# CMPE 352
# Milestone 2 Report

## Group 9


Mert Alkan

Ahmet Melih Aydoğdu

Zehranaz Canfes

Mustafa Emir Çolak

Ömer Barış Erkek

Kadir Kalkan

Melih Özcan

Ahmet İbrahim Şentürk

Niyazi Ülke

# Table of Contents

**Table of Contents**
**1. Executive Summary**
- 1.1. Introduction
- 1.2. Project Status and Work Done So Far
- 1.3. Moving Forward
- 1.4. Challenges We Met as a Group

**2. Project**
- 2.1. URI of the Deployed Application
- 2.2. API URL
- 2.3. Functionality of the Project

**3. List and Status of Deliverables**
**4. Evaluation of Tools and Processes**
- 4.1. Evaluation of Tools
  - 4.1.2. AWS
  - 4.1.3. Docker
  - 4.1.4. Git
  - 4.1.5. GitHub
  - 4.1.6. Visual Studio Code
  - 4.1.7. Django
  - 4.1.8. Postman
  - 4.1.9. Swagger
  - 4.1.10. MySQL
  - 4.1.11. Discord
  - 4.1.12 WhatsApp
- 4.2. Evaluation of Processes
  - 4.2.1. Team Meetings
  - 4.2.2. TA Meetings
  - 4.2.3. Issue Creation
  - 4.2.4. Code Reviews

**5. Project Plan**
**6. Work Done by Each Team Member**
**7. Conclusion**

# 1. Executive Summary

## 1.1. Introduction

As a group in CmpE 352 course, we aim to design a platform for people who want to share their stories based on locations they have been to and experienced. We are to implement a web and an Android application for our users to share stories with others and view them.
In this Milestone, we were to implement a practice app that uses various RESTful APIs and has multiple functionalities. Each team member was asked to implement an endpoint for the practice app. We decided for the practice app to have similar functionalities to our main project such as sharing location-based stories and viewing them. Users can view the story's title, content, location on the map, its owner, the weather of that location, nearby places, and cities. In addition, they can translate the story, view a daily joke and a quote related to the story. The practice app will help us implement the project next semester as well. We were finally assigned to design and implement a front end for our practice app and deploy it using Docker and AWS.

## 1.2. Project Status and Work Done So Far

Before the practice app, we have finished our first Milestone that included the following:
We started our project by defining the requirements of the project. We prepared questions to ask the customer to get an insight into how the project should look. We then created mockups and scenarios to have a visual of the application. We designed Use Case Diagrams, Class Diagrams, Sequence Diagrams to help us understand the main functionalities of the project and how to implement them in the future. We created a Project Plan and RAM to plan our next steps and see what we have done so far.

After planning and learning the necessities of designing a project, we became ready to implement it. However, starting with the implementation also requires us to learn Git, GitHub features, branches, and pull requests in more detail. Also, we had to learn Docker and AWS and how to use CI/CD in developing software. For this purpose, we have done the necessary research and attended PS sessions. To apply what we learned during the first Milestone and the PSs, we started with a **practice app**.

During the planning of the practice app assignment, we designed our app similar to our main project. Our app provides its users with the ability to share their stories including a title, username, location, tag, and content. We were asked to implement the backend using different RESTful APIs and also a frontend for our functionalities. For more information on the APIs and the basic functionalities please refer to Project of this Milestone Report.

During the implementation of the practice app, to be able to work collaboratively, we worked in different branches under *main*. We all implemented an endpoint using Django and different RESTful APIs. We then tested our endpoints and merged all branches to *main*.
After merging, we moved on with the frontend and Docker. We split into two groups for that purpose and reviewed each other's works at each step. Reviewing each other's codes and having regular meetings helped us be up to date with each other and improve our implementation.

After the endpoints were tested, we created a Docker image and implemented the frontend. We then deployed the app on AWS and created stories for the database.

Finally, we created our API documentation and the Milestone Report.

At each step of the practice app, we have worked on bugs and improved our code. We have reviewed each other's works and had regular meetings with the TA and with the group. We are ready to move forward.

