# Coaching Management System

CASE STUDY REPORT

By

**Prateek: RA2211028010001**

**Parth Mishra: RA2211028010039**

**Ujjawal Sharma : RA2211028010026**

Under the guidance of

Dr. S. Thenmalar

*In partial fulfilment for the Course*

of

**21CSE261T – FUNDAMENTALS OF CLOUD COMPUTING**

in Department of Networking and Communications



**FACULTY OF ENGINEERING AND TECHNOLOGY SCHOOL OF COMPUTING**

**SRM INSTITUTE OF SCIENCE AND TECHNOLOGY KATTANKULATHUR**

**MAY 2024**

**SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**

**(Under Section 3 of UGC Act, 1956)**

**BONAFIDE CERTIFICATE**

Certified that this case study report for the course **21CSE261T FUNDAMENTALS OF CLOUD COMPUTING** entitled " Coaching Management System" is the bonafide work of Prateek(RA2211028010001),Parth Mishra (RA2211028010039) and Ujjawal Sharma (RA2211028010026) who carried out the work under my supervision.

**FACULTY IN-CHARGE HEAD OF DEPARTMENT**

Dr. S. Thenmalar Dr. Annapurani K.

Associate Professor Professor and Head

Department of Networking and Communications Department of Networking and Communications

SRM Institute of Science and Technology SRM Institute of Science and Technology

Kattankulathur Kattankulanthur

**ABSTRACT**

The Online Coaching Management System (CMS) website is a comprehensive platform designed to streamline the management and delivery of coaching services in a digital environment. This web-based application serves as a centralized hub for coaches, clients, and administrators, offering a range of features to enhance communication, scheduling, content delivery, and administrative tasks.

Key functionalities of the CMS include user authentication and authorization, allowing coaches and clients to securely access personalized accounts. Coaches can create and manage coaching sessions, define course curriculums, and upload educational materials such as videos, documents, and quizzes. Clients can browse available coaching services, enroll in courses, schedule sessions, and track their progress through interactive dashboards.

The CMS facilitates seamless communication between coaches and clients through messaging systems, discussion forums, and virtual meeting tools. It also supports online payments and invoicing for coaching services, providing a convenient and secure payment gateway for transactions.

The Online Coaching Management System website empowers coaches to deliver personalized coaching services efficiently, enables clients to access high-quality educational resources conveniently, and provides administrators with the tools to effectively manage and monitor coaching operations in an online environment.

|  |  |  |
| --- | --- | --- |
|  | **TABLE OF CONTENTS** |  |
| **S.NO** | **CONTENTS** | **PAGE NO** |
| **1** | **APPLICATION OVERVIEW** | **5** |
| **2** | **SLA REPORT** | 7 |
| **3** | **DESIGN OF APPLICATION**  **INFRASTRUCTURE** | 9 |
| **4** | **CREATION OF AWS EC2 INSTANCES** | 11 |
| **5** | **CREATION OF S3 BUCKET IN AWS** | 18 |
| **6** | **CREATION OF IAM IN AWS** | 26 |
| **7** | **IMPLEMENTATION** | 31 |
| **8** | **CONCLUSION** | 33 |
| **9** | **REFERENCES** | 34 |

**1.Application Overview**

The Online Coaching Management System (CMS) is a robust web-based application designed to facilitate the management and delivery of coaching services in a digital environment. This comprehensive platform serves as a centralized hub for coaches, clients, and administrators, offering a range of features to streamline various aspects of coaching operations.

Key Components:

User Management: The CMS provides user authentication and authorization functionalities to manage access for coaches, clients, and administrators. Users can create personalized accounts, update profiles, and securely log in to access relevant features and content.

Coaching Session Management: Coaches have the ability to create, schedule, and manage coaching sessions within the system. They can define session details, including duration, topics, and availability, and clients can browse and enroll in available sessions based on their preferences and schedules.

Course Management: Coaches can create and manage course curriculums within the CMS, organizing content into modules or lessons. They can upload educational materials such as videos, documents, presentations, and quizzes, providing clients with a structured learning experience.

Communication Tools: The CMS facilitates seamless communication between coaches and clients through various channels, including messaging systems, discussion forums, and virtual meeting tools. Coaches can provide personalized feedback, answer questions, and engage with clients in real-time.

Scheduling and Calendar Integration: Clients can view coaches' availability and schedule coaching sessions directly within the CMS. The system integrates with calendar applications to sync session schedules and send reminders to both coaches and clients.

Payment Processing: The CMS supports online payments and invoicing for coaching services, providing a secure payment gateway for transactions. Clients can make payments for coaching sessions or course enrollments through the platform, and administrators can manage billing and invoicing processes.

Administrative Tools: Administrators have access to comprehensive management tools to oversee user accounts, monitor system performance, generate reports, and manage content. They can track coaching activities, analyze user engagement, and make data-driven decisions to optimize coaching operations.

Technology Stack:The Online Coaching Management System is built using modern web development technologies, including:

Frontend: HTML5, CSS3, JavaScript (React.js or Angular)

Backend: Node.js, Express.js, MongoDB or MySQL (for data storage)

Authentication and Authorization: JSON Web Tokens (JWT), OAuth 2.0

Communication: WebSockets for real-time messaging, RESTful APIs for data exchange

Payment Processing: Integration with payment gateways such as Stripe or PayPal

Deployment: Cloud-based hosting services (e.g., AWS, Google Cloud Platform) for scalability and reliability

Overall, the Online Coaching Management System provides a comprehensive solution for coaches to deliver personalized coaching services, enables clients to access high-quality educational resources conveniently, and empowers administrators to effectively manage and monitor coaching operations in an online environment

**2.SLA REPORT**

1. Introduction:

* The purpose of this SLA report is to outline the service level objectives (SLOs) and performance metrics for the Online Coaching Management System (CMS) project.
* This SLA governs the availability, performance, and support of the CMS platform for coaches, clients, and administrators.

1. Service Overview:

* The CMS platform provides a centralized hub for managing coaching services, including session scheduling, course management, communication tools, and administrative functionalities.
* The primary objective of the CMS is to ensure a seamless and efficient experience for coaches and clients in accessing coaching services and educational resources.

1. Service Level Objectives (SLOs):

* Availability: The CMS platform aims to achieve a minimum uptime of 99.9% per month, excluding scheduled maintenance windows.
* Response Time: The average response time for critical functions, such as login, session scheduling, and content access, should not exceed 1 second under normal operating conditions.
* Incident Resolution: The CMS commits to resolving critical incidents impacting service availability or functionality within 1 hour during business hours and within 4 hours outside of business hours.

1. Service Availability:

* Planned Maintenance: Scheduled maintenance windows will be communicated to users in advance, and downtime during these windows will not be counted towards uptime metrics.
* Unscheduled Downtime: In the event of unplanned downtime, the CMS will provide timely notifications to users and work diligently to restore service as quickly as possible.

1. Response Time:

* The CMS monitors response times for key functions and aims to maintain fast and consistent performance across all user interactions.
* Response time metrics will be regularly monitored and reported, with efforts made to optimize performance based on user feedback and system analytics.

1. Incident Management:

* Incident Reporting: Users can report incidents or service disruptions through designated channels, including a dedicated support email or ticketing system.
* Incident Escalation: Critical incidents will be escalated to the appropriate support personnel for immediate resolution, with regular updates provided to affected users until the issue is resolved.

1. Monitoring and Reporting:

* Performance Monitoring: The CMS utilizes monitoring tools to track system performance, including uptime, response times, and error rates.
* Reporting: SLA performance reports will be generated and shared with stakeholders on a monthly basis, detailing uptime percentages, response time averages, and incident resolution metrics.

1. Compliance and Governance:

* The CMS platform complies with industry standards and regulations related to data security, privacy, and accessibility.
* Regular audits and assessments will be conducted to ensure ongoing compliance with relevant laws and regulations.

1. Review and Improvement:

* This SLA will be periodically reviewed and updated to reflect changes in service requirements, user feedback, or technological advancements.
* Continuous improvement efforts will be undertaken to enhance the performance, reliability, and usability of the CMS platform over time.

**3.DESIGN OF APPLICATION INFRASTRUCTURE**

1. High-Level Architecture:

- The Online Coaching Management System (CMS) is designed as a scalable and resilient web-based application.

- The architecture follows a multi-tiered approach, with separate layers for presentation, application logic, and data storage.

2. Frontend Layer:

- The frontend layer is responsible for rendering the user interface and facilitating interactions with the CMS platform.

- Technologies: HTML5, CSS3, JavaScript (React.js or Angular) for dynamic and responsive UI design.

- Client-side routing to enable smooth navigation between pages and components.

- Integration with communication APIs for real-time messaging and collaboration features.

3. Backend Layer:

- The backend layer serves as the central processing unit of the CMS, handling business logic, data processing, and communication with external services.

- Technologies: Node.js, Express.js for building scalable and performant server-side applications.

- RESTful APIs for handling requests from the frontend layer and interacting with the database.

- Authentication and authorization middleware to secure access to resources and endpoints.

- Integration with third-party services for payment processing, email notifications, and analytics.

4. Database Layer:

- The database layer stores and manages the application's data, including user profiles, coaching sessions, course content, and system configurations.

- Technologies: MongoDB or MySQL for flexible and scalable data storage.

- Database schema design optimized for efficient data retrieval and management.

- Indexing and caching strategies to improve query performance and reduce latency.

5. Authentication and Authorization:

- The CMS implements robust authentication and authorization mechanisms to secure user accounts and sensitive data.

- Technologies: JSON Web Tokens (JWT) for token-based authentication and session management.

- OAuth 2.0 integration for third-party authentication providers (e.g., Google, Facebook).

- Role-based access control (RBAC) to enforce fine-grained permissions and access levels for different user roles (coaches, clients, administrators).

6. Communication Infrastructure:

- The CMS includes built-in communication tools to facilitate seamless interactions between coaches and clients.

- Technologies: WebSockets for real-time messaging and notifications.

- Integration with WebRTC for video conferencing and virtual meeting capabilities.

- Message queuing systems for asynchronous processing of communication events and notifications.

7. Scalability and Performance:

- The application infrastructure is designed to be horizontally scalable to accommodate growing user demand and data volume.

- Deployment on cloud-based platforms such as AWS, Google Cloud Platform, or Microsoft Azure for elastic scaling and high availability.

- Load balancing and auto-scaling configurations to distribute traffic evenly and optimize resource utilization.

- Performance monitoring and optimization techniques to identify and address bottlenecks in the application stack.

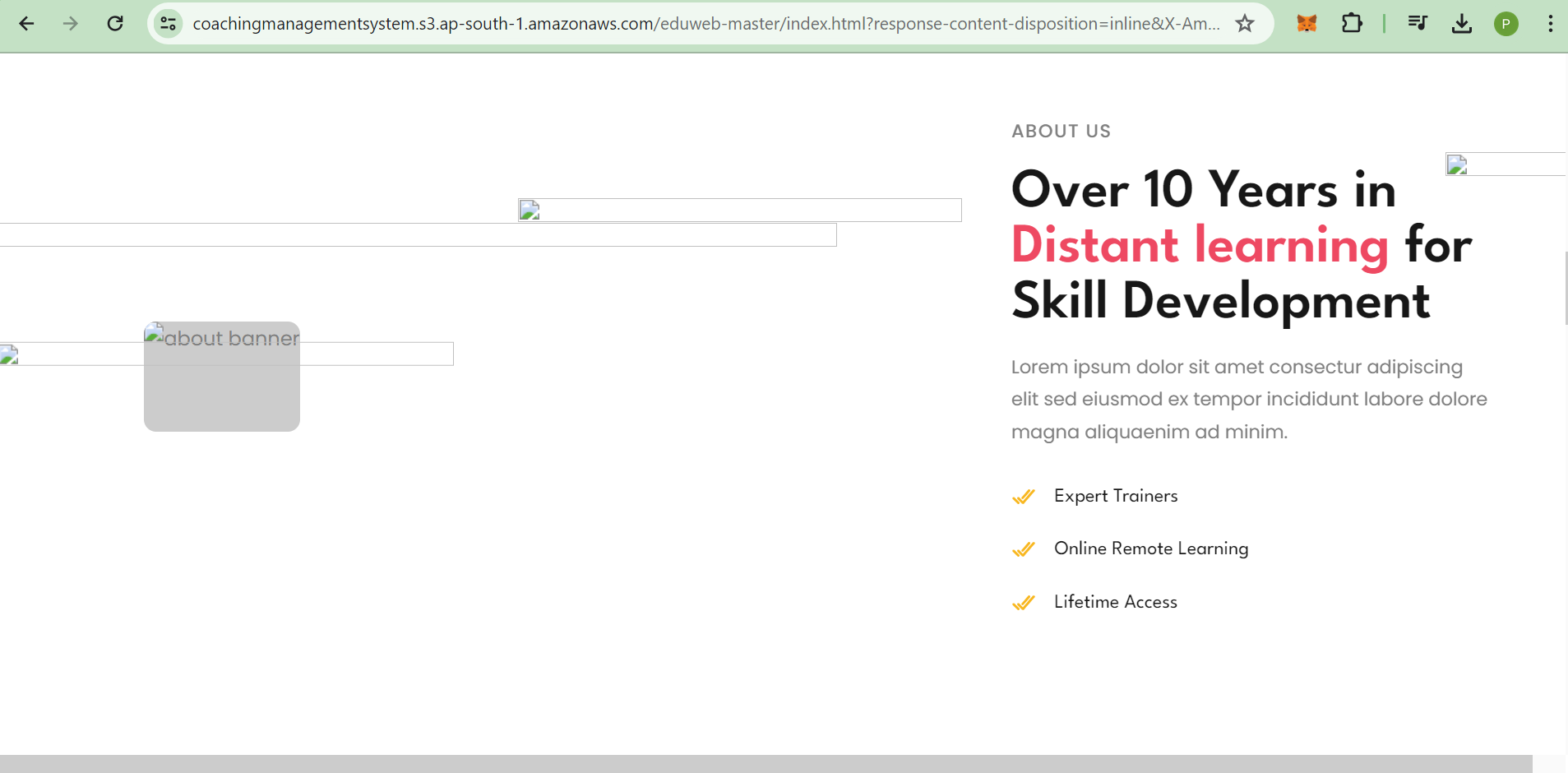
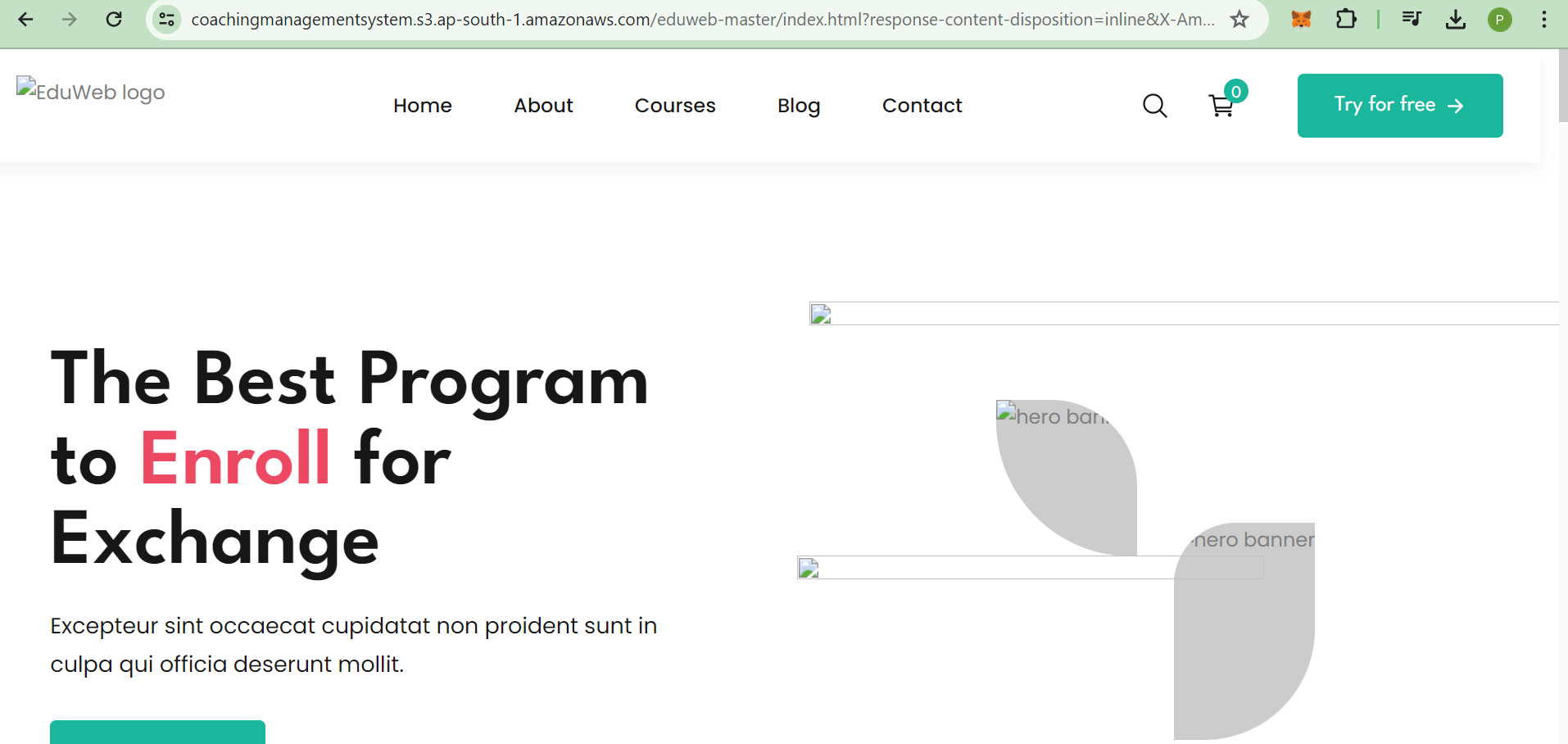
8. Security and Compliance:

