CmpE 451 - Milestone 3 Report

GROUP 1

January 31st, 2021

Table of Contents

1. Final Project Assessment

1.1. Frontend

What Went Well

- **We had fruitful meetings.** Usually all team members joined the weekly meetings of the frontend team and contributed to the project. We used the meetings to show each other the work we had done in the previous week, get feedback about our work and share ideas to improve our project. We also discussed the tasks that needed to be done in the following week and distributed those tasks among the team members. Overall, I think that our meetings were very effective.
- **We had good communication.** All team members were understanding, helpful and responsive during the whole process. We helped each other when one of the team members was struggling with an issue and responded to questions and comments in a timely manner.
- **We agreed on a design before we implemented it.** When a new interface was going to be implemented we took some time during our meetings to draw a very simple sketch of what we were expecting it to look like. This helped us to have a clearer idea of how our design should look like.
- **We did not miss internal deadlines.** Even though there were some exceptions we usually met our internal deadlines.
- **We managed to provide the necessary functionalities.** The interfaces we created were functional and enabled the users to carry out their tasks.

What did not Go Well

- **Our progress was often slower than we anticipated.** None of us had experience in writing software for a frontend application, so we were all amateurs. Thus, we had to do research and learn things that were new to us quite often. And it was hard to predict the amount of time that was necessary to do so. Thus, our time estimations were often wrong and it was frustrating.
- **Our designs could have been better.** Since we invested a lot of time to get the functionalities to work, we did not have enough time and motivation to enhance our designs.
- **We did not consider UI and UX as much as we should have.** After we had completed the implementation of a functionality we sometimes realized that the usage of the functionality felt inconvenient and unnatural. But at that point it was often too late to change it. We should have considered these issues beforehand, however due to our lack of experience we often failed to do correct estimations.

Overall Assessment

We made some mistakes along the way but overall we created a good and functional product. We always tried to react to the feedback that we got from the customer as well as the one we shared internally. Also we learnt a lot during the process.

Lessons Learnt

- We learnt to use Git very effectively.
- We learnt how to hold fruitful meetings and maintain a good communication between team members.
- We learnt how to use React to create a frontend application.
- We learnt how to give good presentations to a customer.
- We learnt to accept criticism and use it to enhance our work.

1.2. Backend

What Went Well

- **We had fruitful meetings.** Our weekly meetings were very effective, we were great on syncing weekly and deciding upon the next steps we need to take. We did a good job on demonstrating the features on the meetings as well so that we can improve our functionalities before demonstrating it to the customer.
- **We had good communication.** We used our communication channels efficiently. When we needed to get an opinion on something or share something we used Slack and it worked out great. On the important decisions to make, we discussed them through our channels and did zoom meetings if needed. Everyone was quick to respond so we did not run into delays due to communication. We also communicated with the customers for uncertain things or for getting feedback.
- **We did not miss internal deadlines.** We worked hard to meet our deadlines on our plan. We tried to finish up things early to leave some time for the web/android integration as well.
- **We managed to provide the necessary functionalities.** We implemented all the functionalities and more that we decided at the beginning of the project. We communicated with the other teams as much as within the team constantly to provide all needed functionalities.
- **We implemented unit tests for our functionalities.** For the last 2 milestones we were careful to implement unit tests for our big functionalities. Since we needed to change things along the way, those unit tests were helpful.
- **We made improvements along the way.** Even though functionality is implemented, we did not stop improving it if it needed improvements.

What did not Go Well

- **We could have spent more time on designing the endpoints.** Before implementing the endpoints, we could have communicated more with the other teams to understand what they need exactly. When they needed extra endpoints we implemented them but we could have been more structural on the design process.
- **We could have spent more time on security.** For example for the products, we were showing the product ids as they are in the webpage. It would be better if we have public profile/product id's to avoid it.
- **We could have been more prepared for the demos.** Especially for the second demo our scenario was not good enough to satisfy the customer.

Overall Assessment

We were good at planning ahead. We decided on the task distribution early and we managed to make all the functionalities we promised. We also worked good as a team. We took necessary steps to get the work done, for example one of our team members changed his team to Android to be able to finish the project on time. We payed needed attention to the unit tests as well.

Lessons Learnt

- We learnt to use Github features effectively.
- We learnt the importance of being able to work as a group work when building a product.
- We learnt how important communication is when working as a group.
- We learnt how to hold efficient meetings.
- We learnt the process of building a product from scratch.
- We learnt how to use Django to create a functional API.
- We learnt how to implement good unit tests for our functions.
- We gained experience on AWS, CI/CD, Dockerization
- We learnt how to give good presentations to a customer.
- We learnt to accept criticism and use it to enhance our work.

1.3. Android

What Went Well

- **We had fruitful meetings**. All team members attend weekly meetings of the android team and contribute to the project.We showed each other what we did last week and got feedback about our work.
- **We had good communication**.We have made effective use of communication channels, and we have helped each other when one of the team members is trying to solve a problem.
- **We did not miss internal deadlines**. We usually met our internal deadlines.
- **We managed to provide the necessary functionalities.** We have indeed succeeded in implementing all the functionalities.

What did not Go Well

- **We could have spent more time on testing all functionality and layouts in the application on different android phones.**The layouts that we designed appeared with different portions in different phones, we needed to make it flexible.
- **We could have been more prepared for the presentation**. A bug occurred due to the size of the newly added product images but we fixed the bug in a short time by resizing all images.

Overall Assessment

At the end of the semester, we see that we have indeed succeeded in implementing all the functionalities that are being asked as requirements from our customer. At the end of the project, we are all happy to deliver tursu.

Lessons Learnt

From our point of view, we learned a lot of tools and frameworks in the project. More importantly, we learned about the advantages and disadvantages of teamwork. We learned how to initialize short- and long-term goals and track the work done to achieve those goals. In addition, the Android team has also experienced that designing beautiful interfaces for users is not as easy as it sounds.

2. List and Status of Deliverables

Deliverable	**Status**
Backend	
Shopping Cart Endpoints	Completed
Profile Editing Endpoints	Completed
Order Endpoints	Completed
Login & Signup Endpoints	Completed
Filtering & Sorting Endpoints	Completed
Semantic Search Endpoints	Completed
Vendor Search Endpoints	Completed
Product Photo Related Endpoints	Completed
Shopping List Endpoints	Completed

Recommendation Endpoints	Completed
Google Login & Signup Endpoints	Completed
Delete/Verify Product for Admin Endpoints	Completed
Customer Profile Page Endpoints	Completed
Vendor Profile/Product Page Endpoints	Completed
Commenting Endpoints	Completed
Ban User for Admin Endpoints	Completed
Notification and Alert Endpoints	Completed
Messaging Endpoints	Completed
Verification and Forgot Password Endpoints	Completed
Activity Stream	Completed
Documentations	Completed
CI/CD	Completed
Android	
Sign In Page - Customer/Vendor	Completed
Sign Up Page - Customer/Vendor	Completed
Forgot Password Page - Customer/Vendor	Completed
Email Verification Page - Customer/Vendor	Completed
Homepage	Completed
Product Page	Completed
Search	Completed
Filtering	Completed
Shopping Cart/List Page	Completed
Shopping Lists Page	Completed
User Profile Page	Completed
Order Page	Completed
My Products Page	Completed

Add Product page	Completed
Commenting and rating functionality	Completed
Payment Page	Completed
Public vendor page visible to all customers	Completed
Recommendation	Completed
Messaging Page	Completed
Notification System	Completed
Google Login/Sign up	Completed
Frontend	
Sign In Page - Customer/Vendor	Completed
Sign Up Page - Customer/Vendor	Completed
Forgot Password Page - Customer/Vendor	Completed
Email Verification Page - Customer/Vendor	Completed
Homepage	Completed
Product Page	Completed
Search	Completed
Filtering	Completed
Shopping Cart/List Page	Completed
Shopping Lists Page	Completed
User Profile Page - My Information	Completed
User Profile Page - My Orders	Completed
User Profile Page - My Lists	Completed
User Profile Page - My Products	Completed
Add Product page	Completed
Commenting and rating functionality	Completed
Admin Panel	Completed
Payment Page	Completed

Order Page	Completed
Public vendor page visible to all customers	Completed
Messaging Page	Completed
Notification System	Completed
Footer	Completed
Activity Streams page for Admin	Completed
3rd party software acknowledgements page	Completed
Google Login/Sign up	Completed

3. Summary of Work Done by Each Member

TEAM MEMBER	CONTRIBUTION
Ali Batır	All the following things are implemented for the android for Milestone 3: • Implemented adding comment feature • Implemented public vendor page feature • Implemented uploading photo feature • Implemented multi filtering feature • Implemented the stock alert feature for the notification • Some bug fixes (details are included in commit messages) • Refactored interfaces • Reviewed merge requests For final deliverable: • Wrote the assessment and status of deliverables for android • Created and uploaded apk file
Asena Karolin Özdemir	Following work is done for Milestone 3: • Implemented messaging page together with Mehmet Çelimli. • Connected the forgot password endpoint to the interface. • Implemented the email verification page. • Implemented a page where the activity streams are displayed. • Implemented a page where the 3rd party software acknowledgements are displayed. • Added password strength checks (also tested them.) • Wrote the assessment and status of deliverables sections for frontend in this report.
Baran Deniz Korkmaz	

Barış Alhan	Following works are done for Milestone 3: ● Implemented vendor public profile page ● Implemented remaining parts of the notification system(alert set, backend connection), its other parts are implemented by Yagiz. ● Implemented footer of the webpage. ● Presented the milestone 3 to customers.
Barış Mutlu	Implementations that I completed for final milestone for Android: ● Payment page input formats added. (exp: CVC,Card number as desired format and limits) ● I implemented the profile editing feature for customer and vendor, for editable features of them in Api ● I moved the Log Out button from Profile to the Home Page Slider Menu. It was an advice that was mentioned in presentation 2. ● Some layout fixes.. ● I took some meeting notes and added Git in Android meetings.. Documentations for final milestone: ● I moved the requirements ,design document, user manual, system manual to the Final Report.
Buse Kabakoğlu	Implementations that I completed for final milestone for Android: ● Messaging between vendor-admin and vendor-customer ● Getting and refreshing notifications of a user ● Some layout fixes for different devices ● Transition fixes Documentations for final milestone: ● Preparing Android User Manual
Eray Sezgin	Android: ● Google sign-in and sign-up ● Email verification for user sign-up. ● Price alerts for products. ● Miscellaneous bug fixes.
Mehmet Çelimli	Following work is done for Milestone 3: ● Filtering multiple categories and vendors features is added to the filter bar. ● Filter bar is moved to the left side of the page. ● The bug regarding the search page is fixed. ● Messaging page is implemented with Asena. ● Chat bubbles, list of message flows are implemented. ● Admin panel is refactored and the navigation bar is added to the admin panel. ● User manual for the web is documented. ,
Murat Ekici	Following works are done for Milestone 3: ● Set up utilities for sending email. ● Implemented forgot password functionality. ● Implemented the user verification functionality using the new emailing utilities. ● Implemented the whole notification and alert logic. ● Implemented miscellaneous improvements for existing endpoints. ● Implemented unit tests for the new functionalities I implemented.

	• Set up the aws configurations for the web page domain and linked our new domain tursu.ml with Yagiz. • Communicated and worked together with other team members to integrate and improve new functionalities. • Wrote the final assessment for the backend to the final milestone report.
Onur Kılıçoğlu	All the following things are implemented for the backend for Milestone 3 • Refactored filtering to make multiple selection available • Implemented messaging endpoints and its tests • Implemented public vendor page endpoint • Implemented activity stream processing and storing For final deliverable: • Prepared the report template • Took the database dump • Wrote the activity streams part in the report • Wrote the system manual for backend
Ömer Ak	• I implemented multiple rating feature for a purchased product. • I added some products and vendors to our database. • I wrote the assessment for customer presentation for the milestone report.
Yağız Can Çolak	Following works are done for Milestone 3: • Implemented the layout of the notification system (Notification pop-up and my notifications tab in the profile pages) • Made the connection between Google Sign In and backend endpoints (Google Sign In and Google Sign Up requests) • Changed the rating system from 1 rating to 2 separate rating including product and vendor • Got a new domain address (tursu.ml) • Changed the frontend deployment domain with Murat

4. Unit Tests Committed by Each Member

4.1. Ali Batır

commit "**93aeb0e355be5f9e0161fe6acb3f7d23855015e3"**

Product Add Fragment Test

Unit tests to verify price and stock inputs. priceVerificationCheck() stockVerificationCheck()	Tests different product prices and stocks.

commit "**7815a9aa8ebc45dfb6c0d1b1cfe1a7b744462f07"**

Vendor Product Page Fragment Test

Unit tests to verify price and stock inputs. priceVerificationCheck() stockVerificationCheck()	Tests different product prices and stocks.

4.2. Asena Karolin Özdemir

commit "**3021c73b15a11bd74520c4b27001faed86db91c4"**

Unit tests about the function that checks the strength of the password which the user enters. **checkPassword()**	Tests different password examples.

4.3. Baran Deniz Korkmaz

4.4. Barış Alhan

commit "**275d7fef88adffeb13bdedf9ca5a3a63c78c63f4"**

Unit tests about sum functionality in shopping cart page.	Tests different product quantities and prices.

4.5. Barış Mutlu

I didn't implement specific unit tests for features. For the Android side, I thought testing them manually is fine because while I was implementing ,I connected my device and test & debug them step by step simultaneously.

4.6. Buse Kabakoğlu

I wrote different test cases for Payment Fragment and Home page activity

commit **7c98a2eab31096554b402c320190156d64a99dcb**

Home page test

does customer not have products on sale	Tests if the user is a customer, products on sale item will not be shown
does vendor have products on sale	Tests if the user is a vendor, products on sale item will be shown
does customer not have products add item	Tests if the user is a customer, products add item will not be shown

does vendor have products add item	Tests if the user is a vendor, products add item will be shown
does customer have shopping list item	Tests if the user is a customer, shopping lists item will be shown
does vendor does not have shopping list item	Tests if the user is a vendor, shopping lists item will be shown

Payment Fragment tests

is false for credit card number with string	Tests whether a string is given as the input for credit card number
is false for non existing month	Test whether given moth is valid
is true for different delimiter	Tests whether ., /, - are okay between dates
is true for wrong year	Tests whether the given year is valid

4.7. Eray Sezgin

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

Login Tests

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

Login View Tests

test_correct_username	Tests login request with correct username and password.
test_correct_email	Tests login request with correct email and password.
test_wrong_username	Tests login request with wrong username.

test_wrong_email	Tests login request with wrong email.
test_wrong_password	Tests login request with wrong password.
test_empty_username	Tests login request with empty username field.
test_empty_password	Tests login request with empty password field.

Sign Up Tests

test_customer_all_fields_provided	Tests creating(signing up) a user of type customer with all fields provided.
test_customer_missing_field	Tests creating(signing up) a user of type customer with the “first_name” field missing.
test_vendor_all_fields_provided	Tests creating(signing up) a user of type vendor with all fields provided.
test_vendor_missing_first_name	Tests creating(signing up) a user of type vendor with the “first_name” field missing.
test_vendor_missing_city	Tests creating(signing up) a user of type vendor with the “city” field missing.

4.8. Mehmet Çelimli

I was working in the frontend team and all the features I’ve implemented are tested via console or by using user scenarios.

Commit: **“25b0158c10a7419b22e875a5e3e0df19effbc570”**

	Unit tests for ban user and delete comment

4.9. Murat Ekici

Tests for shopping cart endpoints in backend:

Commit: "**e1e6f940dc8c90ff237ad2054c095544d98bb563**"

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

Tests for order endpoints in backend:

Commit: "**ece5941faad381692df92eb69b6a76a585bd2802**"

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

Tests for notifications/alert endpoints in backend:

Commit: "**9641f31694d32840eef2b20b06f716730e41bca6**"

test_create_alerts	Tests for creating alerts with all the existing types and checking the correctness of alerts using the get alerts endpoint.
test_delete_alerts	Tests for creating alerts with all the existing types and deleting them one by one, then checking if they are deleted correctly.
test_product_notifications	This tests if the notifications are created correctly by First creating different types of alerts for a product, then editing the product so that alerts are triggered. Then checking the notifications to check if they are created correctly.
test_set_read	Test for checking if the set_read functionality for notifications works correctly.

4.10. Onur Kılıçoğlu

Tests for search endpoint in backend:

Commit “**5ad4e2b07f58ed8d3bea840ce80c76c551f4887c”**

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

Commit “**0d50ac674281265f1871218d988d2b1d5143de3c”**

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

Tests for product/add endpoints

Commit **“11938a7a5b597f488f2239e287bf83c5756a6de0”**

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

Tests for product delete and edit endpoints:

Commit **“338271bbbd47fb35dd370a23530e328e9d35e0b1”**

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

Tests for shopping list endpoints:

Commit **“a16491d6119e02f36fbe31991f62145bca0510ce”**

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

Tests for messaging endpoints:

Commit **“4984548706b78df90ff96e6e3941e7219bf7fb54”**

test_create_flow_customer	Tests if a customer can start a chat with a vendor
test_create_flow_vendor	Tests if a vendor can start a chat with admin
test_get_flow_vendor	Tests if a vendor can get the inboxes of the chats with the customers and the vendors
test_get_flow_customer	Tests if a customer can get the inboxes of the chats with the vendors
test_get_flow_admin	Tests if admin can get the inboxes of the chats with the vendors
test_send_message_admin	Tests if admin can send message to vendor
test_send_message_vendor	Tests if vendor can send message to

	customers and admins
test_send_message_customer	Tests if a customer can send message to vendor

4.11. Ömer Ak

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

4.12. Yağız Can Çolak

commit "**1b3296ac2c02583c2a66609775ce47eec09cae25**"

Unit tests about editing a product as a vendor.	Tests different values for price and stock editing.

5. Activity stream implementation and W3C Compliance

The system processes and stores activity streams using Django Activity Stream module in Python. Currently the system processes 2 different activities:

1. Vendor discounted a product
2. Vendor added a product

To implement this the activity streams app "actstream" is added to the Django apps. Then each model that will be tracked in an activity is registered in its configuration class in "apps.py" file under the application that the model is stored. Then the activities that needs to be processed are processes in the necessary endpoints by sending the action to the Django Activity Stream function. Then the Django Activity Stream module stores the action for further use. We implemented two endpoints to get activities related to the specific vendors and the products (see as/vendor and as/product endpoints in API Document). In addition, Django Activity Stream module provides several endpoints to get activity information about the activities processed which can be found in https://django-activity-stream.readthedocs.io/en/latest/feeds.html. We added model serializers to the model classes to provide activity information in JSON format. The Django Activity Stream module implementation complies with JSON Activity Streams 1.0 "https://activitystrea.ms/specs/atom/1.0/" and in the wiki of W3C about Activity Streams "https://www.w3.org/wiki/Activity_Streams" it is mentioned that Django Activity Stream module is a previous implementation of the W3C Activity Streams.

6. API documentation

"/search/"

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

Method: GET

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

Example query:
http://3.232.20.250/search?search_string=apple&search_type=product

Response:
JSON object: List of products with product_info if search_type is product
List of vendors with vendor_info if search_type is vendor

Notes:

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

"/product/"

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

Method: GET

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

Example query:
http://3.232.20.250/product?id=1

Response:
JSON object: detailed_product_info

Notes:

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

---

"/product/category/"

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

Method: GET

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

	=numComments: sorting by number of comments	

Example query:
http://3.232.20.250/product/category?name=Home

Response:
JSON object: List of products with product_info

Notes:

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

---

"/product/add/"

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

Method: POST

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

Example query:
http://3.232.20.250/product/add/

Response:
HTTP Response: Success or fail messages

Notes:

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

---

"/product/edit/"

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

Method: POST

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

Example query:
http://3.232.20.250/product/edit/

Response:
HTTP Response: Success or fail messages

Notes:

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

---

"/product/delete/"

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

Method: POST / DELETE

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

Example query:
http://3.232.20.250/product/delete/

Response:
HTTP Response: Success or fail messages

Notes:

- Auth token should be added to the header of the request as: “Authorization: ‘Token <auth_token>’” where key is Authorization and value is string with the word “Token” and the actual auth_token seperated with a whitespace

---

"/product/addphoto/"

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

Method: POST

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

Example query:
http://3.232.20.250/product/addphoto/

Response:
HTTP Response: Success or fail messages

Notes:

- You should give the parameters in the form data.
- Auth token should be added to the header of the request as: “Authorization: ‘Token <auth_token>’” where key is Authorization and value is string with the word “Token” and the actual auth_token seperated with a whitespace

---

"/product/deleteallphotos/"

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

Method: POST / DELETE

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

Example query:

http://3.232.20.250/product/deleteallphotos/

Response:
HTTP Response: Success or fail messages

Notes:

- Auth token should be added to the header of the request as: “Authorization: ‘Token <auth_token>’” where key is Authorization and value is string with the word “Token” and the actual auth_token seperated with a whitespace

---

"/product/deletephoto/"

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

Method: POST / DELETE

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

Example query:
http://3.232.20.250/product/deletephoto/

Response:
HTTP Response: Success or fail messages

Notes:

- Auth token should be added to the header of the request as: “Authorization: ‘Token <auth_token>’” where key is Authorization and value is string with the word “Token” and the actual auth_token seperated with a whitespace

---

"/shoppingcart/add/"

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

Method: POST

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

quantity	int: quantity to be added	Optional

Example query:
http://3.232.20.250/shoppingcart/add/
Response:
HTTP Response: Success or fail messages

Notes:

- Auth token should be added to the header of the request as: “Authorization: ‘Token <auth_token>’” where key is Authorization and value is string with the word “Token” and the actual auth_token seperated with a whitespace

---

"/shoppingcart/increase/"

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

Method: POST

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

Example query:
http://3.232.20.250/shoppingcart/increase
Response:
HTTP Response: Success or fail messages

Notes:

- Auth token should be added to the header of the request as: “Authorization: ‘Token <auth_token>’” where key is Authorization and value is string with the word “Token” and the actual auth_token seperated with a whitespace

---

"/shoppingcart/decrease/"

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

Method: POST

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

Example query:
http://3.232.20.250/shoppingcart/decrease

Response:
HTTP Response: Success or fail messages

Notes:

- Auth token should be added to the header of the request as: “Authorization: ‘Token <auth_token>’” where key is Authorization and value is string with the word “Token” and the actual auth_token seperated with a whitespace

---

"/shoppingcart/delete/"

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

Method: POST / DELETE

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

Example query:
http://3.232.20.250/shoppingcart/delete
Response:
HTTP Response: Success or fail messages

Notes:

- Auth token should be added to the header of the request as: “Authorization: ‘Token <auth_token>’” where key is Authorization and value is string with the word “Token” and the actual auth_token seperated with a whitespace

---

"/shoppingcart/all/"

Returns the shopping cart for the authenticated customer.

Method: GET

Parameter	Description	Required/ Optional

Example query:
http://3.232.20.250/shoppingcart/all
Response:
JSON Response: List of products in the cart with their quantities.

Notes:

- Auth token should be added to the header of the request as: “Authorization: ‘Token <auth_token>’” where key is Authorization and value is

string with the word “Token” and the actual auth_token seperated with a whitespace

---

"/shoppinglist/createlist/"

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

Method: POST

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

Example query:
http://3.232.20.250/shoppinglist/createlist/
Response:
HTTP Response: Success or fail messages

Notes:

- Auth token should be added to the header of the request as: “Authorization: ‘Token <auth_token>’” where key is Authorization and value is string with the word “Token” and the actual auth_token seperated with a whitespace

---

"/shoppinglist/deletelist/"

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

Method: POST / DELETE

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

Example query:
http://3.232.20.250/shoppinglist/deletelist/
Response:
HTTP Response: Success or fail messages

Notes:

- Auth token should be added to the header of the request as: “Authorization: ‘Token <auth_token>’” where key is Authorization and value is

string with the word “Token” and the actual auth_token seperated with a whitespace

---

"/shoppinglist/addtolist/"

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

Method: POST

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

Example query:
http://3.232.20.250/shoppinglist/addtolist/
Response:
HTTP Response: Success or fail messages

Notes:

- Auth token should be added to the header of the request as: “Authorization: ‘Token <auth_token>’” where key is Authorization and value is string with the word “Token” and the actual auth_token seperated with a whitespace

---

"/shoppinglist/deletefromlist/"

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

Method: POST / DELETE

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

Example query:
http://3.232.20.250/shoppinglist/deletefromlist/
Response:
HTTP Response: Success or fail messages

Notes:

- Auth token should be added to the header of the request as: “Authorization: ‘Token <auth_token>’” where key is Authorization and value is

string with the word “Token” and the actual auth_token seperated with a whitespace

---

"/shoppinglist/getlists/"

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

Method: GET

Parameter	Description	Required/ Optional

Example query:
http://3.232.20.250/shoppinglist/getlists/
Response:
JSON Response: **List of strings** with the list names i.e. [“mylist1”, “mylist2”]

Notes:

- Auth token should be added to the header of the request as: “Authorization: ‘Token <auth_token>’” where key is Authorization and value is string with the word “Token” and the actual auth_token seperated with a whitespace

---

"/shoppinglist/products/"

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

Method: GET

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

Example query:
http://3.232.20.250/shoppinglist/products/
Response:
JSON Response: List of products with product_info

Notes:

- Auth token should be added to the header of the request as: “Authorization: ‘Token <auth_token>’” where key is Authorization and value is string with the word “Token” and the actual auth_token seperated with a whitespace

---

"/order/create_orders/"

Creates the order from the customer's shopping list.

Method: POST

Parameter	Description	Required/ Optional

Example query:
http://3.232.20.250/order/create_orders
Response:
HTTP Response: Success or fail messages

Notes:

- Auth token should be added to the header of the request as: “Authorization: ‘Token <auth_token>’” where key is Authorization and value is string with the word “Token” and the actual auth_token seperated with a whitespace

---

"/order/set_delivered/"

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

Method: POST

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

Example query:
http://3.232.20.250/order/set_delivered
Response:
HTTP Response: Success or fail messages

Notes:

- Auth token should be added to the header of the request as: “Authorization: ‘Token <auth_token>’” where key is Authorization and value is string with the word “Token” and the actual auth_token seperated with a whitespace

---

"/order/set_delivery/"

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

Method: POST

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

Example query:
http://3.232.20.250/order/set_delivery
Response:
HTTP Response: Success or fail messages

Notes:

- Auth token should be added to the header of the request as: “Authorization: ‘Token <auth_token>’” where key is Authorization and value is string with the word “Token” and the actual auth_token seperated with a whitespace

---

"/order/cancel_order/"

Sets the given orders as cancelled.

Method: POST

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

Example query:
http://3.232.20.250/order/cancel_order
Response:
HTTP Response: Success or fail messages

Notes:

- Auth token should be added to the header of the request as: “Authorization: ‘Token <auth_token>’” where key is Authorization and value is string with the word “Token” and the actual auth_token seperated with a whitespace

---

"/order/get_orders/"

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

Method: GET

Parameter	Description	Required/ Optional

Example query:
http://3.232.20.250/order/get_orders/
Response:
JSON Response: List of orders nested into the corresponding carts at the time.

Notes:

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

---

"/comment/"

Comment adding endpoint is used by customers to add a comment to a delivered product.

Method: POST

Parameter	Description	Required/ Optional
product_id	int: ID of the product to which the comment will be added	Required
text	string: Comment text	Required
product_rating	int: Rating for the product	Required
vendor_rating	int: Rating for the vendor that sells the product	Required

Example query:
http://3.232.20.250/comment

Response:
HTTP Response: Success or fail messages

Notes:

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

---

"/recommendation/recommended"

Recommended endpoint is used by customers to customized recommendations

Method: GET

Parameter	Description	Required/ Optional

Example query:
http://3.232.20.250/recommendation/recommended
Response:
JSON Response: List of products with product_info

Notes:

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

---

"/recommendation/bestseller"

Bestseller endpoint returns 10 bestsellers of Tursu

Method: GET

Parameter	Description	Required/ Optional

Example query:
http://3.232.20.250/recommendation/bestseller
Response:
JSON Response: List of products with product_info

Notes:

"/recommendation/toprated"

Top rated endpoint returns 10 top rated products of Tursu

Method: GET

Parameter	Description	Required/ Optional

Example query:
http://3.232.20.250/recommendation/toprated

Response:
HTTP Response: List of products with product_info

Notes:

"/recommendation/newest"

Newesr endpoint returns 10 newest added products of Tursu

Method: GET

Parameter	Description	Required/ Optional

Example query:
http://3.232.20.250/recommendation/newest

Response:
JSON Response: List of products with product_info

Notes:

"/recommendation/recommendation_pack"

Recommendation pack endpoint returns all recommendation types at the same request

Method: GET

Parameter	Description	Required/ Optional

Example query:
http://3.232.20.250/recommendation/recommendation_pack

Response:
JSON Response: **recommendation_pack:**

- **recommended** → List of products with product_info
- **bestseller**→ List of products with product_info
- **top_rated**→ List of products with product_info
- **newest_arrival**→ List of products with product_info

Notes:

- Auth token should be added to the header of the request as: “Authorization: ‘Token <auth_token>’” where key is Authorization and value is string with the word “Token” and the actual auth_token seperated with a whitespace

---

"/admin/verificationpending/products/"

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

Method: GET

Parameter	Description	Required/ Optional

Example query:
http://3.232.20.250/admin/verificationpending/products/
Response:
JSON Response: List of products with product_info

Notes:

- Auth token should be added to the header of the request as: “Authorization: ‘Token <auth_token>’” where key is Authorization and value is string with the word “Token” and the actual auth_token seperated with a whitespace

---

"/admin/removeproduct/"

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

Method: POST / DELETE

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

Example query:
http://3.232.20.250/admin/removeproduct/

Response:
HTTP Response: Success or fail messages

Notes:

- Auth token should be added to the header of the request as: “Authorization: ‘Token <auth_token>’” where key is Authorization and value is string with the word “Token” and the actual auth_token seperated with a whitespace

"/admin/verifyproduct/"

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

Method: POST

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

Example query:
http://3.232.20.250/admin/verifyproduct/

Response:
HTTP Response: Success or fail messages

Notes:

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

---

"/admin/deletecomment/"

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

Method: POST / DELETE

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

Example query:
http://3.232.20.250/admin/deletecomment/

Response:
HTTP Response: Success or fail messages

Notes:

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

---

"/admin/banuser/"

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

Method: POST

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

Example query:
http://3.232.20.250/admin/banuser/

Response:
HTTP Response: Success or fail messages

Notes:

- Auth token should be added to the header of the request as: “Authorization: ‘Token <auth_token>’” where key is Authorization and value is string with the word “Token” and the actual auth_token seperated with a whitespace

---

"/user/signup/google"

Sign up with Google endpoint.

Method: POST

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

Example query:
http://3.232.20.250/user/signup/google

Response:
JSON Response: **login_info**
Notes:

"/user/login/google"

Login with Google endpoint

Method: POST

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

Example query:
http://3.232.20.250/user/login/google

Response:
JSON Response: **login_info**

Notes:

"/user/signup"

Sign up endpoint. Sends a verification code to given email address

Method: POST

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

		for Vendor

Example query:
http://3.232.20.250/user/signup
Response:
JSON Response: {}

Notes:

"/user/login"

Login endpoint

Method: POST

Parameter	Description	Required/ Optional
email	string: Email or username of the user	Required
password	string: Password of the user	Required
verification_code	string: Verification code for the user if needed	Optional

Example query:
http://3.232.20.250/user/login
Response:
JSON Response: **login_info**

Notes:

"/user/resend_verification_code"

Sends verification email to the given email, doesn't override the old ones

Method: POST

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

Example query:
http://3.232.20.250/user/resend_verification_code
Response:
JSON Response: {}

Notes:

"/user/forgot_password"

Resets the password and sends an email for the new one.

Method: POST

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

Example query:
http://3.232.20.250/user/forgot_password
Response:
JSON Response: **{}**

Notes:

"/helper/allbrands"

Returns all existing brands in a list.

Method: GET

Parameter	Description	Required/ Optional

Example query:
http://3.232.20.250/helper/allbrands
Response:
JSON Response: **string list**

Notes:

"/helper/allvendors"

Returns all verified vendors in a list.

Method: GET

Parameter	Description	Required/ Optional

Example query:
http://3.232.20.250/helper/allvendors
Response:
JSON Response: string list

Notes:

"/helper/allcategories"

Returns all existing categories in a list.

Method: GET

Parameter	Description	Required/ Optional

Example query:
http://3.232.20.250/helper/allcategories
Response:
JSON Response: string list

Notes:

"/customerpage"

Returns customer information.

Method: GET

Parameter	Description	Required/ Optional

Example query:
http://3.232.20.250/customerpage
Response:
JSON Response: customer_info

Notes:

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

---

"/vendorpage"

Returns vendor information.

