



ORTA DOĞU TEKNİK ÜNİVERSİTESİ
MIDDLE EAST TECHNICAL UNIVERSITY
KUZEY KIBRIS KAMPUSU ♦ NORTHERN CYPRUS CAMPUS

CNG 495

CLOUD COMPUTING

FALL 2024 – Progress Report

Team Members	
Ekrem Cagatay Goz	2526374
Haya Arabi Katibi	2542520
Engin Eray Kabalak	2526424

Table of Contents

Table of Figures	3
Milestones Achieved.....	4
Week 7 : November 11 – November 17	4
Week 8 : November 18 – November 24	4
Week 9 : November 25 – December 1	4
Week 10 : December 2 – December 8	4
User Interface.....	5
Login and Signup	5
Dashboard (Home).....	6
Society Page.....	6
Header	7
Tutorials for AWS Services	7
AWS Relational Database Service (RDS)	7
AWS Amplify	9
AWS IAM.....	11
AWS S3	12
GitHub Repositories:	14
Client Side:.....	14
Server Side:	14
Milestones Remained.....	14
Week 11 : December 9 – December 15	14
Week 12 : December 16 – December 22	14
Week 13 : December 23 – December 29	14

Table of Figures

Figure 1 Login Page	5
Figure 2 Signup Page	5
Figure 3 Dashboard	6
Figure 4 Society Page	6
Figure 5 RDS Engine options	7
Figure 6 RDS Settings	7
Figure 7 RDS Access Options	8
Figure 8 RDS link	8
Figure 9 Amplify Deploying Options	9
Figure 10 Amplify Repository Selection	9
Figure 11 Amplify Build Settings	10
Figure 12 Amplify Review	10
Figure 13 IAM Access Management	11
Figure 14 IAM Administration Access	11
Figure 15 S3 URL	12

Milestones Achieved

Week 7 : November 11 – November 17

All members:

- Finalized the project idea and drafted the initial requirements.
- Created AWS account

Week 8 : November 18 – November 24

Ekrem Çağatay Göz:

- Set up the GitHub repository with directories for client and server.

Engin Eray Kabalak:

- Learning Spring Boot

Haya Arabi Katibi:

- Learned how to use AWS Amplify

Week 9 : November 25 – December 1

Haya Arabi Katibi:

- Developed Login and Signup frontend
- Developed Dashboard frontend
- Developed Society Page frontend

Ekrem Çağatay Göz:

- Learned how to use AWS RDS
- Developed login system at the server side
- Developed signup system at the server side
- Implemented JWT system at the server side

Engin Eray Kabalak:

- Learned how to use AWS S3 and AWS IAM
- Created groups on IAM
- Implemented core APIs for Societies and Announcements