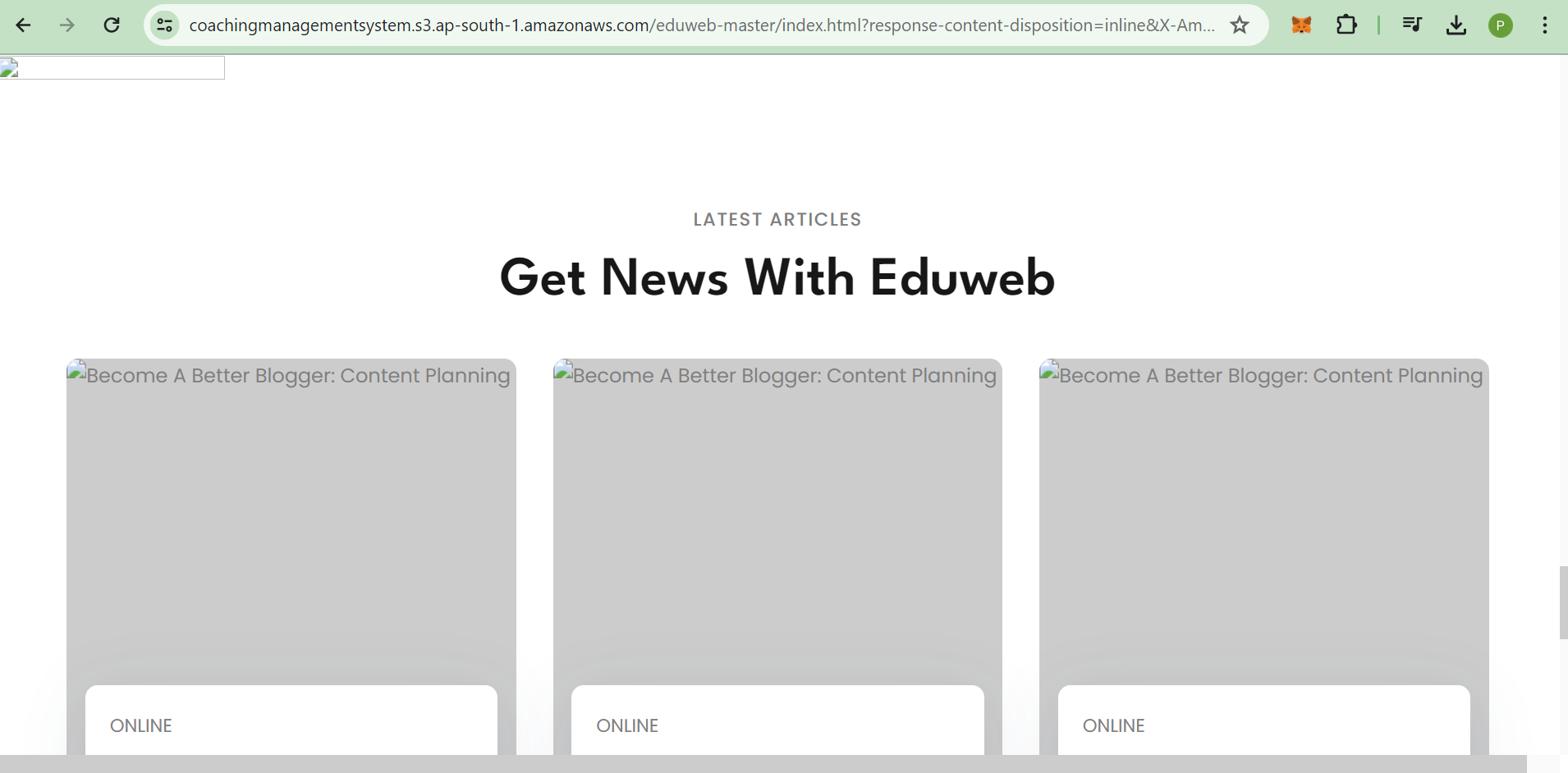
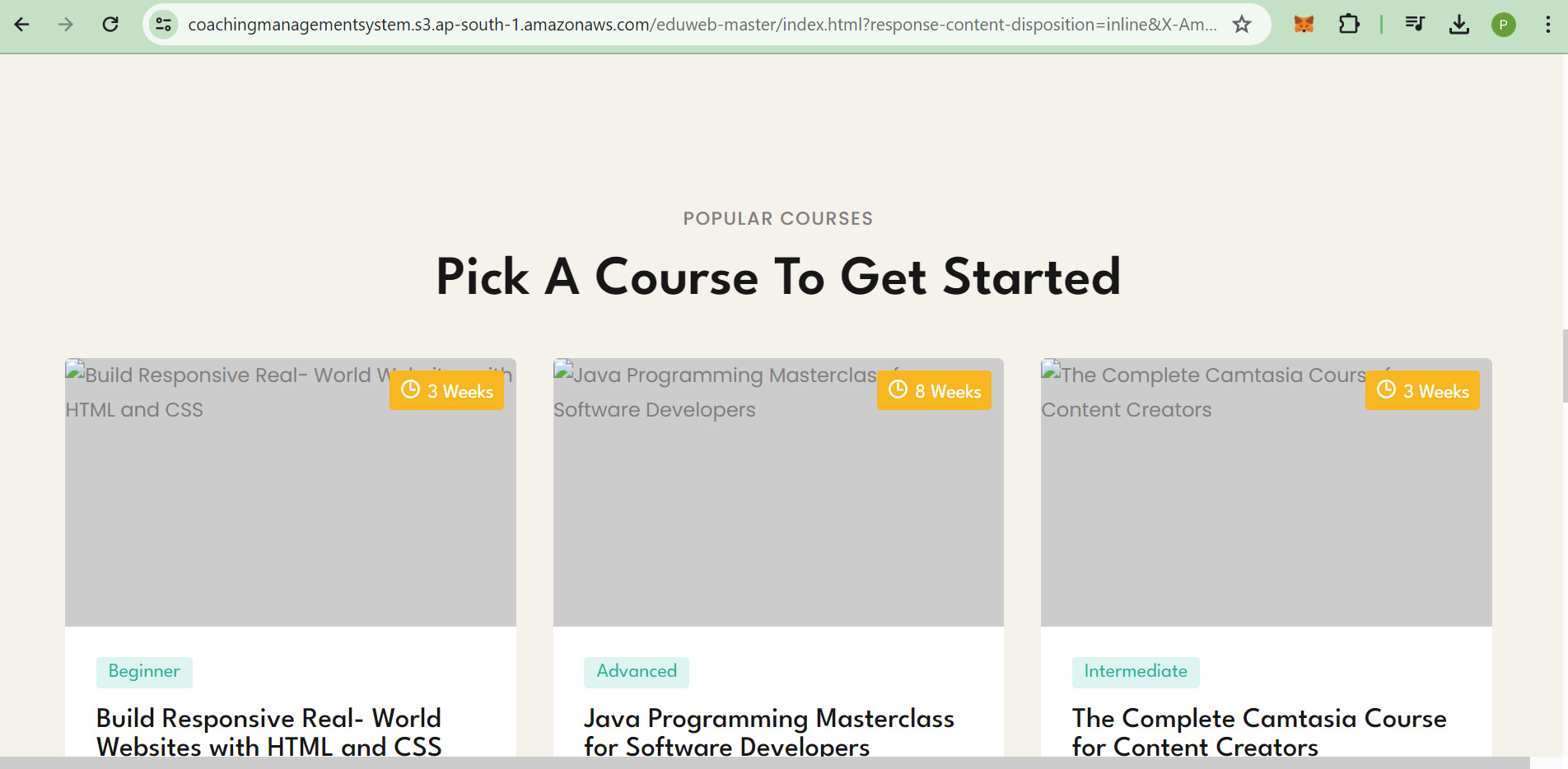
- The CMS prioritizes security and compliance with industry standards and regulations to protect user data and maintain trust.

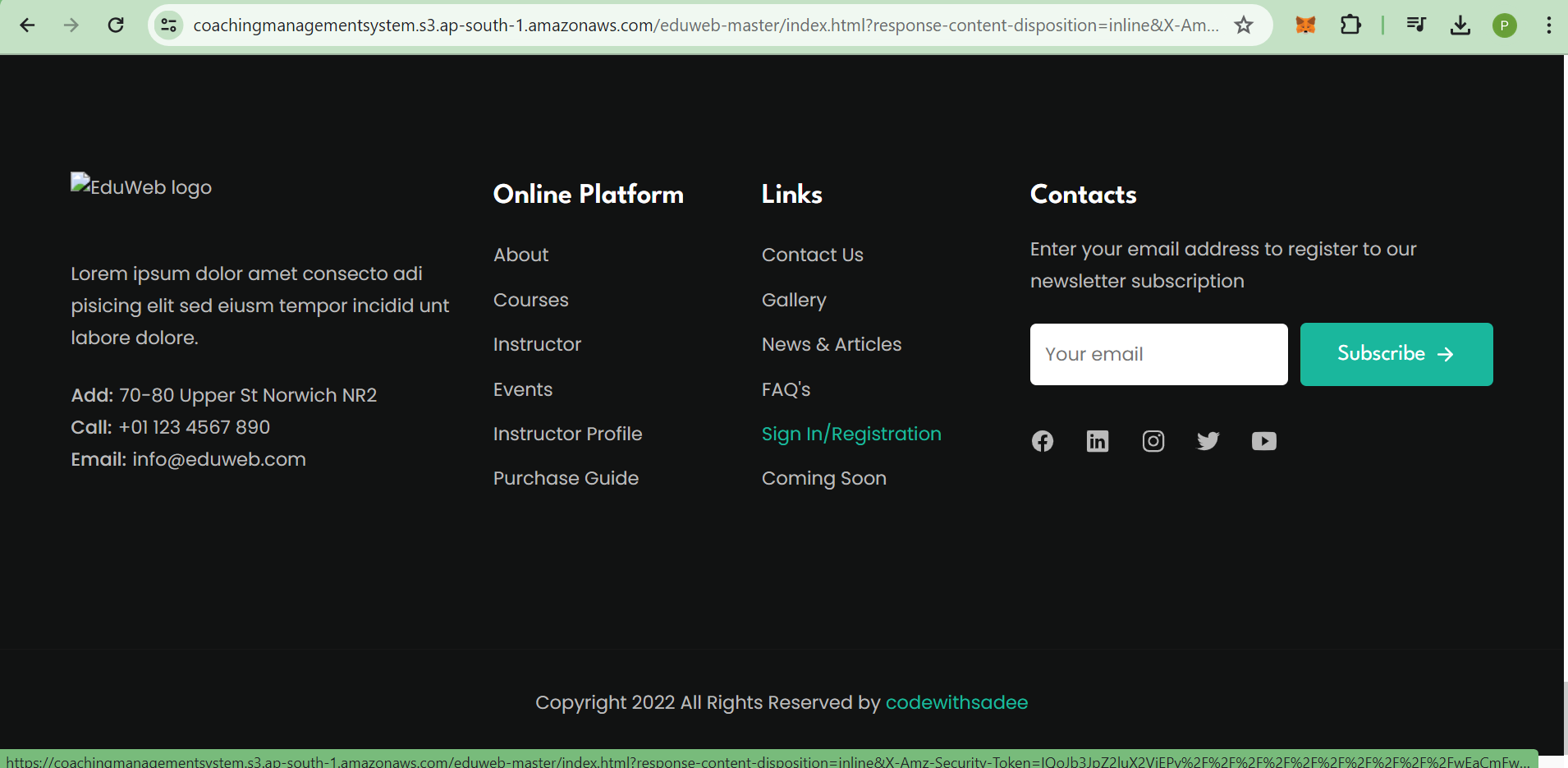
- Encryption mechanisms (e.g., HTTPS/TLS) for securing data transmission over the network.

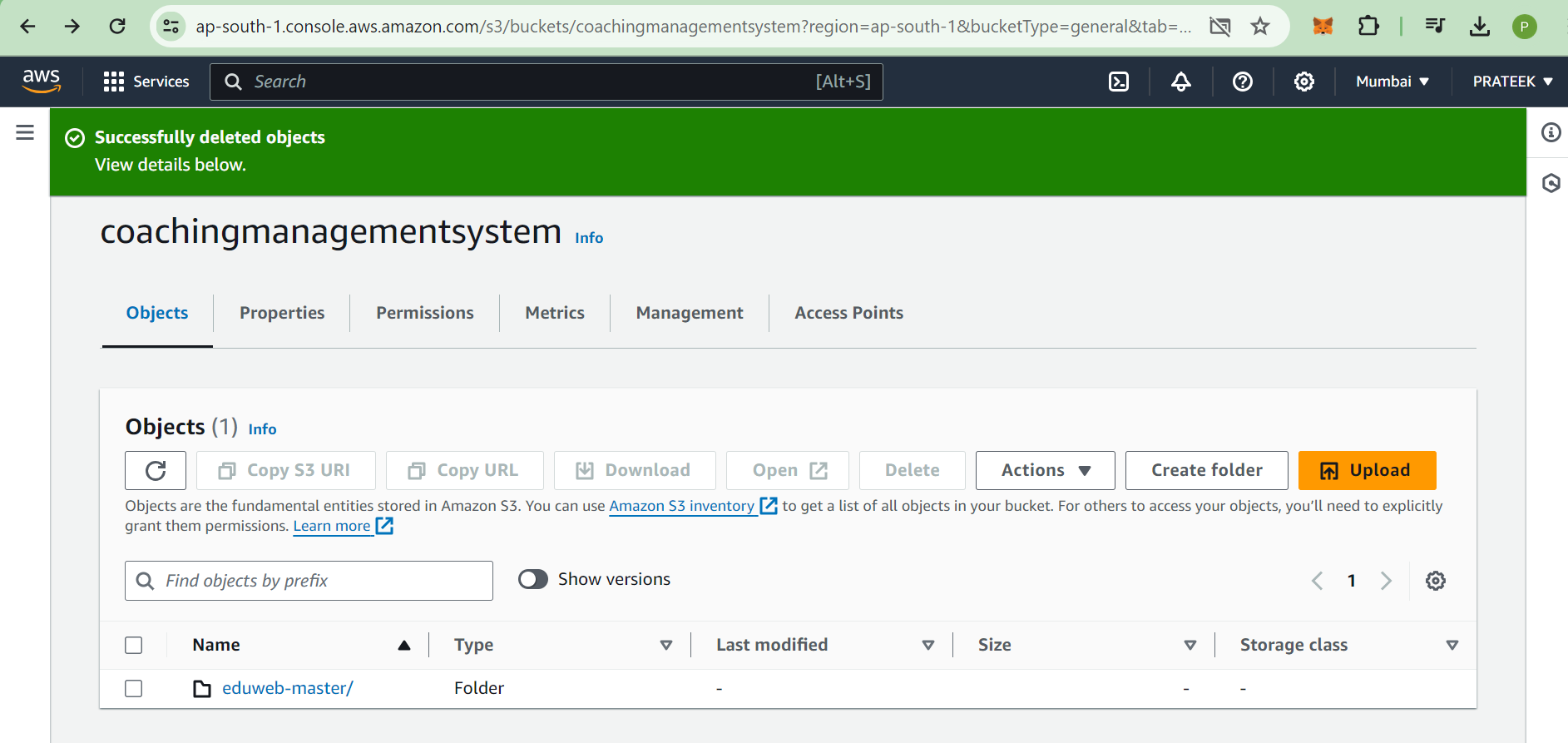
- Regular security audits and vulnerability assessments to identify and mitigate potential threats.

