NCC Campus Event Manager

Final Report of CNG 495 Term Project

Fall-2023

Team Members:

Berke Diler (2401503)

Keremcan Adanur (2452902)

Gökberk Lük (2453405)

January 9, 2024

1 Introduction

The NCC Event Manager is a comprehensive system designed to streamline and optimize the management of campus events. It offers a user-friendly interface for both event organizers and attendees who are the residents of the METU NCC Campus.

This project stands out in its ability to become a solution for the chaos of events in our campus. Offering various aspects of event management into a single platform where users can see the events that are upcoming to reserve their spot and events that happened. The system's innovative approach lies in its integration with cloud services, ensuring scalability and reliability.

Comparable systems in the market often lack the scalability and customization options (such as user authentication) that NCC Event Manager provides. Our project fills this gap, offering a tailored solution to meet the specific needs of a university setting.

An Admin account was created so that the admin functionality can be tried:

1. **E-mail:** *berke.diler@metu.edu.tr*
2. **Password:** *5lr1pqL8A0VX*

The project's GitHub repository can be cloned using this link:
https://github.com/bercats/ncc$_{c}ampus_{e}vent_{m}anager.git$

And without any cloning, the project can be tested from this link:
https://codesandbox.io/p/sandbox/github/bercats/ncc$_{c}ampus_{e}vent_{m}anager/tree/master/$

2 Structure of the Project

The NCC Event Manager is divided into several key components:

- **Event Management:** Allows event organizers to create and manage event schedules with ease.
- **Event Scheduling:** Offers the functionality see event capacity, poster, and other credientials and shows the events in the sorted schedule.
- **Authentication-Security:** Ensures that the attendees are only using the university domain to register.

Below is a table showing which cloud services are used and their purposes in the project.

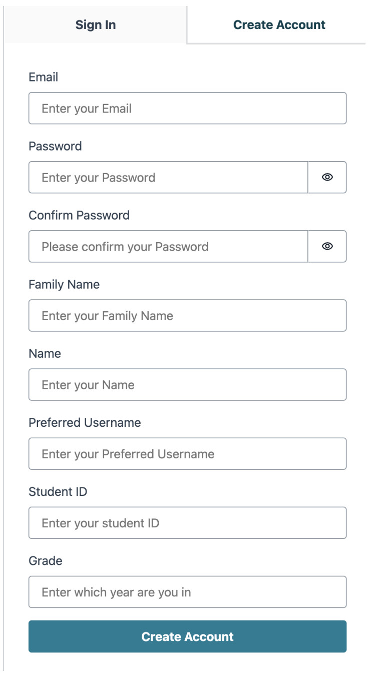
AWS Service	Usage in Project
Cognito	Managed user authentication and security. Also stored information about the user.
S3	Stored static resources and data files.
Lambda	Executed back-end functions. (Example: Form validation)
Amplify	Simplified deployment and management of the web app.
DynamoDB	NoSQL database for scaleable applications.

Table 1: AWS Services Utilized in the NCC Event Manager Project

2.1 User Manual

2.1.1 Registration and Login (Berke Diler)

In order to register to the application, user needs to fill the following form on Figure 1, the form is processed using **AWS Cognito**, and checked if the user is using a *@metu.edu.tr* domain with **AWS Lambda** if not, an error is returned. The registration information is stored in the **AWS Cognito** user pool.

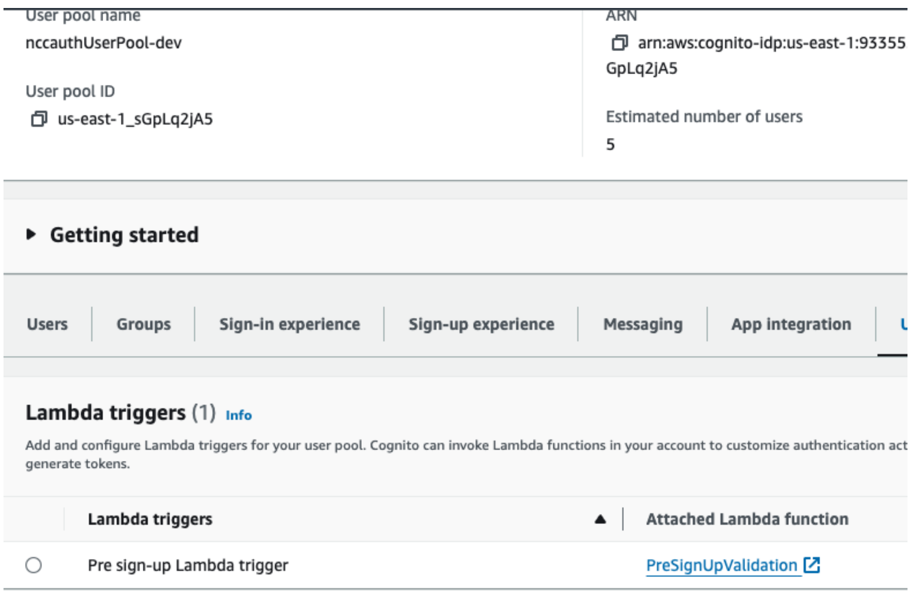

Sign In
Create Account

Email
Enter your Email
Password
Enter your Password
Confirm Password
Please confirm your Password
Family Name
Enter your Family Name
Name
Enter your Name
Preferred Username
Enter your Preferred Username
Student ID
Enter your student ID
Grade
Enter which year are you in

Create Account

Figure 1: Registration

The form validation's implementation using **AWS Lambda** to check *@metu.edu.tr* domain was fairly straightforward but the researching done to understand the process (since the web resources are not clear for our implementation) I had difficulties implementing it and I crashed the **AWS Amplify** project. Then created an another, and migrated all our functionalities to the new project. **This part of the project can not be seen as a commit on GitHub because it was done inside AWS and not inside the codebase.** The AWS Lambda pre-signup trigger can be seen in Figure 2 and the Lambda function can be seen in Figure 3, node.js was used for the source code. And a diagram showing the data flow between the application, AWS Cognito, and AWS Lambda can be seen in 4 and gathered from [1] .