Method: GET

Parameter	Description	Required/ Optional

Example query:
http://3.232.20.250/vendorpage
Response:
JSON Response: **vendor_info**

Notes:

- Auth token should be added to the header of the request as: “Authorization: ‘Token <auth_token>’” where key is Authorization and value is string with the word “Token” and the actual auth_token seperated with a whitespace

---

"/vendorpage/public"

The public vendor page endpoint returns public information about given vendor

Method: GET

Parameter	Description	Required/ Optional
vendor_name	string: Name of the vendor	Required

Example query:

Response:
JSON Response: **public_vendor_info**

"/message/flow/admin/"

Message flow for admin endpoint returns the message flow of the admin user

Method: GET

Parameter	Description	Required/ Optional

Example query:

Response:
JSON Response: List of message flows with **message_flow_type_1**

Notes:

- Auth token should be added to the header of the request as: “Authorization: ‘Token <auth_token>’” where key is Authorization and value is string with the word “Token” and the actual auth_token seperated with a whitespace

"/message/flow/customer/"

Message flow for customer endpoint returns the message flow of the customer user

Method: GET

Parameter	Description	Required/ Optional

Example query:

Response:
JSON Response: List of message flows with **message_flow_type_2**

Notes:

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

---

"/message/flow/vendor/"

Message flow for vendor endpoint returns the message flow of the vendor user

Method: GET

Parameter	Description	Required/ Optional

Example query:

Response:
JSON Response: List of message flows with **message_flow_type_1**

Notes:

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

---

"/message/startflow/customer/"

Message start flow for customer endpoint is used by customer to start a chat with a vendor with given vendor name

Method: POST

Parameter	Description	Required/ Optional
vendor_name	string: Name of the vendor to start a chat with	Required

Example query:

Response:
HTTP Response: Success or fail messages

Notes:

- Auth token should be added to the header of the request as: “Authorization: ‘Token <auth_token>’” where key is Authorization and value is string with the word “Token” and the actual auth_token seperated with a whitespace

---

"/message/startflow/vendor/"

Message start flow for vendor endpoint is used by customer to start a chat with a admin about an order or a product
Method: POST

Parameter	Description	Required/ Optional
context	string: “**product”** or “**order”**	Required
object_id	int: **id** of the object that the context is related to i.e. MessageFlow is created to discuss on Product with id 37 → context=product, object_id=37	Required

Example query:

Response:
HTTP Response: Success or fail messages

Notes:

- Auth token should be added to the header of the request as: “Authorization: ‘Token <auth_token>’” where key is Authorization and value is string with the word “Token” and the actual auth_token seperated with a whitespace

---

"/message/send/admin/tovendor/"

Admin send message to vendor endpoint is used by admin to send message to a vendor in the given flow
Method: POST

Parameter	Description	Required/ Optional
message	string: message with max 200 characters	Required
flow_id	int: id of the related flow	Required

Example query:

Response:
HTTP Response: Success or fail messages

Notes:

- Auth token should be added to the header of the request as: “Authorization: ‘Token <auth_token>’” where key is Authorization and value is string with the word “Token” and the actual auth_token seperated with a whitespace

---

"/message/send/customer/tovendor/"

Customer send message to vendor endpoint is used by customer to send message to a vendor in the given flow
Method: POST

Parameter	Description	Required/ Optional
message	string: message with max 200 characters	Required
flow_id	int: id of the related flow	Required

Example query:

Response:
HTTP Response: Success or fail messages

Notes:

- Auth token should be added to the header of the request as: “Authorization: ‘Token <auth_token>’” where key is Authorization and value is string with the word “Token” and the actual auth_token seperated with a whitespace

---

"/message/send/vendor/tocustomer/"

Vendor send message to customer endpoint is used by vendor to send message to a customer in the given flow
Method: POST

Parameter	Description	Required/ Optional
message	string: message with max 200 characters	Required
flow_id	int: id of the related flow	Required

Example query:

Response:
HTTP Response: Success or fail messages

Notes:

- Auth token should be added to the header of the request as: “Authorization: ‘Token <auth_token>’” where key is Authorization and value is string with the word “Token” and the actual auth_token seperated with a whitespace

---

"/message/send/vendor/toadmin/"

Vendor send message to admin endpoint is used by vendor to send message to a admin in the given flow
Method: POST

Parameter	Description	Required/ Optional
message	string: message with max 200 characters	Required
flow_id	int: id of the related flow	Required

Example query:

Response:
HTTP Response: Success or fail messages

Notes:

- Auth token should be added to the header of the request as: “Authorization: ‘Token <auth_token>’” where key is Authorization and value is string with the word “Token” and the actual auth_token seperated with a whitespace

---

"/message/chat/ofcustomer/"

Chat of customer endpoint is used to get messages in the message flow with the given flow id

Method: GET

Parameter	Description	Required/ Optional
flow_id	int: ID of the message flow	Required

Example query:

Response:
JSON Response: List of **message_info:**
message_info:

- **sender**→ string: “self” if current user sent the message, “other” if current user received the message
- **customer** → string: username of the customer messaginf
- **vendor_name** → string: name of the vendor messaged
- **message** → string: message text
- **date_sent** → string: datetime repr of the time of message

Notes:

- Auth token should be added to the header of the request as: “Authorization: ‘Token <auth_token>’” where key is Authorization and value is string with the word “Token” and the actual auth_token seperated with a whitespace

---

"/message/chat/ofvendor/wadmin/"

Chat of vendor endpoint is used to get messages of vendor with admin in the message flow with the given flow id

Method: GET

Parameter	Description	Required/ Optional
flow_id	int: ID of the message flow	Required

Example query:

Response:

JSON Response: List of **message_info:**

message_info:

- **sender**→ string: “self **“**if current user sent the message, “other” if current user received the message
- **admin** → string: username of the admin messaged
- **vendor_name** →string: name of the vendor messaging
- **message** → string: message text
- **date_sent** → string: datetime repr of the time of message

Notes:

- Auth token should be added to the header of the request as: “Authorization: ‘Token <auth_token>’” where key is Authorization and value is string with the word “Token” and the actual auth_token seperated with a whitespace

---

"/message/chat/ofvendor/wcustomer/"

Chat of vendor endpoint is used to get messages of vendor with customer in the message flow with the given flow id

Method: GET

Parameter	Description	Required/ Optional
flow_id	int: ID of the message flow	Required

Example query:

Response:
JSON Response: List of **message_info:**

message_info:

- **sender** → string: “self” if current user sent the message, “other” if current user received the message
- **customer**→string: username of the customer messaged
- **vendor_name** → string: name of the vendor messaging
- **message** → string: message text
- **date_sent** → string: datetime repr of the time of message

Notes:

- Auth token should be added to the header of the request as: “Authorization: ‘Token <auth_token>’” where key is Authorization and value is string with the word “Token” and the actual auth_token seperated with a whitespace

---

"/message/chat/ofadmin/"

Chat of admin endpoint is used to get messages in the message flow with the given flow id

Method: GET

Parameter	Description	Required/ Optional
flow_id	int: ID of the message flow	Required

Example query:

Response:
JSON Response: List of **message_info:**

message_info:

- **sender** → string: **“**self” if current user sent the message, “other” if current user received the message
- **admin** → string: username of the admin messaging
- **vendor_name** → string: name of the vendor messaged

- **message** → string: message text
- **date_sent** → string: datetime repr of the time of message

Notes:

- Auth token should be added to the header of the request as: “Authorization: ‘Token <auth_token>’” where key is Authorization and value is string with the word “Token” and the actual auth_token seperated with a whitespace

---

"/as/vendor"

Returns all verified vendors in a list.

Method: GET

Parameter	Description	Required/ Optional
vendor_name	string: Name of the vendor to get the activity stream of	Required

Example query:
http://3.232.20.250/as/vendor?vendor_name=Apple
Response:
JSON Response: Activity Stream

Notes:

"/as/product"

Returns all verified vendors in a list.

Method: GET

Parameter	Description	Required/ Optional
product_id	int: ID of the product to get the activity stream of	Required

Example query:
http://3.232.20.250/as/product?product_id=1
Response:
JSON Response: Activity Stream

Notes:

"/notifications/get_notifications"

Returns a list of notifications for user.

Method: GET

Parameter	Description	Required/ Optional

Example query:

Response:
JSON Response: List of **notification:**

notification:

- **id** → int: id of the notification
- **type** → int: type of the notification (look below notification types for valid types)
- **read** → bool, true if read, false otherwise
- **product_name** → string: name of the product
- **product_id** → int: id of the product
- **order_id** → int: only if order related
- **status** → string: status order, only if order related
- **new_value** → int: changed new value, only if product related

Notes:

- Auth token should be added to the header of the request as: “Authorization: ‘Token <auth_token>’” where key is Authorization and value is string with the word “Token” and the actual auth_token seperated with a whitespace

---

"/notifications/set_read"

Set's given notification as read

Method: POST

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

Example query:

Response:
HTTP Response: Success or fail messages

Notes:

- Auth token should be added to the header of the request as: “Authorization: ‘Token <auth_token>’” where key is Authorization and value is string with the word “Token” and the actual auth_token seperated with a whitespace

---

"/notifications/create_alert"

Creates an alert with given type, product and value

Method: POST

Parameter	Description	Required/ Optional
product_id	int: id of the product to set an alam	Required
type	int: type of the alert	Required
value	int: value for according type	Optional

Example query:

Response:
Json Response: ID of the created alert

Notes:

- Auth token should be added to the header of the request as: “Authorization: ‘Token <auth_token>’” where key is Authorization and value is string with the word “Token” and the actual auth_token seperated with a whitespace

---

"/notifications/get_alerts"

Returns the alerts that the user created. If product_id is given, returns related alerts.

Method: GET

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

Example query:

Response:
JSON Response: List of **alerts:**
alert:
- **id** → int: id of the notification

- **type** → int: type of the notification (look below alert types for valid types)
- **value** → value for the according alert type
- **product_name** → string: name of the product
- **product_id** → int: id of the product

Notes:

- Auth token should be added to the header of the request as: “Authorization: ‘Token <auth_token>’” where key is Authorization and value is string with the word “Token” and the actual auth_token seperated with a whitespace

---

"/notifications/delete_alert"

Deletes given alert.

Method: POST, DELETE

Parameter	Description	Required/ Optional
id	int: ID of the alert to delete	Required

Example query:

Response:

HTTP Response: Success or fail messages

Notes:

- Auth token should be added to the header of the request as: “Authorization: ‘Token <auth_token>’” where key is Authorization and value is string with the word “Token” and the actual auth_token seperated with a whitespace

DATA STRUCTURES

product_info:

Field	Description
id	int: ID of the product
name	string: Name of the product
photo_url	string: Link to the photo of the product
vendor_name	string: Name of the vendor of the product
category	string: Name of the category of the product

rating	float: Rating of the product
stock	int: Available stock of the product
price	float: Price of the product
brand	string: Name of the brand of the product

detailed_product_info:

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

login_info:

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

customer_info:

Field	Description
username	string: User name of the customer
email	string: Email of the customer

first_name	string: First name of the customer
last_name	string: Last name of the customer
money_spent	float: Total amount of money spent of the customer
orders	order_info: Orders of the customer
lists	list of product: Product lists of the customer

order_info:

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

vendor_info:

Field	Description
username	string: User name of the vendor
email	string: Email of the vendor
first_name	string: First name of the vendor
last_name	string: Last name of the vendor
attitude	string: Attitude of the location of the vendor
longtitude	string: Longtitude of the location of the vendor
iban	string: Iban of the vendor
rating	float: Rating of the vendor

orders	order_info: Orders of the vendor
products	product_info: Products of the vendor

comment:

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

message_flow_type_1:

Field	Description
id	int: Flow id
notify	bool: True if unseen messages are in flow
context	string: “**product”** or “**order”**
object_id	int: **ID** of the object that the context is related to i.e. MessageFlow is created to discuss on Product with id 37 → context=product, object_id=37
product	string: Name of the product if the context is “product”, name of the ordered product if the context is “order
customer	string: Username of the customer who ordered the product if the context is “order”, null if context is product
type	string: “**admin”** for vendor-admin communication “**customer”** for customer-vendor communication
vendor_name	string: name of the vendor messaged

message_flow_type_2:

Field	Description
id	int: Flow id
notify	bool: True if unseen messages are in flow
vendor_name	string: name of the vendor messaged

public_vendor_info:

Field	Description
username	string: Username of the vendor
email	string: Email of the vendor
latitude	float: Latitude of the location of the vendor
longitude	float: Longitude of the location of the vendor
city	string: City of the location of the vendor
first_name	string: Name of the vendor
last_name	string: Empty string
iban	string: IBAN of the vendor
rating	float: Rating of the vendor
products	List of **product_info**

notification_types:

Field	Value
ORDER STATUS CHANGE	1
PRODUCT VERIFIED	2
PRICE BELOW ALERT	3
PRICE CHANGE ALERT	4
STOCK ABOVE ALERT	5

alert_types:

Field	Value
PRICE BELOW ALERT	1
PRICE CHANGE ALERT	2
STOCK ABOVE ALERT	3

7. Requirements

Glossary

- Banning a user: Limiting the available actions of a user to the actions that can be taken by a guest.
- Category: A type that denotes a grouping of products based on common features of the products in that group.
- Comment: A text about a product written by a customer who purchased the product to explain the interaction between the product and the customer.
- (Product) Description: A text written by a vendor, describing a product.
- Location: The physical address of a vendor.
- Order: A set of products purchased by a customer which will be delivered to the customer.
    - Active Orders: The orders that are pending delivery.
    - Delivering Stage: After processing stage product enters delivering stage in which the product is on the move to the customer and during this process, customers can keep track of the cargo.
    - Processing Stage: An ordered product comes to the processing stage and stays there before the vendor enters the unique cargo id.
- Payment Failed: The end state of payment process where the transaction of money via provided payment information cannot be completed because of a problem.
- Payment Successful: The end state of payment process where the transaction of money via provided payment information is completed with no problems.
- Product: An item to be sold on the platform by a vendor.
    - Rating: An integer score from 0 to 5 which denotes the measurement of satisfaction about the product where higher scores shows a better level of satisfaction.
    - Rating a product: Providing a rating for a product.
    - Adding a product: An action taken by the vendor to display the item to be sold on the platform by providing the necessary information to the system.
- Profile Page: A page displaying some(as much as allowed) information about a registered user.
- Shipping Information: The address where the purchased products will be delivered to.
- Shopping Cart: A set of products selected by a customer to be purchased.
- Shopping List / List: A set of products which contains the products that a customer is interested in but not willing to buy soon.
- Stock: Information about the available quantity of a product.
- Tag: An entity denoting a feature or set of features about a product with a corresponding value of the entity.
- User: A person using the e-commerce platform on any device possible.
    - Guest: A non-logged-in user that can search, display and read the comments of products.
    - Registered User : A person who completed registration process by providing necessary information including a unique email and a password to the system

and who is able to login with the unique email and password provided beforehand.
    - Admin: A user type that has special privileges to manage and organize the platform.
    - Customer: A logged-in user profile with privileges of buying products, tracking orders, creating lists, etc.
    - Vendor: A logged-in user profile that sells products using the platform.
- Verified Product: A product endorsed by admins.
- Verified Vendor: A vendor endorsed by admins.

---

1. Functional Requirements

1.1. User Requirements

1.1.1. Sign Up

- 1.1.1.1. Registered Users shall register to the system by providing an email and a username via creating a password.
- 1.1.1.2. Registered Users should register to the system using their Google account.
- 1.1.1.3. Registered Users shall provide information about whether they are a vendor or a customer while making registration.
- 1.1.1.4. Registered Users shall provide a unique email address to register.
- 1.1.1.5. A verification email shall be sent to the provided email address to complete the registration for both vendors and customers.
- 1.1.1.6. Vendor users additionally shall provide:
    - 1.1.1.6.1. their IBAN to the system.
    - 1.1.1.6.2. the location of their store through Google Maps in registration.

1.1.2. Sign In

- 1.1.2.1. Already registered users shall be able to sign in with their email and password.
- 1.1.2.2. Already registered users should be able to sign in via Google account.
- 1.1.2.3. Users should be able to reset their passwords in case they forget it.
- 1.1.2.4. The resetting process should be started by sending an email to the address provided.

1.1.3. Guest Users

- 1.1.3.1. Guests shall be able to search for products
- 1.1.3.2. Guests shall be able to view the price of a product, but shall not be able to add them to their lists or carts.
- 1.1.3.3. Guests shall be able to read user comments about products.
- 1.1.3.4. Guests shall be able to discover categories.

1.1.4. Registered Users

1.1.4.1. Common Features

- 1.1.4.1.1. Users shall provide a set of information according to account type.
- 1.1.4.1.2. Users should be able to edit their information.
- 1.1.4.1.3. Users shall be able to reset their password by using email authentication.

1.1.4.2. Vendor Profile

- 1.1.4.2.1. Vendors shall have a rating voted by customers.
- 1.1.4.2.2. In vendor profiles, one shall be able to see products of that vendor.

1.1.4.3. Admin Profile

- 1.1.4.3.1. Admin shall be able to verify products as only the verified products are for sale.
- 1.1.4.3.2. Admin shall be able to ban any registered user from the platform.
- 1.1.4.3.3. Admin shall be able to manage tags on a product.
- 1.1.4.3.4. Admin shall be able to delete a comment.
- 1.1.4.3.5. Admin shall be able to remove a product.
- 1.1.4.3.6. Admin shall be able to accept/reject cancellation requests coming from vendors.

1.1.5. Commenting on & Rating Products

- 1.1.5.1. Customers shall be able to comment on the products they have purchased.
- 1.1.5.2. Customers shall be able to read user comments about products.
- 1.1.5.3. Customers should be able to delete/edit an old comment they have committed.
- 1.1.5.4. Customers shall be able to review their products after a transaction has been completed.
- 1.1.5.5. Customers shall be able to rate a product from 1 to 5.
- 1.1.5.6. Customers shall be able to see these rates when making a purchase decision.

1.1.6. Shopping Cart

- 1.1.6.1. Customers shall be able to add products to their shopping carts to order.
- 1.1.6.2. Customers shall be able to edit their shopping carts.
- 1.1.6.3. Customers shall be able to view their shopping carts.
- 1.1.6.4. Customers shall be able to add and remove products from their shopping carts.

1.1.7. List of Favorite Products

- 1.1.7.1. Customers shall be able to create a list of products they would like to keep an eye on.
- 1.1.7.2. Lists shall be private for each user.
- 1.1.7.3. Customers shall be able to create their lists.
- 1.1.7.4. Customers shall be able to delete their lists.

- 1.1.7.5. Customers shall be able to name their lists.
- 1.1.7.6. Customers shall be able to edit their lists.

1.1.8 Setting A Notification for A Product

- 1.1.8.1. Customers shall be able to set an alarm for price changes of a specific product.
- 1.1.8.2. Customers shall be able to set an alarm for a certain price, and choose to be notified if the price of the product goes below the chosen price.

1.1.9. Adding a Product

- 1.1.9.1. Only vendors shall be able to add a product.
- 1.1.9.2. When adding a product, vendor shall select a category for the product.
- 1.1.9.3. The added product shall have a price.
- 1.1.9.4. The added product shall have its description and details.
- 1.1.9.5. Vendors shall specify the stock information of the added product.

1.1.10 Messaging Between Registered Users

- 1.1.11.1. Vendors shall be able to message directly to admins about a certain product.
- 1.1.11.2. Vendors shall be able to communicate with admins about a certain order.
- 1.1.11.3. Customers shall be able to message directly to vendors about their orders.

1.2 System Requirements

1.2.1. Search

- 1.2.1.1. The system shall provide product searching for all users.
- 1.2.1.2. The system shall provide vendor searching for admin users.
- 1.2.1.3. The system shall only list the verified products.
- 1.2.1.4. The system shall support search with the id, name, category, vendor name or brand name of the product.
- 1.2.1.5. The system shall support search with the id, name or location of the vendor.
- 1.2.2.6. The system shall support semantic search based on the context of information, variation of words, synonyms and concept matching.

1.2.2. Filtering

- 1.2.2.1. The system shall provide filtering for the searched products based on:
    - 1.2.2.1.1. Rating of the product
    - 1.2.2.1.2. Price range of the product
    - 1.2.2.1.3. Vendor of the product
    - 1.2.2.1.4. Brand of the product
    - 1.2.2.1.5. Category of the product

1.2.3. Sorting

- 1.2.3.1. The system shall provide sorting the searched products based on:

        - 1.2.3.1.1. Bestsellers
        - 1.2.3.1.2. Newest arrivals
        - 1.2.3.1.3. Prices
        - 1.2.3.1.4. Rating
        - 1.2.3.1.5. Number of comments

1.2.4. Categories

- 1.2.4.1. The admin users shall be able to manage (add/delete/update) product categories.
- 1.2.4.2. The added product shall have a category.
- 1.2.4.3. The system shall support searching for products categorically.

1.2.5. Distinguishability of Products

- 1.2.5.1. Two products are considered different if any of the following is different:
    - 1.2.5.1.1. Names of the products
    - 1.2.5.1.2. Vendors of the products
    - 1.2.5.1.3. Categories of the products
- 1.2.5.2. Each product that is considered different by the criteria mentioned in requirement 1.2.5.1 shall be given a unique ID by the system.

1.2.6. E-Mail Verification

- 1.2.6.1. For registration, an e-mail shall be valid and unique for each user.
- 1.2.6.2. For e-mail verification, a link shall be sent to the users' e-mail account. Users shall be able to validate their e-mail addresses by visiting the link.

1.2.7. Order Page

- 1.2.7.1. The system shall enable customers and vendors to see the status of their orders.
- 1.2.7.2. Status of the orders shall include sufficient information such as but not limited to order date, product price, cargo information and delivery date.
- 1.2.7.3. The system shall enable customers to cancel their active orders.
- 1.2.7.4. The system shall enable vendors to send cancelling requests to admin for cancelling their orders during the processing stage.
- 1.2.7.5. The system shall enable customers to return their delivered orders within 14 days by providing the reason of return.
- 1.2.7.6. The system should enable customers to see the total amount of points they have achieved on the platform by placing orders.

1.2.8. Notification System

- 1.2.8.1. The system shall enable customers and vendors to use the notification system.
- 1.2.8.2. The system shall send a notification to the vendor when the vendor sold a product.

- 1.2.8.3. The system shall enable customers to get notifications about price changes for products that they already set an alarm for.
- 1.2.8.4. The system shall enable customers to set an alarm on the price of a certain product to be notified when the price of the product goes below that price.
- 1.2.8.5. The system shall enable customers to get notifications when a vendor cancels an order.

1.2.9. Recommendation System

- 1.2.9.1. The system shall provide product recommendations to users.
- 1.2.9.2. Recommendation system shall take information such as but not limited to the search history, the products that are viewed and the products that are bought by the users into consideration for recommendations.

1.2.10. Payment Page

- 1.2.10.1. The system shall enable customer users to complete the purchases of the products in their shopping carts after they fill their payment information.
- 1.2.10.2. The system shall support payment via credit card and IBAN.
- 1.2.10.3. The system shall take payment information of the customer to process the payment.
    - 1.2.10.3.1. The system should be able to use the payment information of the customer stored in the system if customer provided any payment information to be stored beforehand.
- 1.2.10.4. The system shall take shipping information of the customer.
    - 1.2.10.4.1. The system should be able to use the shipping information of the customer stored in the system if customer provided any shipping information to be stored beforehand.
- 1.2.10.5. The system shall provide links to the Terms and Conditions and Privacy Policy documents.
- 1.2.10.6. The system shall ask customer to approve the “Terms and Conditions“ and the Privacy Policy.
- 1.2.10.7. The system shall ask customer to confirm the order and proceed with the payment.
- 1.2.10.8. The system shall show the status (successful, pending, failed) of the payment to the customer.

2. Non-Functional Requirements

2.1. Availability

- 2.1.1. The application shall be available as a native web site in browsers.
- 2.1.2. Users shall be able to reach a product by URL.
- 2.1.3. The application shall be available as a native mobile application on Android platforms.
- 2.1.4. The application shall be deployable on a remote and manually configurable server.

- 2.1.5. The application shall be dockerized, to make the development and deployment process easier.
- 2.1.6. The application shall support the Turkish and English characters.

2.2. Security

- 2.2.1. The system shall use the HTTPS protocol to transfer encrypted data over the web.
- 2.2.2. To ensure that sensitive information shall be unreadable to everyone but the destination server, the system shall use SSL Certificates.
- 2.2.3. To quickly identify potential hacking activity the website shall be monitored periodically.
- 2.2.4. The system shall be secure against SQL injection and cross-site scripting.
- 2.2.5. The system shall regularly scan for malware.

2.3. Performance

- 2.3.1. The system shall respond to up to 1000 users without a crash.
- 2.3.2. The system shall respond to each user in less than 2 seconds.

2.4. Privacy

2.4.1. The system shall follow the W3C Activity Stream Protocol.

2.4.2. The system shall follow the standards of Wide Web Consortium.

2.4.3. Personal data shall be processed in a manner that ensures GDPR and KVKK requirements.

2.4.4. The system shall seek, consent to use personal data in research.

8. All Design Documents

8.1. Scenarios & Mockups

Scenario 1 - Buying a Product

Barış Mutlu

Persona

- 22 years old
- Computer Engineering student
- Intelligent(:)) and funny
- Agile
- Loves playing football

Story

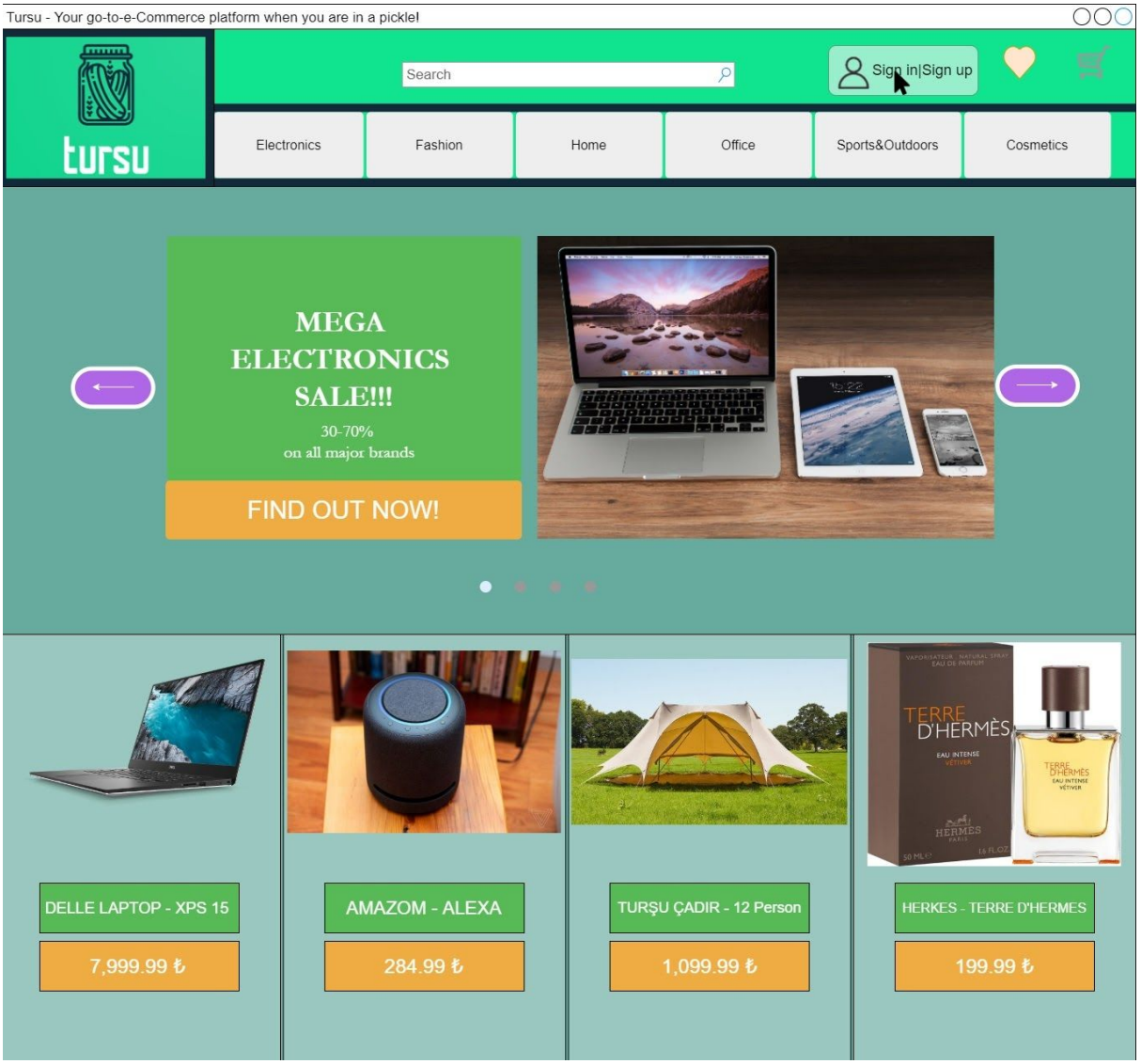
Later this week Barış has a football match with his engineering student friends. Since he plays so much, his shoes had worn out and he needs a new pair now. So, by a recommendation of his friend, he decided to try tursu.com and search for a new pair of shoes there.

Preconditions

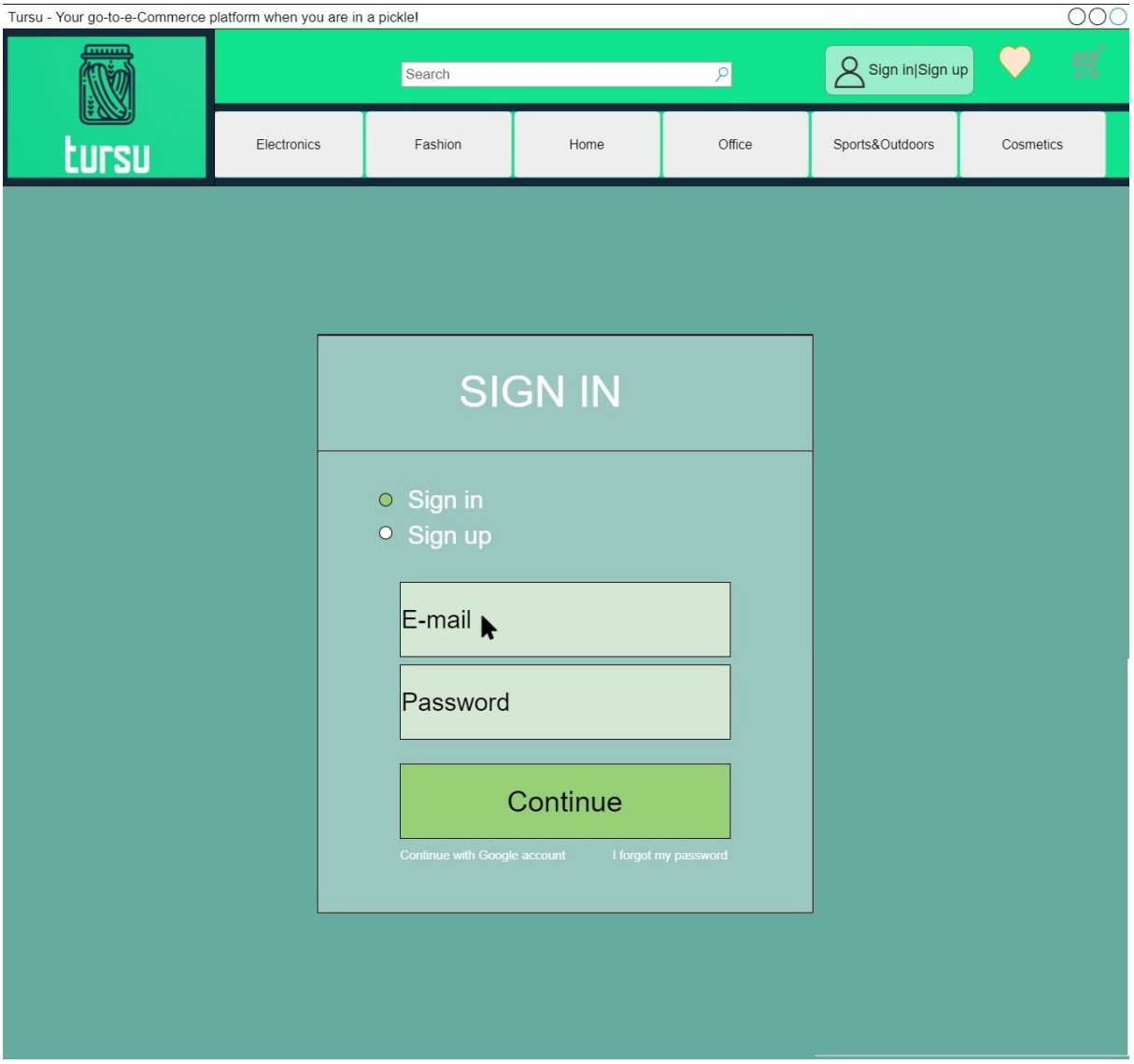
- Barış must already have signed up to the platform.
- His user type must be customer.