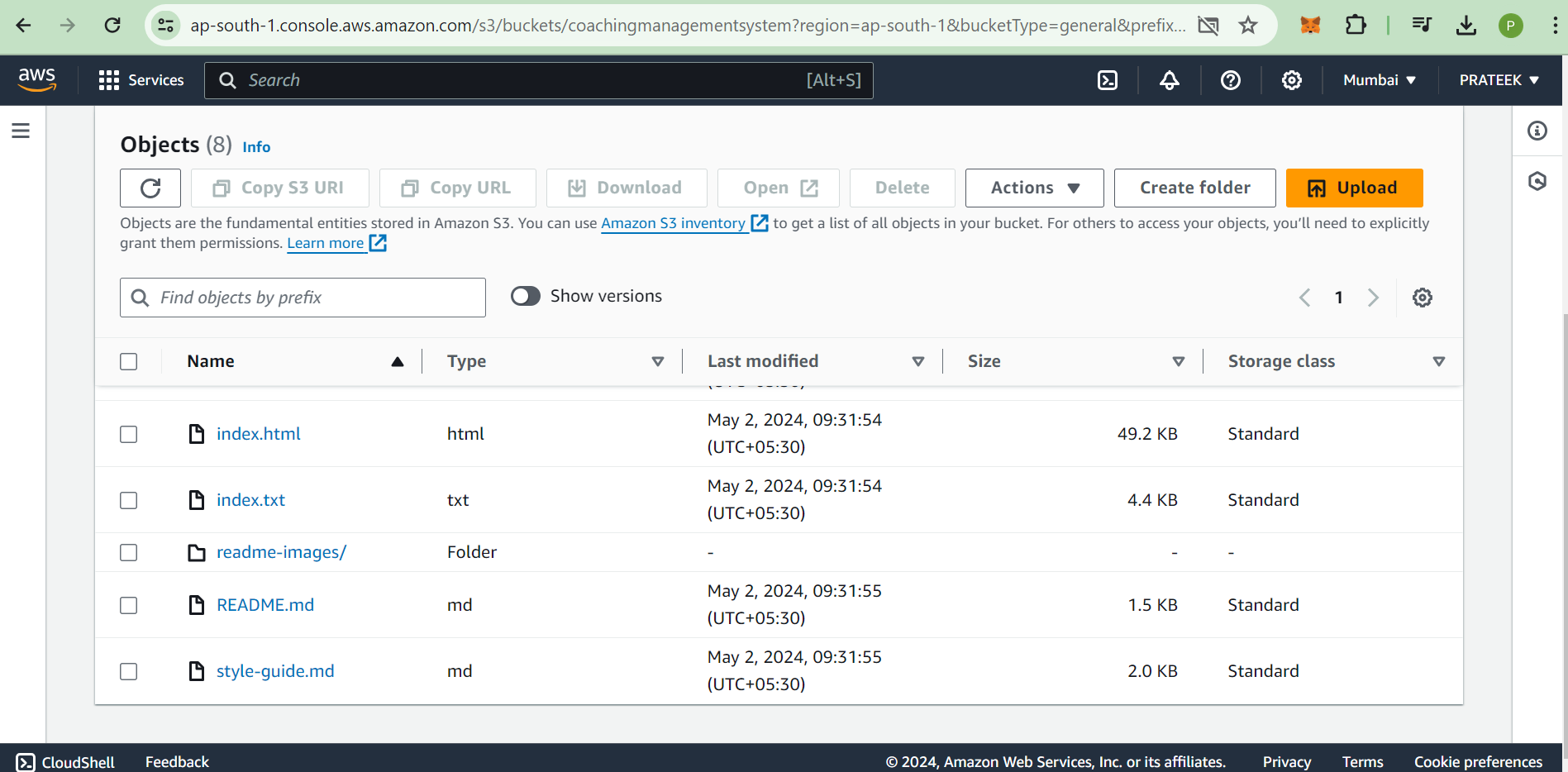
- Compliance with data privacy regulations (e.g., GDPR, CCPA) and industry-specific security standards.











**4. CREATION OF AWS EC2 INSTANCES**

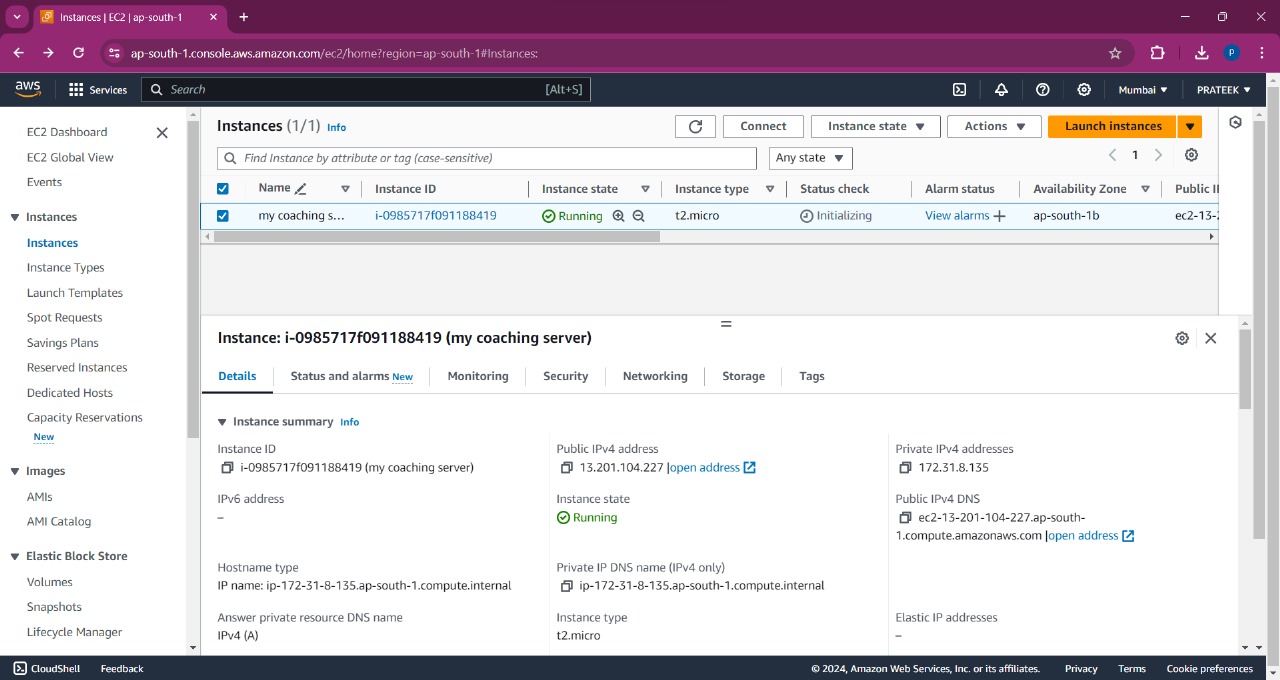
An EC2 instance has been created on through the AWS Management Console which enables us to store the files in a folder in the backend through which we can access any website.

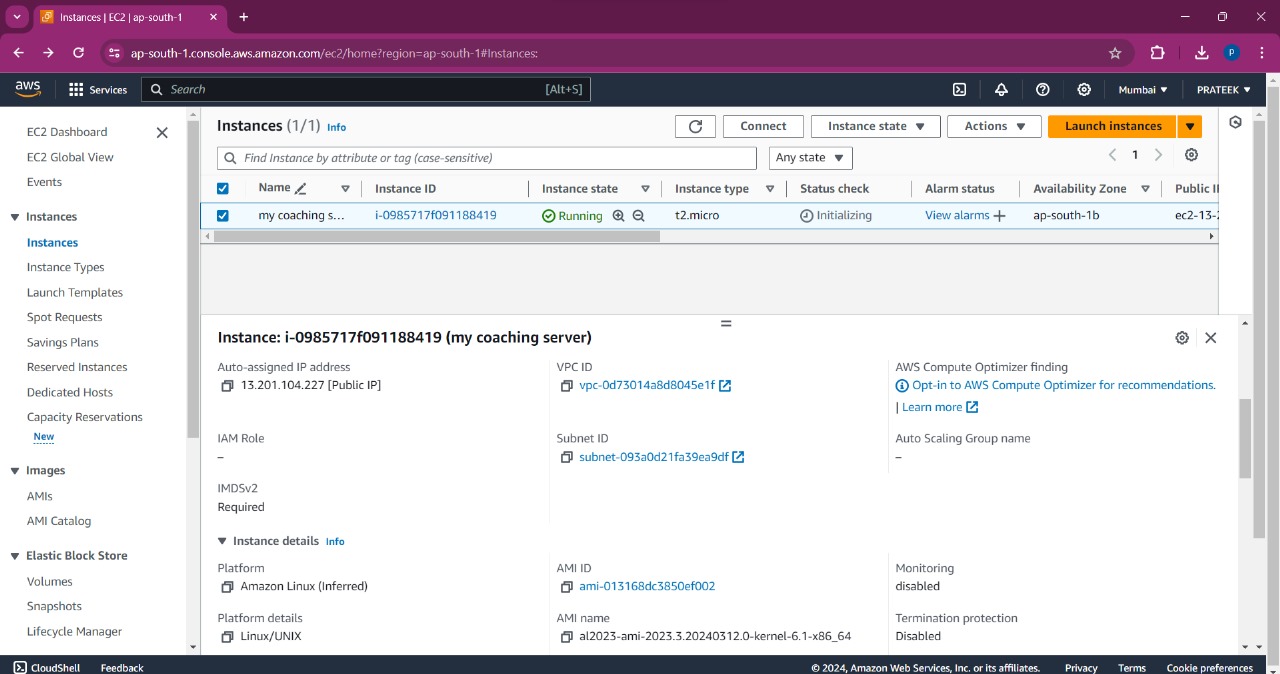
In the initial stage of this project, a web application has been made through which the user can upload files from their local storage. The uploaded files will be displayed on the website as a list from which the user can download the files.

## EC2 Instance

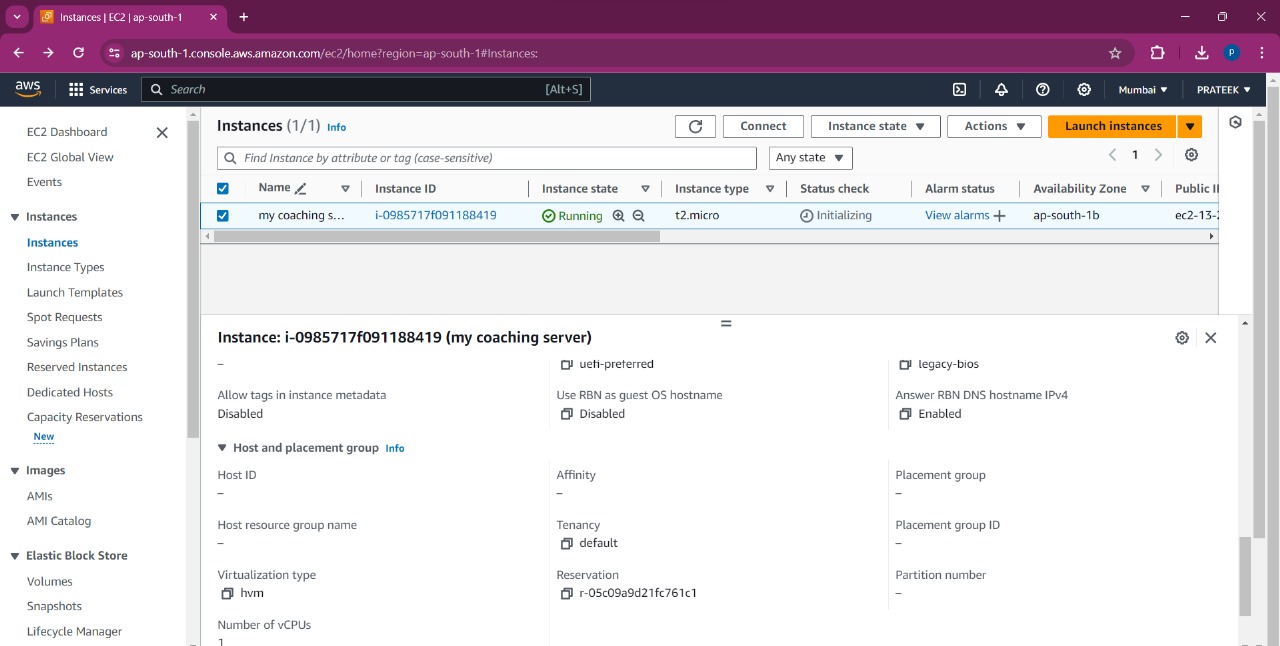
An EC2 instance named **coaching management system** has been created for our task.

The following images display the details of the instance.



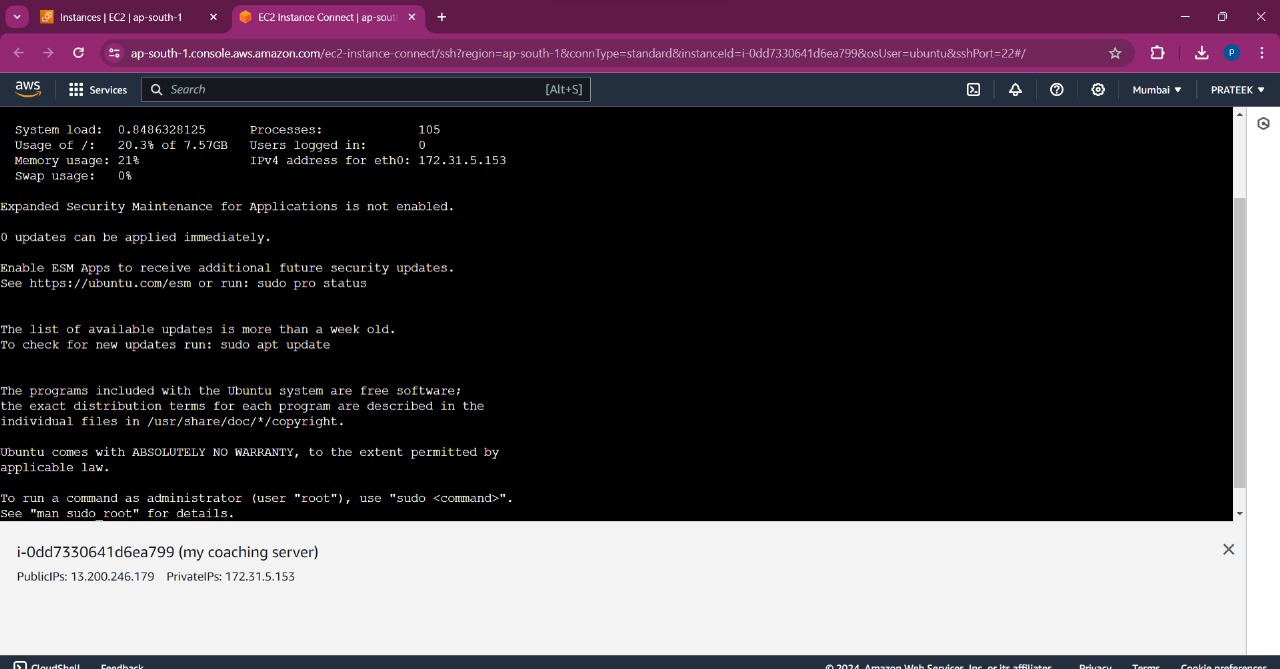






## Connecting to the instance

After running this instance, a connection was made through which we can access the terminal where we can perform various operations.



