MIDDLE EAST TECHNICAL UNIVERSITY
NORTHERN CYPRUS CAMPUS

Computer Engineering Program

CNG 495 - Cloud Computing
2024-2025 Fall

Term Project Progress Report II
“SoccerMatch Scheduler”

Team Members

Elif Ilgın Savaş - 2453512

Alper Kutay Ören - 2526556

Yiğit Berk Atcı - 2526101

Table of Content

User Interface

- Registration Page

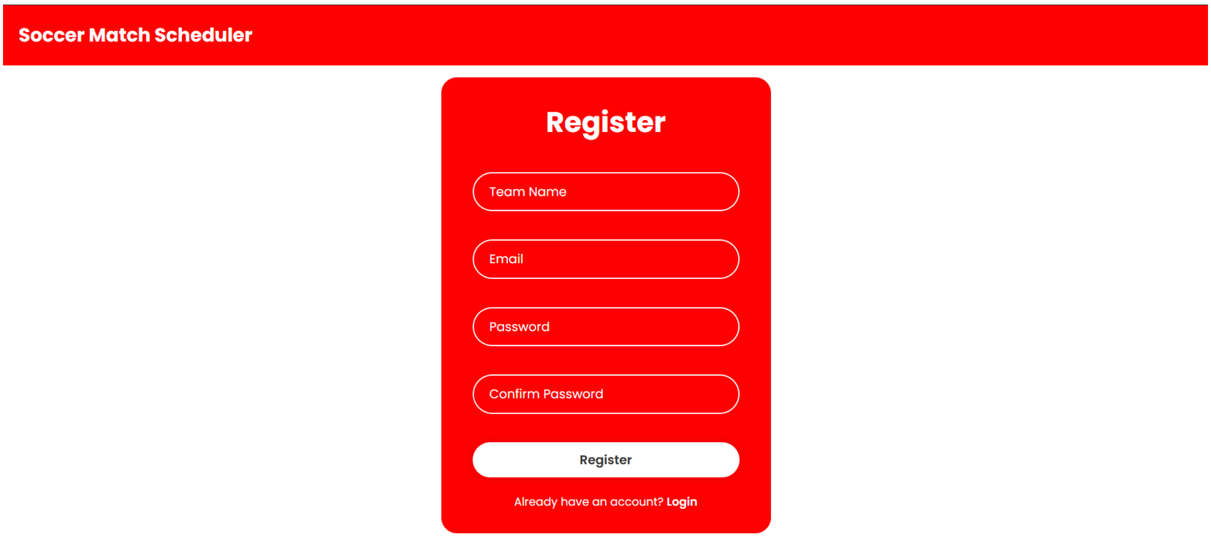


Figure 1: Register Page

Registration page takes the related inputs from the user. It includes team name, email, and password. There is a confirmation box to double-check for password. In addition, there is an option for already registered users to go into the login page.

- Login Page

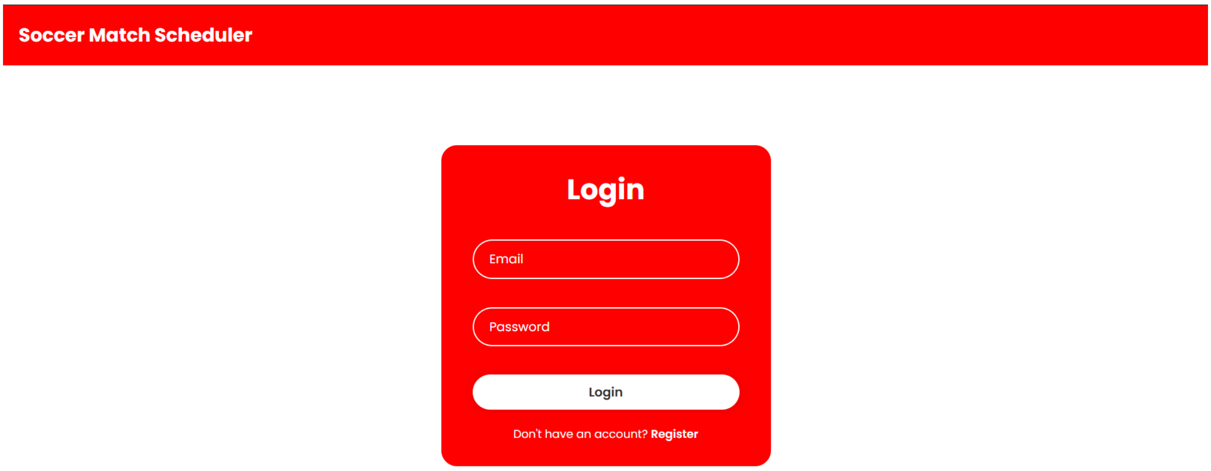
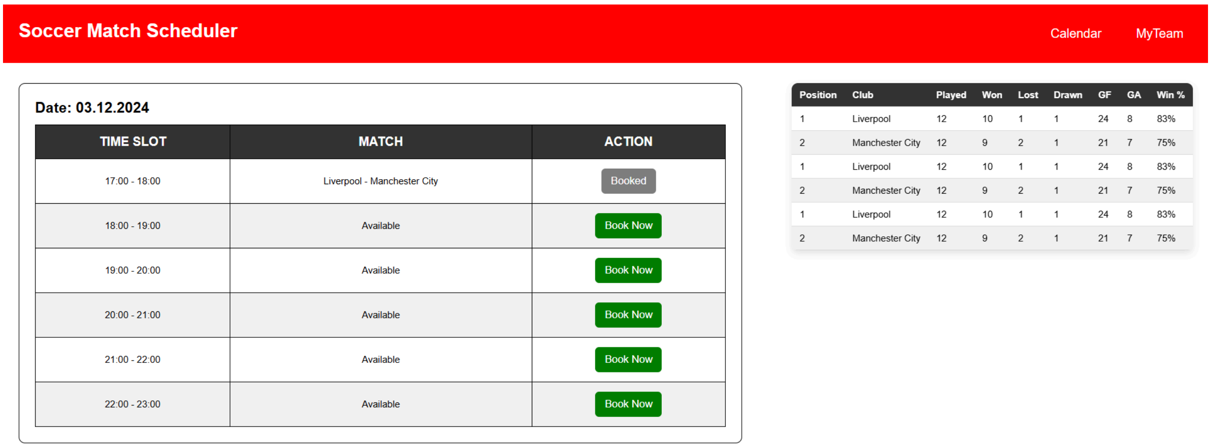


Figure 2: Login Page

Login page asks the users to enter their email and password. For those who do not have an account, there is an option to register in the system.

- Main Page



Soccer Match Scheduler

Date: 03.12.2024

TIME SLOT	MATCH	ACTION
17:00 - 18:00	Liverpool - Manchester City	Booked
18:00 - 19:00	Available	Book Now
19:00 - 20:00	Available	Book Now
20:00 - 21:00	Available	Book Now
21:00 - 22:00	Available	Book Now
22:00 - 23:00	Available	Book Now

Position	Club	Played	Won	Lost	Drawn	GF	GA	Win %
1	Liverpool	12	10	1	1	24	8	83%
2	Manchester City	12	9	2	1	21	7	75%
1	Liverpool	12	10	1	1	24	8	83%
2	Manchester City	12	9	2	1	21	7	75%
1	Liverpool	12	10	1	1	24	8	83%
2	Manchester City	12	9	2	1	21	7	75%

Figure 3: Main Page

Main page's header has a navigation bar which contains Calender and MyTeam items. Calendar page will show each day and time slots. Users will be able to modify their team information in the MyTeam page. There is a daily schedule on the left side of the main page. Users can see the available time slots for the current day, and they can send a book request. The Book Now button will send the user to another screen where they can send a request. In addition, users can see the other registered teams statistics on the right side table. Teams will be ranked in this table according to their winning rate.