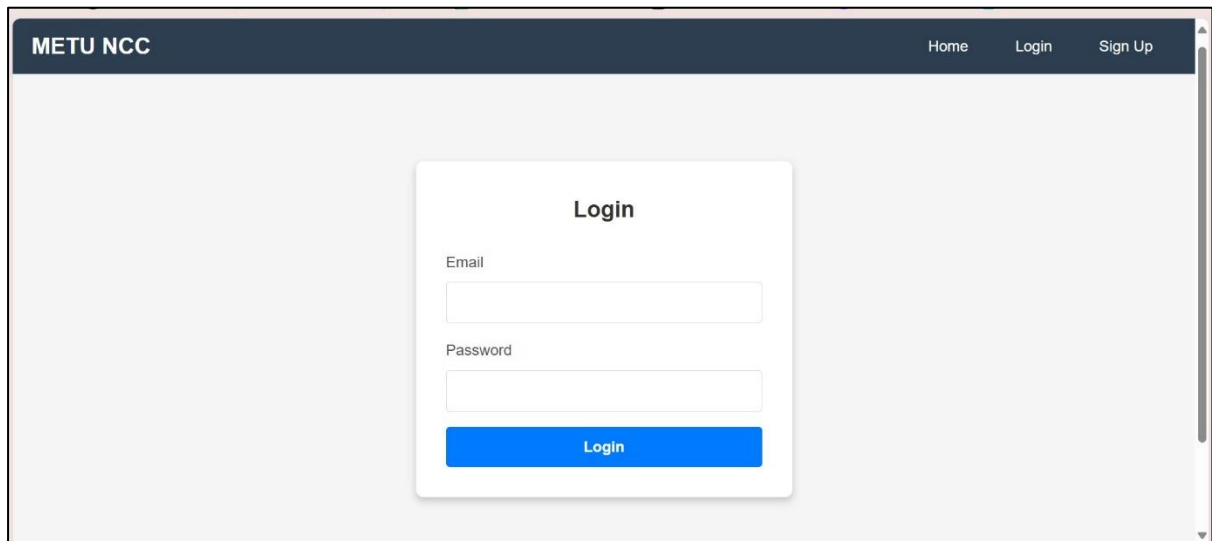
Week 10 : December 2 – December 8

1) All Team:

- Prepared report

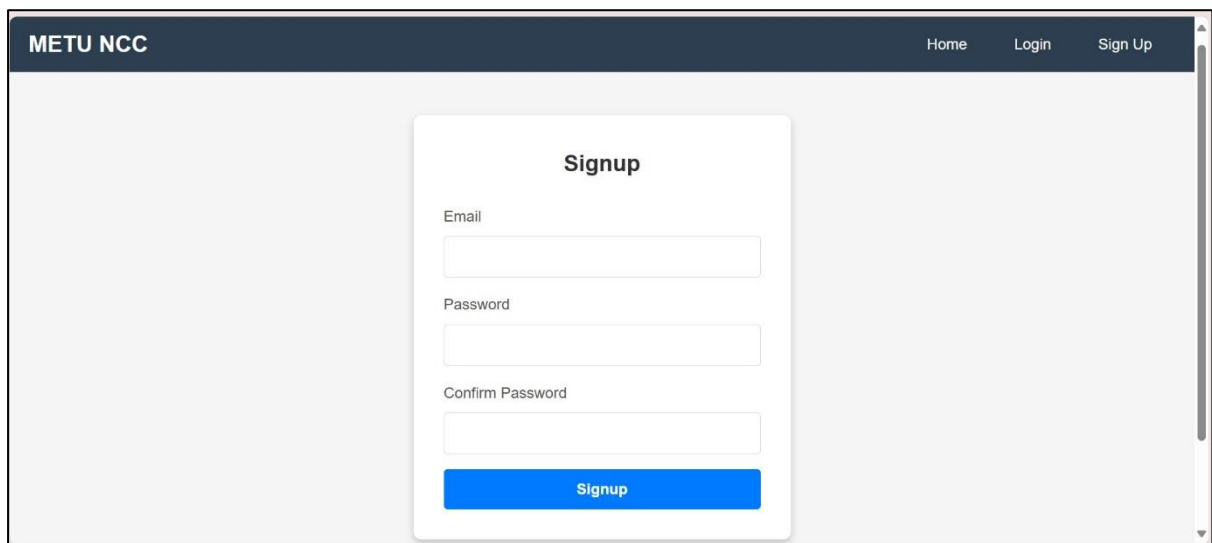
User Interface

Login and Signup



The image shows a web browser window with a dark blue header bar. On the left of the header is the text "METU NCC". On the right are three links: "Home", "Login", and "Sign Up". The main content area has a light gray background. In the center is a white rounded rectangle with a shadow. At the top of this rectangle is the title "Login". Below the title are two input fields: the first is labeled "Email" and the second is labeled "Password". Below these fields is a blue button with the text "Login" in white. A vertical scrollbar is visible on the right side of the browser window.

Figure 1 Login Page



The image shows a web browser window with a dark blue header bar. On the left of the header is the text "METU NCC". On the right are three links: "Home", "Login", and "Sign Up". The main content area has a light gray background. In the center is a white rounded rectangle with a shadow. At the top of this rectangle is the title "Signup". Below the title are three input fields: the first is labeled "Email", the second is labeled "Password", and the third is labeled "Confirm Password". Below these fields is a blue button with the text "Signup" in white. A vertical scrollbar is visible on the right side of the browser window.

