BCSE355L-Cloud Architecture Design-Project

JusLearn: A Cloud-Based E-Learning Platform Leveraging Core AWS Services.

Submitted in partial fulfillment of the requirements for the Internals in Course Project

in

B. Tech [All Branches]

by

23BCI0126 RADHANJAN NEELAMRAJU

23BCI0118 RAMASANI VISHWAS SAI

Under the Supervision of

Dr.Yoganand S.

Assistant Professor (Sr.Grade-1)

School of Computer Science and Engineering (SCOPE)

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

November 2025

DECLARATION

I hereby declare that the project entitled JusLearn: A Cloud-Based E-Learning Platform Leveraging Core AWS Services. submitted by me, for the award of the degree of *Bachelor of Technology in Computer Science and Engineering* to VIT is a record of bonafide work carried out by me under the supervision of Prof. / Dr. Yoganand S.

I further declare that the work reported in this project has not been submitted and will not be submitted, either in part or in full, for the award of any other degree ordiploma in this institute or any other institute or university.

Place : Vellore

Date : 28/10/2025

Signature of the Candidate

CERTIFICATE

This is to certify that the project entitled JusLearn: A Cloud-Based E-Learning Platform Leveraging Core AWS Services submitted by Radhanjan Neelamraju (23BCI0126) **School of Computer Science and Engineering**, VIT, for the award of the degree of *Bachelor of Technology in Computer Science and Engineering*, is a record of bonafide work carried out by him / her under my supervision during Fall Semester 2024-2025, as per the VIT code of academic and research ethics.

The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma in this institute orany other institute or university. The project fulfills the requirements and regulations ofthe University and in my opinion meets the necessary standards for submission.

Place : Vellore

Date : 28/10/2025

Signature of the Guide

CERTIFICATE

This is to certify that the project entitled JusLearn: A Cloud-Based E-Learning Platform Leveraging Core AWS Services submitted by Ramasani Vishwas Sai (23BCI0118) **School of Computer Science and Engineering**, VIT, for the award of the degree of *Bachelor of Technology in Computer Science and Engineering*, is a record of bonafide work carried out by him / her under my supervision during Fall Semester 2024-2025, as per the VIT code of academic and research ethics.

The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma in this institute orany other institute or university. The project fulfills the requirements and regulations ofthe University and in my opinion meets the necessary standards for submission.

Place : Vellore

Date : 28/10/2025

Signature of the Guide

TABLE OF CONTENTS

ABSTRACT

JusLearn is a prototype web application demonstrating a fundamental 3-Tier Architecture deployed on Amazon Web Services (AWS). The system allows students to access course content, track topic completion through a centralized database, and upload assignments. The project operates within the E-Learning domain, focusing on delivering accessible resources for Computer Science subjects (Operating Systems, DBMS, and Computer Networks). The primary objective of this project is to showcase the practical integration of core AWS components—specifically Compute (EC2), Storage (S3), Database (RDS - conceptual), and Networking (VPC/Security Groups)—to build a functional, resilient, and database-driven e-learning solution. The core challenge addressed is providing a reliable, scalable, and highly available architecture for hosting video content, static notes, and assignment submission functionality.

CHAPTER 1

1. INTRODUCTION

1.1 Background

The educational sector is undergoing a massive transformation, driven by the explosive global demand for flexible and accessible learning. The E-Learning market is now a multi-billion dollar industry defined by continuous technology integration aimed at maximizing student engagement and learning outcomes.

The modern educational paradigm is defined by a transition from traditional classroom settings toward self-paced, digital learning. Within highly specialized, technical fields—such as Computer Science (encompassing Operating Systems, DBMS, and Computer Networks)—the system's availability and quality are critical competitive factors.

Building a modern platform in this environment demands solutions to core technical and logistical challenges:

Performance and Scalability: As enrollment surges and content libraries expand, the platform must consistently deliver a seamless and engaging experience. Slow load times or inadequate server capacity are primary causes of user frustration and high drop-off rates.

Focusing on User Experience : Accessibility across devices necessitates a mobile-first design. To avoid technical hurdles for remote learners, high uptime and unwavering system reliability are absolute necessities.

Emphasizing Security Components: Protecting user data is essential, requiring robust authentication protocols and the secure, persistent storage of all critical information, including course progress records.

JusLearn's Solution The JusLearn project is explicitly designed to meet these modern demands by leveraging a cloud-native architecture. We propose a robust 3-Tier Architecture built on foundational AWS services. This foundation is designed to reliably host all core functionality—including streaming video, static study notes, and assignment submission—establishing a resilient and secure system that can be cost-effectively scaled using managed cloud infrastructure.

1.2 Motivations

The core motivation for creating JusLearn is to actively address significant shortcomings—both technological and pedagogical—in today's digital learning environment, specifically concerning highly focused Computer Science curricula.

The primary motivations are:

Addressing Accessibility and Centralization: To solve the need for a centralized, accessible hub for learning core Computer Science subjects (Operating Systems, DBMS, and Computer Networks), providing students with a single, reliable source for learning resources.

Focusing on Competence: Our motivation is to establish a prototype that clearly showcases competence in deploying a cloud-native solution by effectively utilizing foundational AWS services like EC2 (Compute), S3 (Storage), RDS (Database), and relevant Networking components. The project directly showcases our proficiency in deploying a cloud-native solution.

Focusing on the Architecture's Capabilities: The goal is to design a reliable, highly available, and scalable architecture that can seamlessly host substantial resources such as video libraries, static documentation, and assignment submission features. The 3-Tier architecture was adopted specifically to improve maintenance and scaling.

Implementing Secure Practices: To integrate modern security principles into the deployment, specifically utilizing bcrypt for robust password hashing to secure user authentication, aligning with modern security standards.

Bridging Educational Gaps: To provide students with tools for tracking topic completion through a centralized database, which enhances self-paced learning and provides clear metrics on progress.

1.3 Scope of the Project

The scope of the JusLearn E-Learning Platform is strictly defined to cover the full implementation of a functional prototype demonstrating a robust 3-Tier Web Architecture utilizing the core services of Amazon Web Services (AWS).

Functional Scope

The platform exclusively provides content for the three foundational Computer Science subjects: Operating Systems, DBMS, and Computer Networks.Functionality includes secure user authentication covering Sign Up and Sign In , dynamic delivery of video content and downloadable static notes for each topic , and assignment submission capabilities. The scope mandates database-driven progress tracking, enabling the UI to visualize topic completion status. The backend must utilize bcrypt for password hashing to ensure secure credential management.