Milestones Achieved

October 21 - October 25

Elif Ilgın Savaş

In the first week as a group, we have designed how our system should work, what pages we should have, should we create an account for teams or each user, etc. Afterwards, we have decided on the tables to be implemented for the database for our vision of design. Even though Figure x was reverse engineered from the created database, I have implemented the design by hand in the first week. The implementation of the DB was developed in the x week. At the same time the design of pages was being implemented and the research of how the instances are supposed to be implemented and deployed were being done.

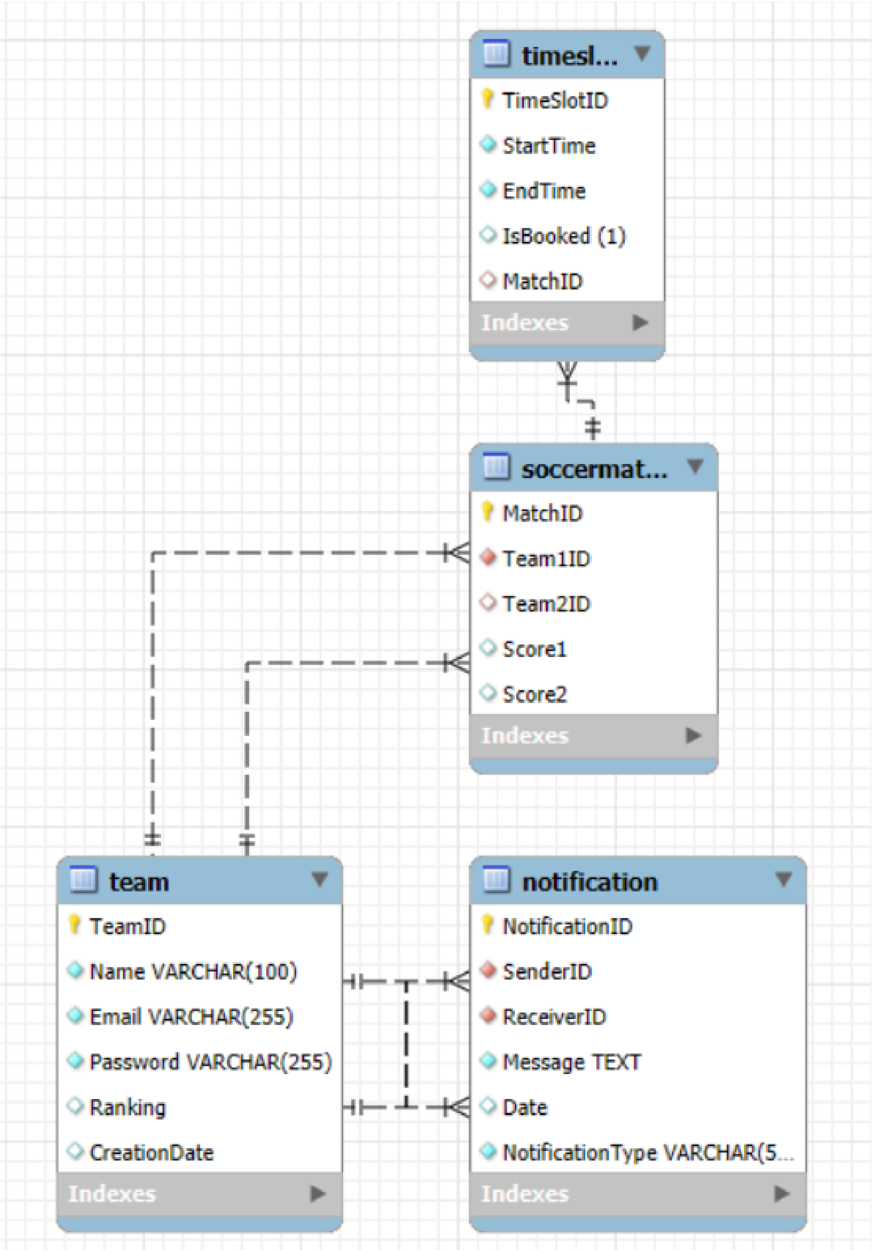


Figure 4: ER Diagram of Database

Yiğit Berk Atcı

In the first week of the project, we formed our project team and started to think about a project that would utilize cloud technology. The fact that the project would also be available for use after course would be an extra motivation for us. Therefore, we decided that our SoccerMatch Scheduler project, which could also be used in school, was suitable after consulting the instructor.

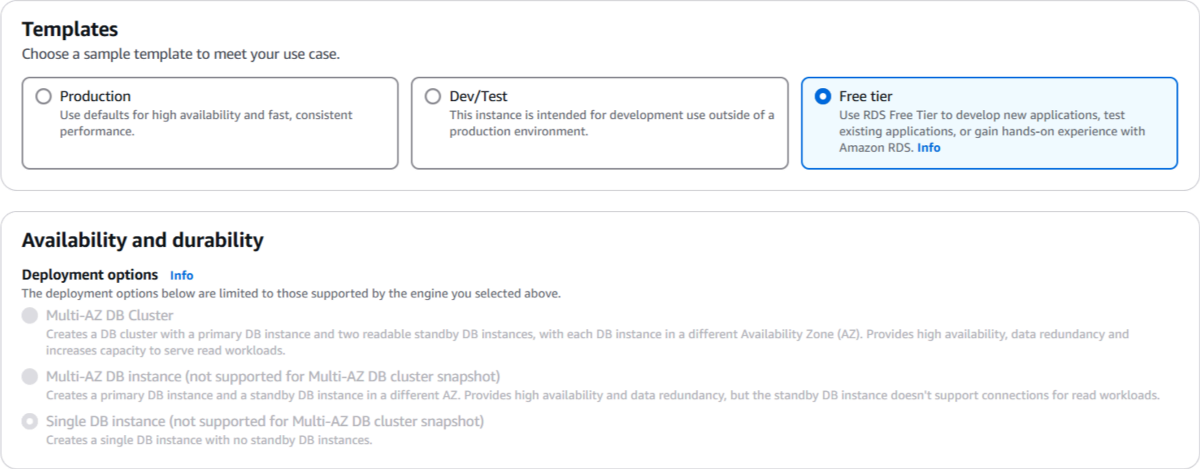
Alper Kutay Ören

In the first week, we explored sample projects utilizing cloud technologies to get an idea of what kind of project we could develop. Then, we came up with the idea of a Soccer Match Scheduler program that we could also use for ourselves. We decided the features of the project such as what kind of pages we need, what kind of database relations we need.

October 28 - November 1

Elif Ilgın Savaş

In the second week I explored through the depths of creating the Amazon RDS instances with free tier. Firstly, I created the database with some configuration. MySQL was selected as Engine Option then as a Template we selected Free Tier to be able use the instance for free. When the Free Tier is selected, configurations for the Free Tier are automatically selected. Therefore, not much modification was needed. For example, the deployment method is only limited to 'Single DB instance' which doesn't support Multi-AZ DB Cluster snapshot.