Actions

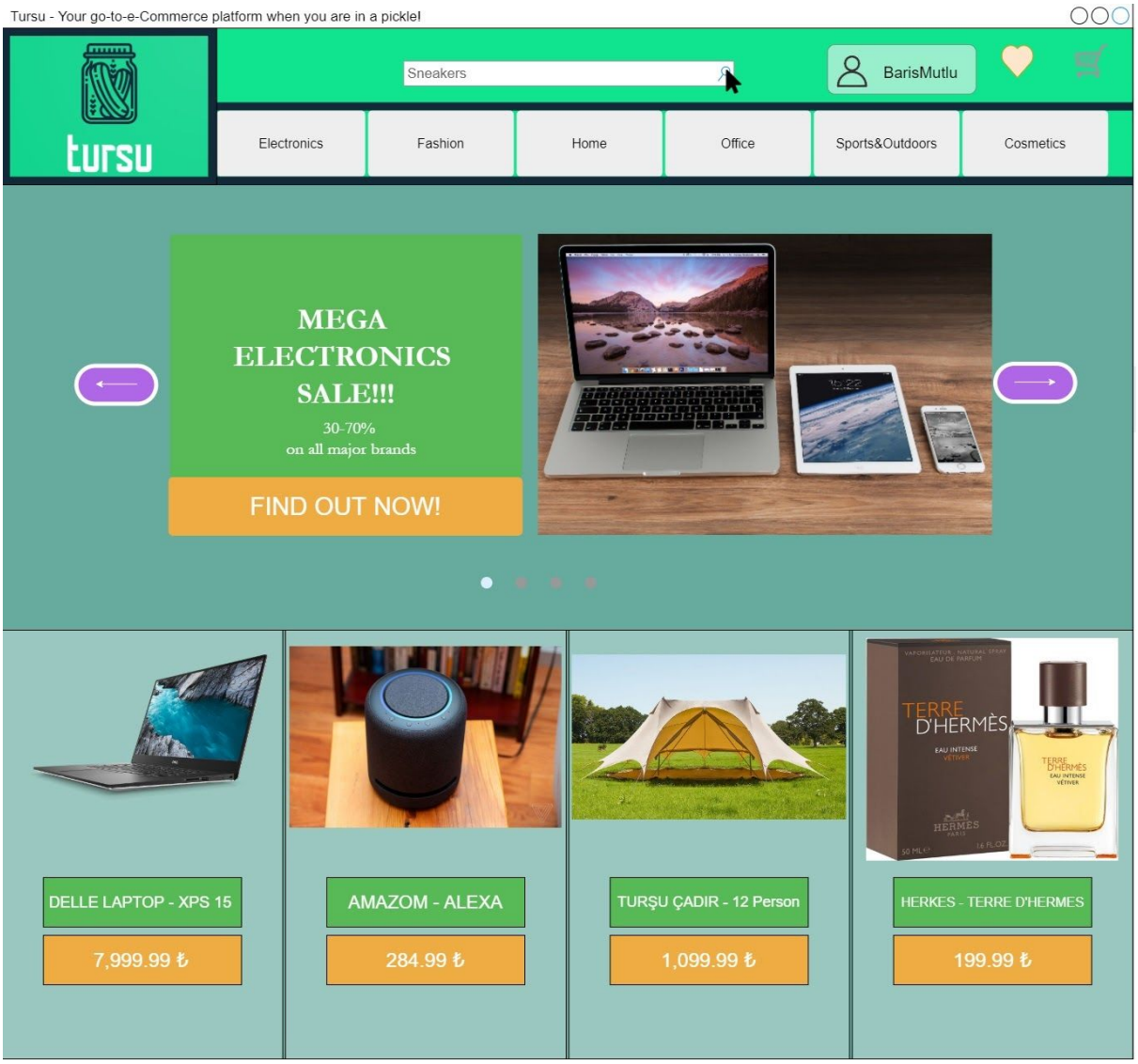
1. Barış visits tursu.com and clicks the "Sign in" button.



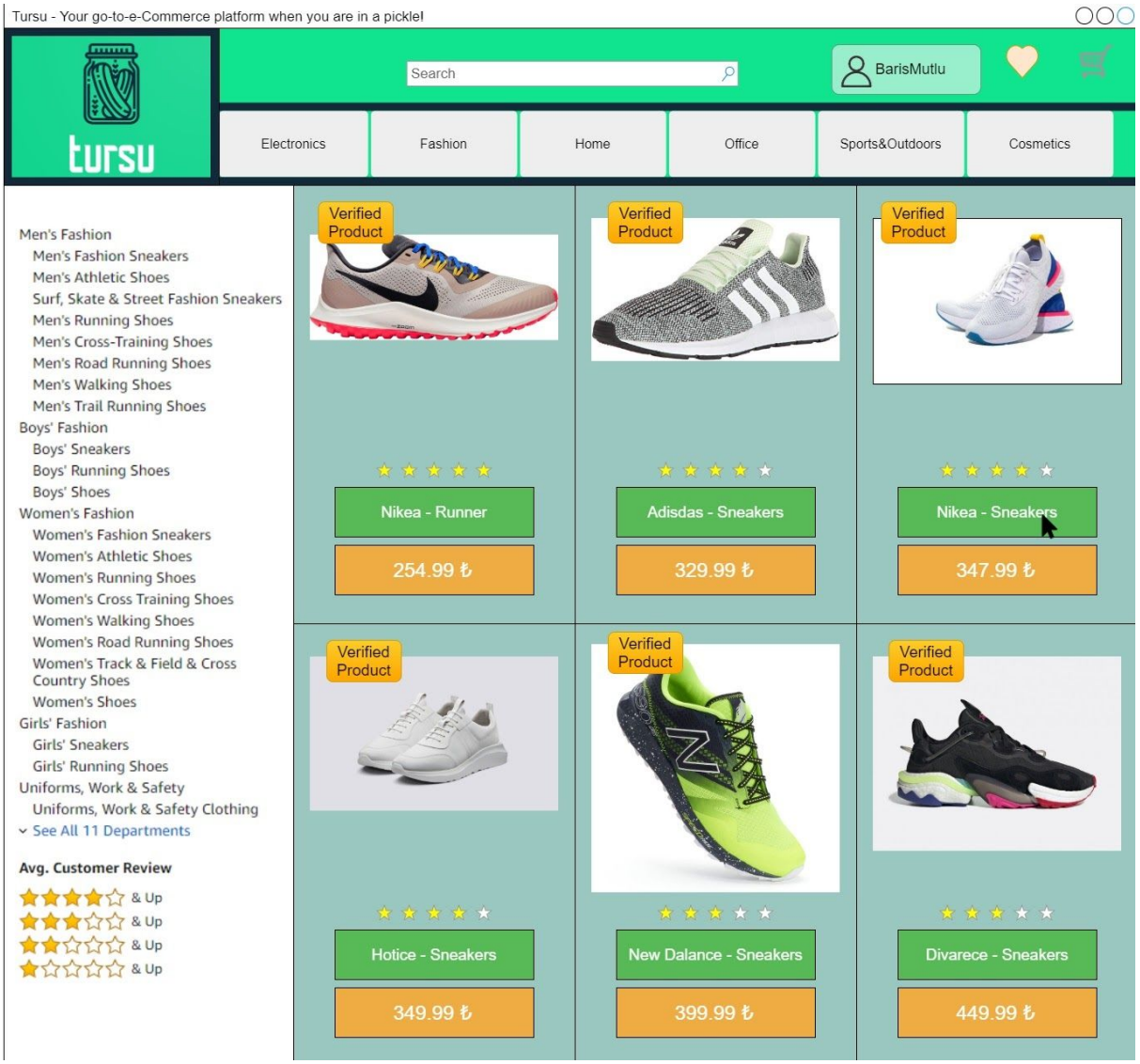
2. He encounters the sign in page and fills in his user information.



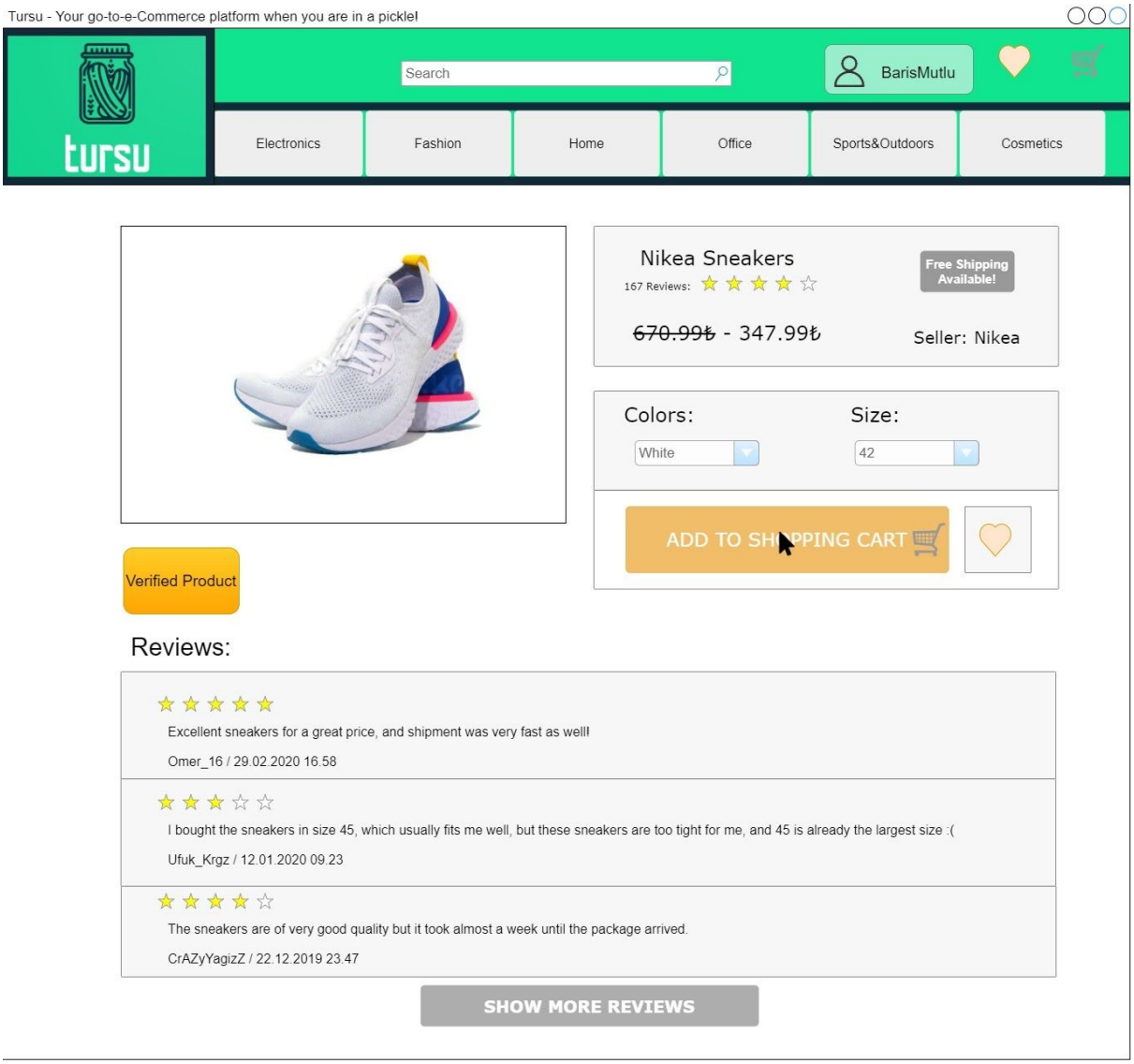
3. He is back in the home page and searches for a pair of sneakers.



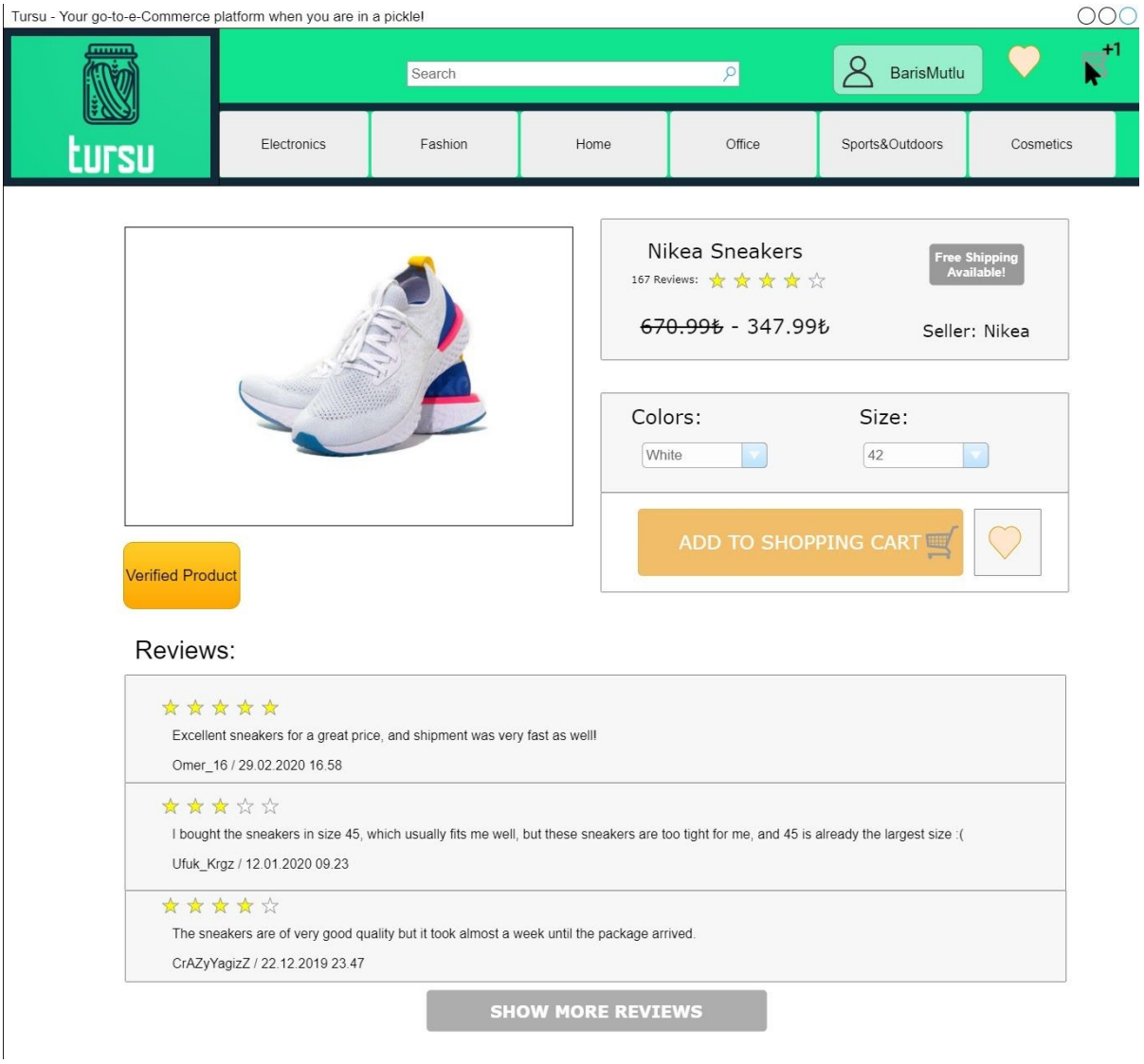
4. He sees several sneakers and decides to click on one based on its rating, its price and his liking.



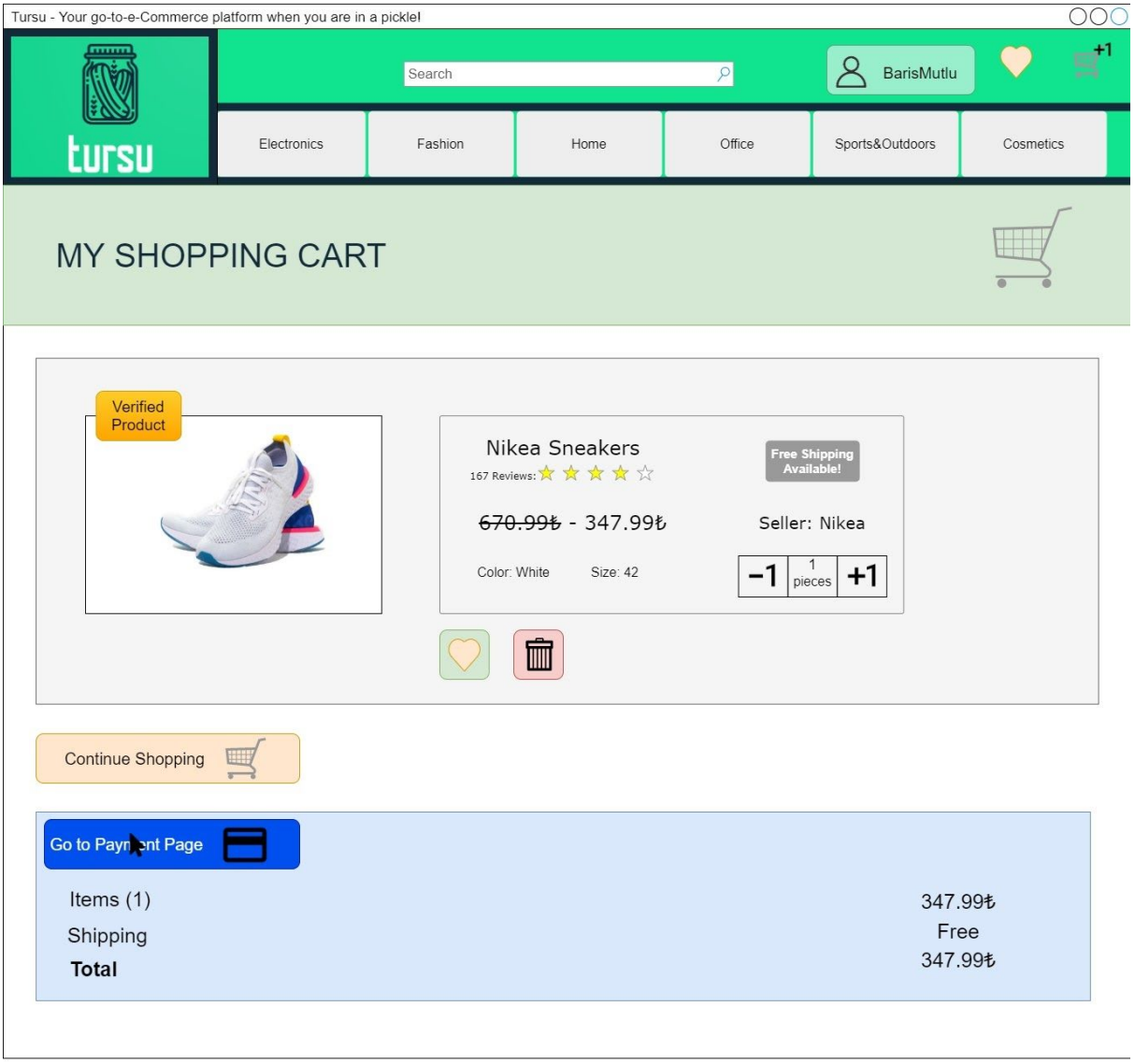
5. He reads the comments of the product and sees his shoe-size is in the stock. He decides to buy the sneakers and adds the sneakers to his shopping cart.



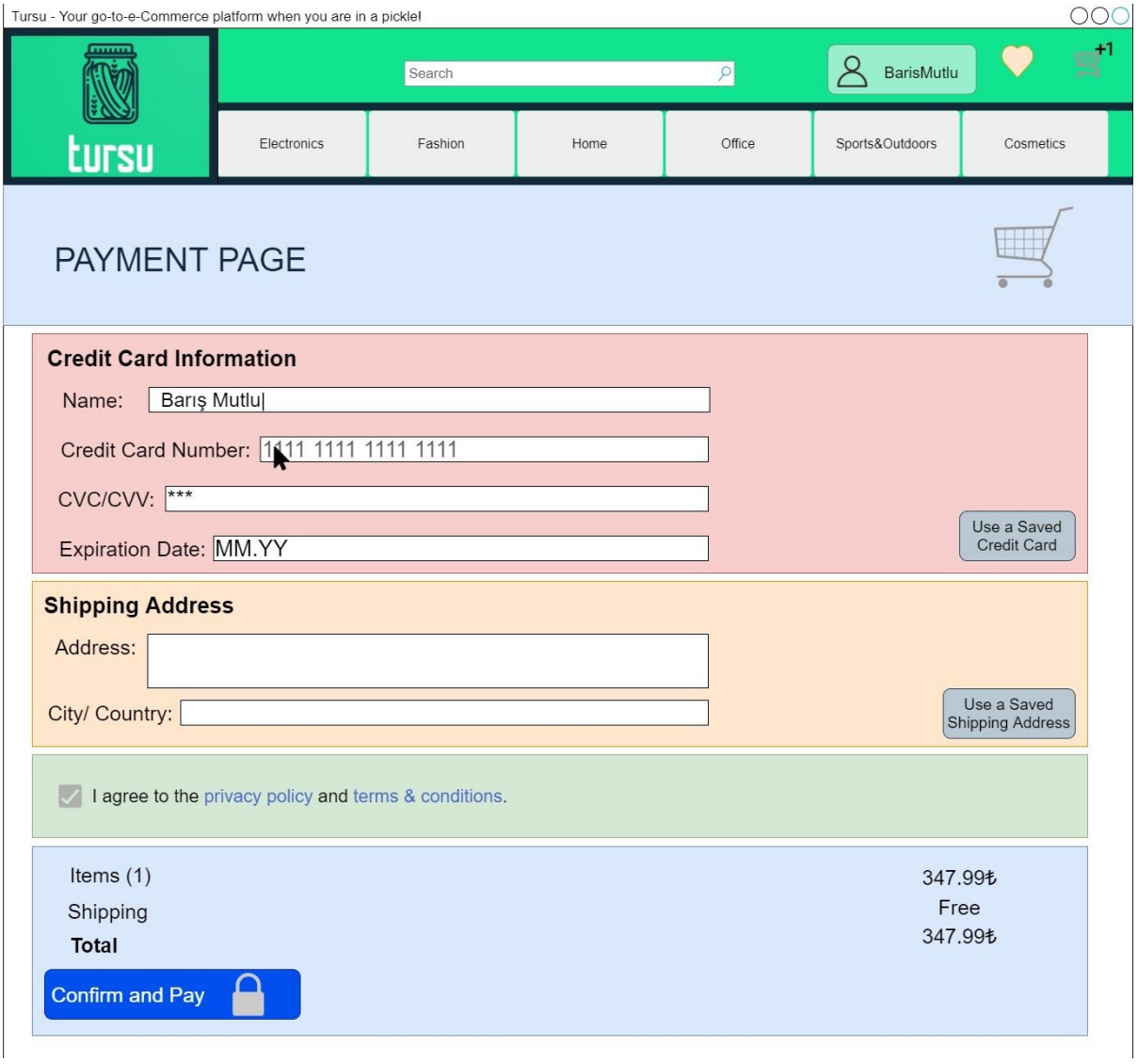
6. He clicks the shopping cart button to complete his purchase.



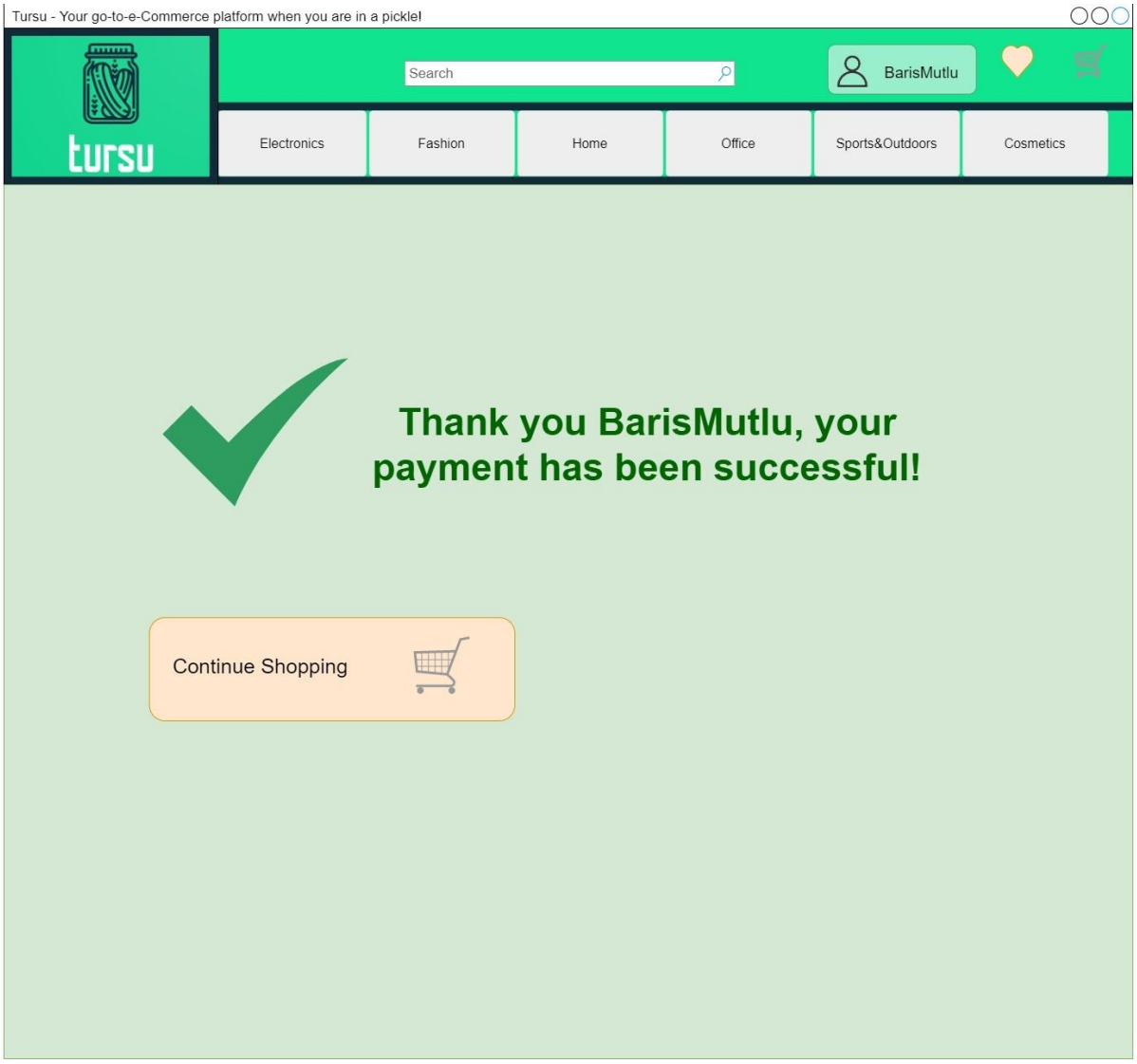
7. He clicks the "Go to Payment Page" button.



8. He encounters the payment page and fills in his credit-card information and shipping address. He also has to agree to the privacy policy (GDPR and KVKK) and the terms & conditions, which he can read by clicking on the links.



9. He sees the "Payment Successful" page.



Barış is happy with his new shoes.

Acceptance Criteria

1. Already registered users shall be able to sign in with their email and password. (1.1.2.1.)

2. The system shall provide searching for products and vendors. (1.2.1.1.)

3. Customers shall be able to see these rates when making a purchase decision. (1.1.5.6.)

4. Customers shall be able to read user comments about products (1.1.5.2.)

5. Customers shall be able to add products to their shopping carts to order.. (1.1.6.1.)

6. The system shall enable customer users to complete the purchases of the products in their shopping carts after they fill their payment information. (1.2.10.1.)

Scenario 2 - Cancelling an Order

Afife Şensoy

Persona

- Owner of an anonymous textile company
- 65 years old
- Housewife
- Energetic
- Ambitious
- Wants to keep up to date
- Skilled in needlework
- Wants to use her free time and make money

Story

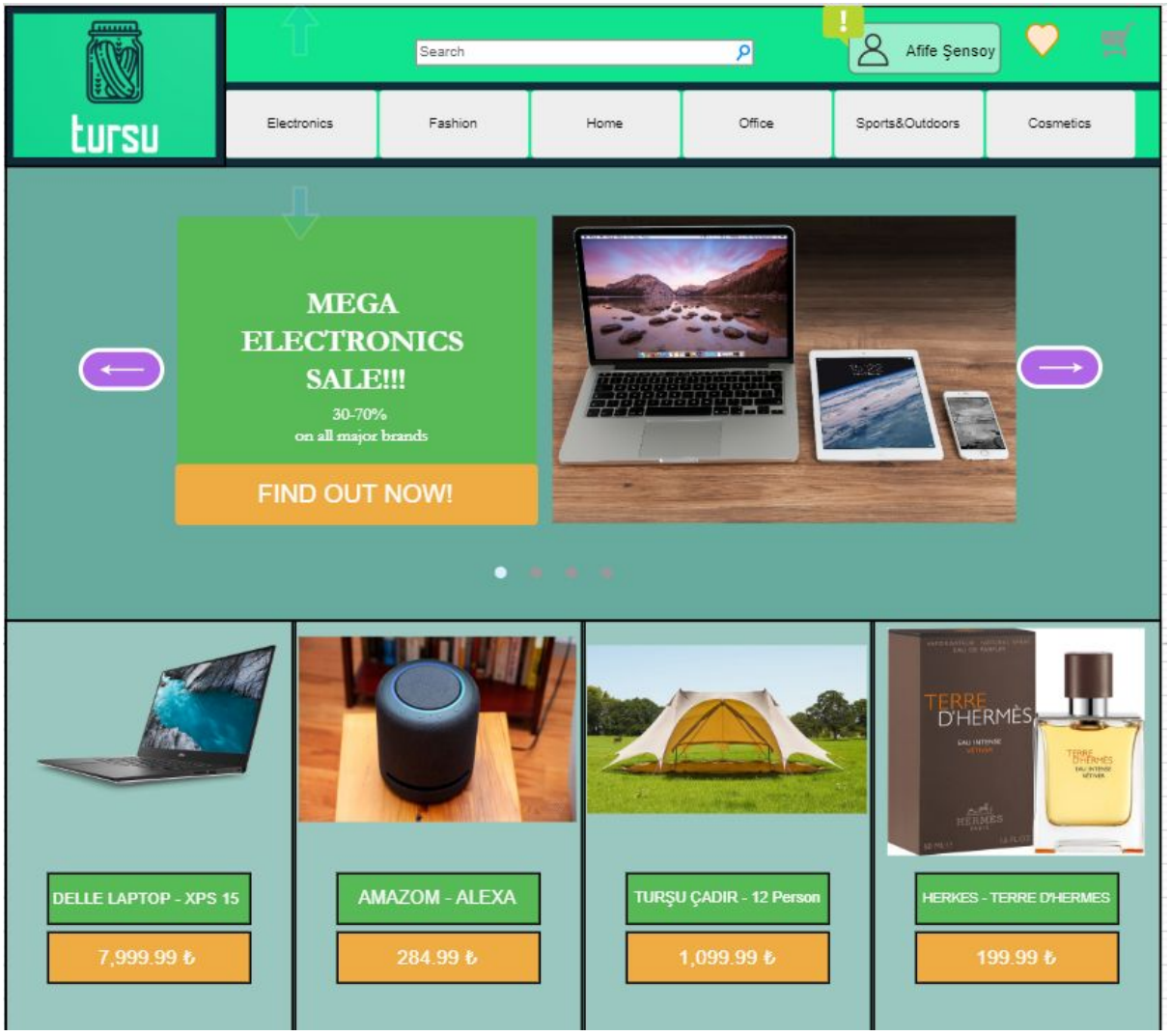
All of Afife Şensoy's children left their village home to work in İstanbul. Afife feels bored. She is an energetic person and looking for something to keep her busy and make her feel useful. On Bayram holiday one of her grandsons gifted her, his old computer to able to communicate with each other via skype. With her grandson's help and ambition, she learns the basics of computer in no time. She sews her grandson a scarf as a thank you gift. Grandson loves the scarf and encourages her to start a business. The two build an anonymous textile company. Grandson advises our easy-to-use website. Afife looks forward to this opportunity to evaluate her time and earn some money in the meanwhile.