## app.js (Backend file)

This `app.js` file sets up an Express.js server to handle HTTP requests for fetching coaches and creating new coaches. It interacts with DynamoDB to store and retrieve coach data. Replace `'your-region'` and `'CoachesTable'` with your AWS region and DynamoDB table name, respectively.

// Import required libraries

const express = require('express');

const bodyParser = require('body-parser');

const AWS = require('aws-sdk');

// Initialize AWS SDK

AWS.config.update({region: 'your-region'}); // Specify your AWS region

const dynamodb = new AWS.DynamoDB.DocumentClient();

// Initialize Express app

const app = express();

app.use(bodyParser.json());

// Define routes

app.get('/coaches', async (req, res) => {

try {

// Fetch coaches data from DynamoDB table

const params = {

TableName: 'CoachesTable' // Specify your DynamoDB table name

};

const data = await dynamodb.scan(params).promise();

res.json(data.Items);

} catch (error) {

console.error('Error fetching coaches:', error);

res.status(500).json({ error: 'Failed to fetch coaches.' });

}

});

app.post('/coaches', async (req, res) => {

try {

// Create a new coach in DynamoDB

const { name, expertise, experience } = req.body;

const params = {

TableName: 'CoachesTable', // Specify your DynamoDB table name

Item: {

id: Date.now().toString(), // Generate a unique ID

name,

expertise,

experience

}

};

await dynamodb.put(params).promise();

res.status(201).json({ message: 'Coach created successfully.' });

} catch (error) {

console.error('Error creating coach:', error);

res.status(500).json({ error: 'Failed to create coach.' });

}

});

// Start the server

const port = process.env.PORT || 3000; // Use environment port or default to 3000

app.listen(port, () => {

console.log(`Server running on port ${port}`);

});

## index.html (Frontend file)

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Coaching Management System</title>

<!-- Add any necessary CSS stylesheets here -->

</head>

<body>

<h1>Coaching Management System</h1>

<!-- Form to add a new coach -->

<h2>Add New Coach</h2>

<form id="addCoachForm">

<label for="name">Name:</label>

<input type="text" id="name" name="name" required><br><br>

<label for="expertise">Expertise:</label>

<input type="text" id="expertise" name="expertise" required><br><br>

<label for="experience">Experience:</label>

<input type="number" id="experience" name="experience" required><br><br>

<button type="submit">Add Coach</button>

</form>

<!-- Container to display existing coaches -->

<h2>Existing Coaches</h2>

<ul id="coachesList">

<!-- Existing coaches will be populated here dynamically -->

</ul>

<!-- Add any necessary JavaScript scripts here -->

<script>

document.addEventListener('DOMContentLoaded', async () => {

const addCoachForm = document.getElementById('addCoachForm');

const coachesList = document.getElementById('coachesList');

// Function to fetch and display coaches

const fetchCoaches = async () => {

const response = await fetch('/coaches');

const coaches = await response.json();

coachesList.innerHTML = ''; // Clear existing list

coaches.forEach(coach => {

const listItem = document.createElement('li');

listItem.textContent = `Name: ${coach.name}, Expertise: ${coach.expertise}, Experience: ${coach.experience}`;

coachesList.appendChild(listItem);

});

};

**5. CREATION OF S3 BUCKET IN AWS**

**Step 1: Sign in to the AWS Management Console**

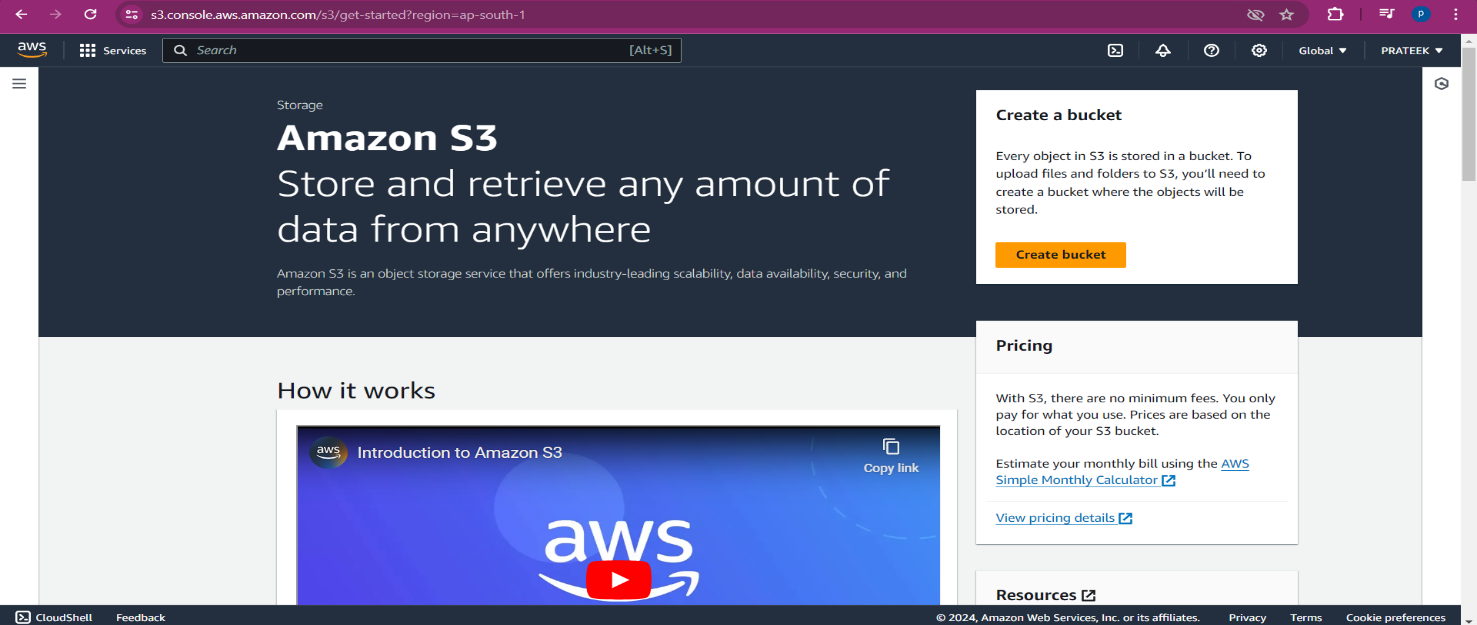
1. If you haven’t already, sign up for an AWS account. Amazon provides a **1-year free tier** for new users.
2. Log in to the **AWS Management Console**.

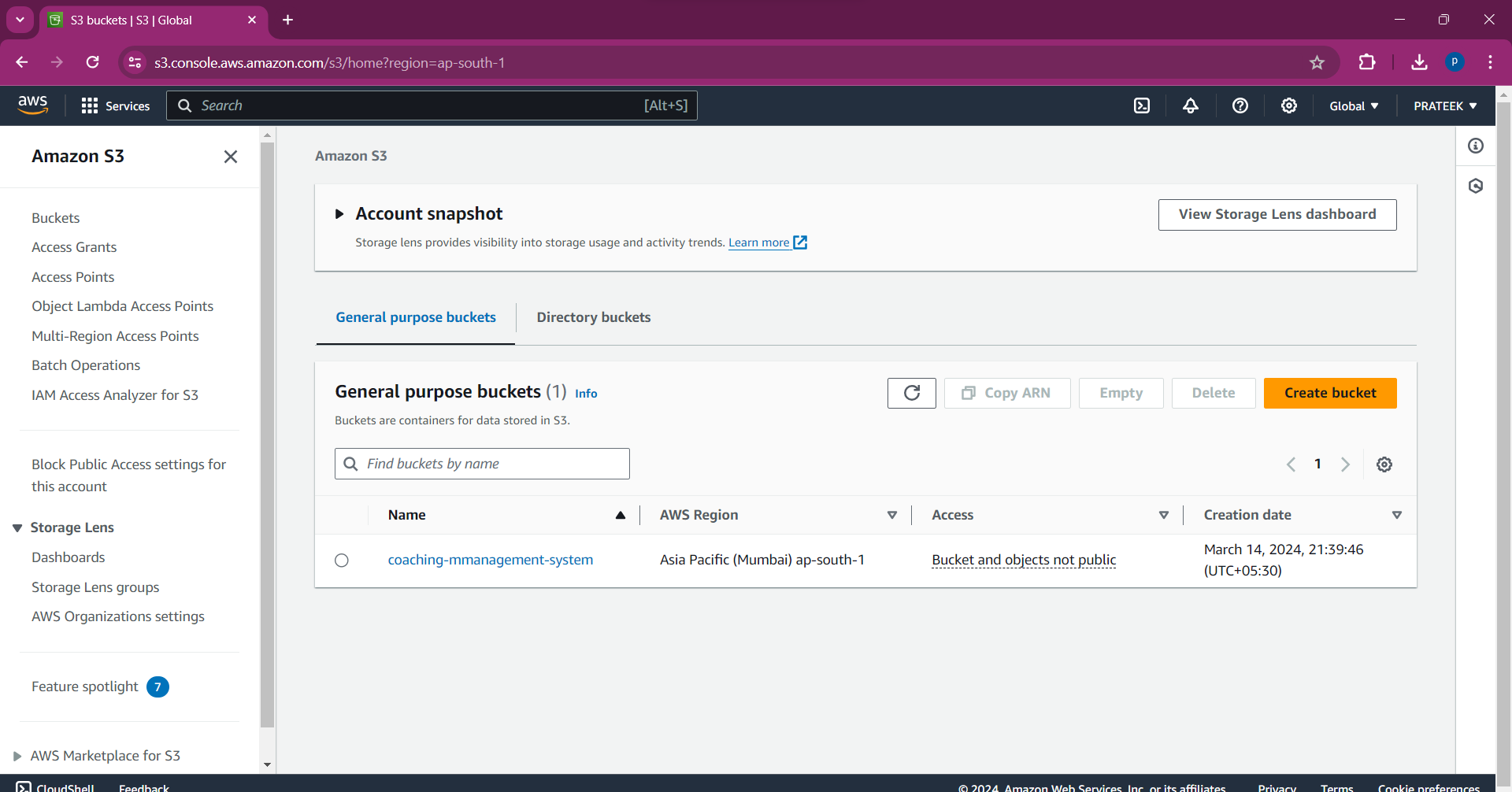
**Step 2: Create Your First S3 Bucket**

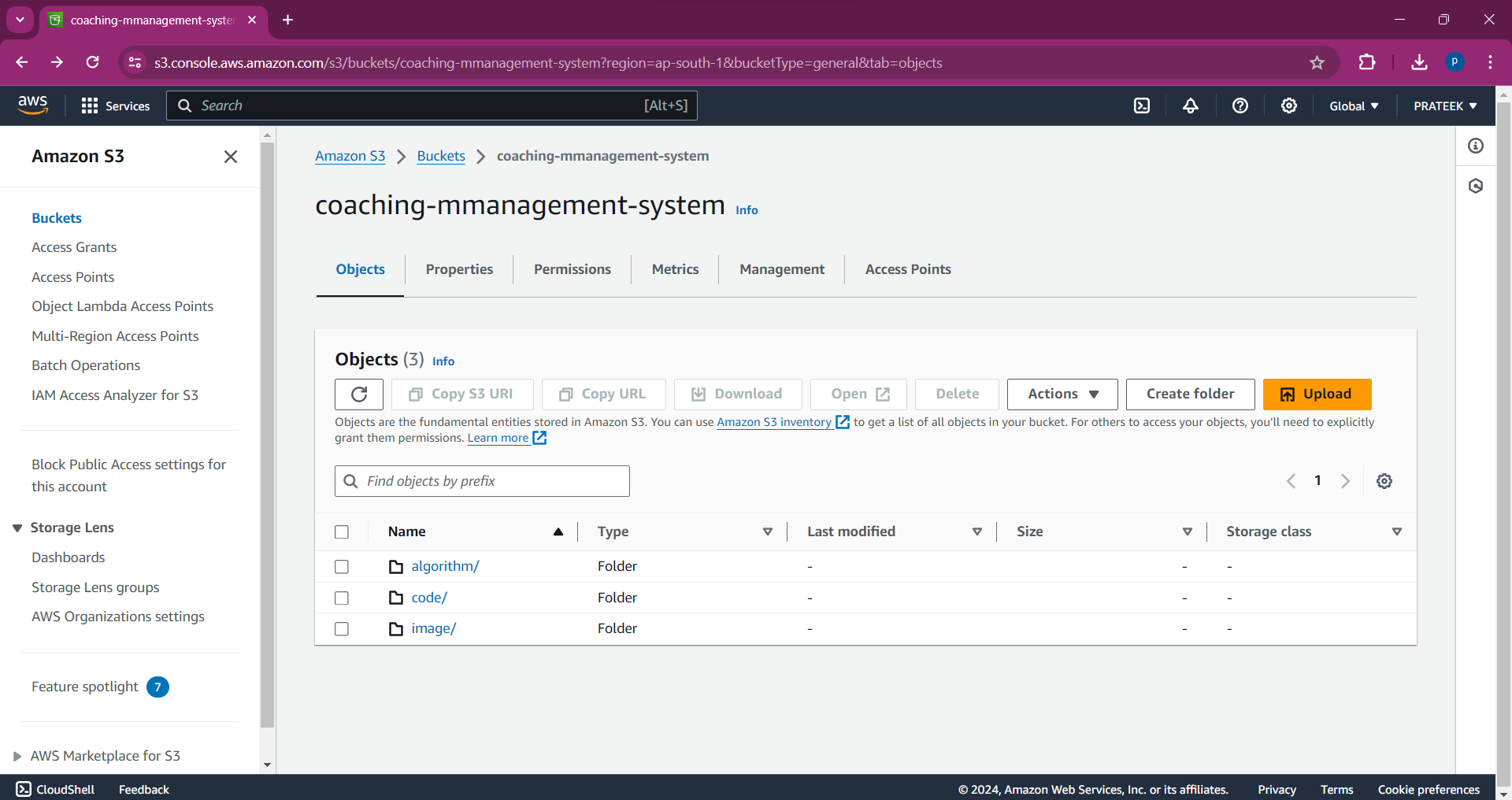
1. In the search bar at the top of the console, type “**Amazon S3**.”
2. Choose “**Amazon S3**” from the search results.

