Milestone 2 Report
Group 5

Members
Kayacan Vesek
Emre Hoşer
Volkan Bulca
M.Zeynep Çayırçimen
Mısra Yavuz
Sertay Akpınar
İsmet Sarı
Yaşar Selçuk Çalışkan
Muhammed Halas
Emre Demir

Contents

1 Executive Summary

1.1 Introduction

Our project is an e-commerce platform for everybody who wants to buy or sell products online. Main goal of this project is to implement such an online e-commerce application both for web and Android platforms. The name of our e-commerce platform is bupazar.
Bupazar's functionalities changes for various user types. A non-registered user is a guest user. Guest users can only view, search and filter the products, view the ratings and comments given to the products and add the products that they want to buy to the shopping cart provided by the system; however, they cannot buy any of those products, make comments and give ratings, and they also cannot send messages to people who sell the products i.e. vendors. A registered user can either be a customer or a vendor, but a user cannot be both types of users in a single account. In order for a user to buy and sell products via our platform must sign in from different accounts. Registered users must sign up to the bupazar.
Registered users can use all the functionalities that the platform offers to guest users. In addition to those, customers can give credit card information and shipment address to the platform, buy products and make comments and ratings to the products which they buy. They can create custom private lists of products for different purposes and see their purchase history, and create notifications for themselves to remind them some events such as when the a product is restocked or there is a discount for some products. Customers can also see some recommended products by the system itself. Vendors, on the other hand, must provide their address of operation as well as the stock status of the products they sell, take orders from the customers and send those via a cargo firm. Both customers and vendors can trace the shipment of the products. Both vendors and customers send messages via the chat functionality of the platform in order to get or give additional information to each other. Both customers and vendors can receive notifications when they receive a message or when there is a change in the shipment status of the products.
Last but not least, while implementation of the bupazar platform, W3C standards for web must be followed and the ethical considerations must be taken into account. Privacy of all users are important. Bupazar platform requires for customers and vendors to share some personal as well as payment, credit card and bank account information with the system so that the purchases and the shipments are done. Therefore, during sign up phase, the system asks both customers and vendors to accept the terms of use document that follows all the rules that GDPR and KVKK oblige. All registered users are expected to read the that document before signing up.
For more detailed information and functionalities, please refer to Project Requirements section in our GitHub wiki page.
For this particular assignment, we are supposed to find some API ideas and implement them. Our task is first research for some API that we find interesting. Finding an API that is useful for our project would be better for the development of our project. We should work from distance so using GitHub for version management is important. Each team member should implement an endpoint in different branches than master and create pull requests for merging branches into master. Also, each team member should test their API using some tools. We should also provide a simple front-end for end user interaction and deploy our application to the Amazon server that we create.

1.2 Work Done So Far

First, as the group members who are chosen randomly, we got to know each other. Fortunately, most of our group members were friends so this was an advantage for us in case of communication and team chemistry. Then we oriented ourselves with our Github repository. We prepared our communication plan, decided on the issue labels, created Slack workspace and determined the weekly meeting day. We decided to use Whatsapp and Slack for communication. Each one of us researched some interesting repositories and documented them by providing their references and describing what we liked about them.

Second, we started to specify the project requirements such as functional and nonfunctional. We attended a customer meeting session and we updated the requirements according to the customer's feedback. We spent a few weeks working on the project requirements until we sure that it met with the customer's demand. We continued to update the requirements as we progress. We added a glossary part to make requirements easier to understand.

Third, we created user scenarios and mockups related to these scenarios. Thanks to these scenarios, we learned to think from the user's perspective and mockups gave us a better understanding of what our project is going to look like. After that, we determined the logo and the name of our project.

Fourth, we started to work on software design of the project. We prepared project diagrams in an object-oriented manner to shed light on how the project will be done. These diagrams are consisting of use case diagram, sequence diagrams and class diagram. Since software design is one of most important parts of the project, we spent a 2-3 weeks working on it.

Fifth, we prepared our project plan which also includes the future part of the project. We created the RAM (Responsibility Assignment Matrix) to see each person's responsibilities and contribution to the project. We revised what has been done each week and discussed if there is anything to be add or change. Each one us documented our contribution to the project. So far, we can proudly say that we overcame the difficulties encountered during the process and completed all of the duties.

