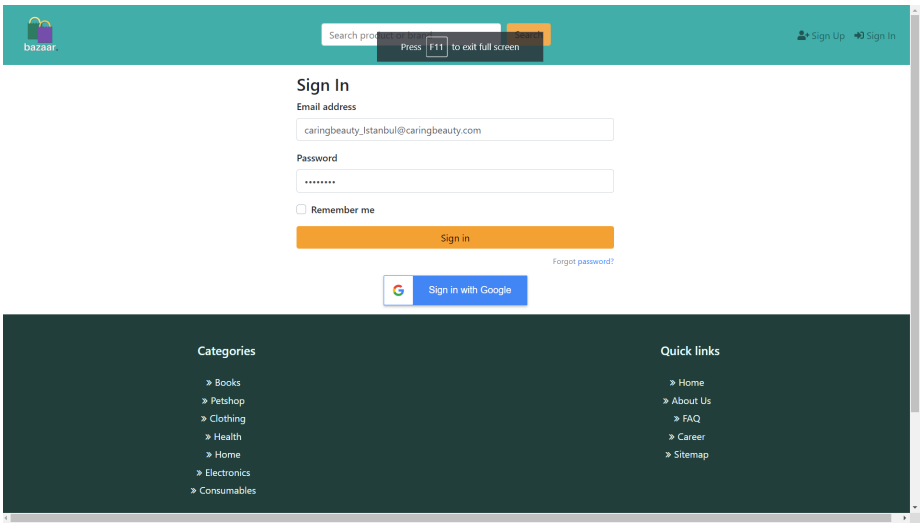
1. He confronts the main page.



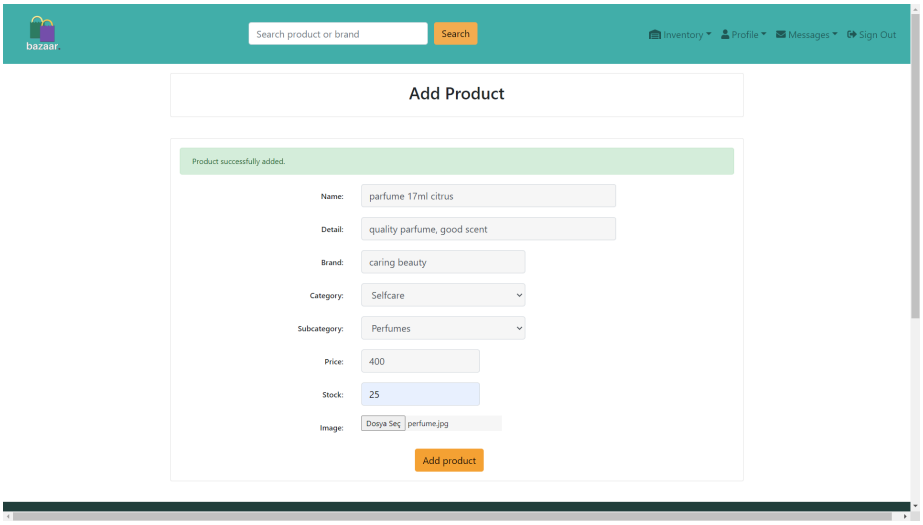
2. He wants to create an account therefore, he clicks on sign-up.
3. He clicks on 'sign-up as vendor' link and proceeds to the sign-up page
4. Then, he enters his credentials in detail.

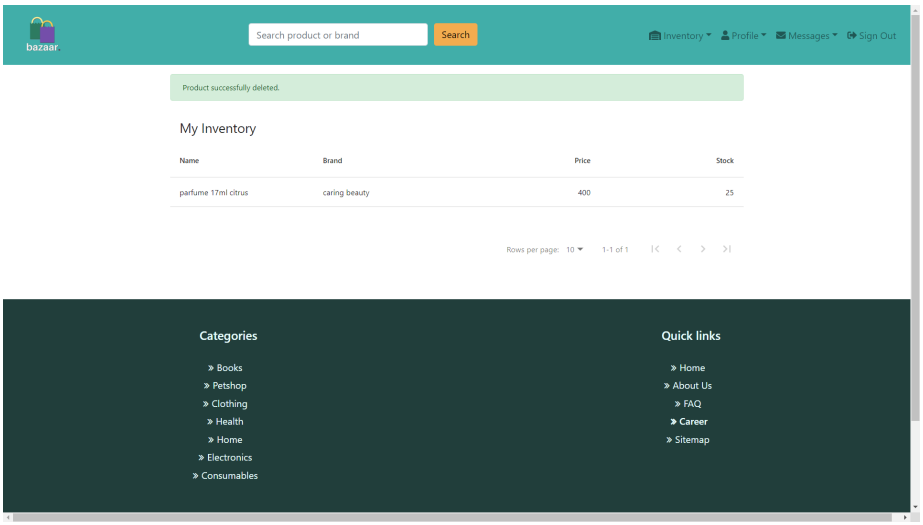


5. When he clicks on 'signup', he is confronted with the message informing him about the confirmation email.
6. He opens up his email account, clicks on the link and confirms his email.
7. Then, he proceeds to signin to his new account.