Templates

Choose a sample template to meet your use case.

○ Production
Use defaults for high availability and fast, consistent performance.

○ Dev/Test
This instance is intended for development use outside of a production environment.

● Free tier
Use RDS Free Tier to develop new applications, test existing applications, or gain hands-on experience with Amazon RDS. Info

Availability and durability

Deployment options Info
The deployment options below are limited to those supported by the engine you selected above.

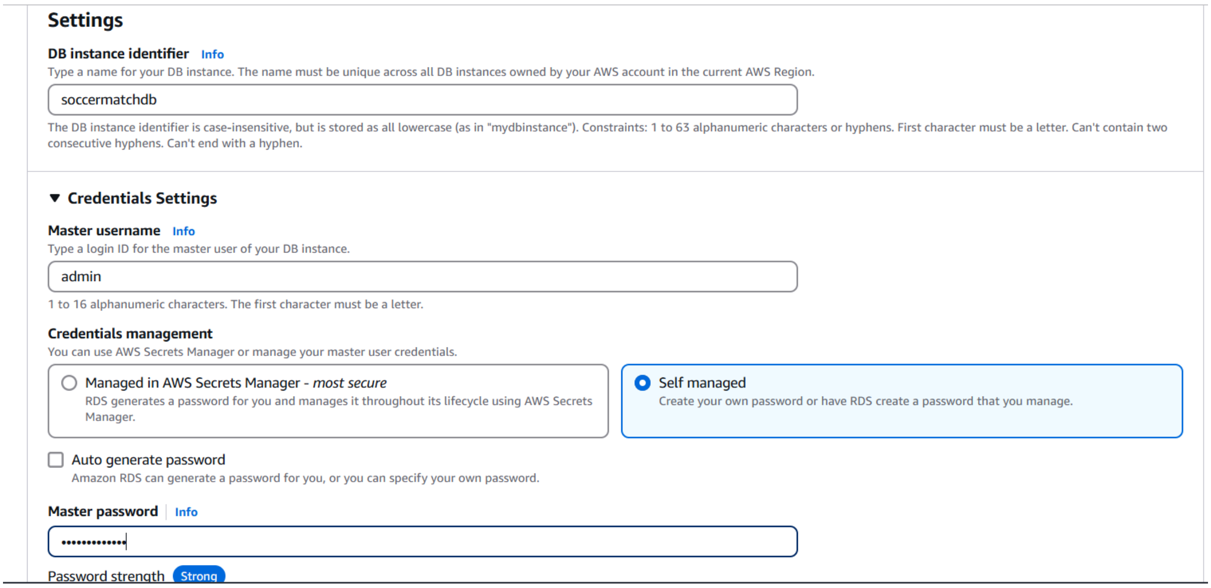
○ Multi-AZ DB Cluster
Creates a DB cluster with a primary DB instance and two readable standby DB instances, with each DB instance in a different Availability Zone (AZ). Provides high availability, data redundancy and increases capacity to serve read workloads.

○ Multi-AZ DB instance (not supported for Multi-AZ DB cluster snapshot)
Creates a primary DB instance and a standby DB instance in a different AZ. Provides high availability and data redundancy, but the standby DB instance doesn't support connections for read workloads.

● Single DB instance (not supported for Multi-AZ DB cluster snapshot)
Creates a single DB instance with no standby DB instances.

Figure 5: Database Creation on RDS – Part 1

I have set the database name, master username and password to connect to the database on MySQL Workbench.



Settings

DB instance identifier Info
Type a name for your DB instance. The name must be unique across all DB instances owned by your AWS account in the current AWS Region.

soccermatchdb

The DB instance identifier is case-insensitive, but is stored as all lowercase (as in "mydbinstance"). Constraints: 1 to 63 alphanumeric characters or hyphens. First character must be a letter. Can't contain two consecutive hyphens. Can't end with a hyphen.

▼ **Credentials Settings**

Master username Info
Type a login ID for the master user of your DB instance.

admin

1 to 16 alphanumeric characters. The first character must be a letter.

Credentials management
You can use AWS Secrets Manager or manage your master user credentials.

○ Managed in AWS Secrets Manager - *most secure*
RDS generates a password for you and manages it throughout its lifecycle using AWS Secrets Manager.

● Self managed
Create your own password or have RDS create a password that you manage.

☐ Auto generate password
Amazon RDS can generate a password for you, or you can specify your own password.

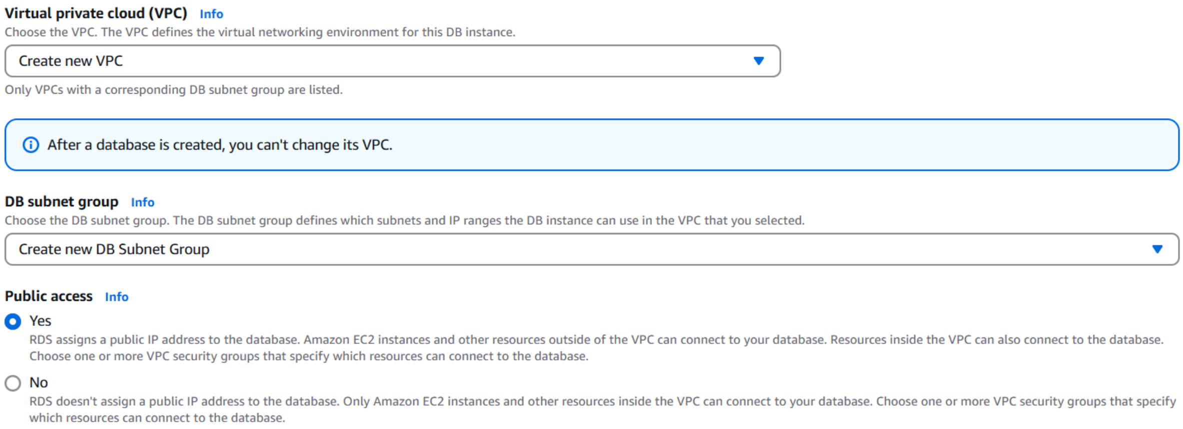
Master password Info

•••••••••••••

Password strength Strong

Figure 6: Database Creation on RDS – Part 2

After that, the only important part is to make the instance publicly accessible under Virtual Private Cloud (VPC) to connect the MySQL. Actually, I needed to re-create the database all over again due to a mistake of not making the instance publicly accessible.



Virtual private cloud (VPC) Info

Choose the VPC. The VPC defines the virtual networking environment for this DB instance.

Create new VPC ▼

Only VPCs with a corresponding DB subnet group are listed.

ⓘ After a database is created, you can't change its VPC.

DB subnet group Info

Choose the DB subnet group. The DB subnet group defines which subnets and IP ranges the DB instance can use in the VPC that you selected.

Create new DB Subnet Group ▼

Public access Info

● Yes

RDS assigns a public IP address to the database. Amazon EC2 instances and other resources outside of the VPC can connect to your database. Resources inside the VPC can also connect to the database. Choose one or more VPC security groups that specify which resources can connect to the database.

○ No

RDS doesn't assign a public IP address to the database. Only Amazon EC2 instances and other resources inside the VPC can connect to your database. Choose one or more VPC security groups that specify which resources can connect to the database.