User pool name
nccauthUserPool-dev

User pool ID
us-east-1_sGpLq2jA5

ARN
arn:aws:cognito-idp:us-east-1:93355
GpLq2jA5

Estimated number of users
5

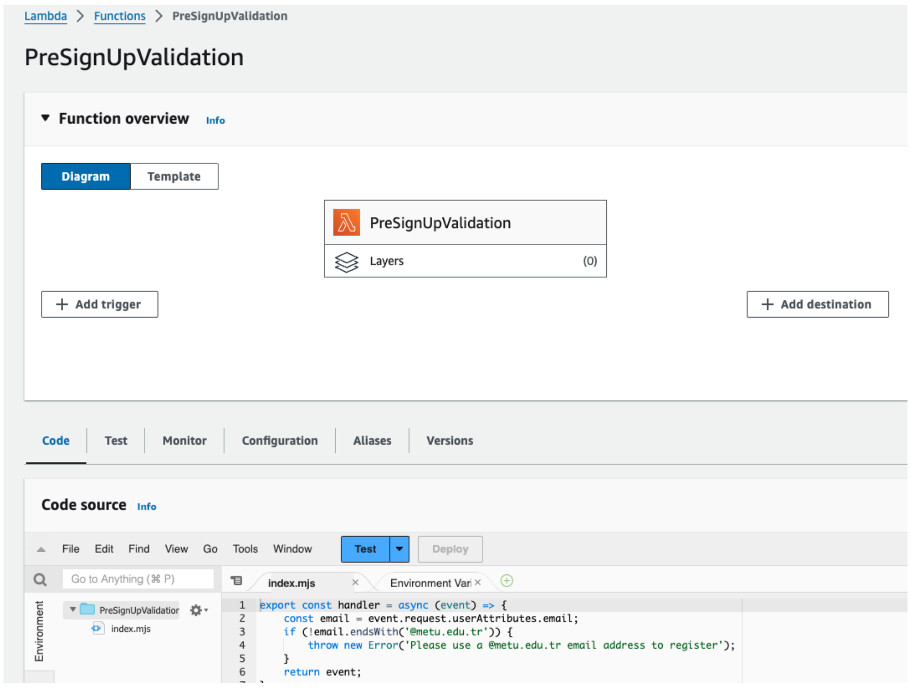
Lambda triggers (1) Info

Add and configure Lambda triggers for your user pool. Cognito can invoke Lambda functions in your account to customize authentication act
generate tokens.

Lambda triggers	Attached Lambda function
Pre sign-up Lambda trigger	PreSignUpValidation

Figure 2: AWS Cognito pre-signup Lambda Trigger

Once the user successfully registers to the application, he needs to login from the login screen in order to be directed to the home page on Figure 5. The user interfaces for registration and login screens are from the **AWS Amplify** React.js UI Library, registration screen has configurations on it so that we can get information such as Student ID, Name, Family Name, Grade(which year the student is in), and preferred username. And these information are stored in the user pool under **AWS Cognito** which can be seen in Figures 6 and 7.



```
export const handler = async (event) => {
    const email = event.request.userAttributes.email;
    if (!email.endsWith('@metu.edu.tr')) {
        throw new Error('Please use a @metu.edu.tr email address to register');
    }
    return event;
```

Figure 3: AWS Lambda Function

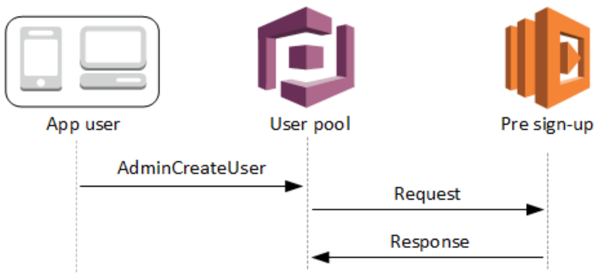


Figure 4: Relationship between AWS and Application

2.1.2 Home Page UI (Berke Diler)

If the user is an admin, they will see the ADMIN and Create Event button in the home page, if they are a regular user, they won't see the button, and this differentiation is done by checking the user's pool group from **AWS Cognito**, the admins are in a user pool called **admins**. Figure 8 is a screenshot from the home page as an admin user. In the home page, user has the ability to navigate through different tabs and sign out. By navigating through tabs, the user is able to see the upcoming events, past events and more. For the Home Page, I have used javascript, HTML, and CSS. This part was especially challenging since I did not have any prior knowledge about web development.

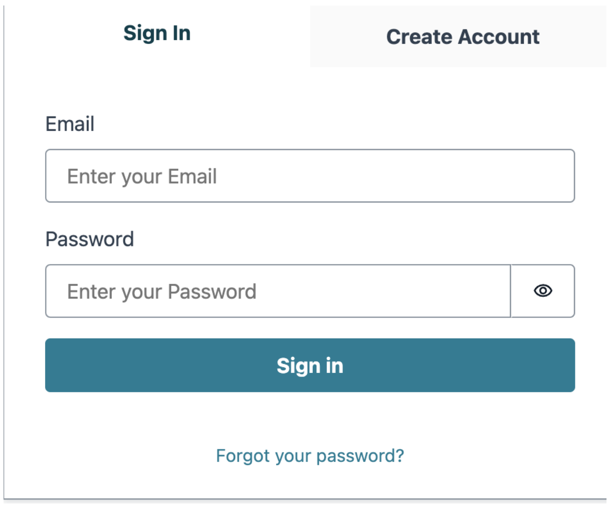
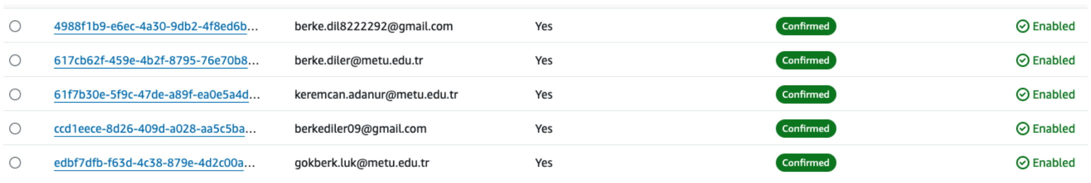
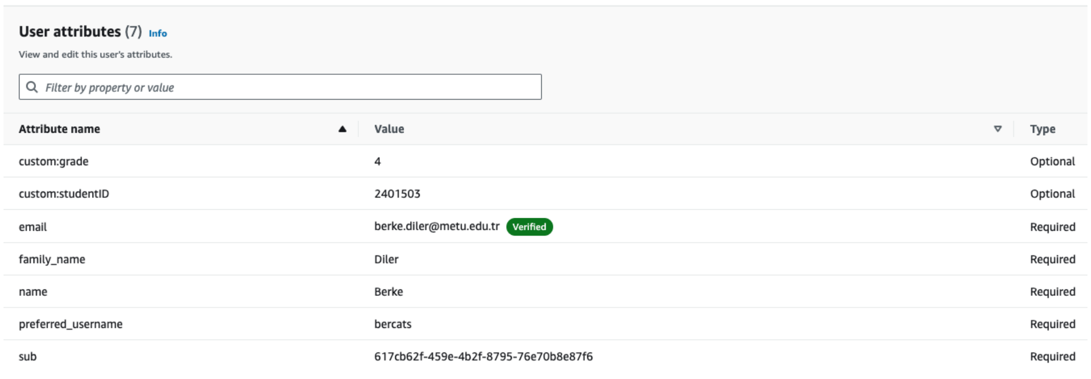