Last, we discussed how to distribute the roles within the group for this milestone. Our communicator created the base for the back-end development. After that we started to implement our end points. We used Twitter and a currency API. Basically, our goal was to connect the social media and e-commerce. That's why we chose one of the most popular social media tools which is Twitter. In our application, you can easily get some general ideas about the e-commerce platforms, vendors and the customers according to their tweets and their popularity. Since they are good criterions of showing the user's popularity, we addressed to the number of followers of the user, number of tweets mentioned to it etc. Everybody contributed the back-end development part of the assignment. In the front-end part we used React for the implementation. After the front-end part, we created an Amazon AWS EC2 instance, and deployed our application to the server successfully. During this assignment, we used Git tools as much as possible. Plus, we cared about the communication within the group, so we progress in a planned way. Finally, we can clearly say that we completed the implementation assignment successfully.

1.3 Road Ahead

Until this time, we have done very good group work and worked very coordinately. We have gained experience on the importance of front-end development and the challenges of back-end development from milestone-1 to milestone-2. Using our previous experience, we realized the importance of planning and designing a business. Now the first thing we will do is to elaborate our planning based on the difficulties we face and the experiences we have gained. As front-end, we consider planning the mobile design required for mobile development. As a back-end, it will be to further extend our unit tests and determine what we need as API. Then we will act to divide and write these API endpoints.

1.4 Challenges We Met as a Group

One of the most important challenge is the communication. Since we are using online platforms to communicate, duration of the meetings is increasing while efficiency is decreasing. In addition, it took a little more time for us to implement the endpoints because most of the team members didn't even know what an API was. We have learned a lot about using Git besides writing an API with this project. Of course, there were also minor difficulties in issues such as creating pull request, resolving conflicts with the master branch. Lastly, we did our best to avoid unbalanced load distribution, but still some people were assigned more than the others to review endpoint part.