Architectural and Technical Scope

The project's scope is confined to integrating and demonstrating the functionality of core AWS components: Amazon EC2 (Application Layer Compute), Amazon S3 (Presentation and Storage Layer), and VPC/Security Groups (Networking). The architecture is built to use Amazon RDS for data layer resilience in production. To maintain security, inbound traffic will be restricted using Security Groups to the minimum necessary ports: Port 22 (SSH) and Port 5000 (API).The project excludes advanced LMS features like live grading or complex user roles.

CHAPTER 2

2. PROJECT DESCRIPTION AND GOALS

2.1 Literature Review

The contemporary E-Learning environment is dominated by major platforms (e.g., Coursera, EdX, Moodle) that provide extensive content catalogs. While these systems offer robust educational content delivery, the underlying technical architectures often remain proprietary or are deployed across opaque, non-standardized infrastructure. This review focuses on two critical areas: the established practice of secure web application design and the operational benefits of cloud-native deployment.

2.1.1 Architectural Precedent: The 3-Tier Model

The 3-Tier Architecture remains the foundational model for deploying scalable, reliable, and secure enterprise web applications. This model logically separates the application into three independent layers, which are ideally hosted on separate physical or virtual infrastructure:

1. Presentation Tier (Web Layer): Handles the user interface and serves content (HTML, CSS, JavaScript). It receives user input.
2. Application Tier (Logic Layer): Contains the business logic, processing user requests, and managing communication between the Presentation and Data Tiers. This isolation is crucial for security.
3. Data Tier (Database Layer): Stores persistent application data, accessible only by the Application Tier.

This separation offers distinct advantages essential to the JusLearn project, including independent scalability (e.g., scaling the application servers without touching the database), enhanced reliability through redundancy, and a significant improvement in security posture by isolating the critical data layer from direct public access.

2.1.2 Cloud Infrastructure and Security Literature

Literature confirms that utilizing managed cloud services simplifies compliance and scalability. AWS best practices, specifically the Well-Architected Framework, emphasize the use of native security controls like Security Groups and Virtual Private Clouds (VPCs) to enforce the principle of least privilege. Deploying the Data Tier using Amazon RDS is the standard for high-performance, fault-tolerant relational databases, as it abstracts complex administrative tasks like patching, backups, and failover across Availability Zones (AZs). The systematic adoption of this managed, tiered approach provides a verifiable model for cloud competency that is superior to custom, monolithic deployments.

2.2 Research Gap

The review confirms a distinct practice-to-theory gap in the public documentation of secure, managed cloud architectures for specialized, high-availability technical subject matter.

2.2.1 The Architectural Gap

The core architectural gap is two-fold: existing solutions either lack robust, auditable technical deployment or employ overly complex microservices that sacrifice structural simplicity and cost-effectiveness for unnecessary hyper-scale.

JusLearn directly addresses this by establishing a verifiable cloud reference implementation. This prototype validates that a standard 3-Tier pattern utilizing core AWS managed services can achieve high availability, strict security isolation, and optimized operational costs suitable for focused educational content delivery.

2.2.2 Project Scope and Constraints

The JusLearn project is constrained by the following defined boundaries:

The Content Scope is strictly limited to three foundational Computer Science subjects: Operating Systems, DBMS, and Computer Networks. The project explicitly excludes additional subjects, live video conferencing, or interactive simulations.

The Architecture is defined as a Standard 3-Tier model leveraging AWS managed services, specifically excluding complex microservices or reliance on unmanaged infrastructure. Core AWS Services utilized include EC2 (Compute), S3 (Storage), RDS (Database), and VPC/Security Groups (Networking); advanced services like AWS Lambda or DynamoDB are constrained to maintain simplicity.

In terms of Security, the scope includes bcrypt hashing for authentication and strict Security Group firewalling (Port 22, Port 5000 only), with advanced features like Web Application Firewalls (WAF) or Identity Federation (OIDC) deferred outside the V1 scope.

Finally, Functionality encompasses user authentication, static content delivery, progress tracking via database, and assignment submission (using S3 file upload). Real-time chat, in-platform grading, or automated code execution are explicitly excluded.

2.3 Project Plan

Given the project's goal of creating a functional, testable prototype with well-defined constraints, an Iterative Development Approach has been selected. This approach facilitates continuous integration, rapid testing of functional components, and organized adaptation without compromising the fixed architectural goals, organizing work into distinct implementation phases.

2.3.1 Implementation Phases

The project will be divided into three distinct Phases, each culminating in a functional, deliverable milestone:

Phase 1: Frontend and User Interface Development

Goal: Create the complete, functional user interface and presentation layer.

Tasks: Develop the responsive HTML/CSS/JavaScript structure; implement all client-side logic and validation; establish the static structure for content display.

Deliverable: A complete, navigable client-side application prototype.

Phase 2: Backend Logic and Data Layer Integration

Goal: Implement the core application features, authentication, and database connectivity.

Tasks: Develop the Python/Flask application logic; implement user registration/login using secure password hashing; configure the application to connect to Amazon RDS; implement the database schema for user profiles and progress tracking.

Deliverable: Functional login/logout capability and persistent user state with the database.

Phase 3: Cloud Deployment and Infrastructure Hardening

Goal: Deploy the application components onto AWS infrastructure and finalize security controls.

Tasks: Provision the VPC, subnets, and essential Security Groups (EC2, RDS); deploy the EC2 instance (Application Tier) and Amazon RDS instance (Data Tier) into the configured network; configure Amazon S3 for storing and retrieving static content and assignment files; conduct final performance and security testing.

Deliverable: A Minimum Viable Product (MVP) of the JusLearn platform fully deployed on AWS, ready for final security audit.

CHAPTER 3

3. TECHNICAL SPECIFICATION

3.1 Requirements

The system must satisfy both the essential actions it needs to perform (Functional Requirements) and the quality attributes it must maintain (Non-Functional Requirements) to meet the goals established in the Project Motivation and Scope.

3.1.1 Functional Requirements

The JusLearn platform must support the following core user actions and system processes:

User Authentication: The system must allow users to register with a unique username and password and securely log in to access restricted course materials.

Secure Password Management: The system must securely store user passwords using a secure password hashing algorithm (e.g., a modern salted, adaptive function).

Course Content Access: The system must present static notes and links to video content for the three defined Computer Science subjects (OS, DBMS, CN).

Assignment Submission: The system must allow authenticated users to upload assignment files via the web interface. These files must be securely stored in Amazon S3.