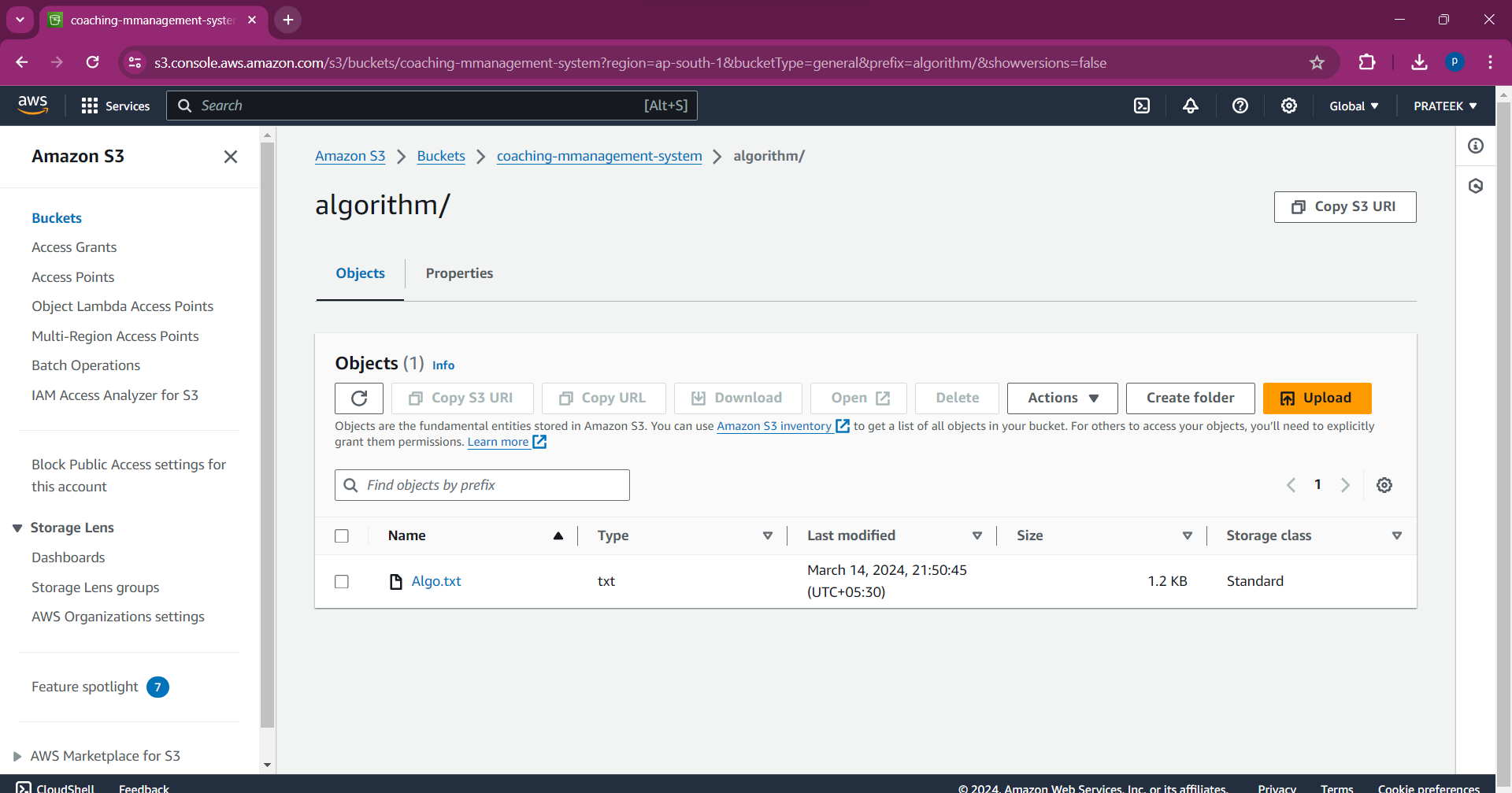
**Step 3: Configure Your Bucket**

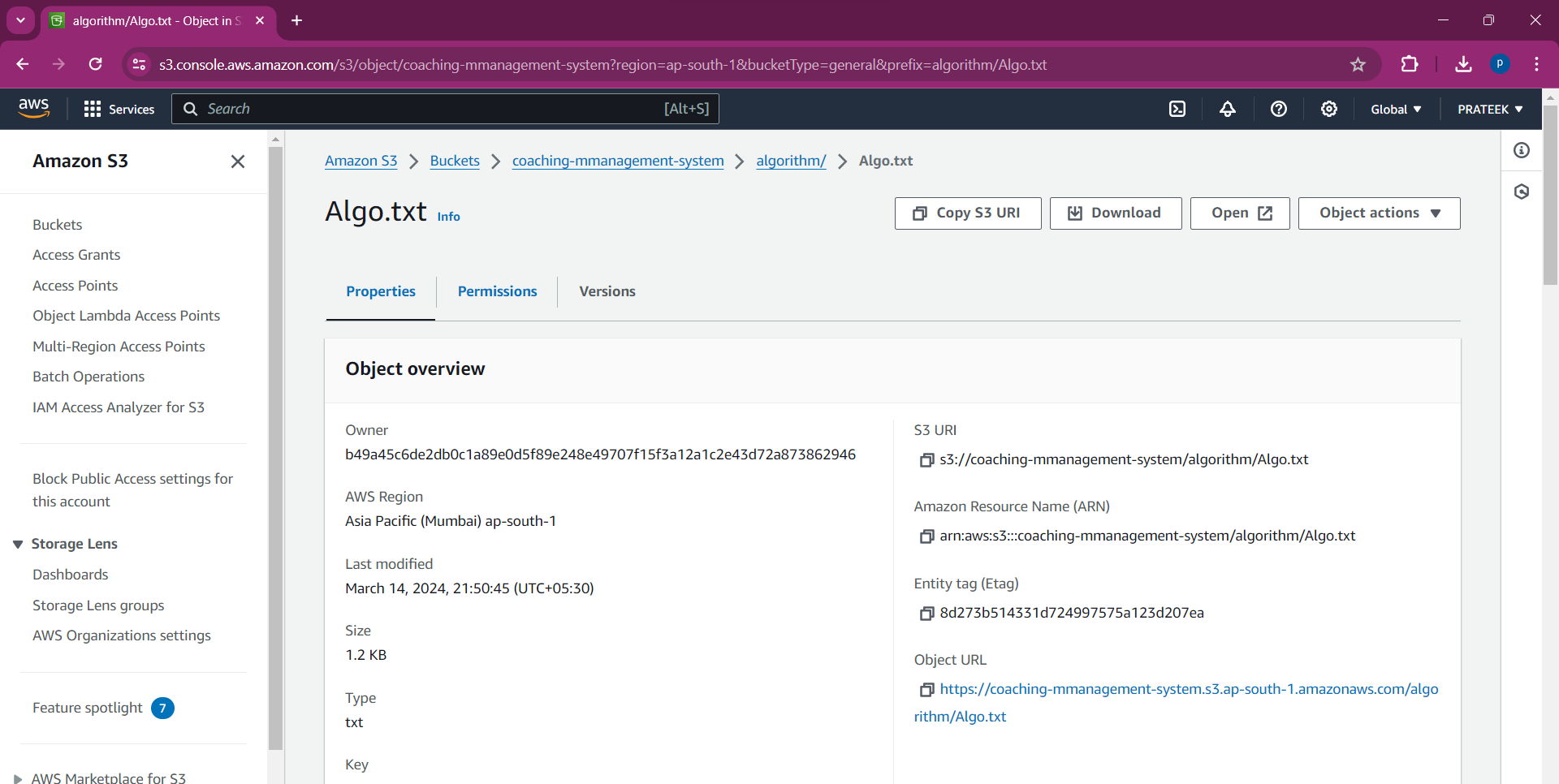
1. On the **Amazon S3 console**, navigate to the **Buckets** section.
2. Click **“Create bucket”**.
3. The **Create bucket** page opens. Here are the details you need to fill in:
   * **Bucket name**: Enter a unique name for your bucket. It must be between **3 and 63 characters long**, consist only of lowercase letters, numbers, dots (.), and hyphens (-). Avoid using dots (.) in bucket names unless it’s for static website hosting. Once created, you can’t change the bucket name.
   * **Region**: Choose the AWS Region where you want the bucket to reside. Select a region close to you to minimize latency and costs.
   * **Object Ownership**:
     + **ACLs disabled**: By default, ACLs are disabled. The bucket owner automatically owns and has full control over every object in the bucket. ACLs no longer affect access permissions.
     + **Bucket owner enforced (default)**: The bucket owner exclusively controls object ownership using policies. This is recommended for most use cases.
   * **Important**: Avoid including sensitive information (like account numbers) in the bucket name, as it’s visible in the URLs pointing to objects in the bucket.

