CmpE352 – Spring 2020

Milestone 2

Group 4

1. Executive Summary

Introduction

Our project, Buyo, is an e-commerce platform that aims to connect online vendors to their potential customers. The platform invites 4 types of users: Guests, Customers, Vendors and Admins.

Guest users are free to search, filter or sort any product they desire according to the criteria such as product name, vendor name, vendor location, ratings and price. Guest users are free to sign up as a customer anytime they want by providing email, user name and a password. Customers have all the functionalities of guest users in addition to being able to make lists to keep track of items wish to buy, having a cart page which they can add items before checkout and an orders page which they are able to keep track of the items they have bought, cancel or return an item and comment and rate on the products they have bought.
Vendors can sign up as by providing at least one store location in addition to an email address, username and a password. Vendors are able to for example keep track of their orders, cancel them, make discounts on their products and read user comments on their products.
Admins are able to block users and communicate with Customers and Vendors via messages sent through the platform.

What's been done so far

Application Decisions

Since everyone in the world in is in quarantine these days, we decided to make our application related to Covid-19. We wanted to make it an application that provides its users with all necessary information they may need while staying in quarantine and some additional features to keep their spirits up.

Endpoint Decisions

Our endpoints provide things we thought might be useful amids the pandemic. We chose coronavirus related news, daily coronavirus data in Turkey, weather in 3 big cities, weather forecast for the next 7 days in Istanbul, Try/Usd rate, Nearest hospitals for given coordinates. Also, we added some endpoints suggesting playlists from Spotify and books. Lastly, we added an astronomy picture and quote of the day in our application. We used various APIs for these endpoints.

Implementation Decisions

We decided to use NodeJS for our backend operations, MongoDB for database operations and HTML, CSS, Javascript, JQuery for frontend operations. We created a database which includes dummy data (vendor locations) and used it for our nearest vendor endpoint.

Challenges we met

Since almost everyone in our group were new to using APIs and NodeJS, one of the most important challenge for us was learning both how to use an API and how to get data from one, and how to implement an endpoint using NodeJS. Also, many of us didn't know how to code unit test for our endpoints. We learned them by working on the tutorial Koray prepared for us and the other resources. We were also new in implementing frontend and we learned it by searcing from various resources from Google and learned how to implement frontend of our application.

Road Ahead

Next semester, we will develop our e-commerce website. We worked coherently, gave feedbacks to each other if necessary so far and we plan to keep it for the next semester. We noted some APIs we may use while implementing our website. Since many of our members are new to developing websites and apps, we think we can improve ourselves during this summer term so developing the website and apps will be easier for us.

2. List and Status of Deliverables

Deliverable Name	Delivery date	Status
Back-end	May 26, 2020	Delivered
Front-end	May 26, 2020	Delivered
Dockerize	May 26, 2020	Delivered
Deployment on AWS	May 26, 2020	Delivered
API documentation	May 26, 2020	Delivered

3. Evaluation of The Status of Deliverables

By the time of Milestone 2 the whole app is supposed to be working as expected; including backend, frontend and Dockerized app and the AWS EC2 instance. The process of implementation of each deliverable is described in the following chapters.

Back-end

We chose NodeJS to implement the back-end. We found different APIs and selected some endpoints from these APIs. Each group member chose one of these endpoints. Everyone created a new branch and implemented necessary method and unit test for this endpoint in their new branch. When finishing implementation, we created a pull request and add all group members as a reviewer. After group members' feedbacks, we fixed the errors. Again we committed our changes. In case everything was fine, we merged our branch to backend branch.

Front-end

After finishing back-end implementation, we made some meetings for planning and extracting tasks for implementation and deciding on implementation details. We share our knowledge on using Javascript for communicating with the backend. We chose JQuery as the method of communicating with the backend and prepared the general design of the website and also we prepared an example frontend and container for one endpoint. Then we shared endpoints and tried to implement frontend for them using our example. While we were implementing, we faced some problems and in order to handle them we communicated with each other via Slack and Google Meets. Same way with backend, we used pull request property to make clean and reliable code.