## 1.3. Moving Forward

In Milestone 1, we learned how to plan a software project. In Milestone2, the practice app aimed to teach us how to implement a planned project and what to expect during the implementation. In a very short time, we have designed and created a small app that we planned. The practice app has met our expectations. It, helped us understand the road of planning and implementing a software project. It helped us see the challenges we can face in the next step of our main project. It taught us the importance of doing research and planning. Most importantly, it taught us the importance of collaborative work. We have seen that, if we work together and plan, we can overcome the challenges we meet in the future.

## 1.4. Challenges We Met as a Group

First of all, the major challenge for all of us was to get familiar with all the new concepts that were required to implement the practice app. Django in general, the REST Framework and Dockerization and Deployment via AWS were all fields that required research and practice. Another issue was problems with external APIs. It was a bit overwhelming to get keys that were necessary for some of the APIs. Other than this, sometimes the APIs were down or simply stopped working for some reason at times.
Last of all, we struggled a little when dealing with the frontend. The frontend was supposed to show our efforts and functionalities in a manner that would be pleasing to the eye. The end result was satisfying for us all but we struggled a little in order to get there.
In general, the problems stemmed from the fact that these were all new to us and we had to leave our comfort zone to learn new things and struggle a bit.  We tackled all

these problems quite nicely by cooperating with each other and taking on the challenges together.

# 2. Project

## 2.1. URI of the Deployed Application

The web application can be accessed by using either of these:

- http://3.129.194.233/
- http://ec2-3-129-194-233.us-east-2.compute.amazonaws.com/

The stories we have so far can be viewed on the homepage or a new story can be sent with the “Send us a story!” button. Additionally, you can see a random joke at the top of the homepage.

In order to send a post, five pieces of information about the post must be provided:

- Username: Username of the person who wants to share a post
- Location: The specific location of the post(ex: ‘Belgrad Ormanı’,’Boğaziçi Kuzey Kampüs İstanbul’, ‘London’)
- Tag: A one-word tag associated with the post (ex: ‘Programming’, ‘happy’). If a quote with the given tag is not found, it returns a random quote
- Title: The title of the post. The title must be at least two words
- Story: The story of the post

In order to see a story we have so far you have to click on its button on the homepage. At the top of the post’s page again the random joke can be seen.

The post’s page contains the fllowing information:

- Title of the post
- Location of the post
- Weather for the post’s locaiton
- Date and time the post was shared
- COVID-19 cases at the post’s location
- Username of the person who shared the post
- Story of the post
- An image showing the post’s location
- Places close to the post’s location and its vicinity
- Cities close to the post’s location
- Buttons to translate the story of post in three languages: German, Spanish and Turkish
- Quote with the given tag

# 2.2. API URL

All of the endpoints of the developed API are provided below with a brief explanation. The prefix URL of all of the endpoints is: http://3.129.194.233/api

- **/story** endpoint takes no parameter and returns all stories from the database. A returned instance consists of id, date, location, tag, title, story, longitude, latitude of the story and name of the user. There is also a notifyAdmin data. This will be true if the story contains abusive words.
- **/story/int:id** endpoint takes one integer parameter which represents the story ID, and provides the functionalities GET, PUT or DELETE of the specified story.
- **/storypost** endpoint takes no parameter. It posts a story in the form of a JSON string like {title: 'TITLE', story:'STORY', 'location': 'LOCATION', 'tag':'TAG', 'name': 'USERNAME'} which is given by the user via the frontend. Using the /location endpoint it finds the coordinates of a location. Then, it sends the story string to Tisane API to get a result in JSON form. If the result suggests there was abusive content it will set notify_admin = True. At last, it saves the post into the DB.
- **/create** endpoint is the frontend for post creation. It posts various parameters it gets from the user to the endpoint /storypost. It also uses js alerts to send the user some feedback in the cases of successful or unsuccessful post creation.
- **/flagged_stories** endpoint takes no parameter and returns the flagged stories (notify_admin= True) in the form: { 'flagged_stories': [POST IDS] }
- **/location** endpoint takes no parameter and returns the formatted addresses, longitudes and latitudes of all the stories in the database and their corresponding IDs.
- **/location/int:id** endpoint takes one integer parameter which represents the story ID, and returns formatted address, longitude and latitude of specified story.
- **/location/map/int:id** endpoint takes one integer parameter which represents the story ID, and returns the map of the specified story by using its location. A marker is put on the location of the story. The map is also shown in the frontend.
- **/weather/int:id** endpoint takes one integer parameter which represents the story ID, then the GET request will return that specified story's weather information (condition, temperature, feel temperature, wind speed, country, time zone and a comment). The comment is implemented based on the temperature of the given location.
- **/quote/int:id** endpoint takes one integer parameter which represents the story ID, and returns a quote that has the same tag with the specified story.
- **/quote/location/int:id** endpoint takes one integer parameter which represents the story ID, and returns a quote that contains the location of the story in it.
- **/postquote/int:id** endpoint takes one integer parameter which represents the quote ID, and saves the quote with that id to the database with like-number increased. This way, if a quote is liked by the user, that quote is added to the database for later use. If the liked quote is already in the database, then the code only increases its number of likes by 1 and saves it.
- **/nearbyplaces/int:id** endpoint takes one integer parameter which represents the story ID, and it returns a list of places near 2000 meters of the location of the specified story where each place has 'name', 'location', and 'vicinity' properties; 'vicinity' property may be null.