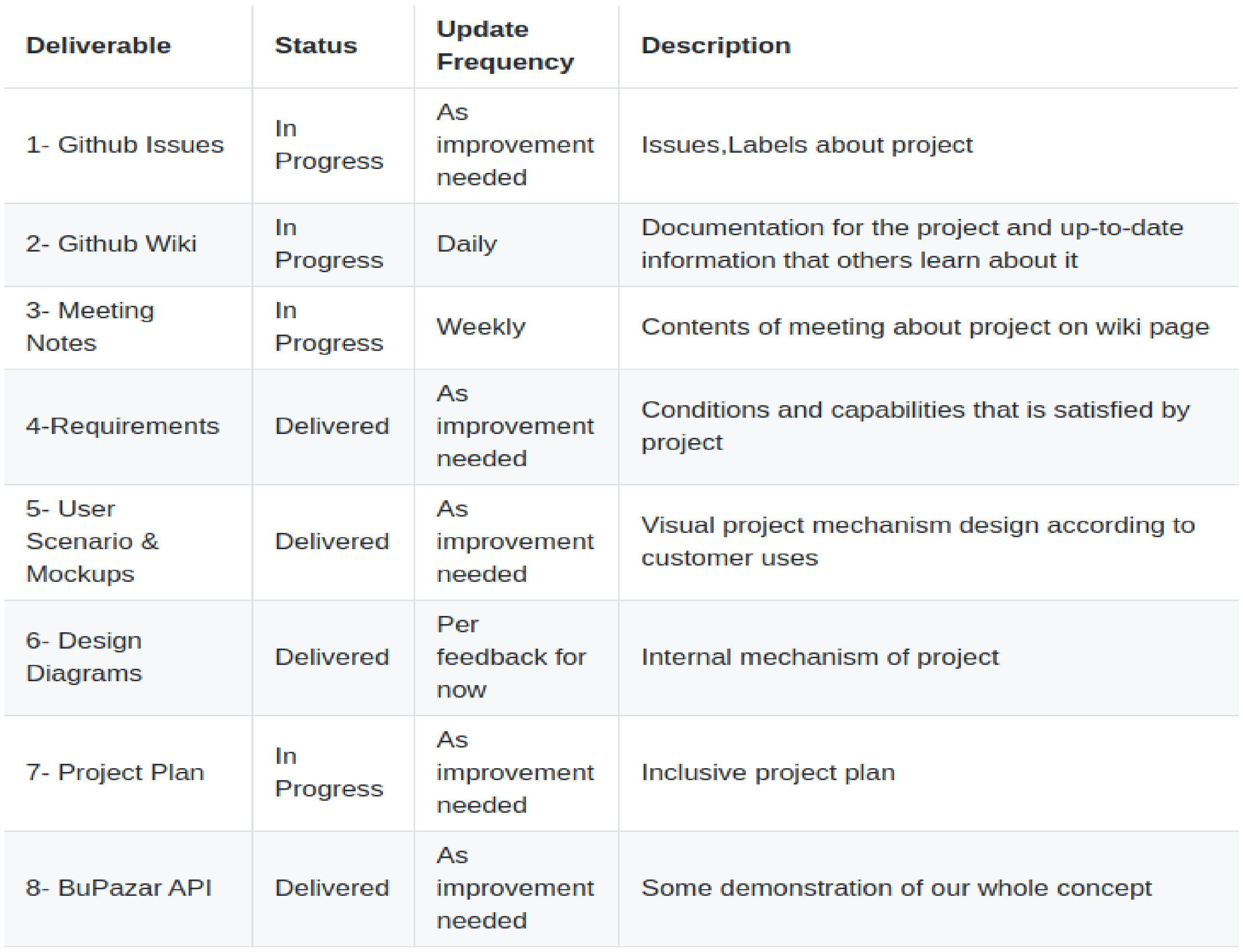
2 Work Done by Each Member

Member Name	Contributions
Kayacan Vesek	- I implemented base code of the back-end part. - Set the app.js and corrected routing - I created frontend branch and initiated front-end using React. - Although, I did not implemented API or Tests in the backhand. I contributed others codes to implemented their api's - I did 3 hotfix commits, also tried to fix code base structure. - I created pull request #66 and #95 to merge front-end branch to master branch. - I implemented front-end for endpoints with Sertay. - I reviewed pull requests #61, #62, #63, #64, #83, #96, #99
Emre Hoşer	- I implemented vendor most trending tweets endpoints. I implemented the API in vendortweets.js file. - I created a branch called vendorcomments worked in that branch. - After I am done with the implementation, I sent the pull request #93 to merge the implemented endpoint to master branch. - I reviewed the pull request #67. - I documented the functionality of my API to the API wikipage.
Volkan Bulca	- I implemented vendor mentions trend endpoint. I implemented the API in searchtrendsforvendor.js file. - I created a branch called vendorTrends worked in that branch. - I tested my code using Postman. - After I am done with the implementation, I sent the pull request #69 to merge the implemented endpoint to master branch. - I reviewed the pull request #68. - I documented the functionality of my API to the API wikipage. - I wrote the unit test for the searchtrendsforvendor endpoint with chai.
M.Zeynep Çayırçimen	- I implemented search trend for product endpoint. - I created a branch and send a pull request to merge the implemented endpoint to master. - I reviewed pull requests #61 and #65. - I tested my endpoint using Postman. - I wrote a unit test for searchTrendForProduct using chai. - I documented my endpoint to the API wikipage.
Mısra Yavuz	- I implemented filter_user_tweets endpoint. - I tested my code using Postman. - I reviewed 3 pull requests: #63, #67 and #93. - I created API wiki page and documented the functionality I introduced. - I wrote unit test for filter_user_tweets endpoint using chai.

Member Name	Contributions
Sertay Akpınar	- I implemented show followers endpoint. - I created a new branch called showfollowers. After I complete my work on showfollowers branch, I sent a pull request to merge the endpoint to master. - I reviewed 3 pull requests: #65, #69 and #93. - I implemented the front-end part of showfollowers, searchtrendsforvendor and vendortweets endpoints using React. - I documented the functionality of my API to the API wikipage. - I implemented the unit test for the showfollowers endpoint using chai.
İsmet Sarı	- I implemented find popular tweets about a product. The whole code stays in tweetscomments.js file in the API. - I created a branch called tweetscomments and send a pull request#61 to merge the completed endpoint to master branch after assigning my friends to review my code. - I review Zeynep's pull request and have propesed her to make some changes. - I tested my endpoint using postman. - I have write the concept of my endpoint to API wikipage.
Yaşar Selçuk Çalışkan	- I implemented exchange rates endpoint. - I created a branch called exchangeRates, did all my changes in there, and after I'm complete I sent it to reviewers. After solving certain conflicts and changing minor things, my code is merged. - I tested my endpoint using Postman. - I reviewed the pull request: #68 - I wrote a unit test for exchangerates endpoint using chai and chai-http modules. - I've documented briefly the functionality of the API I used in my endpoint.
Muhammed Halas	- I implemented the addproduct endpoint in the database branch. - I created an local mongodb instance. - I tested the endpoint using postman. - I wrote unit tests for the endpoint using chai. - Applied those tests with chai. - Added chai and mocha to dependencies. - Created test folder.
Emre Demir	- I implemented the thelist endpoint in the database branch. - I created an local mongodb instance. - I tested the endpoint using postman. - I reviewed pull request: #69 - I wrote unit tests for the endpoint using chai and mocha. - Applied those tests with chai. - Added chai and mocha to dependencies.