Preconditions

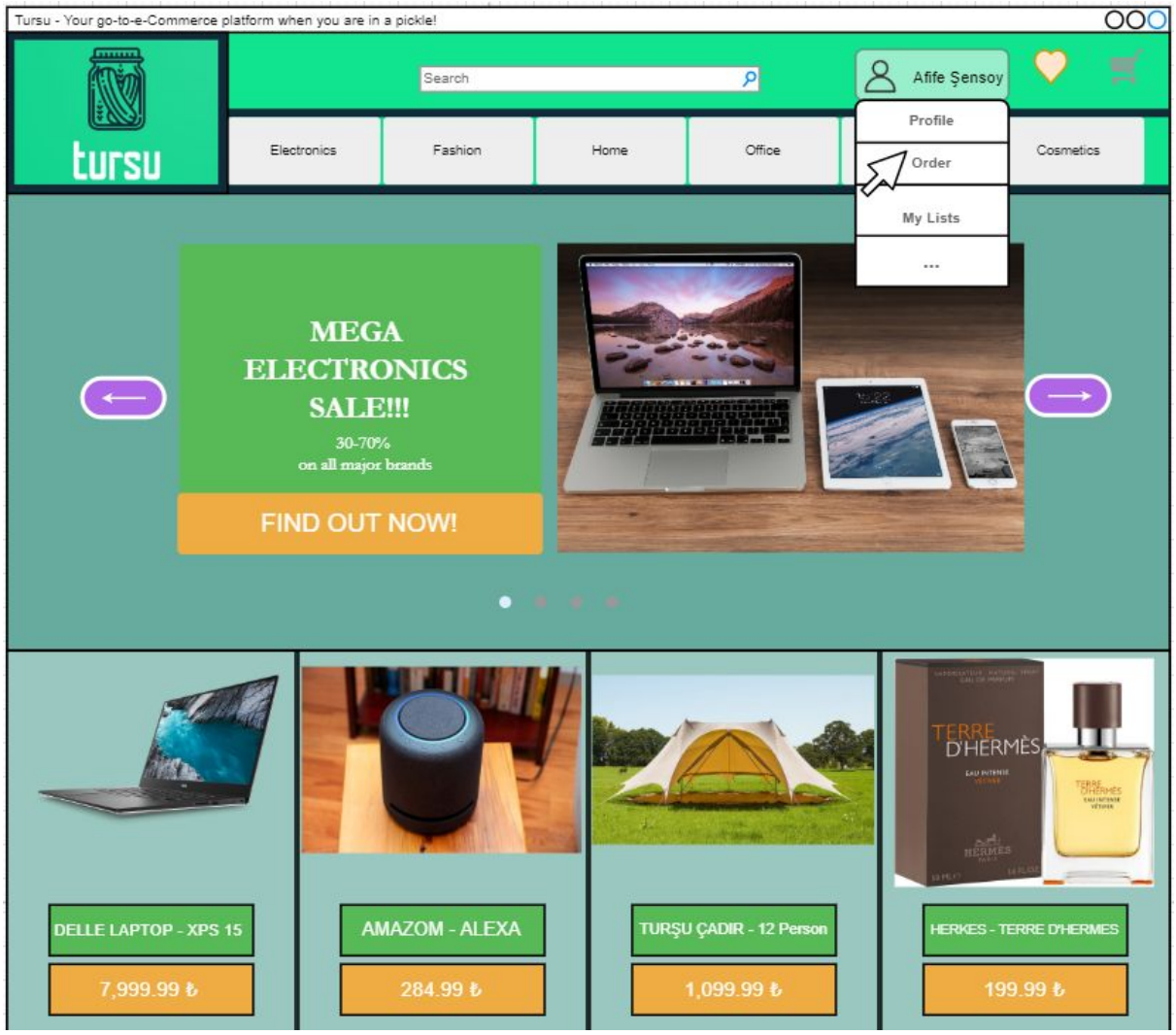
- She has already built her company for first-hand products.
- She signed up via her Google account.
- She already defined her account as a vendor provided her IBAN and specified the location via Google Maps in registration.
- She has already sold a couple of products.
- She has some products added and ready to be sold.

Actions

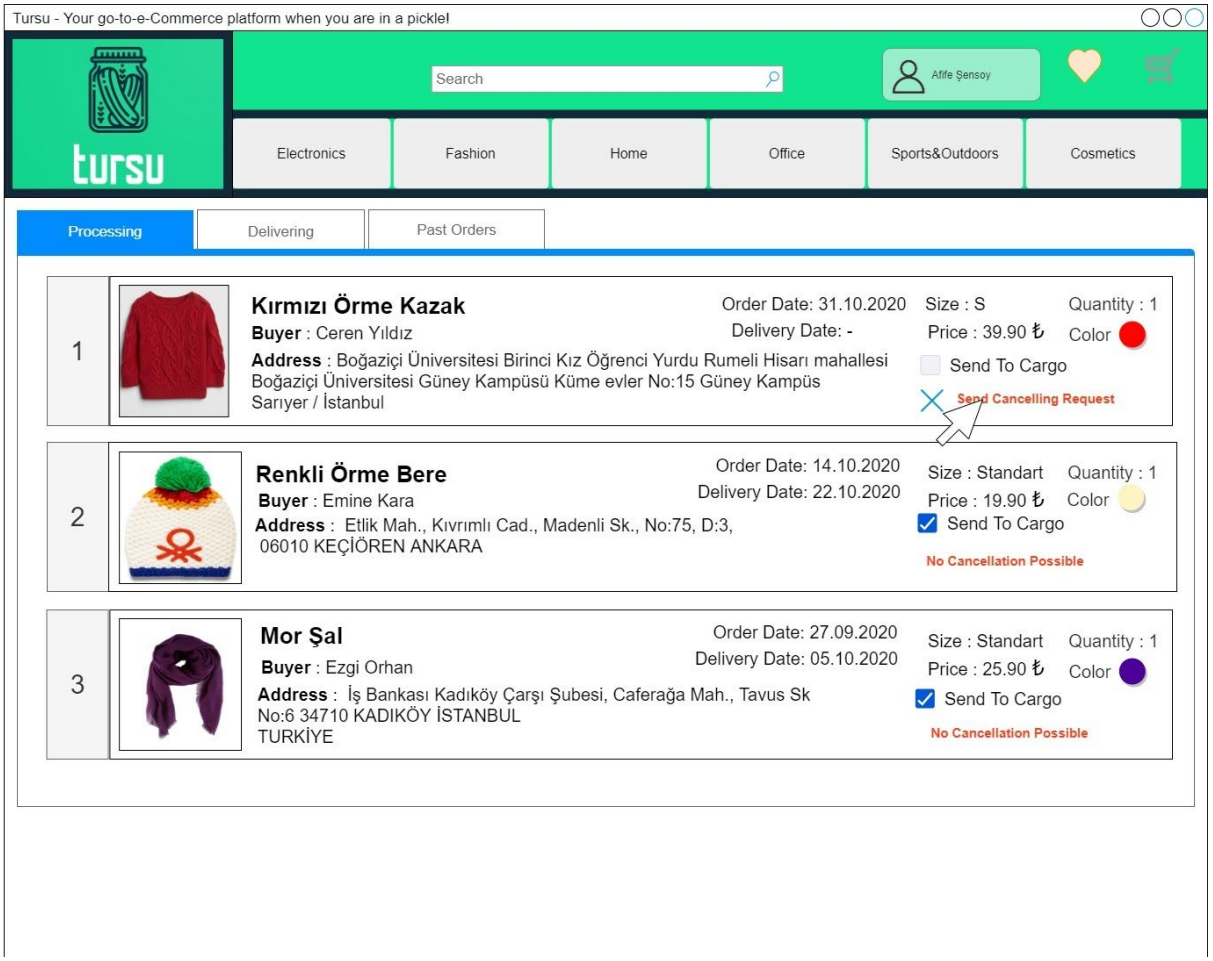
1. Afife realizes that she has a notification about her products.



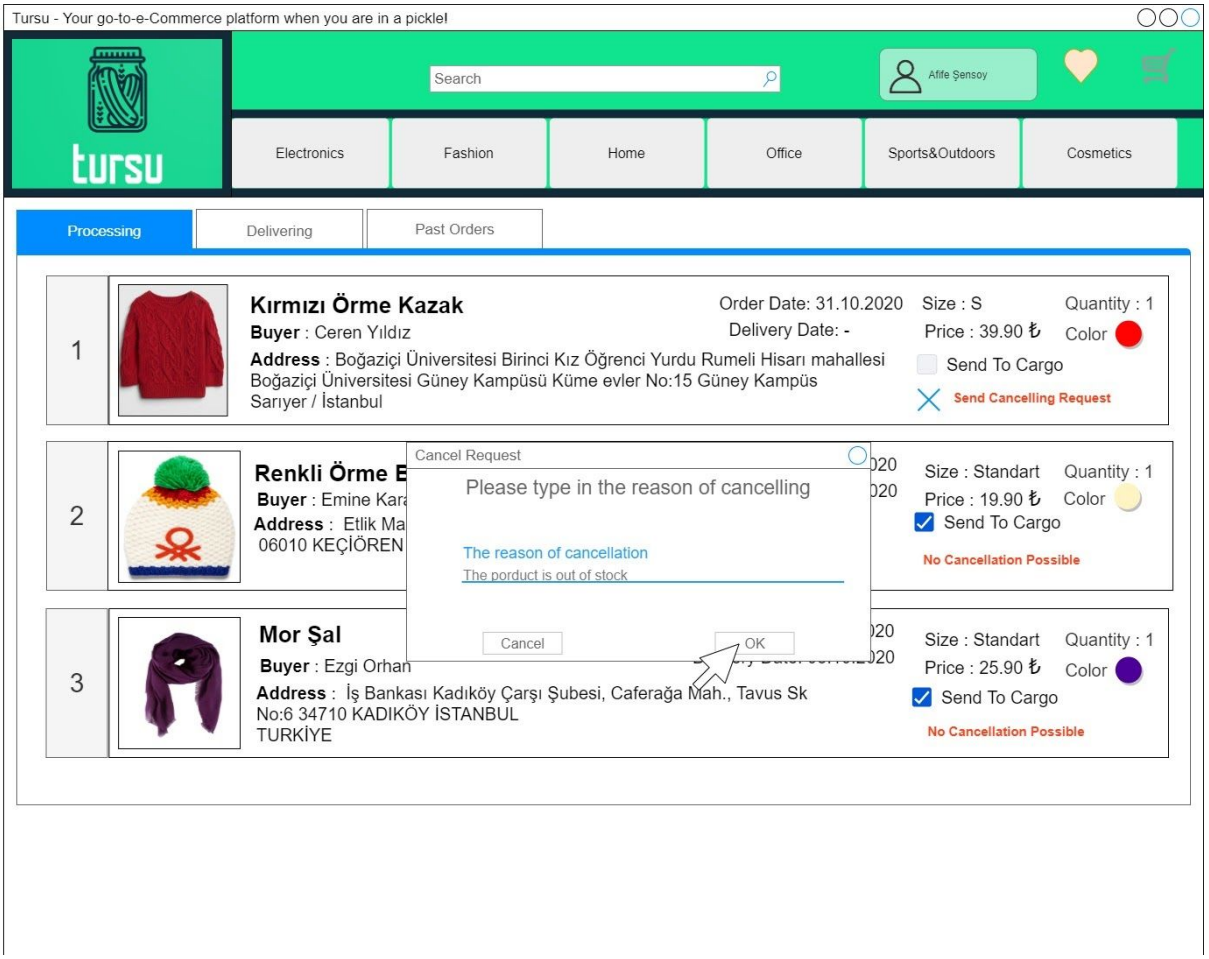
2. Afife clicks to checkout her orders.



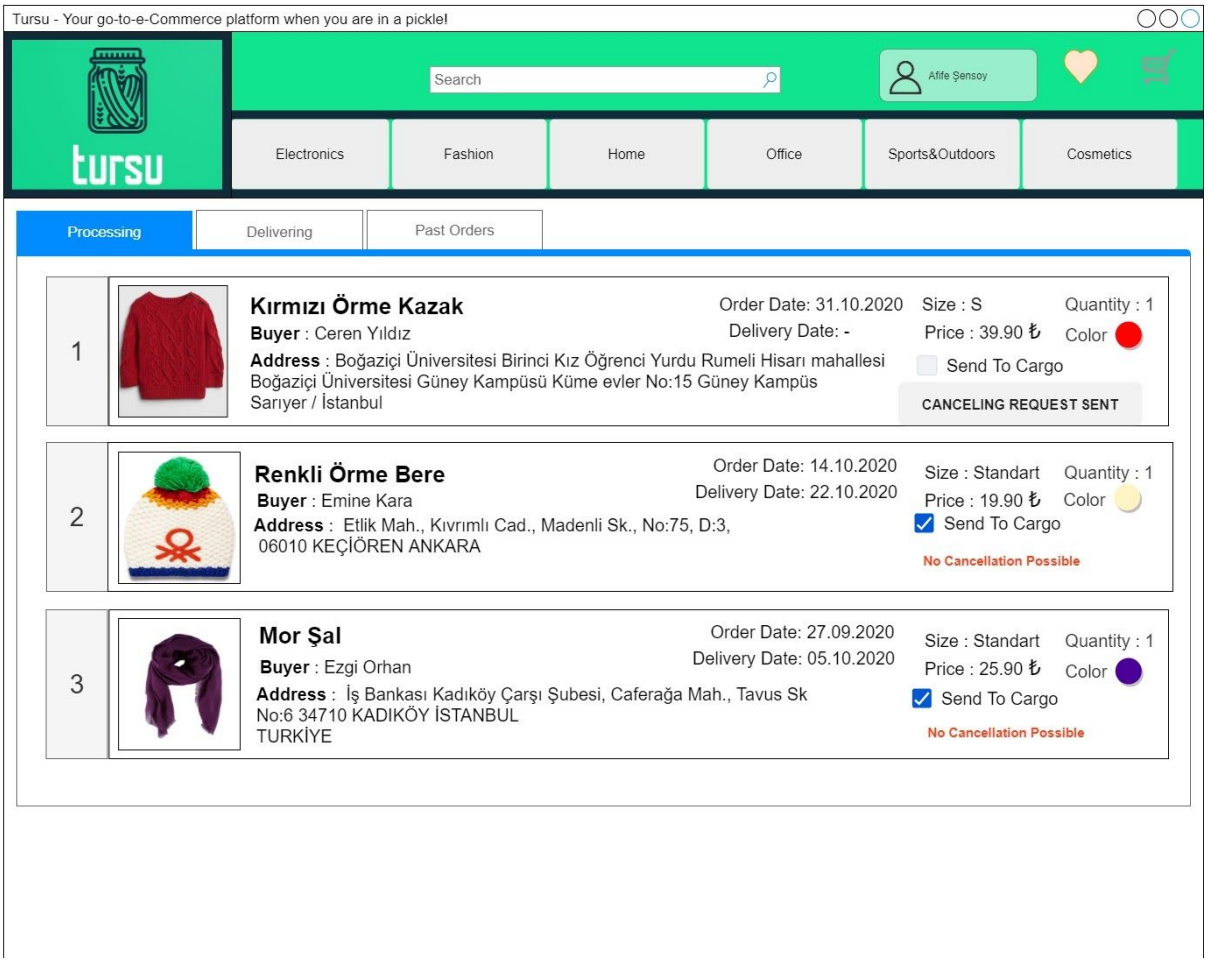
3. She realizes that the ordered product is not in the stock right now. And she decides to request a cancellation of the order by clicking the 'Send Cancelling Request' button.



4. After she pushed the button a pop-up screen shows up. She writes the reason of the cancellation and clicks "Ok".



5. Finally, she sees that the cancellation request for the order is sent.



Acceptance Criteria

1. Registered Users shall provide information about whether they are a vendor or a customer while making registration. (1.1.1.3.)

2. Vendor users additionally shall provide: their IBAN and the location of their store through Google Maps in registration. (1.1.1.6.1. and 1.1.1.6.2.)

3. Vendors shall be able to communicate with admins about a certain order. (1.1.11.2.)

4. The system shall enable vendors to send cancelling requests to admin for cancelling their orders during the processing stage. (1.2.7.4.)

5. The system shall send a notification to the vendor when the vendor sold a product. (1.2.8.2.)

Scenario 3 - Make Comment About a Delivered Product

Ahmet Bardakçı

Persona

- 45 years old
- Banker
- Bored after many years of work
- Expects peace in life
- Recently divorced
- Likes home cooking
- Fast adapter to technology

Story

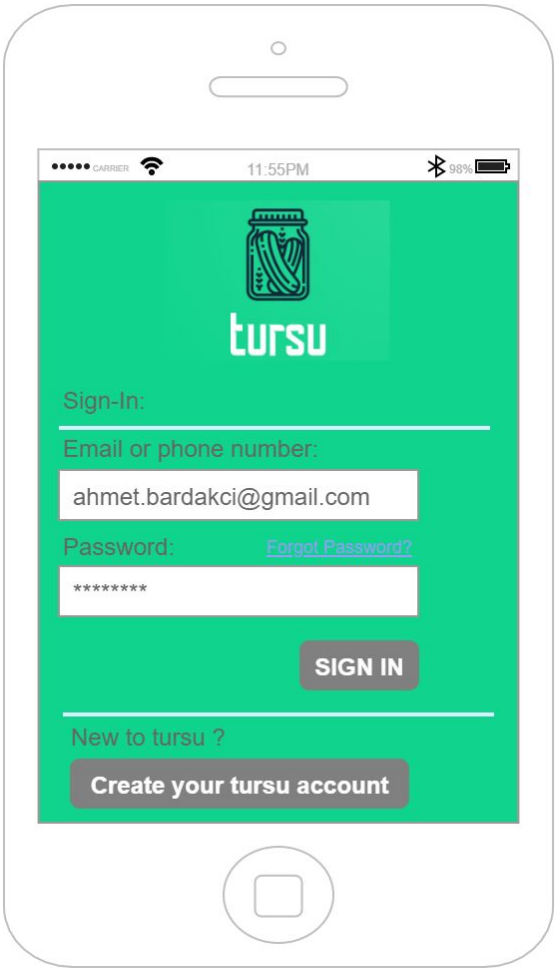
Ahmet is a banker who belongs to upper middle class. He enjoys homemade foods a lot. He have always wanted to live in the countryside, but because of his profession he got stuck in Levent/Istanbul. He likes to try different flavors. And, one day, from one of his friends he heard that some website is selling homemade pickles and he wanted to give it a chance. So, he bought 3 cans of Pickle and he liked them a lot. Now, he wants to leave some feedback to the product to share his experience.

Preconditions

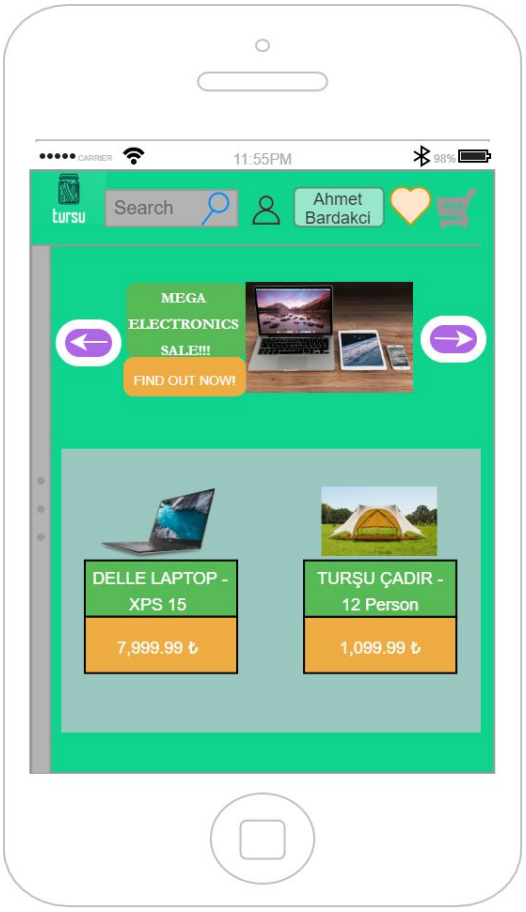
- Ahmet has already downloaded the app to his phone.
- Ahmet has already signed up.
- Ahmet has already bought the product.
- Ahmet has already got the product and tasted it.

Actions

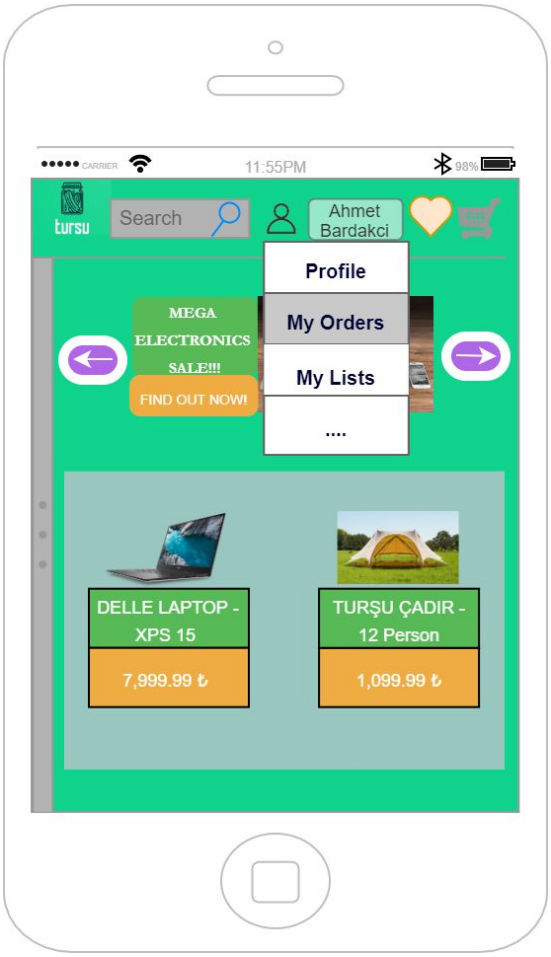
1. Ahmet opens the application and types his credentials to login.



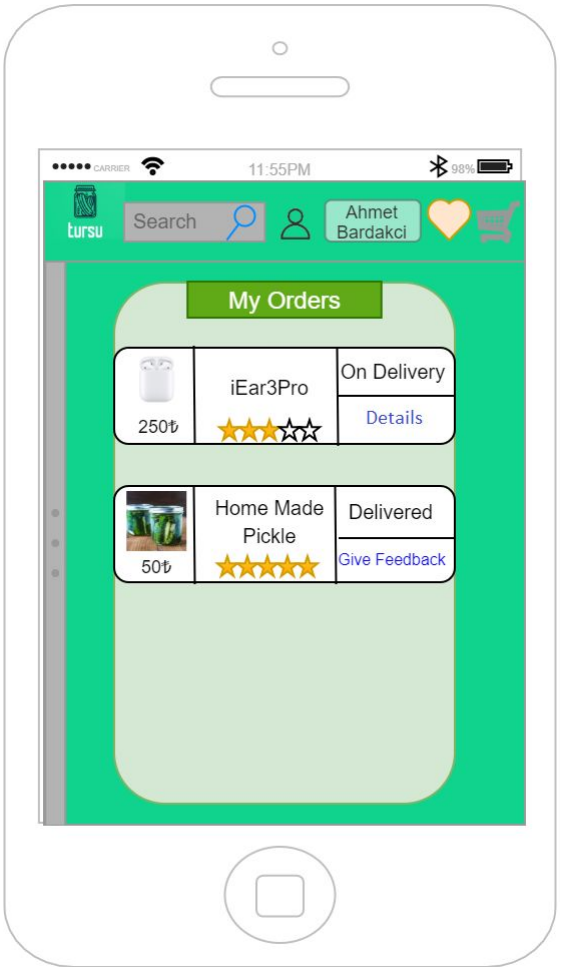
2. He is now in the main page. He opens his user bar by clicking the menu icon.



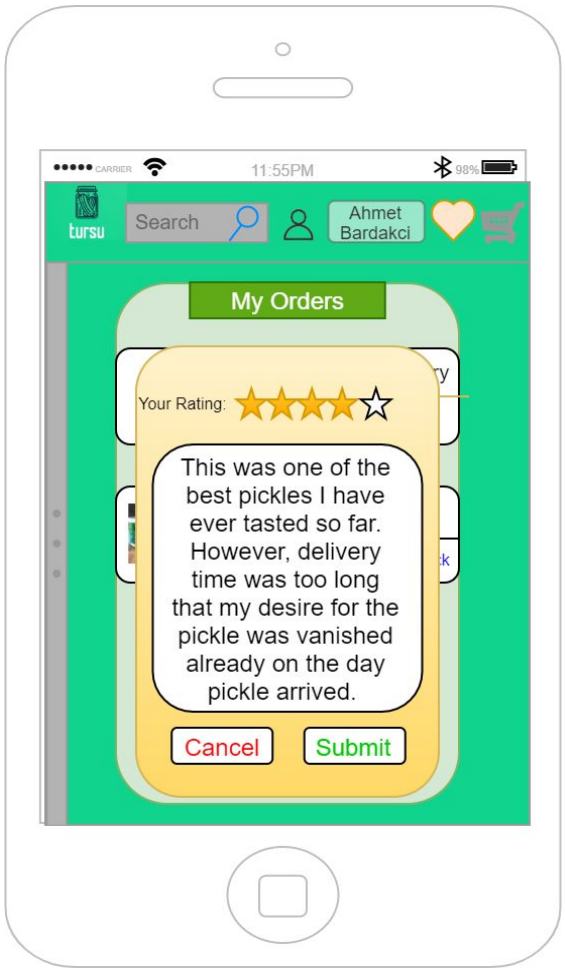
3. In the pop-up bar, he clicks to the my orders section.



4. In the current page, he clicks to give feedback button of his Pickle order.



5. Feedback page pops-up. He fills the form and rates the product(4/5). Finally, he submits



it.

Acceptance Criteria

1. Already registered users shall be able to sign in with their email and password. (1.1.2.1.)

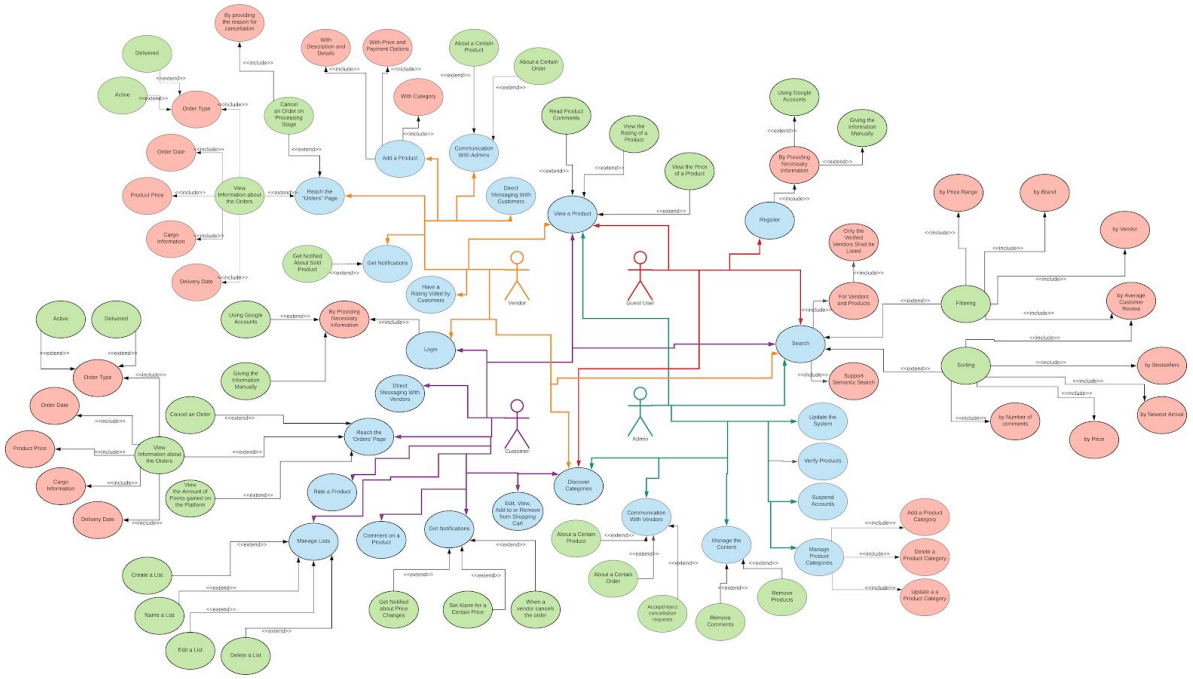
2. The system shall enable customers and vendors to see the status of their orders. (1.2.7.1.)

3. Customers shall be able to review their products after a transaction has been completed. (1.1.5.4.)

4. Customers shall be able to rate a product from 1 to 5. (1.1.5.5.)

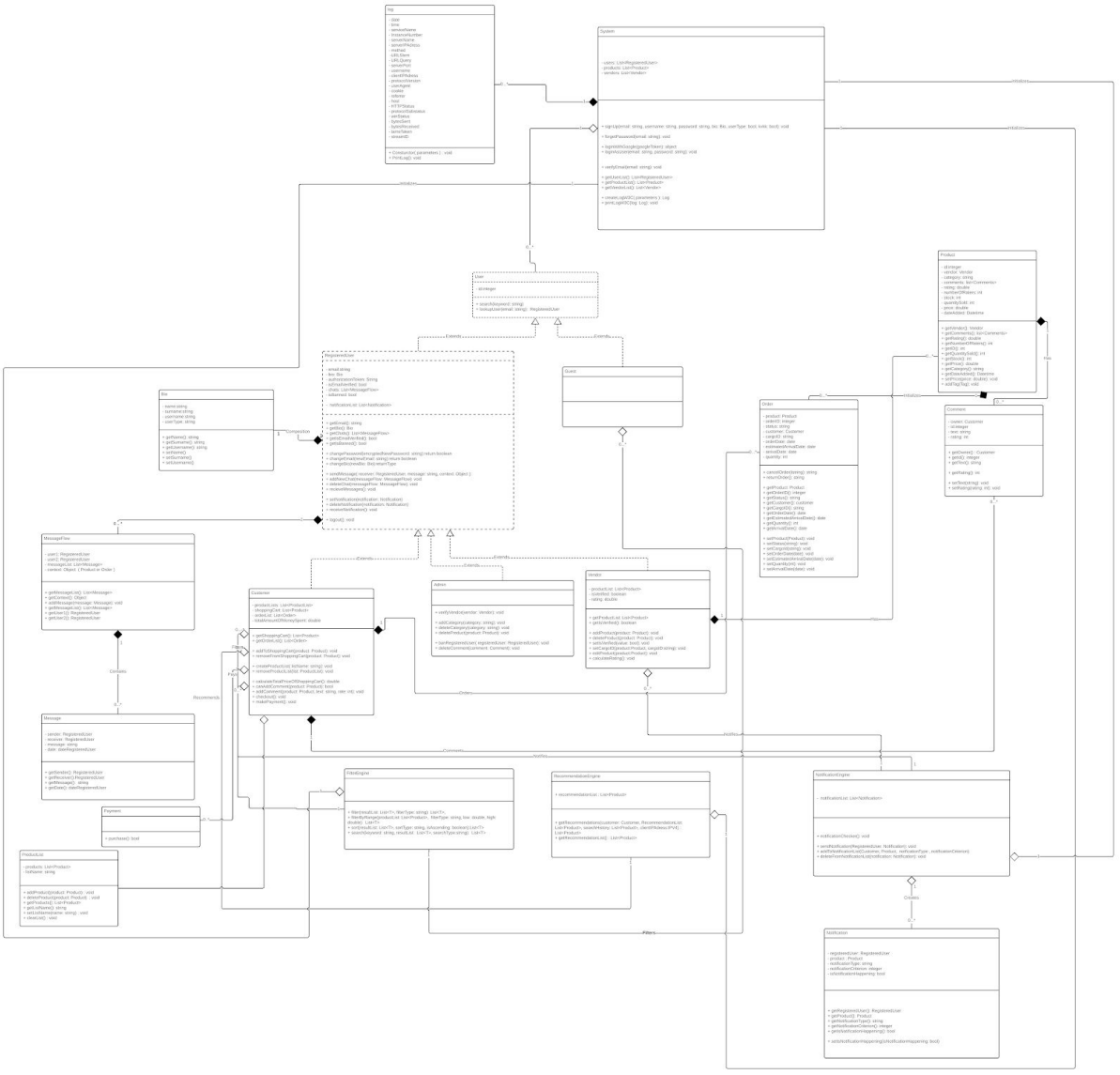
5. Customers shall be able to comment on the products they have purchased. (1.1.5.1.)