Figure 2 Signup Page

Dashboard (Home)

The dashboard serves as a main place for exploring all societies. It shows a list of the available societies as seen in the figure. By selecting a society from the list, users are redirected to the corresponding Society Page.

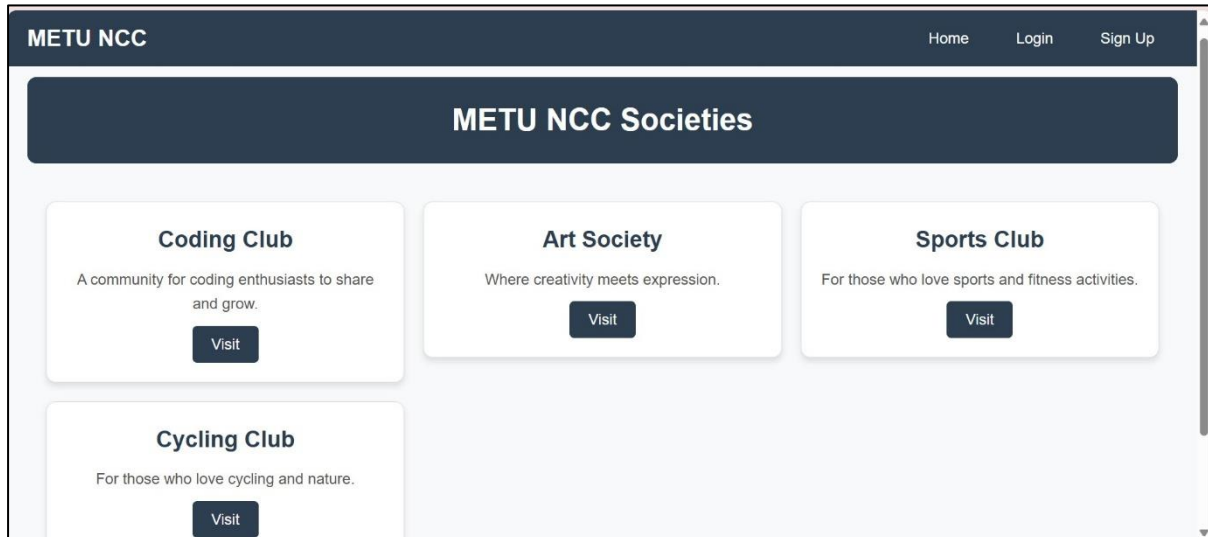


Figure 3 Dashboard

Society Page

The Society Page provides detailed information about a specific society. Including their name, description and announcements with a poster.

