CMPE-451 Final Milestone Report
Group 5

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

Contents

1 Final Project Assessment **4**
- 1.1 Executive Summary of the Project Overall . . . 4
- 1.2 Evaluation & Lessons Learned . . . 6

2 List and Status Deliverables **7**

3 A Summary of Coding Work Done by Each Team Member **9**

4 Unit Tests **12**
- 4.1 Yusuf Yüksel . . . 12
- 4.2 İsmet Sarı . . . 14
- 4.3 Volkan Bulca . . . 16

5 Deployment, Dockerizing, and CI/CD processes **18**

6 API Documentation **19**
- 6.1 Authentication . . . 19
- 6.2 Alarms . . . 19
- 6.3 Users . . . 20
- 6.4 Products . . . 20
- 6.5 Comments . . . 21
- 6.6 Credit Cards . . . 21
- 6.7 Purchases . . . 21
- 6.8 Orders . . . 22
- 6.9 Chats . . . 22
- 6.10 Cart . . . 22
- 6.11 Favorites . . . 23
- 6.12 Product Lists . . . 23
- 6.13 Notifications . . . 23
- 6.14 Admins . . . 24

7 Requirements **25**

8 Design Documents **31**

9 Mockups **32**

10 Project Plan **34**

11 Assessment of the Customer Presentation **40**

12 System Manual **41**
- 12.1 Requirements . . . 41
- 12.2 How to Build & Deploy . . . 41
  - 12.2.1 Clone the repository and navigate to the sources root. . . . 41

12.2.2 Build & run 41
12.3 If you want to run the app separetely without docker you need to follow these steps. . 41
12.3.1 To run backend: 41
12.3.2 To run frontend: 41
12.3.3 To run Mobile 42

1 Final Project Assessment

1.1 Executive Summary of the Project Overall

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

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

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

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

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

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

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

After the second customer presentation, what we did at first is implementing the missing or inadequate features that were mentioned in the second customer milestone. This includes order status pages and order cancellation functionality. On frontend side, we implemented order history page where customers can trace the statuses of their current and previous orders. Also, messaging functionality is added to the frontend. On mobile side, android team also implemented the order history pages. They also completed the vendor add product functionality.

After those implementations related to the second milestone, we started working on the tasks that are left undone until the final milestone. The backend team makes recommendation system for the customers. Recommendations are done according to the search history, products in lists and cart, and the previously bought products. The search history is an important part of the recommendation system, which has not implemented yet, therefore for the registered customers, search history functionality is implemented.

Another important new feature that is implemeted by the backend team is the notification system. Notifications are available only to the customer users, and created once there is a price change in the products which has their alarms set to a certain price level. To achieve this, alarm functionality is added. Providing a price limit, an alarm is set and if the price drops below that amount, a notification is sent. Alarms can be edited and be removed. Additionally, when the stock of a product that is in lists or cart ends, or replenishes, a notification is created. Notifications also created when a customer makes a purchase, and the status of an order changes such as when it is shipped or delivered.

Shipment functionality is also completed on backend side. When a vendor tries to change the status of a valid purchase from 'Preparing' to 'Ship', shipment information is taken from the vendor. Also, an endpoint to receive all shipment information is also added. If a shipment is completed, that is, if a purchase is delivered to a customer, this customer can give a rating score to that vendor out of 10, which is a new feature as well. Finally, there is now an admin user which can ban users and delete inappropriate comments.

On frontend side, there is now google map support for the addresses of both customers and vendors. The address bar is divided into subsections like region, zip code and country, and is shown on the google map. The messaging page is implemented, so a customer and a vendor can see their latest chats. A customer can now set price alarm to the products from the product pages. Product pages also display the rating of the vendor. The notifications and messages icons are now functional. When there is a new notification or message, the corresponding icons shows the number of notification or messages unseen. When the notification icon is clicked, the notification page is opened, from where you can go to the orders page, or to the page of corresponding product, depending on the type of notification. Recommended product is shown in the homepage only for the logged in customers. From the orders page, customers can look at their order statuses, see the shipment information and obtain

the product return information. They can also give ratings to vendors of their delivered products, and display their previous ratings. Additionally, Assessments page is implemented for customers, where they can see their previous reviews about the products they have bought.

Vendors can see their all products that are on sale on Bupazar from my products page and there, vendors can see the sales number, stock and rating score of their products. Vendors can also edit their product information from there. From my orders page, vendors can update the order status, and give shipment information. From their profile page, vendors can also see their average rating score given by the customers.

On Android side, recommendations are now displayed on the homepage. There is also notification page in mobile, which the notifications can be traced. The product commenting system is enhanced for the customers. Chat system is improved and vendor rating is added. Customers can see their orders in detail on mobile as well. Vendor profile page is completed, so vendors can add and edit the product information on mobile.

1.2 Evaluation & Lessons Learned

Considering our overall progress, we have managed to make a functional e-commerce platform that covers the basic operations for both customers and vendors. We gradually got more efficient at implementing the pages/endpoints since we gained more familiarity to the languages and frameworks we have used. Since there were lots of dependencies between different functionalities towards the end, it got more challenging to come up with test cases to test our product from all aspects. Still, we tried to handle as many bugs as we can, to provide a stable and reliable platform. As a technical note, our platform sometimes show buggy behaviors on Safari, we are not sure what is causing those, but it works smoothly on Google Chrome. Therefore, we would appreciate if it is used on Chrome.

One important lesson would be to gain a deeper knowledge of the languages and tools beforehand so that we know the best approach to the problems we encounter later on. But, since we mostly were new to the tools we are using, we usually continued with a working solution we achieve just before. While this method got us through meeting the requirements, we think we would have done a better job at utilizing the capabilities of languages and tools we have used. Another lesson would be to be more accessible and organized through the development stages. Since we are students with different course schedules and health conditions, sometimes it was hard to obtain an equal work-balance among team members. Even so, we tried to compensate for the previous excessive/not-enough contributions on the following weeks/milestones.

We think we have learned a lot about building a software product from all aspects, such as requirements, design, planning, implementation, milestones, testing, presentation. Besides we got through so much concerning the communication and teamwork challenges since Covid-19 emerged at the beginning phases of our project. Overall, if our possible minor mistakes are tolerated, we are quite satisfied with the final product we achieved.

2 List and Status Deliverables

Deliverable	**Status**
Login & Sign up	Done
Home Page	Done
Profile Page	Done
Email Confirmation	Done
Product Page	Done
Comment Section with Ratings	Done
Adding new Comment & Rating	Done
Search and Sort	Done
Cart & Wishlist Page	Done
Category/Subcategory Page	Done
Order Page	Done
Payment & Credit Cards Page	Done
Chat	Done
Category/Subcategory Page	Done
Add Product Page for Vendor	Done
Order Page for Vendor	Done
Order Page for Customer	Done
Notifications	Done
Recommendations	Done
Deployment	Done

Table 1: List and status of the deliverable in Mobile

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

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

Deliverable	**Status**
Login & Sign up	Done
Email Confirmation	Done
Home Page Design	Done
Profile Page	Done
Product Page	Done
Comment Section	Done
Favorites and Lists	Done
Cart Page	Done
Search and Filter/Sort	Done
Payment Page	Done
Chat	Done
Category/Subcategory Page	Done
Add Product Page for Vendor	Done
Order Page for Vendor	Done
Order Page for Customer	Done
Notifications	Done
Recommendations	Done
Vendor Rating	Done
Assessments Page	Done
Admin	Done
Google Maps	Done
Deployment	Done

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

3 A Summary of Coding Work Done by Each Team Member

Member Name	Contributions
Kayacan Vesek	- I implemented the mobile google login, and added sha1 release and debug OAuth keys to google console. - I implemented chat for vendor and customer side, as backend chat model changed I updated the requests. - I designed vendor homepage, and implemented the navigation when vendor logged-in - I made changes for people who have logged in as vendor to use editing profile info and password - I designed adding a new product page for vendors also implemented the service. - I added the file access request to access user's media.
Emre Hoşer	- I implemented the comment and rate functionalities in product page. - I implemented the notification page. - I modified the register page, added the vendor-customer radio view. - I designed, added the value picker view to the rate part on product page. - I designed the notifications view according to the Website. - I added notification icon. - I created notification response model. - I implemented comment and rate view reloading after a new comment added.
Volkan Bulca	- I implemented the Shipment and VendorRating models. - I implemented the required serializers for Shipment and VendorRating models. - I implemented VendorRating endpoints such as orders/add-vendor-rating, orders/avg-rating-product-page, orders/avg-rating-profile-page and orders/get-vendor-rating. - I implemented Shipment endpoints such as orders/add-shipment and orders/get-shipment. - I updated the make-purchase, update-status and vendor-cancel endpoints so that they can create notifications. - I implemented the products/opts/get_user_comments endpoint so that a customer can see all their assessments. - I updated the update-status endpoint so that it takes purchase_id rather than order_id. - I implemented 2 unit tests: AddShipmentTest and AddVendorRatingTest. - I also updated 1 unit test: UpdateStatusTest since the corresponding endpoint is updated slightly.
Sertay Akpınar	- I implemented the recommended, trending, new arrivals, best sellers pages. I also designed the view of the banners at the homepage. - I implemented the log out feature. - I implemented the change password feature. - I connected the categories and subcategories feature with the back-end. - I presented the vendor side in the mobile presentation. - I documented the chat, wish list, categories and subcategories, change password, edit personal information logout (most of the features in the my account section) features in the mobile user manual. - After the milestone, I commented my code and I also removed the unnecessary/unused parts of the codes in the project. - I created the issues number 408, 407, 393, 392, 368 and 367. - I created the pull requests number 416, 415, 391, 389 and 379.

Member Name	Contributions
Mısra Yavuz	- I implemented the Assessments page for customers, where they can see their previous reviews about the products they have bought. - I added the types of products vendor sells section to the My Products page of vendors. - I added vendors' rating scores to product pages and to the private profile page of the vendor. - I implemented some portion of UI of the Orders page, added the total cost of orders, and implemented add vendor rating and cancel order functionalities. - After our Final Milestone Presentation, I added the get vendor rating functionality to customers' orders page and solved the add rating bug. - I have done some minor fixes on the following sections: minimum limit on product rating, address field bug, edit product validation bug, margin and visual fixes, handling some console warnings, removing unnecessary imports and console logs when I noticed.
İsmet Sarı	- I implemented the admin delete comment and user. - I implemented the user with comment id, assign admin and is admin endpoints. - I changed the get all chats endpoint. - I implemented the discount notification model. - I have implemented the recommendation of changing the password regularly. - I have implemented banning user if s/he enters his or her password more than 3 times. - I implemented tests of is_admin, assign_admin, delete_user, delete_user_by_comment_id and delete_comment endpoints. - I have written the docker-compose yml file.
Yaşar Selçuk Çalışkan	- I designed & implemented the vendor orders page, where the previous orders given to the vendor are listed. - I implemented the VendorOrdersAdapter to show the orders in a recyclerview. - I designed & implemented the vendor order details page, where the vendors can go by clicking to one of the orders in the vendor orders page. - I implemented the order status update feature from vendor order details page. - I designed & implemented the customer previous orders page. - I implemented two adapters to show to previous orders in a vertical recyclerview and show the images of the products in a horizontal recyclerview for each of the order items. - I implemented the corresponding API request methods. - I implemented the search bar functionality and product search feature for homepage and the other product pages -recommended, trending, new arrivals, and best sellers. - After the final milestone, I have fixed instant delivery bug and commented my code. Also, I have filled the user manual for the parts that I have implemented.
Muhammed Halas	Implementation of the notification model. Implementation of the notifications/all and notifications/my endpoints. Implementation of the creation of the notifications related to product stocks, which are when a user has a certain product in their cart favorites of product lists, they receive a notification when that product's stock empties or replenishes.

Member Name	Contributions
Algı Kanar	-I have implemented the order pages from both vendor and customer side. adding customer order side the cancel along with return and shipment info. for vendor side adding the order status update switch and its necessary functionality. -I have implemented the my products page of the vendor. -I have implemented the edit product page of the vendor. -I added google map to vendor sign up and user profile pages. -I finished messaging feature. Also added unread messages badge to mail icon -I added alarm feature to product page. -After the presentation I added different currency display to product page. -Also added vendor public view page which accessible from both search and product page - I did minor fixes in address features,search,category and profile pages. - Added some of the connections between pages -I reviewed the pull requests of 325,403,402,377,385
Ramiz Dündar	- Implemented forgot password page. - Implemented feature admins can delete comment. - Implemented feature admins can ban user by their comments. - Implement Terms and Conditions and small readjustments to signup page. - Fix empty cart can proceed bug with Zeynep. - Fix docker-compose decryption bug.
Yusuf Yüksel	- I have implemented recommendation system of api for both guest and authenticated customers by looking search history, cart, favorites, lists and purchases for authenticated customers. - I have implemented SearchHistory, PriceAlarm models and overwrote Notification model. - I have re-implemented search product endpoint to store search history of authenticated users. - I have implemented setting, deleting a price alarm and getting price alarms of a customer api endpoints. - I have implemented getting notifications of user and setting notification as seen api endpoints. - I have implemented a type of notification which notifies customer which set a certain price for a product as alarm to be notified when price of product goes below set price. - I have implemented getting vendor details api endpoint for public access. - I have configured test environment for API and database for local development. - I have implemented 4 unit tests: RecommendProductTest, SetPriceAlarmTest, MyPriceAlarmTest and DeletePriceAlarmTest.
M.Zeynep Çayırçimen	- I implemented the saved credit cards page where they customers can save end delete credit card - I changed address field into more structured in profile and payment pages - I added user agreement policy to payment page after final milestone presentation. - I implemented notification page where customers can see all notifications - I change notification icon. Now it dynamically shows number of new notifications - I added recommended products for customers to home page - I fixed the proceeding next step with empty cart with Ramiz

4 Unit Tests

4.1 Yusuf Yüksel

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

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

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

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

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

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

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

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

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

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

- Commit SHA - **2bb96f291566bca52d3fb7994d9bd2e69c2831b5** (This commit includes 1 unit test.)

-Unit Test for Recommendation System:
It can be found in 'test_recommend_product.py' file. In this unit test, I created customer instance. I got token of customer via AuthUserSerializer and put it to header of request object. I post proper json which includes 'query' as key and value as 'Notebook' to 'search_products' in order to store searched products in customer search history. I post proper json which includes 'number_of_products' as key and value as '15' to 'get_homepage_products' in order to complete recommended products with best sellers when number of recommended products is less than 15. I got recommended products via sending get request to 'recommend_products' function. Then, I compared status code of response with http 200 to check if it is successful recommendation operation or not. Also, I compared recommended products with products which are union of search history and best sellers. It passed all conditions.

- Commit SHA - **5d0f6e4e50ad4f21e0822ac398857d0550e0fbb2** (This commit includes 1 unit test.)

-Unit Test for Setting Price Alarm:
It can be found in 'test_set_price_alarm.py' file. In this unit test, I created category, sub-

category and vendor instances to create a product instance. Also, I created customer instance for price alarm. I got token of customer via AuthUserSerializer and put it to header of request object. I post proper json which includes 'product_id' and 'price' as keys to 'set_price_alarm'. Then, I compared status code of response with http 200 to check if it is successful setting price alarm for customer or not. Also, I compared 'success' message of response with 'Alarm is successfully set.' because 'set_price_alarm' function returns this message when customer successfully sets price alarm for product. It passed all conditions.

- Commit SHA - **f938f5b5e621be673509b6c750a21ab654527037** (This commit includes 1 unit test.)

  -Unit Test for Deleting Price Alarm:
  It can be found in 'test_delete_price_alarm.py' file. In this unit test, I created category, sub-category and vendor instances to create a product instance. Also, I created customer instance and used product and customer instance in creation of price alarm instance. I got token of customer via AuthUserSerializer and put it to header of request object. I post proper json which includes 'product_id' as key to 'delete_price_alarm'. Then, I compared status code of response with http 200 to check if it is successful delete price alarm for customer or not. Also, I compared 'success' message of response with 'Alarm is successfully deleted.' because 'delete_price_alarm' function returns this message when customer successfully deletes a price alarm. It passed all conditions.

- Commit SHA - **f58e588971ecba80b1213c3ae8bb16bfa31a8811** (This commit includes 1 unit test.)

  -Unit Test for Retrieval of Customer Price Alarms:
  It can be found in 'test_my_price_alarm.py' file. In this unit test, I created category, sub-category and vendor instances to create a product instance. Also, I created customer instance and used product and customer instance in creation of price alarm instance. I got token of customer via AuthUserSerializer and put it to header of request object. I got price alarms of customer via sending get request to 'my_price_alarms' function. Then, I compared status code of response with http 200 to check if it is successful getting price alarms operation or not. Also, I compared price alarms in response with price alarms of customer in database. It passed all conditions.

4.2 İsmet Sarı

- SHA code - c967f36c6b609be9717bead287e6476d5f2be2a4

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

- SHA code - a4a7ca986b4b0cc2a2e16a4a317cfadff1df3acb

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

- SHA code - 23193e3d0d206ca357305307b231d316ad90eb8e

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

- SHA code - 985a08a0eb3f1ccf9af6ae3a64a86207ee672e35

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

- SHA code - bf466ab3d138cf771ea7444d9638d783e8df4bef

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

- SHA code - 74a561f533965556b1d38c8e13583c7661ef5af7

  - test_admin_delete_comment

    In this test I expect to delete comment from customer with username "test_customer". First in the setup part I create an admin since admin is only able to delete a comment. Then to create a comment we need to create customer, product, vendor, subcategory and category. After all I get the admin and authoritize it and get its token. Then create a body part for the request by just using global variable of comment id and send the request. I expect that the request returns a http 200 code and a text as "comment is deleted". Besides, Comment cannot be available in object so I check it. If it is found is_found will be true so I assert that is_found variable will be false.

– test_admin_delete_user_by_comment_id

In this test I expect that the endpoint delete user by getting the user from his or her one of the comment ids. In the setup part again I create all the necessary models. In the test method I first get the admin and authoritize it. Then create a body part by using global variable comment_id. From comment model I can reach the user so I first get comment then its customer field after that I can get its user field. I get the email of the user and send the request. Request just contain the comment_id but in the end we check if the user is delete or not by using this email. Endpoint is expected to return us http 200 code and a text as "customer user is deleted". After all we iterate all the user and check if there is a user with email we found earlier and assert that it is not found.

– test_assign_admin

In this test I test if an admin can assign user another admin capability. In the set up part I create an admin user and a simple user. In the main test method I get the admin user and authoritize it. Then create a body part by using this username of the user we created in set up part. After all I send the request and I expect that the endpoint returns us http 201 code and a text as "admin is assigned".

– test_block_user

In this test I check if the user is banned if s/he enters his or her password more than 3 times. In the set up part I simply create a user and in the main test method I send a login request with correct email but with wrong password. In the end I expect that BannedUser model is created. If it is that so then the is_found variable will be correct and the assertion in the last row will be passed.

4.3 Volkan Bulca

- SHA code - bf3be2feb7f00806e9f41d199479f21f9e3d5aff

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

- SHA code - 53c66e718fa4953ef2a73bee9c78b343d3320ec2

In this commit, I wrote a unit test for **adding a credit card for a customer**. The test is done for credit-cards/opts-add endpoint. Only the customers can add credit cards to their

account, so in set up function of the unit test, I created a test customer. Then I make the login with that customer and received the authorization token. After receiving the token, I filled the request body with the necessary fields. Those fields are name, card_owner, card_number, expiration_date and cvc_security_number. If everything goes fine, the unit test should give a response message saying "Credit card is successfully added" with a HTTP_201 status.

- SHA code - 844f579348f7abb9fb766cb47fd67c732299f6a7

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

- SHA code - 2860ee562ecc25edb3c5208a11387629d2fa9df6

  In this commit, I wrote a unit test for **adding a shipment information to a preparing purchase** and **giving a rating to a vendor after a purchase is delivered**.

  - A test is done for add_shipment endpoint. At the beginning, I need to create a vendor, a customer and a product because adding shipment info is done by a vendor whose product is purchased by a customer. After creation of those, I make that customer purchase the product. This way, that vendor would receive that product as a valid order. Since only the purchases which are in preparing phase allows shipment add, I give the status as 'Preparing'. After setting everything up, I make login with that vendor and get the authorization token. Then, I prepared the request body and give purchase_id and the cargo_company as the required fields. After making the request, if nothing goes wrong, the response should have a status code HTTP_201.

  - A test is done for add_vendor_rating endpoint. At the beginning, I need to create a vendor, a customer and a product because rating a vendor is done by a customer whose purchase is in 'Delivered' status. After creation of those, I make that customer purchase the product. This way, that vendor would receive that product as a valid order. Since only the purchases which are delivered allows rating a vendor, I give the status as 'Delivered'. After setting everything up, I make login with that customer and get the authorization token. Then, I prepared the request body and give purchase_id and the rating_score as the required fields. After making the request, if nothing goes wrong, the response should have a status code HTTP_201.