Progress Tracking: The system must allow users to mark topics as complete, with this progress status being reliably recorded and retrieved from the Amazon RDS database.

Session Management: The system must maintain a secure, persistent user session for the duration of the user's activity.

3.1.2 Non-Functional Requirements

The JusLearn platform must adhere to the following quality attributes, primarily driven by the cloud-native architecture:

Reliability and Availability: The platform must maintain a minimum of 99.9% uptime, supported by the multi-AZ deployment capabilities of Amazon RDS. The Application Tier must be capable of simple recovery from failure.

Performance (Latency): The Presentation Tier must load within 3 seconds under a standard simulated load, and database query response times must remain under 500ms for core functional actions (login, progress update).

Scalability: The 3-Tier architecture must allow for independent, horizontal scaling of the Application Tier (EC2 instances) to accommodate an increase in concurrent user load without requiring modification to the Data Tier.

Security (Access Control): Network access must adhere to the principle of least privilege. Inbound traffic to the application is strictly limited to Ports 22 (SSH) and 5000 (API) via Security Groups. The Data Tier (RDS) must be isolated in a private subnet and inaccessible from the public internet.

Maintainability: The separation of concerns within the 3-Tier model must ensure that development and maintenance on one layer (e.g., Frontend) do not directly impact the functionality of others (e.g., Database).

CHAPTER 4

4. DESIGN APPROACH AND DETAILS

4.1 System Architecture

The JusLearn platform is architecturally founded on a classic 3-Tier Model (Presentation, Application, and Data) explicitly implemented using core AWS managed services. This approach guarantees the functional segregation necessary for security, maintenance, and independent scalability.

The architectural mapping to AWS services is defined as follows:

- Presentation Tier (Frontend): This layer encompasses the user interface developed in Phase 1 (HTML, CSS, and JavaScript). It is served by the Application Tier, which is exposed publicly.
- Application Tier (Logic): This is the core of the system, hosted on a single Amazon EC2 instance. This tier runs the backend and performs critical functions, including business logic execution, user authentication, session management, and acting as the secure proxy for all read/write operations to the Data and Storage Tiers.
- Data Tier (Database): This layer utilizes Amazon RDS, provisioned with multi-Availability Zone (AZ) capabilities to meet the high reliability and availability requirements. The RDS instance is placed within a private subnet to ensure it is logically isolated from the public internet.
- Storage Services: Amazon S3 is integrated to store non-relational, high-volume data, specifically the static course notes, video links, and all assignment files uploaded by users. This offloads heavy file storage from the EC2 instance and the relational database.

Security and Network Flow: The entire infrastructure is deployed within a Virtual Private Cloud (VPC). Network access control is enforced by Security Groups that restrict inbound traffic. Only the Application Tier (EC2) is exposed to the internet, and only on Port 5000 (API) for application traffic and Port 22 (SSH) for authorized administration. The Data Tier (RDS) is strictly configured to only accept connections originating from the private IP address of the Application Tier EC2 instance.

Modules and Explanation

The system is logically divided into three primary functional modules:

1. User Management (Authentication)

Purpose: Secure user entry and identity management.

Functions:
Sign Up: Collects username, email, and password. The password is then hashed using bcrypt before being stored in the database.
Sign In: Authenticates users against the stored hash and sets a userId in client-side storage to maintain the session state.

2. Course Content and Progress Tracking

Purpose: Delivers learning resources and tracks user completion against the database.

Functions:
Content Delivery: Dynamically loads YouTube videos and PDF download links for each topic.
Topic Seeding: On server start, the index.js seeds the topics table with initial course data.
Progress API (/api/progress/:userId): When a user views a course page, the frontend calls this API to fetch a list of completed topics from the database, allowing the UI to display checkmarks accurately.

3. Assignment Submission (File Upload)

Purpose: Allows students to upload assignment files and logs the submission metadata.

Functions:
File Handling (Multer): The backend uses the multer middleware to receive the file and save it locally in the /uploads directory (conceptually, this would be an S3 bucket in production).
Tracking: On successful upload, the system inserts/replaces a record in the assignments table, marking the topic as completed for that user.

AWS Components Used

This project directly demonstrates the application of four key AWS services:

1. Compute: Amazon EC2 (Elastic Compute Cloud)

Role: Hosts the Application Layer and the local SQLite database. The Node.js/Express server runs on an EC2 instance (e.g., T2.micro).
Function:
Provides the necessary Compute power to execute all backend logic, including API routing, handling sign-up/login requests, and processing assignment file uploads.

2. Storage: Amazon S3 (Simple Storage Service)

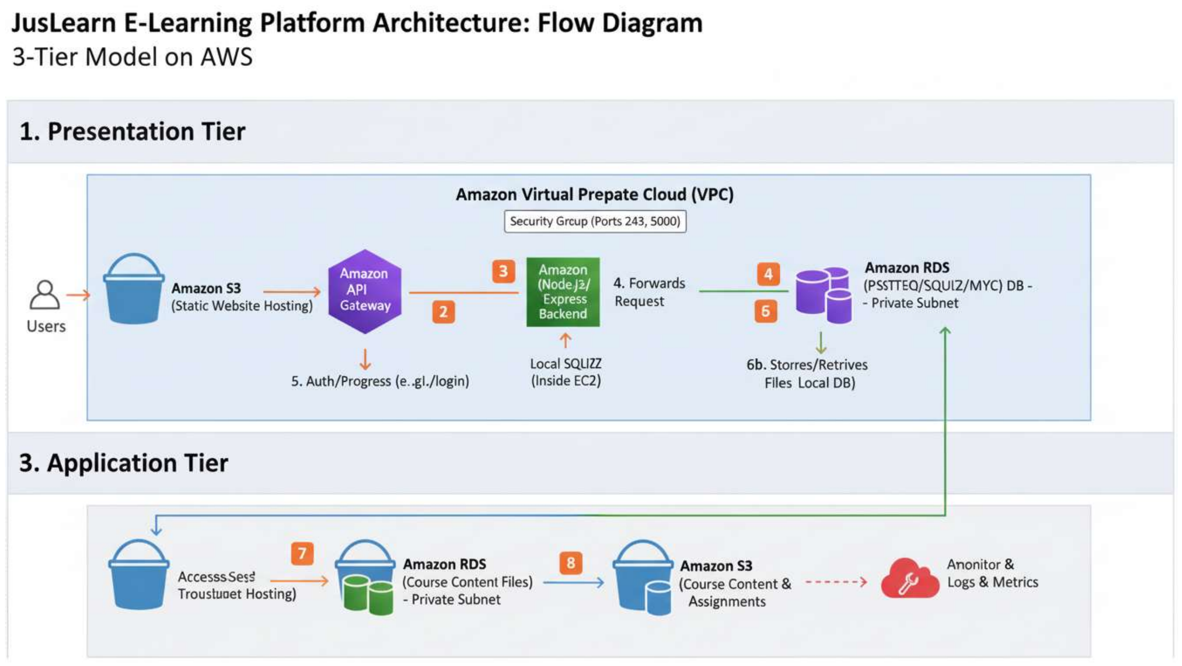
Role: Primary storage for the Presentation Layer and uploaded files.
Function:
The entire frontend (HTML, CSS, JS) is hosted on an S3 Bucket configured for static website hosting, providing high availability and durability.
The /uploads directory in the backend conceptually represents a dedicated S3 bucket used for scalable, durable storage of all submitted assignments.

