Executive Summary

Summary of the overall project

We prepared a practice-app that provides many different functionalities. Essentially, everyone created their endpoint, providing a separate functionality that they would like to work on. We decided on the API's we would like to use, then implemented on separate branches. Everything came together in the master branch with pull requests.

Deployment URLs

- We have used Swagger to document our Rest API. You can see the Swagger UI here
- Frontend: Deployment

Basic Functionality

We have many different sub-applications you can visit.

- Playing chess
- Analyzing Tweets
- Currency Converter
- Listing cryptocurrency prices
- Quiz Creator
- Lyrics search
- Posts
- Email verification
- Watchlist
- User Authentication

Challenges we met

- Free tier EC2 instances are not powerful enough to serve, even to build, modern server side rendered frameworks. We could not build our frontend since EC2 instance do not have enough ram and CPU credits to complete the process. We had to restart the instance via AWS console. Hence, we had to serve our application in development mode, without getting an optimized build.
- We tried to deploy our applications via HTTPS protocol; however, Let's Encrypt do not give certificates to EC2 urls. The only solution we could find was buying a domain from a provider, which we did not want to do in the first place.
- Using HTTP streams was required for `Chess` and `Twitter` sub-projects. We had a lot of truble to set up, use, and remove streams.
- Some of the external APIs we used were not stable and caused a lot of debug sessions only to realize the problem was not in our project.

Status of deliverables

- Milestone Report: Done
- Requirements:

    - Requirements - Done
- Design:
    - Use Case Diagrams - Done
    - Class Diagrams - Done
    - Sequence Diagrams - Done
- Release from Github: Done
- Code: Ready
- Wiki: Up to date
- Meeting Notes: Up to date

Personel Efforts

Kadir Ersoy

- Researched Auth0 API #168.
- Created users endpoint.
- Added Login and Register features #212, #182.
- Added getUsername feature #214.
- Created authorization middleware239.
- Created Main page#213.
- Wrote unit tests for users endpoint #216.
- Reviewed TweetAnalysis branch pr.
- Documented API using Swagger #222.
- Documented Users requirements #217.
- Designed Use Cases for users endpoint #223.
- Designed User Class #223.
- Designed Sequence Diagrams for "Users" use cases #223.
- Documented Project Plan using projectLibre #215.
- Documented Individual Report #241

Mehmet Gökberk Arslan

- Done research to learn and better understand the tools we are going to use for our implementation such as swagger, mongoDB and postman. The issue for the tools we have used: #236
- Decided to implement a cryptocurrency conversion API and learned express.js, next.js in order to implement my part of the practice-app (Meeting Notes 13)
- Created requirements for the cryptocurrency functionality of the app #226
- Created UML diagrams cryptocurrency part of the app (Use case, class, sequence) #208
- Implemented the required frontend and backend for the cryptocurrency API #169
- Implemented unit tests and prepared API documentation using swagger for the cryptocurrency API and then added them to my code #209
- Created a pull request to merge my code into master (Pull#225)
- Attended all group meetings related to practice-app except one.

Nurlan Dadashov

- Developed initial setup for backend #153.

- Developed initial setup for frontend #154.
- Researched public chess APIs - Lichess API #156.
- Developed backend for game creation #157.
- Developed frontend for game creation #158.
- Developed backend for playing chess game #161.
- Developed frontend for playing chess game #162.
- Developed backend for previous chess game #163.
- Developed frontend for previous chess game #164.
- Researched Express unit testing and Developed unit tests for the server code #177.
- Documented endpoints using Swagger #201.
- Documented User and System requirements for Chess #186.
- Documented Non-functional requirements #185.
- Designed Use case/Class/Sequence Diagrams for Chess feature #202.
- Attended all team meetings.

Muhammet Şen

- Implementing endpoints for managing stream rules #171
- Implementing endpoints for managing stream rules #172
- Analyzing tweets' average text lengths and number of words #173
- Integrating Swagger for Express #178
- Unit testing tweetSearch endpoints #183
- Tracking Class Diagrams #188
- Use Case/ Class/ Sequence Diagrams for Tweet Analytics #203
- Containerization of the application#220
- Tracking statuses of requirements #224
- Documenting User and System Requirements for Tweets #233
- Writing Executive Summary #234
- Reviewed @nurlandadashov02's code

Bilal Aytekin

- Researched TheMovieDB API.
- Researched the tools in #236.
- Created the movies endpoint.
- Covered Issue #174.
- Documented API using Swagger #222.
- Documented requirements for the movies endpoint #207.
- Designed use case, class and sequence diagrams for the movies endpoint #247.