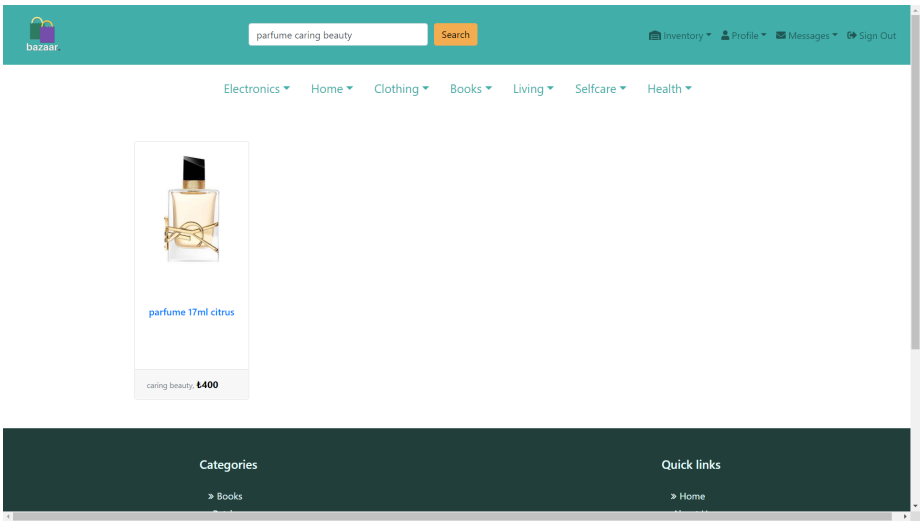


8. He goes ahead and adds 2 products to his inventory. He then realizes he made an error and deletes one of them.





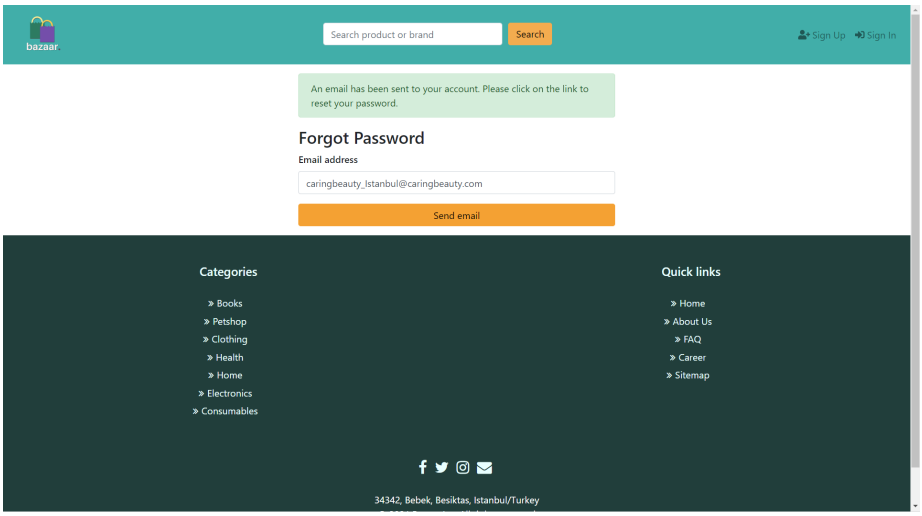
9. Later, he uses the search bar to check if his newly added product appears in the results.



10. He confirms that his product can be seen among the search results. He is a happy vendor.

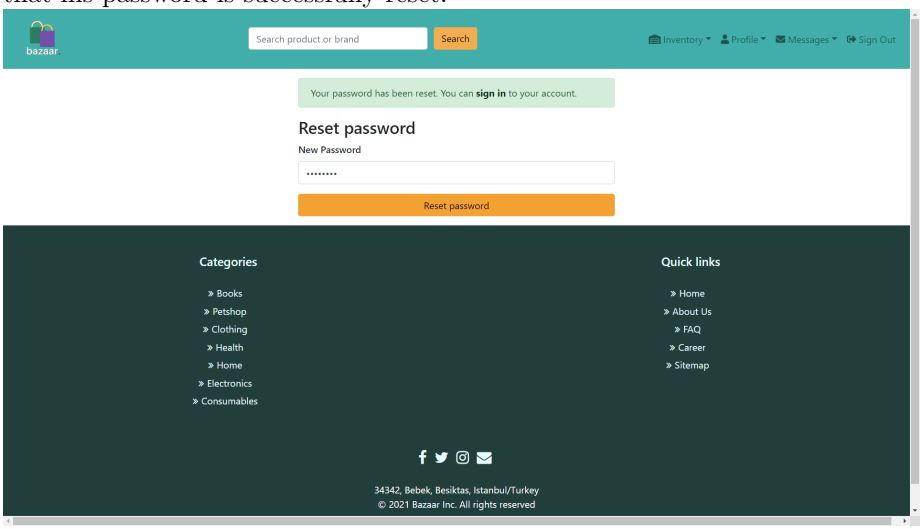
11. He then signs out accidentally. Upon trying to login to the site again, he realizes he forgot his password.