Figure 5: Login



○	4988f1b9-e6ec-4a30-9db2-4f8ed6b…	berke.dil8222292@gmail.com	Yes	Confirmed	✓ Enabled
○	617cb62f-459e-4b2f-8795-76e70b8…	berke.diler@metu.edu.tr	Yes	Confirmed	✓ Enabled
○	61f7b30e-5f9c-47de-a89f-ea0e5a4d…	keremcan.adanur@metu.edu.tr	Yes	Confirmed	✓ Enabled
○	ccd1eece-8d26-409d-a028-aa5c5ba…	berkediler09@gmail.com	Yes	Confirmed	✓ Enabled
○	edbf7dfb-f63d-4c38-879e-4d2c00a…	gokberk.luk@metu.edu.tr	Yes	Confirmed	✓ Enabled

Figure 6: AWS Cognito User Pool



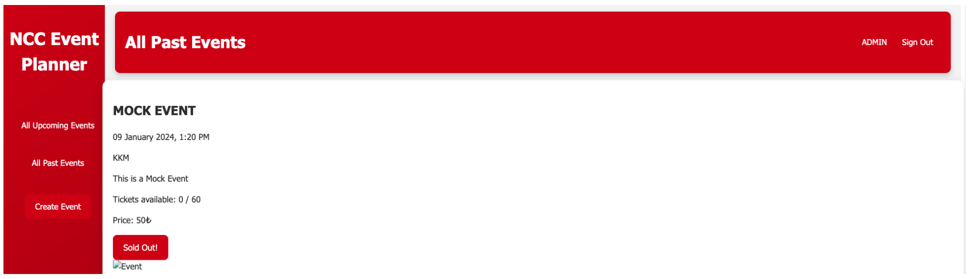
User attributes (7) Info

View and edit this user's attributes.

Filter by property or value

Attribute name	Value	Type
custom:grade	4	Optional
custom:studentID	2401503	Optional
email	berke.diler@metu.edu.tr Verified	Required
family_name	Diler	Required
name	Berke	Required
preferred_username	bercats	Required
sub	617cb62f-459e-4b2f-8795-76e70b8e87f6	Required

Figure 7: AWS Cognito User Details


NCC Event Planner

All Upcoming Events

All Past Events

Create Event

All Past Events

ADMIN Sign Out

MOCK EVENT

09 January 2024, 1:20 PM

KKM

This is a Mock Event

Tickets available: 0 / 60

Price: 50₺

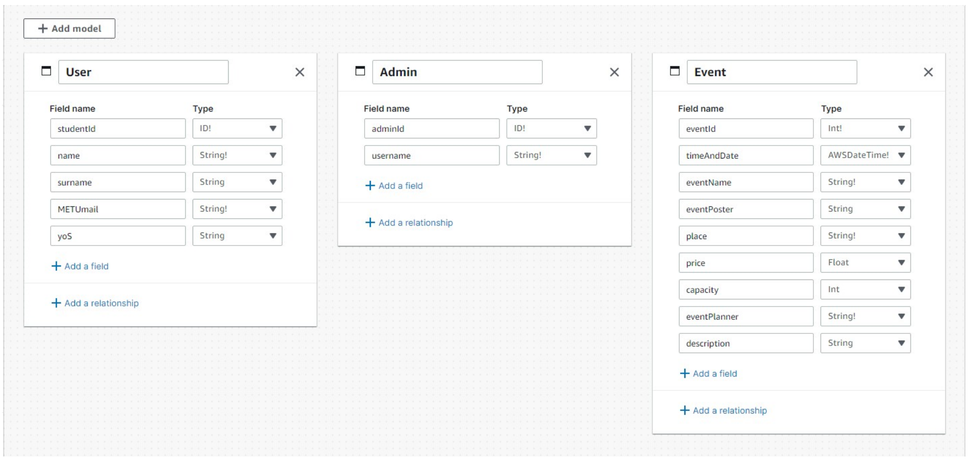
Sold Out!

Event

Figure 8: Home Page

2.1.3 Database (Keremcan Adanur)

In our project we use DynamoDB provided by AWS as our database which is a NoSQL database. Once provided with the graphql schema, Amplify handles the creation of most of the boilerplate code. All the basic functionalities are added. In our project, the graphql schema and corresponding table structure is shown in Figure 9. If the admin user wants to have a backup of the current state of the database, they can do it through the AWS console. The bucket is preconfigured so it is enough to just click ’export to S3’ in Figure 10. While Creating the database I accidentally forgot to make the make the Admin tables firt letter capital, which took reconfiguration of the backend by Gökberk to solve which took off from the development time.



User

Field name	Type
studentId	ID!
name	String!
surname	String
METUmail	String!
yoS	String

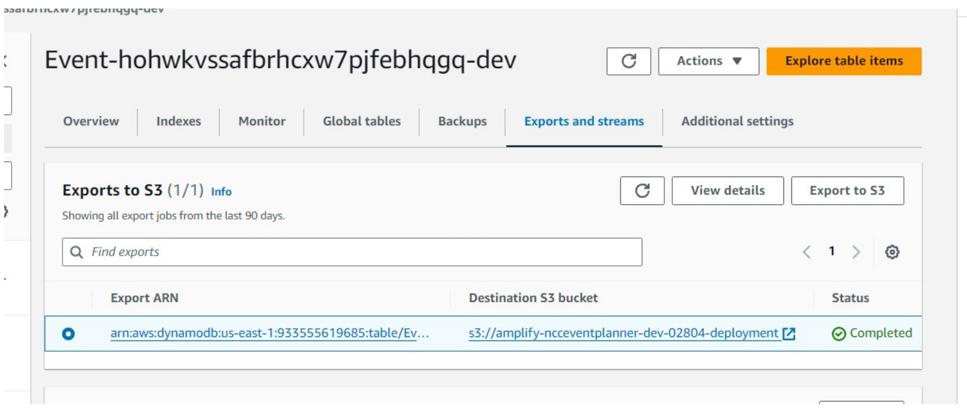
Admin

Field name	Type
adminId	ID!
username	String!

Event

Field name	Type
eventId	Int!
timeAndDate	AWSDateTime!
eventName	String!
eventPoster	String
place	String!
price	Float
capacity	Int
eventPlanner	String!
description	String

Figure 9: Tables in the Database


Event-hohwkvssafbrhcxw7pjfebhqgq-dev

Exports to S3 (1/1) Info

Showing all export jobs from the last 90 days.

Export ARN	Destination S3 bucket	Status
arn:aws:dynamodb:us-east-1:933555619685:table/Ev...	s3://amplify-nccenteventplanner-dev-02804-deployment	Completed

Figure 10: Exporting to S3 to create a backup

2.1.4 Event Cards (Keremcan Adanur)

When both student users and admins enter the website they have the ability to see all the events that are in the system. To show these events in the main page, each event is structured inside their own event card. These cards are shown according to their dates in the main page. Each card consists of name of the event, date, place, description, price and possible tickets available for that event. If the event organisers have their posters, then it is also shown in the event card. These event cards can be seen in Figure 12. Values in these event cards are pulled from DynamoDB through the Amplify API. To render images in events which has a poster, we need to use the url of that image to retrieve it from the S3 bucket it resides in, these operations are shown in Figure 11.