Figure 7: Database Creation with RDS – Part 3

Yiğit Berk Atcı

After determining the project, we determined the cloud computing platform to be used, cloud delivery models and technologies suitable for the project and added them to our project proposal.

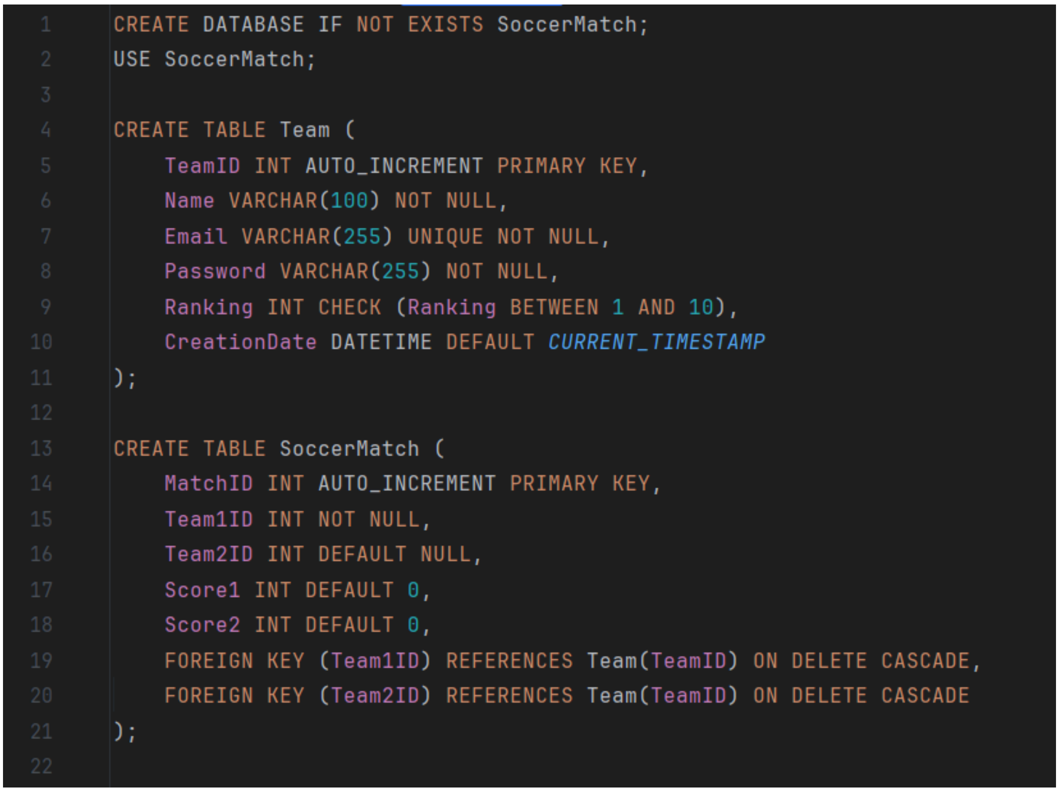
Alper Kutay Ören

In the second week, we decided that using AWS would be the most beneficial approach for us. Then, we conducted research on how to use AWS systems on our Soccer Match Scheduler web application.

November 4 - November 8

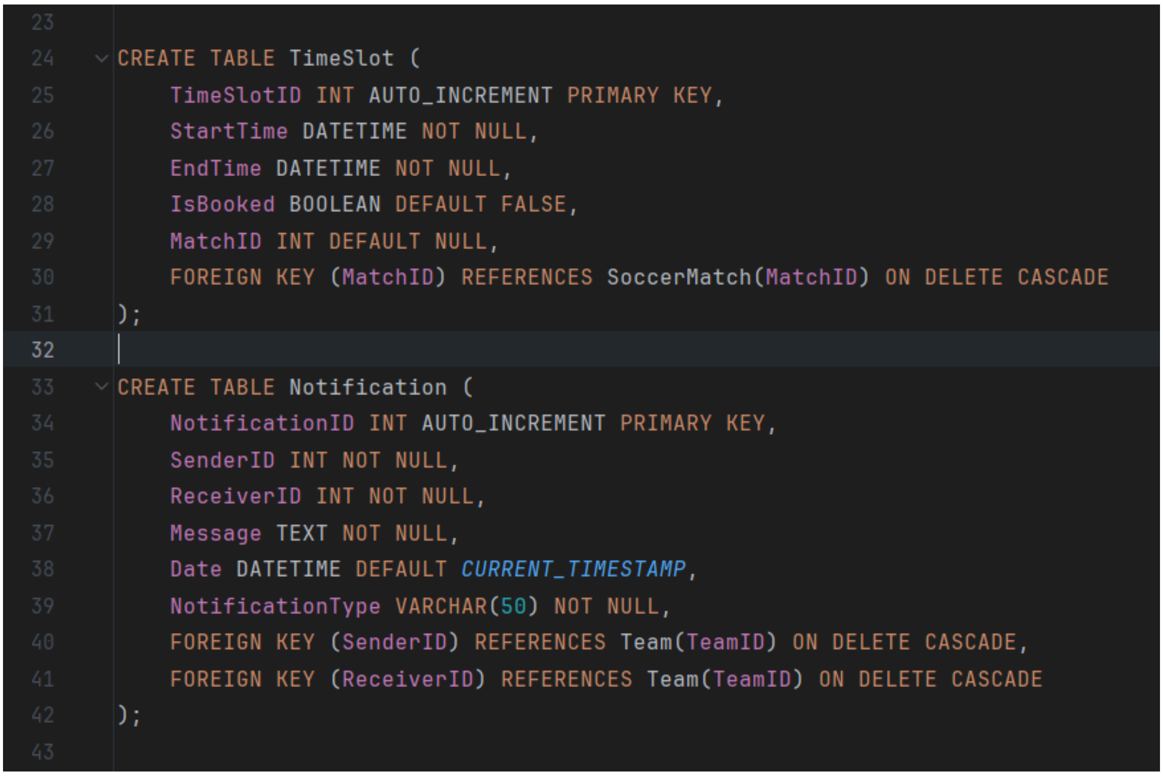
Elif Ilgın Savaş

In the third week I have coded the database codes for the implemented ER diagram done in the first week.



```sql
CREATE DATABASE IF NOT EXISTS SoccerMatch;
USE SoccerMatch;

CREATE TABLE Team (
    TeamID INT AUTO_INCREMENT PRIMARY KEY,
    Name VARCHAR(100) NOT NULL,
    Email VARCHAR(255) UNIQUE NOT NULL,
    Password VARCHAR(255) NOT NULL,
    Ranking INT CHECK (Ranking BETWEEN 1 AND 10),
    CreationDate DATETIME DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE SoccerMatch (
    MatchID INT AUTO_INCREMENT PRIMARY KEY,
    Team1ID INT NOT NULL,
    Team2ID INT DEFAULT NULL,
    Score1 INT DEFAULT 0,
    Score2 INT DEFAULT 0,
    FOREIGN KEY (Team1ID) REFERENCES Team(TeamID) ON DELETE CASCADE,
    FOREIGN KEY (Team2ID) REFERENCES Team(TeamID) ON DELETE CASCADE
);