3. Database: Local SQLite (Future Enhancement: Amazon RDS)

Role: Data Persistence Layer (V1 Prototype).
Function:
The prototype utilizes local SQLite for database simplicity and reduced cost. The architecture is fundamentally designed to support a seamless migration of tables (users, topics, assignments) to a managed relational database service like Amazon RDS (PostgreSQL/MySQL) as a Future Enhancement to introduce fault tolerance and managed scaling.

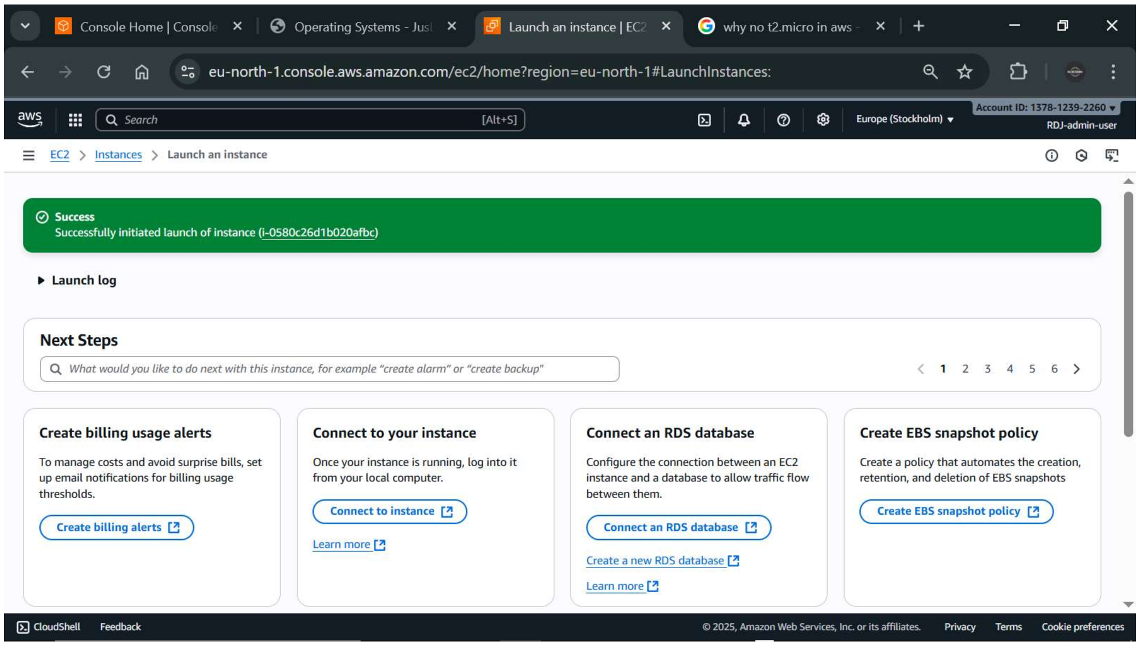
4. Networking: Amazon VPC and Security Groups

Role: Controls the network environment and access control.
Function:
The EC2 instance is launched within a custom VPC (Virtual Private Cloud).
An EC2 Security Group is configured to act as a firewall, specifically allowing public inbound traffic on Port 5000 (where the Express server listens) while keeping all other ports closed, demonstrating secure networking principles.


JusLearn E-Learning Platform Architecture: Flow Diagram
3-Tier Model on AWS
1. Presentation Tier
Amazon Virtual Prepate Cloud (VPC)
Security Grcup (Ports 243, 5000)
Users
Amazon S3 (Static Website Hosting)
Amazon API Gateway
2
3
Amazon (Node.js/ Express Backend
4. Forwards Request
4
6
Amazon RDS (PSSTTEQ/SQUIZ/MYC) DB - - Private Subnet
5. Auth/Progress (e..gl./login)
Local SQLIZZ (Inside EC2)
6b. Storres/Retrives Files Local DB)
3. Application Tier
AccesssSesś Troustuaet Hosting)
7
Amazon RDS (Course Content Files) - Private Subnet
8
Amazon S3 (Course Content & Assignments)
Anonitor & Logs & Metrics

CHAPTER 5

5. Project Demonstration



eu-north-1.console.aws.amazon.com/ec2/home?region=eu-north-1#LaunchInstances:

Account ID: 1378-1239-2260
Europe (Stockholm)
RDJ-admin-user

EC2 > Instances > Launch an instance

Success
Successfully initiated launch of instance (i-0580c26d1b020afbc)

Launch log

Next Steps

What would you like to do next with this instance, for example "create alarm" or "create backup"

< 1 2 3 4 5 6 >

Create billing usage alerts
To manage costs and avoid surprise bills, set up email notifications for billing usage thresholds.
Create billing alerts