12. He clicks on the 'forgot password' link and the site asks for an email address in order to send the 'reset password' link. He enters his email address and checks his mail account again.



13. He sees the 'reset password' link, clicks on it and he is redirected to the site, where he is supposed to enter his new password.

14. He enters a new password and is met with a confirmation message, stating that his password is successfully reset.



15. He then proceeds to login to the page again. Upon successfully logging in, he gives a sigh of relief.

8 Tools and Processes

Go to project's GitHub repository.

- GitHub:
  GitHub allows us to easily handle version management. Apart from the Wiki Page which we used in cmpe352; in cmpe451, we started to work around branches more professionally. Each branch has a name that clearly shows it's parent branch in order to keep up with the development. There are 3 master branches excluding the "master branch" itself: **backend_master, frontend_master and ios_master**.

  Additionally, we use GitHub Desktop, which is a GUI that eases working around git, branches, commits, fetches and so on.

- React:
  React is an open-source, front end, JavaScript library for building user interfaces or UI components. It is maintained by Facebook and a community of individual developers and companies. React can be used as a base in the development of single-page or mobile applications.

- Bootstrap:
  Bootstrap contains free HTML and CSS templates for interface components. Our main page's template is based on Bootstrap. It provided us some very sleek templates and allowed us to have an eye-appealing design.

- Django:
  Django is a powerful python framework, used to create our backend part of the project. It is quite spanning, and it can be extended with extra frameworks. For example, we used **Django REST Framework**, which is an extention of Django, especially designed for RESTful backend projects.

- SQLite: We were planning to use MySQL as our database server, but temporarily switched to SQLite. However SQLite satisfies all of our needs and we are pretty happy to use it. So as our plan suggested, we plan to use SQLite permanently.

- Dockers:
  In order to deploy our project, we benefited from Docker images/containers. Since we were already experienced with Docker, this was a lesser time-consuming part of the project.

- XCode:
  We chose the iOS operating system for mobile development. We are developing the application using swift language with the help of the Xcode software development tool. Xcode is an integrated development environment for macOS, developed by Apple to develop software for iOS. It is used not only for iOS development, but also for macOS, watchOS and tvOS development, but we will only use it for iOS development.

- IDEs/Editors:
  Every project member uses his/her own project development environment they are familiar with. Some examples are: VSCode, PyCharm, WebStorm etc..

- Sourcetree: Apart from GitHub desktop, the iOS team needed an external version control system because Sourcetree offers better functionalities for this purpose than GitHub desktop when it comes to iOS development. In Atlassian's own wording, Sourcetree provides a graphical interface for repositories, between users and Git, in which simplifies its use for beginners, who haven't mastered Git, and experts, who can be more productive focusing solely on the code.

9 API Documentation

9.0.1 /api/ [GET]

list of all other requests, useful for exploration

```
GET {{url}}/api/
```

```
{
    "user": "{{url}}/api/user/",
    "customer": "{{url}}/api/user/customer/",
    "vendor": "{{url}}/api/user/vendor/",
    "product": "{{url}}/api/product/",
    "categories": "{{url}}/api/product/categories/"
}
```

9.1 User Calls

9.1.1 /api/user/ [GET]

list of all users

```
GET {{url}}/api/user/
```

```
[
    ...
    {
        "id": 1,
        "email": "tunaTugcu@bazaar.com",
        "first_name": "Tuna",
        "last_name": "Tuğcu",
        "date_joined": "2020-11-18T12:49:51Z",
        "last_login": "2020-11-23T07:51:49Z",
        "user_type": 2,
        "bazaar_point": 0
    },
    ...
]
```

9.1.2 /api/user/<id>/ [GET]

detailed description of the specified user

```
GET {{url}}/api/user/<id>/
```

```
{
    "id": 1,
    "email": "tunaTugcu@bazaar.com",
```

```
    "first_name": "Tuna",
    "last_name": "Tuğcu",
    "date_joined": "2020-11-18T12:49:51Z",
    "last_login": "2020-11-23T07:51:49Z",
    "user_type": 2,
    "bazaar_point": 0
}
```

9.1.3 /api/user/profile/ [GET]

detailed description user, specified via token

```
GET {{url}}/api/user/profile/
HEADER "Authorization":"Token ..."
```

```
{
    "id": 1,
    "email": "tunaTugcu@bazaar.com",
    "first_name": "Tuna",
    "last_name": "Tuğcu",
    "date_joined": "2020-11-18T12:49:51Z",
    "last_login": "2020-12-08T13:22:35.477892Z",
    "user_type": 2,
    "bazaar_point": 0
}
```