8.2. Use Case Diagram



https://github.com/bounswe/bounswe2020group1/wiki/Use-Case-Diagram

8.3. Class Diagrams



https://github.com/bounswe/bounswe2020group1/wiki/Class-Diagram

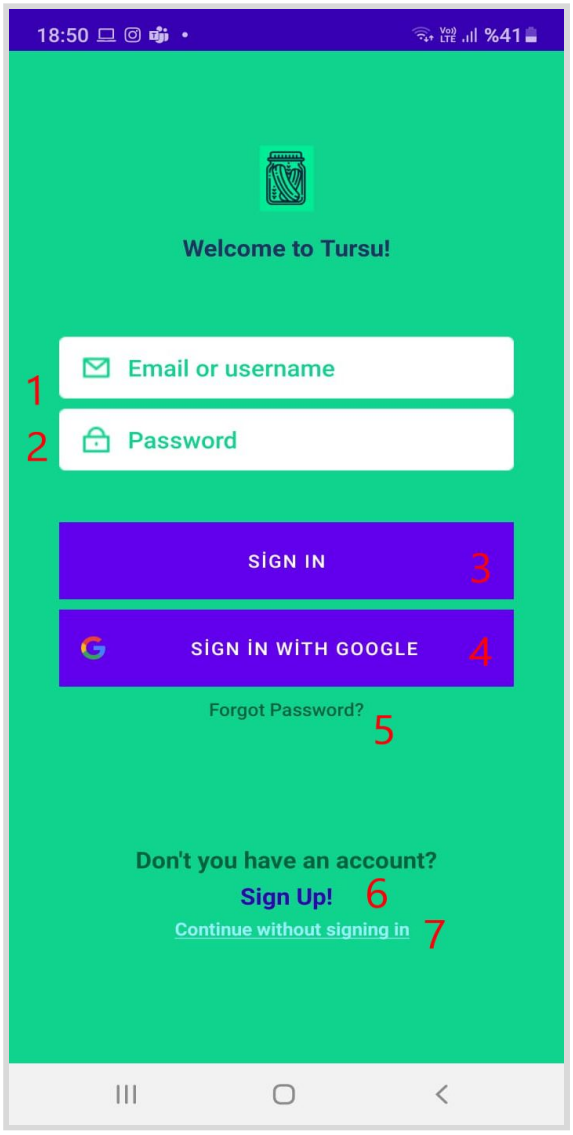
8.4. Sequence Diagrams

https://github.com/bounswe/bounswe2020group1/wiki/Sequence-Diagram

9. User Manual

9.1. Android

-Customer :

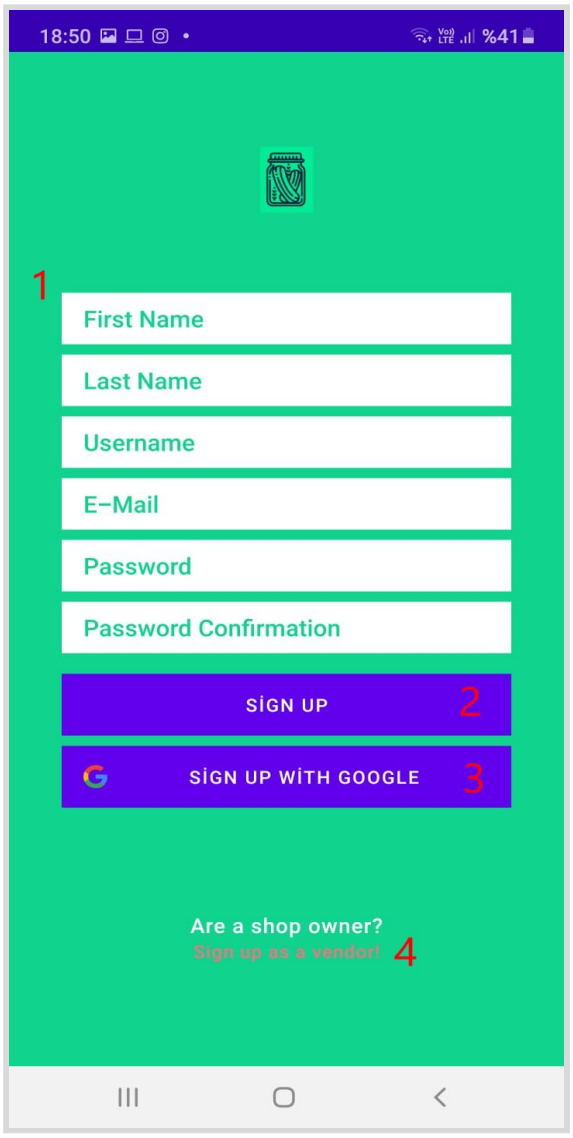


Sign In Page
When you are signing in you should enter an email [1] and a password [2] and continue with the Sign In button
Or you can sign in with a Google account [4]

If you forgot your password, you can change your password [5]

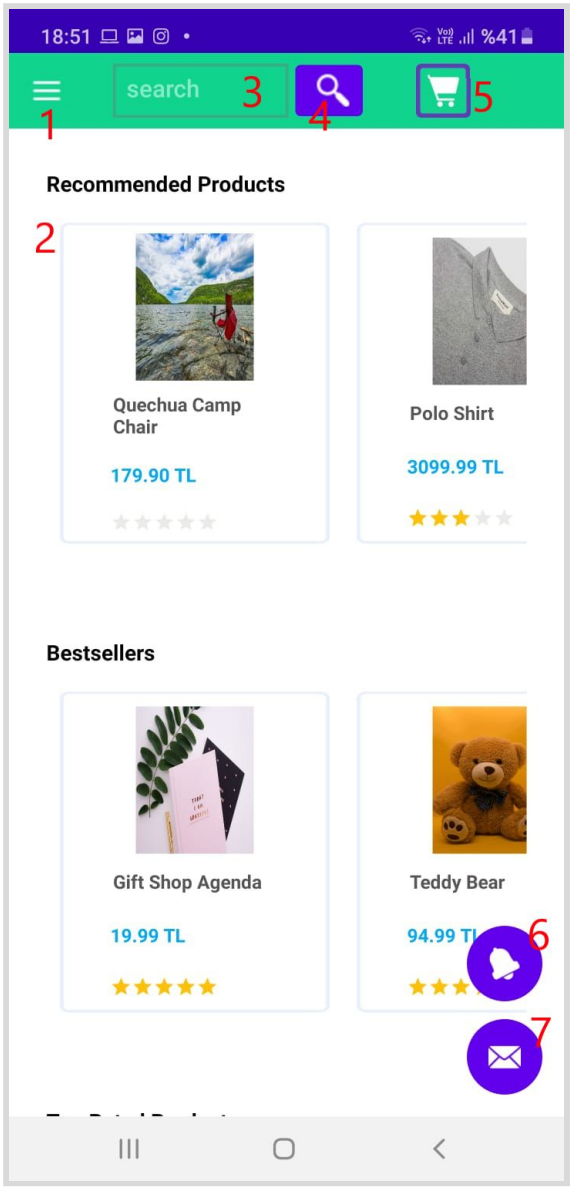
If you don’t have an account, you can sign up via Sign Up button [6]

Finally, you can enter the application as a guest user. [7]



Sign Up Page
when you click to Sign Up, you are directed to this page.
You should fill the necessary information [1] and click to Sign Up [2].
Or you can Sign up via Google [3].

To continue to sign up as a vendor, you can click to “Sign up as a vendor”[4]

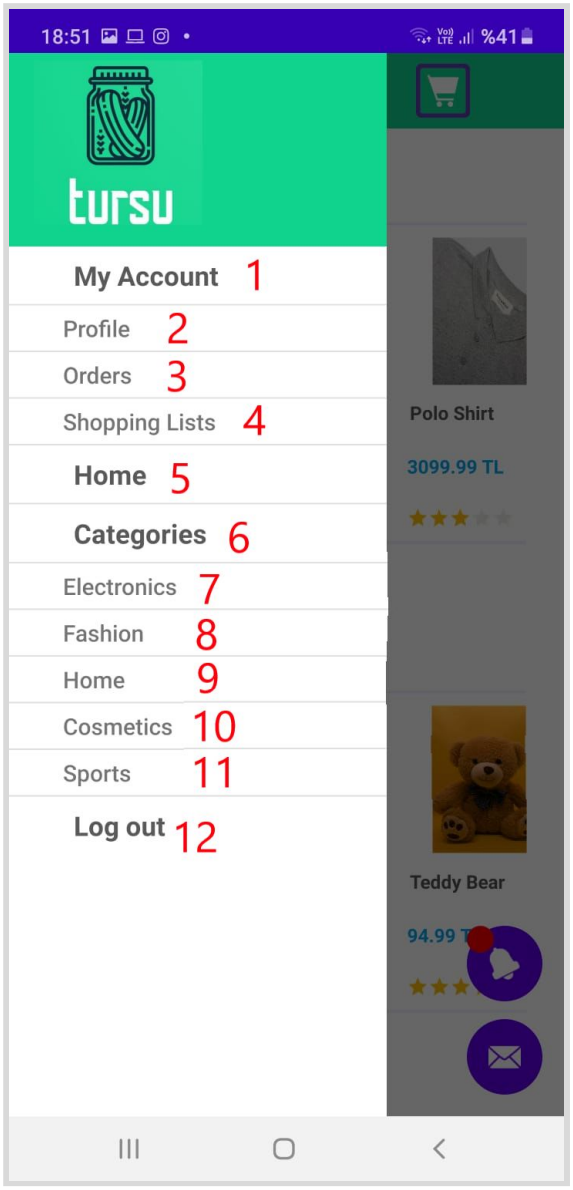


Home Page
After you sign in, you are in the home page where recommended, top rated, bestseller and newest products are listed [2]

There is a side menu that you can use to make transition between pages. [1]

There is a search bar for you to search a desired query (vendor or product) [3] after clicking the search button [4]

You can also go to your shopping cart [5], notifications [6] and messages [7] from the home page.



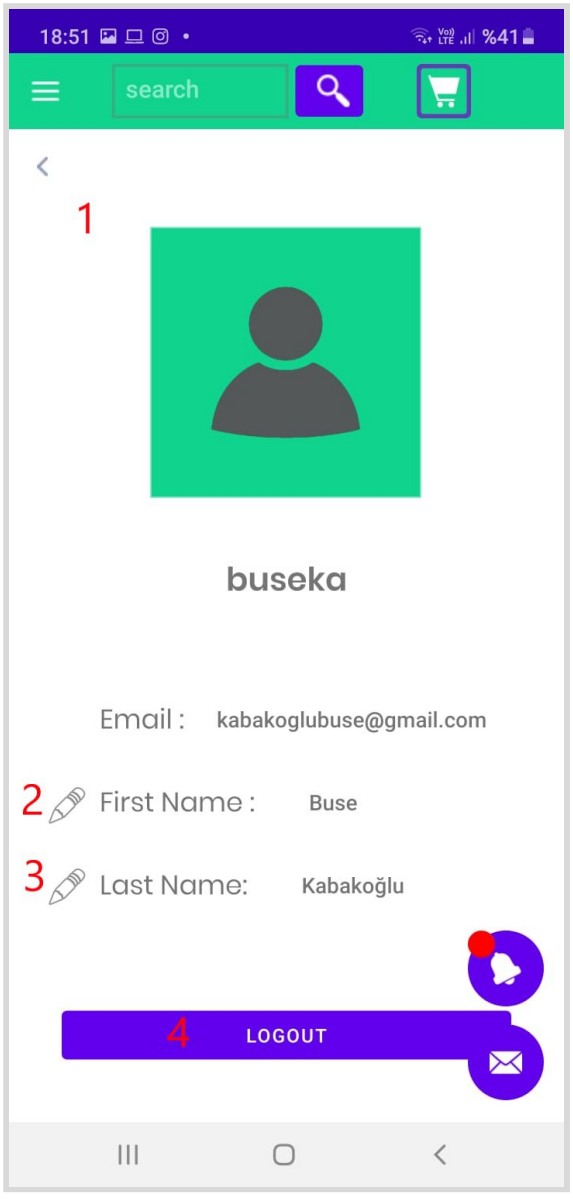
Side Menu
Side menu is used to make transitions between different pages.

You can go to your Profile[2], your orders[3] or created shopping lists[4] from My Account[1]

You choose a category[6] from Electronics[7], Fashion[8], Home[9], Cosmetics[10] or Sports[11].

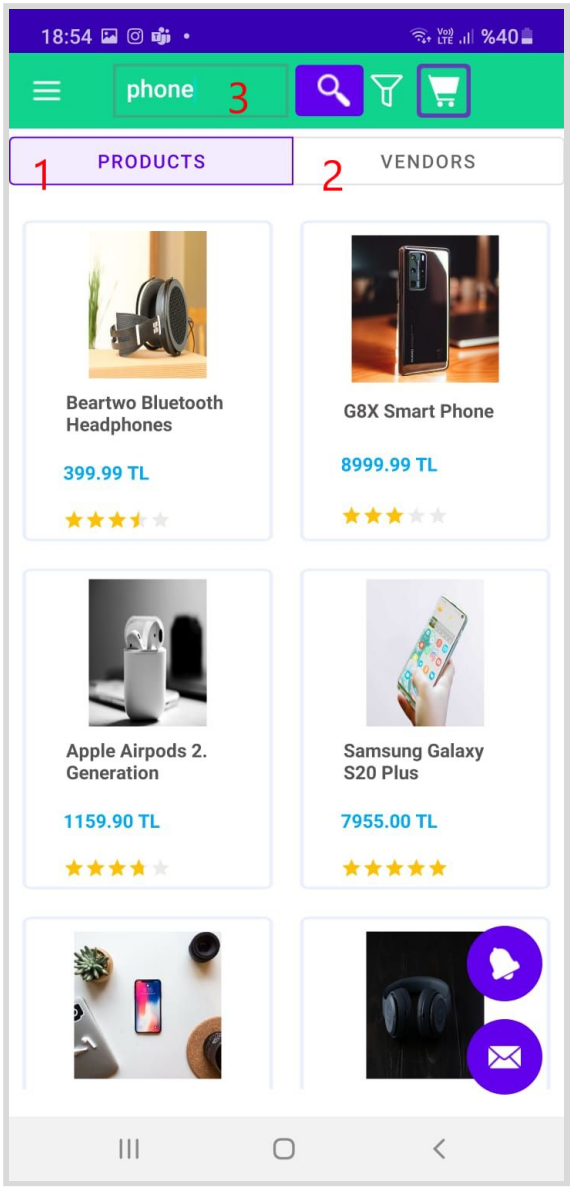
You can log out from the application[12]

Or you can go to the Homepage[5] where your recommended products are listed



Profile page
By clicking Profile, you will be able to see your profile info[1] and change some of the information[2, 3]

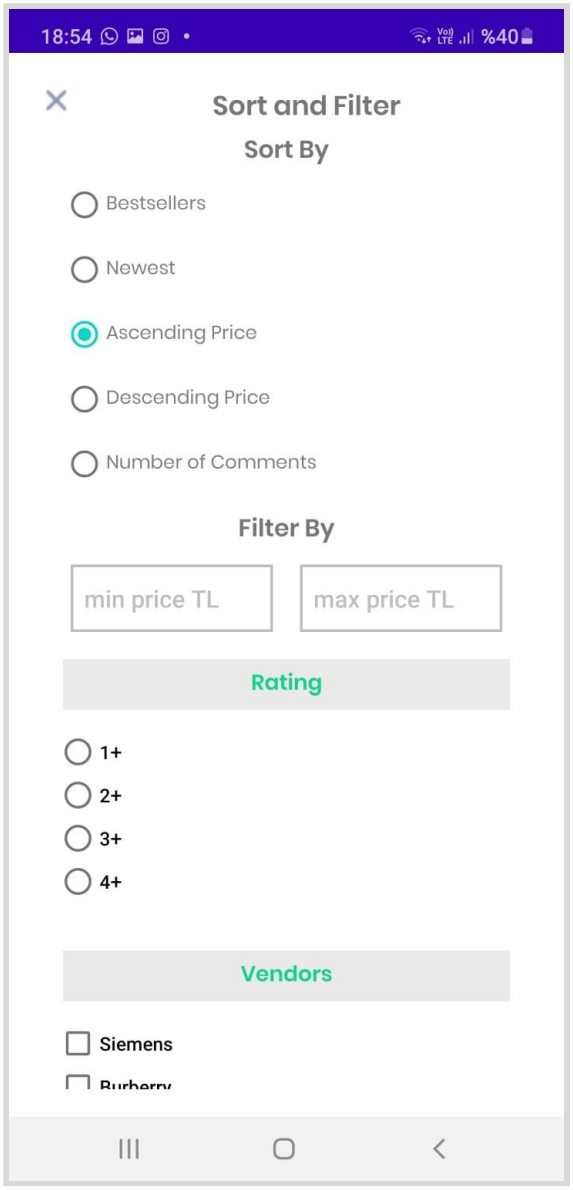
You can also logout[4] from the application on the profile page



Searching a Product

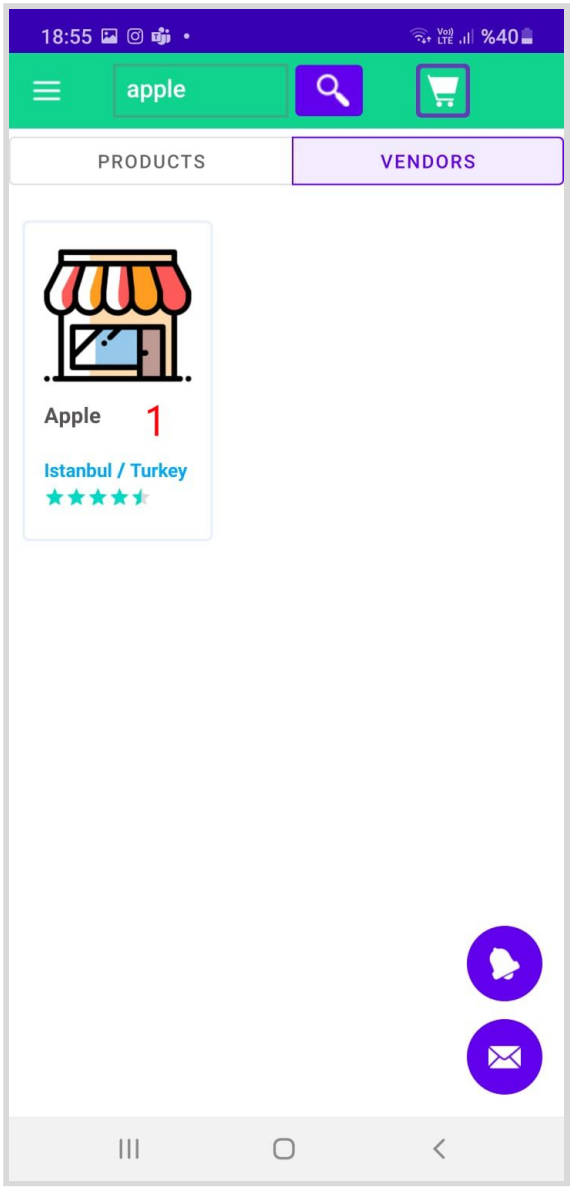
By writing a text in the search bar[3], you can search any product which is in the database.

By using the toggle buttons at the top [1,2], you can both search a vendor or a product.



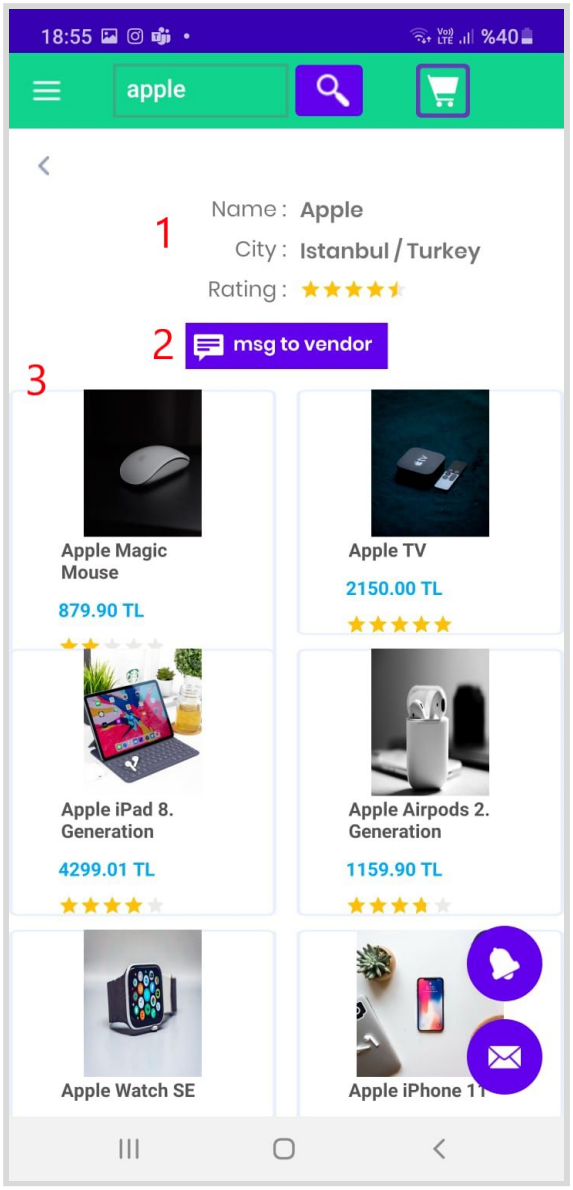
Filtering

After searching a product, you use filter icon to add filters or sorting criteria to your search



Search a Vendor

When vendor toggle button is clicked, you will able to see searched vendors and displaying their public pages by clicking on the items[1]



Displaying Public Vendor Page

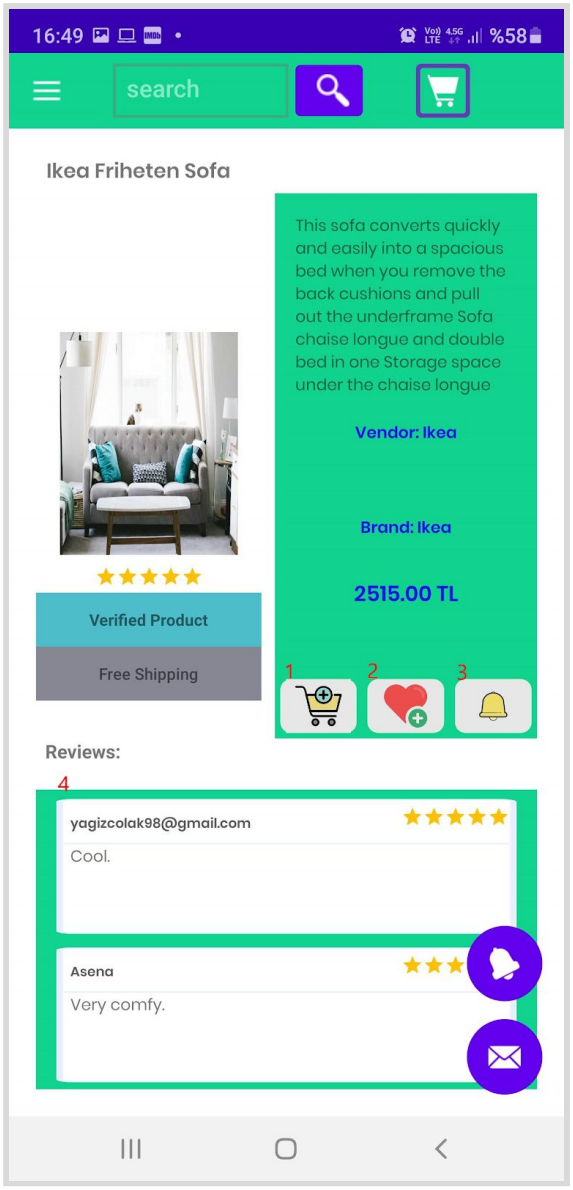
When displaying a vendor page, you will able to display the information and the rating at the top[1]

All the products of the vendor are listed below[3]. You go to the product detail page by clicking on them.

You can also start a chat with the vendor[2]

Messages with vendors
From the message at the button right bottom, you will be able to display your chats

Chat
After clicking on a vendor, the chat page opens that you can send your messages and get responses in real time



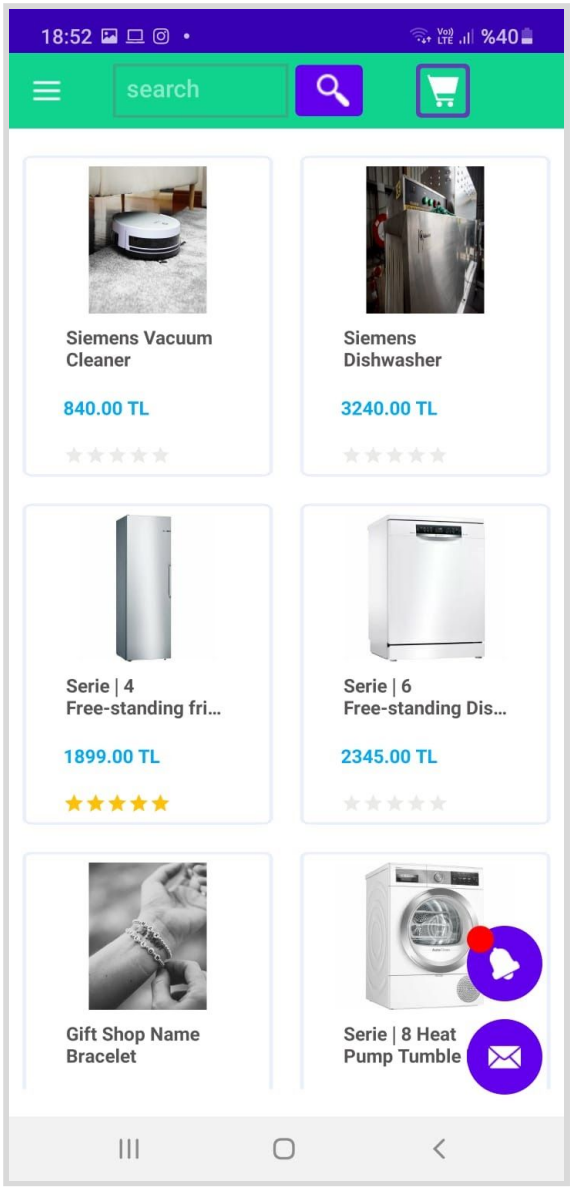
Product Detail Page
In the detailed page of a product(you can open it by clicking on the product from any page.), you can review the details and ranking of the product.

You can add the product to your shopping cart [1]

You review comments[4] and leave a comment if you already bought it.

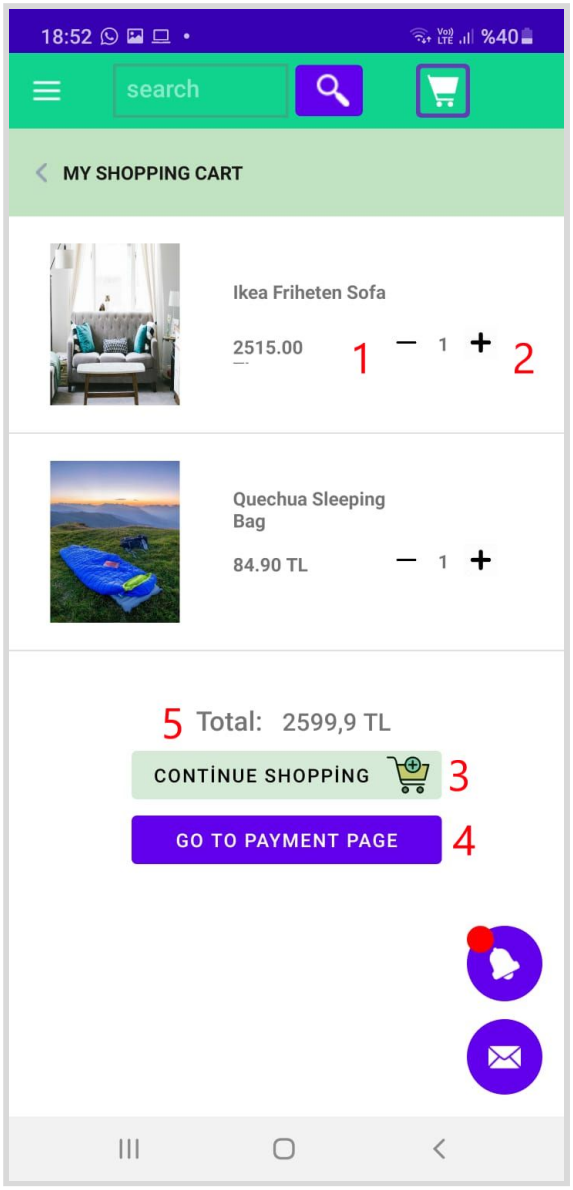
You can add the product to one of your shopping lists[2]

You can set an alert for its price or stock changes.[3]



Displaying Categories

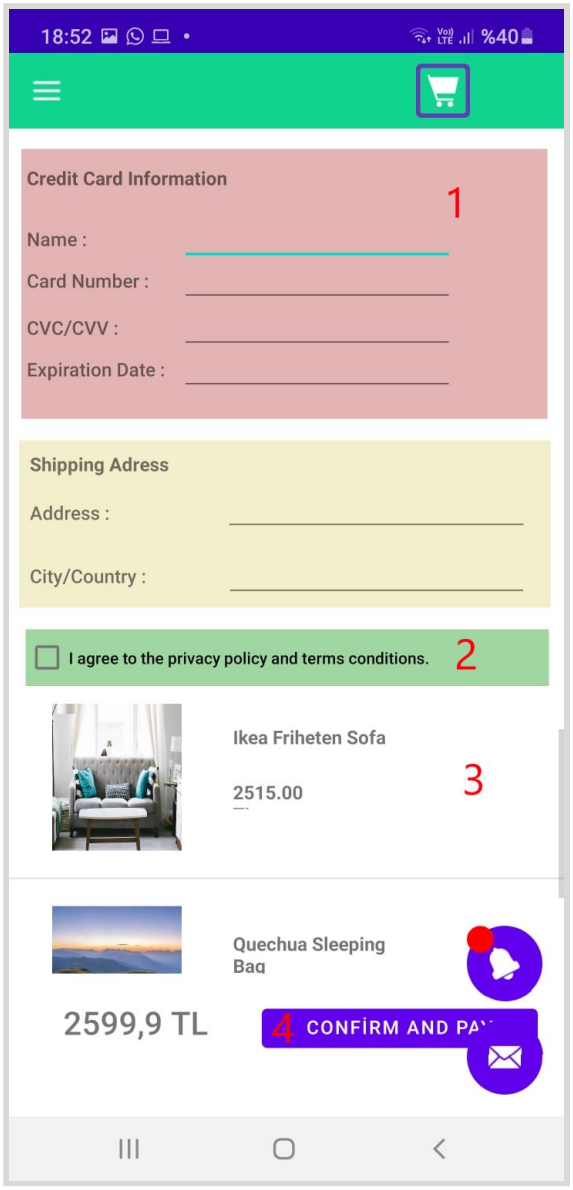
You can display products of a category by choosing the category from the side menu.



Shopping Cart
You can display the products in your shopping cart from the icon at the top.

You can increase[2] and decrease[1] the number of each item and display the total amount of money[5]

By clicking continue to shopping[3], you can return to the main page or you can click to payment page[4] to complete payment process

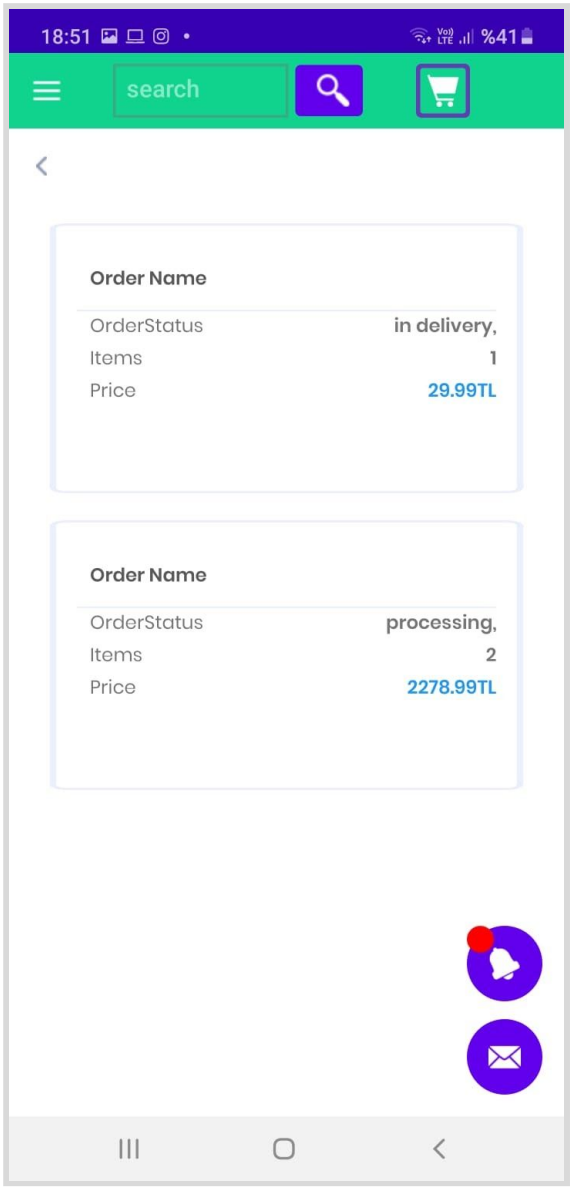


Payment Page
After continuing to pay, you will be directed to the payment page.