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

5 Deployment, Dockerizing, and CI/CD processes

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

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

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

6 API Documentation

Swagger url: bupazar API

6.1 Authentication

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

Table 4: Authentication Endpoints

6.2 Alarms

Functionality	URL	Method	Parameters	Responses
Deleting price alarm of a product for authenticated customers	api/alarms/delete-price-alarm/	POST	product_id*	success*
Setting price alarm with giving price for a product to notify requested customer when price of product goes down below giving price	api/alarms/set-price-alarm/	POST	product_id*, price*	success*
List all price alarms of requested customer	api/alarms/my-price-alarms/	GET	-	customer (username), product: Product, price*

Table 5: Alarms Endpoints

6.3 Users

Functionality	URL	Method	Parameters	Responses
List all users	/api/users	GET	-	id, email, username, first_name, last_name, is_vendor, is_customer, is_active, is_staff, address
Retrieve user details by user id	/api/users/{id}	GET	-	id, email, username, first_name, last_name, is_vendor, is_customer, is_active, is_staff, address
Retrieve vendor details by username	api/users/vendor/details	POST	vendor_username*	email*, username*, first_name*, last_name*, address*, rating*, products*: [Products]

Table 6: User Endpoints

6.4 Products

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

Table 7: Product Endpoints

6.5 Comments

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

Table 8: Comment Endpoints

6.6 Credit Cards

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

Table 9: Credit Card Endpoints

6.7 Purchases

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

Table 10: Purchase Endpoints

6.8 Orders

Functionality	URL	Method	Parameters	Responses
Make purchase	/api/orders/make_purchase	POST	-	success
Get a customer's orders	/api/orders/customer-orders	GET	-	order_id*, [Purchases]
Look whether a product is purchased by customer	/api/orders/customer-purchased	POST	product_id*	true/false
Vendor update status of an order	/api/orders/update-status	POST	order_id*, status*	success
Retrieve all purchases which contain requesting vendor' products	api/orders/vendor-orders/	GET	-	[Purchases]
Cancel purchase for vendor which sends request	api/orders/vendor-cancel/	POST	purchase_id	success
Cancel order for customer which sends request	api/orders/customer-cancel/	POST	order_id*	success
Add shipment for a purchase in an order	api/orders/add-shipment/	POST	purchase_id*, cargo_company*	cargo_no*
Get shipment information for a purchase in an order	api/orders/get-shipment/	POST	purchase_id*	id, purchase, date, cargo_no*, cargo_company*
Add a rating to a vendor	api/orders/add-vendor-rating/	POST	purchase_id*, rating_score*	success
Calculate average vendor rating in a product page	api/orders/avg-rating-product-page/	POST	product_id*	score*
Calculate average vendor rating in a vendor profile page	api/orders/avg-rating-profile-page/	POST	-	score*

Table 11: Order Endpoints

6.9 Chats

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

Table 12: Chat Endpoints

6.10 Cart

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

Table 13: Cart Endpoints

6.11 Favorites

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

Table 14: Favorites Endpoints

6.12 Product Lists

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

Table 15: Product Lists Endpoints

6.13 Notifications

Functionality	URL	Method	Parameters	Responses
List all notifications of requested user	api/notifications/my/	GET	-	[id, text*, notificationType, user, createdAt, product, order*, isSeen]
Setting notification as seen	api/notifications/set_seen/	POST	notification_id*	success*
List all notifications in database	api/notifications/all/	GET	-	[Notifications]

Table 16: Notification Endpoints

6.14 Admins

Functionality	**URL**	**Method**	**Parameters**	**Responses**
Is Admin	/api/admin/is_admin	GET	None	true/false
Assign Admin	/api/admin/assign_admin	POST	username	success
Delete Comment	/api/admin/delete_comment	POST	comment_id	success
Delete User By Comment Id	/api/admin/delete_user_by_comment_id	POST	comment_id	success
Delete User	/api/admin/delete_user_by_username	POST	username	success

Table 17: Admin Endpoints

7 Requirements

Glossary

- **Amazon-EC2:** by allowing users to rent virtual computers on which to run their own computer applications.
- **Admin User:** A person who is responsible for system sustainability and management in general
- **Guest User:** A person who does not have an account and have restricted access to the application
- **Vendor:** A person who supply products to customers.
- **Search:** A tool to help the users find the relevant contents for given input words in the application
- **Server:** something that shares data or resources among multiple clients or performing computation for a client
- **Sign In:** Entering to the application by providing correct email and password
- **Sign Up:** Creating an account to be a member of application
- **User:** A person that interacts with the application
- **Docker:** It is a tool designed to make it easier to create, deploy, and run applications by using containers.
- **Public Profile:** A profile type which shall be visible to all users and guests.
- **Private Profile:** A profile type which shall be visible to limited users and guests.

1. Functional Requirements

- **1.1. User Requirements**
  - **1.1.1. User Basics**
    - **1.1.1.1. Sign Up**
      - **1.1.1.1.1.** Guests shall be able to sign up as a customer by providing their name, surname, e-mail address, choosing a password and a user name.
      - **1.1.1.1.2.** Guests should be able to sign up with their Google or Facebook account.
      - **1.1.1.1.3.** Guests shall be able to sign up as a vendor by providing their name, surname, e-mail address, choosing a password, a user name and their location (at least one).
      - **1.1.1.1.4.** Guest user shall be able to sign up while the purchase process.
    - **1.1.1.2. Sign In**
      - **1.1.1.2.1.** Users shall be able to sign in with their e-mail or user name and password.
      - **1.1.1.2.2.** There shall be a "forgotten password" button in case the user forgets the password.
      - **1.1.1.2.3.** Users should be able to sign in with their Google or Facebook account.

* **1.1.1.3. Profile**

  - **1.1.1.3.1.** Vendor and customer users shall have a profile page.
  - **1.1.1.3.2.** Vendor's profile shall contain vendor's rating, types of the products that vendor sell, products' prices and vendor's contact info.
  - **1.1.1.3.3.** Customer's profile shall contain user's address, user's info (age, sex etc.), the saved credit cards, user's previous orders and user's assessments about the vendors and the products.

* **1.1.1.4. User Types**

  - **1.1.1.4.1.** Guest: A user who is using the platform but has not signed up yet.
  - **1.1.1.4.2.** Customer: A user who is able to use all the functionality of the system other than selling products.
  - **1.1.1.4.3.** Vendor: A user who is able to sell products on the platform.
  - **1.1.1.4.4.** Admin: An administrative user who is able to ban users and manage the whole system processes.

– **1.1.2. User Interactions**

* **1.1.2.1. Rating**

  - **1.1.2.1.1.** Customers shall be able to rate the products they bought, out of 5.
  - **1.1.2.1.2.** Customers shall be able to rate the vendors whom they bought a product from, out of 10.

* **1.1.2.2. Commenting on Products**

  - **1.1.2.2.1.** Users shall be able to comment on the products that they have already bought.
  - **1.1.2.2.2.** Users shall be able to choose to anonymize their comment prior to posting.

* **1.1.2.3. Search**

  - **1.1.2.3.1.** Users shall be able to search for products and vendors. See 1.1.5 for more info.

* **1.1.2.4. Communication**

  - **1.1.2.4.1.** Users shall be able to communicate with vendors through direct messaging.

* **1.1.2.5. Carts**

  - **1.1.2.5.1.** Users shall be able to add products to their cart.
  - **1.1.2.5.2.** Users shall be able to remove products from their cart.

* **1.1.2.6. Lists**

  - **1.1.2.6.1.** Customers shall be able to create private lists.
  - **1.1.2.6.2.** Customers shall be able to add products to their lists.
  - **1.1.2.6.3.** Customers shall be able to delete products from their lists.
  - **1.1.2.6.4.** Customers shall be able to delete their lists.

* **1.1.2.7. Orders**

  - **1.1.2.7.1.** Customers shall be able to make and cancel orders and follow their orders. See 1.1.3 and 1.1.4 for more info.

– **1.1.3. Vendor Specific Interactions**

* **1.1.3.1. Adding New Product**

  - **1.1.3.1.1.** Vendors shall be able to add a new product to the platform with the necessary information of products(price, condition, category, etc.)
  - **1.1.3.1.2.** Vendors shall be able to sell as many products as they want.
  - **1.1.3.1.3.** Vendors shall be able to mention about the stock of a product.

* **1.1.3.2. Communication**

  - **1.1.3.2.1.** Vendors shall be able to communicate with admins about orders.
  - **1.1.3.2.2.** Vendors shall be able to communicate with customers that text themselves through direct message.

* **1.1.3.3. Product Sold by Different Vendors**

  - **1.1.3.3.1.** Vendors shall be able to sell the same product with another vendors.

* **1.1.3.4. Orders**

  - **1.1.3.4.1.** Vendors shall be able to follow all processes about their ordered products as the customers.
  - **1.1.3.4.2.** Vendors shall be able to cancel an order during the order processing stage.

* **1.1.3.5. Recommendation System**

  - **1.1.3.5.1.** Customers shall be able to be recommended based on their interactions on the platform.

– **1.1.4. Customers Specific Interactions**

* **1.1.4.1. Customers' Lists**

  - **1.1.4.1.1.** Customers shall be able to create their own lists and carts.
  - **1.1.4.1.2.** Customers shall be able to add as many products as they want to their

baskets.

        * **1.1.4.2. Communication**

            · **1.1.4.2.1.** Customers shall be able to communicate with vendors through direct messaging.

        * **1.1.4.3. Orders**

            · **1.1.4.3.1.** Customers shall be able to follow their orders via the orders page.
            · **1.1.4.3.2.** Customers shall be able to see their active and delivered orders with sufficient information about the orders.
            · **1.1.4.3.3.** Customers shall be able to cancel their active orders.
            · **1.1.4.3.4.** Customers shall be able to return their delivered orders.

        * **1.1.4.4. Notifications**

            · **1.1.4.4.1.** Customers shall be able to be notified about changes in products that are in their lists or favorites.
            · **1.1.4.4.2.** Customers shall be able to set alarm for a certain price and choose to be notified if the price of product goes below the chosen price.

        * **1.1.4.5. Recommendation System**

            · **1.1.4.5.1.** Customers shall be able to be recommended based on their interactions on the platform.

    – **1.1.5. Searching/Listing**

        * **1.1.5.1. Search Bar**

            · **1.1.5.1.1.** Users shall be able to search for both product pages and vendor profiles using the search bar. Search results should also include semantic results, similar vendors and similar products.

        * **1.1.5.2. Filter**

            · **1.1.5.2.1.** Users shall be able to filter products based on brand, vendor, price range, rating, and discount rate.

        * **1.1.5.3. Sorting**

            · **1.1.5.3.1.** Users shall be able to sort products based on bestsellers, newest arrivals, price, number of customer reviews, rating, and number of comments.

- **1.2. System Requirements**

    – **1.2.1. Security**

        * **1.2.1.1. Sign Up**

            · **1.2.1.1.1.** System shall send a verification email when a customer signs up.
            · **1.2.1.1.2.** System shall allow the passwords to be at least 8 characters which must include at least one uppercase, one lowercase character and a number
            · **1.2.1.1.3.** System shall ask customer to enter their password twice while signing up.
            · **1.2.1.1.4.** System shall verificate vendors when they sign up.

        * **1.2.1.2. Sign In**

            · **1.2.1.2.1.** System shall allow customers to enter wrong password only 3 consecutive times. After that, system shall block the account and send an email to customer for giving information about the trial.
            · **1.2.1.2.2.** System shall remind customers to change their passwords regularly.

        * **1.2.1.3. Payment**

            · **1.2.1.3.1.** System shall ask all credit card information before any transaction if no credit card information is given or no card has already been saved. The information shall include proper card number with 16 characters, expiration date and CVV (3 character security code).

    – **1.2.2. Performance**

        * **1.2.2.1.** The system shall be able to respond to requests within 10 ms in general. Maximum response time should not exceed 1s.
        * **1.2.2.2.** The system shall cache frequently accessed contents to deliver faster and reduce response time.

2. Non-Functional Requirements

- **2.1. Protocol**

    – **2.1.1.** The system shall meet the standards written in the W3C protocol.

    – **2.1.2.** The system shall follow the W3C Activity Streams protocol.

- **2.2. Ethical Issues**

    – **2.2.1.** When users sign up they must accept Privacy Policy.
    – **2.2.2.** When users sign up they must accept Terms of Use.
    – **2.2.3.** User data shall be processed according to the rules specified by GDPR and KVKK.

- **2.3. Deployment**

    – **2.3.1. Deployment on a Server**

        * **2.3.1.1.** The system shall be deployed on Amazon EC2 server

- **2.3.2. Docker**
    - **2.3.2.1.** The system shall has docker technology to ease the development and deployment processes.

- **2.4. Availability**
    - **2.4.1. Web Access**
        - **2.4.1.1.** The system shall have a Web application that supports Chrome browser that supports all versions since 2011.
        - **2.4.1.2.** The system shall have a Web application that supports Firefox browser that supports latest version.
        - **2.4.1.3.** The system shall have a Web application that supports Safari browser that supports all versions since 2014.
        - **2.4.1.4.** The system shall have a Web application that supports Opera browser that supports all versions since 2016.
    - **2.4.2. Mobile Access**
        - **2.4.2.1.** The system shall be compatible with Android 5.1 or higher version.

- **2.5. Accessibility**
    - **2.5.1.** The system shall maintain itself every Monday between 3.00 am and 3.05 am regularly.
    - **2.5.2.** Every user shall receive an alert message 1 hour before the maintenance of the system starts.

8 Design Documents

We have implemented the design documents related to our project during the CmpE 352 course, and we uploaded the documents to our GitHub wiki page. The design documents can be found using the following links:

- **Use Case Diagram:** Click here to display our use case diagram.
- **Class Diagram:** Click here to display our class diagram.
- **Sequence Diagrams:** Click here to display our sequence diagrams.

9 Mockups

We have created the mockups related to our project during the CmpE 352 course, and we uploaded the documents, including the mockup images and the user scenarios back then to our GitHub wiki page. The design documents can be found here.

10 Project Plan

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

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

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

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

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

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

11 Assessment of the Customer Presentation

First of all, we manage to implement all the main functionalities of the project. Comparing final milestone to previous ones, this one was undoubtedly the hardest to put together. This time along with dealing new functionalities we also had to solve the previous and new bugs to come up with a final deliverable. For the final milestone we implemented vendor products page, vendor edit page, completed messages page, google map, recommendation system, notification system, forget password and orders page both for customer and vendor along with further features like alarm. In addition to these we add vendor public view page, admin after and currency feature in product page the presentation.

In frontend part, it mainly progressed smoothly. The customer in the presentation started with going to the bag which was recommended through the recommendation system. After going to the product page of the bag, customer changed previously set alarm for the product. Our alarm feature was well received since it was a good touch for notification types. For the product page it was suggested that it would be good to display several currencies (we implemented this after the presentation). We also corrected the bug of not giving rating after giving comment for this milestone. In purchasing process, it had told us to add a user agreement also which we dealt with this issue immediately after on (we also add the agreement in sign up too). The order pages functionalities and display were also well praised but there was a discussion about whether the ordering should be from oldest to newest or not in vendor's page which was later agreed on being a design choice. Notifications and searching was the main target of the questions after the presentation which explained more clearly to the customer viewers showing semantic search ability and all the notifications types in notification page.

In mobile part the presentation also went well. Since vendor and customer presented at the same time by using two telephone emulators it gave the presentation a dynamic pacing. Recommendation system, searching, ordering, and messaging were showcased in the presentation. There was criticism on messaging not being clear in terms of who is messaging. Moreover, the unread messages weren't being displayed which was a concern. Additionally, there was a bug on the delivery status display which was handled later. The purchase part had a checkbox but not a formal agreement which was also pointed out. Other than these issues; purchasing, ordering, searching, and recommending all well received generally.

When we look at the whole presentation, we can say that we showed what we offered without major mistakes for Final Milestone. The time management and communication problem were resurfaced in this milestone because of the limited time and individual workloads. However, we manage to come up with suitable work divisions between members. Since we had many features to implement for this milestone, of course we faced with more difficulties. Nonetheless, in the end, we can say that for the final milestone, the presentation and the implementation were a success.

12 System Manual

This document describes how to build, configure and deploy Bupazar app.

12.1 Requirements

- Docker[1]
- docker-compose[2]
- git[3]

12.2 How to Build & Deploy

12.2.1 Clone the repository and navigate to the sources root.

- git clone https://github.com/bounswe/bounswe2020group5
- cd bounswe2020group5/app

12.2.2 Build & run

- docker-compose up -d –build

This command run both frontend and backend on your machine. To see the frontend just type http://localhost:3000/.

12.3 If you want to run the app separetely without docker you need to follow these steps.

12.3.1 To run backend:

- cd bounswe2020group5/app/backend/
- Type virtualenv venv to create virtual environment venv.
- In order to activate virtual environment venv,
  - In Linux and macOS, type source venv/Scripts/activate
  - In Windows, type venv\Scripts\activate
- Install required python packages via typing pip3 install -r requirements.txt
- python3 bupazar_config.py 'tM6caMoe7fGqdZejfdLjHSyFmgCCb71sQ2XT1yV3n30='
- python3 manage.py makemigrations
- python3 manage.py migrate
- python3 manage.py runserver

12.3.2 To run frontend:

- cd bounswe2020group5/app/frontend/

[1] <https://docs.docker.com/install/>
[2] <https://docs.docker.com/compose/install/>
[3] <https://git-scm.com/>

- npm install
- cd src/login
- node secrets.js 'tM6caMoe7fGqdZejfdLjHSyFmgCCb71sQ2XT1yV3n30='
- npm start

12.3.3 To run Mobile

- Bupazar android project uses the Gradle build system.
- To run this project, open the project in Android Studio. Then you can run the project gradle run command, or you can build apk directly using the build apk command. (Do not extract apk in "release" mode, google sign-in will need a new sha1 key in this case)

Dependencies

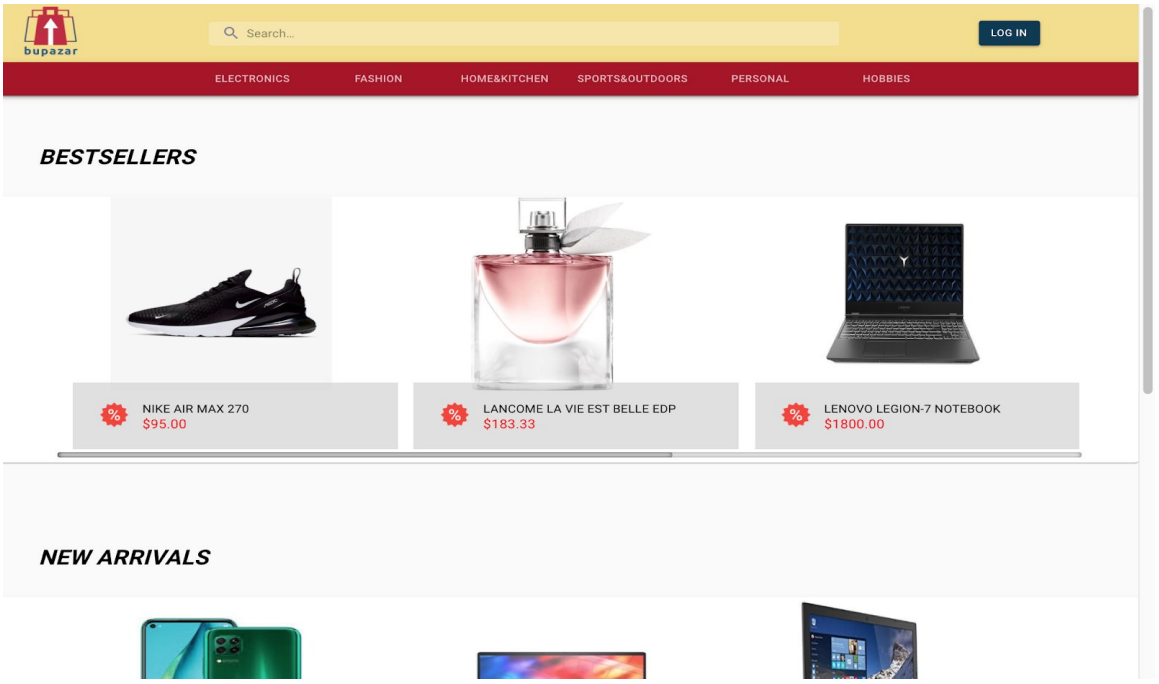
```
- Gradle Version 4.1.0
- Kotlin Version = 1.4.10
- Android Studio 3.x
```

BUPAZAR WEBSITE
USER MANUAL

http://100.25.223.242:3000/
http://localhost:3000/

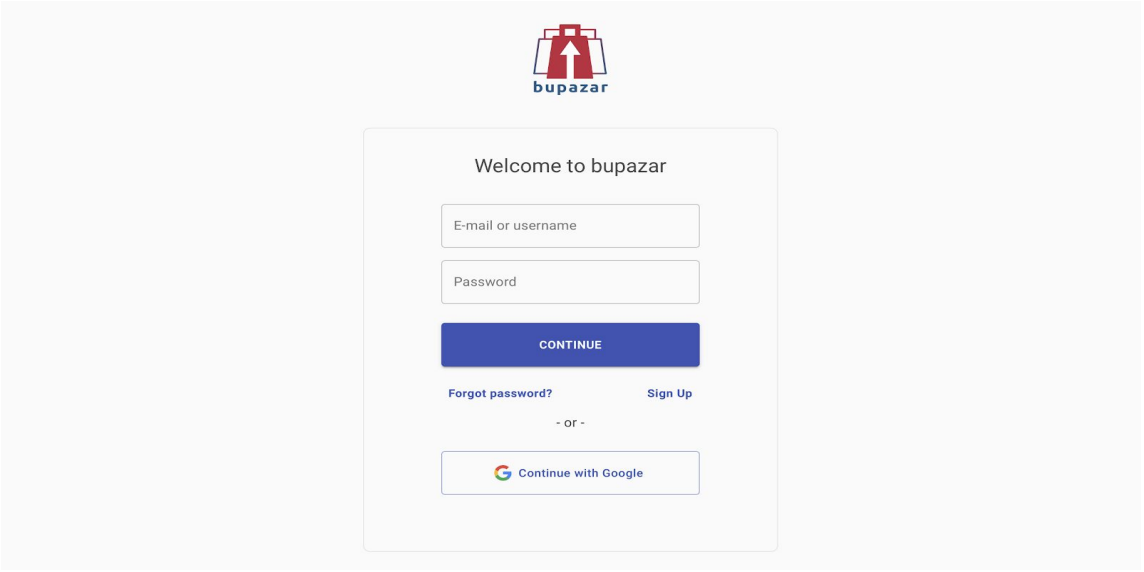
1. COMMON PAGES

1.1. Home Page



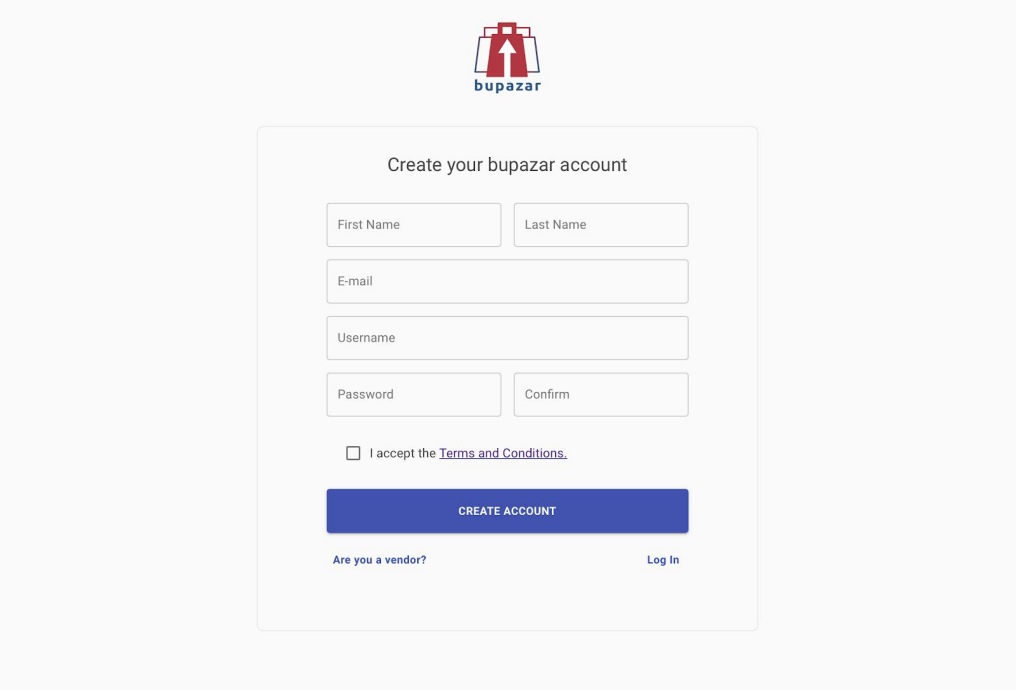
When users first enter the website, the homepage welcomes them. Bestsellers, New arrivals and Trending products are displayed here. Users can login as a customer or a vendor from the login button. Otherwise, users can also continue as guests. Users can search for the product they want from the search bar in the navigation bar. By selecting one of the categories from the category bar, they can view the products in specific categories.