9.1.4 /api/user/profile/ [PUT]

detailed description user, specified via token

```
PUT {{url}}/api/user/profile/
HEADER "Authorization":"Token ..."
```

```
"field" : "value"
```

9.1.5 /api/user/login/ [POST]

login request that returns the user and the token

- requires"username", "password" to authenticate
- returns 400 if failed
- returns 200 and "token", "user_id", "user_type" as JSON

```
POST {{url}}/api/user/login/
BODY:
{
    "username": "username",
    "password": "password"
}
```

```
{
    "token": "**TOKEN**",
    "user_id": id,
    "user_type": 2
}
```

9.1.6 /api/user/resetpwmail/ [POST]

Sends reset password mail

- requires "username"
- returns 201

```
POST {{url}}/api/user/resetpwmail/
BODY:
{
    "username": "username",
}
```

```
{
"message": "An mail has been sent to your email, please check it"
}
```

9.1.7 /api/user/resetpw/ [POST]

new password will be set here

- requires "new_password"
- returns 400 if failed
- returns 201

```
POST {{url}}/api/user/resetpw/
BODY:
{
    "new_password": "new_password",
}
```

```
{
     'status': 'success',
    'code': status.HTTP_200_OK,
    'message': 'Password updated successfully',
}
```

9.1.8 /api/user/resetpwprofile/ [POST]

for users want to set new password from profile page

- requires “new_password”
- returns 400 if failed
- returns 201

```
POST {{url}}/api/user/resetpwprofile/
BODY:
{
    "new_password": "new_password",
    "old_password" : "old_password",
    "user_id" : id(int)
}
```

```
{
     'status': 'success',
    'code': status.HTTP_200_OK,
    'message': 'Password updated successfully',
}
```

9.1.9 /api/user/signup/ [POST]

signup request which sends an email to activate the user later

- requires “username”, “password”, “user_type”
- returns 400 if requirements not satisfied
- returns 201 if succesful, sends mail to the given email(username)

For customer signup body must be

```
POST {{url}}/api/user/signup/
BODY:
{
    "username": "username",
    "password": "password",
    "user_type": 1
}
```

For vendor signup body must be

```
POST {{url}}/api/user/signup/
BODY:
{
    "username": "username",
```

```
    "password": "password",
    "user_type": 2,
    "address_name" : "address_name",
    "address" : "address",
    "postal_code" : postal_code(integer),
    "longitude" : logitude(Float),
    "latitude" : latitude(Float),
}

{
    "message": "An mail has been sent to your email, please check it"
}
```

9.1.10 /api/user/googleuser/ [POST]

signup and signin request for google users. Only customers can signup with google. Vendors should signup via signup API

- requires "username", "password"
- returns {"token": ["token is not valid"]} and 400 if token is not valid
- returns 400 if requirements not satisfied
- returns 201 if succesful

```
POST {{url}}/api/user/googleuser/
BODY:
{
    "username": "username",
    "token" : "some_token",
    "first_name" : "first_name",
    "last_name" : "last_name"
}
```

If user not signed up before response will be

```
{
    "id": id,
    "email": "email",
    "first_name": "first_name",
    "last_name": "last_name",
    "date_joined": "2020-12-24T17:51:26.160826Z",
    "last_login": null,
    "user_type": 1,
    "bazaar_point": 0,
    "company": ""
}
```

If user signed up before and tries to login response will be

```
{
    "token": "some_token",
    "user_id": id,
    "user_type": 1
}
```

9.1.11 /api/user/activate/<uidb64>/ [GET]

account activation link for the specified user

- returns 200 if verification complete
- returns 400 if there has been a problem with a json body explaining the problem, ie: {"message": "user is already verified"}

```
GET {{url}}/api/user/activate/{{uidb64}}/
```

```
{
    "message": "Your Account, username has been activated"
}
```

9.2 Product Calls

9.2.1 /api/product/ [GET, POST]

GET:
list of all products

```
GET {{url}}/api/product/
```

```
[
    ...
    {
        "id": 3,
        "name": "Iphone XS 64GB",
        "brand": "Apple",
        "labels": ["10% off"],
        "category": {
            "name": "Electronics",
            "parent": "Categories",
            "id": 3
        },
        "price": 11211.52,
        "stock": 48,
        "sell_counter": 0,
        "rating": 0.0,
```

```
        "vendor": 2
    },
    ...
]
```