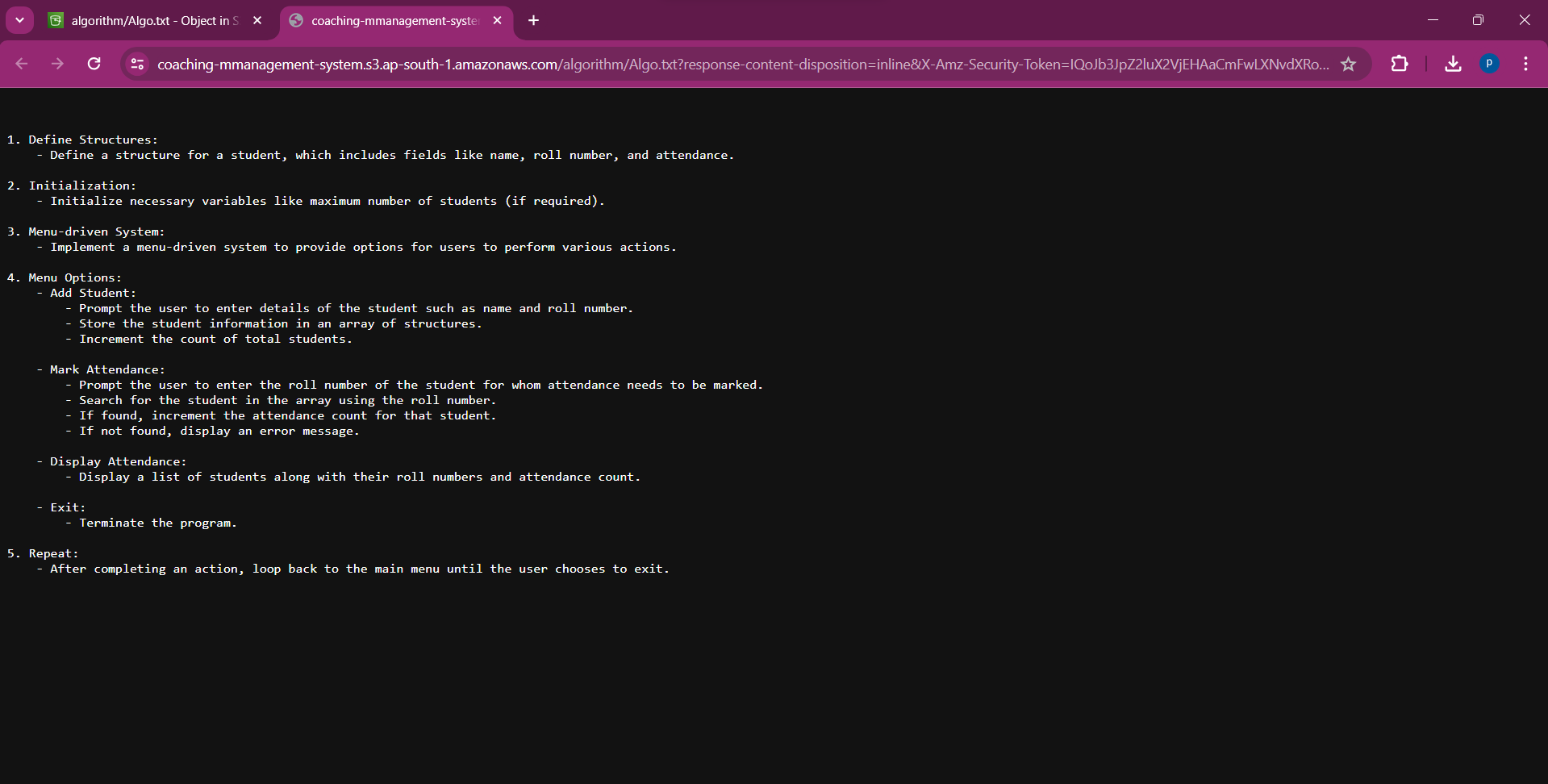


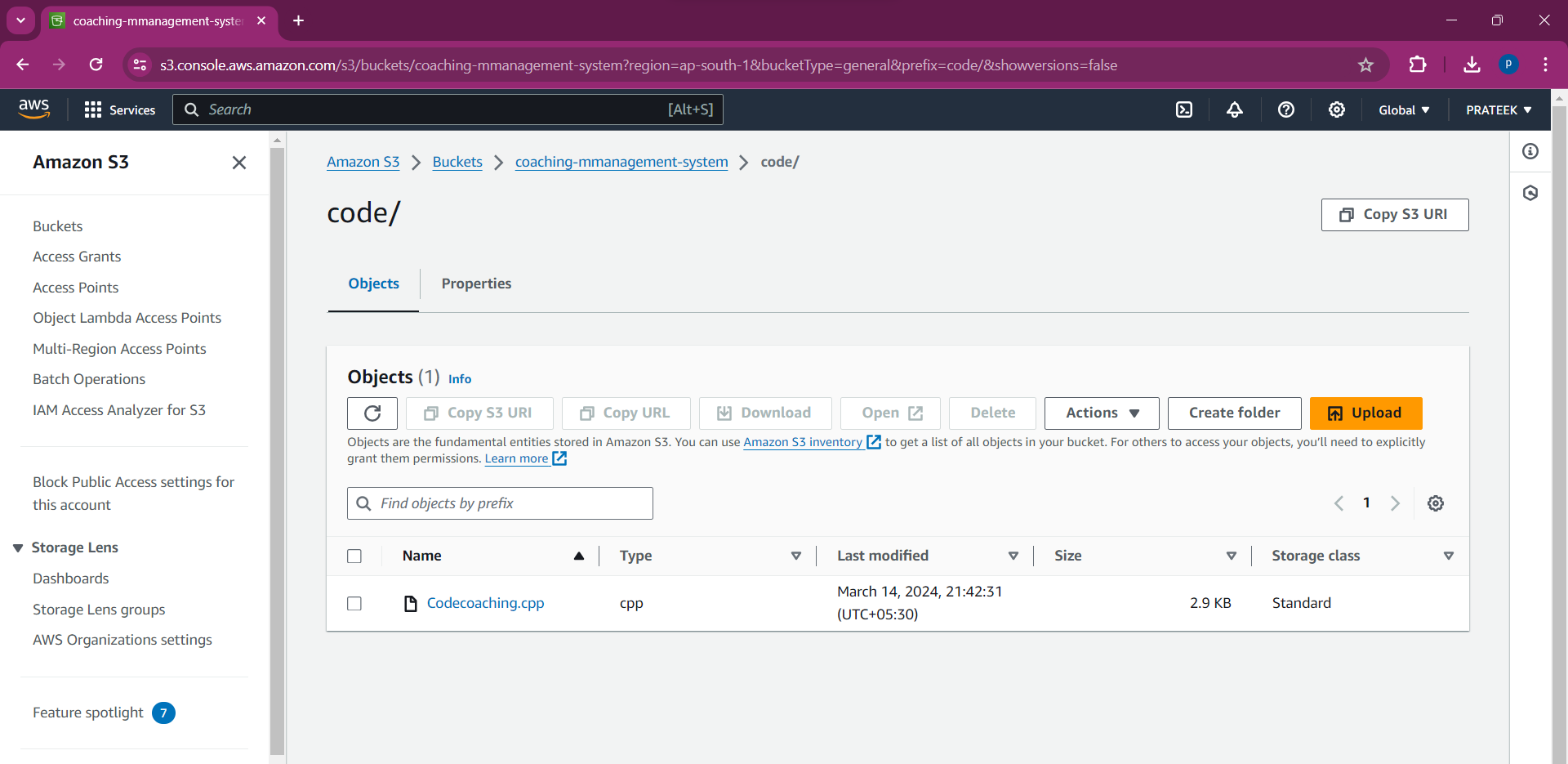


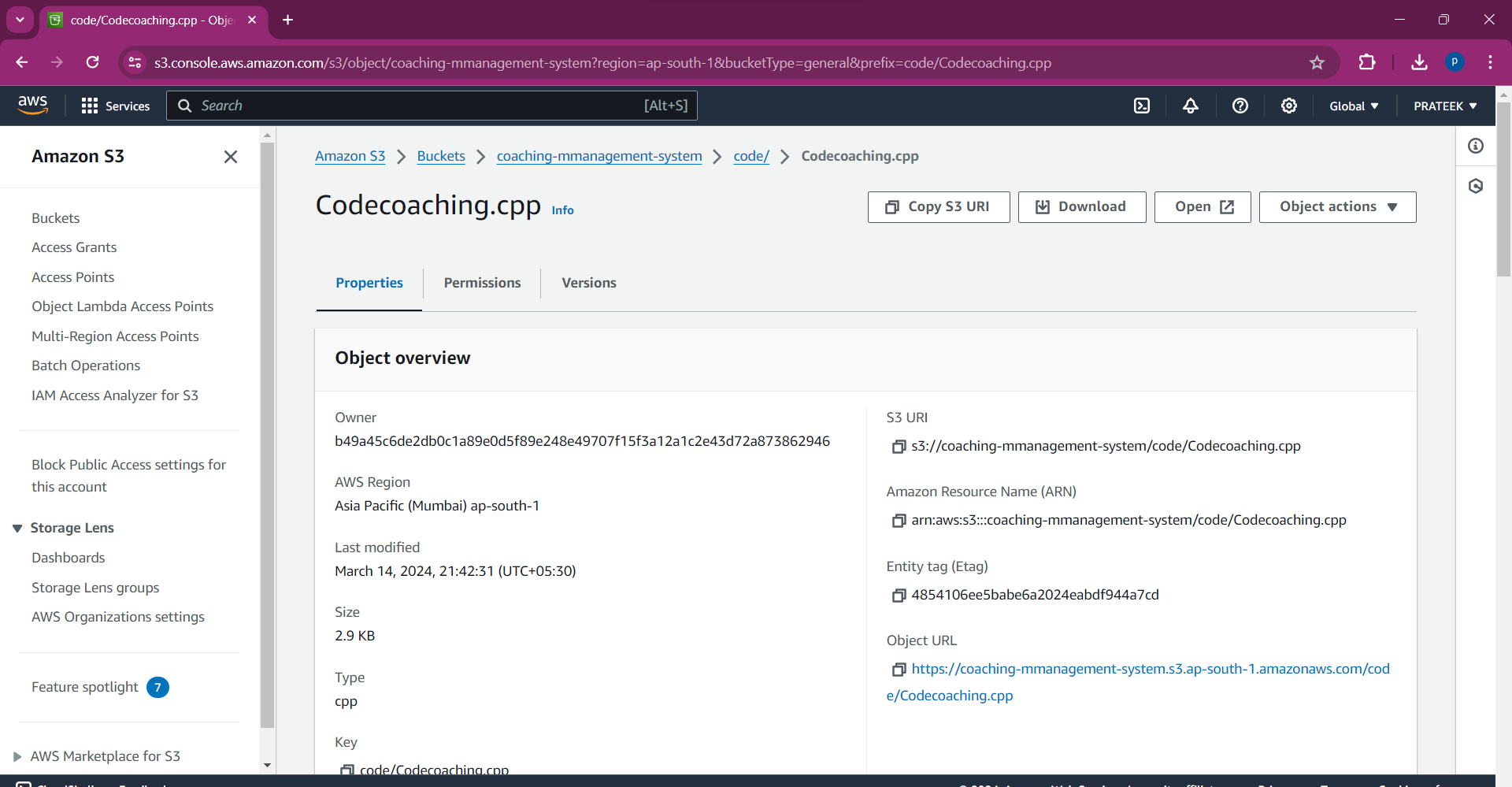


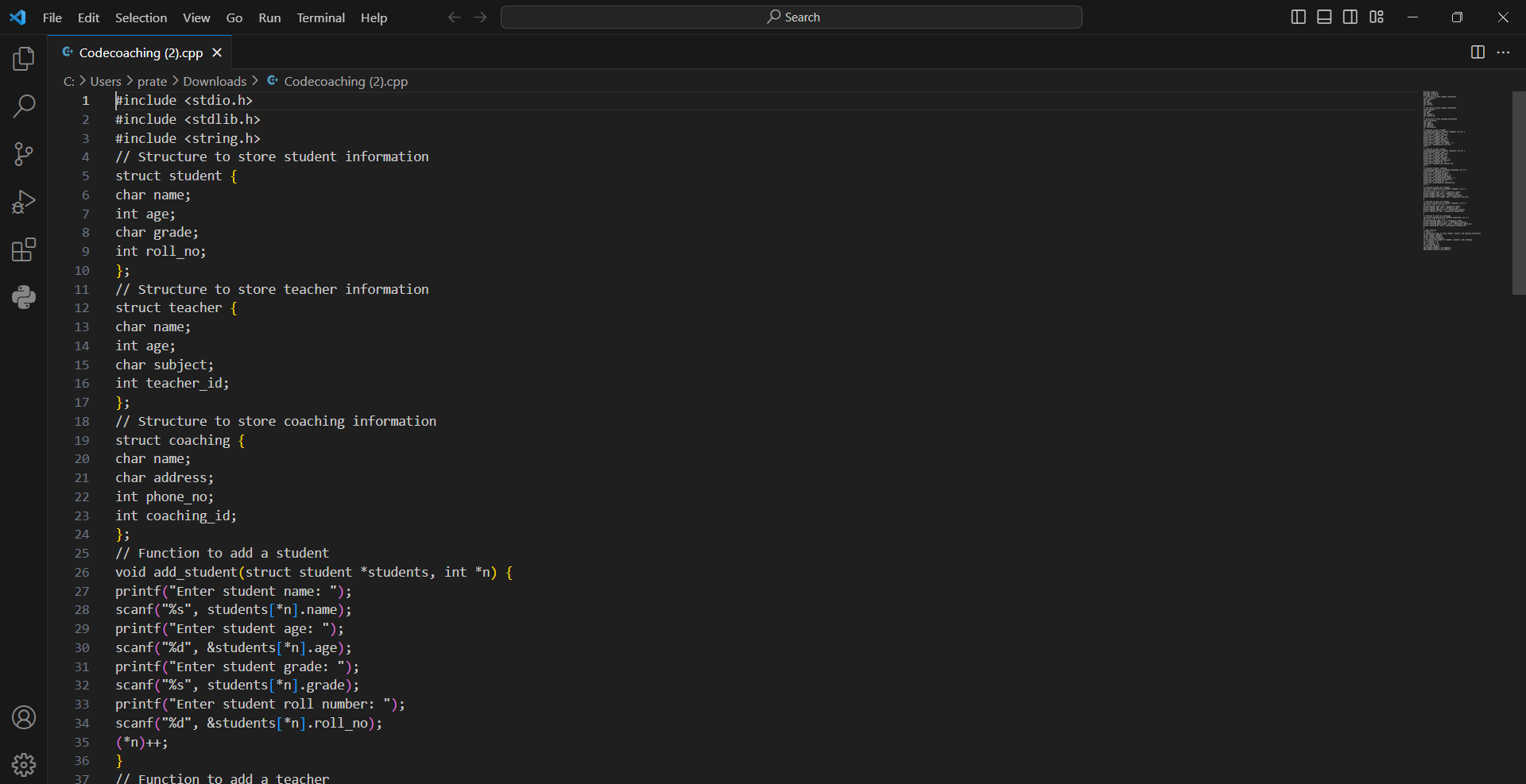


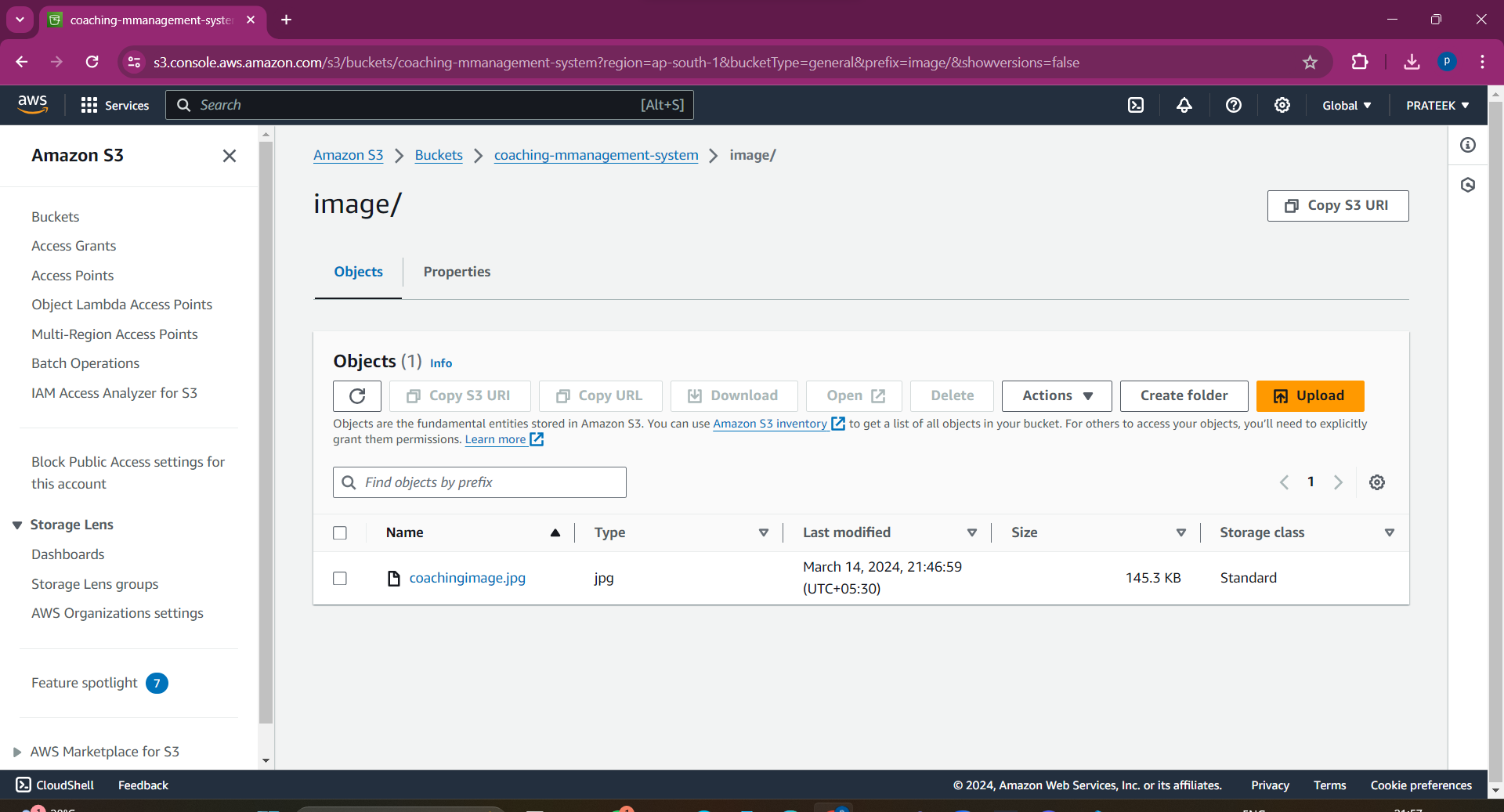


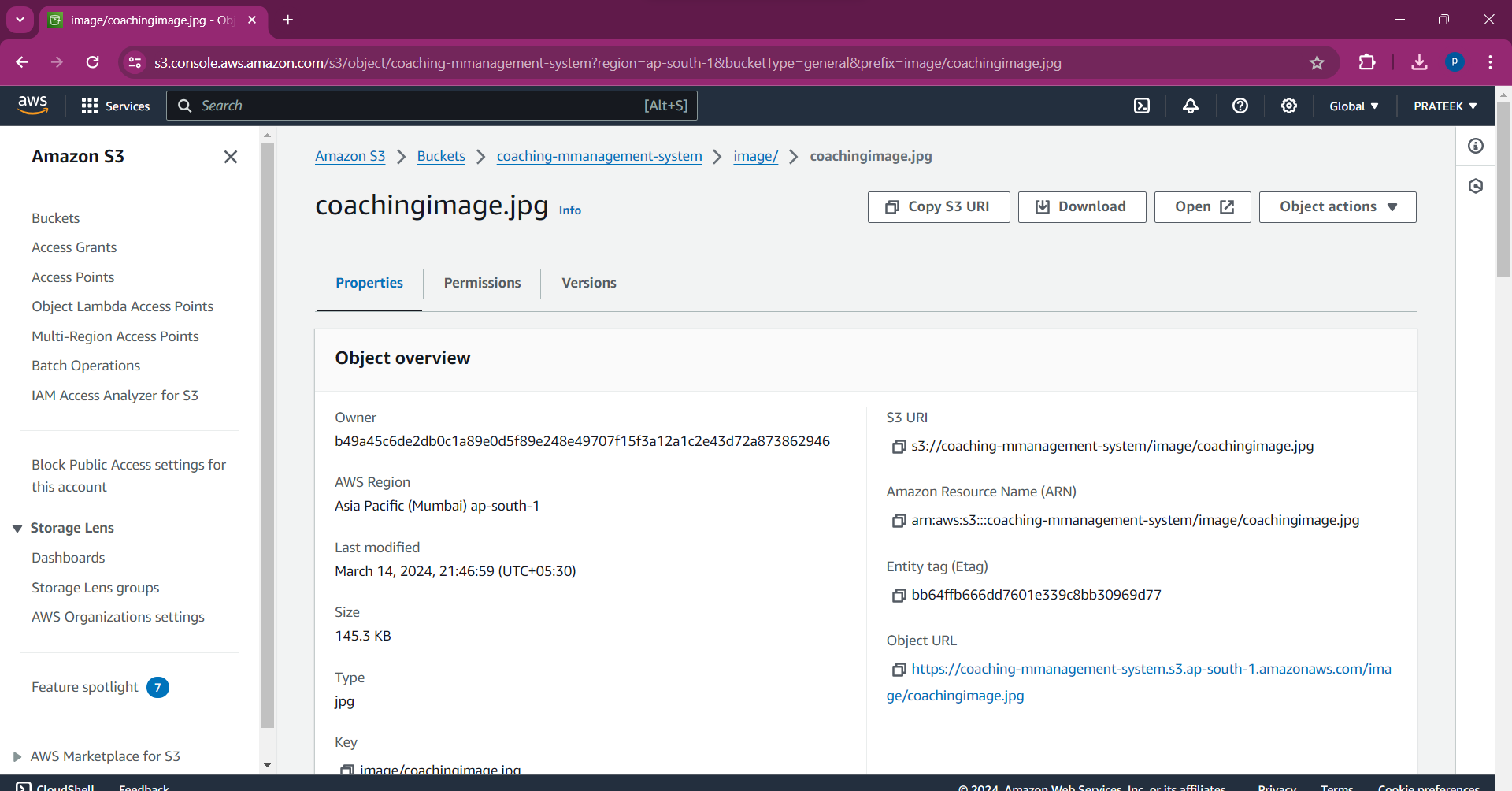


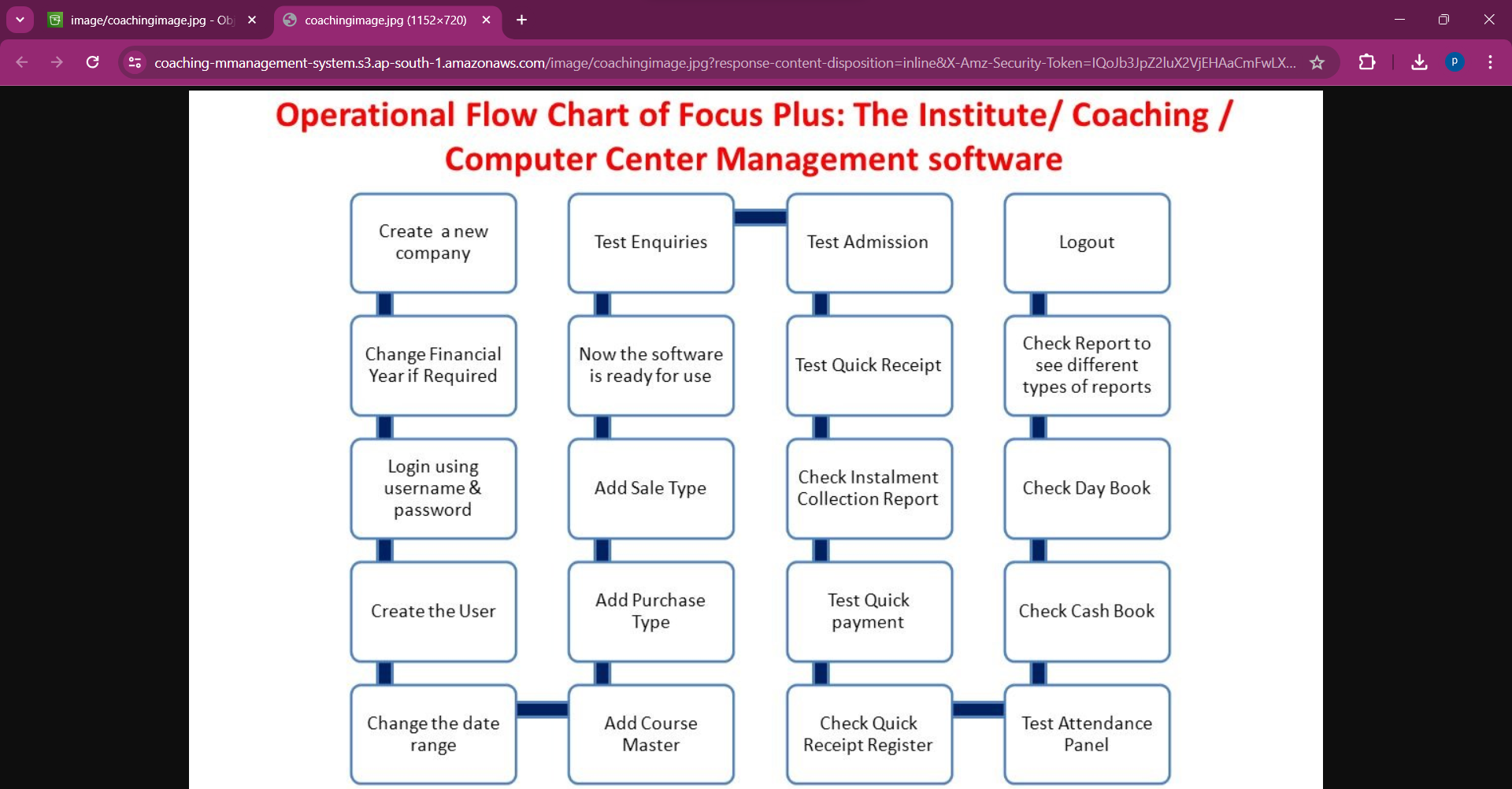












**6. CREATION OF IAM IN AWS**

To create an IAM (Identity and Access Management) user in AWS, you typically follow these steps:

1. Sign in to the AWS Management Console: Go to the AWS Management Console (https://console.aws.amazon.com/) and sign in with your AWS account credentials.

2. Navigate to IAM: Once logged in, navigate to the IAM service. You can find it by typing "IAM" in the search bar or locating it under the "Security, Identity, & Compliance" section.

3. Access the Users Dashboard: In the IAM dashboard, locate the "Users" option in the left-hand menu and click on it to access the Users dashboard.

4. Create a New User: Click on the "Add user" button to start the process of creating a new IAM user.

5. Enter User Details: Provide a username for the new IAM user. You can also choose whether the user will have programmatic access (access to AWS CLI, SDKs, and APIs) and/or AWS Management Console access (access to the AWS Management Console).