1.2. Login

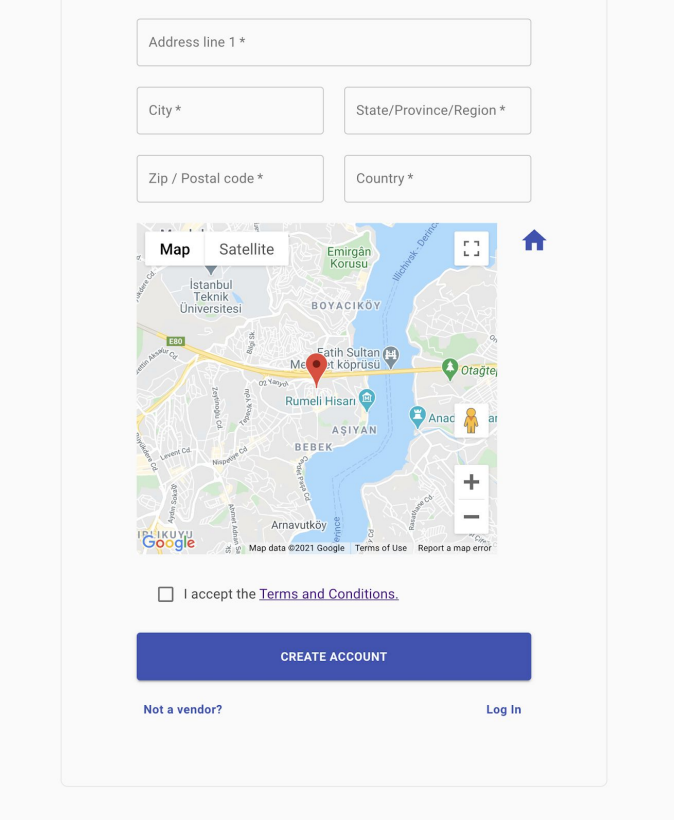


Users can enter their email and password to login. Or they may continue with their Google accounts. If they want to register however, below the continue button there is a sign up button.

1.3. Sign Up

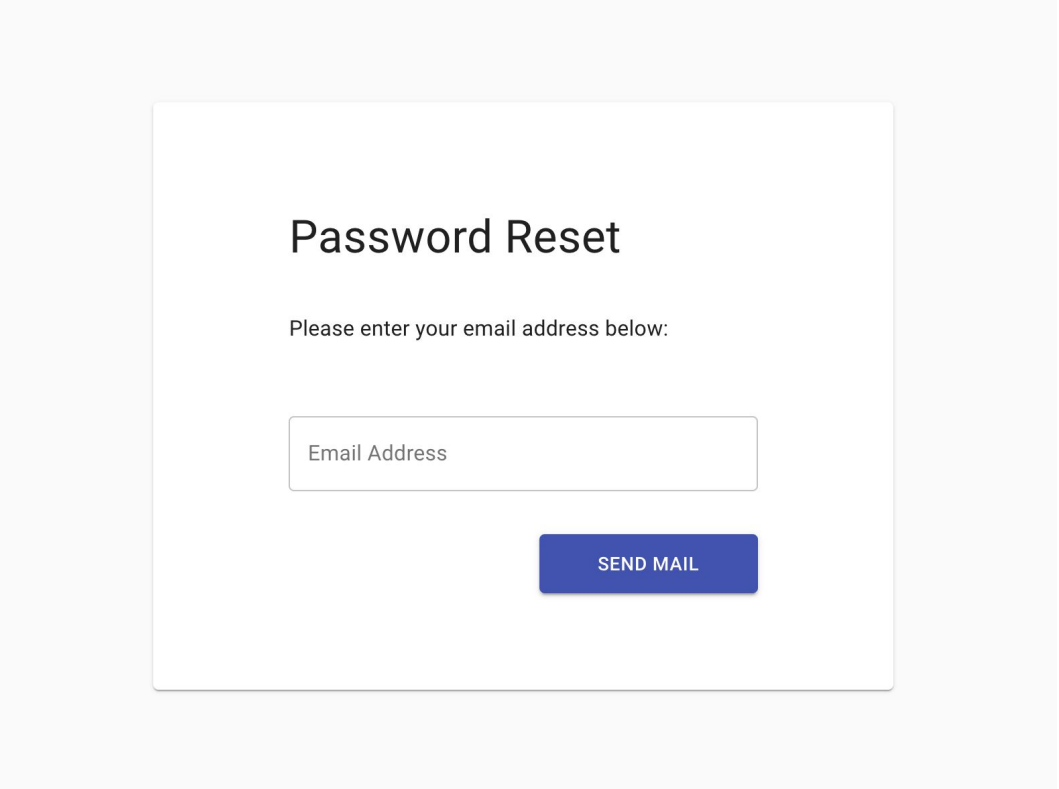


Users may enter required fields and sign up to the bupazar. However, accepting Terms and Conditions are mandatory. If you are a vendor who wants to use bupazar, you may continue with the vendor signup process. The main difference is address is required if you are a vendor. You can enter your current location by clicking the home icon next to the map. Location services must be allowed on the browser for Google Maps to work.

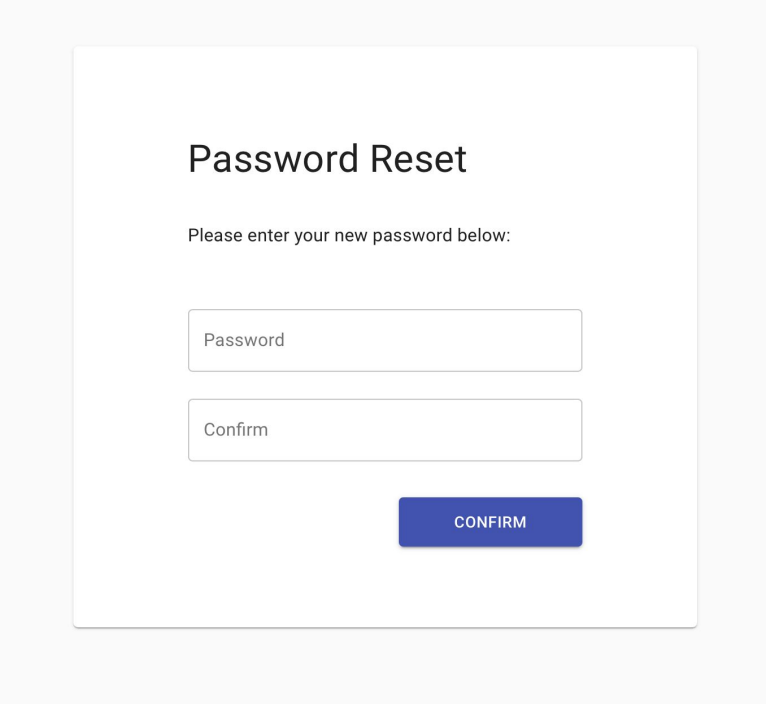


1.4. Forgot Password

If you have an account and want to login, however you forgot your password, then you can click Forgot Password in the Login page.



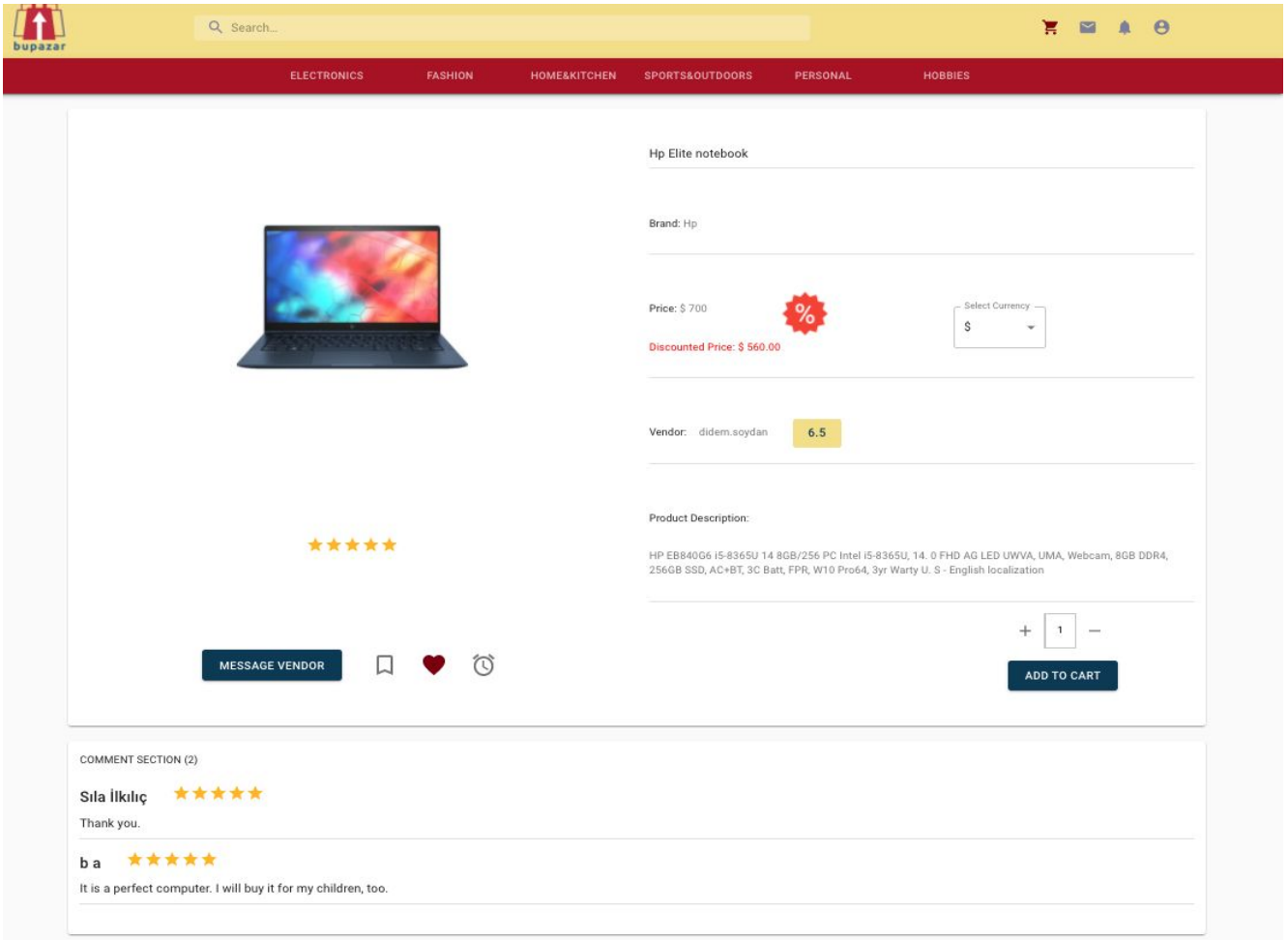
After entering your email address a link will be sent to your account, which you can click to reset your password.



1.5. Product Page

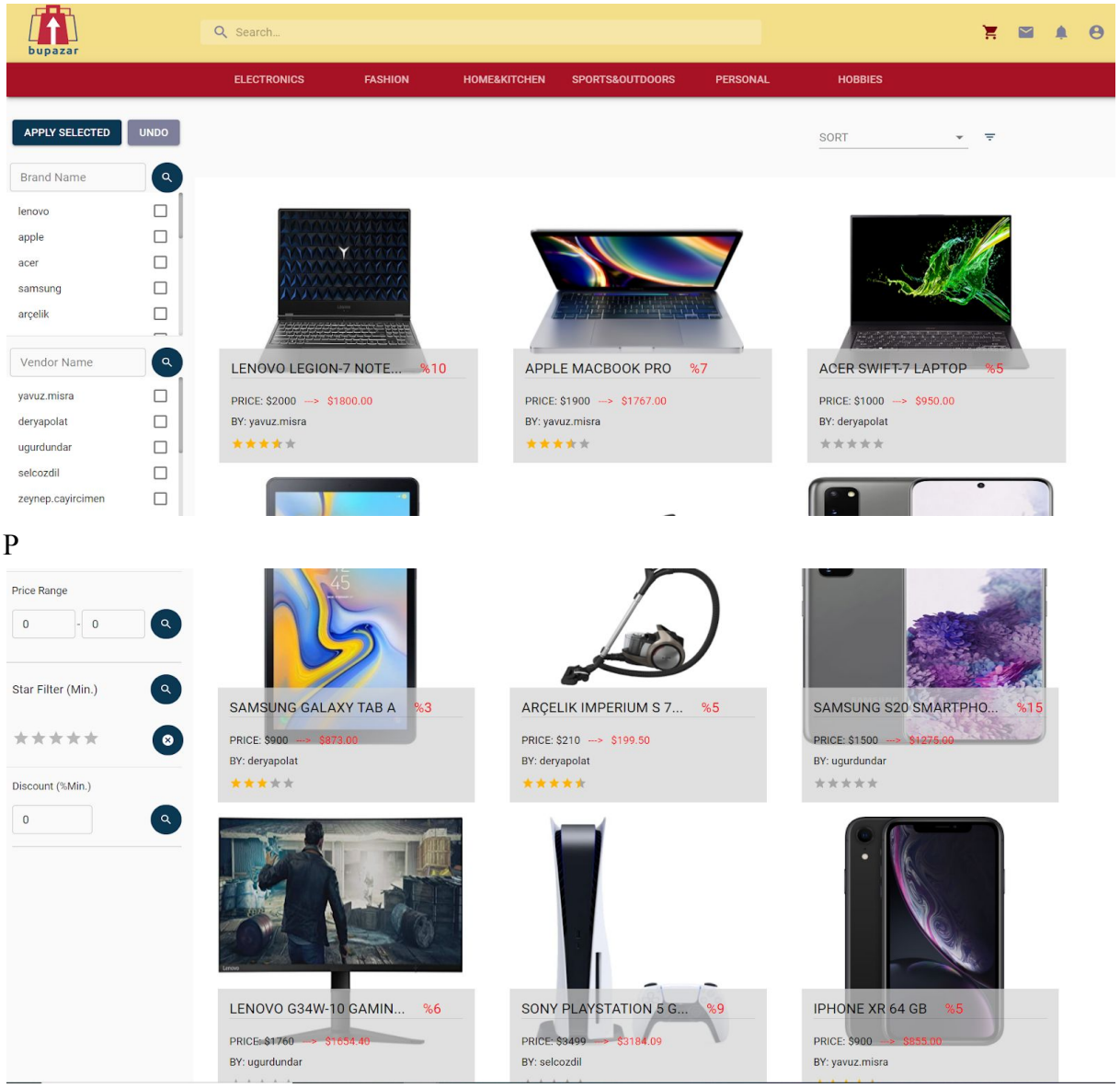
The product page displays the brand, price (in different currencies), vendor name (is clickable and directs to vendor public page) and rating, and product description. This page can be accessed via clicking the product images on the home page, search results, lists, orders, etc.The comment section displays previous comments on the product. Those who choose to be anonymous are shown with only the initials.

Vendors and guests can view only the read-only information regarding the product. A logged-in customer can add this product to the cart with the desired amount. Customers can also send messages to vendors about the product & packaging & shipment. Customers can add this product to their favorites by clicking the heart button. Likewise, by clicking the save button, the product can be added to a new list or a chosen existing list. Additionally, customers can also set an alarm for a price threshold to be notified when there is enough discount. The customers who have bought the product before can give a review for the product with the option of being anonymous. Reviews consist of a product rating and a comment. Accepted rating scores are between 1-5. If not selected, an alert will inform the customer that he/she forgot to give a rating.



1.6. Category & Subcategory Pages

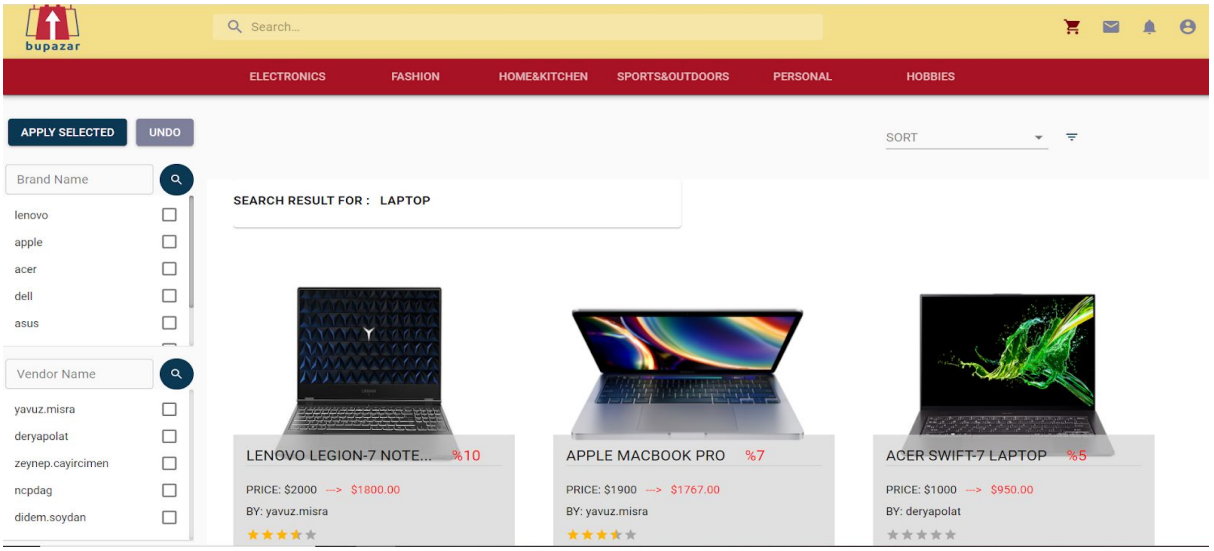
Users can view products in specific categories. To filter these products according to vendor name, brand name, price range, discount rate, users should click on the "apply selected" button after making the selections. Users can also sort the products in that category according to price, number of comments and ratings. For this, users should select one of the options from the "SORT" menu on the top right and click on the icon next to it.

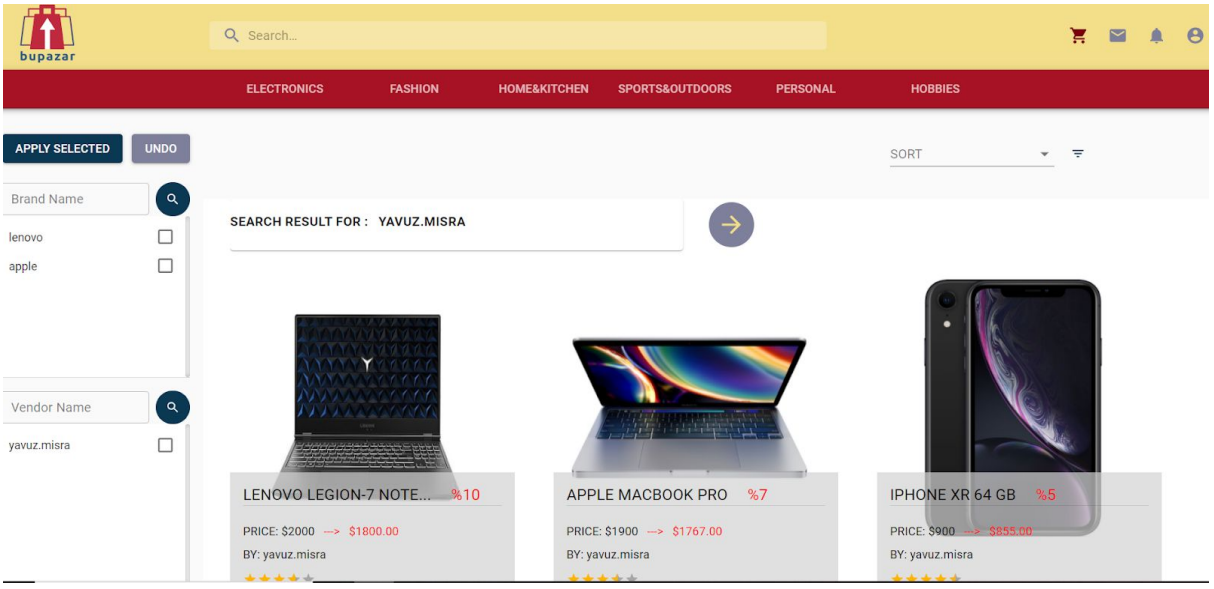


1.7. Search

The time when someone writes something on the search bar located on Navbar and then presses the Enter key, it directly goes to the search page. Search page displays products according to the given search word. In the search page one can filter down the search results and even sort the currently displaying results according to the type that he or she chooses. For sorting, the user needs to press the icon next to the ‘SORT’ bar after selecting the sorting type. For filtering, there are several types that users can do. If a user clicks on the search icon

specific to what he or she wants to filter, it just showcases the results according to that filter disregarding others. However, if a user wants to filter down the results according to more filter types in one go, ‘Apply Selected’ button needs to be clicked. It will display the results according to the intersection of every filter which are correctly selected. If the user wants to get all the filters and sorting undone, the user needs to click the ‘UNDO’ button. In addition, from the search bar one can also search for vendor username. If this happens, an arrow icon shows up in the search page along with the product results. If a user clicks the arrow, he or she is directed to the vendor public view page of that specific vendor. Also, all the products are clickable in the search list in order to go to their product page.