Burak Yilmaz

- Created a release for the practice-app here #248.
- Created class diagrams for currency here #246.
- Created use case diagrams for currency here #245.
- Created sequence diagrams for currency here #198.
- Documented user and system requirements for currency here #197.
- Documented API by Swagger for currency here #194.

- Created unit tests for currency here #192.
- Created backend for currency here #191.
- Created frontend for currency here #190.
- Reviewed the pr Swagger Integration for Express #180.
- Reviewed the pr Containerization #221.
- Reviewed the pr Practice App/Lyrics #253.
- Attended all team meetings

Mertcan Özkan

- Researched [article search apı].(https://developer.nytimes.com/docs/articlesearch-product/1/overview)#227.
- Researched the tools in #236.
- Created the articles endpoint. #240
- Documenting User and System Requirements for Artciles #240
- Designed use case, class and sequence diagrams for the movies endpoint #247.
- Attended all team meetings.

Arif Akbaba

- Done research to learn and better understand the tools we are going to use for our implementation. #187
- Decided to implement a post API and learned express.js, next.js in order to implement my part of the practice-app (Meeting Notes 13)
- Created requirements for the post functionality of the app #226
- Created UML diagrams post part of the app (Use case, class, sequence) #218
- Implemented the required frontend and backend for the post #187
- Implemented unit tests (#232
- Prepared API documentation using swagger for the gorest API and then added them to my code (#231
- Created a pull request to merge my code into master Pull#200
- Attended all group meetings related to practice-app except one.

Hatice Şule Erkul

- Researched about different free APIs, after that I chose Open Trivia DB API
- Wrote user and system requirements #242
- Created class, use case and sequence diagrams #235
- Implemented endpoints : #170 & PR
    - `/quiz/categories`
    - `/quiz/new_quiz`
    - `quiz/view/all`
    - `quiz/view/[quiz_id]`
- Implemented unit tests for the endpoints. #210
- Done API documentation on Swagger #211
- Created pull request to merge my code into master PR
- Done a review on @arfakbaba's PR
- Attended team meetings

Bengisu Takkin

- Created a release for the practice-app
- Created class diagrams for Email Verifier #251
- Created use case diagrams for Email Verifier #251
- Created sequence diagrams for Email Verifier #251
- Documented user and system requirements for Email Verifier #206
- Reviewed @bilalcim's API (TheMovieDB API) #174
- Documented the API by Swagger for Email Verifier #204
- Created unit tests for Email Verifier #205
- Created backend for Email Verifier #175
- Created frontend for Email Verifier #175
- Attended all team meetings

Salim Kemal Tirit

- Researched about different free APIs, after that I chose Genius
- Created class diagrams for Lyrics
- Created use case diagrams for Lyrics
- Created sequence diagrams for Lyrics
- Documented user and system requirements for Lyrics
- Reviewed @Brunettow 's API (Email Verifier) https://github.com/bounswe/bounswe2022group3/issues/175
- Documented the API by Swagger for Lyrics
- Created unit tests for Lyrics
- Created backend for Lyrics
- Created frontend for Lyrics
- Attended team meetings