3 List and Status of Deliverables

3.1 Evaluation of the Status of Deliverables and its Impact on Your Project Plan



Deliverable	Status	Update Frequency	Description
1- Github Issues	In Progress	As improvement needed	Issues,Labels about project
2- Github Wiki	In Progress	Daily	Documentation for the project and up-to-date information that others learn about it
3- Meeting Notes	In Progress	Weekly	Contents of meeting about project on wiki page
4-Requirements	Delivered	As improvement needed	Conditions and capabilities that is satisfied by project
5- User Scenario & Mockups	Delivered	As improvement needed	Visual project mechanism design according to customer uses
6- Design Diagrams	Delivered	Per feedback for now	Internal mechanism of project
7- Project Plan	In Progress	As improvement needed	Inclusive project plan
8- BuPazar API	Delivered	As improvement needed	Some demonstration of our whole concept

- GitHub Issues : After the meeting, everyone create their own issue to inform the other group members about the process of their design. The perform who created a issue cannot close the issue without asking the communicator. You can reach our current issues in issues page on GitHub.
- GitHub Wiki : Every week after completing the our weekly tasks we upload them and update the our wiki page to see the overall project progress.
- Meeting Notes : Every week we are taking the meeting notes to specify the action items and agenda and you can see the whole meeting notes so far in our wiki page.
- Requirements : We completed all the requirements of our project and we will use them during the implementation of the project.

- User Scenario & Mock-ups : We design some mochups according to the user scenarios to show what our project is going to look like. They all are accessible in our wiki page.
- Design Diagrams : We have created some diagrams according to requirements but they might be updated as we move on implementing.
- Project Plan : All the things requirements, class diagrams, user diagrams etc. are the component of project plan and we keep them in a single header as project plan.
- BuPazar API : This is designed to help us when we are implementing the main code. Some of us implemented a basic front-end and back-end platforms to build this API. We completed it before the deadline. We all face some difficulties as a group such as communication. App can be enhanced , however this version mainly get us what we want.

4 Evaluation of Tools and Processes You Have Used to Manage Your Team Project

4.1 Tools

4.1.1 VsCode

We used Visual Studio Code for our development work. It is not an IDE, instead it is a code editor. Yet, we considered it is sufficient for our work. VsCode is very rich in extensions and also can be integrated with GitHub and Docker. It is simple to use and has a nice user-interface.

4.1.2 Twitter API

Twitter API provides access to public data on Twitter. It has a nice documentation on developer.twitter.com. Also, since it is a very popular API of choice, solving problems we encountered during the implementation phase was relatively simple with the help of online resources.

4.1.3 Postman

Postman is a very commonly used tool for API testing. It also has a nice documentation on learning.postman.com and there are lots of helpful tutorials on Postman's YouTube channel for beginners on API testing. We used Postman for testing our endpoints to ensure they are working as expected.

4.1.4 MongoDB

MongoDB is a NoSQL, document oriented and scalable database. Since it is recommended for e-commerce product catalogs, so we thought it would be appropriate for our platform and would make our jobs easier at back-end. Since there is not much development work done so far, we will probably gain more knowledge of it in later phases of our project.

4.1.5 Amazon Web Services

Amazon Web Services (AWS) is a secure cloud services platform with lots of functionalities making the platform very popular for developers. We used AWS to create an EC2 instance, and deployed your application to that server. Figuring out how to deploy on AWS was really challenging. Hence, it is not as beginner-friendly as the other tools we have used so far.

4.2 Processes

4.2.1 API Development Process

- Implementation: Each member should develop their endpoint in a new branch with the branch name indicating the functionality they are introducing. Members should also comment their code clearly and test their endpoint before they open a pull request to master branch.