Dockerizing

After we finalized our implementation, we put our app into a container using Docker. First, we dockerized both the frontend and backend of our app. Then by using “docker-compose”, we integrated our Docker container with the container of MongoDB. Those two containers can be pulled from DockerHub and run together via “docker-compose pull” and “docker-compose up”. The usage and files are shared with team members both on Slack and GitHub so that every one of the team members was able to pull and test our app on their local machine.

Deployment on AWS

We have created an Amazon Web Service(AWS) account. First, we launched an EC2 instance. Then, we followed the below steps to deploy our app on this instance:

- Install nodejs 12x and npm using nvm
- Install our app
- Install dependencies
- Run the app
- Configure security group to access via public URL

We checked the log to find any errors. After we fixed these errors, we again deployed our app.

API Documentation

The API documentation includes endpoint documentations previously written by all group members, general description of the app and instructions for running it. All documentation is gathered in a .md file inside the practice-app. The documentation should have the most relevant information for understanding the endpoints and to make running the app possible.

4. Brief Summary of Work Done by Each Member

Name	Individual Work
Eylül Yalçınkaya	I attended group meetings as well as front-end team meetings, I wrote meeting notes for them. I wrote the endpoint /quote and its unit test. I wrote the front-end for /quote, /apod and /vendor/nearest. I contributed to the web app front-end design. I performed multiple code reviews and resolved conflicts for multiple pull requests. I wrote the documentation for the app with Katariina.
Emre Girgin	I did not miss any weekly meetings. Also, I made a pair meeting with Berke to Dockerize our app. I have participated in the development of API by designing one endpoint (/covid19) and a unit test for testing the data types of the response of the endpoint I've designed. The endpoint returns the latest data about COVID-19 cases in Turkey. I tested it using Postman. I reviewed 8 pull requests that merging different endpoints to the master

	branch and front-end implementation. Also, I made a research on Dockerizing the API to make it easy to deploy. Then Berke and I dockerized our app and composed it with MongoDB container.
Koray Cetin	I've attended the meetings. Implemented our rest api with mongodb connection. Handled the backend configurations and documented my code. Implemented the nearest vendor endpoint by using geolocation api of google, and mock documents in our database. Also explained my code to other members in a meeting. I've tested it with postman and unit tests via mocha library. I reviewed most of the pull requests and merged them to the master branch.
Salih Kasım Benli	I have attended most of the meetings. I was assigned to deploy the app to AWS Elastic Beanstalk along with Olcayto. I have developed the list of nearest hospitals API, and written unit tests for it. My API uses Google's places API to determine the nearest medical centers. I have reviewed and given feedback to others. I have helped the front end team, had meetings with them to illustrate how to communicate with API. I have additionally added some functionality to the front end part. I have also solved conflicts that occurred as merging. Olcayto and I have deployed the last version to AWS EB and make sure it is working as expected. We collected feedback for broken parts and deployed updated versions. I have commented several times to give constructive feedback to follow conventions.
Berke Özdemir	I have attended the meetings. I've met up with Emre to dockerize our app. I have written the dockerfile then worked with

	him to dockerize our app and compose it with MongoDB container. I've implemented '/suggestBook' endpoint to contact to goodreads api a to suggest similar books to the name of the book given in the query. I have written the test for the endpoint and reviewed pull requests.
Burak Çuhadar	I have attended all of the group meetings. I implemented an endpoint for getting news about Covid-19(/covid/news). I also wrote unit tests for that endpoint. I have reviewed the pull requests for endpoints implemented by the other members and checked whether they contain any bugs. I also reviewed the pull requests for frontend, dockerization of our application and the deployment. I evaluated the tools and processes we have used to manage our project in this report.
Çağrı Çiftçi	I've attended regular meetings and front-end meetings. I was assigned to write frontend with some group members and to write evaluation of deliverables. I've implemented /weather-important-cities and its unit test. I wrote the front-end for my endpoint and I did some research for fixing bugs about the front-end. I reviewed some of the pull requests. I generally used Postman to test endpoints.
M.Olcayto Türker	I've attended all of the group meetings. I wrote the executive summary for Milestone 2 report. I was assigned implementing /tryusd endpoint for our practice-app. I also wrote unit test for this endpoint. I reviewed other members' pull requests and tested them on my computer. Also, Salih and I deployed the project to AWS and updated the deployment according to feedbacks.