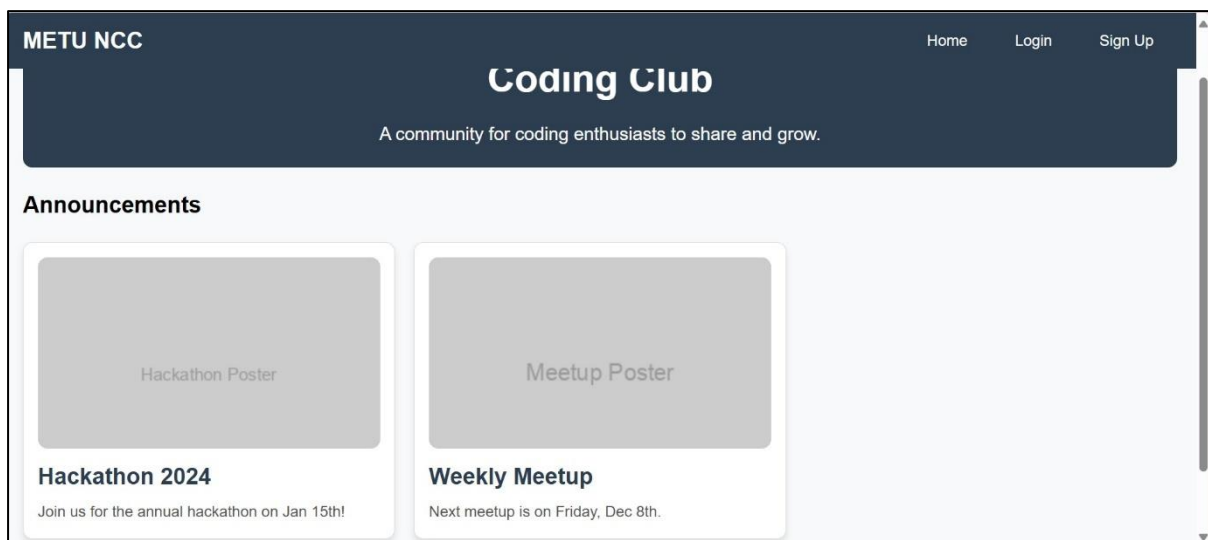


Figure 4 Society Page

Header

From the header the user can navigate to the main pages: Login, Signup and Home page. It is included on all pages to provide consistent navigation.

Tutorials for AWS Services

AWS Relational Database Service (RDS)

This service of AWS is for creating relational databases to store our tables. It supports multiple database management systems, such as MySQL, PostgreSQL etc. Also, it provides different storage options for different needs. Since different options have different costs, we used the options which are selected with 'Free Tier'.

To be able to use this service, we need to enter the RDS page from AWS console and then, use 'Create Database' button to create new database. After that, it redirects to database creation page. This page has lots of options. Our main options are like following images:



Figure 5 RDS Engine options

Since we are familiar to PostgreSQL interface, we wanted to use it.

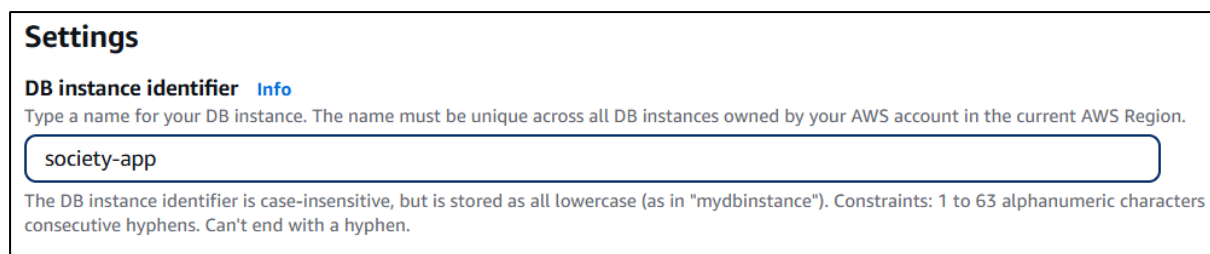
The image shows the 'Settings' section of the AWS RDS console. It includes a 'DB instance identifier' field with the value 'society-app' entered. Below the field, there is a text box explaining the constraints for the identifier: 'The DB instance identifier is case-insensitive, but is stored as all lowercase (as in "mydbinstance"). Constraints: 1 to 63 alphanumeric characters and consecutive hyphens. Can't end with a hyphen.'

Figure 6 RDS Settings

We adjusted the instance name in AWS, and then PostgreSQL authentication password.

Public access [Info](#)

☒ **Yes**
RDS assigns a public IP address to the instance.
Choose one or more VPC security groups to allow connections to the instance from any AWS resource in the VPC.

☐ **No**
RDS doesn't assign a public IP address to the instance.
Choose one or more VPC security groups to allow connections to the instance from resources in the VPC.

Figure 7 RDS Access Options

We selected 'Public access' as Yes because we wanted to try our back-end from our local computers.

After creating our database, we adjusted security settings to access from 5432 port to our database instance.

At the end of this process, AWS gave a public link and then we pasted this link to our project side like the following image:

```
spring.datasource.url=jdbc:postgresql://AWS-LINK:5432/societyapp
spring.datasource.username=postgres
spring.datasource.password=1234
```

Figure 8 RDS link

Then, our back-end side could save user data into deployed database server.