You should fill the payment information to proceed. [1]

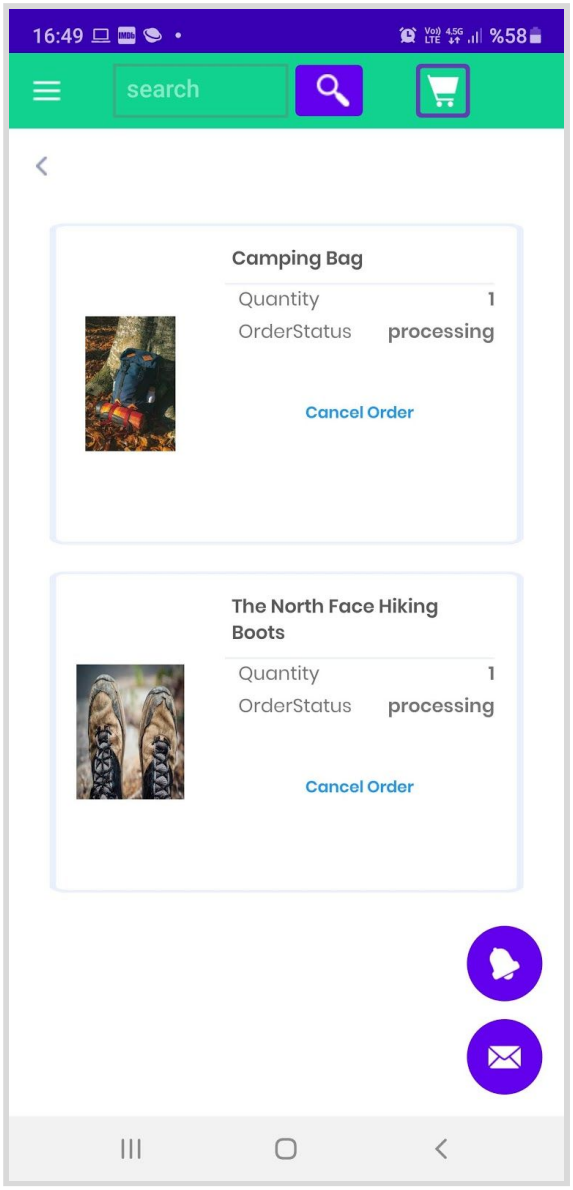
You should agree the terms and conditions [2]

The products you are buying are listed in this page[3]



Orders Page
You can see your orders and their status from the Orders page from the side menu.

You can display the details and the products of the order by clicking one.

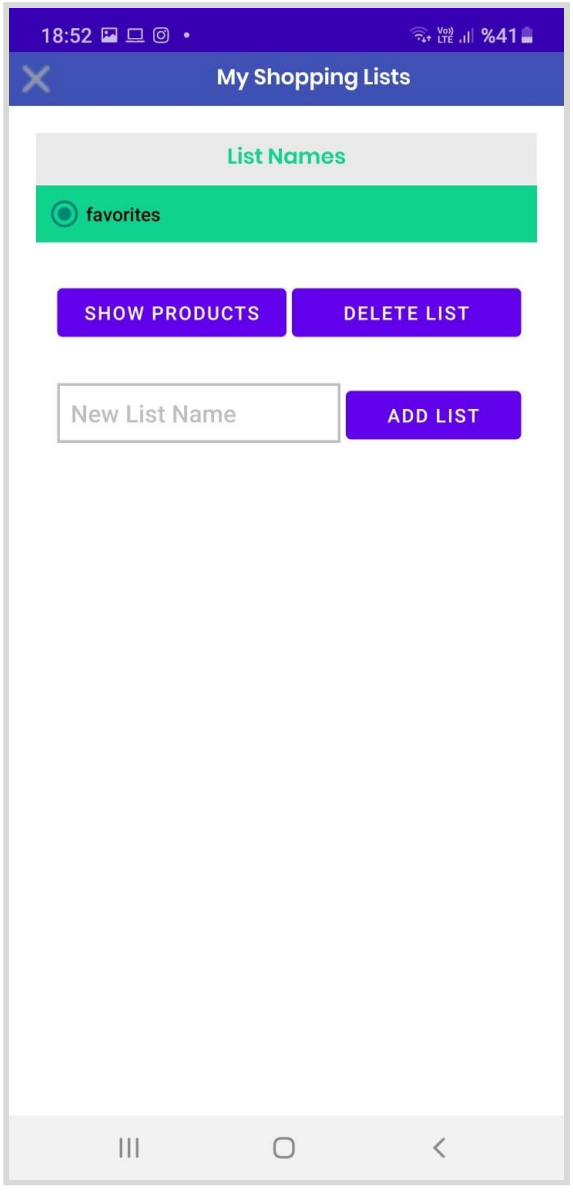


Orders Detail Page

You can cancel an undelivered order or change its status to delivered on this page.

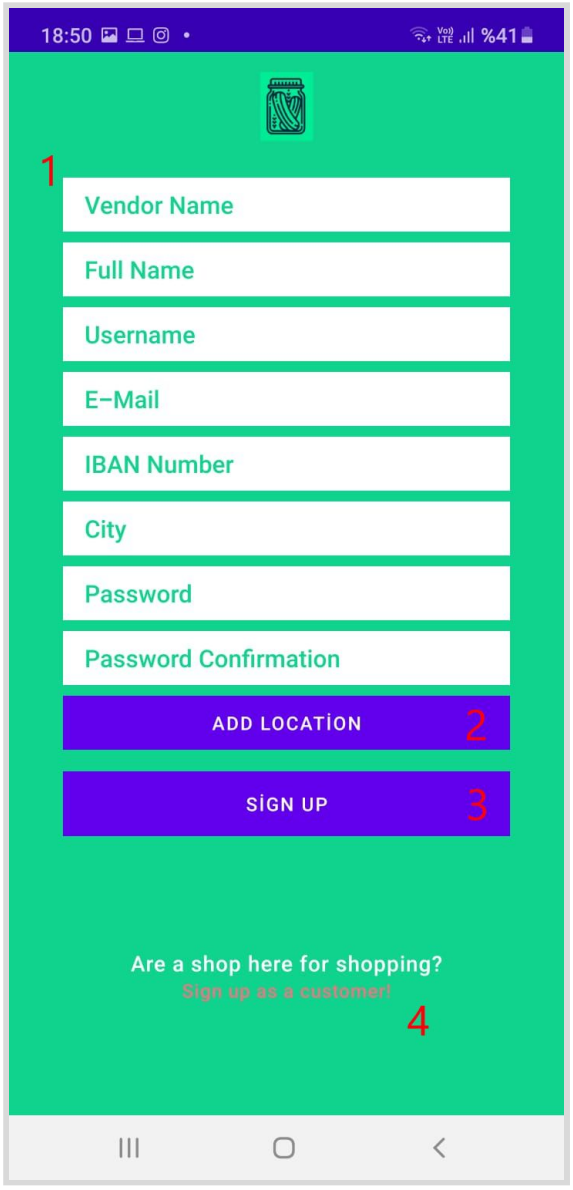
Notifications
You can see your notifications from the notifications button at the right bottom corner.

When clicking on them, you can go to the notification related item’s page.



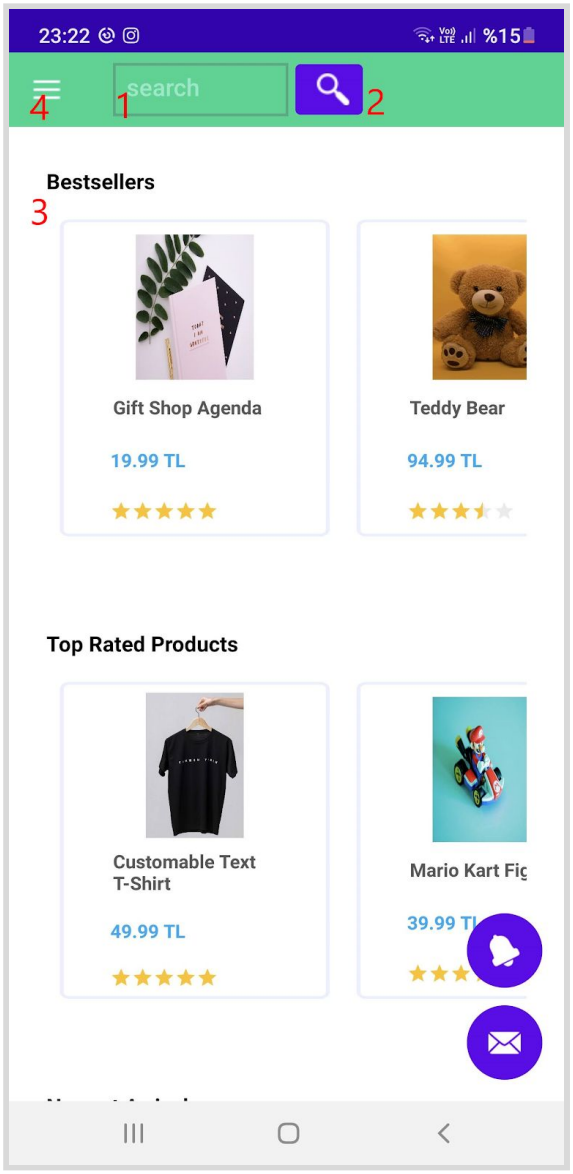
Shopping Lists
You can display and edit your shopping lists from the side manu under My Account.

-Vendor:



Sign Up or Sign in as a Vendor
You can sign up as a vendor after you fill the necessary information [1] and choose a location [2] for your store using Google Maps.
If you want to continue as a customer you can click below button [4].

Or you can use the sign in page to use and existing account

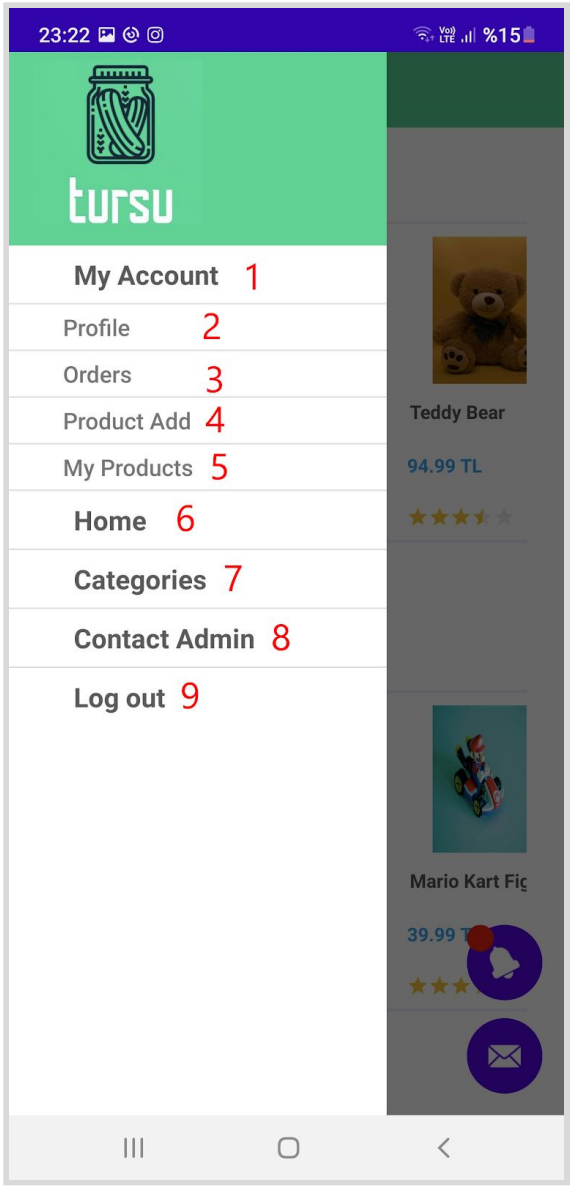


Home Page
After you sign in, you are in the home page where, top rated, bestseller and newest products are listed [3]

There is a side menu that you can use to make transition between pages. [4]

There is a search bar for you to search a desired query (vendor or product) [1] after clicking the search button [2]

You can also go to your notifications and messages from the home page.



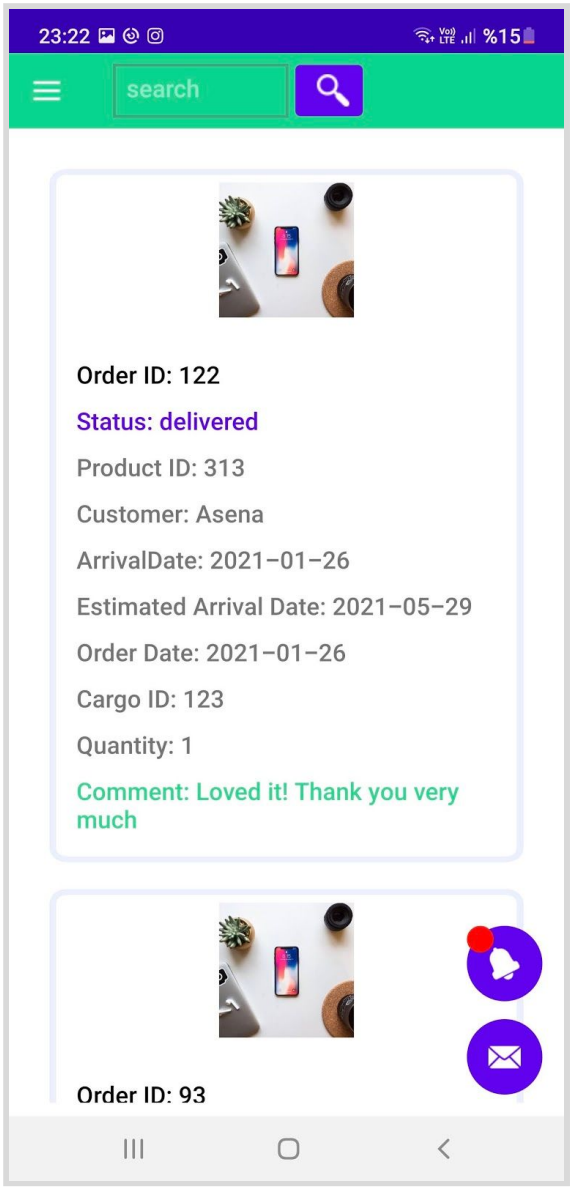
Side Menu
Side menu is used to make transitions between different pages.

You can go to your Profile[2], your orders[3], new product adding [4] or display of your products[5] from My Account[1]

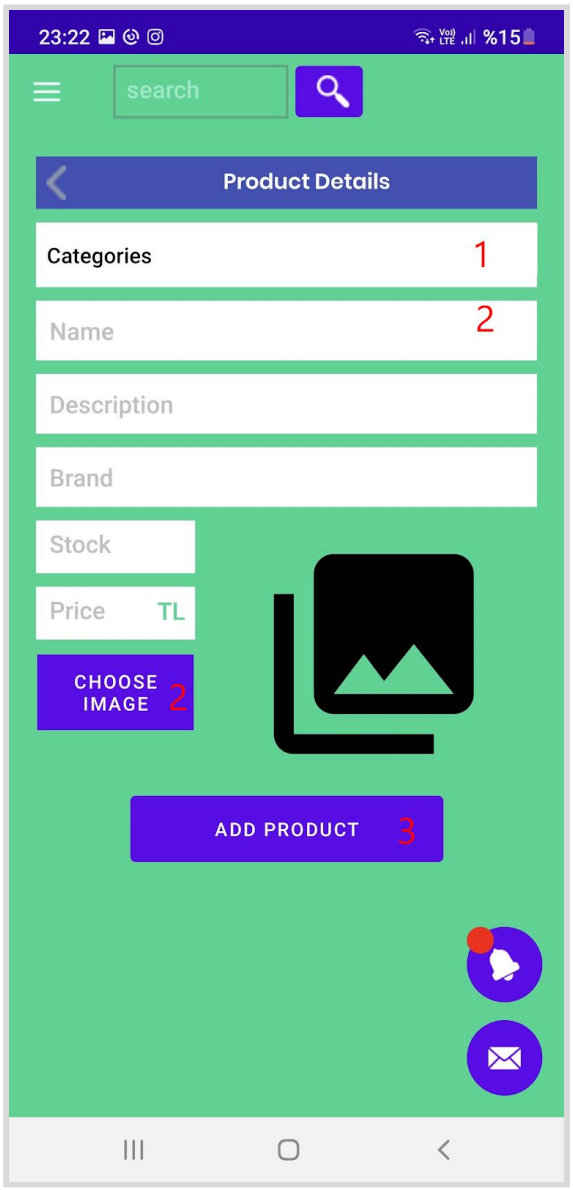
You choose a category [7]

You can contact with the Admin[8]

You can log out from the application[9]



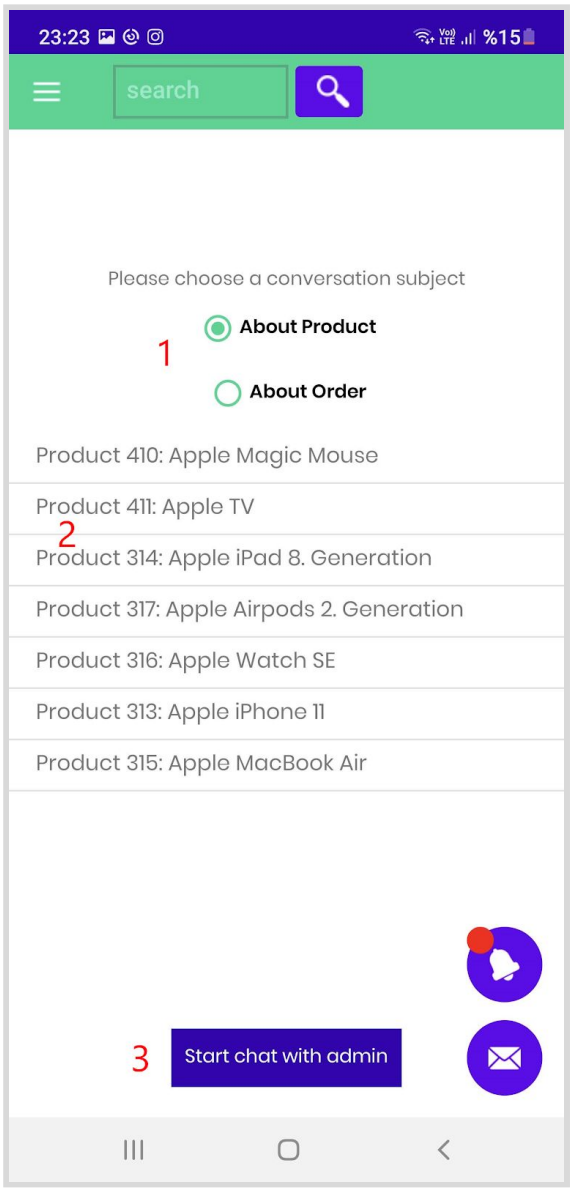
Vendor Orders
You can display your orders and change their status from the orders page.



Adding a Product
You can add a new product from the side menu.
First, you should choose a related category for your product [1]

You should fill the product information and price of it [2]

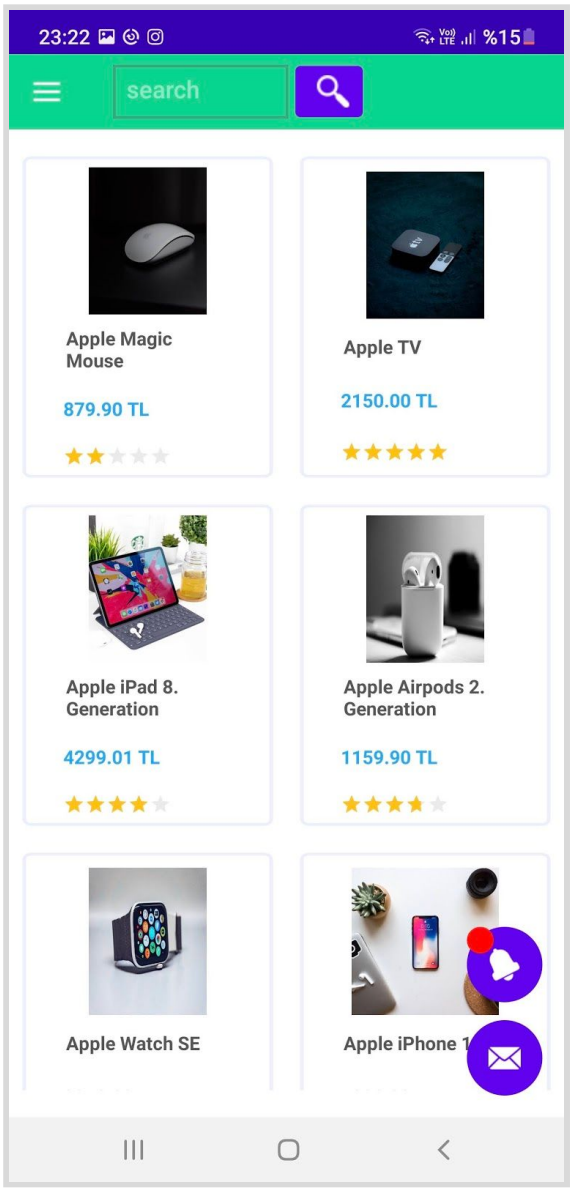
You can add an image[3] to your product



Contacting Admin
You can start a chat with the admin about a product or an order

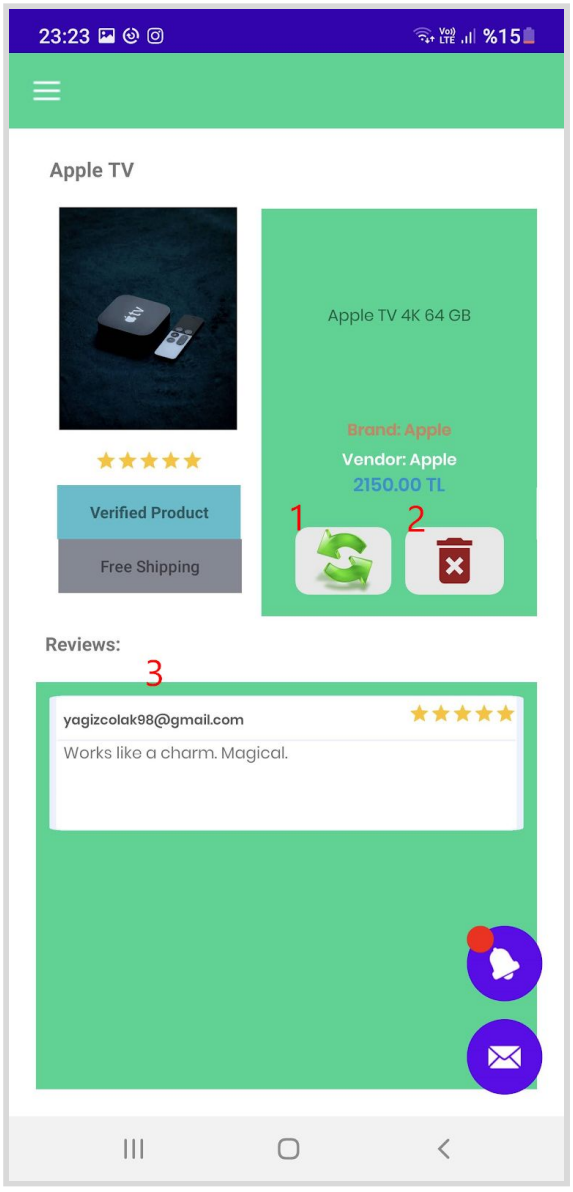
You should choose the context of the conversation[1] and the related item afterards [2].

Then, you can start a chat by clicking the button [3]



Products On Sale
You can display your added products from the side menu.

You can edit them by clicking on the desired product



Editing a product

After you display the product information, you can edit the properties of the product[1] or you can delete the product[2]

You can also see its reviews and comments below [3]

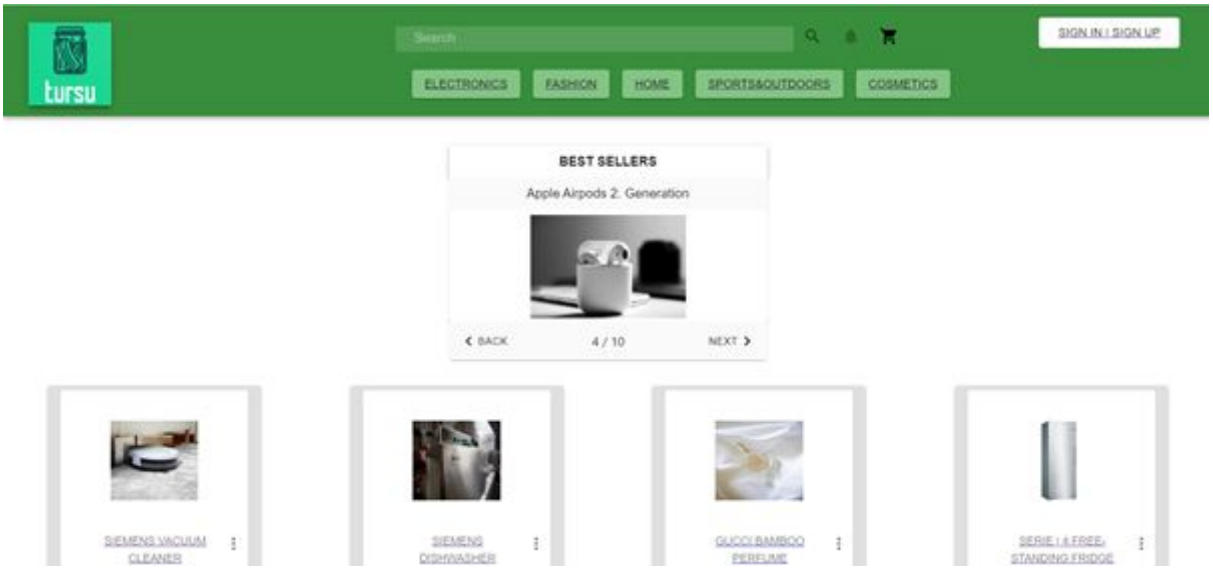
NOTE 1: Displaying categories, product details on the homepage, searching and filtering features are similar to customer users.

NOTE 2: Guest user has same functionalities as customer except shopping cart, shopping list, buying a product, profile, notifications and messaging

9.2. Frontend

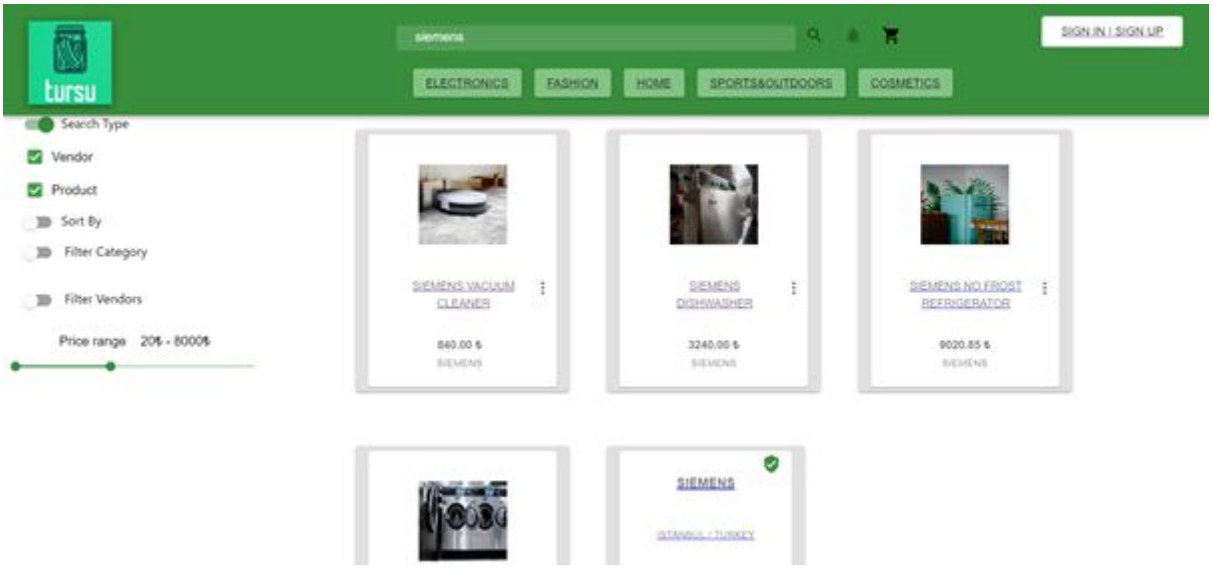
Guest

Home Page



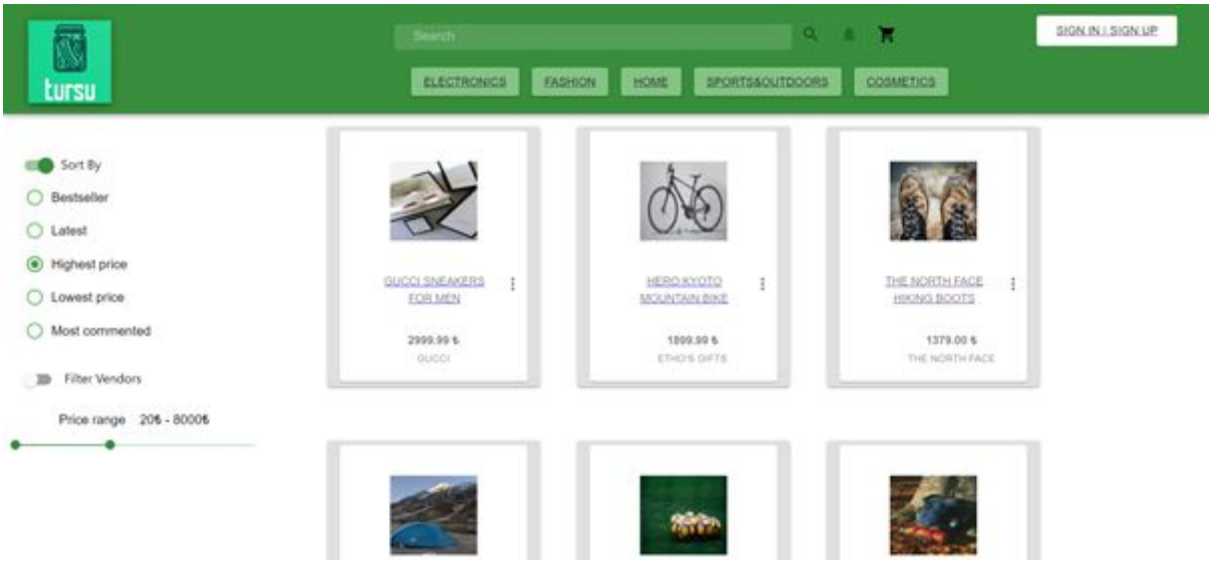
This is the first page seen by guest users. Bestseller products are displayed at the top. On the main page several products are displayed to attract attention.

Search Page and Filtering



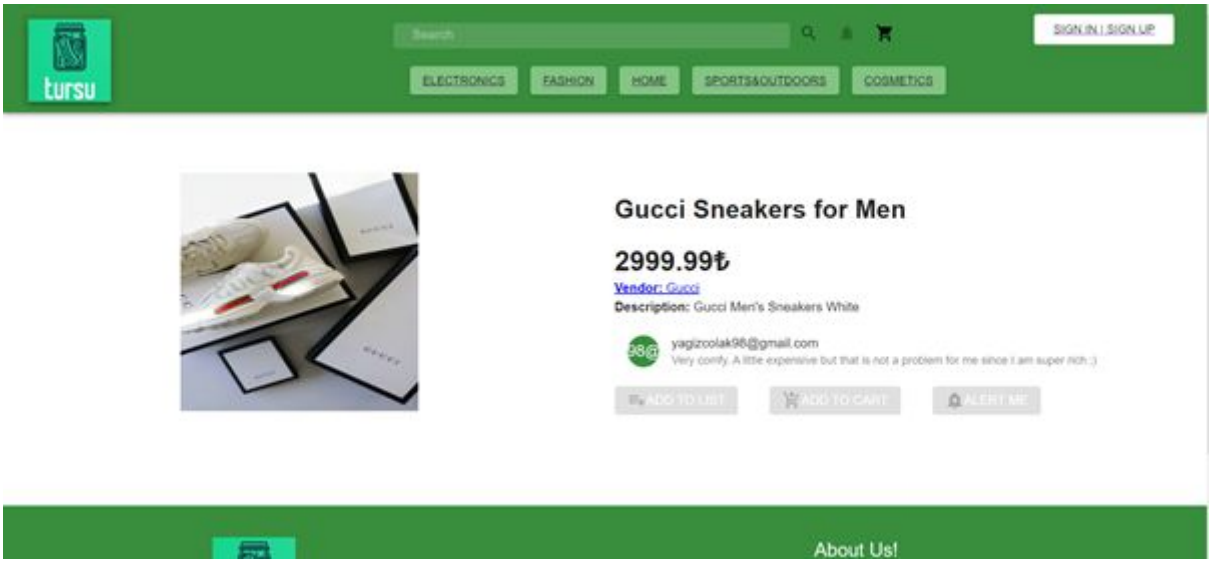
The guest users can search for a keyword. The search bar is positioned on the navigation bar at the top. Both vendors and products are displayed in a single page. Search type (searching for vendors or products) could be determined on the filter bar. Filter bar for search page supports filtering search type, category, vendors, and price range. Moreover, the sorting option could be selected from the sidebar.