Mehmet Erdinc Oguz	I've attended all of the group meetings, also the front-end meetings. I made research about RestAPIs and found APIs to use for our practice-app. I applied for some of the tokens that we used in our API implementation. I've implemented the /random-playlist endpoint and unit tests for it. I participated in implementation of the front-end. I created a template for front-end and implemented three of the components for endpoints, and also contributed to the code in general during front-end implementation. We met 4 times during the implementation of the front-end and we teach stuff to another and make decisions about implementation during these meetings. I reviewed 13 PRs and gave feedback when needed. I pulled branches to test new changes in code and run it on my computer.
Berkay Alkan	I have attended all of the group meetings and front-end meetings. I have implemented the /weather/daily endpoint and unit test for it. I participated in implementation of the front-end. I wrote the /weather/daily, /covid/news and /tryusd endpoints in front-end part. I contributed on the design of the page. I reviewed multiple pull requests, gave feedback to these and resolved some bugs and conflicts. I used Postman to check the endpoints and pulled the new branches to my local to test them.
Katariina Korolainen	I attended one group meeting. I implemented the /apod endpoint and its unit test. I pulled some branches to test them locally and approved them. I wrote documentation for the app with Eylül and compiled this Milestone report for submitting.

5. Evaluation of Tools and Processes

Github

For this milestone the most heavily used feature on Github was pull requests. Using pull requests makes the process of code reviewing easier and provides a systematic way to handle changes made to the code. To implement our group's API each member developed a different endpoint on different branches. Then we created pull requests to make the team members review the code and find bugs. The interface provided by Github was intuitive to use. One can see the files changed, comment on a specific part of the code or report if there is some bug in the code. Finally the pull request can be merged when all the team members approve the changes. The merging process can be made automatically if there aren't any conflicts or thanks to the conflict editor interface provided by Github the conflicts can be resolved easily. Overall, handling the changes made to the code was manageable and easy using pull requests and Github.

We also used issues and wiki page features of Github. Using issues makes it easy to track the progress of tasks assigned to the members. There is also a feature for handling milestones in Github. One can assign issues to a milestone and see the number of issues open and closed for that milestone. Github also provides a nice interface to show the percentage of the issues closed for a milestone which provided us to track our progress on this milestone visually. As we did before we also used our wiki page to manage our meetings and documentation.

Slack

We continued to use Slack as our communication channel. When we had questions about implementing the backend and frontend for our application, we asked them using Slack to the members who are experienced with development of these kinds of things. We also used it to decide on meeting times and share the file containing our api keys which should be kept private to the group. To summarize, Slack fulfilled the need for a private communication channel for the group.

6. API Documentation and URL

Base URL: http://localhost:8080/

Base URL on Amazon server:
http://bounswe2020group4.eu-west-1.elasticbeanstalk.com

What is this app and why you should try it

With the COVID-19 pandemic around, there might be all kinds of information you wish you had. Whether you need to find the current numbers or the nearest hospitals, get suggestions for books and playlists or need some words for inspiration, our app has it all for you in just one place.

How to run the app

Once you are inside the practice-app folder, install docker and run `commands.sh`.

Web app is available on the base URL, you will see a website that shows the responses from our 11 endpoints.

Endpoints

All of the endpoints returns responses in JSON format.

1. Random quote

Returns an inspirational quote from a famous, historical person.

Endpoint: /quote

Method: GET

Authorization: None

Parameters: None

Response: { status: { success: true, code: 200 }, error: false, data: { quote: "In three words I can sum up everything I've learned about life: it goes on.", author: "Robert Frost" } }

2. Corona Playlist

Gets the playlists containing the string "coronavirus" from Spotify Web API and returns a randomly chosen one.

Endpoint: /random-playlist

Method: GET

Authorization: API key from .env file

Parameters: None

Response: { status: { success: true, code: 200 }, error: false, data: { playlist: { randomPlaylist.id: "2KkboYqP5llsT9gU7wcbco", name: "coronavirus", description: "", owner: "Liliana Mendoza", } } }

3. TRY-USD Rate

Returns the latest TRY-USD currency rate.

Endpoint: /tryusd

Method: GET

Authorization: API key from .env file

Parameters: None

Response: { status: { success: true, code: 200 }, error: false, data: { "TRY":6.8085 } }

4. Nearest Vendor

Gets the user location from Google API and vendor objects from the database. Calculates distance between vendors and the user and responds with the nearest vendor.