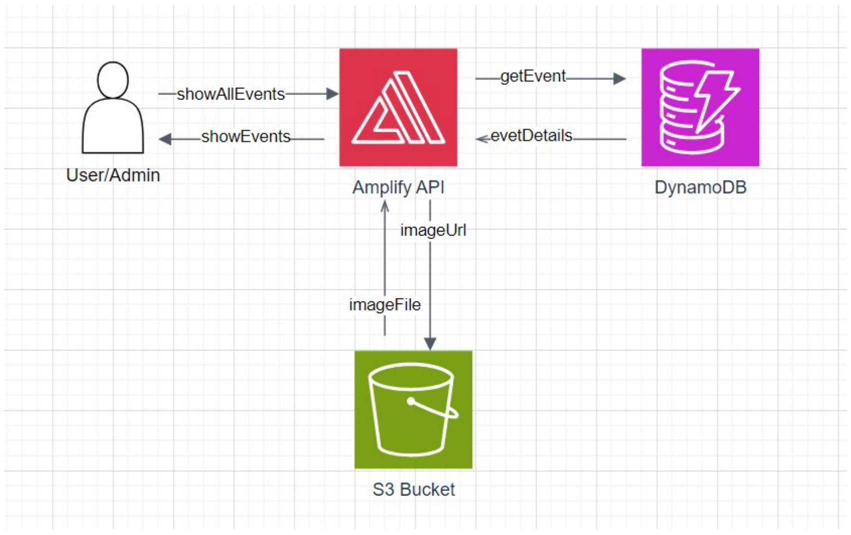
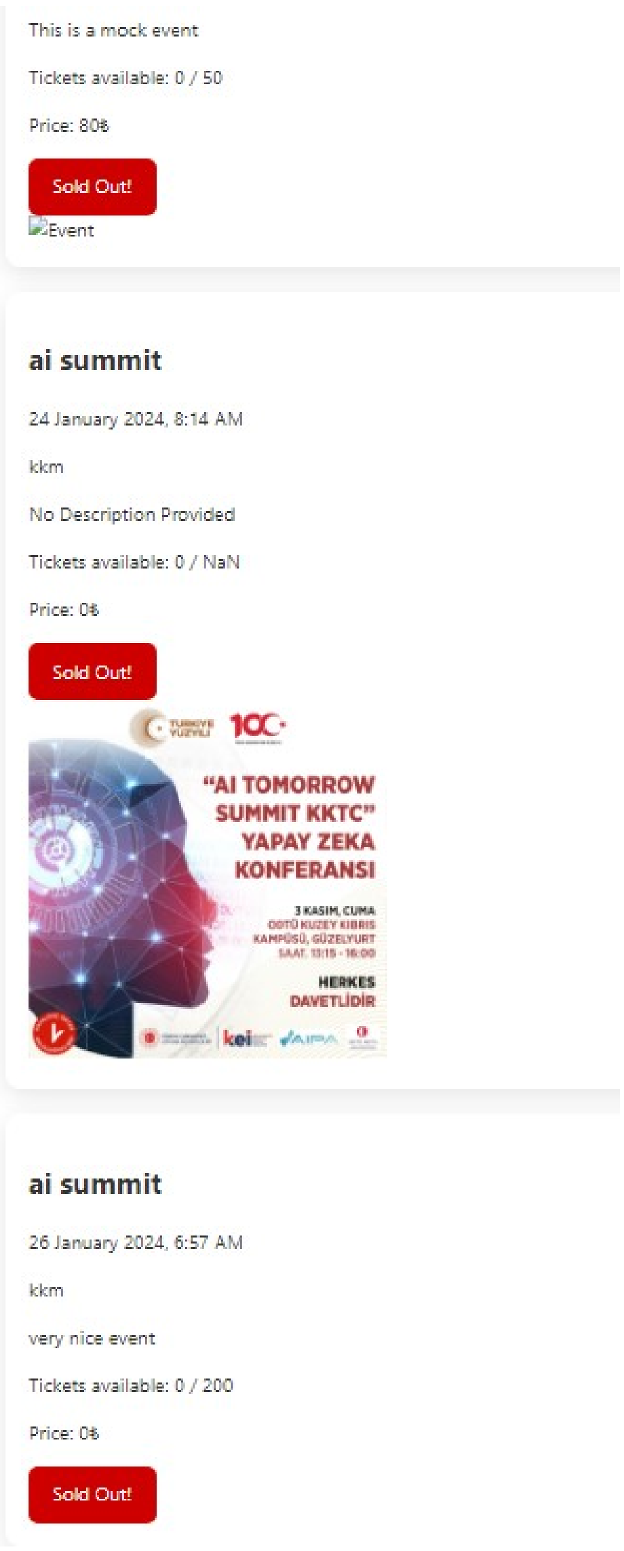


Figure 11: Flow of operations when one events are shown to the user

.


This is a mock event

Tickets available: 0 / 50

Price: 80₺

Sold Out!

Event

ai summit

24 January 2024, 8:14 AM

kkm

No Description Provided

Tickets available: 0 / NaN

Price: 0₺

Sold Out!

"AI TOMORROW SUMMIT KKTC" YAPAY ZEKA KONFERANSI
3 KASIM, CUMA
ODTÜ KUZEY KIBRIS KAMPÜSÜ, GÜZELYURT
SAAT 13:15 - 16:00
HERKES DAVETLİDİR

ai summit

26 January 2024, 6:57 AM

kkm

very nice event

Tickets available: 0 / 200

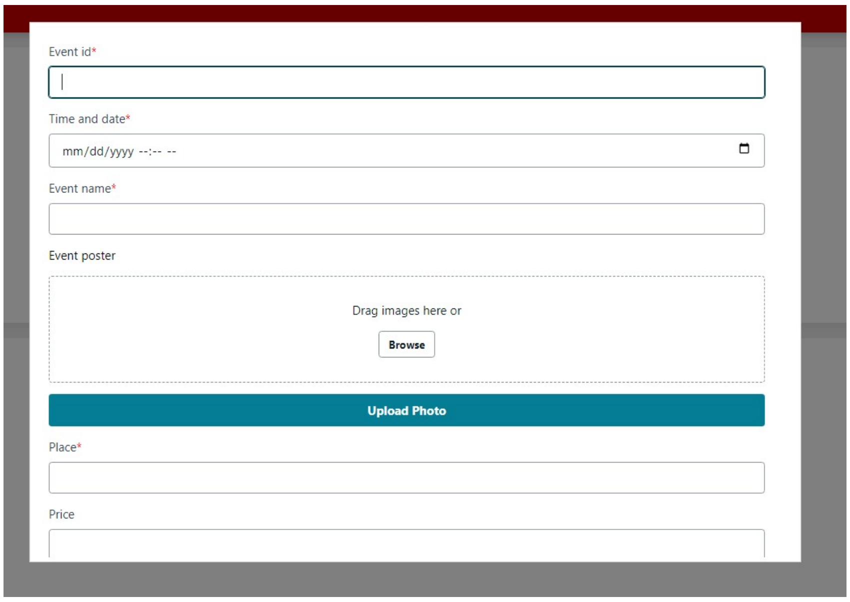
Price: 0₺

Sold Out!