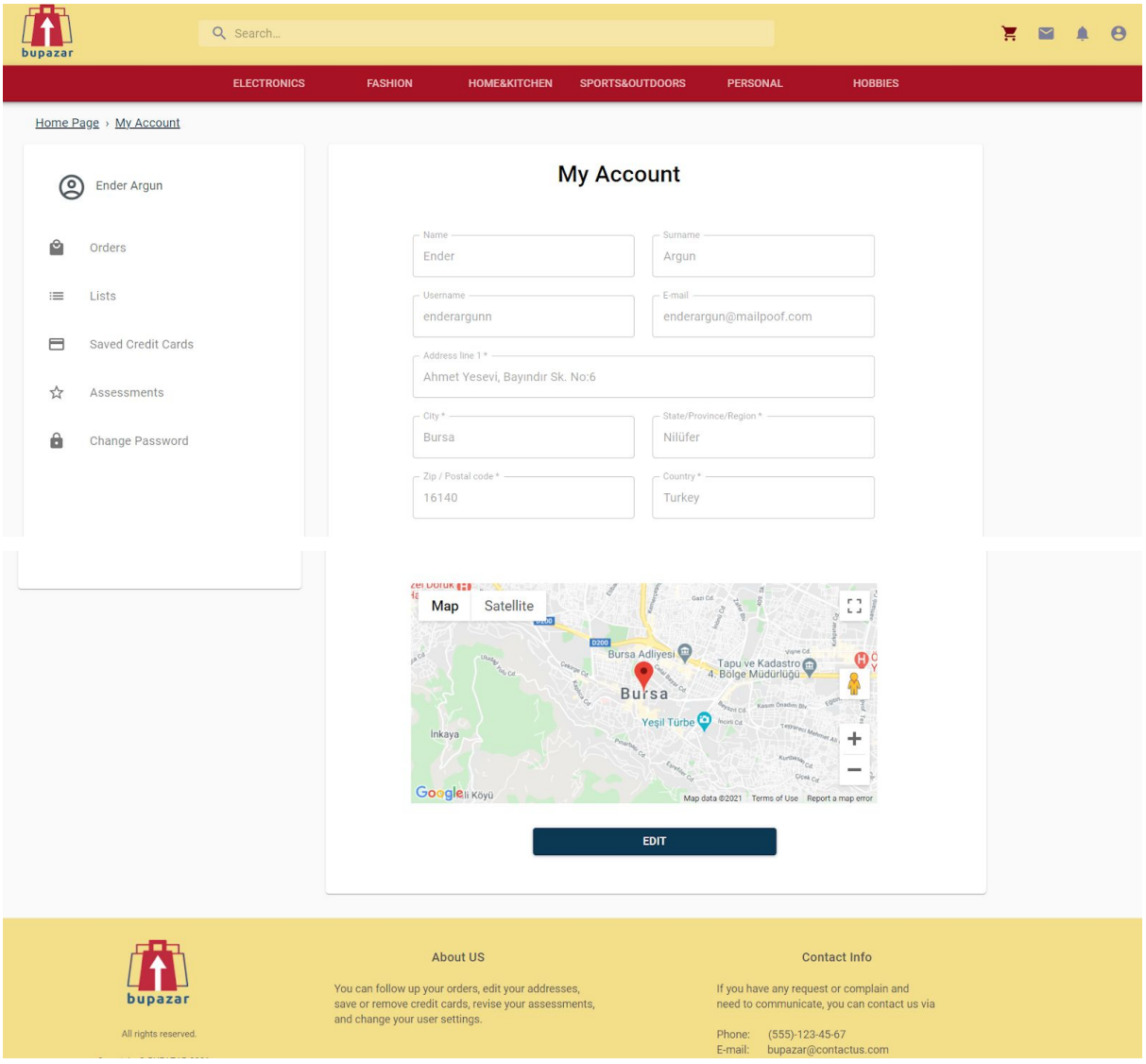


1.8. Profile Page

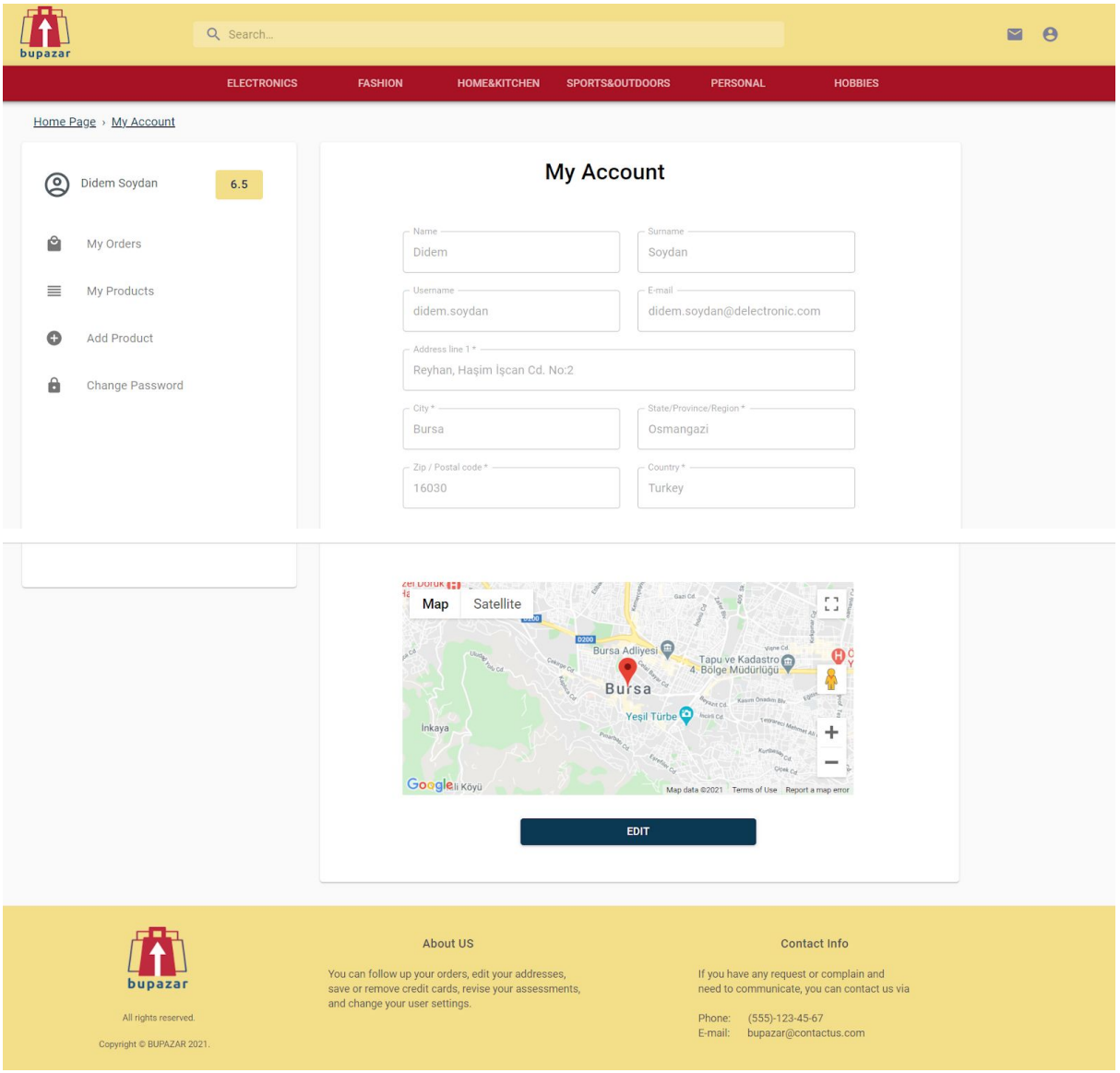
After logging in, users can go to their profile by clicking "My Account" from the user icon. Users can update their information such as name, surname and address on this page. Users can update their address information automatically by using Google maps. In order to set the google map address user needs to press on the house icon when edit is on. Customers can view their past orders, assessments, saved credit cards and the lists they created from the profile page while vendors can view their orders and the products they have added. They can also add new products and view their rating scores.

Note: In order to use Google Maps in deployed website, users need to enable Insecure origins treated as secure with http://100.25.223.242:3000.

Customer Profile

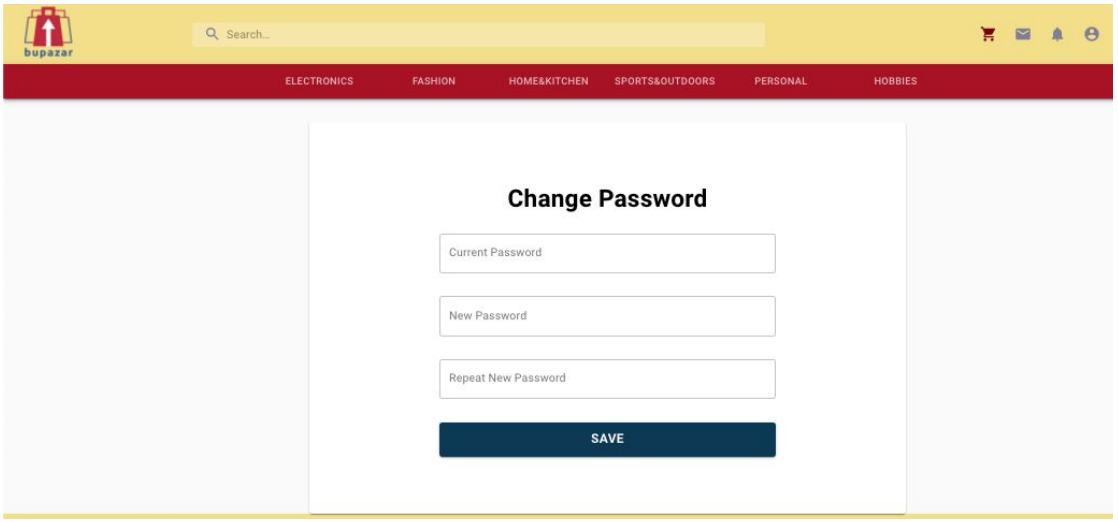


Vendor Profile



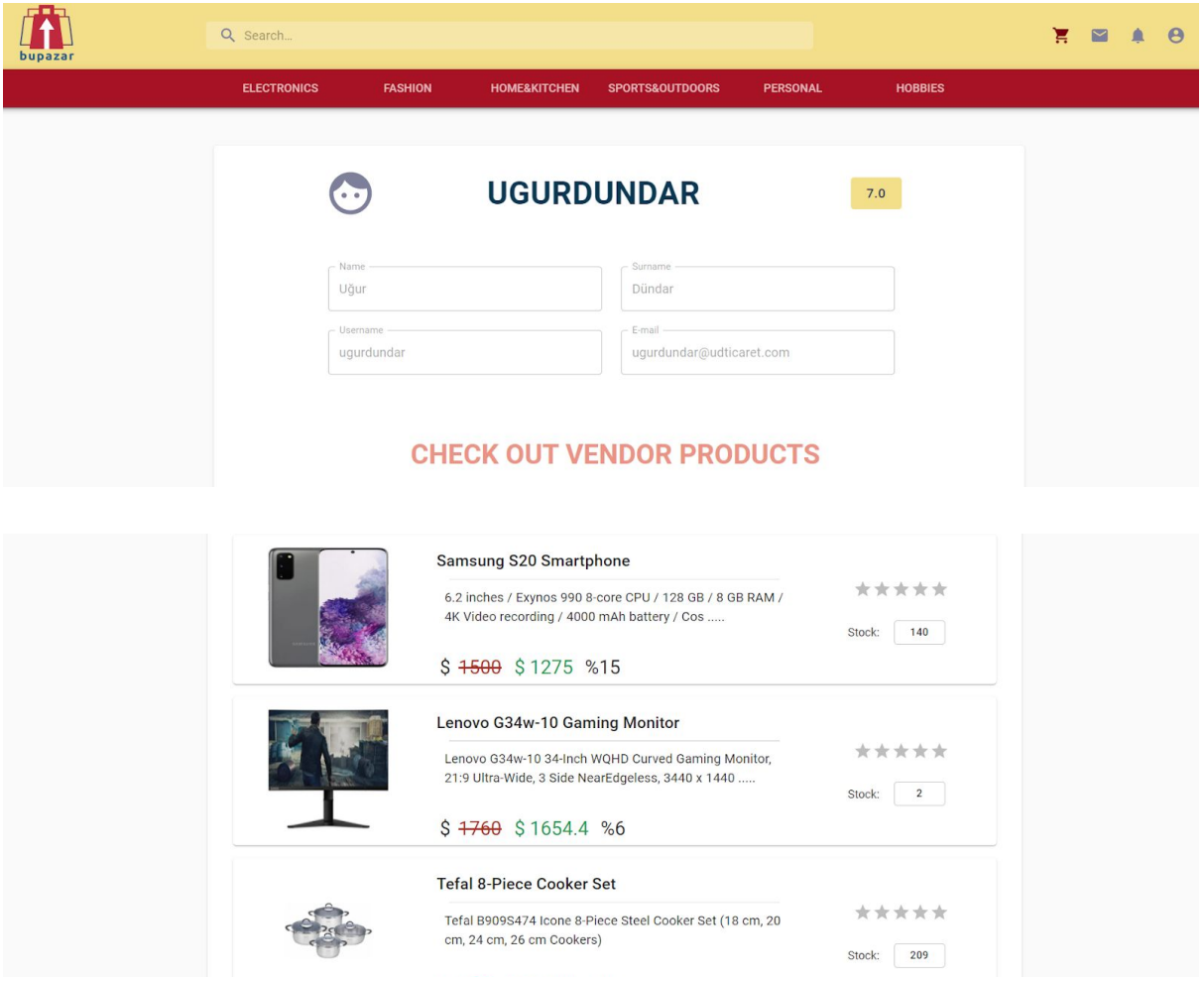
1.9. Change Password

Change password page is accessible from the side-menu at profile pages. All signed up users can change their current password from this page. Validation is checked, covering the following rules: at least 8 characters, at least one uppercase and lowercase letter and number. New password must be different from the current one, and repeated new passwords must match.



1.10. Public Vendor Pages

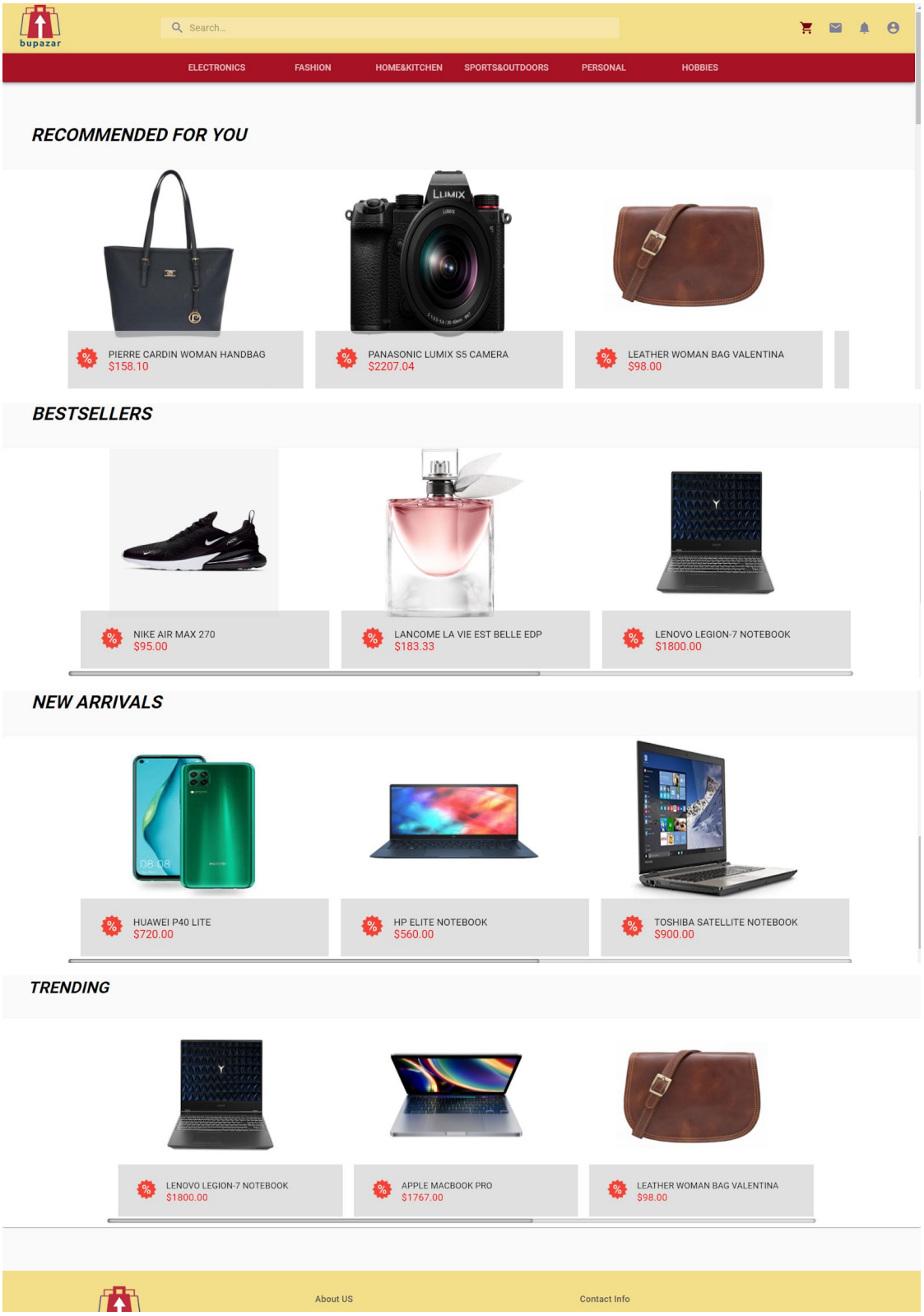
This page is accessible from the search page and product page. It is open to the access of anyone whether being logged in or not. It is the public page of the vendor. It displays the public information of the designated vendor together with the list of products that vendor is selling. One can go to the product page of those products by clicking the picture.



2. CUSTOMER SPECIFIC PAGES

2.1. Home Page (with Recommended Products)

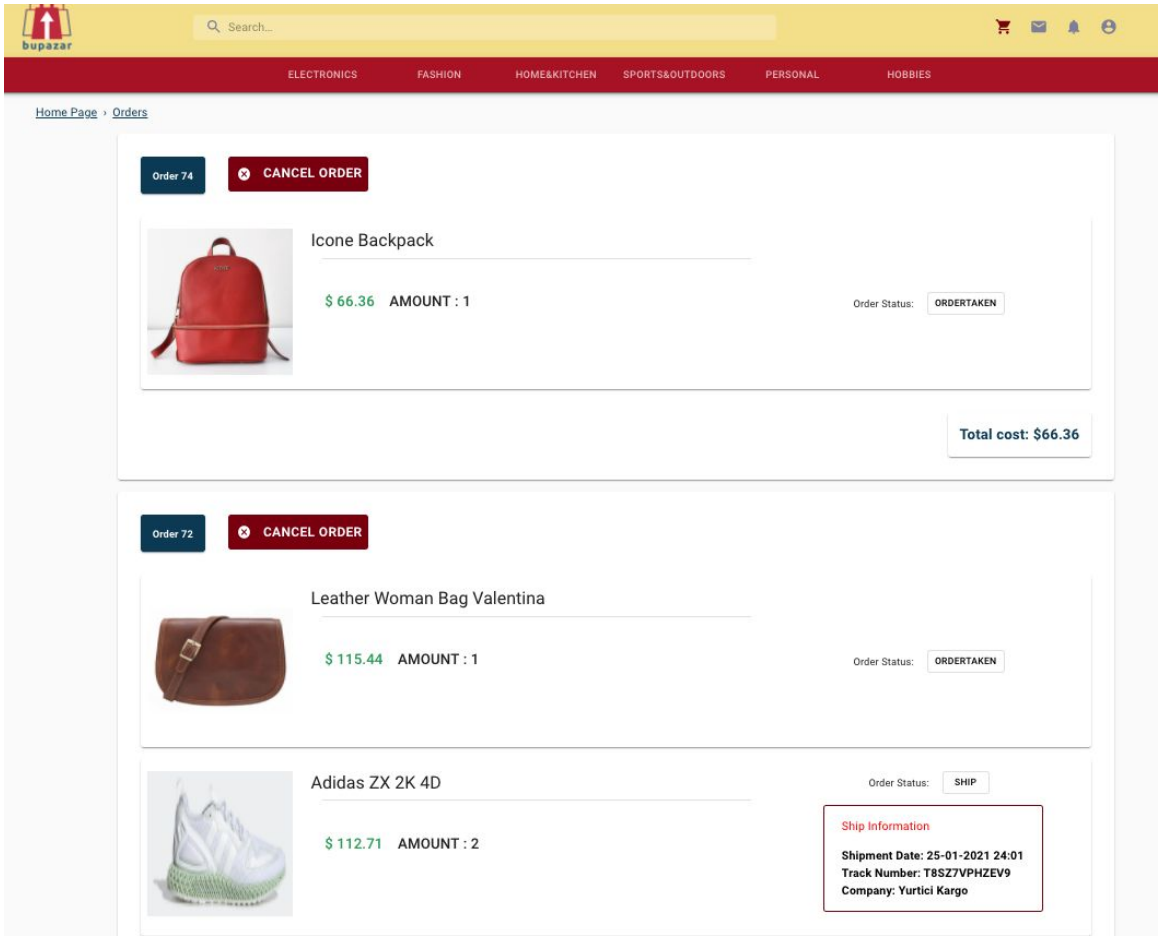
Users who log in as customers can view the products recommended for them in addition to bestsellers, new arrivals and trendings. Recommended products are similar to the purchases made by the customers and the products on their list. If the customer is new to the system, the recommended products are the same as the products in bestsellers.

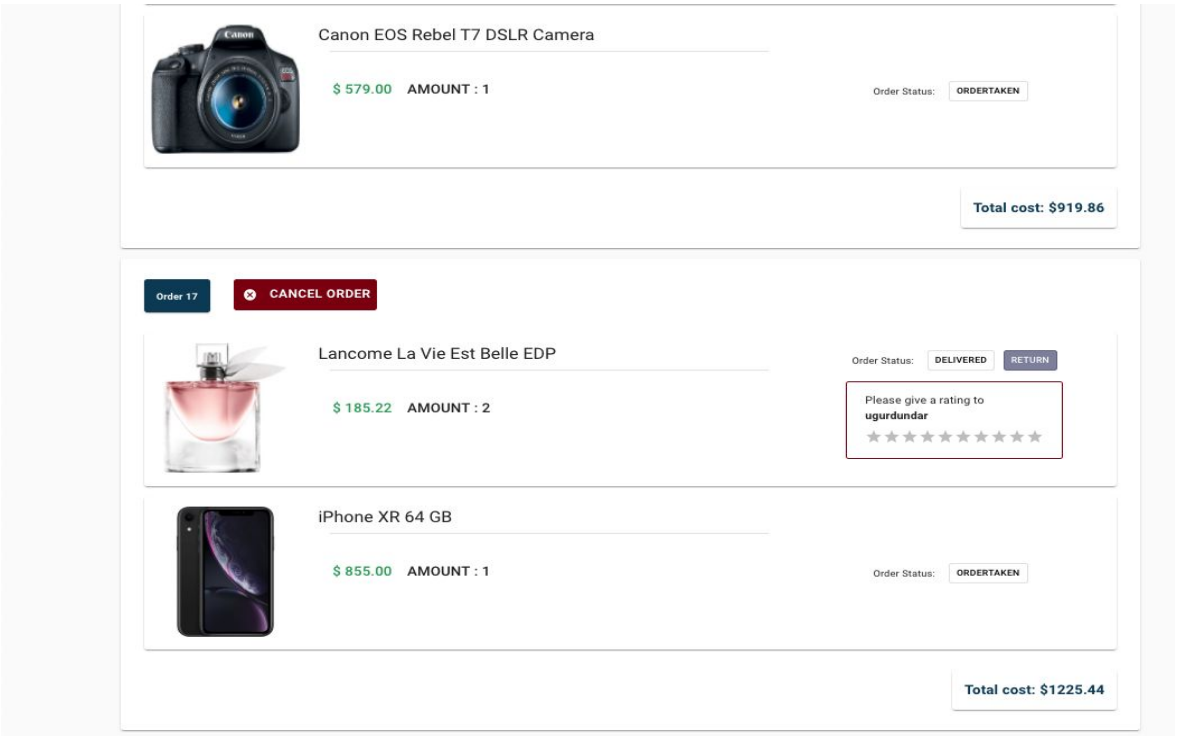


2.2. Orders

Customers can access their previous orders page by clicking on My Orders at the drop-down menu of user icon at the navigation bar. This page is also accessible from the Orders tab of the side menu at profile pages.