- **/city/int:id** endpoint takes one integer parameter which represents the story ID, and returns the name and the countries of the nearby cities for the specified story's location.
- **/joke/str:category** endpoint takes one string parameter which represents a category, then it returns a random joke in the given category, along with it's creation date, id and URL.
- **/covid/int:id** endpoint takes one integer parameter which represents the story ID, and it returns the Covid-19 related metrics (country, day, new cases, new deaths, total active cases) for the country of the specified story's location. To find the country of the story's location it uses the /city/int:id endpoint.
- **/story/int:id/translate__str:target** endpoint takes two parameters: an integer parameter which represents the story ID and a string parameter that represents the target language for the translation of the specified story. It then returns the translation of the specified story to the target language.

## 2.3. Functionality of the Project

- Main functionality of our practice app is similiar to our main project. Stories are posted by our users on our website. Users can also read these stories and see some information about fields of that story. Side functionalities are in the shape of APIs.
- While posting a story, it is tested by Post Story API if it has some abusive context in it. If there is that story is flagged.
- Weather functionality gets the weather information of a story's location and returns it with a comment.
- Translation functionality translates the story into different languages.
- Each story has a tag. Quotes functionality takes that tag and returns a quote related with that tag.
- Nearby places functionality takes the location of the story and returns the nearest places to that location.
- Location functionality provides abstraction to user. When a story is being posted, user only enters the name of the location. But with this functionality, that story is stored in our database with both location and longitude,latitudes.
- Joke functionality gets a random joke and puts it into top of our webpages.
- Covid functionality gets the information of story's country COVID statistics and shows it in that story's page.
- Cities near the location of our story is retrieved by City API. It is also shown on the story's page.
- Each of these functionalities are used on our frontend. Any information related to story is shown on the story's page.

# 3. List and Status of Deliverables

This section lists all deliverables for practice-app; with their delivery status, due date, and delivery date.

| Deliverable Name | Delivery Status | Due Date | Delivery Date |
|---|---|---|---|
| Deliverables Report | Delivered | 10.06.2021 | 10.06.2021 |
| Group Report | Delivered | 13.06.2021 | 13.06.2021 |
| Individual Reports | Delivered (for every member) | 13.06.2021 | 13.06.2021 |
| Code | Delivered | 10.06.2021 | 10.06.2021 (v0.1 release), 12.06.2021 (final release) |
| Deployed App | Delivered | 10.06.2021 | 10.06.2021 (v0.1 release), 12.06.2021 (final release) |
| API Documentations | Delivered (for every member) | 13.06.2021 | 13.06.2021 |
| Other wiki updates | Delivered | 13.06.2021 | 13.06.2021 |

# 4. Evaluation of Tools and Processes

In this section, the tools and processes used during the development of "practice-app" are briefly evaluated.

## 4.1. Evaluation of Tools

The team used several tools to develop the application. Each one of these tools serves different purposes. The evaluations include the opinions and experiences of team members.