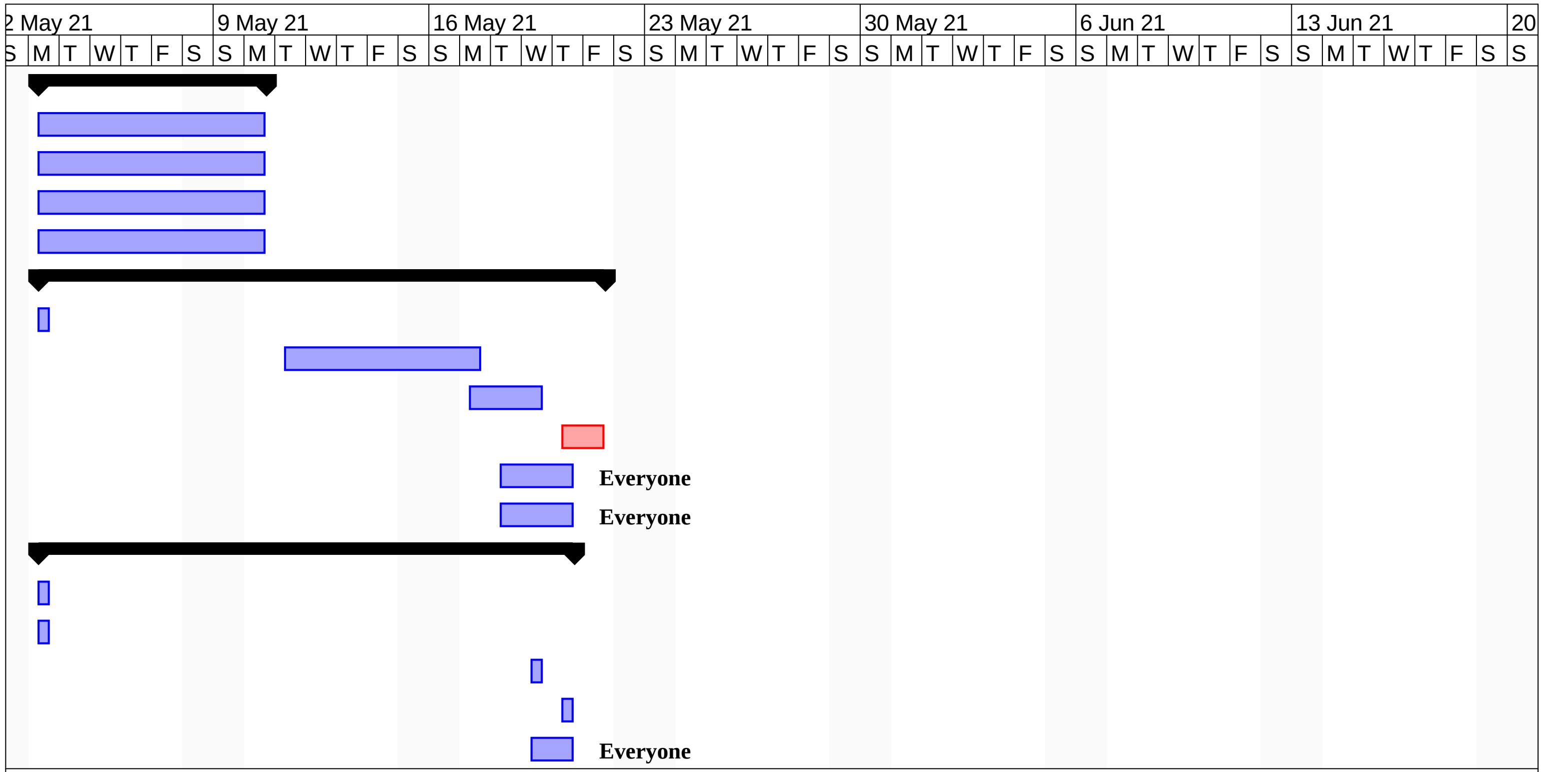
Tool Evaluation

- Github
  Github is the main platform we used in our project. Basically, we did everything and put everything on Github. In the meetings, after discussing the meeting agenda and deciding who will do what we wanted everyone to open a related issue for that task. We have used the issue system to track our progress. It increased our collaboration since we could comment on issues. We have used the branches to work together. Everyone created a seperate branch for their part. The branches were very effective. We used pull requests to check conflicts or errors before merging. We checked each others work.
- Swagger
  Swagger is the tool we have used in our API documentation creation. It is easy to use and write. It made our documentation process easier. The only problem may be writing everything without any help. One may get lost here. Overall it is a good tool and helped us very much.