```

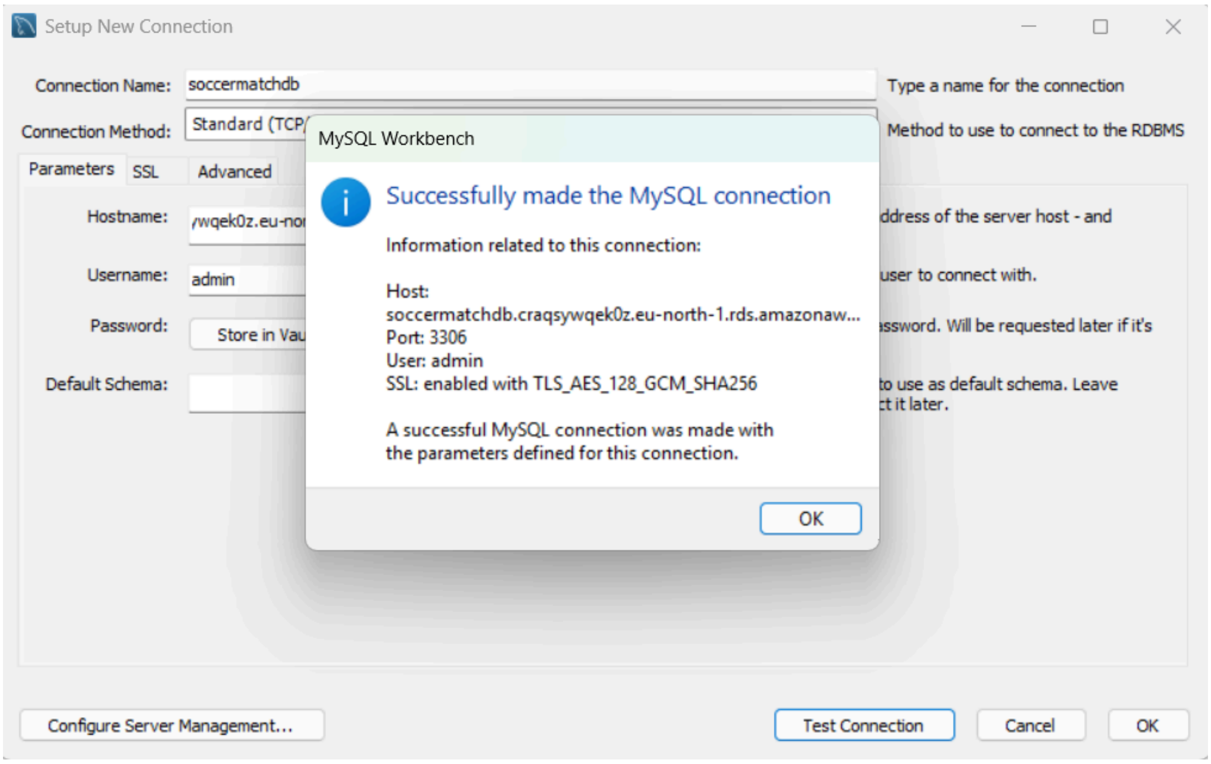
Figure 8: SQL – Part 1



```sql

CREATE TABLE TimeSlot (
    TimeSlotID INT AUTO_INCREMENT PRIMARY KEY,
    StartTime DATETIME NOT NULL,
    EndTime DATETIME NOT NULL,
    IsBooked BOOLEAN DEFAULT FALSE,
    MatchID INT DEFAULT NULL,
    FOREIGN KEY (MatchID) REFERENCES SoccerMatch(MatchID) ON DELETE CASCADE
);

CREATE TABLE Notification (
    NotificationID INT AUTO_INCREMENT PRIMARY KEY,
    SenderID INT NOT NULL,
    ReceiverID INT NOT NULL,
    Message TEXT NOT NULL,
    Date DATETIME DEFAULT CURRENT_TIMESTAMP,
    NotificationType VARCHAR(50) NOT NULL,
    FOREIGN KEY (SenderID) REFERENCES Team(TeamID) ON DELETE CASCADE,
    FOREIGN KEY (ReceiverID) REFERENCES Team(TeamID) ON DELETE CASCADE
);

```

Figure 9: SQL – Part 2

In the previous week I was able to create a database instance. Therefore, after the database on my MySQL Workbench was created, I have configured the connections to be able to connect to the Amazon RDS instance. I have created a MySQL Connection on MySQL Workbench with Hostname as the instance endpoint, Port as 3306 and username and password with the values I have set while creating the RDS database instance.



Setup New Connection

Connection Name: soccermatchdb

Connection Method: Standard (TCP/

Hostname: /wqek0z.eu-nor

Username: admin

Password: Store in Vau

Default Schema:

Type a name for the connection

Method to use to connect to the RDBMS

ddress of the server host - and

user to connect with.

ssword. Will be requested later if it's

to use as default schema. Leave
t it later.

MySQL Workbench

Successfully made the MySQL connection

Information related to this connection:

Host:
soccermatchdb.craqsywqek0z.eu-north-1.rds.amazonaw...
Port: 3306
User: admin
SSL: enabled with TLS_AES_128_GCM_SHA256

A successful MySQL connection was made with
the parameters defined for this connection.

Figure 10: MySQL Connection

Yiğit Berk Atcı

This week, we prepared the project proposal and specified which parts of the project each group member would contribute to, the cloud technologies to be used, and the outline of the project.

Alper Kutay Ören

This week, we wrote the project proposal which included diagrams of the required functions, descriptions of the AWS systems to be used, and specified which part of the project each group member would work on.