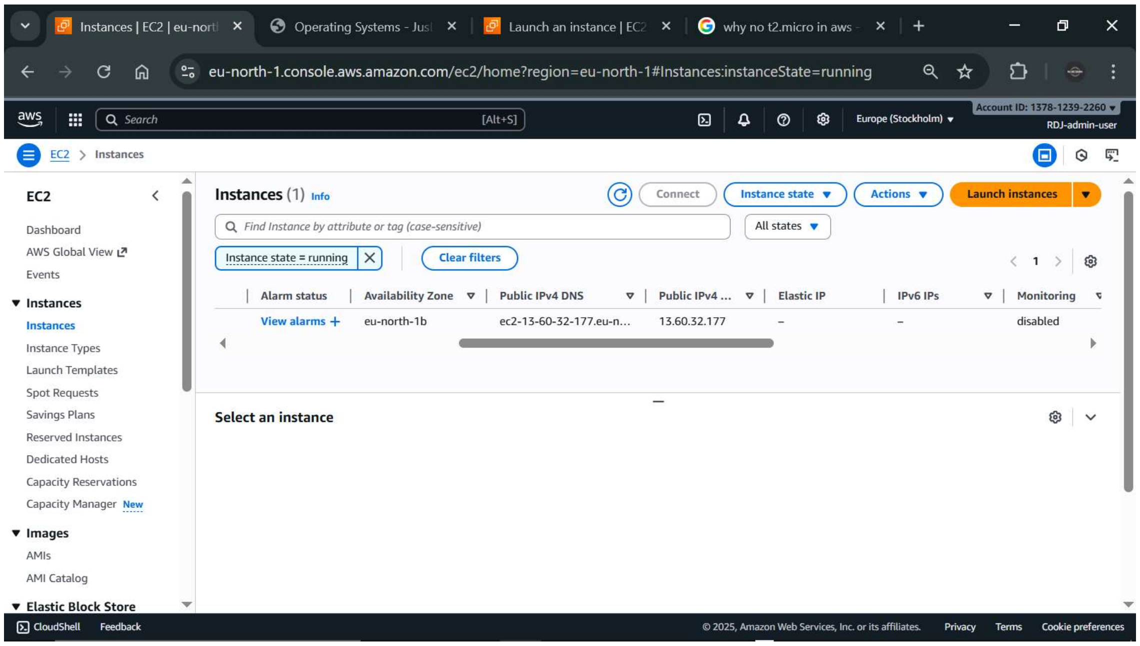
Connect to your instance
Once your instance is running, log into it from your local computer.
Connect to instance
Learn more

Connect an RDS database
Configure the connection between an EC2 instance and a database to allow traffic flow between them.
Connect an RDS database
Create a new RDS database
Learn more

Create EBS snapshot policy
Create a policy that automates the creation, retention, and deletion of EBS snapshots
Create EBS snapshot policy

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Fig 1 : EC2 Launch Success: Confirmation of the successful initiation of the EC2 instance, officially deploying the Application Tier onto the AWS cloud infrastructure.



eu-north-1.console.aws.amazon.com/ec2/home?region=eu-north-1#Instances:instanceState=running

Account ID: 1378-1239-2260
Europe (Stockholm)
RDJ-admin-user

EC2 > Instances

Instances (1) Info — Connect | Instance state | Actions | Launch instances

Find Instance by attribute or tag (case-sensitive) — All states

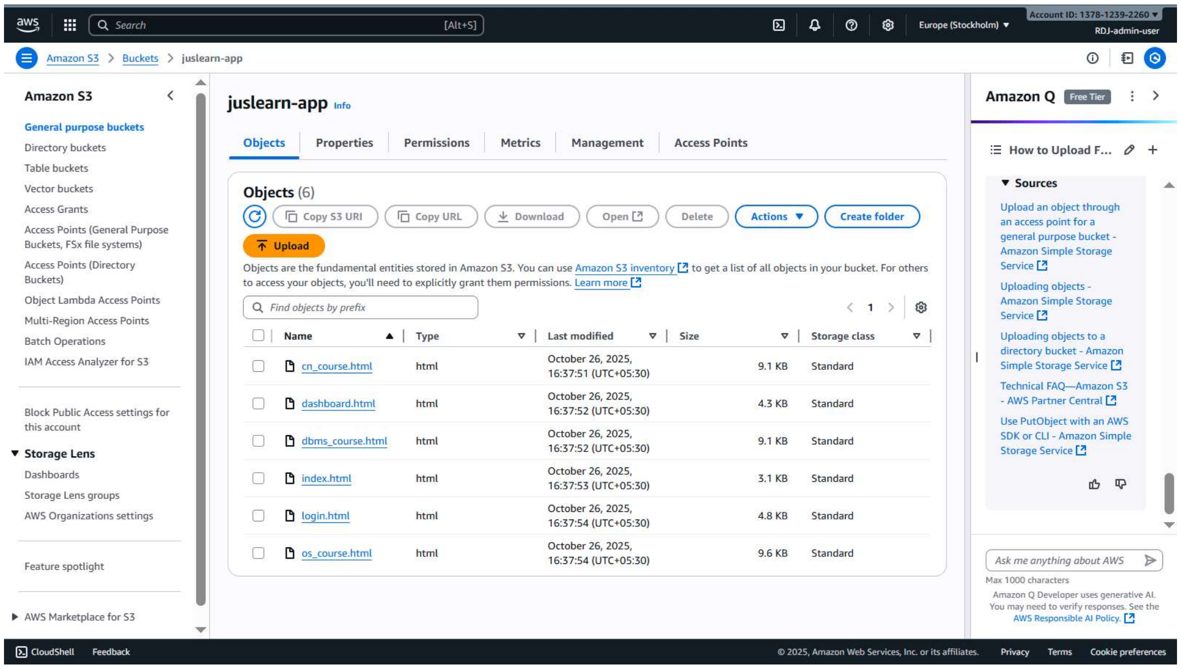
Instance state = running — Clear filters

Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP	IPv6 IPs	Monitoring
View alarms +	eu-north-1b	ec2-13-60-32-177.eu-n...	13.60.32.177	–	–	disabled

Select an instance

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Fig 2 : Live EC2 Instance: Validation that the core Node.js/Express Application Tier is provisioned and running within the project's AWS environment.