Figure 12: Event cards with and without posters

2.1.5 Event Create Form (Keremcan Adanur)

To create these events, only admin users have a button in the side panel, which opens up the form in Figure 13. in this form admins can add all the necessary information regarding the event, including its posters. There is a separate button for poster upload, this must be clicked before form submission if the event want to be shown with the posters. This general template was automatically generated by Amplify but modified to include image upload function. The image is then stored in S3 bucket and its key and url is sent back to the application, the url of the object in S3 is stored inside eventPoster attribute of Event Table. The operations done between the systems can be seen in Figure 14. One problem I faced while modifying this form was to comply to the standards of Amplify UI and adding components from that library. Learned that connecting new components to a auto-generated form is lot harder than creating it from scratch, but it allowed us to have consistency between form elements in our project.


Event id*

Time and date*

mm/dd/yyyy --:-- --

Event name*

Event poster

Drag images here or

Place*

Price

Figure 13: Event Create Form

.

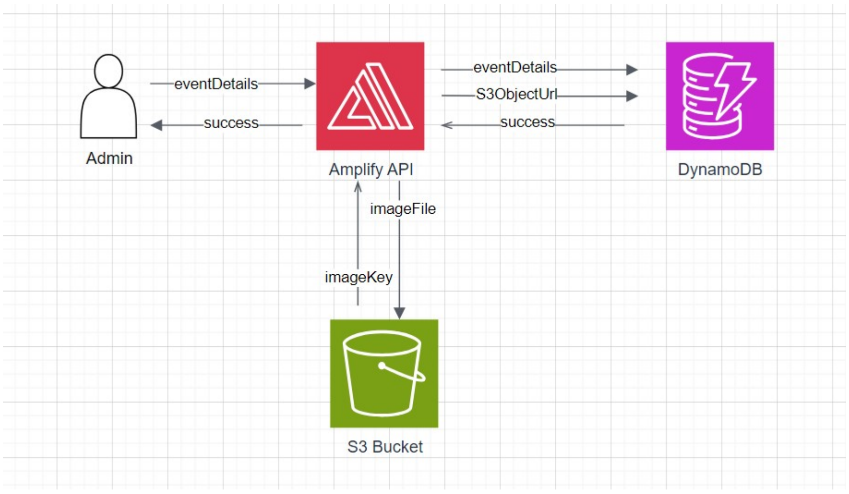
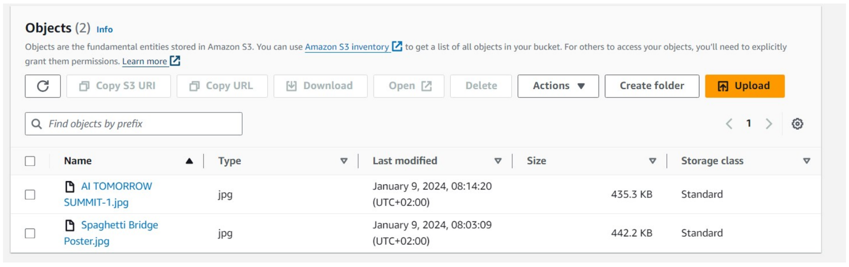


Figure 14: Flow of operations when adding a new event

2.1.6 Image Upload & Retrieval (Keremcan Adanur)

As explained in the above section upload and retrieval of the images was done through S3 buckets. This was one of the important steps of this porject where we wanted the application to be more appealing to users and event organisers by allowing them to show their own advertisements for the event in the website. Since most events create posters to hang outside the dorms, cafeteria etc. It would be beneficial for them to use these posters in the online environment as well. Each image is stored as a separate object inside the S3 bucket and retrieved whenever the event is required to be shown. Example of an image inside the S3 bucket can be seen in Figure 15. Connecting was not problematic but we were having access problems, which we solved by making the bucket for images public, since these posters do not contain private information, and are hanged around buildings in real life for whole school to see this should not cause problems for the organisers.


Objects (2) Info

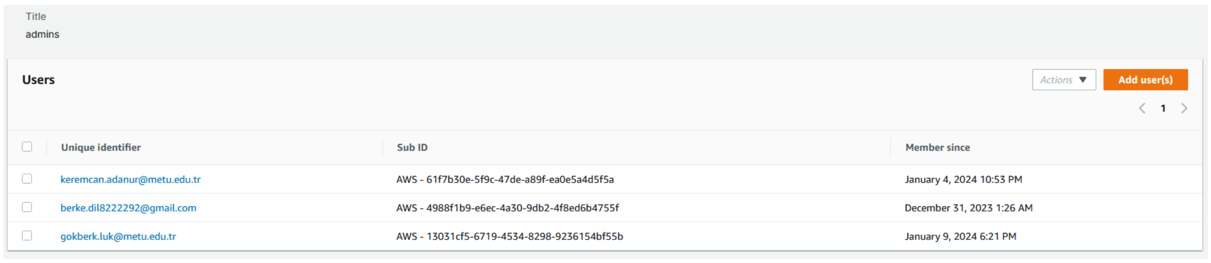
Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

Name	Type	Last modified	Size	Storage class
AI TOMORROW SUMMIT-1.jpg	jpg	January 9, 2024, 08:14:20 (UTC+02:00)	435.3 KB	Standard
Spaghetti Bridge Poster.jpg	jpg	January 9, 2024, 08:03:09 (UTC+02:00)	442.2 KB	Standard

Figure 15: Images stored in S3 bucket as s3 objects

2.1.7 Admin User(Gökberk Lük)

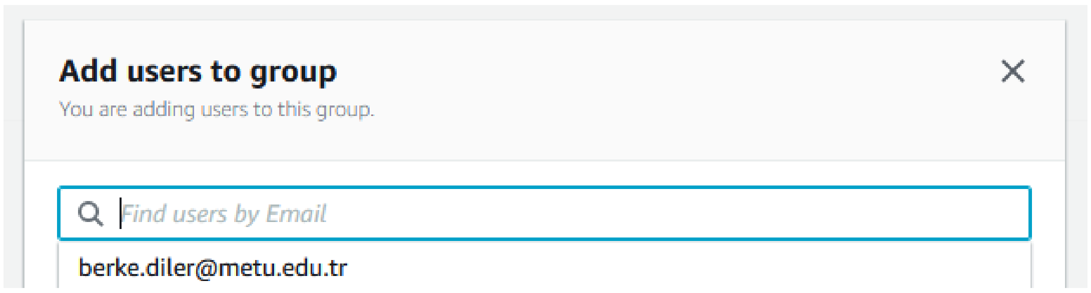
The panel provided to us by **AWS Amplify Studio** also serves as an admin panel. Therefore, here we can see the information of the users or perform admin operations or configure forms. As seen in Figure 16, one can choose one of the existing users in the system and be given the admin role.