Category Page and Filtering



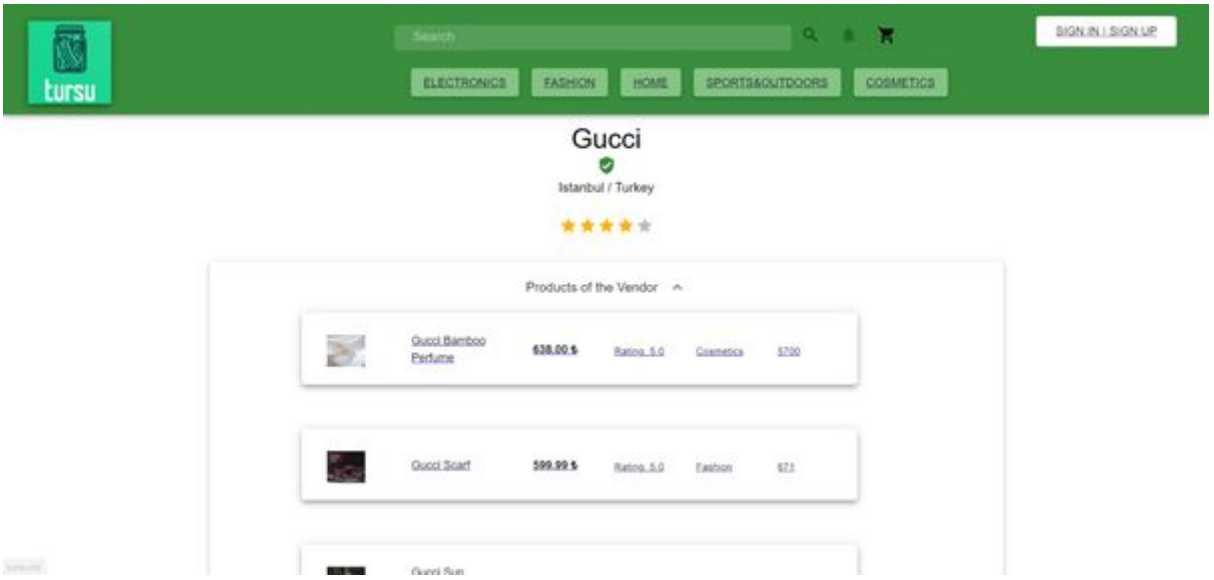
Guest users can display category pages and use filtering options. Category page supports filtering vendors, filtering price range and sorting.

Product Detail Page (Guest)



Guest users can see the price and the comments of a product. They can also navigate to the vendor of the related product.

Vendor Page



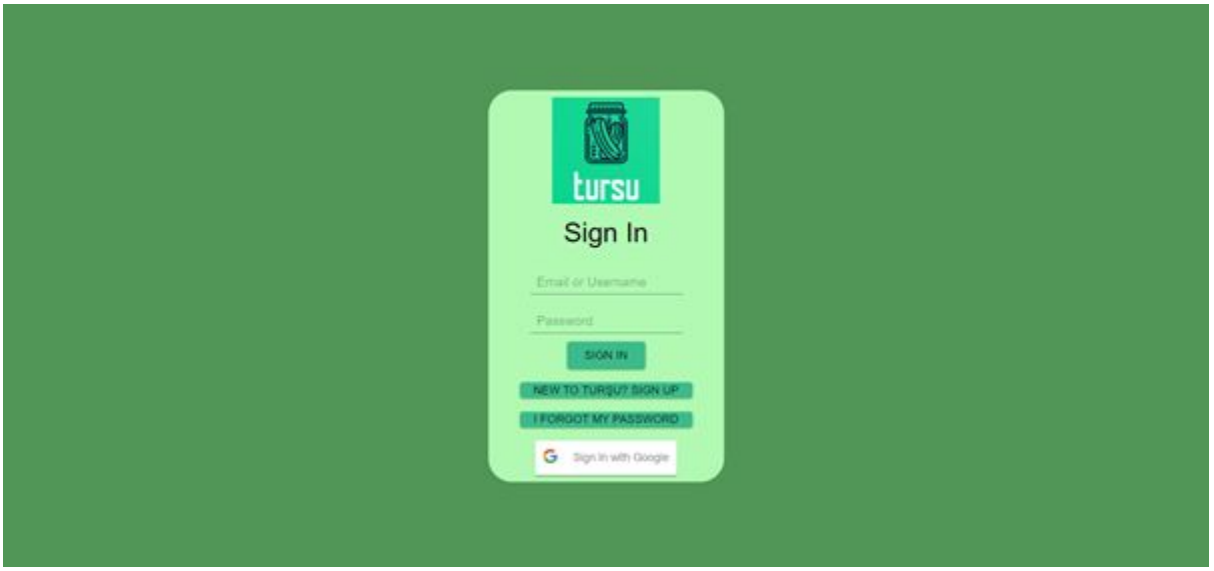
On the vendor page products of the vendor are displayed. Vendors location and rating are displayed on the top. Guest users could navigate to another product of the same vendor from the vendor page. Brief information about each product is displayed one under the other.

Functionalities that are offered to guest users, offered to all types of users. To prevent reduplication already presented functionalities are not going to be mentioned again.

Customer

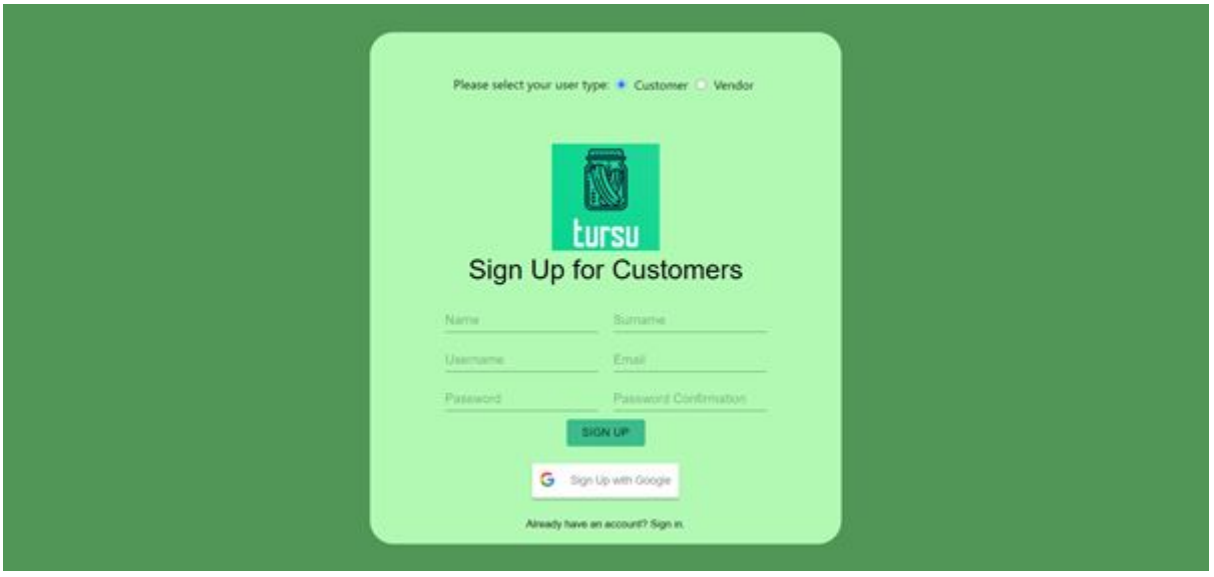
From the corner of the navigation bar top guest users are redirected to Sign In Page

Sign In Page



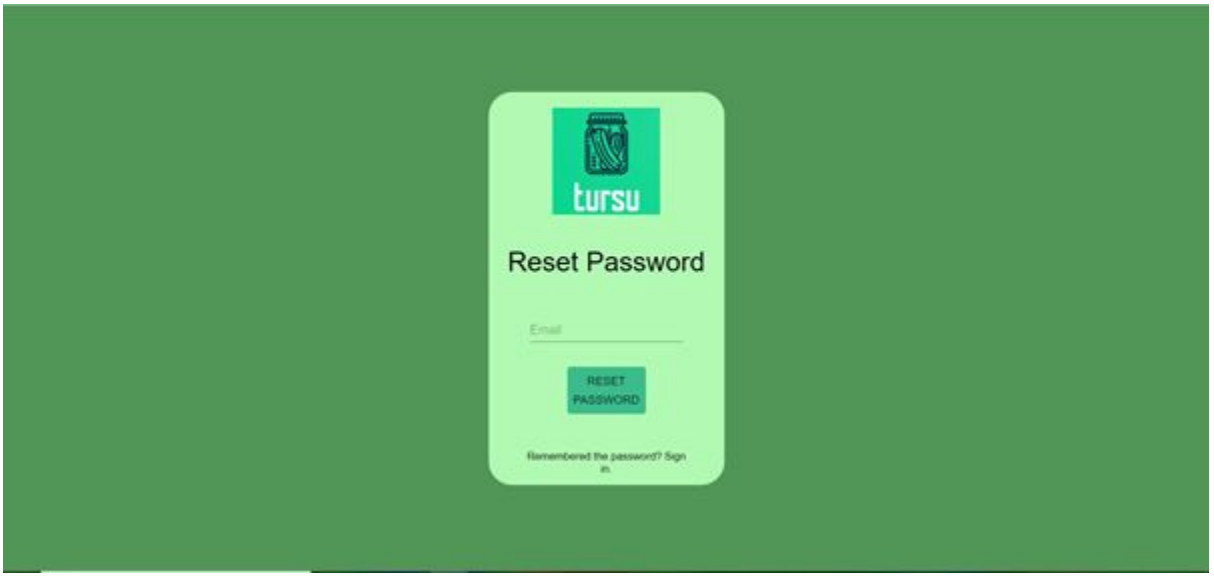
On the sign in page, already registered users can sign in. They can navigate to sign up page from here or they can navigate to forgot password page. To sign in via google, users should be signed up via a google account beforehand.

Sign Up Page



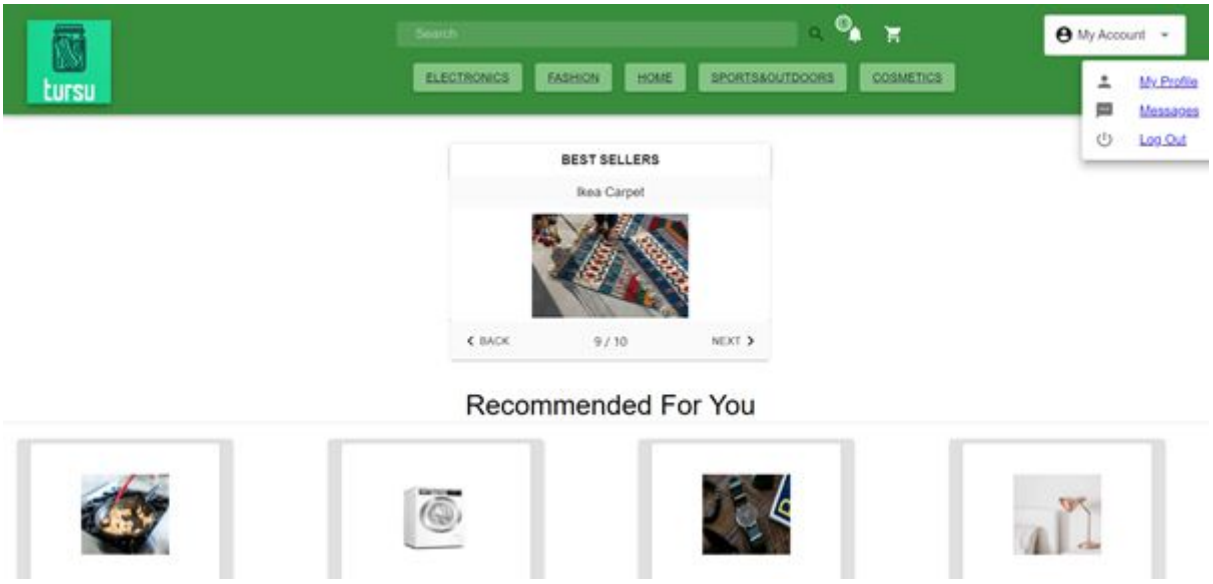
To sign up as a customer option on the top should be selected and necessary fields should be filled. A user may choose to sign up via google as well.

Forgot My Password Page



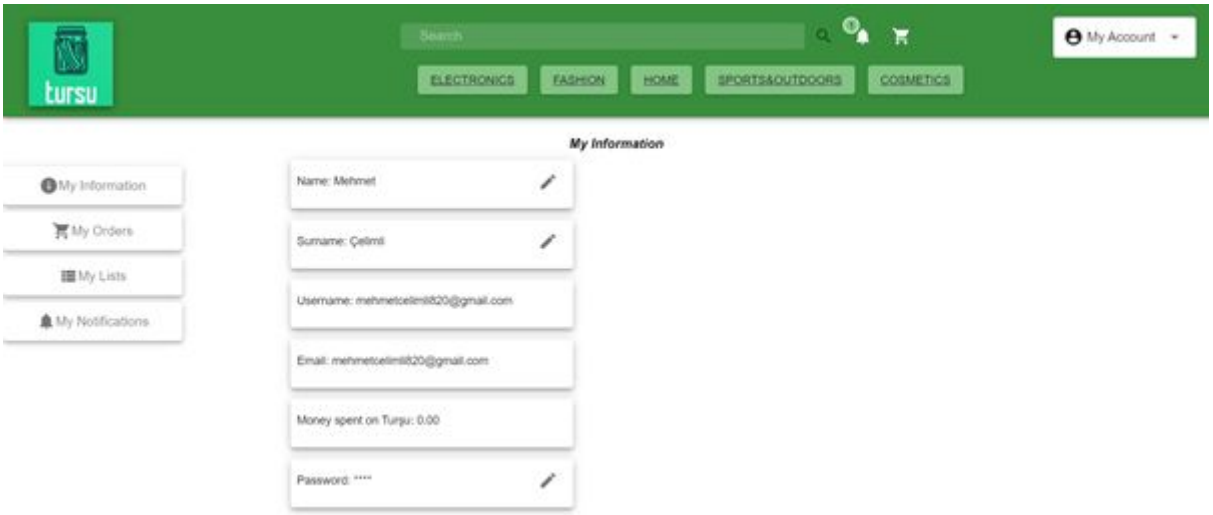
Via entering a registered email address a new random password is going to be sent to the specified email address.

Home Page (Logged In)



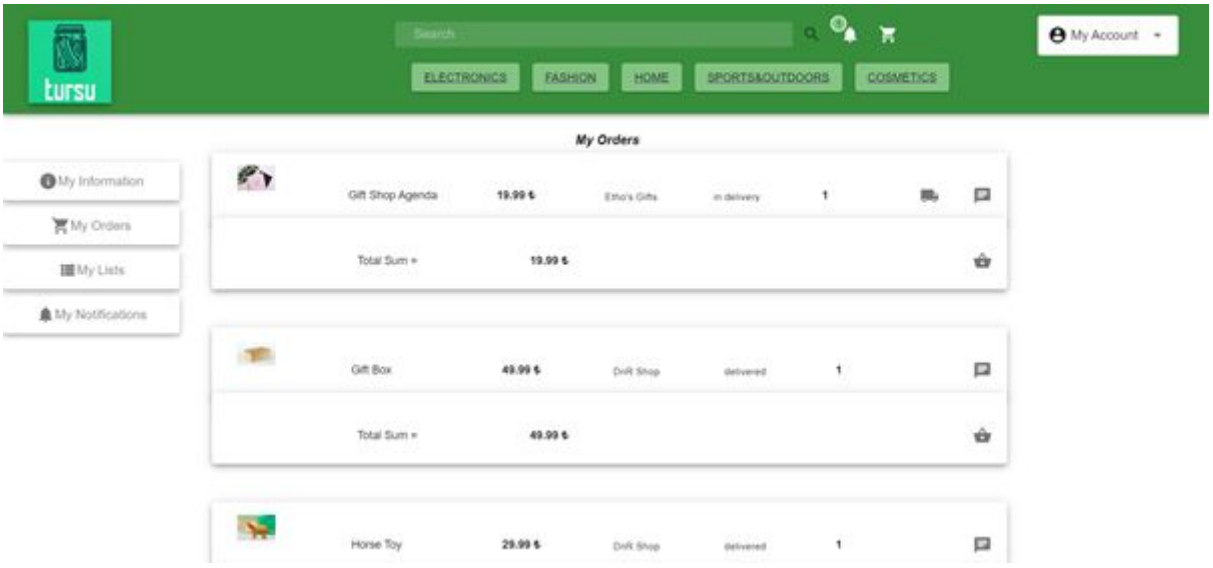
Recommended products are displayed on this page. Drop down menu on the navigation bar offers several options including navigation to profile page, navigation to message page and log out option.

Profile Page / My Information Tab



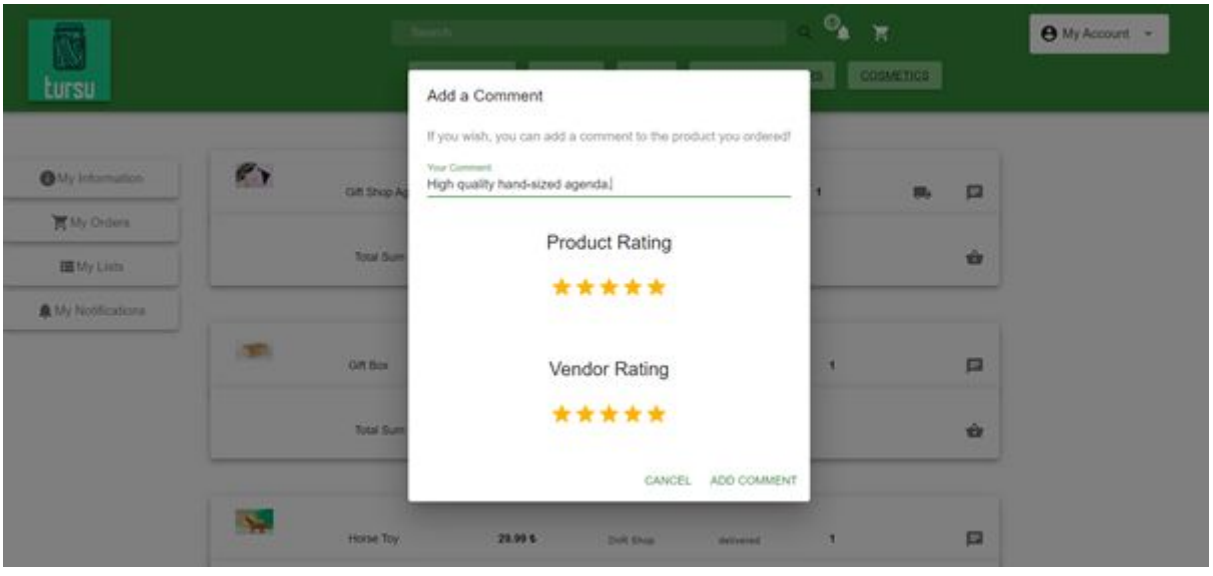
Profile information is displayed; and name, surname and password fields can be edited.

Profile Page / My Orders Tab



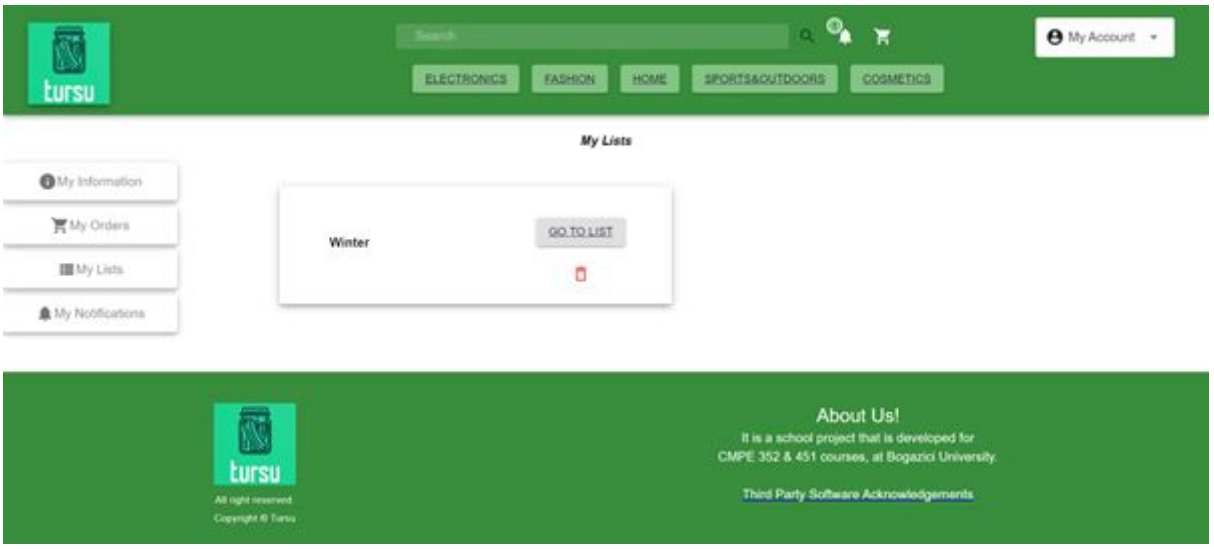
History of passed orders are displayed one under the other. From here a customer can set the status of the order to “delivered” by clicking the truck icon and can comment by clicking the comment icon.

Add Comment



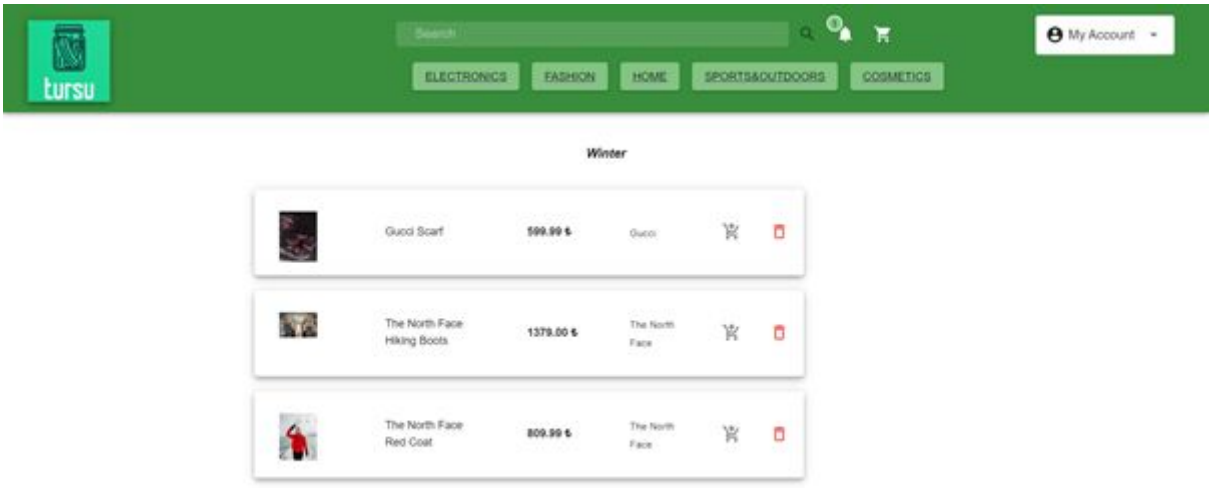
To add a comment, write a comment and click on the add button.

Profile Page / My Lists Tab



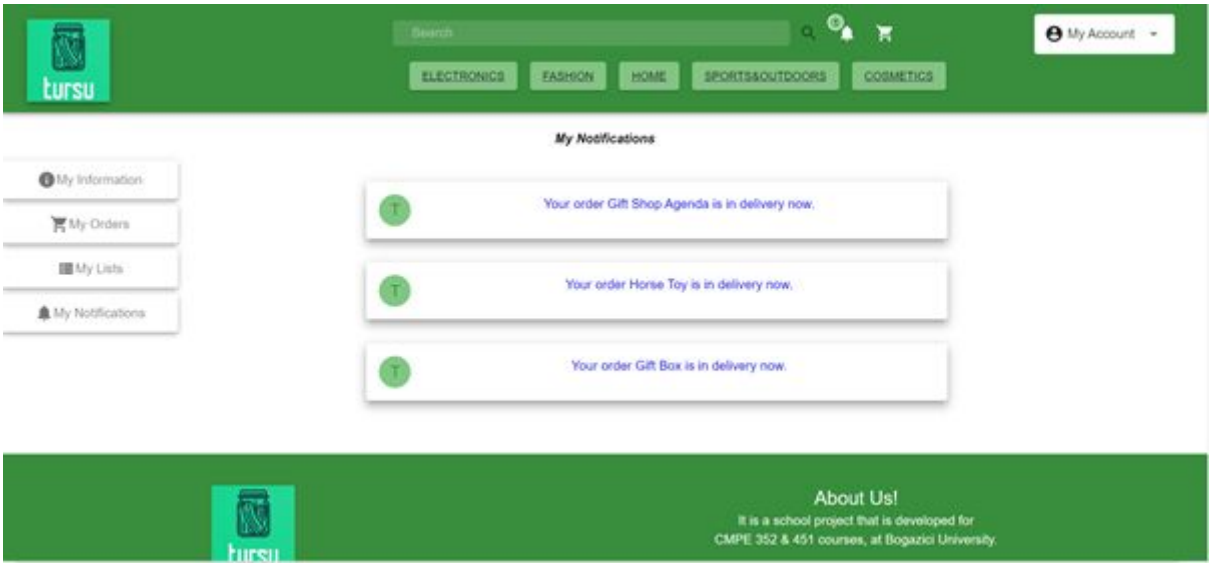
Lists are displayed on My List tab one under the other. To delete a list, click on the red trash can icon. To check the items in the list, click on the relevant button.

Profile Page / My Lists Tab (Selected List)



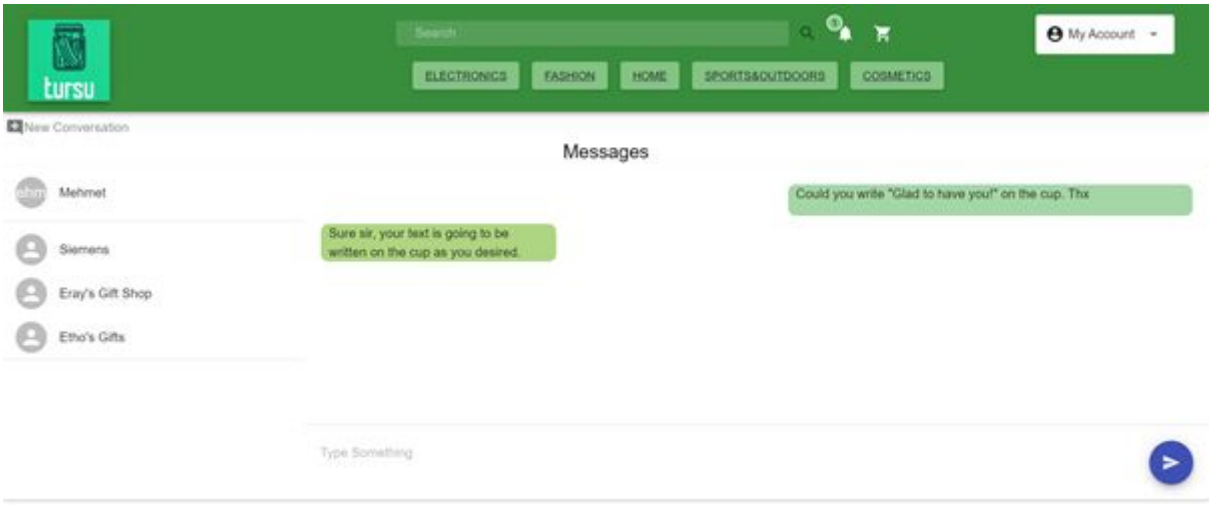
Products that are members of the list are listed. Customers can choose to delete a product from a list by clicking the red trash can icon or they can add a product to cart by clicking the gray shopping cart icon.

Profile Page / My Notifications Tab

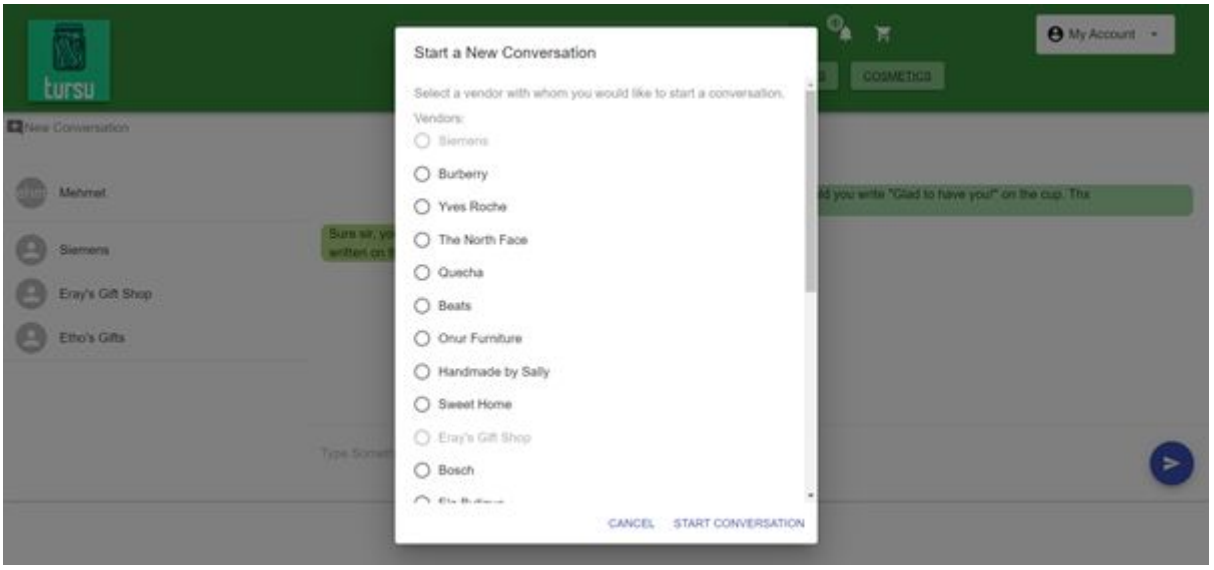


Notifications are stored and displayed here.

Messaging Page

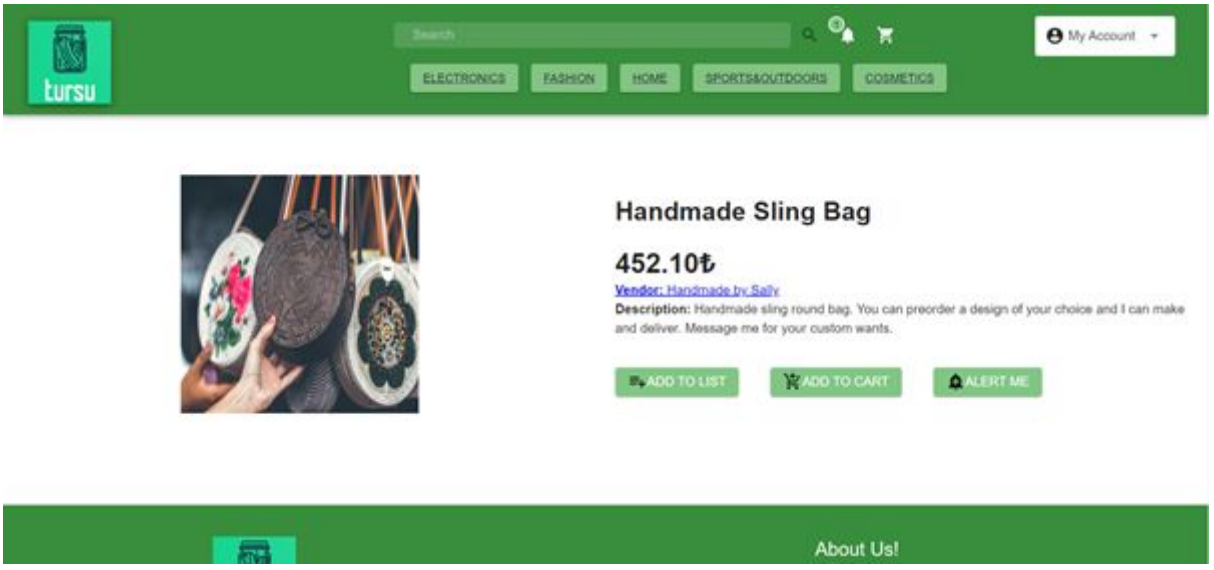


On the messaging page customers can select conversation flows and send messages to vendors.