AWS Amplify

This service of AWS is designed to simplify the development of full-stack web and mobile applications. It provides tools to quickly set up backend services like authentication, APIs, and storage, as well as hosting for frontend applications. Amplify supports integration with popular frameworks such as React and Angular. To manage costs, we utilized the free-tier options available during development.

In order to use this service, we need to enter the amplify page from AWS console. Then chose to deploy an app. Then we are redirected to a page that lists different options to deploy. In our case we chose GitHub repository:

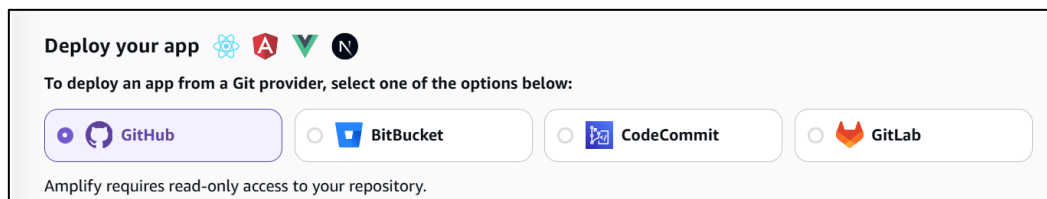


Figure 9 Amplify Deploying Options

Next step is to give permission to Amplify to fetch the available repositories. Once authorized, we selected the repository containing the client-side code.

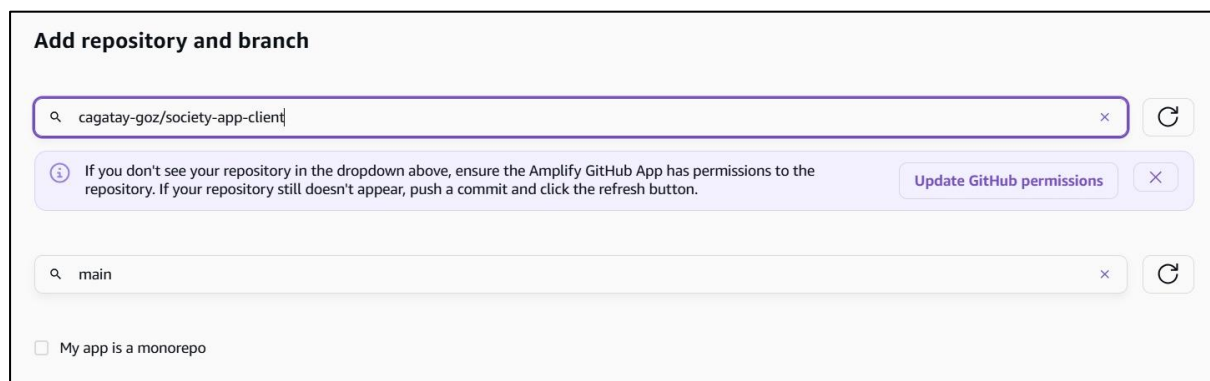
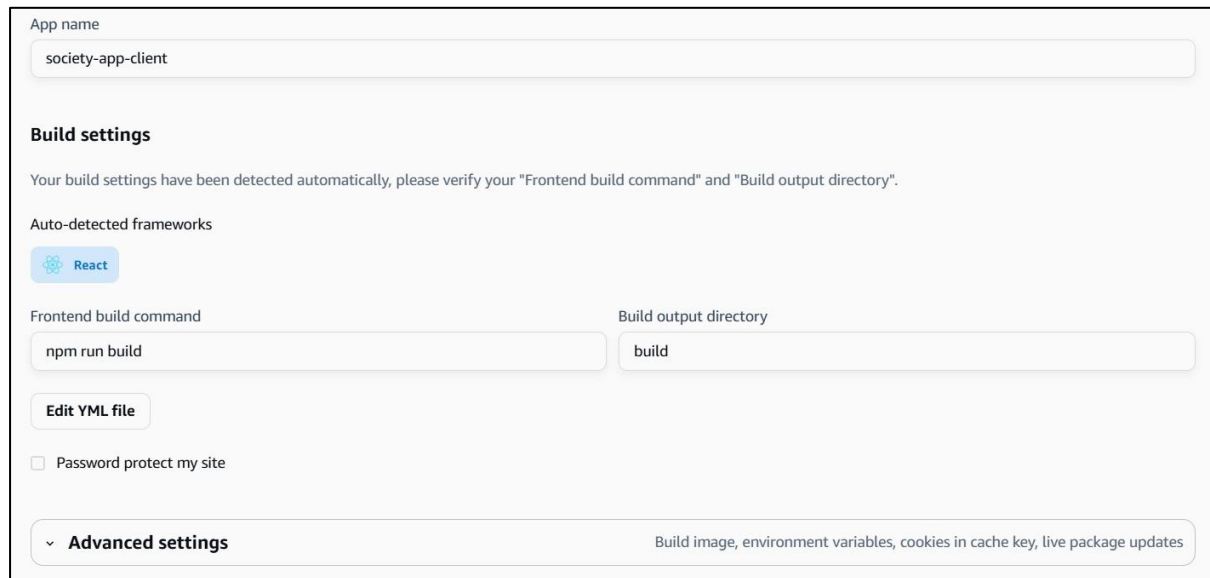