Title
admins

Users

Unique identifier	Sub ID	Member since
keremcan.adanur@metu.edu.tr	AWS - 61f7b30e-5f9c-47de-a89f-ea0e5a4d5f5a	January 4, 2024 10:53 PM
berke.dil8222292@gmail.com	AWS - 4988f1b9-e6ec-4a30-9db2-4f8ed6b4755f	December 31, 2023 1:26 AM
gokberk.luk@metu.edu.tr	AWS - 13031cf5-6719-4534-8298-9236154bf55b	January 9, 2024 6:21 PM

Figure 16: AWS Amplify User Panel


Add users to group

You are adding users to this group.

berke.diler@metu.edu.tr

Figure 17: Admin Selection

As seen in Figure 18, the user assigned as admin on the UI side is indicated as **Admin** in the upper right corner when he enters the system.

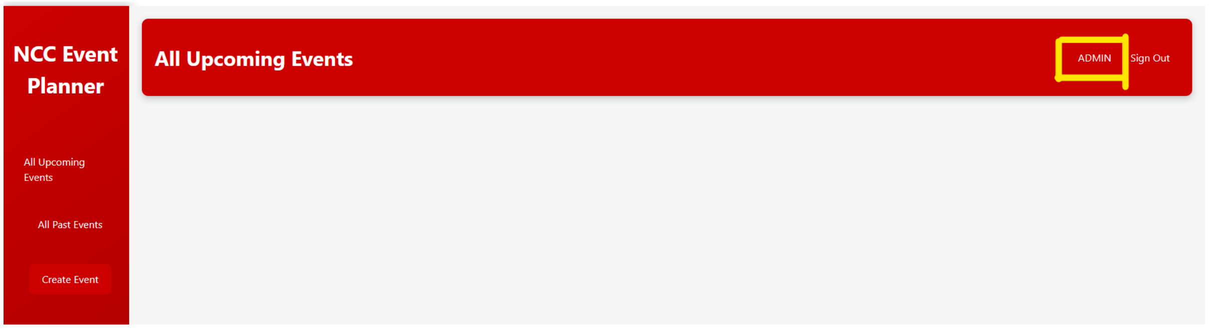
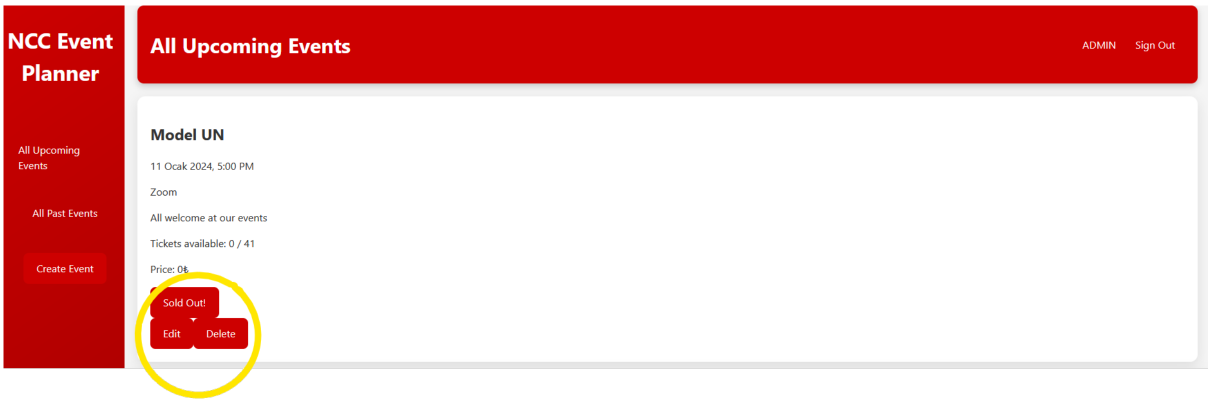


Figure 18: UI Side of Admin

2.1.8 Event Update Form (Gökberk Lük)

Events that exist in the system and appear on the UI screen must be edited by the admin. That's why the Event Edit Form is included. If the logged-in user is an admin, he can see this section. Admin users assigned through **AWS Amplify Studio** will be able to access these sections. As seen in Figure 19, you can see who is an admin in **AWS Amplify Studio**.



Model UN

11 Ocak 2024, 5:00 PM

Zoom

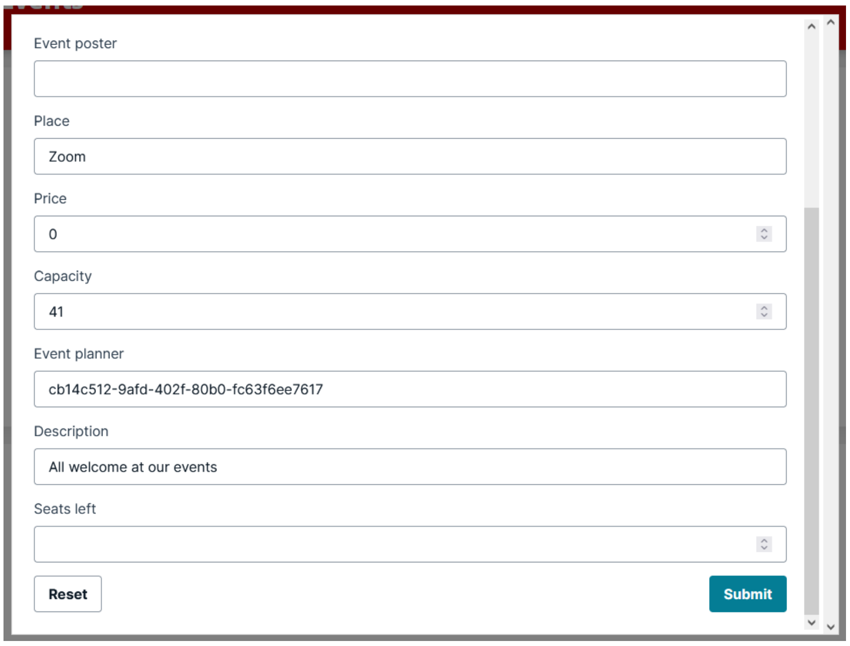
All welcome at our events

Tickets available: 0 / 41

Price: 0₺

Figure 19: Admin Panel

The event update form can be seen in Figure 20. In this way, the admin could easily update the event.