POST:
Create a product as a vendor

- picture is set to null by default, for now
- must be logged in as a vendor
- requires `"name", "brand", "price", "stock"` parameters to work, `"detail"` is available but not mandatory
- returns 201 if created succesfully
- returns 400 if failed

```
POST {{url}}/api/product/
HEADER "Authorization":"Token ..."
BODY:
{
    "name":"myproduct",
    "brand":"mybrand",
    "price":3.95,
    "stock":10000
}

{
    "id": 21,
    "name": "myproduct",
    "brand": "mybrand",
    "labels": [],
    "detail": "",
    "category": {
        "name": "Categories",
        "parent": "",
        "id":<int>
    },
    "price": 3.95,
    "stock": 10000,
    "sell_counter": 0,
    "rating": 0.0,
    "vendor": 31,
    "picture": null
}
```

9.2.2 /api/product/<id>/ [GET, DELETE]

GET:
detailed information of the specified product

```
GET {{url}}/api/product/<id>/ [GET]
```

```
{
    "id":"...",
    "name":"...",
    "brand":"...",
    "labels":[
        "..."
    ],
    "category": {
    ...
    }
    "price":"...",
    "stock":"...",
    "sell_counter":"...",
    "rating":"...",
    "vendor":"...",
    "picture":"...",
}
```

DELETE:
deletes the product

- must be logged in as the owner of the product

```
DELETE{{url}}/api/product/<id>/
HEADER "Authorization":"Token ..."
```

```
{
    "message": "product id <id> deleted"
}
```

9.2.3 /api/product/vendor/<id>/ [GET]

list of all products of the vendor

```
GET {{url}}/api/product/vendor/2/
```

```
[
    ...
    {
        "id": 3,
        "name": "Iphone XS 64GB",
```

```
        "brand": "Apple",
        "labels": ["10% off"],
        "category": {
            "name":"Mobile Devices",
            "parent":"Electronics",
            "id": 44
        }
        "price": 11211.52,
        "stock": 48,
        "sell_counter": 0,
        "rating": 0.0,
        "vendor": 2
    },
    ...
]
```

9.2.4 /api/product/search/str:filter_type/str:sort_type/ [POST]

- returns the list of products according to searched word

  `POST /api/product/search/<str:filter_type>/<str:sort_type>/`

  Filter Types:
  "none" = No filter
  "prc=num1-num2" = returns searched products with prices between num1 and num2(Example "prc=100-200")
  "pr=num1" = returns searched products with rating equals to num1 or higher (Example "pr=3")
  "br=brand1" = returns searched products whose brands are brand1 (Example "br=Ibrahim-Textile")
  "ss=num1-num2" = returns searched products with stock numbers are between num1 and num2 (Example "ss=100-200")

For multiple filters use "&" between queries (Example "prc=100-200&br=Nike")

`Sort Types:`

"none" = No sorting "bs" = sorts products according to their sell counter. Best sellers on top "mf" = sorts products according to their rating. Most Favorite on top "pr_des" = sorts products according to their price in descending order "pr_asc" = sorts products according to their price in ascending order

`Example Usage`

`/api/product/search/prc=100-300\&br=Spalding/bs/`

```
HEADER "Authorization":"Token ..."
BODY :
{
   "searched" : "word"
}
```

```
{
  "product_list" : [
    {
        "id": 3,
        "name": "Iphone XS 64GB",
        "brand": "Apple",
        "labels": ["10% off"],
        "category": {
            "name": "Electronics",
            "parent": "Categories",
            "id": 3
        },
        "price": 11211.52,
        "stock": 48,
        "sell_counter": 0,
        "rating": 0.0,
        "vendor": 2
    },
    {
        some product
    },
   ]
}
```

9.3 Product List Calls

9.3.1 /api/user/<user_id>/lists/ [GET, POST]

GET:
lists of the given user

- returns the **public** lists of the user
- returns both **public** and **private** lists, if the given user is the one logged in

```
GET {{url}}/api/user/6/lists/
```

```
[
...
    {
        "id": 1,
```

```
        "name": "custom_list",
        "customer": 6,
        "products": [
            {
                "id": 4,
                "name": "50 Soruda Görelilik Kuramları",
                "brand": "BILIM VE GELECEK KITAPLIĞI",
                "labels": [],
                "detail": "",
                "category": {
                    "name": "Books",
                    "parent": "Categories",
                    "id": 8
                },
                "price": 29.02,
                "stock": 2842,
                "sell_counter": 158,
                "rating": 3.8,
                "vendor": 2,
                "picture": null
            }
        ],
        "is_private": false
    },
...
]
```

POST:
create new list for given user

- must be logged in

```
POST {{url}}/api/user/<id>/lists/
HEADER "Authorization":"Token ..."
BODY:
{
    "name": "mylist",
    "customer": 6,
    "is_private":"false"
}

{
    "id": 27,
    "name": "mylist",
    "customer": 6,
```

```
    "products": [],
    "is_private": true
}
```

9.3.2 /api/user//list/<list_id>/ [GET, PUT]