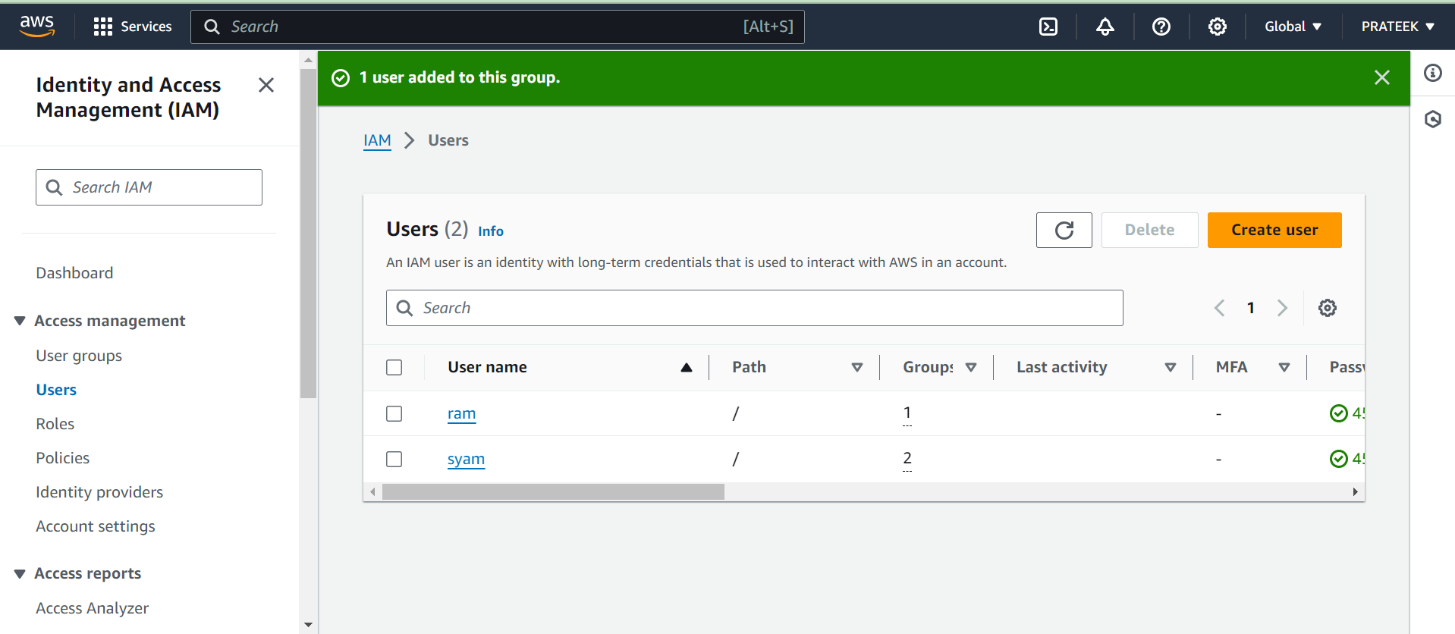
6. Set Permissions: Next, you'll need to attach permissions policies to the user. You can choose to add the user to existing groups with predefined permissions or attach policies directly to the user. Policies define what actions the user can perform on which AWS resources.

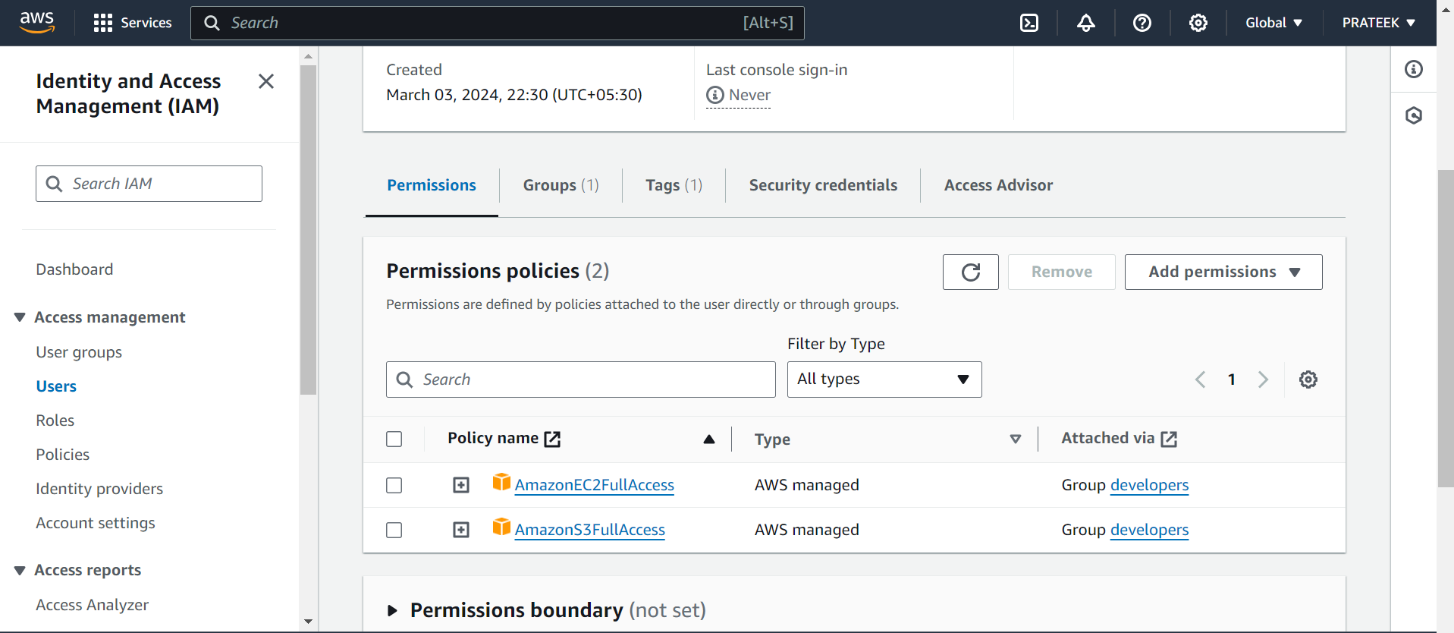
7. Review and Create: Review the user details and permissions to ensure they are correct. Once you are satisfied, click on the "Create user" button to create the IAM user.

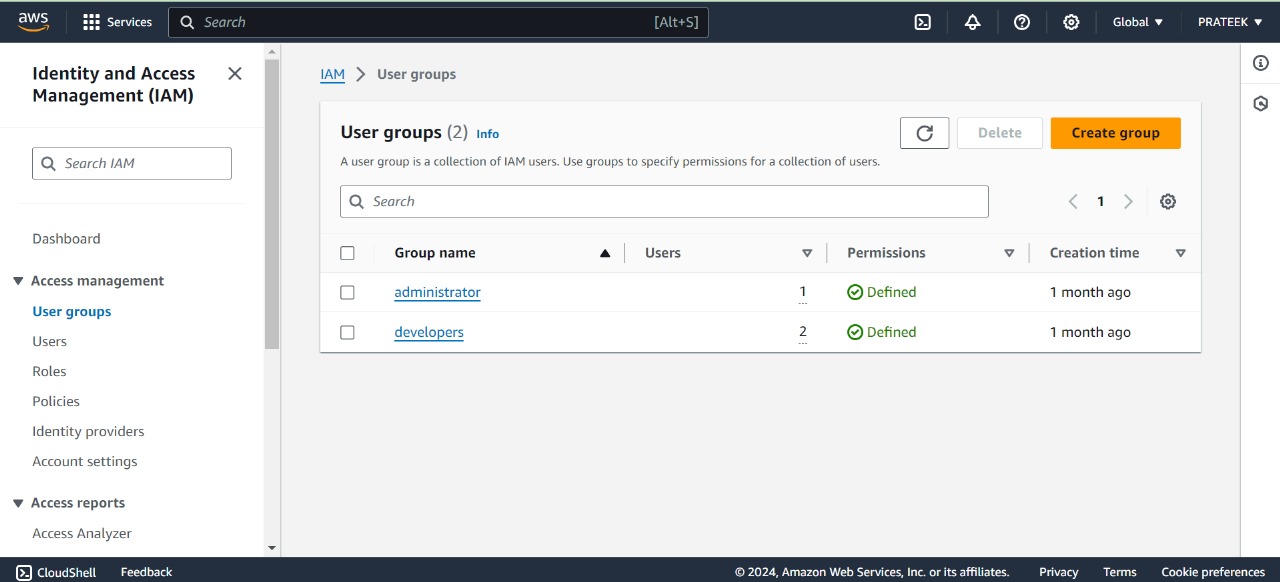
8. Access Key and Secret Access Key: If you've chosen to provide programmatic access, you will be shown the user's Access Key ID and Secret Access Key. These credentials are used to authenticate programmatic access to AWS services and should be securely stored.

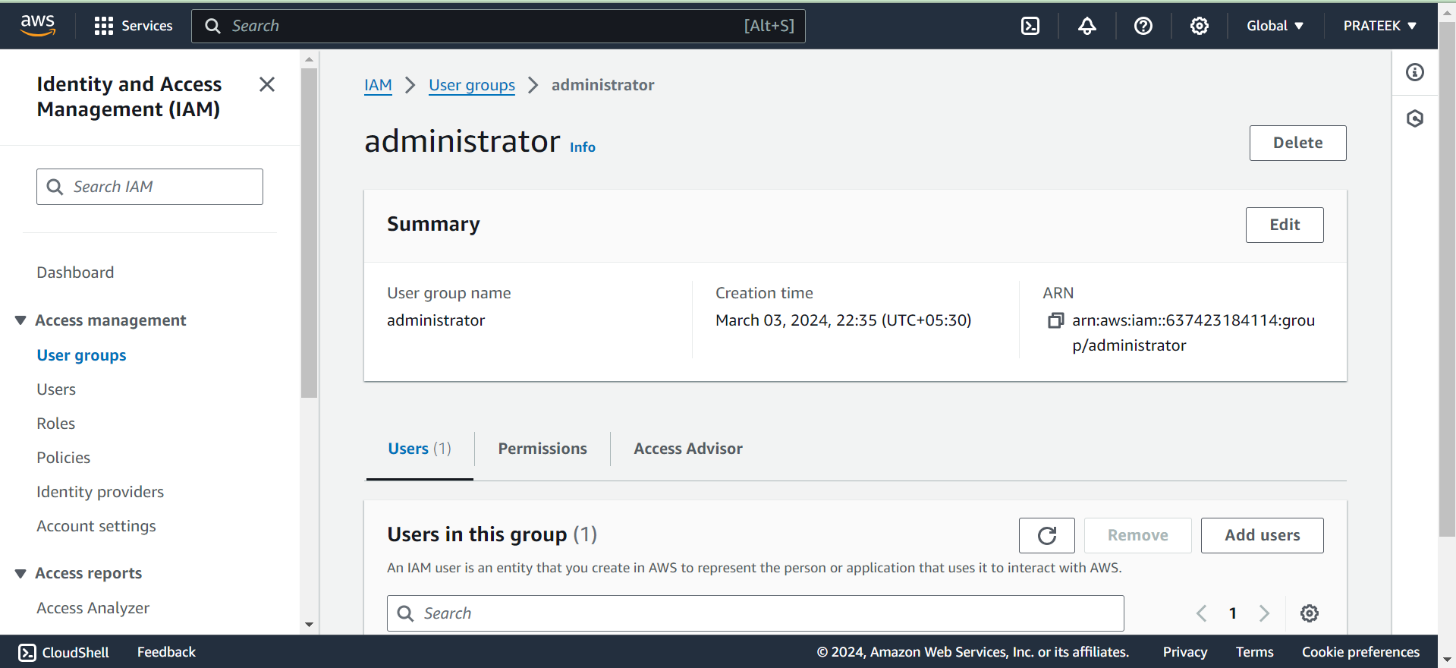
9. Notify User: If necessary, you can send the IAM user's login information (such as username, password, and login URL) to the user via email. This step is optional and depends on your organization's policies.