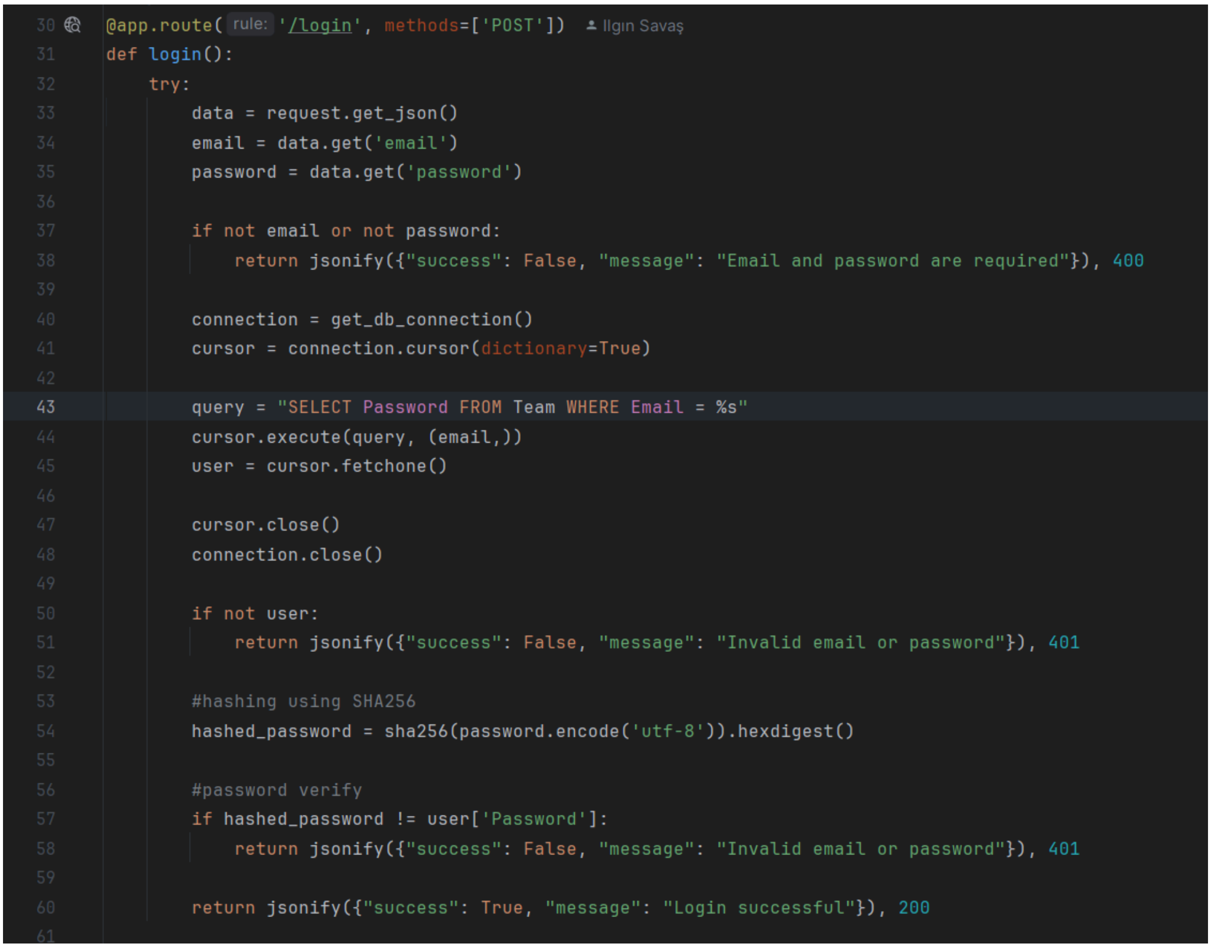
November 11 - November 15

Yiğit Berk Atcı

After the evaluation of our project proposal, I researched cloud technologies in more detail and used the "AWS educate" training offered by AWS at this stage. The learning phase took longer than expected due to the complexity of the AWS system interface.

Elif Ilgın Savaş

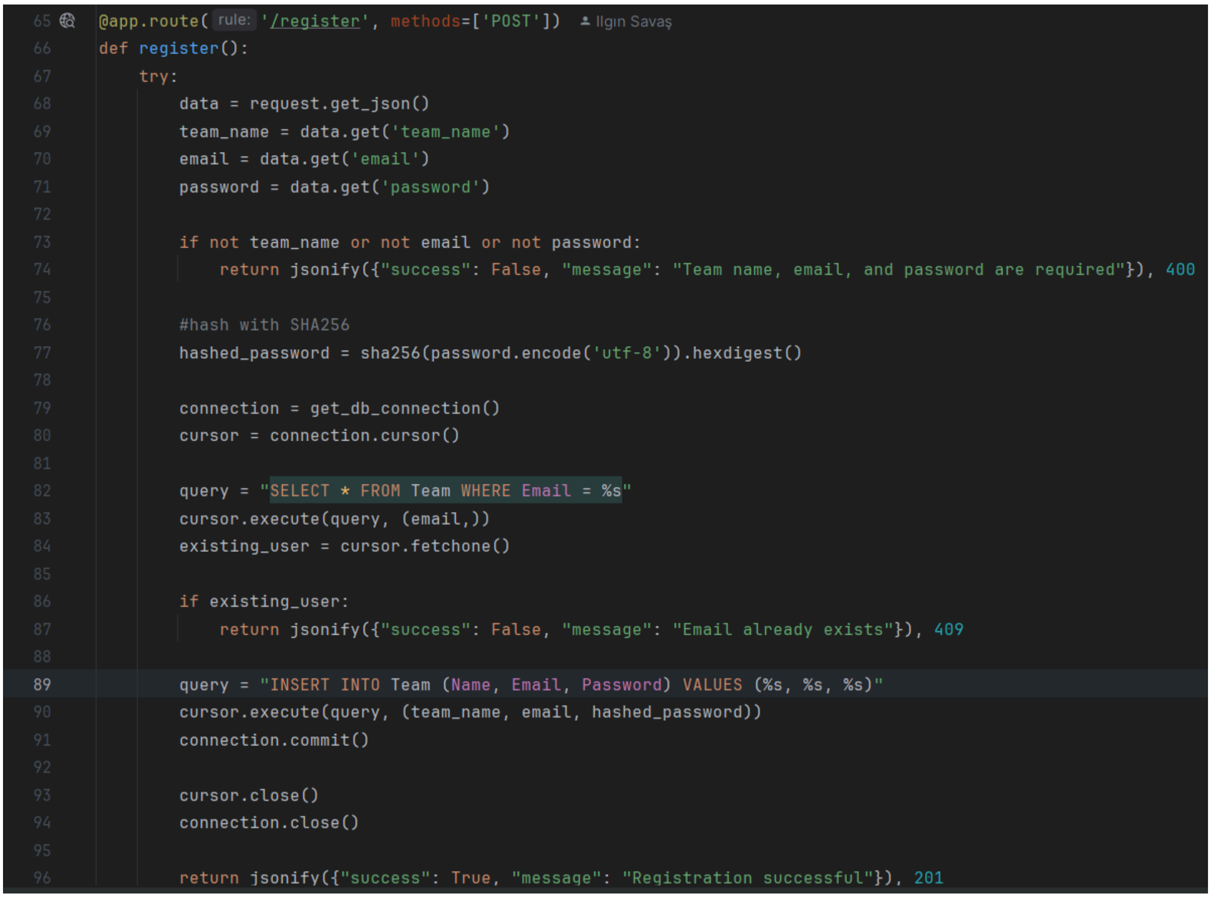
After we were billed with some unfortunate event we have decided to focus more on the application implementation of the project. Therefore, this week I have started to code the backend of the ‘Login’ and ‘Register’ pages. Also, tested the code with the database.



```
@app.route('/login', methods=['POST'])
def login():
    try:
        data = request.get_json()
        email = data.get('email')
        password = data.get('password')

        if not email or not password:
            return jsonify({"success": False, "message": "Email and password are required"}), 400

        connection = get_db_connection()
        cursor = connection.cursor(dictionary=True)

        query = "SELECT Password FROM Team WHERE Email = %s"
        cursor.execute(query, (email,))
        user = cursor.fetchone()

        cursor.close()
        connection.close()

        if not user:
            return jsonify({"success": False, "message": "Invalid email or password"}), 401

        #hashing using SHA256
        hashed_password = sha256(password.encode('utf-8')).hexdigest()

        #password verify
        if hashed_password != user['Password']:
            return jsonify({"success": False, "message": "Invalid email or password"}), 401

        return jsonify({"success": True, "message": "Login successful"}), 200
```

Figure 11: Flask API for Login Verification

In our database the password of each team is kept encrypted for extra security. We have chosen the SHA256 hashing algorithm since there is no option for decryption for the sake of security of user information. Therefore, before sending a password to the database, we need to encrypt.