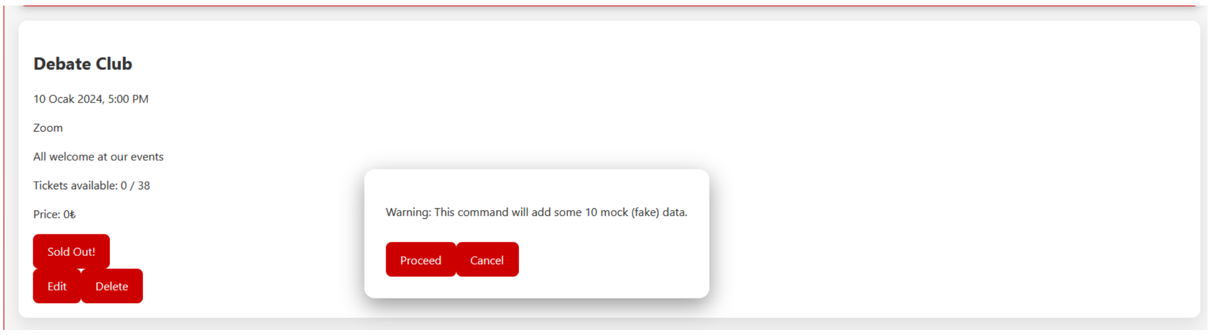


Event poster	
Place	Zoom
Price	0
Capacity	41
Event planner	cb14c512-9afd-402f-80b0-fc63f6ee7617
Description	All welcome at our events
Seats left	

Figure 20: Event Update Form

2.1.9 Adding Mock Data (Gökberk Lük)

When you click on the **Admin** text to add mock data in the admin panel on the UI side, a popup appears. As seen in Figure 21, Admin can add 10 mock data from the pop up. After you approve this process, the newly added data appears on the upcoming event page.



Debate Club

10 Ocak 2024, 5:00 PM

Zoom

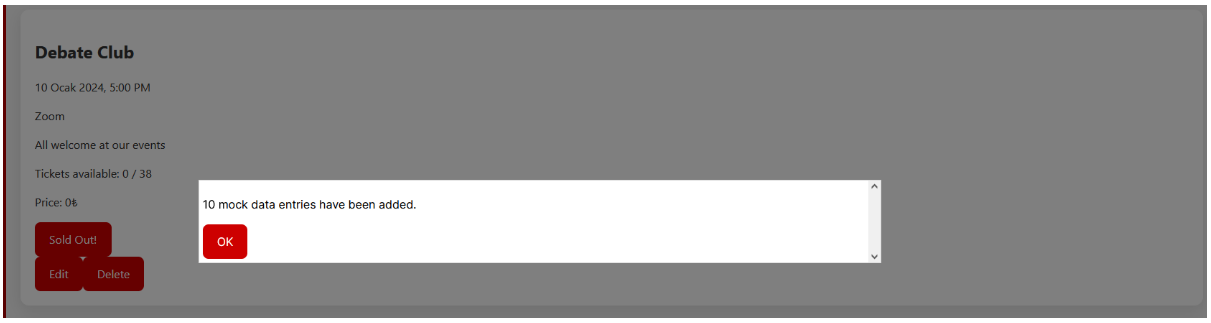
All welcome at our events

Tickets available: 0 / 38

Price: 0₺

Warning: This command will add some 10 mock (fake) data.

Figure 21: Admin Mock Data Panel



Debate Club

10 Ocak 2024, 5:00 PM

Zoom

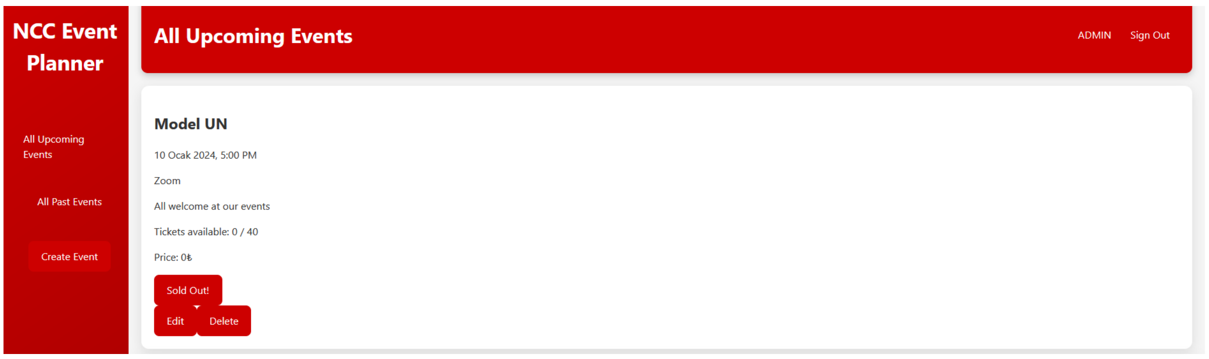
All welcome at our events

Tickets available: 0 / 38

Price: 0₺

10 mock data entries have been added.

Figure 22: Pop Up for Mocking



Model UN

10 Ocak 2024, 5:00 PM

Zoom

All welcome at our events

Tickets available: 0 / 40

Price: 0₺

Figure 23: New Mock Data

As seen in Figure 23, there is newly added mock data. The number of these 10 mock data can be changed if desired. There are currently more than 100 pieces of data in the system. Expired events can be seen in past events. Also, As seen in Figure 22, a pop-up appears informing the admin that mock data has been added. Also, As seen in Figure 22, a pop-up appears informing the admin that mock data has been added.

.

3 Project Statistics

The project was developed over several months, with each team member contributing to specific aspects. Key statistics include:

1. **Development Time:** 3 months.
2. **Team Contributions:** Berke focused on frontend development and Authentication, Keremcan on backend integration, and Gökberk on database management and admin panel.
3. **Codebase:** 4638062 lines of code (calculted using cloc, *https://github.com/AlDanial/cloc.git*), predominantly in JavaScript. The output of running cloc in the codebase can be seen in Figure 24. **The statistics can be disinformative since most of these lines of codes are coming from AWS resources.**
4. **Database Limitations:** There is no limit on number of rows a table can hold. Number of tables are restricted by region to 2500 which our app is not close to exceeding. Size of every item must be smaller than 400kb but our tables are small, we are not concerned with this limitation.
5. **Berke Diler:** From these statistics, approximately 400 lines of code were written for Authentication with AWS Cognito and for AWS Lambda, and the homepage UI by Berke Diler using Javascript, HTML and CSS. Which approximately took 1,5 month of work.
6. **Gökberk Lük:** From these statistics, approximately 250 lines of code were written for Admin panel, and the homepage UI by Gökberk Lük using Javascript and HTML. Which approximately took 1.5 month of work.
7. **Keremcan Adanur:** Approximately 400-450 lines of code is written for GraphQL Schema, Event Cards, modifications on Event Creation Form, image retriaval and upload. Many elements that Amplify provides are utilized. These all took around 1.5 month of development time.