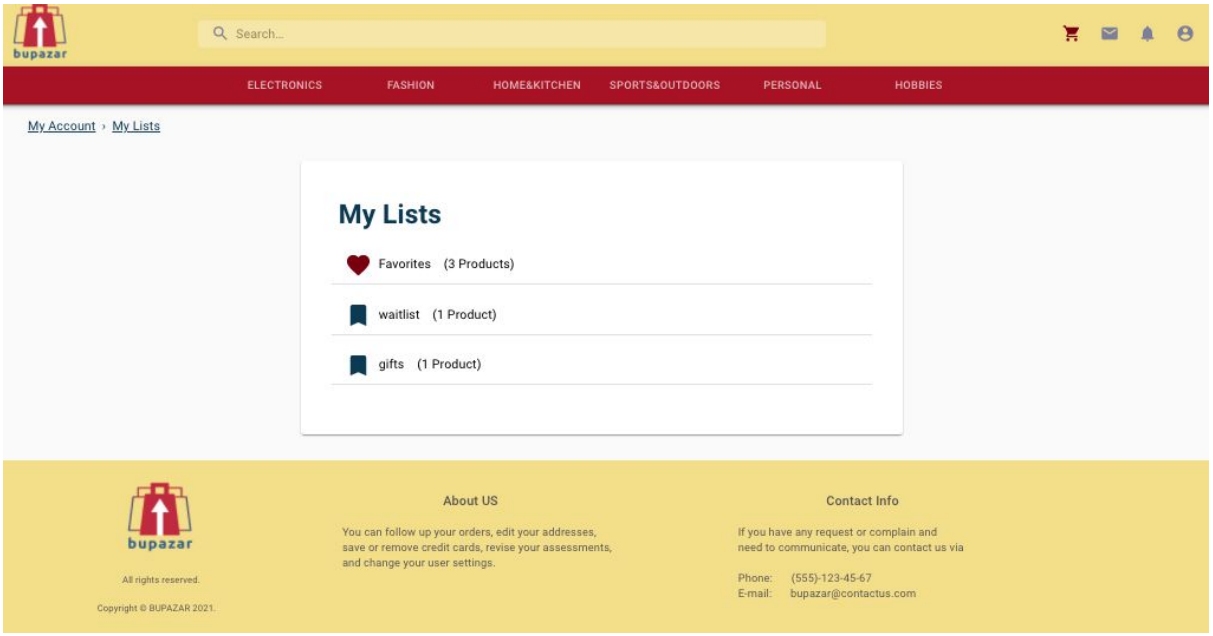
This page displays all orders of the customer in most-recent to least-recent order, separate purchases in the order, with the amount of which product was bought, statuses of orders, and the total cost. Our terminology defines order as consisting of multiple types of products, i.e. the accepted cart. Purchase is accepted as an individual product in the order, its amount may be more than one. If a purchase is in the OrderTaken or Preparing stages, customers can cancel it. One important note is that in our platform, orders are taken as a whole, so there is no option to cancel a specific purchase of the order. Cancel order will cancel all of the purchases that can be cancelled(in OrderTaken or Preparing statuses), and other purchases will not be affected. If the purchase is at Shipment stage, corresponding shipment information(shipment date, tracking number, company) is displayed. If the purchase is delivered, customers can give a rating to the vendor if they haven't before, otherwise, their previous rating is displayed as read-only. Additionally, by clicking on return, customers have the option to return the product with an assigned return code and shipment company, if they want to. Status VCancelled indicates that the purchase was cancelled by the vendor, and CCancelled indicates that the purchase was cancelled by the customer.





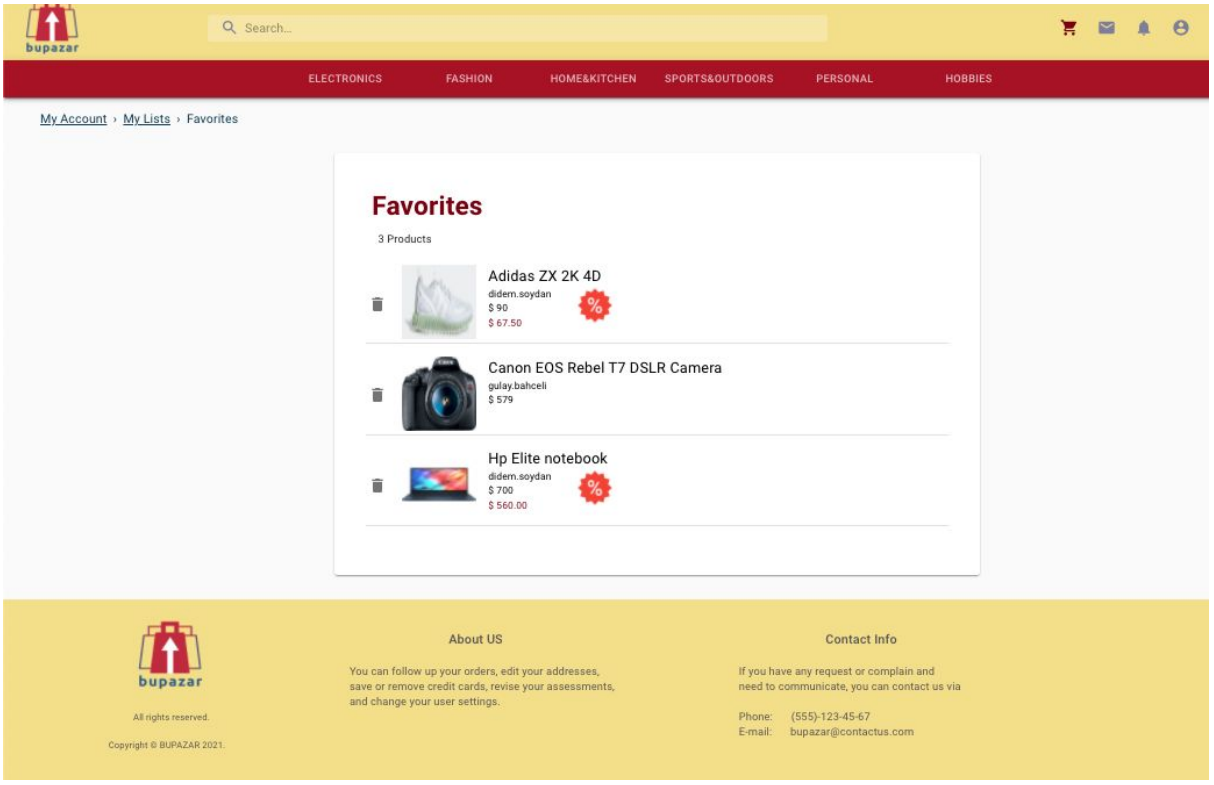
2.3. My Lists

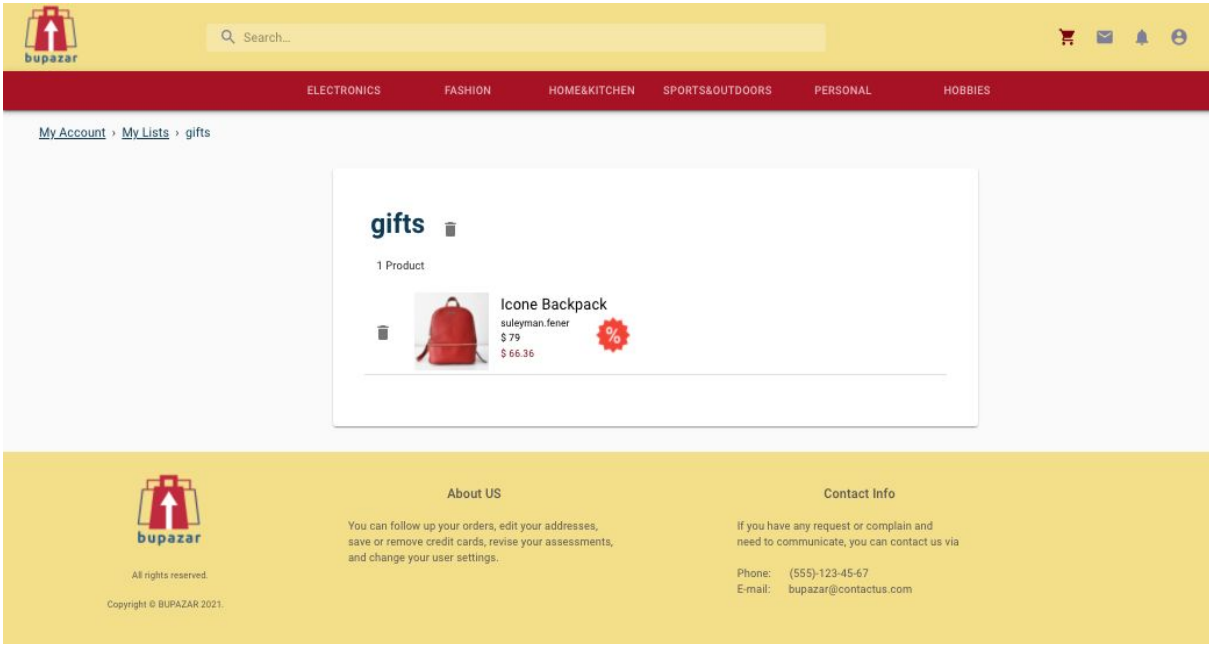
Customers can view their existing lists via clicking My Lists item at the drop-down menu of user icon at the navigation bar. Customers can also access this page from the side menu at their profile page. This page lists all lists of the customer with the number of products in it. Favorites list is initialized by default when a customer signs up. Therefore, if the customer has not created any list, he/she will only see the Favorites here. If clicked on the list name, the customer is directed to products in list page.



2.4. Products in list

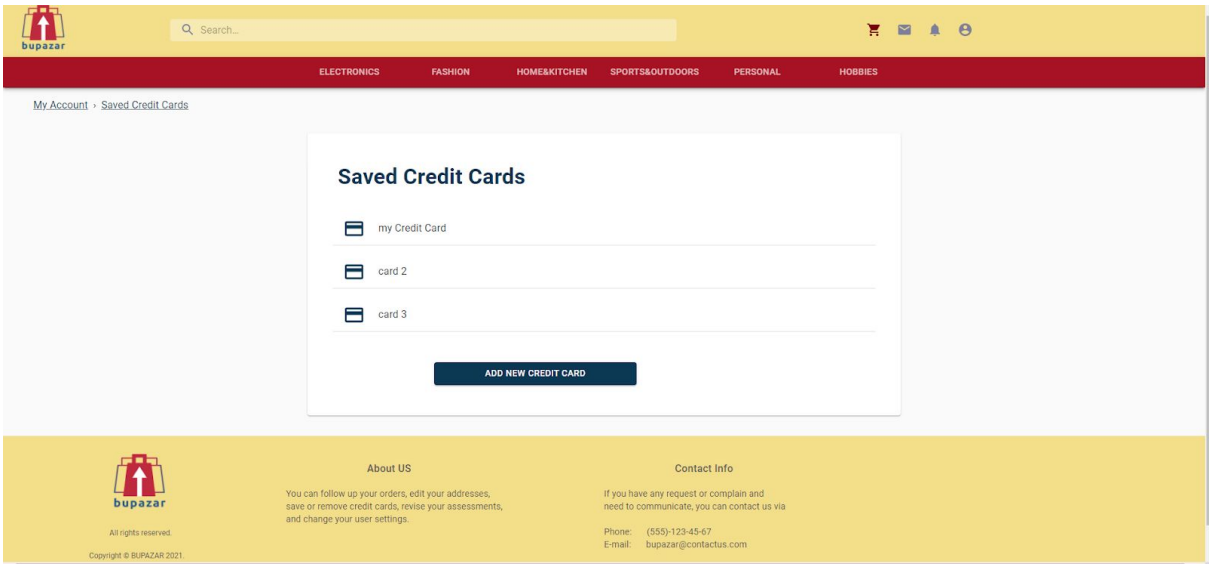
On this page, customers can view the products they have added to the current list before. Additionally, if clicked on the items, they are directed to the product pages. Customers can delete the list or any product in it by clicking on the delete icon buttons. Since the Favorites list exists by default, deletion of it is not permitted.

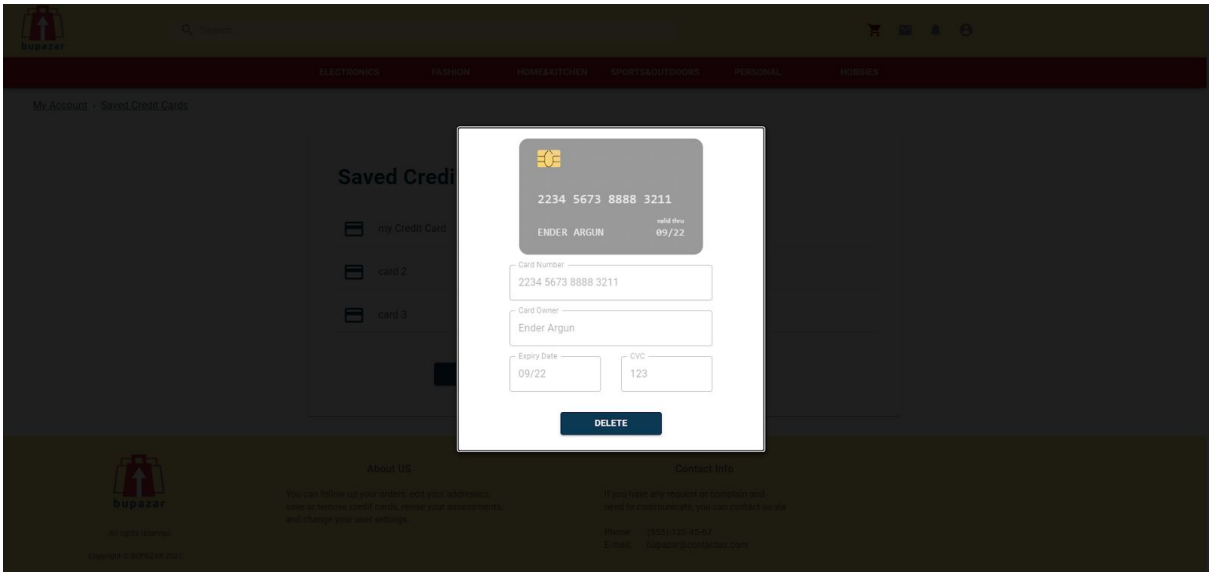




2.5. Saved Credit Cards

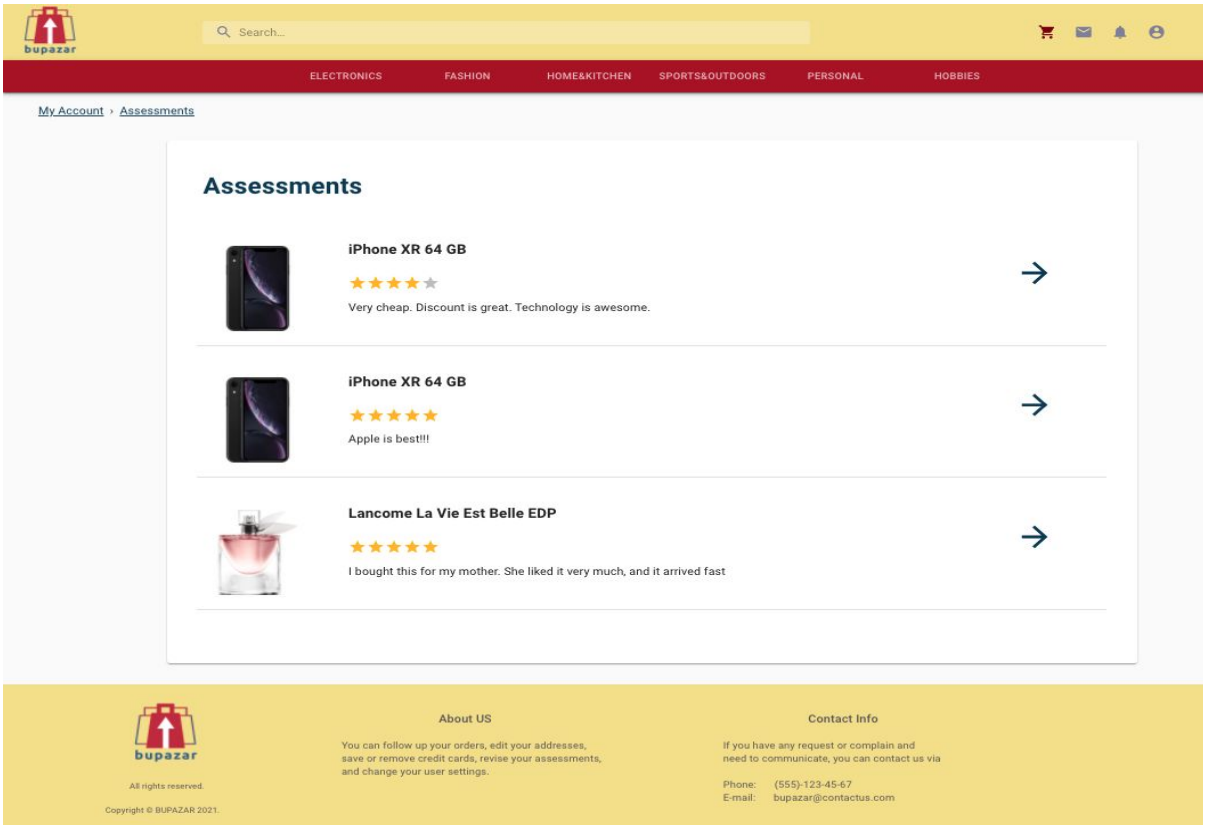
Customers can access this page at the side menu in the profile page. Here, customers can view all their saved credit cards, save a new credit card or delete a credit card.





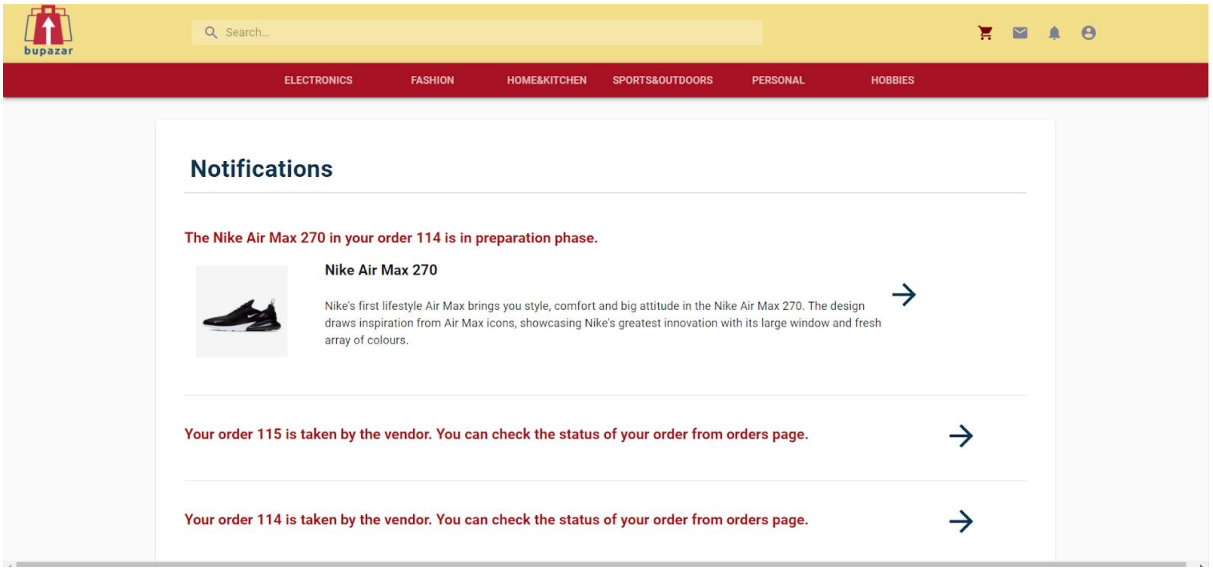
2.6. Assessments

This page displays all of the previous product reviews of the customer. Customers can access this page from the side menu at their profile pages. Product name and image, rating score and comment is displayed. Customers can also follow the arrow to go to the corresponding product's page. Our platform does not allow customers to edit or delete their reviews, but they can post multiple reviews if they change their opinion.