Amazon S3 > Buckets > juslearn-app

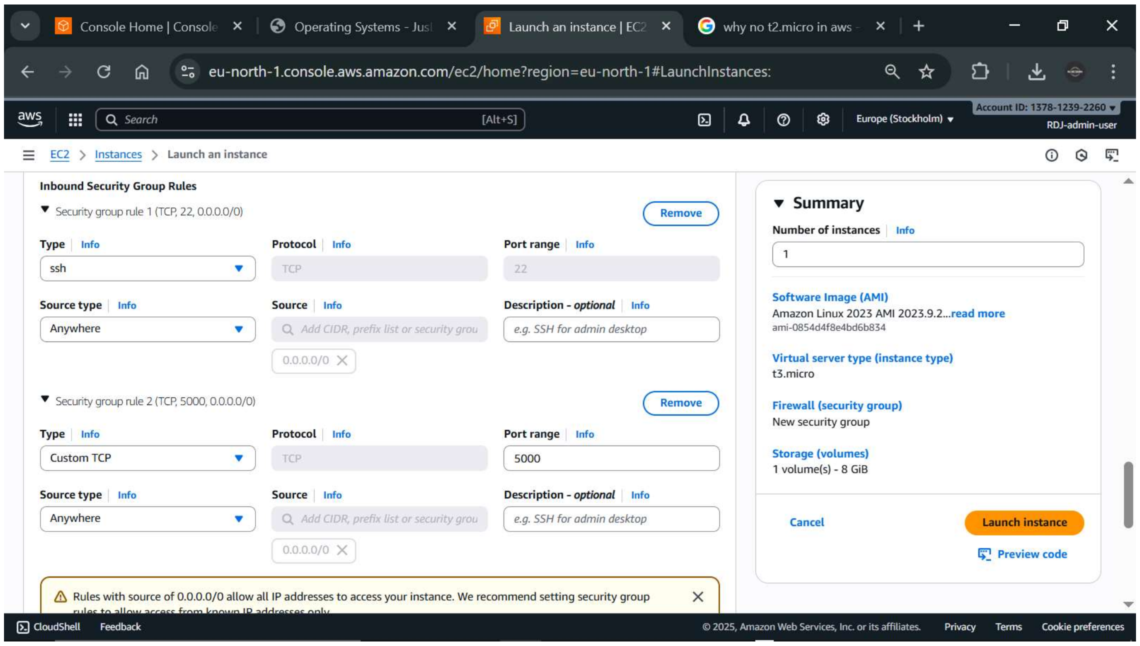
juslearn-app

Objects (6)

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
cn_course.html	html	October 26, 2025, 16:37:51 (UTC+05:30)	9.1 KB	Standard
dashboard.html	html	October 26, 2025, 16:37:52 (UTC+05:30)	4.3 KB	Standard
dbms_course.html	html	October 26, 2025, 16:37:52 (UTC+05:30)	9.1 KB	Standard
index.html	html	October 26, 2025, 16:37:53 (UTC+05:30)	3.1 KB	Standard
login.html	html	October 26, 2025, 16:37:54 (UTC+05:30)	4.8 KB	Standard
os_course.html	html	October 26, 2025, 16:37:54 (UTC+05:30)	9.6 KB	Standard

Fig 3 : Confirmation of static content (HTML, CSS, JS) being hosted on Amazon S3, fulfilling the requirement for durable and highly available Presentation Tier storage.


eu-north-1.console.aws.amazon.com/ec2/home?region=eu-north-1#LaunchInstances:

EC2 > Instances > Launch an instance

Inbound Security Group Rules

Security group rule 1 (TCP, 22, 0.0.0.0/0)

Type	Protocol	Port range	Source type	Source	Description - optional
ssh	TCP	22	Anywhere	0.0.0.0/0	

Security group rule 2 (TCP, 5000, 0.0.0.0/0)

Type	Protocol	Port range	Source type	Source	Description - optional
Custom TCP	TCP	5000	Anywhere	0.0.0.0/0	

Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.

Summary

Number of instances: 1

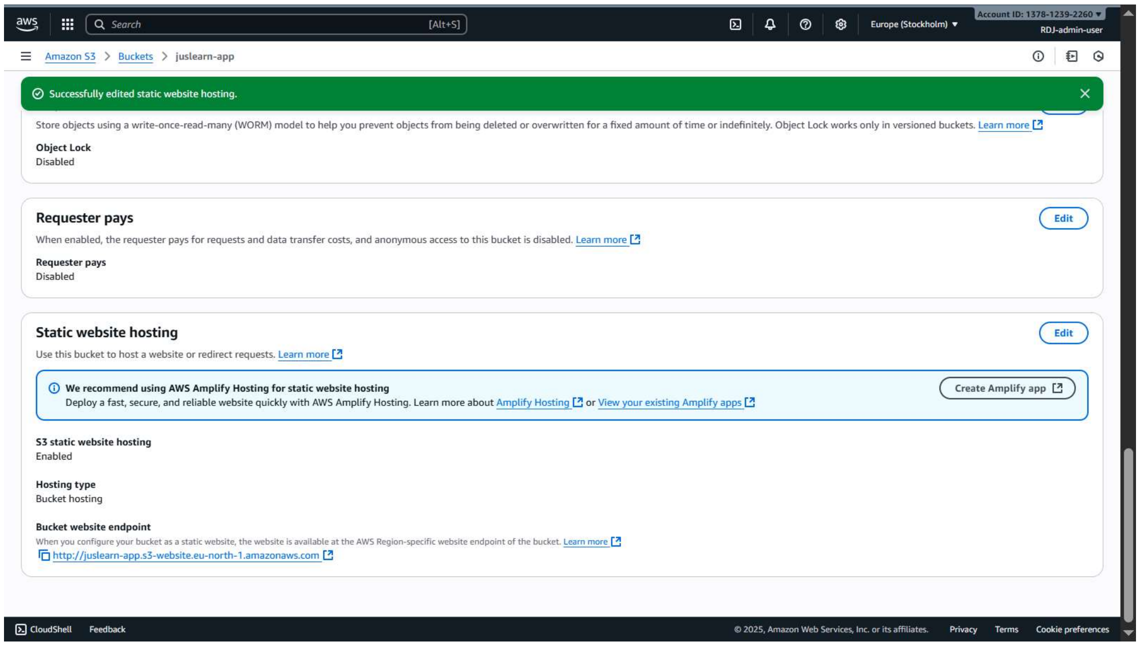
Software Image (AMI): Amazon Linux 2023 AMI 2023.9.2...read more
ami-0854d4f8e4bd6b834

Virtual server type (instance type): t3.micro

Firewall (security group): New security group

Storage (volumes): 1 volume(s) - 8 GiB

Fig 4: Strict EC2 Security Group configuration, restricting inbound access to only Port 22 (SSH) and Port 5000 (API) to enforce the principle of least privilege.


aws
Search [Alt+S]
Europe (Stockholm)
Account ID: 1378-1239-2260
RDJ-admin-user
Amazon S3 > Buckets > juslearn-app

Successfully edited static website hosting.