- Postman
  Postman is the tool we used to test API requests and responses. It has a nice user interface and there are many resources to learn how to use it. It helped us in every stage of our implementation since we first tested the external APIs and our own APIs.
- MongoDB
  We used mongoDB as our database. It makes database usage easier and more effective. It has nice documentation and very large community. It is an important tool to learn.
- Lucid.app
  We used the lucid app to draw our diagrams. It allows us to work together on the same document. For example in the class diagrams everybody created their own class that they implemented. For the use case diagram, we build it part by part in the same graph and reviewed it in a meeting. It provides a solid, collaborative environment with many beautiful options of shapes. It is easy to use, moving objects and connecting them with arrows is pretty easy.
- PlantUML
  We used plantUML to create sequence diagrams. It takes care of the drawing part itself and it is closer to coding. It creates arrows, "alt" or "for" boxes in a structured fashion. It creates the actors, participants, and database automatically. It is also symmetric about these one on the top and one on the bottom. It can also be used to create other types of diagrams but it is easier to do them with lucid app
- ProjectLibre
  We used project libre to prepare our project plan. It is open-source, free software. It is a desktop application and it is stated that there will be a cloud version. Apparently, the cloud version is being waited for a long time. Since it is only available as a desktop application we could not efficiently work on it. It was not practical to share the screen on zoom and try to fill a table together. It is not stable on different operating systems. We didn't like this program at all. It didn't make our job easier than it is. We are very disappointed about this.
- Slack
  We used slack as a platform to communicate with our friends and teachers about the project. We asked questions and checked the answers. We created a separate workspace for our group only. We used it to communicate between the group members, share helpful links, and bring together questions. Creating different text channels helped us increase our productivity since we could easily follow multiple topics at a time.
- WhatsApp
  We used WhatsApp to communicate instantly and discuss small things. Since we all are used to checking WhatsApp frequently it helped us communicate. WhatsApp is generally hard to track so we didn't use it for very important subjects that need to be checked by all team members. Generally, when we needed help or were not sure about a subject we used WhatsApp. We also used it for general chit-chat from time to time.
- Zoom
  We used zoom for meetings. The problem with the zoom is that since there is a time limit to the free version of the zoom our meetings got interrupted. We shared our screens when we needed them and used the chat feature to share some links.

Project Plan

	ⓘ	Name	Duration	Start	Finish	Predecessors	Resource Names	F	S
1	❗	**Research**	**6 days?**	**5/3/21 8:00 AM**	**5/10/21 5:00 PM**		**Everyone**		
2		Express.js	6 days?	5/3/21 8:00 AM	5/10/21 5:00 PM				
3		Next.js	6 days?	5/3/21 8:00 AM	5/10/21 5:00 PM				
4		API usage	6 days?	5/3/21 8:00 AM	5/10/21 5:00 PM				
5		Javascript	6 days?	5/3/21 8:00 AM	5/10/21 5:00 PM				
6	❗	**Implementation**	**15 days?**	**5/3/21 8:00 AM**	**5/21/21 5:00 PM**		**Everyone**		
7		Deciding on APIs to use	1 day?	5/3/21 8:00 AM	5/3/21 5:00 PM				
8	📅	Implementing Backend-Frontend	5 days?	5/11/21 8:00 AM	5/17/21 5:00 PM				
9	📅	Unit-testing	3 days?	5/17/21 8:00 AM	5/19/21 5:00 PM				
10	📅	Deployment	2 days?	5/20/21 8:00 AM	5/21/21 5:00 PM				
11	📅	Requirements	3 days?	5/18/21 8:00 AM	5/20/21 5:00 PM		Everyone		
12	📅	Design	3 days?	5/18/21 8:00 AM	5/20/21 5:00 PM		Everyone		
13	❗	**Meetings**	**14 days?**	**5/3/21 8:00 AM**	**5/20/21 5:00 PM**		**Everyone**		
14		First Meeting	1 day?	5/3/21 8:00 AM	5/3/21 5:00 PM				
15		Review for Backend	1 day?	5/3/21 8:00 AM	5/3/21 5:00 PM				
16	📅	Total Review of implementation	1 day?	5/19/21 8:00 AM	5/19/21 5:00 PM				
17	📅	Review of report	1 day?	5/20/21 8:00 AM	5/20/21 5:00 PM				
18	📅	Report	2 days?	5/19/21 8:00 AM	5/20/21 5:00 PM		Everyone		


2 May 21
9 May 21
16 May 21
23 May 21
30 May 21
6 Jun 21
13 Jun 21
20
S M T W T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S
Everyone
Everyone
Everyone