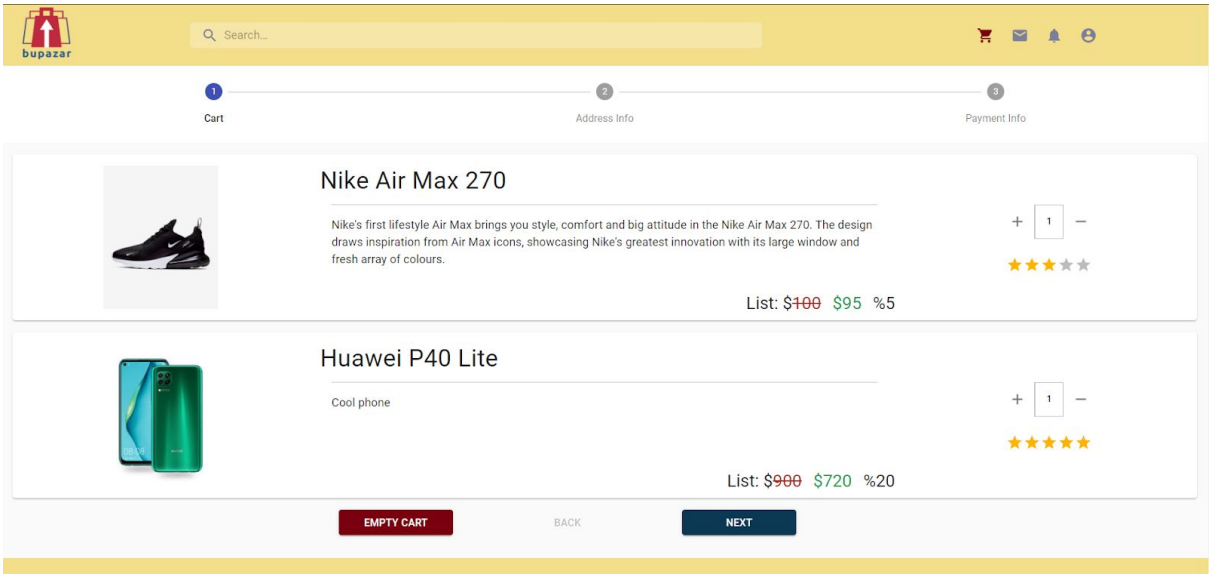
2.7. Notifications

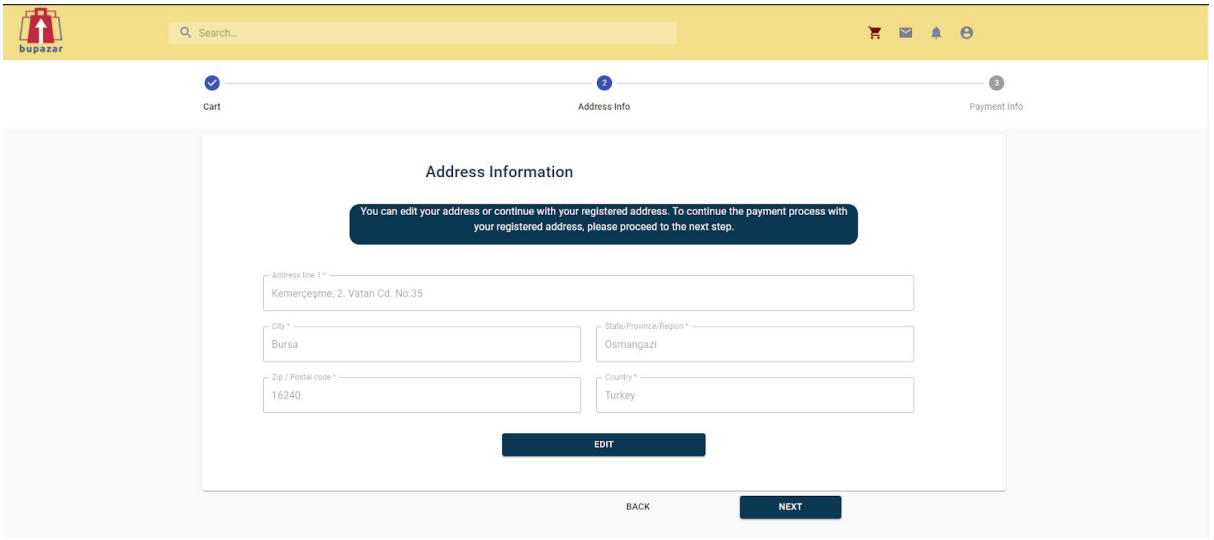
When new notification comes up, notification icon on the navigation bar is updated and the logged in user can view the number of new notifications. If the notification icon is clicked, the notification page opens and all notifications are displayed. Customers are notified in cases like when the price of one of the products on their list decreases,when discount rate is increased, when order is taken, when order is cancelled, when a product is out of stock or renewed. Also, new notification comes if the price of an alarmed product falls below the alarm price.

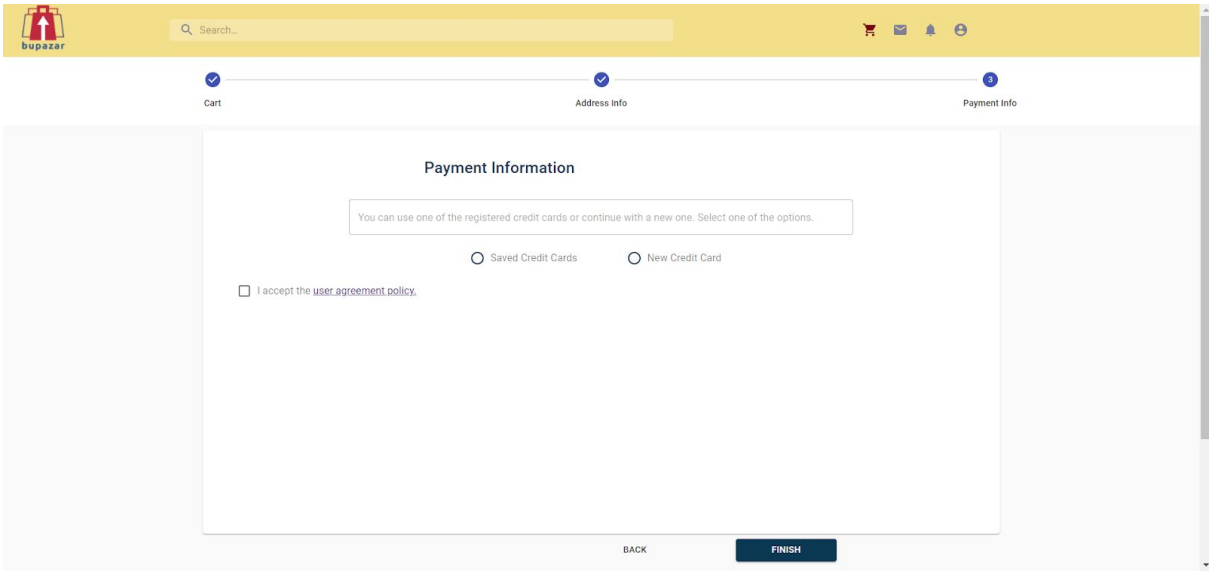


2.8. Payment

Users can proceed to the payment stage by clicking the cart icon from the navigation bar. Payment part has 3 steps. In the first step, the products in the cart are displayed. By clicking the “Empty Cart” button, users are able to delete all products in their cart. Users can decrease or increase the amount of products in their cart and see the ratings of the products in their cart. Also, discounted price, non-discounted price and the discount rate of the products are displayed. In the second step, the delivery address is filled. If there is a saved address, it is displayed by default. If the user wants to use a different address, they can edit their address. At the last step, credit card information is requested. Customers can choose one of their saved credit cards or continue with a new card. Finally, users who accept the user agreement policy can complete the payment process.



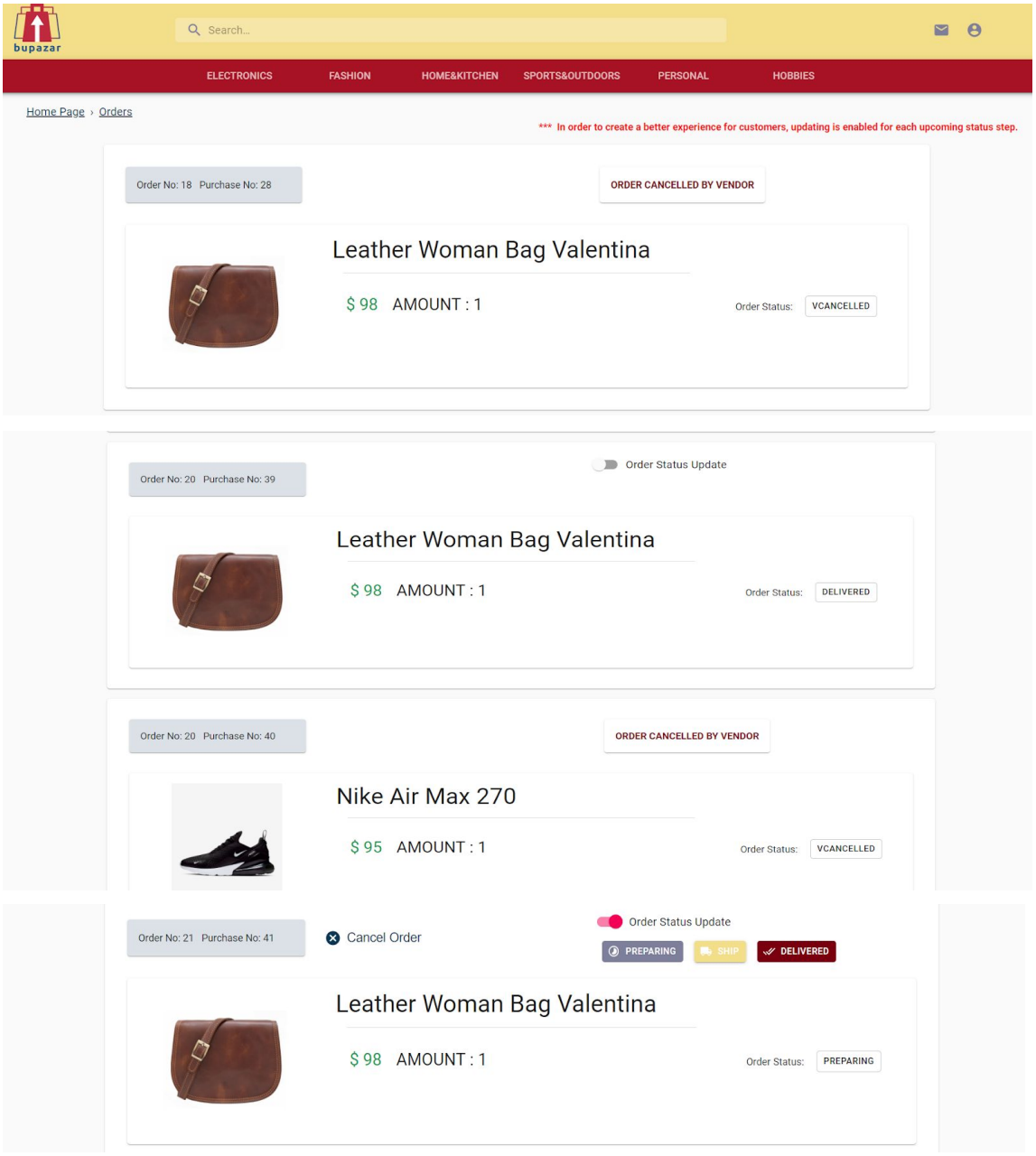



bupazar
Search...
Cart
Address Info
3
Payment Info
Payment Information
You can use one of the registered credit cards or continue with a new one. Select one of the options.
Saved Credit Cards
New Credit Card
I accept the user agreement policy.
BACK
FINISH

3. VENDOR SPECIFIC PAGES

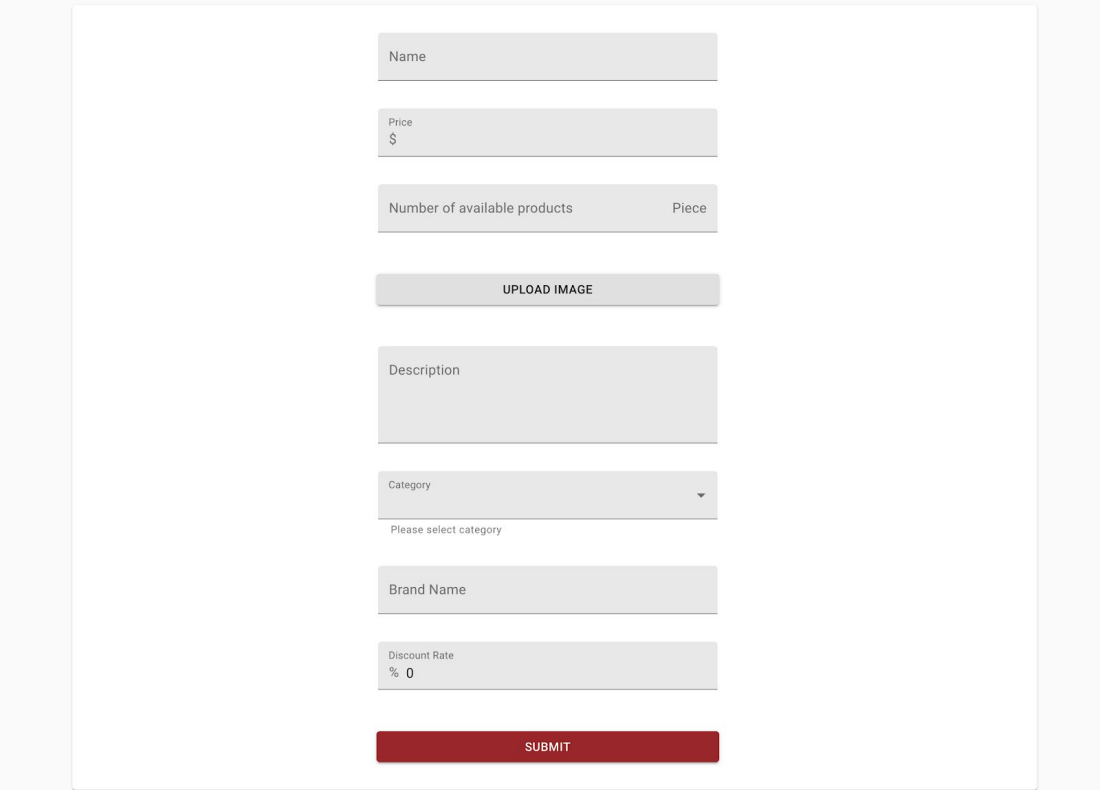
3.1. Orders

On this page, vendors can see which orders came to them. The orders go from oldest to newest. There are two main actions that a vendor can do on this page. First is to cancel the order and second is to update the status of the order. Order cancel button is only visible when the status of the order is whether ‘OrderTaken’ or ‘Preparing’. If the vendor wants to cancel the order, the cancel order button needs to be clicked. After clicking Cancel order, order status change option will be lifted off. For the second action, there is an order status update switch. Vendors can update order status one by one by clicking the emerging buttons. Only when the Ship is clicked there is an extra step. Vendor needs to write a courier name to the prompted screen in order to proceed. In every cancel and update action window is refreshed so the vendor can see the changes.



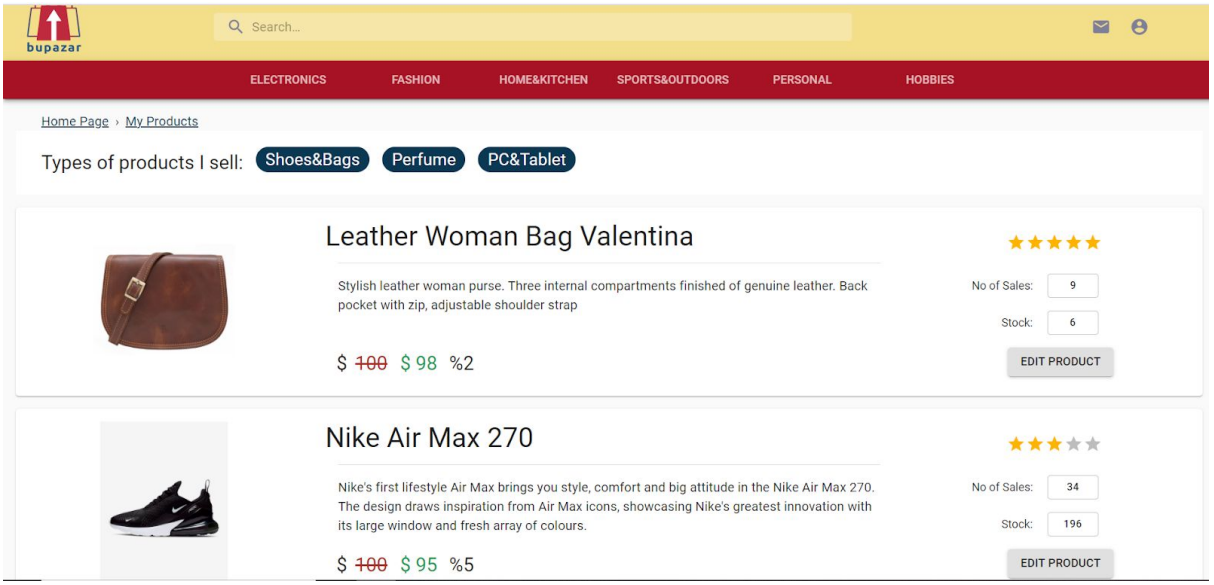
3.2. Add Product

For vendors who want to add products, they may click the Add product button in the user profile. They will be directed to Add Product page where they can fill necessary fields in order to add a product.



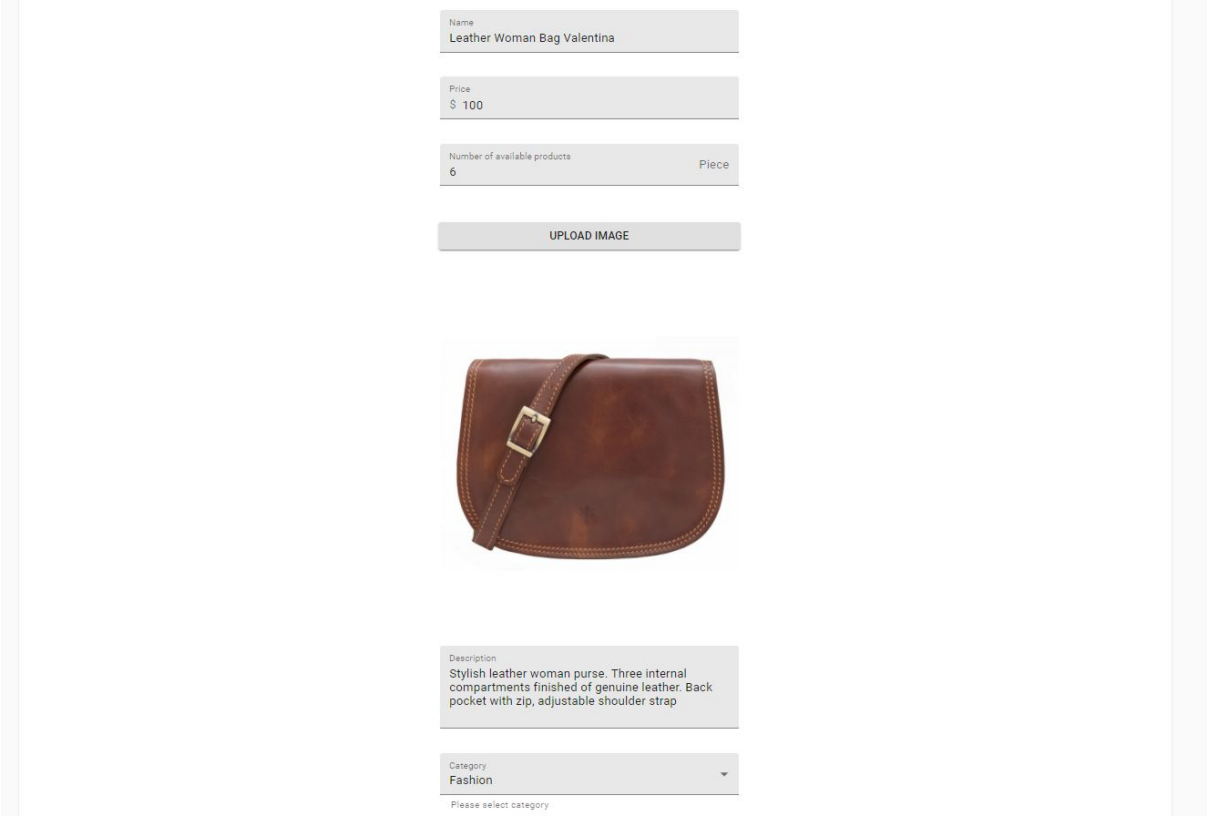
3.3. My Products

This page is for vendors and it is accessible by clicking My Products in vendor. It is to make the vendor see which products they sell. The list items consist of the information of product image, name, number of sales, stock, price etc. List item pictures are clickable as well. In addition, by clicking the edit button the vendor can go to the vendor edit product page.



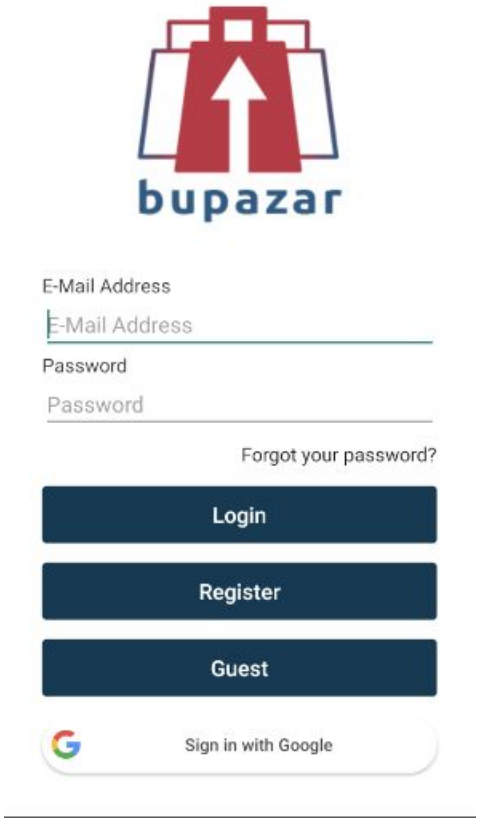
3.4. Edit Product

After clicking the edit button from the vendor products page vendor is directed to the editing page. He or she can change any information about the product. If the information is not changed properly it uses helper texts for giving directives to vendors in order to get the expected format. Otherwise, when the edit is successful it automatically directs the vendor to the vendor product page. So, the vendor would be able to see the changes.