4 Videos

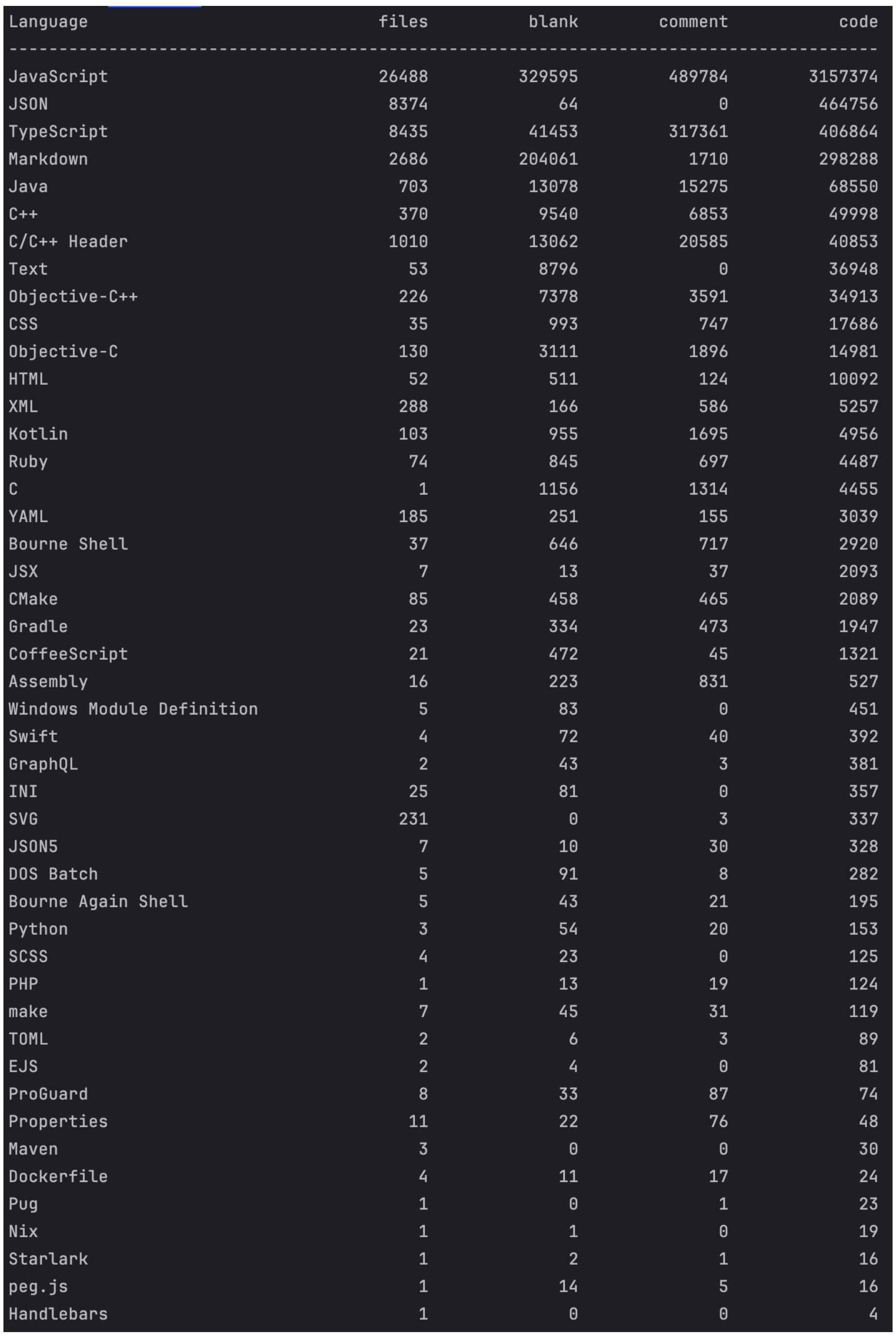
Videos of each member can be reached from the following list.

1. **Berke Diler:** https://youtu.be/iODWasYnFVo
2. **Gökberk Lük:** https://youtu.be/K3Owgx$_3 - Dg$
3. **Keremcan Adanur:** https://youtu.be/4OOxLR42IPM

References

[1] Pre sign-up Lambda trigger - Amazon Cognito *https://docs.aws.amazon.com/cognito/latest/developerguide/user-pool-lambda-pre-sign-up.html*. Amazon, N.d.

[2] Create, update, and delete application data. Amplify Documentation - AWS Amplify *https://docs.amplify.aws/react/build-a-backend/graphqlapi/mutate-data/* Amazon, N.d.



```
Language                             files          blank        comment           code
---------------------------------------------------------------------------------------
JavaScript                           26488         329595         489784        3157374
JSON                                  8374             64              0         464756
TypeScript                            8435          41453         317361         406864
Markdown                              2686         204061           1710         298288
Java                                   703          13078          15275          68550
C++                                    370           9540           6853          49998
C/C++ Header                          1010          13062          20585          40853
Text                                    53           8796              0          36948
Objective-C++                          226           7378           3591          34913
CSS                                     35            993            747          17686
Objective-C                            130           3111           1896          14981
HTML                                    52            511            124          10092
XML                                    288            166            586           5257
Kotlin                                 103            955           1695           4956
Ruby                                    74            845            697           4487
C                                        1           1156           1314           4455
YAML                                   185            251            155           3039
Bourne Shell                            37            646            717           2920
JSX                                      7             13             37           2093
CMake                                   85            458            465           2089
Gradle                                  23            334            473           1947
CoffeeScript                            21            472             45           1321
Assembly                                16            223            831            527
Windows Module Definition                5             83              0            451
Swift                                    4             72             40            392
GraphQL                                  2             43              3            381
INI                                     25             81              0            357
SVG                                    231              0              3            337
JSON5                                    7             10             30            328
DOS Batch                                5             91              8            282
Bourne Again Shell                       5             43             21            195
Python                                   3             54             20            153
SCSS                                     4             23              0            125
PHP                                      1             13             19            124
make                                     7             45             31            119
TOML                                     2              6              3             89
EJS                                      2              4              0             81
ProGuard                                 8             33             87             74
Properties                              11             22             76             48
Maven                                    3              0              0             30
Dockerfile                               4             11             17             24
Pug                                      1              0              1             23
Nix                                      1              1              0             19
Starlark                                 1              2              1             16
peg.js                                   1             14              5             16
Handlebars                               1              0              0              4
```

Figure 24: Cloc output