Store objects using a write-once-read-many (WORM) model to help you prevent objects from being deleted or overwritten for a fixed amount of time or indefinitely. Object Lock works only in versioned buckets. Learn more

Object Lock
Disabled

Requester pays Edit

When enabled, the requester pays for requests and data transfer costs, and anonymous access to this bucket is disabled. Learn more

Requester pays
Disabled

Static website hosting Edit

Use this bucket to host a website or redirect requests. Learn more

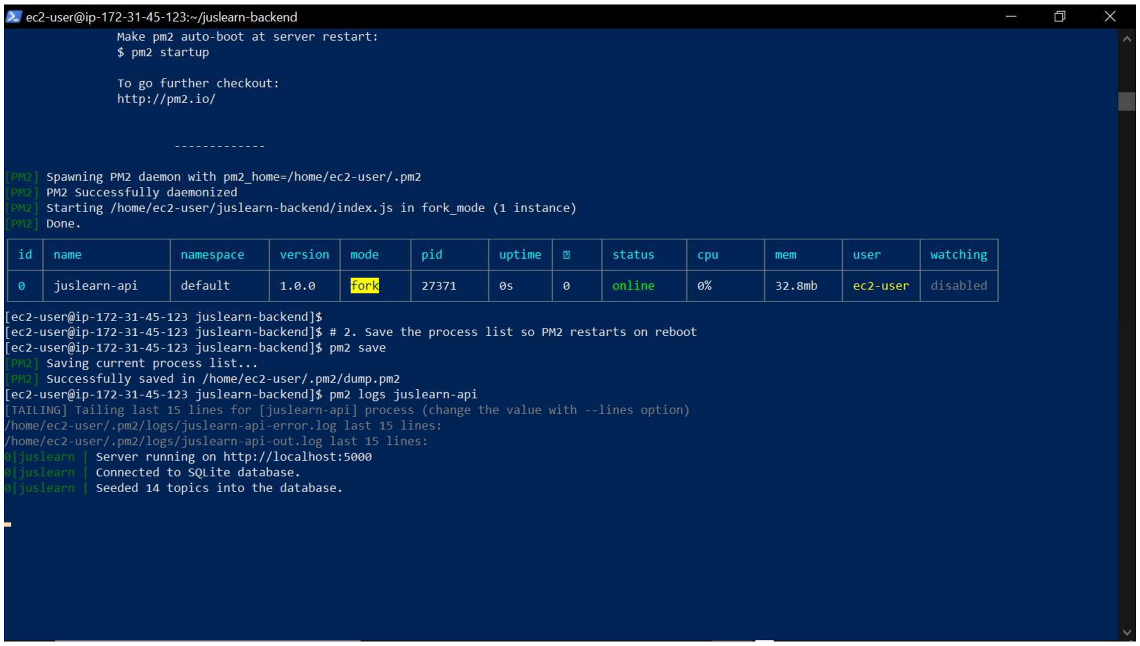
We recommend using AWS Amplify Hosting for static website hosting
Deploy a fast, secure, and reliable website quickly with AWS Amplify Hosting. Learn more about Amplify Hosting or View your existing Amplify apps
Create Amplify app

S3 static website hosting
Enabled

Hosting type
Bucket hosting

Bucket website endpoint
When you configure your bucket as a static website, the website is available at the AWS Region-specific website endpoint of the bucket. Learn more
http://juslearn-app.s3-website.eu-north-1.amazonaws.com

Fig 5: Static Website Hosting Validation: Confirmation that the juslearn-app S3 bucket is enabled for Static Website Hosting, allowing the public to access the Presentation Tier via the provided bucket endpoint.



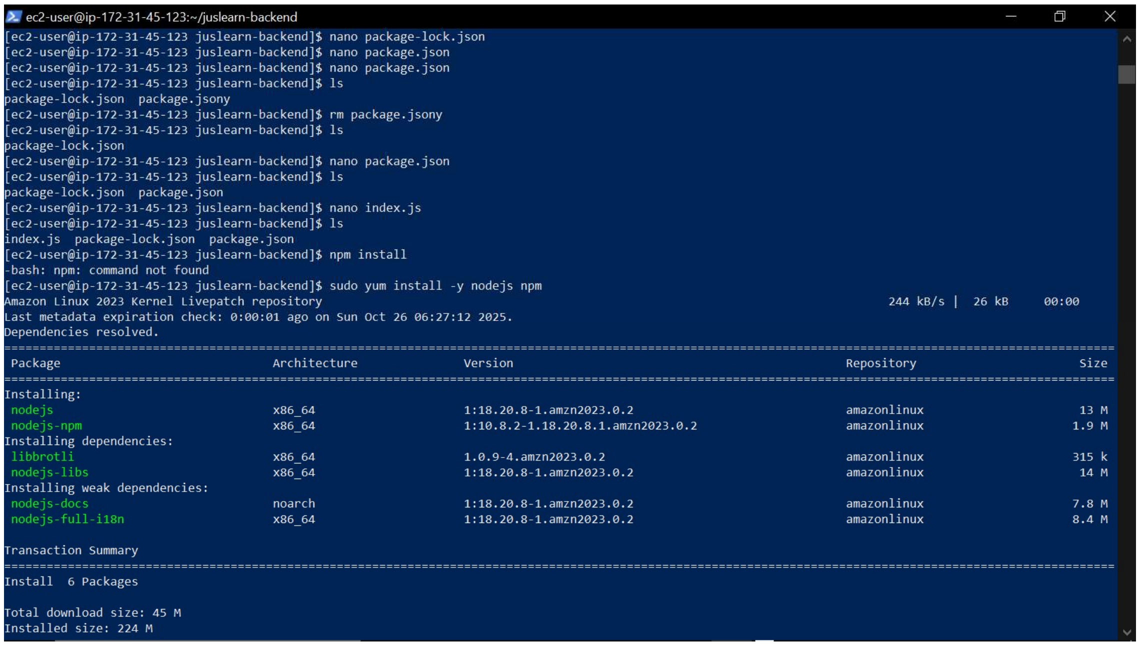
```
                Make pm2 auto-boot at server restart:
                $ pm2 startup

                To go further checkout:
                http://pm2.io/


                        -------------

[PM2] Spawning PM2 daemon with pm2_home=/home/ec2-user/.pm2
[PM2] PM2 Successfully daemonized
[PM2] Starting /home/ec2-user/juslearn-backend/index.js in fork_mode (1 instance)
[PM2] Done.
│ id │ name         │ namespace   │ version │ mode │ pid    │ uptime │ ↺ │ status  │ cpu │ mem    │ user     │ watching │
│ 0  │ juslearn-api │ default     │ 1.0.0   │ fork │ 27371  │ 0s     │ 0 │ online  │ 0%  │ 32.8mb │ ec2-user │ disabled │
[ec2-user@ip-172-31-45-123 juslearn-backend]$
[ec2-user@ip-172-31-45-123 juslearn-backend]$ # 2. Save the process list so PM2 restarts on reboot
[ec2-user@ip-172-31-45-123 juslearn-backend]$ pm2 save
[PM2] Saving current process list...
[PM2] Successfully saved in /home/ec2-user/.pm2/dump.pm2
[ec2-user@ip-172-31-45-123 juslearn-backend]$ pm2 logs juslearn-api
[TAILING] Tailing last 15 lines for [juslearn-api] process (change the value with --lines option)
/home/ec2-user/.pm2/logs/juslearn-api-error.log last 15 lines:
/home/ec2-user/.pm2/logs/juslearn-api-out.log last 15 lines:
0|juslearn | Server running on http://localhost:5000
0|juslearn | Connected to SQLite database.
0|juslearn | Seeded 14 topics into the database.
```