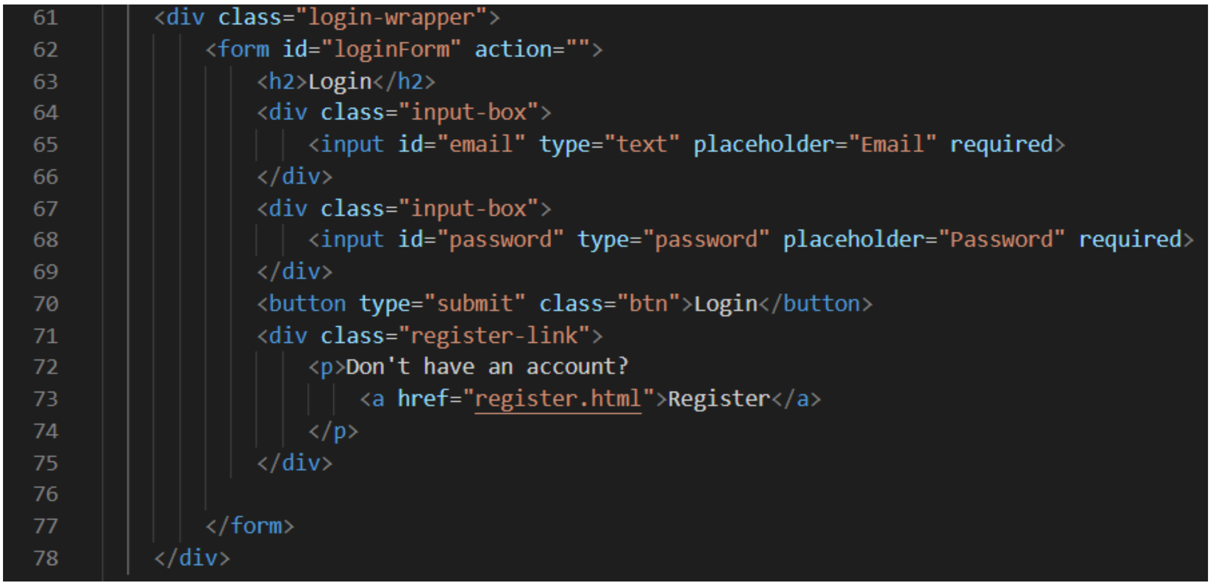


```
@app.route( rule: '/register', methods=['POST'])  Ilgın Savaş
def register():
    try:
        data = request.get_json()
        team_name = data.get('team_name')
        email = data.get('email')
        password = data.get('password')

        if not team_name or not email or not password:
            return jsonify({"success": False, "message": "Team name, email, and password are required"}), 400

        #hash with SHA256
        hashed_password = sha256(password.encode('utf-8')).hexdigest()

        connection = get_db_connection()
        cursor = connection.cursor()

        query = "SELECT * FROM Team WHERE Email = %s"
        cursor.execute(query, (email,))
        existing_user = cursor.fetchone()

        if existing_user:
            return jsonify({"success": False, "message": "Email already exists"}), 409

        query = "INSERT INTO Team (Name, Email, Password) VALUES (%s, %s, %s)"
        cursor.execute(query, (team_name, email, hashed_password))
        connection.commit()

        cursor.close()
        connection.close()

        return jsonify({"success": True, "message": "Registration successful"}), 201
```

Figure 12: Flask API for Registration

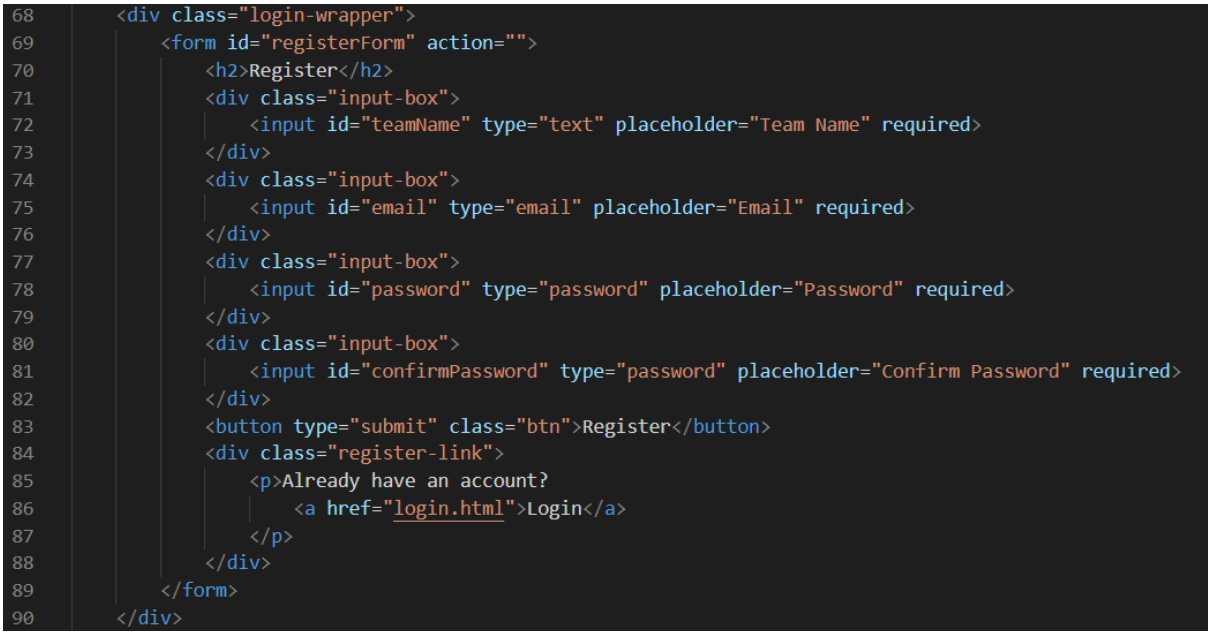
Alper Kutay Ören

I started to develop frontend codes for the project. I developed register and login pages by using HTML, CSS and JavaScript. In the initial stage, I designed a simple and user-friendly interface. To ensure it aligns with the school's colors, I used a red and white theme.



```html
<div class="login-wrapper">
    <form id="loginForm" action="">
        <h2>Login</h2>
        <div class="input-box">
            <input id="email" type="text" placeholder="Email" required>
        </div>
        <div class="input-box">
            <input id="password" type="password" placeholder="Password" required>
        </div>
        <button type="submit" class="btn">Login</button>
        <div class="register-link">
            <p>Don't have an account?
                <a href="register.html">Register</a>
            </p>
        </div>

    </form>
</div>
```

Figure 13: HTML file for Login Page



```html
<div class="login-wrapper">
    <form id="registerForm" action="">
        <h2>Register</h2>
        <div class="input-box">
            <input id="teamName" type="text" placeholder="Team Name" required>
        </div>
        <div class="input-box">
            <input id="email" type="email" placeholder="Email" required>
        </div>
        <div class="input-box">
            <input id="password" type="password" placeholder="Password" required>
        </div>
        <div class="input-box">
            <input id="confirmPassword" type="password" placeholder="Confirm Password" required>
        </div>
        <button type="submit" class="btn">Register</button>
        <div class="register-link">
            <p>Already have an account?
                <a href="login.html">Login</a>
            </p>
        </div>
    </form>
</div>
```

Figure 14: HTML file for Register Page

November 18 - November 22

Yiğit Berk Atcı

This week, I gave them IAM user privileges as root user and arranged the authorizations. As a result of this, other group members can use the common AWS account and follow the work by the group members.

Elif Ilgın Savaş

This week I have worked on the integration of Frontend and Backend. Added script codes the connection and tested the system all together for Login and Register pages. For the details of the script please refer to GitHub repository.