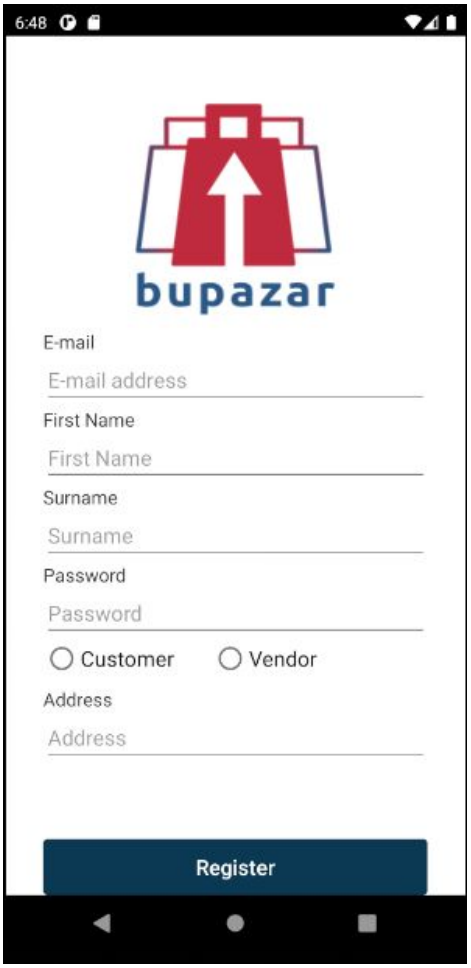
ANDROID USER MANUAL

Login



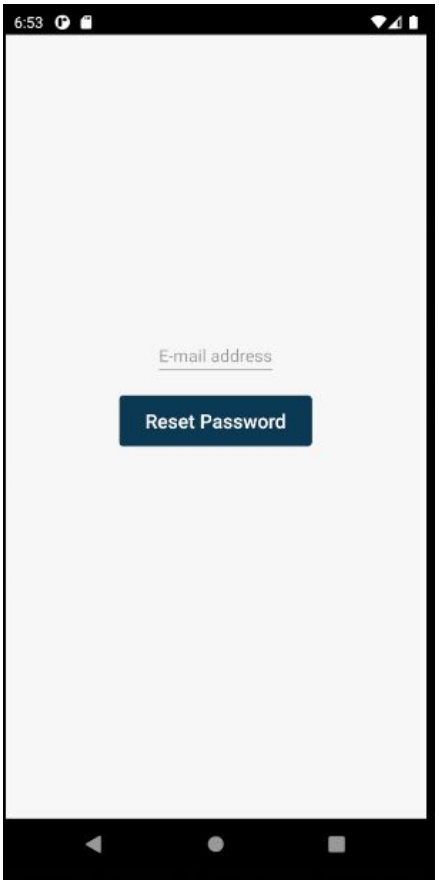
Login page is where you can login with your email and password, if you have an account for the application. If you do not have an account, you can go to the register page and register. Another option is if you want to browse the application without creating an account, you can log in as a guest user from the guest button and explore the application, but remember, guest users can do limited operations. Besides, you can also tap the "forgot my password" text, if you forgot your password. At last, if you have an account with a google email, you can also use the google login option to login the application.

Register



Register page is the page where you can register by entering the necessary information when you want to have an account in the application. Filling the username, email, firstname, last name, customer, vendor selection and address fields are mandatory fields that must be filled in to register.

Forgot My Password

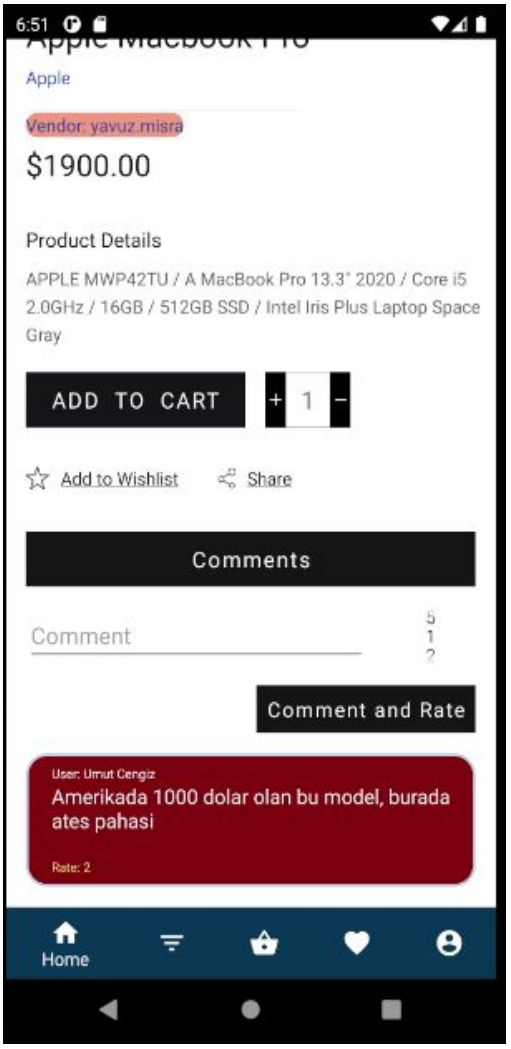


In this page, user can reset his/her password by entering his/her password. Then, user can reset the password on link sent to the email address.

Home

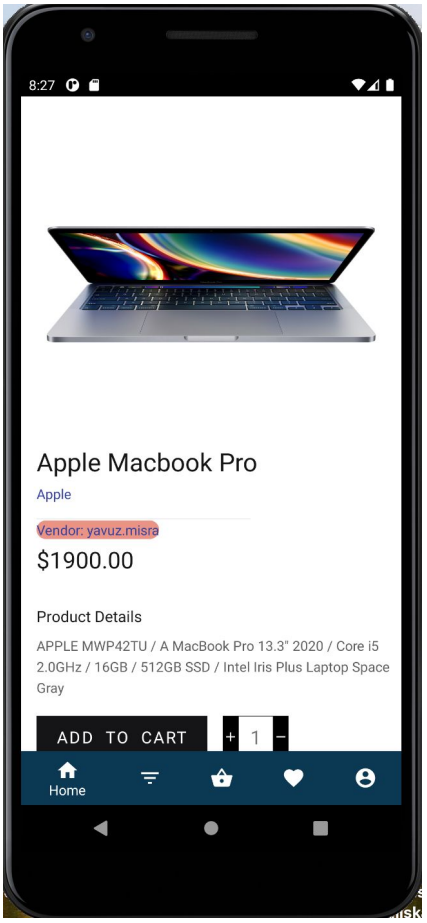
This page is the home page of the application. On this page, you can scroll down to view the products in the application and on sale with their prices. At the same time, if there is a product you are specifically looking for, you can see the product or products by typing the product name or type in the search button. From this page, you can also go to the best seller page, where you can see the best selling products, the recently trending products page, and a page where the products recommended for you are listed according to your previous purchases or the products you have added to your favorites. You can go to categories, chart, favorites and personal account page from the navigation buttons at the bottom of the page.

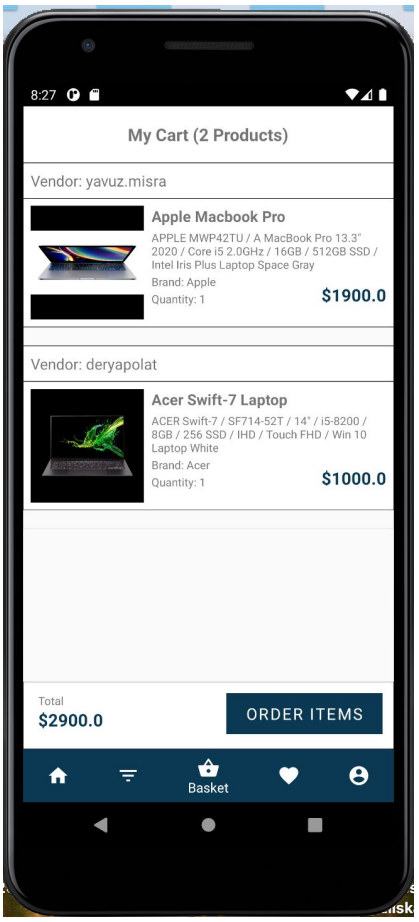
Product



This page is which you can see the details of an product. Big and clear picture, name, brand, details,price and vendor of the product is displayed on this page. Customer can inspect the mentioned information of the product. Customer can add the product to the cart with the amount , when he wants to purchase it. Besides, if customer like the products but not sure about the buying it, adding to wishlist to save products is an another choice. Customers also share the product link with their friends who can like and tend to buy it. Another information customer see on this page is the comments and rates of the product given by customer that purchased the product before. Therefore, customer can also comment and rate the product only if he/she bought the product before.

Adding Product to Cart

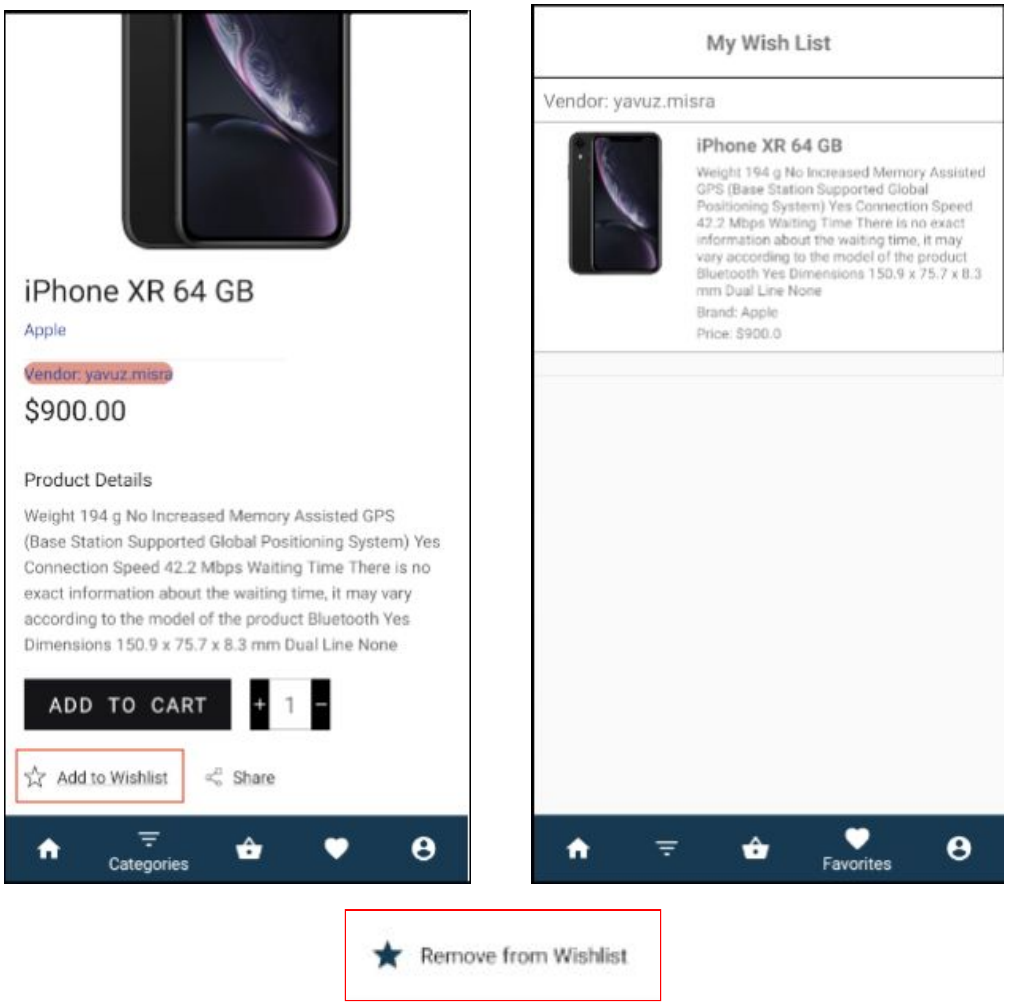




To add a product to the cart, you must click to the “ADD TO CART” button that lies at the bottom of the product details page. Quantity should also be chosen from the quantity increase/decrease section that lies at the right of the button. Please note that the default quantity is set to 1.

After pressing the add to cart button, your item would be added to the cart. You review and order the product in your cart from going to the cart page from the navigation bar. In this page, products you have added to your cart will be listed with their quantities, prices, vendors, and the total cost will be shown at the bottom of the page. To order the items, you must click the “ORDER ITEMS” button at the very bottom of the page, next to the total price of the cart.

Adding Product to Wishlist



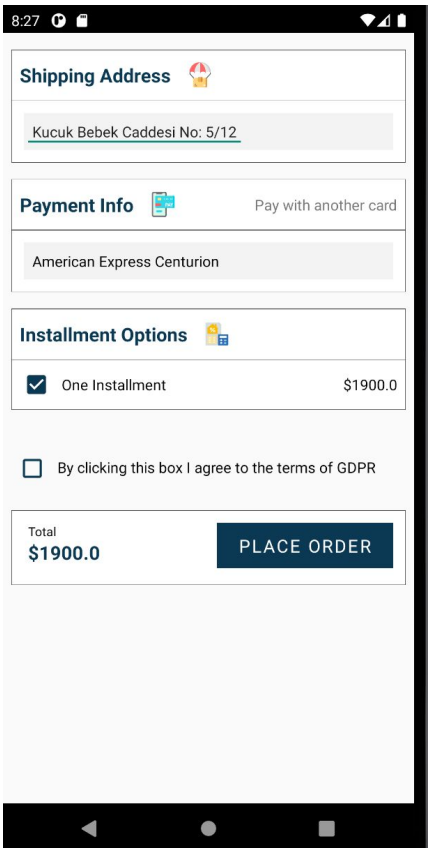
To add a product to the wishlist, you must click to the “Add to Wishlist” button that lies at the bottom of the “ADD TO CART” button.

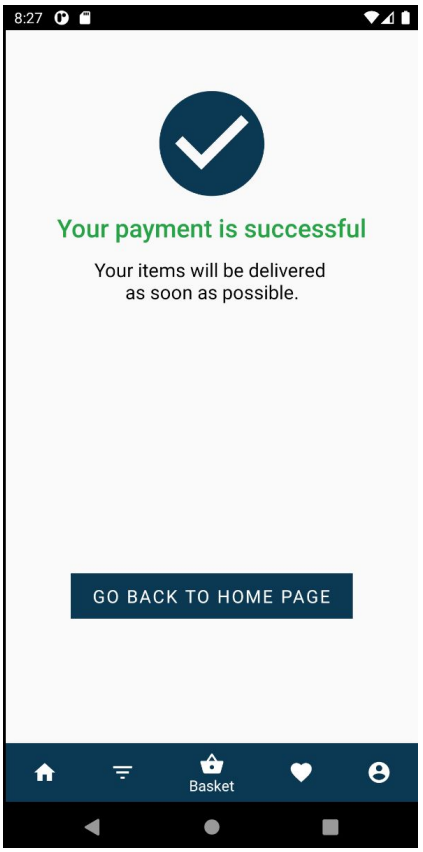
After pressing the “Add to Wishlist” button, your item would be added to the wishlist and the “Add to Wishlist” button would be changed to “Remove from Wishlist” button.

To remove the product from your wishlist, you can simply press the “Remove from Wishlist” button in the product page.

You can review the products in your wish list from going to the favorites page from the navigation bar. In this page, products you have added to your wish list will be listed with their details

Ordering Items Added to Cart





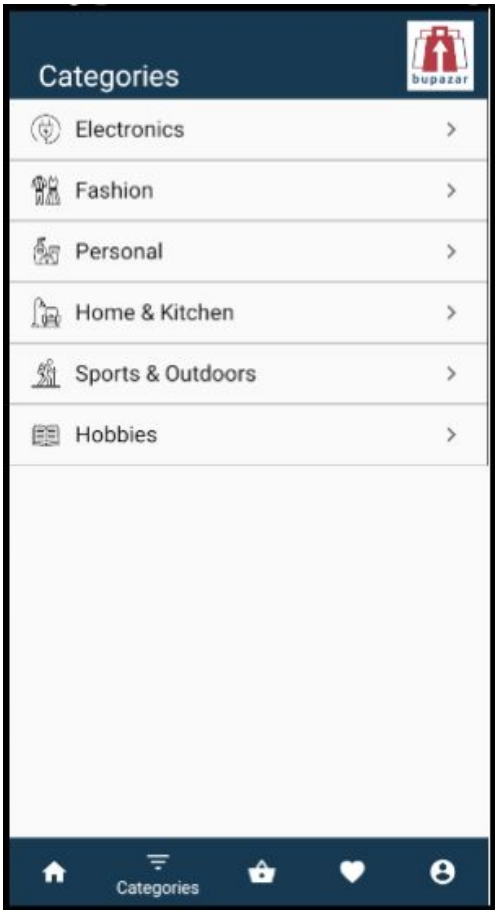
After pressing the order items button back in the cart page, you will be redirected to the order page that is shown on the left above. In this page, you must enter your delivery address, choose one of your saved credit cards, or add a new one, choose one of the available installment options that is based on your credit card, and lastly you must agree to the terms & conditions of the GDPR. After completing these steps, you can safely press the “PLACE ORDER” button to order the items that you have previously added to your cart.

Please note that without accepting the terms & conditions, you will not be able to place your order.

After successfully placing your order, the page on right above will be shown to you to further guarantee that your order has been received by the vendor.

Categories & Subcategories

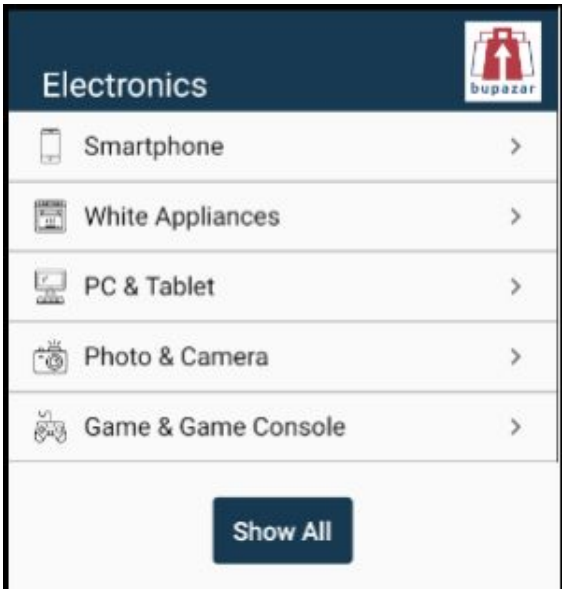
Categories

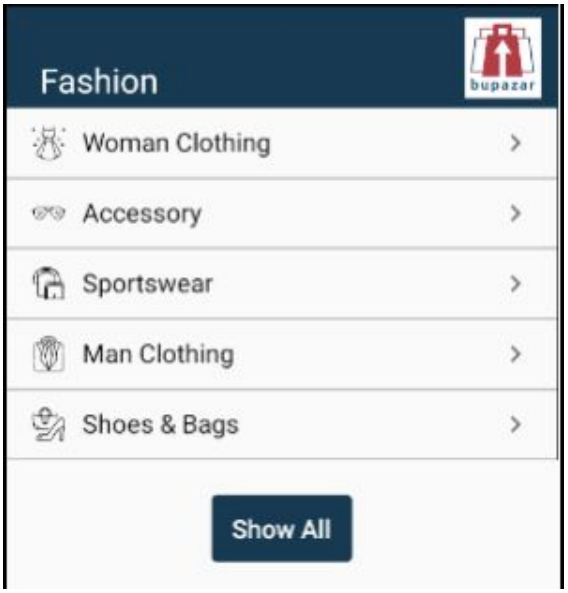


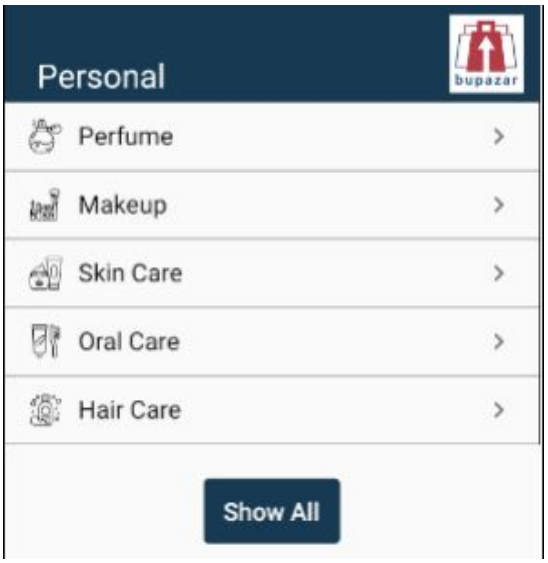
This page is the categories page, you can go there by simply pressing to the categories page from the navigation bar. There are 6 main categories including electronics, fashion, personal, home & kitchen, sports & outdoors and hobbies.

Subcategories

The pages shown in the next page are the subcategories mock ups. You can go this pages after pressing the corresponding category from the categories page.

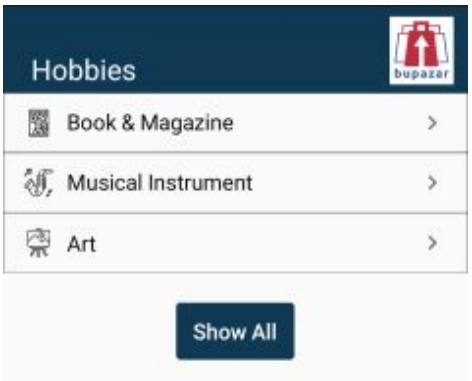

Electronics
Smartphone
White Appliances
PC & Tablet
Photo & Camera
Game & Game Console
Show All


Fashion
Woman Clothing
Accessory
Sportswear
Man Clothing
Shoes & Bags
Show All

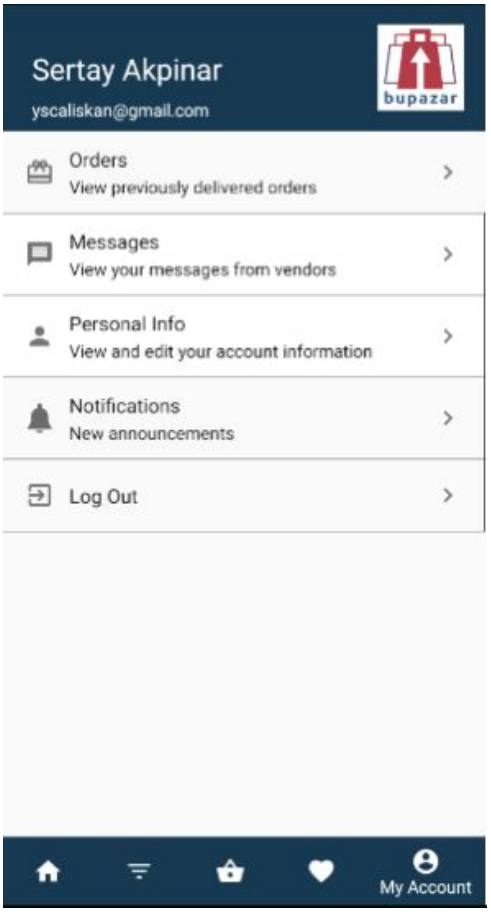

Personal
Perfume
Makeup
Skin Care
Oral Care
Hair Care
Show All


Home & Kitchen
Kitchenware
Beds
Decoration
Office Furniture
Show All


Sports & Outdoors
Sport Clothing
Fitness
Show All

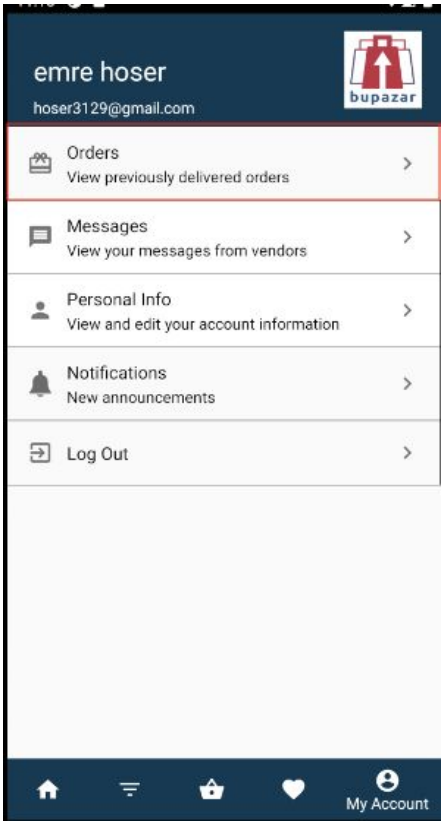

Hobbies
Book & Magazine
Musical Instrument
Art
Show All

Personal Account



This page is the page that directs the user to the pages where the personal data of their account can be viewed and changed. From this page, the user can see the order page where he can see his current and previous orders, the messages page where he can see his messaging with vendors, the addresses page where he can see the addresses registered in the system, the personall info page where he can see and change his personal information, the notification page where he will follow the developments about the products and orders he is interested in, and finally the in-app questions. It can go to the help page where it can find the answer. Also, it can log out from the current account with the log-out button.