Figure 10 Amplify Repository Selection

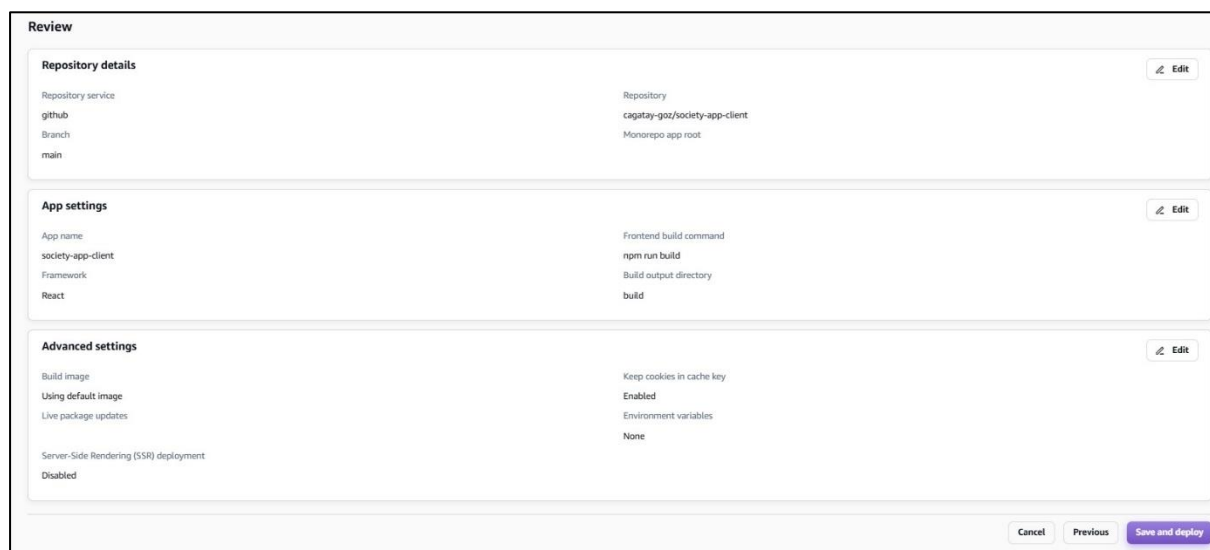
After fetching the repository, we set the applications name as society-app-client and set the frontend build command and build output directory and press next.



The screenshot shows the 'Build settings' screen in the Amplify CLI. At the top, the 'App name' is set to 'society-app-client'. Below this, a message states: 'Your build settings have been detected automatically, please verify your "Frontend build command" and "Build output directory"'. Under 'Auto-detected frameworks', 'React' is selected. The 'Frontend build command' is 'npm run build' and the 'Build output directory' is 'build'. There is an 'Edit YML file' button and a checkbox for 'Password protect my site' which is unchecked. At the bottom, there is an 'Advanced settings' section with a dropdown arrow and a link to 'Build image, environment variables, cookies in cache key, live package updates'.