### 4.1.2. AWS

AWS allowed us to deploy our application without much effort. There are plenty of tutorials that teach how one can deploy a containerized application. Also, the free services seem sufficient to create simple websites. We have not had huge problems regarding AWS, but since it was our first use, we spent some time understanding it.

### 4.1.3. Docker

Docker is a virtualization tool commonly used in modern software development. Understanding how Docker works took some time. However, Docker containers allowed us to deploy the application quickly. Once you create a container correctly, it is for sure that the application will run in the correct environment. Moreover, the deployment of Docker containers is facile. The only disadvantage is that Windows users need to install WSL(Windows Subsystem for Linux) to work with Docker properly.

### 4.1.4. Git

Git is a popular version control system that allows people to develop code in an organized manner. Nothing gets lost when one accurately use Git.  In addition, everything becomes much more trackable.

Thanks to the problem sessions, we understood the basics of collaborative working on Git very well.  For most team members, using Git commands was a little frightening at first. We feared, for instance, we could spoil the whole repository by pushing something wrong. In the end, no such thing occurred because we created separate branches for each functionality. We had some insignificant problems stemming from our inexperience; however, we do not think these are deficiencies of Git. Finding solutions for these problems were trivial because of the abundance of documentation and posts in forums such as StackOverflow.

### 4.1.5. GitHub

GitHub provides a beautiful user interface and environment that enhance the functionalities of Git. Firstly, GitHub Wiki allows us to make documentation for our project in the same environment as the code. Issues are beneficial for clarifying who is responsible for each task. They are also useful in referencing other wiki pages, issues, collaborators and pull requests.

Previously, we had not used GitHub for code development. The interface allowed us to examine any branch on GitHub just in seconds. Thanks to GitHub, code reviews were fast. The reviewers can quickly see changes made. They can also put comments on specific lines so that there is no need to check the whole code to detect the problem. During the development of "practice-app", we noticed that GitHub might occasionally become inaccessible for short periods, but this did not cause significant problems for us.

### 4.1.6. Visual Studio Code

Visual Studio Code(VSCode) is a code editor made by Microsoft. It supports a wide variety of plugins that simplify the development process. Integration with Git allows reverting local changes on specific lines. It also supports the use of the terminal without changing focus. We had decided to use VSCode because it supports connecting remotely with team members, but we did not try this functionality since we worked on different tasks. In summary, VSCode is a perfect tool for developers, and we have not noticed any problems.

### 4.1.7. Django

Django is a Web framework written for Python. Its Object-Relational Mapper(ORM), made it much easier to communicate with our MySQL server. Its built-in admin panel also helped us for creating, updating and deleting DB objects.
As we were building a REST API, we decided to use Django Rest Framework with Swagger. Django Rest Framework provides a further abstraction for interacting with DB objects named serializers. Using its GenericAPIView it is easier to divide the endpoints to GET, POST, PUT and DELETE requests.

### 4.1.8. Postman

Postman is a tool that is used for verifying the APIs you develop. We used it to test our API code while the frontend was not yet implemented. It allowed us to name our requests and save them for the purpose of using them again later instead of writing the same request each time.

### 4.1.9. Swagger

Swagger lets you document your APIs easily. We used the Django module version of Swagger in our project. This module documents your code automatically if you used the Django Rest framework while developing the API. We had to change some parts of our code, changing from bare Django to Django Rest constructs, for swagger to recognize it as an API. But it was a huge help overall.

### 4.1.10. MySQL

MySQL is a free-to-use database management system that we used in our project. Because it was easier to dockerize MySQL, as opposed to SQLite which comes as the default DB server in Django, we decided to use MySQL as our DB.
Django's migrations create the database tables automatically, and its ORM handles the communication with the DB server, so we did not have to tamper with the DB server that much.

### 4.1.11. Discord

We use discord as our main communication tool. Weekly meetings are held on discord. It has multiple voice and text channels so we can hold multiple meetings at the same time (was useful when we separated into multiple groups). It has a feature for pinning messages, we use that feature for saving important links. It allows multiple screen sharings at the same time which is useful at times.

### 4.1.12 WhatsApp

We used WhatsApp for short communications and finding some time for emergency meetings. It is useful for spontaneous communication but it has no way of providing structure. Because of that, we try to use discord for longer discussions.