Endpoint: /vendor/nearest

Method: GET

Authorization: API key from .env file

Parameters: None

Response: { status: { success: true, code: 200 }, error: false, data: { vendor: { name: 'John', location: { latitude: 32.23, langitude: 38.21 } }, distance: 32811.13 } }

5. Nearest Hospitals

Gets the list of nearest hospitals to a location. Accepts latitude and longitude and optionally radius in meters. Due to Google API restrictions, only 1000 request per 24 hours can be made.

Endpoint: /nearesthospitals

Method: GET

Authorization: API key from .env file

Parameters: { lat: 41.0862, long: 29.0444, radius: 1000 }

Response: { status: { success: true, code: 200 }, error: false, data: { count: 3, names: ["Bahçeşehir University Health Center", "T. C. The Ministry of Health Sinanpaşa Family Health Center", "İstanbul Medical Group"] } } }

6. Current Covid19 numbers

Gets the lastest update on the numbers of Coronovirus cases in Turkey.

Endpoint: /covid19

Method: GET

Authorization: None

Parameters: None

Response: { status: { success: true, code: 200 }, error: false, data: { NewConfirmed: 1114, TotalConfirmed: 139771, NewDeaths: 55, TotalDeaths: 3841, NewRecovered: 3089, TotalRecovered: 95780, ActiveCases: 40150, lastUpdate: "2020-05-12" }

7. Covid19 News

Returns up to ten news articles about COVID-19.

Endpoint: /covid/news

Method: GET

Authorization: API key from .env file

Parameters: None

Response: { status: { success: true, code: 200 }, data: { articles: [{ title: 'Headline for the first article', description: 'A short description of this article', url: 'https://www.example.org' }, { title: 'Headline for the second article', description: 'A short description of this article', url: 'https://www.example2.org' }, ... //the number of articles varies, max num = 10] } }

8. Weather in Istanbul

Returns the weather of Istanbul for the next 8 days.

Endpoint: /weather/daily

Method: GET

Authorization: API key from .env file

Parameters: None

Response: { status: { success: true, code: 200 }, data: { day1: { minTemp: 18, maxTemp: 26, desc: 'broken clouds', icon: '04d' }, day2: { minTemp: 17, maxTemp: 27, desc: 'broken clouds', icon: '04d' }, day3: { minTemp: 19, maxTemp: 26, desc: 'overcast clouds', icon: '03d' }, day4: { minTemp: 16, maxTemp: 20, desc: 'light rain', icon: '09d' }, day5: { minTemp: 15, maxTemp: 19, desc: 'broken clouds', icon: '04d' }, day6: { minTemp: 14, maxTemp: 18, desc: 'broken clouds', icon: '04d' }, day7: { minTemp: 13, maxTemp: 18, desc: 'clear sky', icon: '01d' } } }

9. Weather in Important Cities

Returns the current weather of some important cities.

Endpoint: /weather-important-cities

Method: GET

Authorization: API key from .env file

Parameters: None

Response: { status: { success: true, code: 200 }, data: { currentWeathers: [{ cityName: 'name of the City1', currentTemp: 'current temperature for given city', description: 'A short description of current weather', iconUrl: 'https://www.example.org' }, { cityName: 'name of the City2', currentTemp: 'current temperature for given city', description: 'A short description of current weather', iconUrl: 'https://www.example2.org' }, ... // max num = 20] } }

10. Book Suggestions

Makes a search api call to the Goodreads API with a book name given by the user, gets the first book's ID in the search results and gathers all relevant info on the book by using the ID. Returns a list of similar books from the query result.

Endpoint: /suggestBook

Method: GET

Authorization: API key from .env file

Parameters: None

11. Astronomy Picture of the Day

Gets NASA's Astronomy Picture of the Day.

Endpoint: /apod

Method: GET

Authorization: API key from .env file

Parameters: None

Response: { status: { success: true, code: 200 }, data: { apodURL: ""https://apod.nasa.gov/apod/image/1707/M63-HST-Subaru-S1024.jpg", apodTitle: "Messier 63: The Sunflower Galaxy" } }