GET:
specified list of the given user

- returns 400 if list is **private** and given user is not logged in

```
GET {{url}}/api/user/6/list/1/
```

```
{
    "id": 1,
    "name": "custom_list",
    "customer": 6,
    "products": [
        {
            "id": 4,
            "name": "50 Soruda Görelilik Kuramları",
            "brand": "BILIM VE GELECEK KITAPLIĞI",
            "labels": [],
            "detail": "",
            "category": {
                "name": "Books",
                "parent": "Categories",
                "id": 8
            },
            "price": 29.02,
            "stock": 2842,
            "sell_counter": 158,
            "rating": 3.8,
            "vendor": 2,
            "picture": null
        }
    ],
    "is_private": false
}
```

PUT:
allows to edit the name and whether is is public/private of given list

```
PUT {{url}}/api/user/6/list/1/
BODY:
```

```
{
    "name": "edited name",
    "is_private":"false"
}

{
    "id": 1,
    "name": "edited name",
    "customer": 6,
    "products": [...],
    "is_private": false
}
```

DELETE:
deletes list

- must be logged in as the owner

```
DELETE {{url}}/api/user/6/list/42/
HEADER "Authorization":"Token ..."

{
    "message": "list, id: 42, is deleted"
}
```

9.3.3 /api/user//alert_list/ [GET]

alert_list of the given user

- special list which must always be private
- user must be logged in
- to be able to add products to the alerted_list, one needs to use **/api/user/<user_id>/list/<list_id>/edit/**.

```
GET {{url}}/api/user/6/alert_list/
HEADER "Authorization":"Token ..."

{
    "id": 18,
    "name": "",
    "customer": 6,
    "products": [],
    "is_private": true
}
```

9.3.4 /api/user/<user_id>/list/<list_id>/edit/ [POST,DELETE]

POST:
adds product to the given list

- user must be logged in

```
POST {{url}}/api/user/6/list/18/edit/
HEADER "Authorization":"Token ..."
BODY:
{
    "product_id":"3"
}
```


```
{
    "id": 18,
    "name": "",
    "customer": 6,
    "products": [
        {
            "id": 3,
            "name": "Iphone XS 64GB",
            "brand": "Apple",
            "labels": [
                "10% off"
            ],
            "categories": [],
            "price": 11211.52,
            "stock": 48,
            "sell_counter": 0,
            "rating": 0.0,
            "vendor": 2,
            "picture": null
        }
    ]
}
```

DELETE:
removes product from given list

```
DELETE {{url}}/api/user/6/list/18/edit/
HEADER "Authorization":"Token ..."
BODY:
{
    "product_id":"3"
}
```

```
{
    "id": 18,
    "name": "",
    "customer": 6,
    "products": []
}
```

9.3.5 /api/product/categories/ [GET]

- returns the list of available categories
- it does not return the default category, named `Categories`

```
GET /api/product/categories/

[
...
    {
        "name": "Electronics",
        "parent": "Categories",
        "id": 3
    },
    {
        "name": "Home",
        "parent": "Categories",
        "id": 4
    },
...
]
```

9.4 Cart Calls

9.4.1 /api/user/cart/ [GET, POST, PUT, DELETE]

- all requests return the final cart with a response code `200` (except DELETE, see below)
- requires to be logged in as a customer

GET:
returns the cart

```
GET /api/user/cart/
HEADER "Authorization":"Token ..."

[
    ...,
    {
```

```
        "id": 5,
        "amount": 18,
        "product": 19,
        "customer": 6
    },
    ...
]
```

POST:
adds product to the cart

- if product already in the cart, increases the amount inside the object

```
POST /api/user/cart/
HEADER "Authorization":"Token ..."
BODY:
{
    "product_id": 3,
    "amount": 18
}
```

```
[
    ...,
    {
        "id": 5,
        "amount": 18,
        "product": 19,
        "customer": 6
    },
    ...
]
```

PUT:
edit the amount in the cart

```
PUT /api/user/cart/
HEADER "Authorization":"Token ..."
BODY:
{
    "product_id": 19,
    "amount":1755
}
```

```
[
    ...,
    {
        "id": 5,
```

```
        "amount": 1755,
        "product": 19,
        "customer": 6
    },
    ...
]
```

DELETE:
remove product from the cart

- if product not in cart, status code will be `204`

```
DELETE /api/user/cart/
HEADER "Authorization":"Token ..."
BODY:
{
    "product_id": 5
}

[...]
```

Note that object with `product = 5` is deleted

9.5 Location Calls

9.5.1 /api/location/byuser/ [GET]

```
GET {{url}}/api/location/byuser/
HEADER "Authorization":"Token ..."

{
        "id": 9,
        "address_name": "address_name",
        "address": "address",
        "postal_code": 12345,
        "longitude": 0.0,
        "latitude": 0.0,
        "user": 1
    }
```

9.5.2 /api/location/byuser/ [POST]