Figure 11 Amplify Build Settings

Finally, we rechecked our settings to save and deploy. Currently, we have not deployed the application yet, as the cost would increase significantly. However, we plan to deploy it by the end of the semester once the application is finalized.



The screenshot shows the 'Review' screen in the Amplify CLI. It contains three sections: 'Repository details', 'App settings', and 'Advanced settings'. Each section has an 'Edit' button. The 'Repository details' section shows 'Repository service' as 'github', 'Repository' as 'cagatay-goz/society-app-client', and 'Branch' as 'main'. The 'App settings' section shows 'App name' as 'society-app-client', 'Framework' as 'React', 'Frontend build command' as 'npm run build', and 'Build output directory' as 'build'. The 'Advanced settings' section shows 'Build image' as 'Using default image', 'Keep cookies in cache key' as 'Enabled', 'Live package updates' as 'Enabled', 'Environment variables' as 'None', and 'Server-Side Rendering (SSR) deployment' as 'Disabled'. At the bottom, there are three buttons: 'Cancel', 'Previous', and 'Save and deploy'.

Figure 12 Amplify Review

AWS IAM

IAM allows you to control who has what kind of access to resources. For example, a user can only view or modify a specific folder in S3. However, all three of us will have access to everything since we granted admin access. We believe this will improve coordination within the group, as whenever one of us uses a service, the other group members will also be aware of it.

We created a group on Identity Access Management.

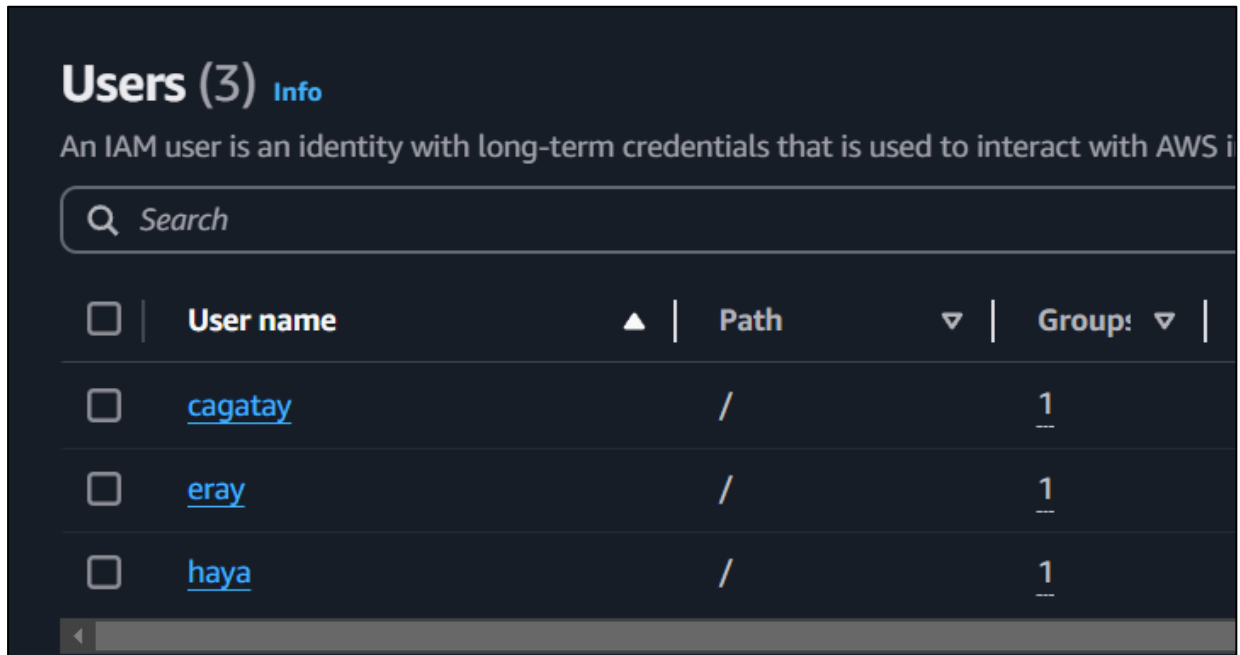


Figure 13 IAM Access Management

Each group member has Administrator access.