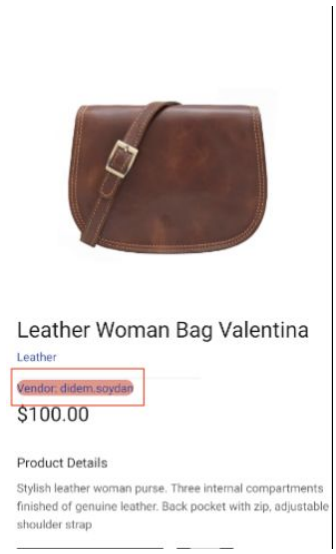
Previous Orders



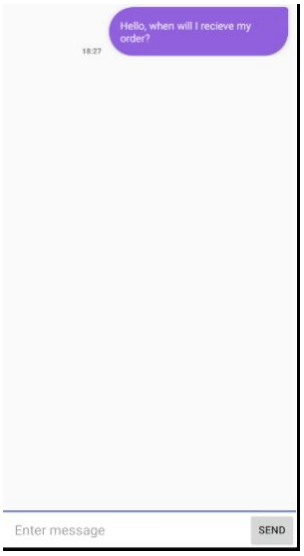
Customer profiles can view their previous orders by first going to MyAccount tab from the bottom navigation bar and then choosing from the MyAccount page. You can here view the order id, the price you paid, your order status, and the photos of items that you included in your offer.

Messages

a) Customer Side

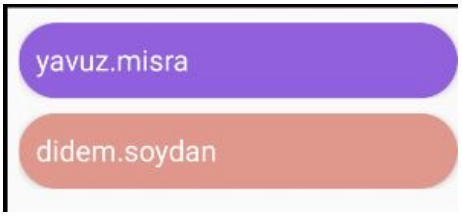


Product Page

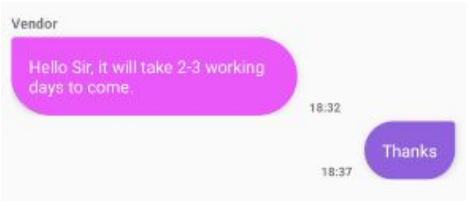


Chat Page

The customer can ask your questions to the vendor by pressing to the vendor name on the product page. After that program redirects you to the chat page. The customer can type your question and send it to the vendor by pressing the send button located on the right below in the chat page.



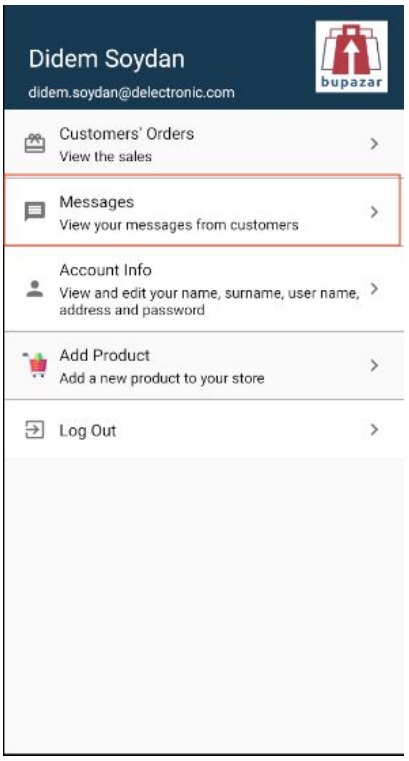
Choose vendor to ask

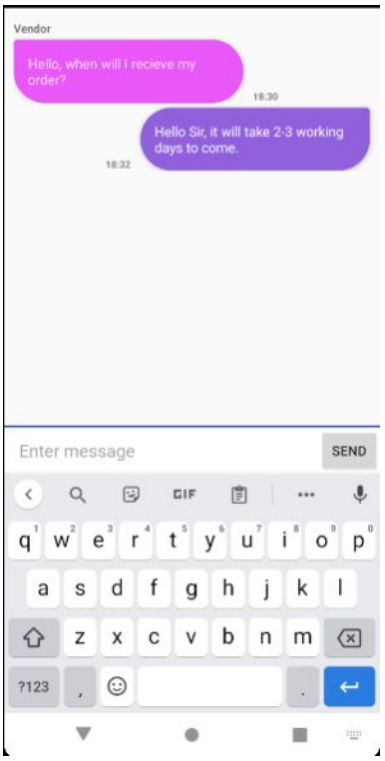


Type your message

The customer can choose the vendor to continue the chat from the messages page and reply to the incoming message in the chat page.

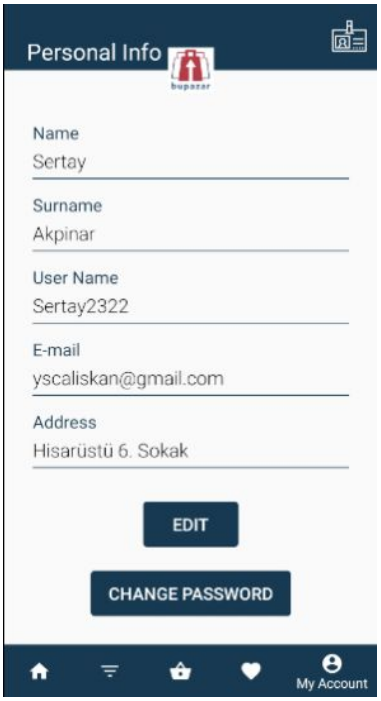
b) Vendor Side





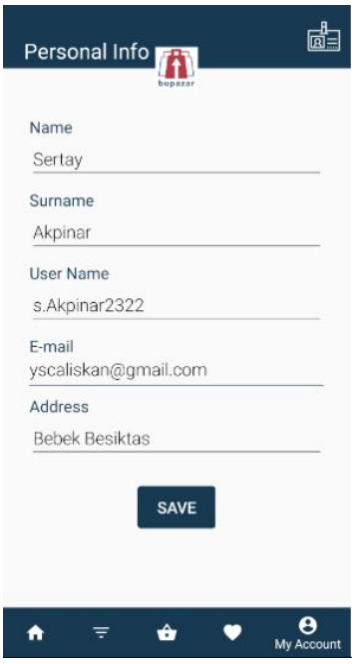
The vendor can review the messages by pressing the messages button from the vendor home page. After that the vendor can see the new message and also the vendor can reply to the message at the chat page.

Personal Info

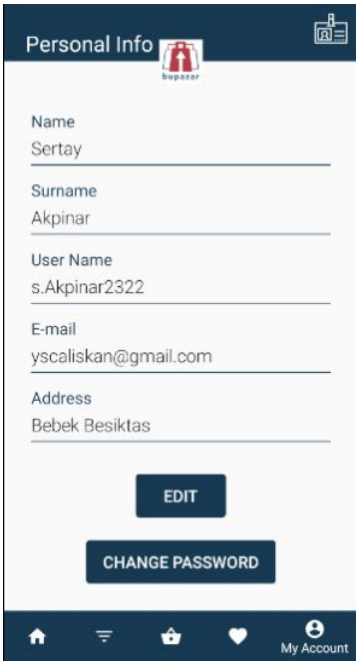


The user can review or edit his/her personal info and change your password from the personal info page.

Edit Personal Info



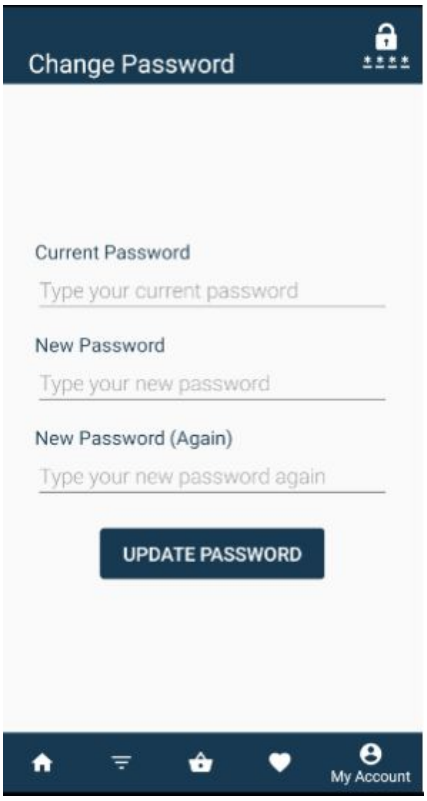
Edit your Info

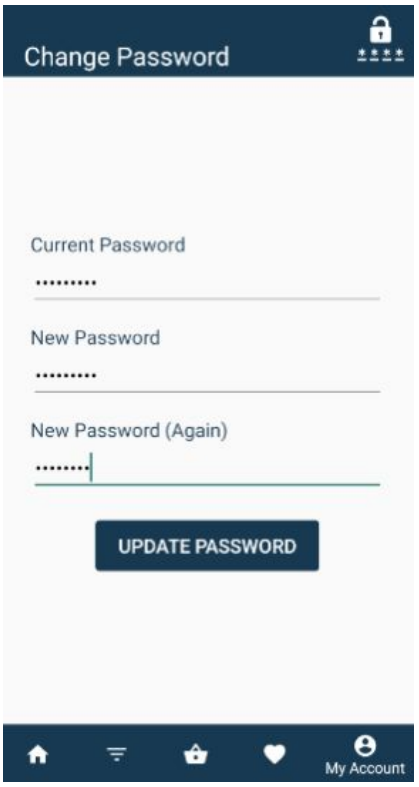


After saving the info

The user can edit your personal info by pressing the edit button in the personal info page. After the editing is completed the user saves the edited info by pressing the save button.

Change Password





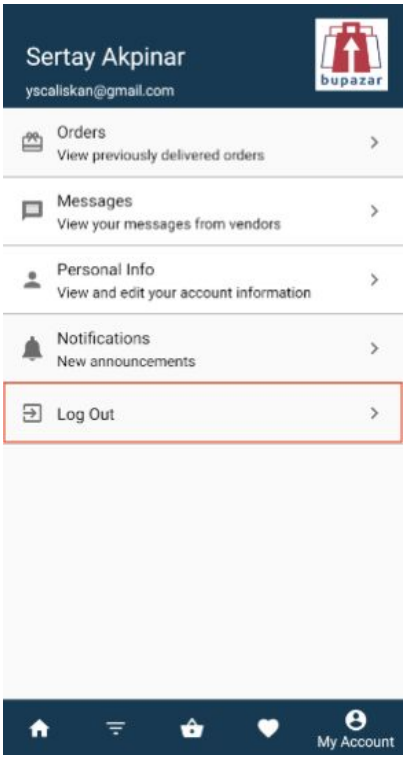
The user can change your password by pressing the change password button in the personal info page. After that the program redirects you to the change password page. The user can type the current and new password(twice). After that the user can update his/her password by pressing the update password. If the new password is a valid password the program will redirect the user to the personal info page.

In case the user typed the current password wrong or the new passwords does not match or the new password is not a valid password in terms of security, the app will not accept user to change his/her password.

Notification

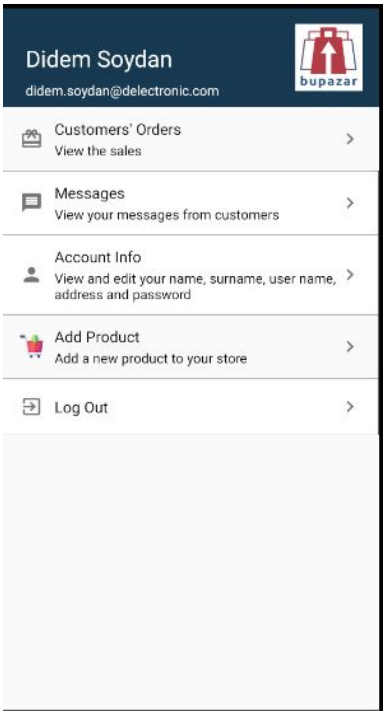
This page is where the user can view notifications about their orders, cart and products added to their favorites.

Logout



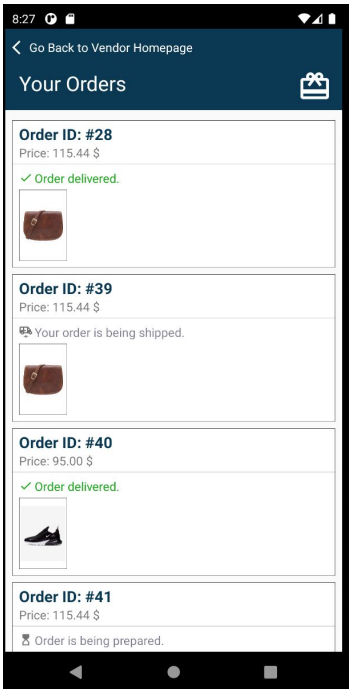
The user logout by simply pressing the logout button in the my account page. The program will redirect you to the login page.

Vendor Home Page



This is the vendor homepage.

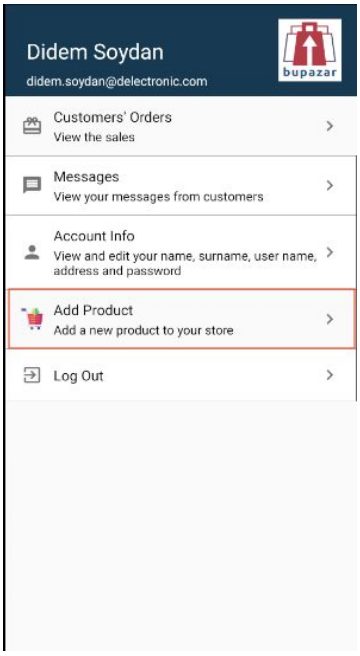
Customer Orders



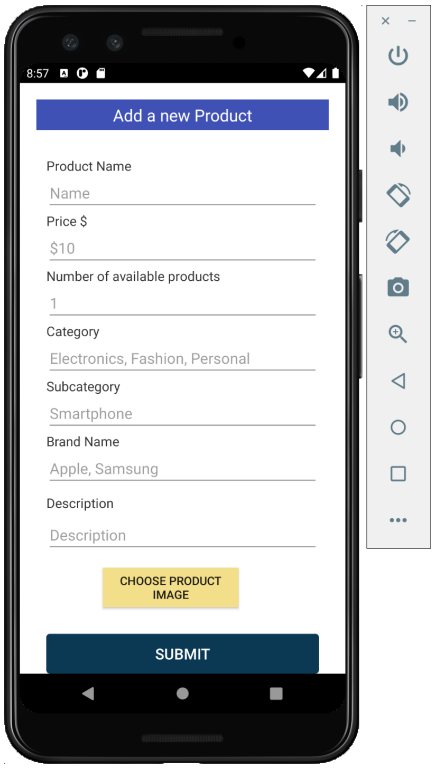
As a vendor, you can view the orders given to you by going to your Orders page from the vendor homepage. In this page, you can view the orders listed with their order ids, prices,

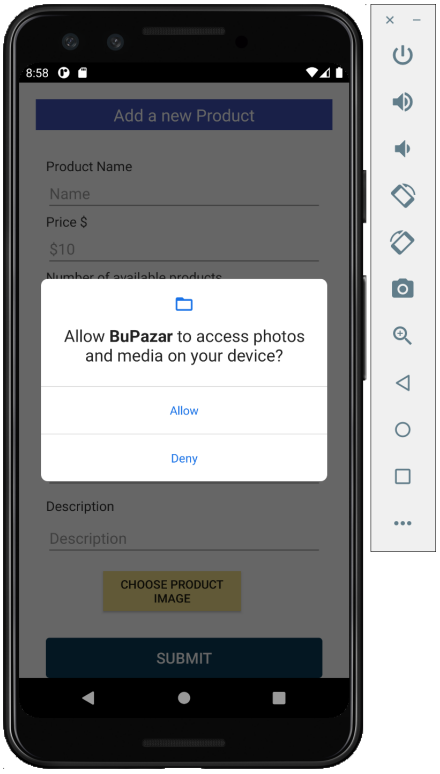
order status, and the product photo. To see more details and modify the order status, you can click on any item in this list.

Add Product



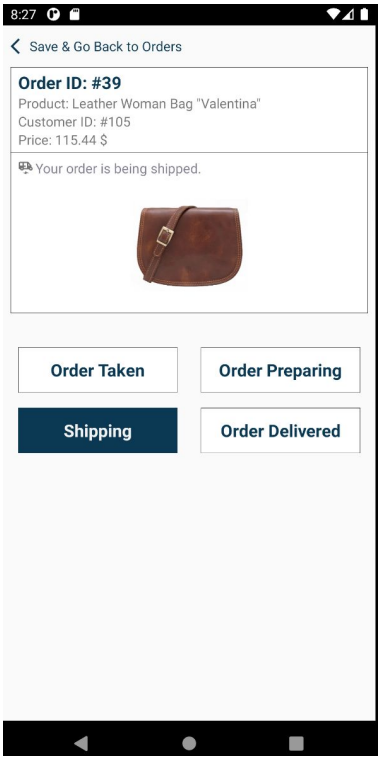
The vendor user can add product by simply pressing “add product” from the vendor home page.

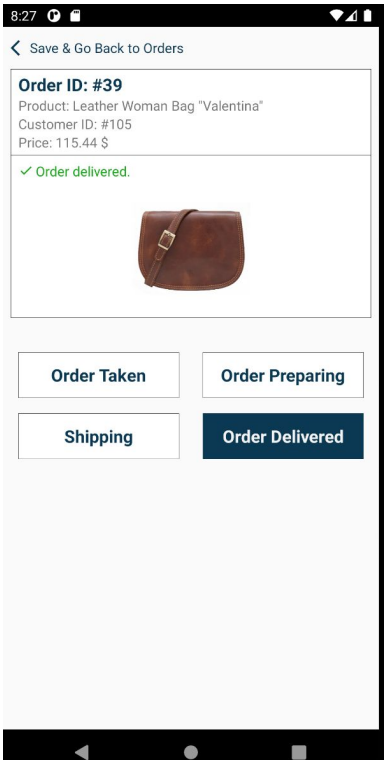




As a vendor, you are able to add products to your store. You can add product name, price, category etc. Also you can choose an image from your device then you can use it as an image for your product.
Additionally, users need to grant access to apps to allow them to access the media.

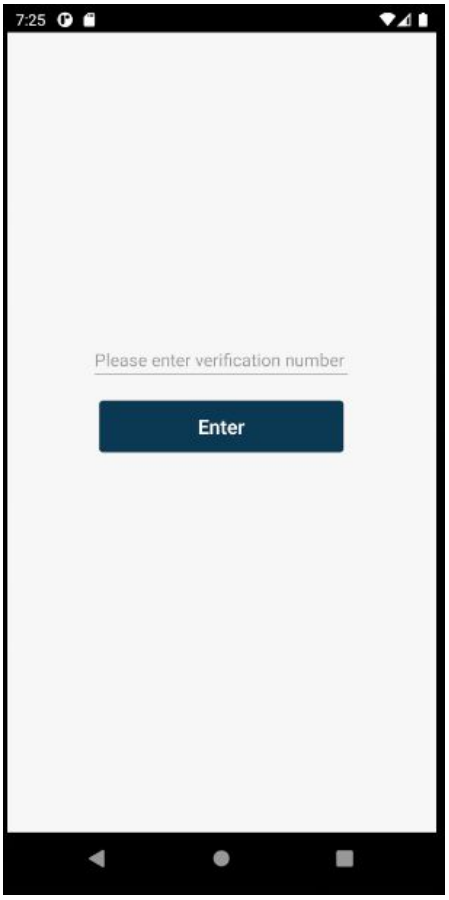
Vendor Order Detail & Modify Status





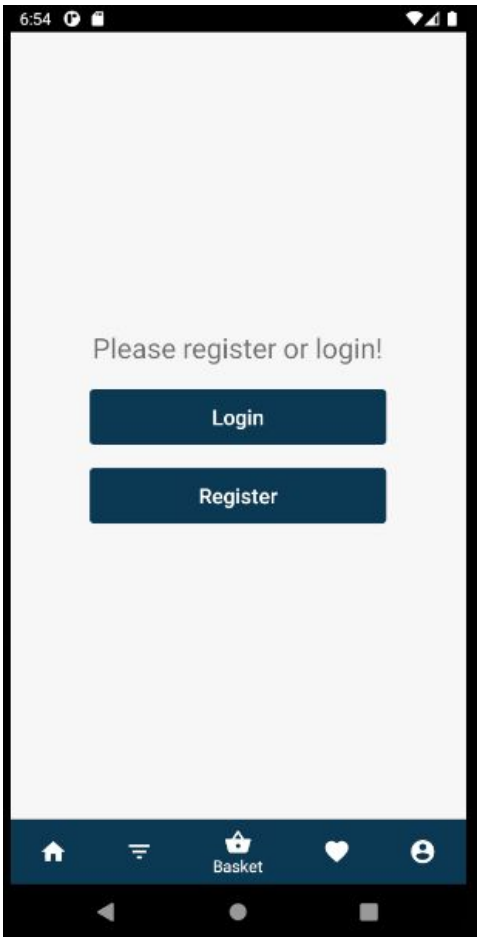
By clicking on any item from the Vendor Orders, you can view the order details page as shown above. Here, you will see the details of the order, and you can modify the status of the order. As shown in the above example, when the order details page was first opened, status of the order was in shipping, with the shipping button being highlighted. User clicks on the order delivered button on the bottom right, and the status of the order changes to “Order Delivered”. After you make the change, a notification will be sent to the customer, and the order status on his/her order will be updated as well.

Verification page



This page is opened after register page to verificate the account. Verification code is sent to user email addess. User has to insert that code to the remarked area.

Guest User



This page is shown when a guest user try to open cart , favorites and my accont page. Guest users have to login or register for using these pages.