- longitude and latitude fields can be used for google maps

```
POST {{url}}/api/location/byuser/
HEADER "Authorization":"Token ..."
BODY:
{
```

```
    "address_name": "address_new name",
    "address": "address_user2",
    "postal_code": 12345,
    "user": 2
}

{
    "id": 11,
    "address_name": "address_new name",
    "address": "address_user2",
    "postal_code": 12345,
    "longitude": 0.0,
    "latitude": 0.0,
    "user": 2
}
```

9.5.3 /api/location/byuser/<id>/ [PUT]

- id = location id

```
PUT {{url}}/api/location/byuser/<id>/
HEADER "Authorization":"Token ..."
BODY:
{
    "address_name": "address_name",
    "address": "address_2",
    "postal_code": 12345,
    "user": "2"
}

{
    "id": 11,
    "address_name": "address_name",
    "address": "address_2",
    "postal_code": 12345,
    "longitude": 0.0,
    "latitude": 0.0,
    "user": 2
}
```

9.5.4 /api/location/byuser/<id>/ [DELETE]

- id = location id
- returns 204

```
DELETE {{url}}/api/location/byuser/<id>/
HEADER "Authorization":"Token ..."

{
    "id": 11,
    "address_name": "address_name",
    "address": "address_2",
    "postal_code": 12345,
    "longitude": 0.0,
    "latitude": 0.0,
    "user": 2
}
```

9.6 Comment Calls

9.6.1 /api/product/comment/<id>/ [GET]

- list of all comments of a product
- id = product id

```
GET {{url}}/api/product/comment/<id>/

[
    {
        "id": 1,
        "timestamp": "2020-12-07T02:34:29.373809Z",
        "body": "my comment updated again and again",
        "rating": 2,
        "is_anonymous": true,
        "customer": null,
        "product": 9
    },
    {
        "id": 28,
        "timestamp": "2020-12-07T06:34:28.693269Z",
        "body": "my comment 4",
        "rating": 3,
        "is_anonymous": false,
        "customer": 28,
        "product": 9,
        "email": "rajah.faolan@extraale.com",
        "first_name": "",
        "last_name": "",
        "date_joined": "2020-11-26T14:36:15.093169Z",
        "last_login": "2020-11-26T14:45:13.455166Z",
```

```
        "user_type": 1,
        "bazaar_point": 0
    }
]
```

9.6.2 /api/product/comment/<pid>/<uid>/ [GET]

- user's comment if available
- pid = product id
- uid = user id

```
GET {{url}}/api/product/comment/\<pid\>/\<uid\>/
HEADER "Authorization":"Token ..."

[
    {
        "id": 4,
        "timestamp": "2020-12-07T06:34:28.693269Z",
        "body": "my comment 4",
        "rating": 3,
        "is_anonymous": false,
        "customer": 28,
        "product": 9
    }
]
```

9.6.3 /api/product/comment/ [POST]

- You should check whether user has a comment on that product.
- If user has a comment on that product, use put request to update it.
- Rating shouldn't be null.

```
POST {{url}}/api/product/comment/
HEADER "Authorization":"Token ..."
BODY:
{
    "body": "my comment 4",
    "rating": 3,
    "customer": 28,
    "is_anonymous": false,
    "product": 9
}
```

```
{
        "id": 4,
        "timestamp": "2020-12-07T06:34:28.693269Z",
        "body": "my comment 4",
        "rating": 3,
        "is_anonymous": false,
        "customer": 28,
        "product": 9
}
```

9.6.4 /api/product/comment/update/<id>/ [PUT]

- id = comment id

```
PUT {{url}}/api/product/comment/update/<id>/
HEADER "Authorization":"Token ..."
BODY:
{
    "id": 1,
    "body": "my comment updated",
    "rating": 2,
    "customer": 28,
    "is_anonymous": false,
    "product": 9
}
```

```
{
        "id": 1,
        "timestamp": "2020-12-07T06:34:28.693269Z",
        "body": "my comment updated",
        "rating": 2,
        "is_anonymous": false,
        "customer": 28,
        "product": 9
}
```

9.6.5 /api/product/comment/update/<id>/ [DELETE]

- id = comment id
- returns 204

```
DELETE {{url}}/api/product/comment/update/<id>/
HEADER "Authorization":"Token ..."
```

10 Assessment of the Presentation

For the mobile part of the presentation demo, there were no negative feedbacks. The app was presented according to the previously planned scenario and there were no bugs or problems that we could see.

For the frontend part, we presented the work we have done successfully. We have completed most of the core functionalities we proposed in the project plan and had no issues while presenting them. One criticism I would make would be that we could have worked on the scenario a bit more. Since we wanted to present some overlapping features such as 'reset password' and 'change password', we had a hard time coming up with a clean scenario.

Overall, we are satisfied with our presentation. Only setback is that we exceeded the time limit by 2-3 minutes. Even though the presentation was efficiently divided between frontend and mobile parts, it was a bit longer than anticipated in total.