## 4.2. Evaluation of Processes

Working as a team is an intricate task. Good processes ensure that a team might collaborate well. The processes adopted for the development of "practice-app" are evaluated in this section.

### 4.2.1. Team Meetings

The team held meetings very frequently. At meetings, the members discussed the distribution of tasks. We also completed some tasks directly at the meetings. For instance, some merge requests required us to discuss the changes. In addition, we needed the same files or the same pieces of code for different features; so we wrote some of them during the meetings.  Also, some team members had specific issues about Git, their codes or external APIs. We solved some of such problems at the meetings. Team members took meeting notes rotationally and posted them on GitHub Wiki for review.

### 4.2.2. TA Meetings

We hold weekly meetings with our TA. The teaching assistant explained concepts about software engineering to us in detail. We also asked questions related to the content and scope of the assignment. In a nutshell, TA meetings were very beneficial for the development of the application.

### 4.2.3. Issue Creation

We continued the same approach that each person should create issues for herself. Our repository has a high number of GitHub issues, and it seems trackable. Different from before, we have also created GitHub issues related to bugs and pull requests while developing the application.

### 4.2.4. Code Reviews

The team decided to create branches for each new feature. After implementation is completed, each team member created a pull request for her code. Many people reviewed pull requests that include significant changes. Code reviews helped us to merge changes without damaging the whole system. Also, we learned from each other while examining the codes.

# 5. Project Plan

| | Ad | Süre | Başlat | Bitirme | Önceki | Kaynak Adlar |
|---|---|---|---|---|---|---|
| 1 | **Research and Learn Tools** | **4,5 günler** | **22.05.2021 08:00** | **26.05.2021 20:00** | | |
| 2 | Research Different map APIs | 4,5 günler | 22.05.2021 08:00 | 26.05.2021 20:00 | | Everyone |
| 3 | Install Django and Start Learning | 4,5 günler | 22.05.2021 08:00 | 26.05.2021 20:00 | | Everyone |
| 4 | Finding API Functionalities for Project | 4,5 günler | 22.05.2021 08:00 | 26.05.2021 20:00 | | Everyone |
| 5 | Learn how to Use Git Properly | 4,5 günler | 22.05.2021 08:00 | 26.05.2021 20:00 | | Everyone |
| 6 | Research AWS and Dockerization | 4,5 günler | 22.05.2021 08:00 | 26.05.2021 20:00 | | Everyone |
| 7 | **Implementing API Functionalities** | **5 günler** | **26.05.2021 21:00** | **31.05.2021 21:00** | **1** | |
| 8 | Weather API for Getting the Weather of Location | 5 günler | 26.05.2021 21:00 | 31.05.2021 21:00 | | Mert Alkan |
| 9 | Location API for Getting the Map and Coordinates | 5 günler | 26.05.2021 21:00 | 31.05.2021 21:00 | | Ahmet Melih Aydodu |
| 10 | Quote API for Getting Quotes about Tags | 5 günler | 26.05.2021 21:00 | 31.05.2021 21:00 | | Zehranaz Canfes |
| 11 | Tisane API for Checking Abusive Content | 5 günler | 26.05.2021 21:00 | 31.05.2021 21:00 | | Mustafa Emir Çolak |
| 12 | Nearby Places API for Getting Near Places to Location | 5 günler | 26.05.2021 21:00 | 31.05.2021 21:00 | | Ömer Bar Erkek |
| 13 | City API for Getting Near Cities to Location | 5 günler | 26.05.2021 21:00 | 31.05.2021 21:00 | | Kadir Kalkan |
| 14 | Chuck Norries Jokes API for getting Random Jokes | 5 günler | 26.05.2021 21:00 | 31.05.2021 21:00 | | Melih Özcan |
| 15 | COVID-19 API for Getting Info about COVID-19 Cases | 5 günler | 26.05.2021 21:00 | 31.05.2021 21:00 | | Ahmet Ibrahim Sentürk |
| 16 | Translator API for Translating the Story | 5 günler | 26.05.2021 21:00 | 31.05.2021 21:00 | | Niyazi Ülke |
| 17 | Write Unittests for API Functionalities | 1 gün | 31.05.2021 21:00 | 01.06.2021 21:00 | 7 | Everyone |
| 18 | Create Pull Requests for API Functionalities | 1 gün | 01.06.2021 21:00 | 02.06.2021 21:00 | 7;17 | Everyone |
| 19 | Review Pull Requests | 2 günler | 02.06.2021 21:00 | 04.06.2021 21:00 | 18 | Everyone |
| 20 | Merging Pull Requests in Meeting 16 | 0,139 günler | 04.06.2021 21:00 | 05.06.2021 00:20 | 19 | Everyone |
| 21 | Implementing Frontend of the Application | 3,5 günler | 05.06.2021 08:00 | 08.06.2021 20:00 | 20 | Ahmet Ibrahim Sentürk;Kadi... |
| 22 | Dockerizing and Deploying Application | 3,5 günler | 05.06.2021 08:00 | 08.06.2021 20:00 | 20 | Ahmet Melih Aydodu;Zehra... |
| 23 | Deployment | 1 gün | 08.06.2021 20:00 | 09.06.2021 20:00 | 21;22 | Ahmet Melih Aydodu |
| 24 | Release Description of Our Application | 1,16 günler | 09.06.2021 20:00 | 10.06.2021 23:50 | 23 | Everyone |