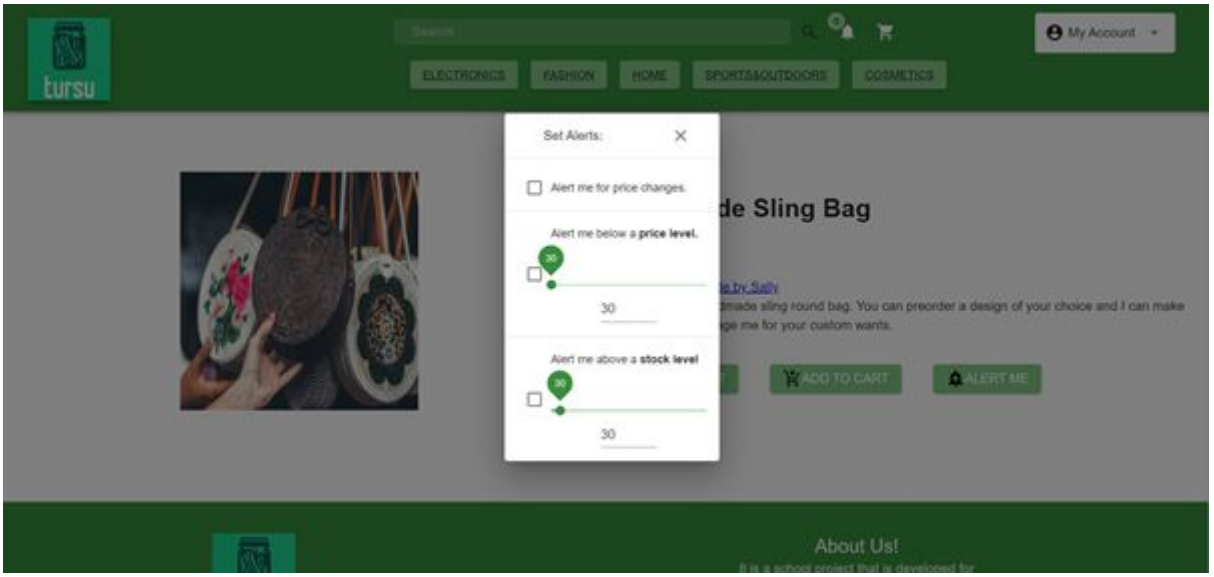


To start a new conversation with a new vendor, customer clicks the button on the top left “New Conversation” and selects the vendor.

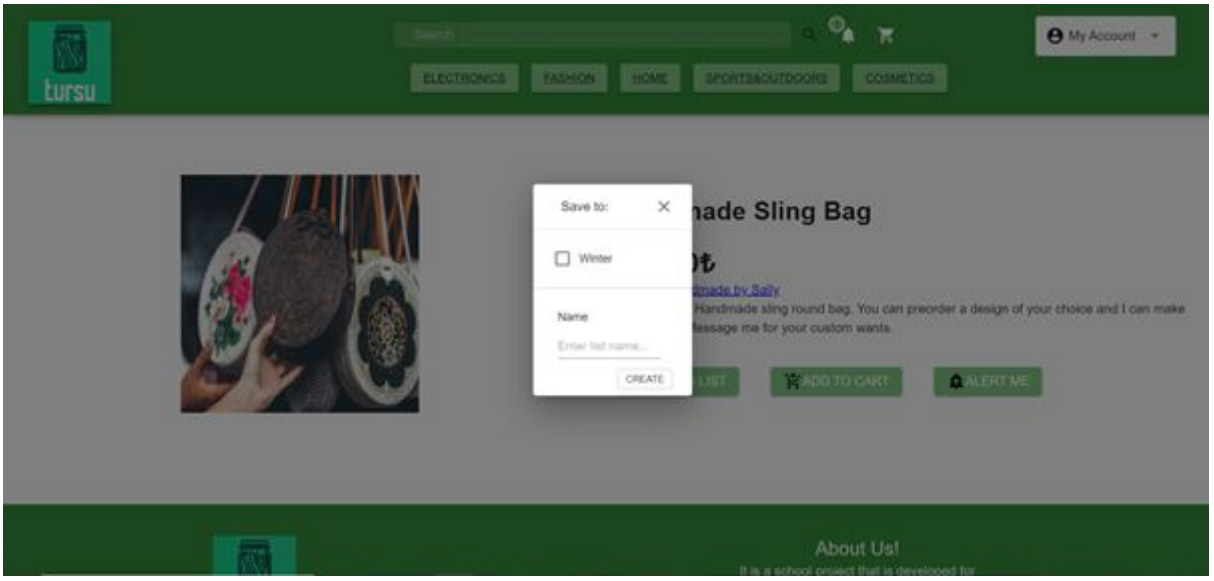
Product Detail Page (Customer)



On the detailed product page, a customer can do all the features that guest users can do and besides can add the product to cart or to a list.

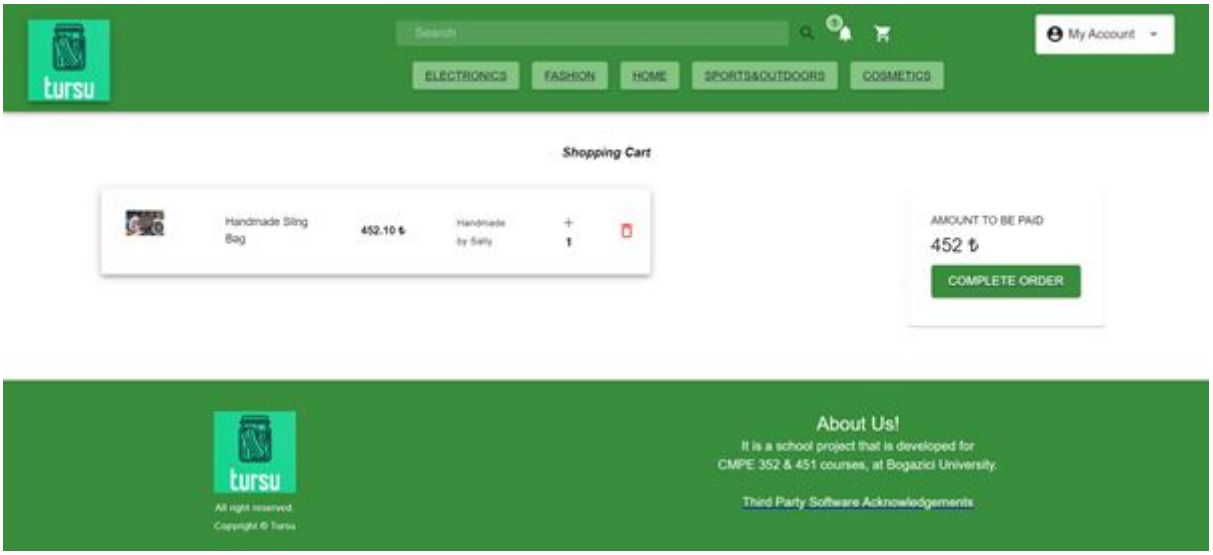


On this page a customer can also set an alert for price or stock changes.



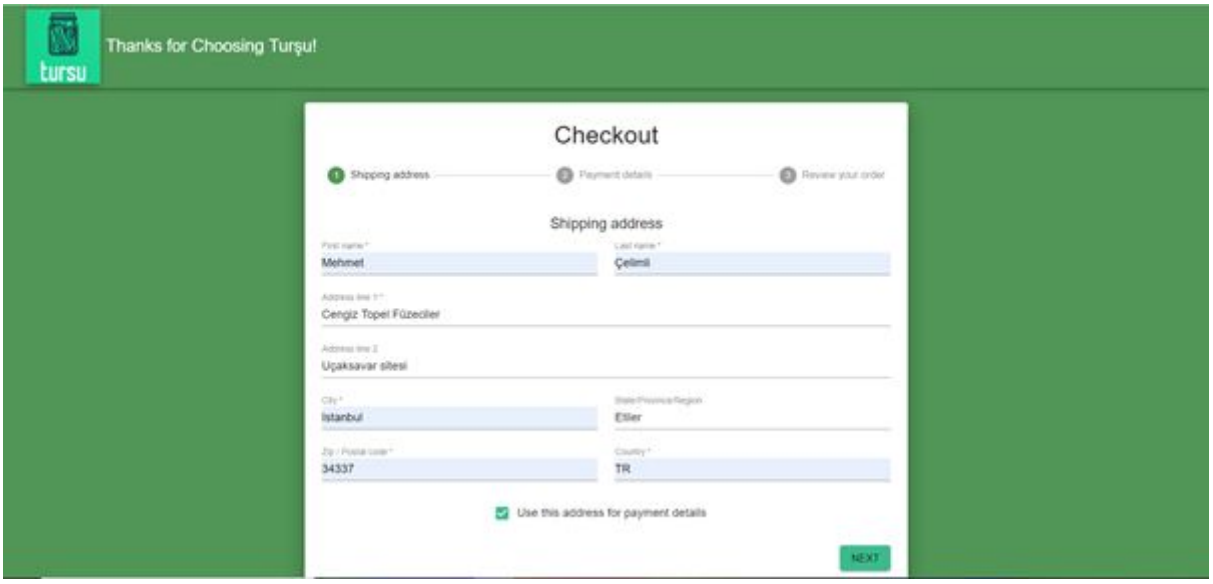
Customers create a new list by entering the new name or select a previously created list.

Shopping Cart

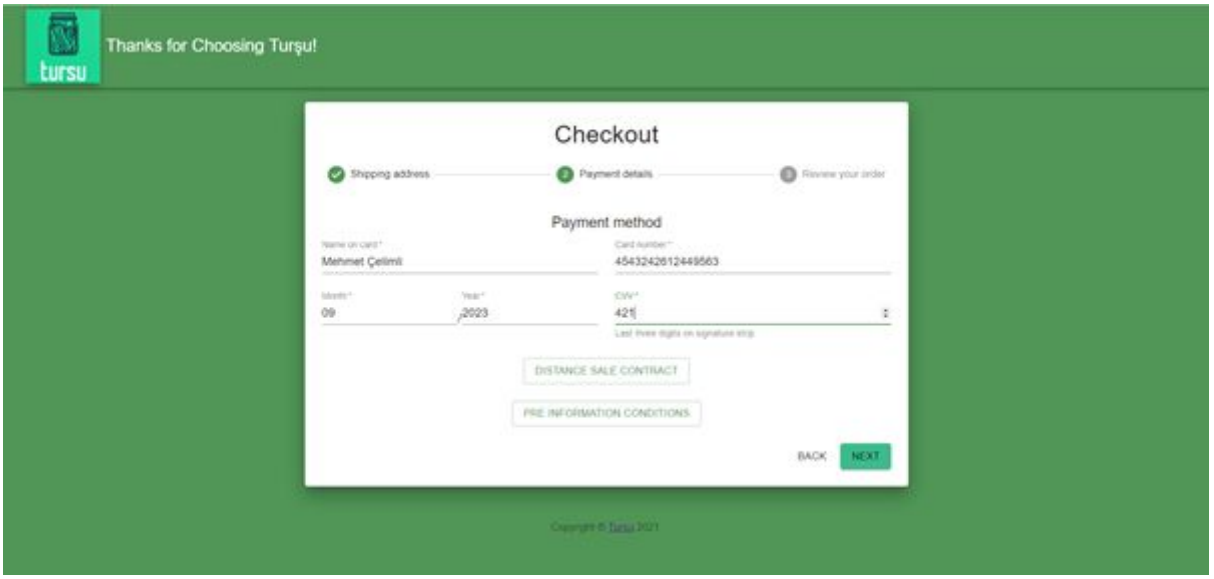


By clicking the shopping cart icon on the navigation bar, customers are redirected to the shopping cart page. On the shopping cart customers can remove a specific product by clicking the trash bar icon or they can decrease the quantity of the product. To complete the order customers should click the “Complete Order” button.

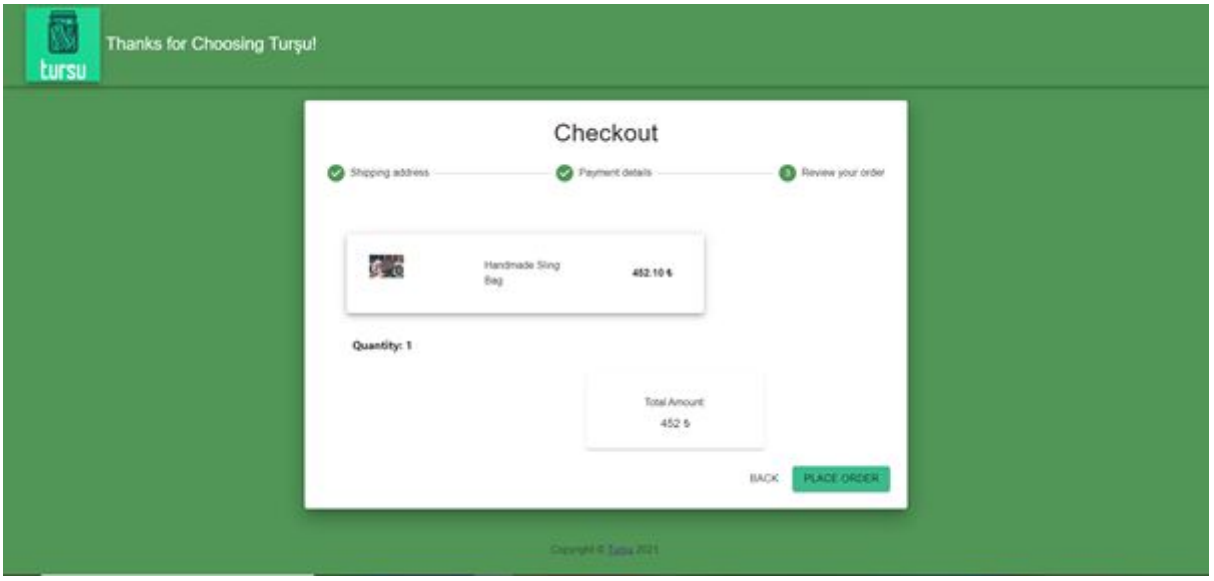
Checkout / Shipping Address



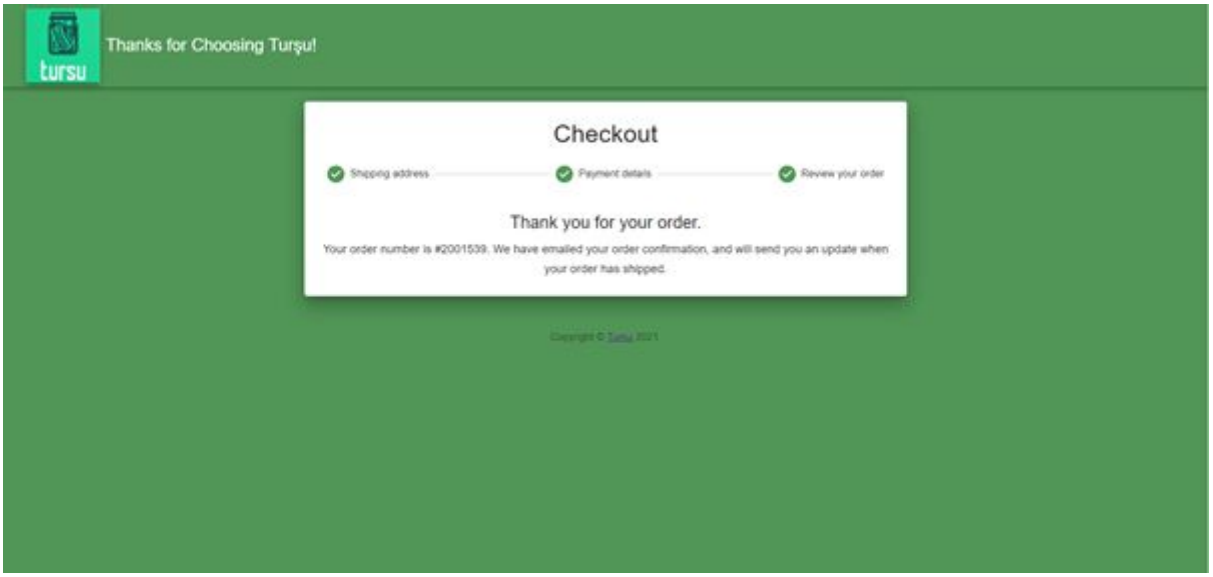
Customers should fill the address information.



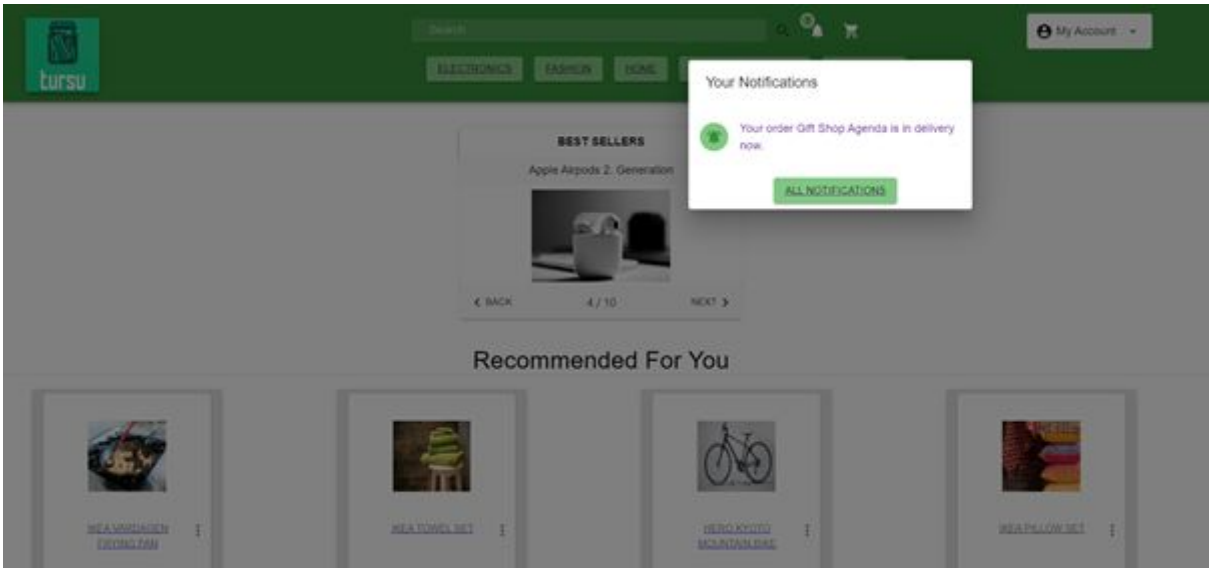
Customers should fill payment details and agree on both distance sale contract and pre information conditions in order to complete the checkout.



Customers can review their order before placing their order. This is the last step of the checkout. After clicking the “PLACE ORDER” button checkout is completed.



Notifications

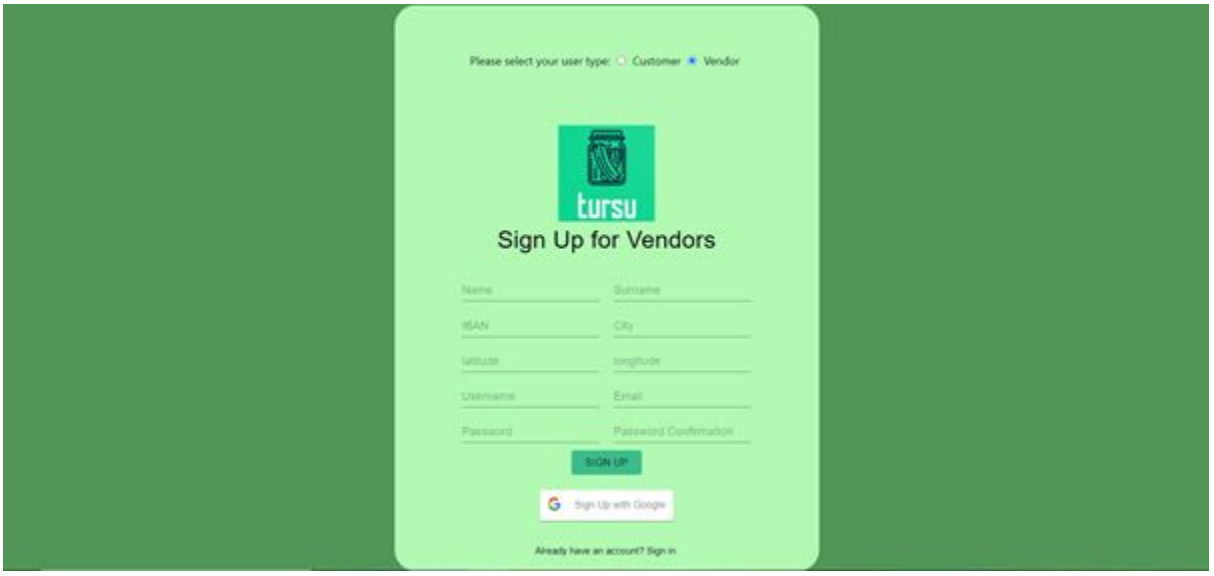


On the navigation bar a bell icon is positioned, by clicking the bell icon customers can check their notifications.

Vendors have the same my information tab, sign in, notification and messaging functionalities as customers. To prevent reduplication these functionalities are not going to be mentioned again.

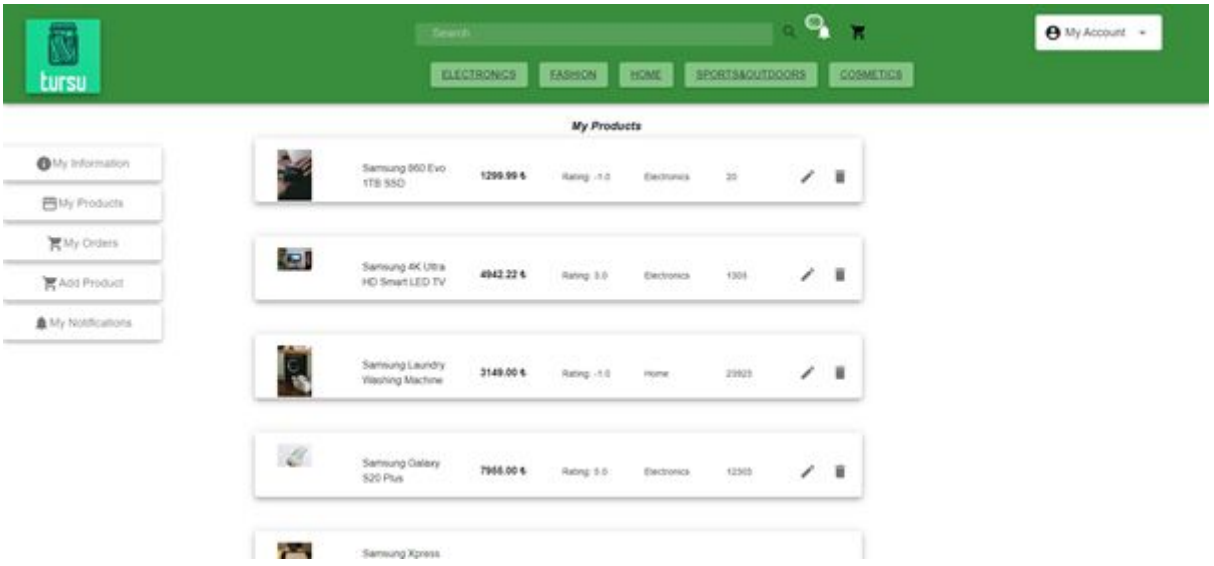
Vendor

Sign Up



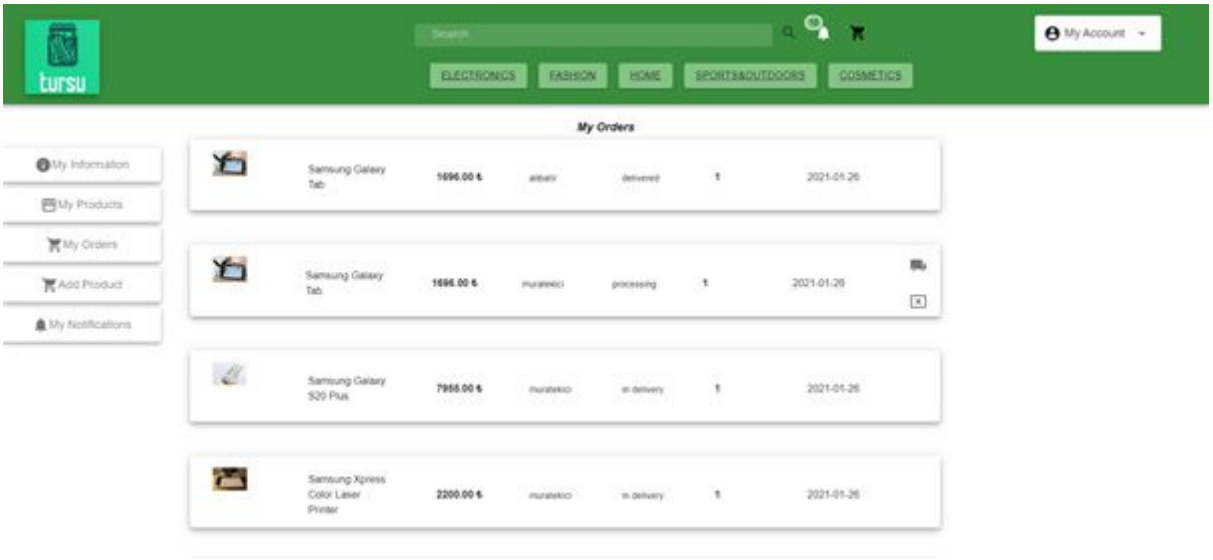
Signing up as a vendor is very similar to signing up as a customer. Additional fields including IBAN number and location information are asked.

Profile Page / My Products Tab



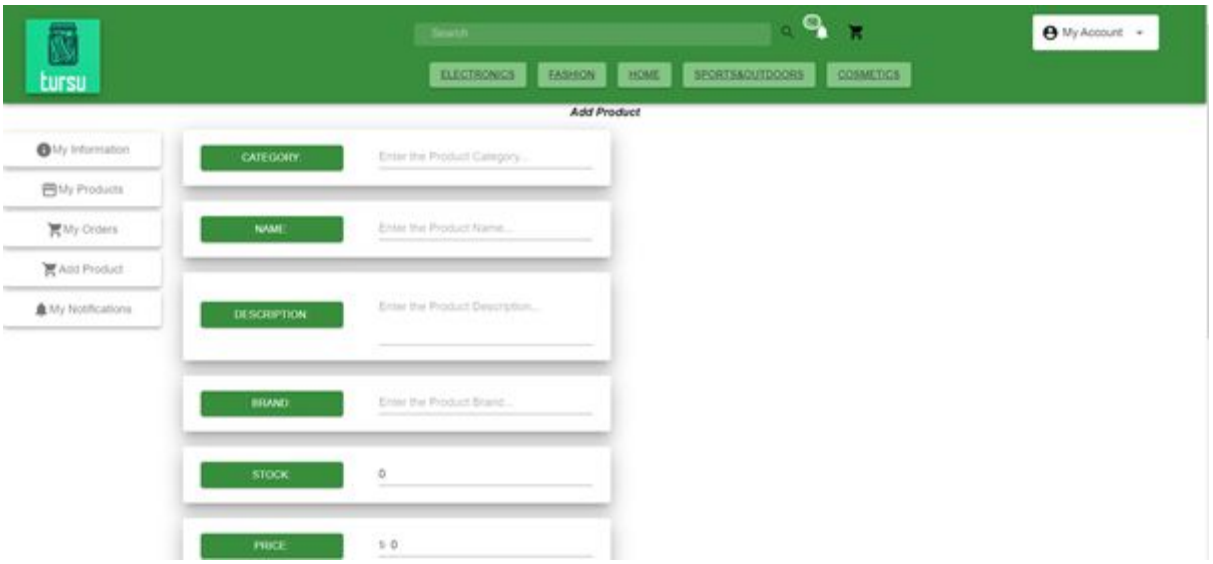
Each product is listed in the “My Products” tab. Vendors can edit their products by clicking the pencil icon.

Profile Page / My Orders Tab



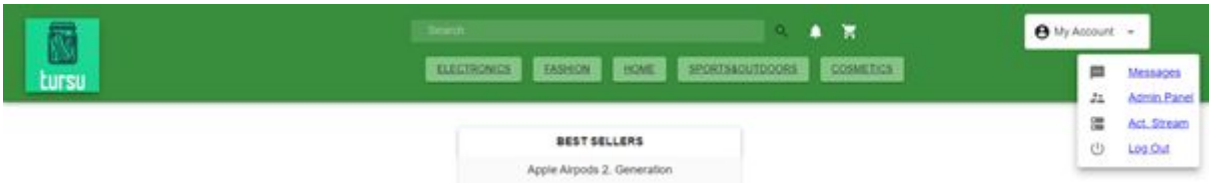
On this tab, a vendor can choose to confirm an order by clicking the truck icon or cancel it by clicking the “x” icon.

Profile Page / Add Product Tab



A vendor should fill all fields and submit in order to add a new product.

Admin



Admin has additional options on the dropdown menu: Admin Panel and Activity Stream.

Admin Panel / Verify Product

To verify a non-verified product admin should click the check icon. To deny a non-verified product admin should click the trash can icon.

Admin Panel / Ban User

To ban a user, admin should type username and click on the trash can icon.

Admin Panel / Delete Comment

To delete a comment, the admin should type a comment id and click on the trash can icon.

Activity Stream

Activity stream of the selected vendor is displayed.

10. System Manual

The frontend and the backend applications should be deployed to different machines since both of them use the HTTP port to communicate with the client. For deploying the instances you can follow the steps below.

10.1. Frontend

These instructions are to deploy the frontend web application of Tursu to a Linux based OS preferably Ubuntu 20.04 LTS.

Requirements

1. Docker
2. Docker compose
3. sed command line tool

Deployment Steps

Step 1: Clone the repository

```
git clone https://github.com/bounswe/bounswe2020group1.git
```

Step 2: Change directory to the repository

```
cd bounswe2020group1/Frontend
```

Step 3: Change backend host URL to the new backend host.

For MacOS

```
  TURSU_HOST_URL="3.232.20.250"

  NEW_URL=<ENTER-HOST-URL>

  find . -type f -name "*.js" -print0 | xargs -0 sed -i '' -e
's/'$TURSU_HOST_URL'/'$NEW_URL'/g'
```

For Ubuntu and other Linux

```
  TURSU_HOST_URL="3.232.20.250"

  NEW_URL=<ENTER-HOST-URL>

  find . -type f -name "*.js" -print0 | xargs -0 sed -i
's/'$TURSU_HOST_URL'/'$NEW_URL'/g'
```

Step 4: Build docker images with docker compose

```
sudo docker-compose build
```

Step 5: Run the docker images with docker compose

```
sudo docker-compose up -d
```

10.2. Backend

These instructions are to deploy the backend web application of Tursu to a Linux based OS preferably Ubuntu 20.04 LTS.

Requirements

1. Docker
2. Docker compose
3. Database dump document (You can find it from Wiki sidebar)
4. sed command line tool

Deployment Steps

Step 0: Download the DB dump and move it to the deployment directory

Step 1: Clone the repository

```
git clone https://github.com/bounswe/bounswe2020group1.git
```

Step 2: Change directory to the repository

```
cd bounswe2020group1/
```

Step 3: Checkout to the backend branch

```
git checkout origin/backend
```

Step 4: Create .env file

```
cp env.conf .env
```

Step 5: Add the current host to the DJANGO_ALLOWED_HOSTS variable in .env and to TURSU_HOST_URL variable in backend/tursu/settings.py

For MacOS

```
  TURSU_HOST_URL="3.232.20.250"

  NEW_URL=<ENTER-HOST-URL>

  find backend/tursu/ -type f -name "*.py" -print0 | xargs -0 sed -i '' -e
's/'$TURSU_HOST_URL'/'$NEW_URL'/g'

  find . -type f -name "*.env" -print0 | xargs -0 sed -i '' -e
's/'$TURSU_HOST_URL'/'$NEW_URL'/g'
```

For Ubuntu and other Linux

```
  TURSU_HOST_URL="3.232.20.250"

  NEW_URL=<ENTER-HOST-URL>

  find backend/tursu/ -type f -name "*.py" -print0 | xargs -0 sed -i
's/'$TURSU_HOST_URL'/'$NEW_URL'/g'

  find . -type f -name "*.env" -print0 | xargs -0 sed -i
's/'$TURSU_HOST_URL'/'$NEW_URL'/g'
```

Step 6: Build docker images with docker compose

```
sudo docker-compose build
```

Step 7: Run the docker images with docker compose

```
sudo docker-compose up -d
```

Step 8: Restore the database dump

```
sudo docker exec -i db psql --username dbadmin --password tursu_db <
../dump.sql
```

11. Project Plan

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

Gantt Chart - page1

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

Gantt Chart - page2

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

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

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

12. Assessment of the Customer Presentation

Last customer presentation was tough because we had some unexpected errors that we never encountered before in development or testing of the system or in our presentation rehearsals in the team. However we fixed that issue in a short time and we were good to go. Our presentation was better compared to the second customer presentation, we started demonstrating functionalities of our Android app and continued with the Web demonstration and finalized our presentation with the Android part again. We mainly focused on showing notification and messaging related features and we believe that we showed what our system is capable of. We had some minor issues about showing the ratings in the Web application and showing the messaging context. The customers gave valuable feedback on how to improve our system further. In Q&A session we received several questions about our system, our design choices and how we were accomplishing some tasks like semantic searching. We believe that we answered these questions properly while demonstrating them with an example. After the Q&A we proudly concluded our section in the customer presentation day.