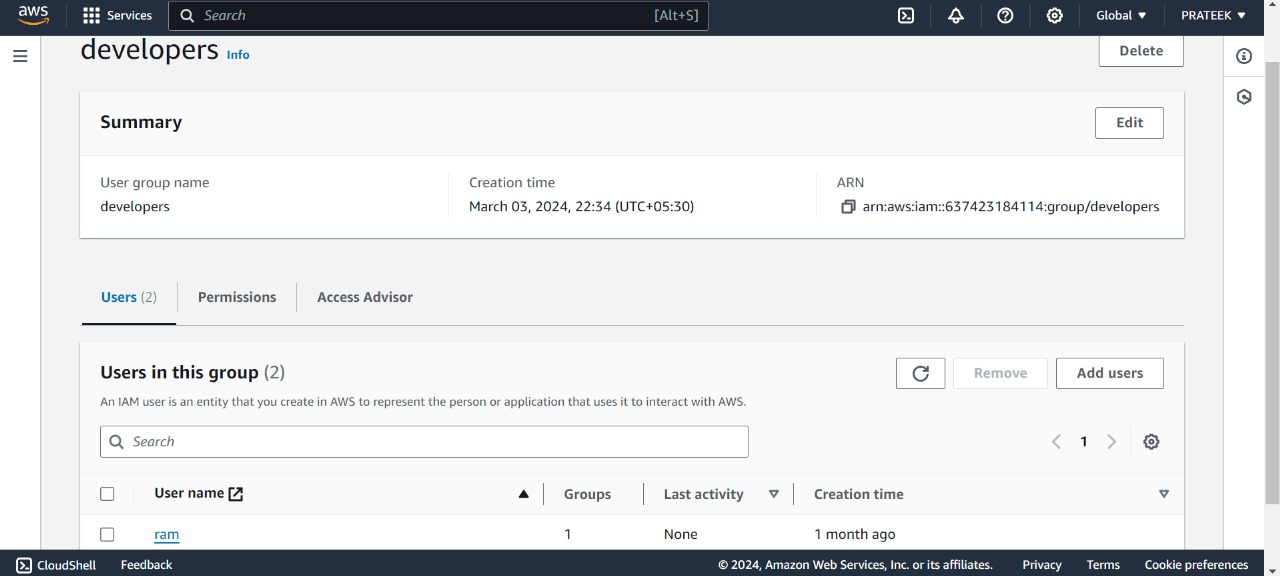
By following these steps, you can successfully create an IAM user in AWS and grant them the necessary permissions to access AWS resources according to your organization's requirements. Remember to follow best practices for security, such as regularly reviewing permissions and rotating access keys.

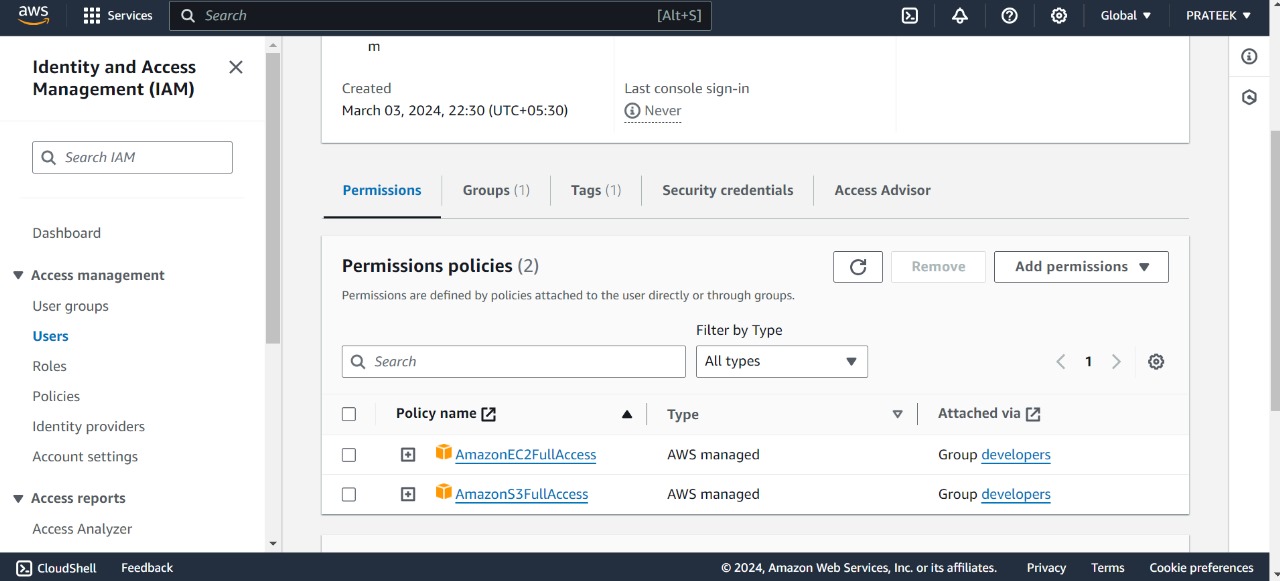


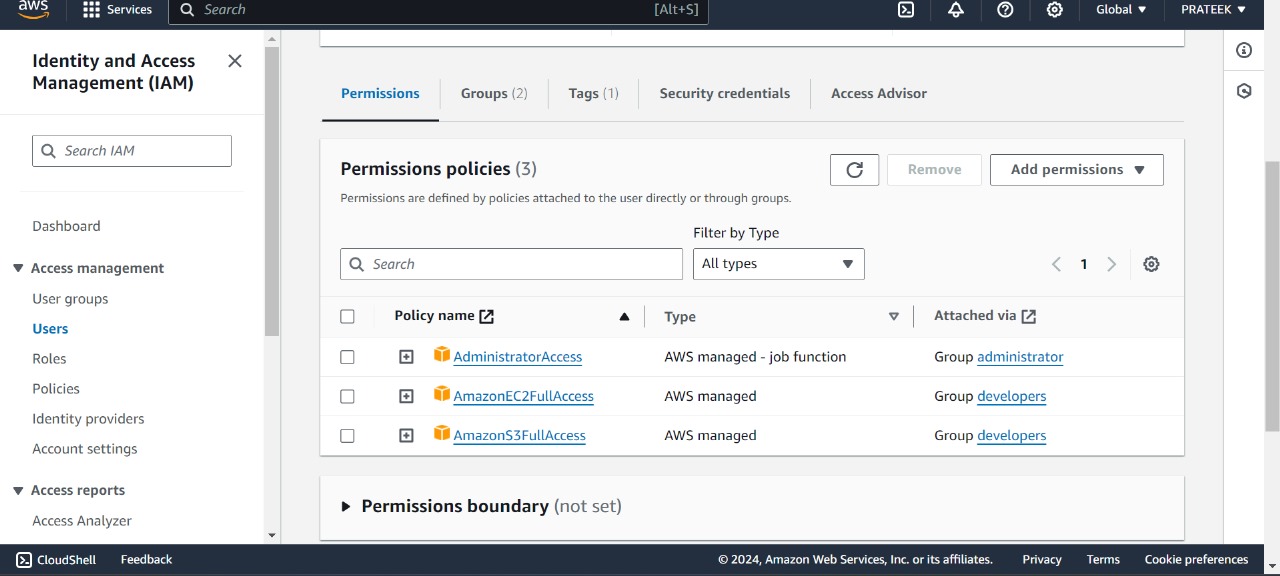


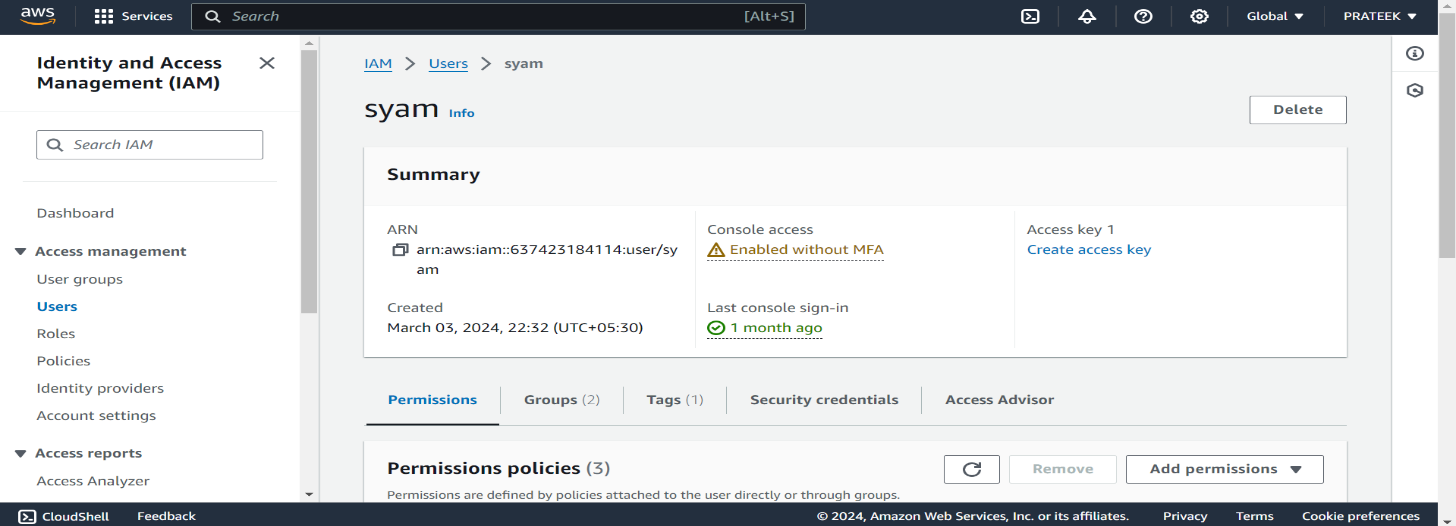












**7.IMPLEMENTATION**

1. Project Scope and Timeline:

- Define the scope of the implementation, including key features and functionalities to be developed.

- Break down the project into manageable tasks and milestones, with estimated durations for each.

- Create a timeline for the implementation, outlining start and end dates for each phase of development.

2. Resource Allocation:

- Assign roles and responsibilities to team members based on their skills and expertise.

- Identify any additional resources or external vendors needed for specialized tasks or services.

- Ensure adequate training and onboarding for team members involved in the implementation.

3. Development Methodology:

- Select a suitable development methodology for the project, such as Agile, Scrum, or Kanban.

- Establish sprint cycles, daily stand-up meetings, and other Agile practices to facilitate iterative development and collaboration.

- Set up project management tools and communication channels to support the chosen methodology.

4. Technical Architecture:

- Set up development environments for the frontend and backend components of the CMS application.

- Install necessary software and dependencies, including code editors, version control systems, and development frameworks.

- Configure development, staging, and production environments to ensure consistency and reliability across different stages of the implementation.

5. Database Setup and Configuration:

- Choose and deploy the appropriate database solution (MongoDB or MySQL) for storing application data.

- Define database schemas and tables based on the data model designed during the application infrastructure phase.

- Set up database users, permissions, and access controls to ensure security and data integrity.

6. Backend Development:

- Begin development of backend APIs and services using Node.js and Express.js.

- Implement authentication and authorization mechanisms, including JWT-based authentication and role-based access control.

- Build endpoints for handling CRUD operations, user management, session scheduling, and communication features.

7. Frontend Development:

- Develop responsive and interactive user interfaces using HTML5, CSS3, and JavaScript frameworks (React.js or Angular).

- Implement client-side routing for navigation between pages and components.

- Integrate with backend APIs to fetch and display dynamic content, manage user sessions, and handle user interactions.

8. Integration and Testing:

- Integrate frontend and backend components to create a fully functional CMS application.

- Conduct unit tests, integration tests, and end-to-end tests to ensure the reliability and performance of the application.

- Use testing frameworks and tools such as Jest, Mocha, and Selenium for automated testing and quality assurance.

9. Deployment and Rollout:

- Deploy the CMS application to staging environments for initial testing and validation.

- Conduct user acceptance testing (UAT) to gather feedback and identify any issues or bugs.

- Once approved, deploy the application to production environments using automated deployment pipelines or manual deployment processes.

10. Training and Support:

- Provide training sessions for administrators, coaches, and clients on how to use the CMS application effectively.

- Develop user documentation, tutorials, and FAQs to support self-service and troubleshooting.

- Establish channels for ongoing technical support and feedback, such as help desks, forums, or ticketing systems.

11. Monitoring and Maintenance:

- Set up monitoring and alerting systems to track application performance, uptime, and error rates.

- Establish procedures for proactive monitoring and maintenance, including regular software updates, security patches, and database backups.

- Continuously monitor user feedback and usage metrics to identify areas for improvement and future enhancements. **8.CONCLUSION**

Implementing a coaching management system on AWS creates a robust, secure, and scalable solution for managing your coaching programs. Here's how key AWS services contribute:

Amazon S3: Provides secure and scalable storage for a variety of coaching system data, including student/coach information, learning materials, and session recordings (if applicable). S3 buckets enable organized data storage with controlled access. You can grant coaches access to specific student data folders and allow students to access their assigned course materials.

IAM (Identity and Access Management): Functions as the security layer of your coaching system. IAM authenticates users (students, coaches, administrators) and defines their permissions. Students can view their profiles and course materials. Coaches can manage assigned students, upload materials, and access recordings (if applicable). Administrators have full access for system management. MFA and role-based access control (RBAC) further enhance security.

This combined functionality empowers your coaching management system with:

Security: Encrypted data storage, controlled access through IAM, and robust user authentication ensure a secure environment.

Scalability: S3 and IAM can easily scale to accommodate a growing number of users and data within the coaching system.

Efficiency: Streamlined data storage, sharing, and access management through S3 and IAM improve website efficiency.

By leveraging AWS's secure, scalable, and cost-effective cloud services, you can create a coaching management system that empowers both coaches and students, fostering a thriving learning environment.

**9.REFERENCES**

1. Online Learning Platforms:

- Platforms like Coursera, Udemy, and Khan Academy offer insights into the design, features, and functionalities of online learning systems.

- Website: [Coursera](https://www.coursera.org/), [Udemy](https://www.udemy.com/), [Khan Academy](https://www.khanacademy.org/)

2. Software Development Resources:

- Online resources such as Stack Overflow, GitHub, and Medium provide a wealth of information on web development technologies, frameworks, and best practices.

- Website: [Stack Overflow](https://stackoverflow.com/), [GitHub](https://github.com/), [Medium](https://medium.com/)

3. Technical Documentation and Tutorials:

- Official documentation and tutorials for technologies used in the project, such as Node.js, React.js, MongoDB, and MySQL, offer detailed guidance on development and implementation.

- Website: [Node.js Documentation](https://nodejs.org/en/docs/), [React.js Documentation](https://reactjs.org/docs/getting-started.html), [MongoDB Documentation](https://docs.mongodb.com/), [MySQL Documentation](https://dev.mysql.com/doc/)

4. Online Courses and Tutorials:

- Online courses and tutorials on platforms like Udemy, Pluralsight, and Codecademy provide in-depth learning opportunities for web development, database management, and application deployment.

- Website: [Udemy - Web Development Courses](https://www.udemy.com/courses/development/web-development/), [Pluralsight - Web Development Paths](https://www.pluralsight.com/paths/web-development), [Codecademy - Full-Stack Engineer Path](https://www.codecademy.com/learn/paths/full-stack-engineer-career-path)

5. Books and eBooks:

- Books such as "Learning Node.js" by Marc Wandschneider, "React.js Essentials" by Artemij Fedosejev, and "MongoDB: The Definitive Guide" by Kristina Chodorow offer comprehensive insights into relevant technologies and frameworks.

- Websites: [Learning Node.js](https://www.amazon.com/Learning-Node-js-Development-Node/dp/1491943122), [React.js Essentials](https://www.amazon.com/React-js-Essentials-Artemij-Fedosejev-ebook/dp/B00ZCI7LOK), [MongoDB: The Definitive Guide](https://www.amazon.com/MongoDB-Definitive-Guide-Powerful-Scalable/dp/1449344682)