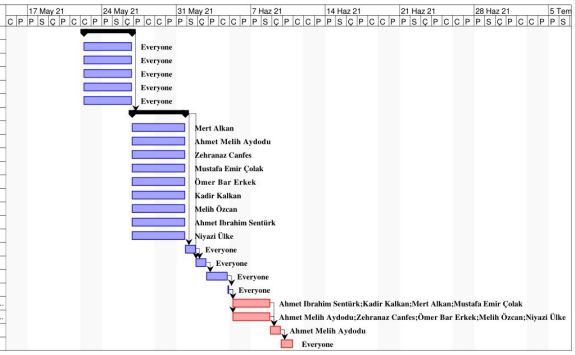


Figure 1: Project Plan for Practice-App

# 6. Work Done by Each Team Member

| Name | Work Done |
|---|---|
| *Mert Alkan* | • Learned about REST APIs, Django, HTML, HTTP, Postman. Issue#127<br>• Searched APIs to use one. Decided on OpenWeatherMap API.<br>• Learned about sending requests to third party API.<br>• Implemented Weather API for our practice app. Issue#113<br>• Learned about Unit Tests and how to implement them on Django.<br>• Wrote tests for my functionality.<br>• Created the documentation of my API. Issue#186<br>• Created the template for frontend of our practice-app. Issue#147<br>• Implemented the frontend functionality of Weather API, Translation API, Quotes API. Issue#157<br>• Opened pull request for my API, frontend template and my part of frontend functionalities. (#132,#158,#162)<br>• Reviewed many pull requests. (#129,#130,#131,#133,#134,#135,#136,#137,#148,#159,#160,#161,#164) (More information can be found on individual report)<br>• Reviewed other team members frontend implementations.<br>• Added styling to frontend.<br>• Learned Docker and reviewed my teammates progressions.<br>• Wrote about my API to Deliverables Report.<br>• Wrote Basic Functionality part of Milestone 2 Group Report. Issue#190<br>• Wrote individual part of Group Milestone Report 2. Issue#201<br>• Wrote individual Milestone Report 2. Issue#202<br>• Took notes on meeting 14. Meeting#14<br>• Attended all meetings. |

| Name | Work Done |
| --- | --- |
| *Ahmet Melih Aydoğdu* | • Learned Django and Django REST Framework<br>• Learned how to use unittest<br>• Researched different Map APIs<br>• Researched and worked on Git to use it effectively<br>• Researched and learned how to dockerize and deploy to AWS EC2<br>• Learned how to use Google Maps API<br>• Got Google Maps API key for project<br>• Implemented Location API by using geocoding of Google Maps API and documented it<br>• Implemented GET, PUT, DELETE functionalities of our application and documented it<br>• Implemented Swagger UI for our application (It can be accessed here)<br>• Made addition to some API functionalities to make them visible in Swagger UI<br>• Implemented unittest for my API functionalities<br>• Pushed the image of our application to Docker Hub<br>• Deployed our application to AWS EC2 by using the image on Docker Hub<br>• Worked on issues: #115, #116, #117, #140, #153<br>• Opened pull requests: #118, #130, #138, #148<br>• Wrote reviews to pull requests: #129, #130, #131, #132, #133, #134, #135, #136, #137, #162, #182<br>• Prepared the project plan in this report<br>• Wrote the individual report for Milestone 2<br>• Took meeting notes in Meeting 15 |