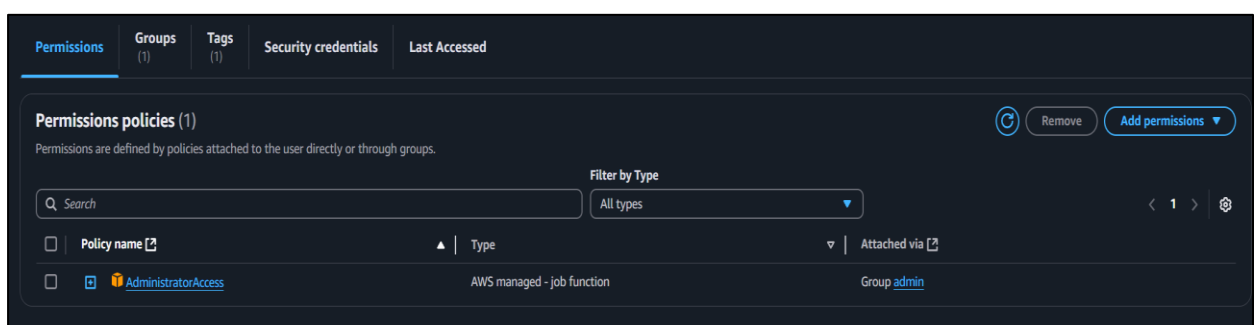


Figure 14 IAM Administration Access

AWS S3

AWS S3 (Simple Storage Service) is a reliable and flexible storage service that makes it easy to store and retrieve data, like images, videos, and backups, from anywhere. It's also commonly used for hosting static websites. The service integrates well with other AWS tools, making it a practical choice for various projects. To keep costs low during development, we took advantage of the free-tier options.

We will use Amazon S3 service in our project. The following figure shows the usage of poster URL.

```
@Entity
@Table(name="announcements")
@Data
public class Announcement {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    Long id;
    String title;
    String content;
    String posterUrl;

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "society_id", nullable = false)
    @OnDelete(action = OnDeleteAction.CASCADE)
    @JsonIgnore
    Society society;
}
```

Figure 15 S3 URL

How It Works

1. Storing images in amazon S3:
 - Images for announcements (posters) will be uploaded to Amazon S3.
 - After uploading the images, S3 generates a unique URL that can be used to access the file.
2. Database design:
 - In the Announcement class, instead of saving the image itself, the URL pointing to the image in S3 will be stored in the database.
3. Api to fetch announcements:
 - A backend API endpoint will retrieve announcements for a specific society from the database. Each announcement will include its details, such as the title, description, and the posterUrl for its associated image.
4. Client request to fetch announcements:

- The frontend will request announcements for a specific society through the API. Then, the backend will respond with all announcement data, including the posterUrl for each image.
5. Rendering images on the client side:
- The frontend can use the posterUrl directly in an tag to display the image fetched from S3.

GitHub Repositories:

Client Side:

<https://github.com/cagatay-goz/society-app-client.git>

In this repository, we pushed our front-end side. For front-end, we used React.js technology with JavaScript since its implementation is easier than pure HTML and CSS.

Server Side:

<https://github.com/cagatay-goz/society-app-api.git>

This repository consists of our back-end side. For this side, we used Spring Boot technology with Java language since its deployment duration is faster than the other frameworks.

Milestones Remained

Week 11 : December 9 – December 15

- Developing the admins panel
- Developing the reservation page
- Developing the roles system on the server side
- Developing reservation system on the server side

Week 12 : December 16 – December 22

- Integrating back-end with front-end
- Testing on our locals

Week 13 : December 23 – December 29

- Debugging for possible bugs
- Deploying front-end and back-end using AWS