11 Summary of coding work done

Hasan Demirkiran	• Improved home page design. • Edited product card layout in home page/search results pages. • Implemented product details page. • Implemented shopping cart/custom lists pages. • Added 'Add product to cart/list' option to the product details page, as well as a comments/ratings section. • Changes in navbar depending on Customer/Vendor login. • Modifications in navbar menu. • Fix some bugs in terms of git usage. • Bugfixing in general.

Muhsin ETKI	• Perform general cleaning and editing on the project after first milestone. • Change the login and sign up flow • Remove login and logout buttons from profile page • Update the continue as guest functionality • Add Google Sign-in framework • Import Google Sign in configuration on AppDelegate • Add extension GIDSignInDelegate to LoginViewController and SignUpViewController • Redesign the profile page • Add icons for profile page and categories page • Fix bugs in both my branches and other branches • Adapt api changes made in backend to mobile side • Develop and design the My Account page • Add 'reset pasword by email' functionality • Get user profile information from backend • Update reset password on my account page • Develop setProfileInformation api functions • Add forgot password functionality in login page • Add GoogleSignIn connection with backend • Add vendor sign up design and implementation • Add MapViewController and get location from map

Ömer Cihan Benzer	• Initial Table(Django.Model) design for SubOrder • Table update on: User, Vendor, Product, Category • Initial Serializer design for: – ProductList – AlertList – Cart(SubOrder) • Serializer updates on: User, Product, Categories, Label • API-url pattern design, forming API Json standarts for: – ProductList(GET, POST, PUT, DELETE) – Cart(GET, POST, PUT, DELETE) – *Vendor(GET All)* – *Customer(GET All)* – *Categories(GET All)* • Update on API views in: Main URL, User(PUT), Product(PUT) • Image storage system for products • Various bug fixes/error handling on: GoogleUserLogin, Signup, Search, Categories • Pull requests/conflict handling at: backend_master, backend_live

Halil İbrahim Orhan	• Implementation of search functionality by name and brand • Implementation of Datamuse API call • Implementation of filter and sort functionalities on search results • Implementation of user unit tests • Implementation of Google user sign up and login • Implementation of resetting password on user profile functionality • Implementation of SearchHistory model • Changed Sign up API for vendors • Several Bug fixes

Emine Alcan Unsal	• Searching functionality for the search bar in MainViewController. • SearchResultsViewController and all of its functionalities and backend connections for the app. • FilterViewController design, all functionalities and backend connections for the app. • Product image downloading and caching system for the whole application. • ProductDetailViewController design, all functionalities and backend connection except for "adding a product to a list" functionality. • MyCartViewController design, all functionalities and its backend connections for the app. • MainViewController backend connection for productsTableView. • Adding the application logo to the assets. • Searching by brand and category in the app, since it is not available in the backend. • Local search history implementation for the app. • Addition of DropDown, MarkRangeSlider CocoaPods and StarRatingView to the application.

Mehmet Berk Kemaloğlu	• Implementing location API
	  – Location(GET, POST, PUT, DELETE)
	• Implementing product comment and rating API
	  – Comment of User(GET, POST, PUT, DELETE)
	  – Comments of Product(GET)

Fırat Bulut	• Fix some leftover issues from the first presentation. • Implement an inventory page and add/edit/remove product page for vendors. • Design signup page for vendors. • Design reset password, forgot password pages, implement email confirmation and reset password features. • Implement category structure. Categories can be accessed from each page in the navbar. • Implement search functionality. • Add 'Success' and 'Something went wrong' messages to add product, sign up, forgot password, reset password pages.

Beste Göger	• WishlistViewController design is created and completed. • Backend connection for product lists, fetching user's product lists • Add new list, Edit list, Delete list functionalities • ListDetailViewController design, delete item from the list functionality and backend connections • Adding an item to the list functionality is completed in the ProductDetailViewController. • ReviewViewController design and its backend connections for product reviews and creation the segue from ProductDetailViewController

Şadi Uysal	• CategoriesViewController is created and completed. • Created and completed SubCategories Table View • Created and designed subCategory Cells • Backend connection for subcategories • Integrated search funtionality into Categories page • Added icons and updated design of the Categories page according to customer feedbacks • Resolved bugs about the compatible IOS versions • Resolved bugs in main storyboard

Ahmet Bilal Uçan	• -

Mehmet Can Ünal	• Review and making changes on requirements and design documents. • Discussion about design issues with front-end team. • Creation and design of Navbar. • Changed navbar was rendered after authentication . • Adding sign up, sign in and sign out parts navigation to other pages. • Profile, massages and cart dropdowns were added and will be rendered for each user. • Sign in with google • Transparent change on logo

Gökhan Tekel	• Review requirements and design patterns for the first time • Being involved to the team • Update project plan • Create User API • Being Cognizant of Front and Back side coherence • Review and update Login Signup Endpoints • Decide on the structure of Database tables • Order API created • Decide on the structure of Database tables • Frontend Google Map created • few Bugs Handled