| Name | Work Done |
| --- | --- |
| *Zehranaz Canfes* | • Chosen an API as an external API (See FavQs)<br>• Researched APIs and RESTful APIS (Research links can be found in Weekly Efforts)<br>• Researched Django and DjangoREST (see Weekly Efforts)<br>• Researched how to build a Django app (see Weekly Efforts)<br>• Understood the FavQs API and thought of functionalities to implement<br>• Learned Docker and contributed to dockerizing the app<br>• Learned AWS and tried to deploy the app<br>• Struggled with errors and understanding the app<br>• Opened and worked on issues: #119, #121, #140, #149, #152, #154, #166<br>• Written unit tests and research how to write unit tests (see Weekly Efforts and Quotes API Tests)<br>• Opened pull requests #136, #164, and #182.<br>• Worked on and solved a bug that caused the tests not work after dockerizing the app<br>• Worked on improving the api and the app<br>• Wrote reviews to 6 other group members, more precisely to 11 pull requests: #163, #162, #148, #137, #135, #134, #133, #132, #131, #130<br>• Done research on how to write documentation to APIs (see Weekly Efforts)<br>• Implement the Quotes API<br>• Completed the Quotes API documentation<br>• Created the page and template for the API Documentations in wiki (here)<br>• Wrote Introduction, Project Status and Work Done So Far, Moving Forward and Conclusion part of Milestone 2 Report.<br>• Wrote Individual Report for Milestone 2<br>• Wrote the explanation of my API to the Deliverables Report<br>• Took meeting notes in Meeting 16<br>• Attended all meetings |

| Name | Work Done |
| --- | --- |
| *Mustafa Emir Çolak* | • Researched about REST APIs, Django and using thid party APIs.<br>• Using Tisane Parse API made an API on /api/storypost that analyzes the story it received from the user for abusive words. Then posts the story into the DB specifying with a boolean flag that if a mean word appeared.<br>• Writtern unit tests for the API /api/storypost.<br>• Created documentation for my api.<br>• Using the API at /api/storypost made the frontend for post creation at /api/create. Then, we combined it with the frontend for post viewing Mert, Kadir and Ahmet made.<br>• Attended all the group meetings and written the meeting report for meeting 17.<br>• Evaluated some of the tools we used in the project (Django, Postman, Swagger, MySQL, Whatsapp, Discord) for this report.<br>• Opened two pull requests, namely #133 and #159<br>• Reviewed pull requests, they are namely #129, #130, #131, #132, #134, #135, #136, #162.<br>• Reviewed the work Docker group has done. |
| *Ömer Barış Erkek* | • Learned about how HTTP protocol, RESTful APIs, Django, AWS, and Docker works<br>• Learned about Google Maps Geolocation API and its Nearby Search API method<br>• Learned about writing API documentations<br>• Implemented Nearby Places API functionality for practice-app<br>• Implemented unit tests for Nearby Places API functionality<br>• Written documentation for Nearby Places API functionality<br>• Attended Meetings 12, 13, 14, 15, 16, 17, 18, 19, and 20<br>• Took notes of Meeting 18 and published meeting report on wiki<br>• Created pull request #131, then fixed bugs in the code in accordance with my peers' comments<br>• Reviewed pull requests of other group members (#129, #130, #132, #133, #134, #135, #136, #137, #162) and tested their code<br>• Contributed to Dockerizing the app and deploying it<br>• Contributed to Group Report<br>• Contributed to Deliverables Report<br>• Written my individual report<br>• Created and managed issues #112, #151, #171, #173, #187, #188 |