- Pull request: After members are done with developing first versions of their endpoint, they should open a pull request with an informative title, description, labels and they should assign other members as reviewers. All members should be assigned in different pull requests so that each member takes a part in reviewing phase.

- Review & Testing: Assigned reviewers should examine and analyze the files changed, distinguish errors or unnecessary parts of the code. They can ask for changes as they see fit. After files changed are all reviewed and corrected, reviewers should test the endpoint either on their localhost or with other testing tools. If test results are successful, reviewers approve the request. Otherwise, they ask for changes. After all reviewers approve the request, and all conflicts are resolved with the master branch, pull request can be merged and the branch may be deleted. For any further possible modifications, branches can be restored and the procedure starts all over again.

5 API Document

This API contains 9 endpoints.

- **/searchtrendforproduct** endpoint takes 1 query parameter that is the product name. It returns the last 100 tweets related to this product.To be able to calculate the popularity, the time difference between 1st and 100th tweets is calculated. Based on the time difference, the average number of tweets posted within 24 hours is estimated and returned to the user. Request should be sent as http://52.87.219.95:3000/searchtrendforproduct?product="product name". Response should be a number that indicates the popularity of the product.

- **/vendortweets** endpoint takes 1 query parameter that is vendor name. It returns the last 100 tweets of the vendor account.Returned tweets are pushed in a dictionary and sorted. By sorting the array, API is able to return most favorite 10 tweets of the vendor. The API returns the 10 most favorite tweets of a vendor account. Request should be sent as http://52.87.219.95:3000/vendortweets?vendor_name="vendor name". Response should be an array.

- **/searchtrendforvendor** endpoint takes 1 query parameter called vendor. Its aim is to search for the tweets that contain the vendor name provided as mentions. This search is done in most recent 100 tweets. Then, looking at the first and the last tweets returned, it calculates the time difference between those and approximately finds the number of tweets posted in 24 hours. The request should be sent as *http://52.87.219.95:3000/searchtrendforvendor?vendor="vendor name"*. The response is just a number given in the JSON format. This number basically indicates the popularity of a vendor.

- **/twittercomments** endpoint takes exactly 1 parameter called product. After the question mark you need to write product="name of the product such as iphone8". After the request the API returns 100 tweets about the product with different popularity. The API returns the other elements of tweets in addition to text of the tweets such as favorite count, location, when the tweet is typed, etc. To pick up the most 10 popular tweets, all tweets are sorted by the favorite count variable in ascending order and returned the first ten tweets as JSON format.

- **/exchangerates** enpoint takes no parameters and returns the real-time exchange rates of 33 different currencies based on the Turkish lira. It uses a relatively simple API to fetch this data where the base currency and the date specified. However, we chose to stick with basing it on only Turkish lira. Request should be send as *http://52.87.219.95:3000/exchangerates* and the response should be a JSON object with three keys: rates, base and date. Rates is also another JSON objects that contains the code names for 33 currencies as keys and the exchange rates as values. This API uses European Central Bank data to provide the rates.

- **/filter_user_tweets** endpoint returns the mentioned tweets of a vendor by another user. A request to this endpoint takes 3 query parameters: user, mentioned and count. count is for the number of tweets wanted, user is the Twitter username of whom tweets will be searched and mentioned is the Twitter username of the vendor. A request example can be given like this: ./filter_user_tweets?user=ARealityEvent&mentioned=Apple&count=10. Response should be a JSON string of an array consisting of mentioned tweets of the vendor by user, ordered from most recent to least. If there is none, it is an empty list.

- **/showfollowers** endpoint returns the number of twitter followers of the user. A request to this endpoint takes 1 query parameters which is name. Name is the Twitter username of the user. A request example can be given like this:
  http://52.87.219.95:3000/showfollowers?name=mansuryavas06.
  Response should be a JSON string represents the number of twitter followers of the user.

- **/database/addproduct** endpoint is implemented in the way that it takes parameters "name", "price", "color", "rating", "size", "comments" as form-data/key-value-pairs and then inserts a new product to the database and returns the information of the new product.

- **/database/thelist** endpoint takes no parameters and is fully public. It returns the top-rated 20 products in the database starting from best rated to worst. The aim of this endpoint is to publish the best products that is rated by the users, and also to determine the vendors who sell the best rated products, giving a chance to reward those vendors.

The URL of our API is `http://52.87.219.95:3000/`,
Also, you can access our front-end website via `http://52.87.219.95:3001/`