Alper Kutay Ören

I started working on the home page this week. There are two main elements on the home page. The first of these is the ranking table, which ranks the teams according to their win rates. The second is the daily scheduler, which shows the available time slots on the current day. Initially these tables were filled manually to create an example visual. In addition, I wrote the header bar which includes the application name, “Calendar” and “MyTeam” items.

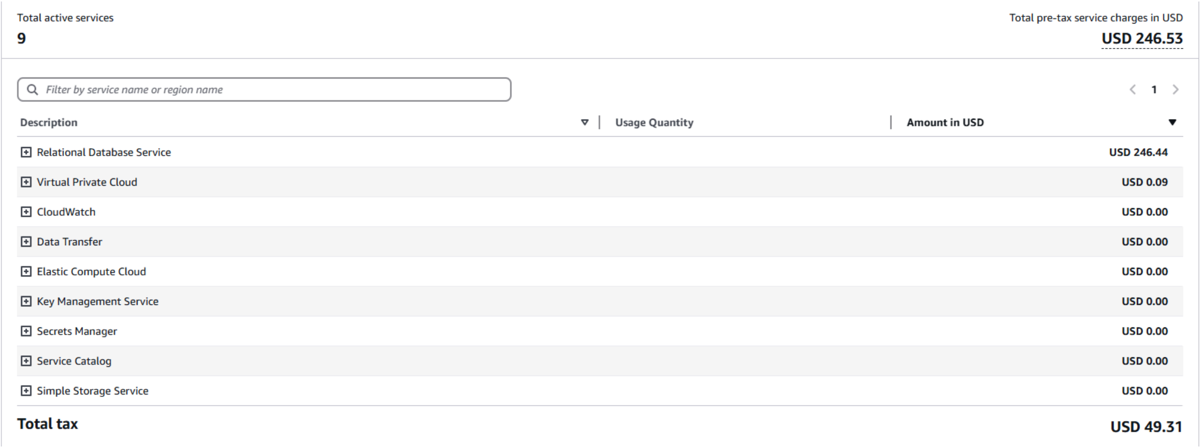
November 24 - November 29

Elif Ilgın Savaş

This week we have worked on report writing of the project. Thankfully, I had the screenshots of each step we have completed to deploy or use the cloud services. With what we had in our hands we tried to complete the report for the final submission.

Yiğit Berk Atcı

This week we noticed that we were being billed after the work was deployed to AWS. Although we used the free usage options offered by AWS, after a bill of approximately $300, we decided to remove all our work deployed to AWS and work locally and deploy our work to stage 3. As a result, in order not to be charged too much, our work will be kept local at this stage and cloud technologies will be actively used in the next stage.



Total active services
9

Total pre-tax service charges in USD
USD 246.53

Filter by service name or region name

Description	Usage Quantity	Amount in USD
Relational Database Service		USD 246.44
Virtual Private Cloud		USD 0.09
CloudWatch		USD 0.00
Data Transfer		USD 0.00
Elastic Compute Cloud		USD 0.00
Key Management Service		USD 0.00
Secrets Manager		USD 0.00
Service Catalog		USD 0.00
Simple Storage Service		USD 0.00
Total tax		USD 49.31

Figure 15: AWS usage bill for November

Alper Kutay Ören

This week, we started writing our report while continuing to integrate the project's code. Due to an issue, we were unable to deploy our project to the AWS system and test the login and register functions on AWS.

Utilization of Cloud Technologies

It was decided to use Amazon Web Services to develop the project and to use cloud technologies. In this stage, basic information about cloud technologies and AWS usage was obtained by using AWS educate, which AWS offers to its users. In AWS, it was decided to use Relational Database Service (RDS), and Amazon Elastic Compute Cloud (Amazon EC2) for stage 2. Amazon Simple Notification Service (Amazon SNS) was planned to be done in stage 3.

Amazon Elastic Compute Cloud (Amazon EC2)

Within the Amazon Web Services (AWS) Cloud, Amazon Elastic Compute Cloud (Amazon EC2) offers scalable, on-demand processing power. Because Amazon EC2 lowers hardware expenses, you can create and launch apps more quickly. With Amazon EC2, you can set up networking and security, control storage, and launch as many or as few virtual servers as you require. A virtual server in the AWS Cloud is called an EC2 instance. The hardware available to your instance depends on the instance type you pick when you launch an EC2 instance. Every instance type provides a unique ratio of compute, memory, network, and storage resources.[1]

This service is planned to be used to provide the computational resources required to run the application

Amazon Relational Database Service

An online service called Amazon Relational Database Service (Amazon RDS) facilitates the setup, management, and scalability of relational databases in the AWS Cloud. It handles common database administration duties and offers an industry-standard relational database with scalable, cost-effective capacity.[2]

This service is planned to be used to store information about teams, matches and daily time slot table in the system.

Remaining Milestones

Milestones	**Responsible**
Amazon Simple Notification Service (SNS)	Yiğit Berk Atcı
AWS Authorizations	Yiğit Berk Atcı
Finishing Backend and Integration	Elif Ilgın Savaş
Deployment of API on the Cloud	Elif Ilgın Savaş
Deployment of Database on the Cloud	Elif Ilgın Savaş
Finishing Frontend and Integration	Alper Kutay Ören
Testing Project on AWS	Alper Kutay Ören

Figure 16: Planned Milestones List

December 2 - December 6

Yiğit Berk Atcı

This week, I will do detailed research on the Simple Notification System that needs to be done in the cloud part of the project.

Alper Kutay Ören

I will continue to develop frontend of the project. We need to have an extra page for time slots booking. Also, I will start to develop Calendar and MyTeam pages.

Elif Ilgın Savaş

In this period I plan to continue coding the backend. I will implement the necessary codes for time slot table, time slot booking operations, ranking table.

December 9 - December 13

Yiğit Berk Atcı

This week I will try to implement the SNS part using the research I did last week.

Alper Kutay Ören

I'm planning to utilize the pages in order to make them more user-friendly and finish the frontend of the project. In addition, I will connect the new pages with the backend part.

Elif Ilgın Savaş

I will continue with the integration part of the coding with both the database and the frontend code. Also, I will do some testing to ensure coherent system design.

December 16 - December 20

Yiğit Berk Atcı

This week, I plan to work on deploying the frontend and backend parts of the project to AWS. I will follow the authorizations required by the group members and the project budget from the system.

Alper Kutay Ören

This week, I will add my codes into the AWS system and establish the required connections. Also, I will test the system in order to detect and prevent the errors.

Elif Ilgın Savaş

This week I will work on the API for triggering the Amazon SNS and test the system all together since the application will be deployed on the cloud systems fully.

December 23 - December 27

Yiğit Berk Atcı

This week, I plan to create a report of the project developed throughout the semester and present information about the technologies used in AWS and the project process in the report.

Alper Kutay Ören

This week, I'm planning to start writing the report presenting the project.

Elif Ilgın Savaş

In the last week of this project I will check if there is any error to be corrected and improve if needed. Afterwards, I will continue on with the report for final presentation.

GitHub Repository

https://github.com/yigitberkatci/CNG495-CloudComputing.git

References

1. *What is Amazon EC2? - Amazon Elastic Compute Cloud*. (n.d.). https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/concepts.html
2. *What is Amazon Relational Database Service (Amazon RDS)? - Amazon Relational Database Service*. (n.d.). https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Welcome.html