| Name | Work Done |
| --- | --- |
| *Kadir Kalkan* | • Researched about Django, REST Framework, HTML, HTTP and Postman<br>• Learned about GeoDB Cities API<br>• Createded and managed issues: #120, #142, #143, #144, #156, #174, #175, #177, #191, #192<br>• Opened pull requests: #137, #141, #160, #176<br>• Wrote reviews to pull resquests and approved them: #129, #130, #131, #132, #133, #134, #135, #136, #148, #162, #182<br>• Attended Meetings 12, 13, 14, 15, 16, 17, 18, 19, and 20<br>• Implemented Nearby Places API functionality for practice-app<br>• Learned about Unit Tests and how to implement them on Django.<br>• Wrote tests for City API<br>• Took notes of Meeting 19 and published meeting report on wiki<br>• Implemented the frontend functionality of City API, Nearby Places API and Jokes API<br>• Wrote documentation for City API functionality<br>• Contributed to Group Report<br>• Contributed to Deliverables Report<br>• Prepared the individual report |
| *Melih Özcan* | • Researched about Django, REST Framework, Docker and AWS<br>• Implemented the Chuck Norris Jokes API<br>• Implemented the necessary test for the API<br>• Reviewed the frontend development<br>• Review the dockerization process<br>• Explained the API implementation and functionality<br>• Took the meeting notes of Meeting 20<br>• Attended Meetings 12, 13, 14, 15, 16, 17, 18,19, 20<br>• Prepared the individual report<br>• Contributed to the Deliverables Report<br>• Contributed to the Gruop Report<br>• Wrote the documentation for the Joke API<br>• Created the issues 123, 124, 150, 167, 208, 209, 210 and 211<br>• Created the pull requests 126 and 129 |

| Name | Work Done |
| --- | --- |
| *Ahmet İbrahim Şentürk* | • Learned about REST APIs, Django, HTML, HTTP, Postman.<br>• Searched APIs to use one. Decided on COVID-19 API.<br>• Learned about sending requests to a third party API.<br>• Implemented COVID-19 API for our practice app. Issue#111<br>• Learned about Unit Tests and how to implement them on Django.<br>• Wrote tests for my functionality.<br>• Created the documentation of my API. Issue#207<br>• Implemented the frontend functionality of COVID-19 API, Location API. Issue#155<br>• Opened pull request for my API and my part of frontend functionalities. (#134,#139,#161)<br>• Reviewed pull requests. (#129,#130,#131,#134,#135,#136,#137,#162)<br>• Reviewed other team members frontend implementations.<br>• Learned Docker and reviewed my teammates progressions.<br>• Wrote about my API to Deliverables Report.<br>• Wrote API URLs and their brief explanations for Milestone 2 Group Report. Issue#206<br>• Wrote individual part of Group Milestone Report 2. Issue#203<br>• Wrote individual Milestone Report 2. Issue#204 |
| *Niyazi Ülke* | • Researched REST APIs, HTTP and Postman.<br>• Researched Django framework.<br>• Researched Docker and AWS.<br>• Chose IBM Watson Language Translator API as a third party API. Issue #110<br>• Learned the functionalities of IBM Watson Language Translator API.<br>• Implemented Translation API for the project. Issue #110<br>• Wrote tests for my API methods. Issue #128<br>• Wrote documentation for my API. Issue #185<br>• Created and managed issues: #110, #125, #128, #140, #146, #163, #165, #170<br>• Opened pull requests #135, #163<br>• Attended all meetings.<br>• Took the meeting notes at Meeting 13.<br>• I took part in dockerizing the application. Issue #146<br>• Wrote reviews for pull requests : #129, #130, #132, #133, #134, #136, #137, #148, #162, #164<br>• Evaluated "AWS", "Docker", "Git", "GitHub", "VSCode" tools for this report. Issue #195<br>• Evaluated "Team Meetings", "TA Meetings", "Issue Creation", "Code Review" processes for this report. Issue #195 |

# 7. Conclusion

Although the implementation of the practice app was hard and challenging for us, we learned a lot of new tools, software and improved ourselves by applying our theoretical knowledge. Moreover, we had fun during the process and enjoyed having an app as a product at the end. Now, after this assignment, we can say with confidence that we are ready for the next step of our project. We are ready to start implementing and working on our main project next semester in CmpE 451 and we look forward to working together.