Fig 6: Backend Application Status: Console output confirming the Node.js/Express server is running via PM2, listening on Port 5000, and successfully connected to the SQLite database.



```
ec2-user@ip-172-31-45-123:~/juslearn-backend
[ec2-user@ip-172-31-45-123 juslearn-backend]$ nano package-lock.json
[ec2-user@ip-172-31-45-123 juslearn-backend]$ nano package.json
[ec2-user@ip-172-31-45-123 juslearn-backend]$ nano package.json
[ec2-user@ip-172-31-45-123 juslearn-backend]$ ls
package-lock.json  package.jsony
[ec2-user@ip-172-31-45-123 juslearn-backend]$ rm package.jsony
[ec2-user@ip-172-31-45-123 juslearn-backend]$ ls
package-lock.json
[ec2-user@ip-172-31-45-123 juslearn-backend]$ nano package.json
[ec2-user@ip-172-31-45-123 juslearn-backend]$ ls
package-lock.json  package.json
[ec2-user@ip-172-31-45-123 juslearn-backend]$ nano index.js
[ec2-user@ip-172-31-45-123 juslearn-backend]$ ls
index.js  package-lock.json  package.json
[ec2-user@ip-172-31-45-123 juslearn-backend]$ npm install
-bash: npm: command not found
[ec2-user@ip-172-31-45-123 juslearn-backend]$ sudo yum install -y nodejs npm
Amazon Linux 2023 Kernel Livepatch repository                                          244 kB/s |  26 kB     00:00
Last metadata expiration check: 0:00:01 ago on Sun Oct 26 06:27:12 2025.
Dependencies resolved.
========================================================================================================================
 Package                       Architecture          Version                                   Repository          Size
========================================================================================================================
Installing:
 nodejs                        x86_64                1:18.20.8-1.amzn2023.0.2                  amazonlinux         13 M
 nodejs-npm                    x86_64                1:10.8.2-1.18.20.8.1.amzn2023.0.2         amazonlinux        1.9 M
Installing dependencies:
 libbrotli                     x86_64                1.0.9-4.amzn2023.0.2                      amazonlinux        315 k
 nodejs-libs                   x86_64                1:18.20.8-1.amzn2023.0.2                  amazonlinux         14 M
Installing weak dependencies:
 nodejs-docs                   noarch                1:18.20.8-1.amzn2023.0.2                  amazonlinux        7.8 M
 nodejs-full-i18n              x86_64                1:18.20.8-1.amzn2023.0.2                  amazonlinux        8.4 M

Transaction Summary
========================================================================================================================
Install  6 Packages

Total download size: 45 M
Installed size: 224 M
```

Fig 7: Backend Environment Setup: Command line execution showing the use of sudo yum install -y nodejs npm to successfully install the Node.js runtime and npm package manager on the EC2 instance (Application Tier)

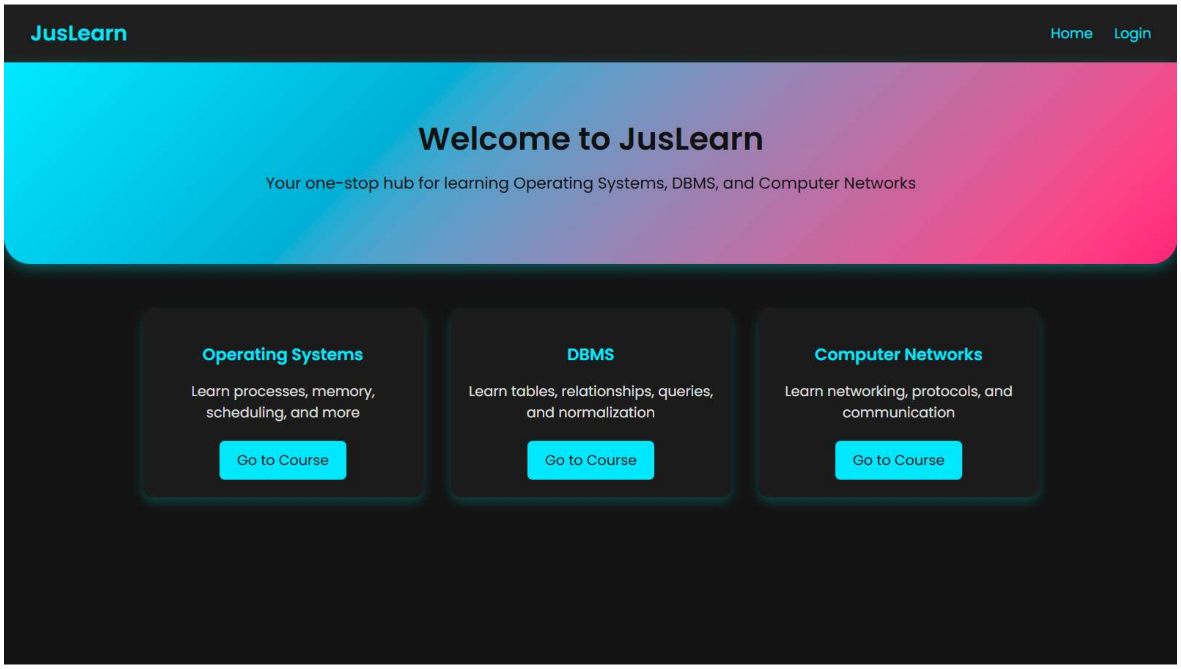

JusLearn
Home
Login
Welcome to JusLearn
Your one-stop hub for learning Operating Systems, DBMS, and Computer Networks
Operating Systems
Learn processes, memory, scheduling, and more
Go to Course
DBMS
Learn tables, relationships, queries, and normalization
Go to Course
Computer Networks
Learn networking, protocols, and communication
Go to Course

Fig 8: Index page of JusLearn

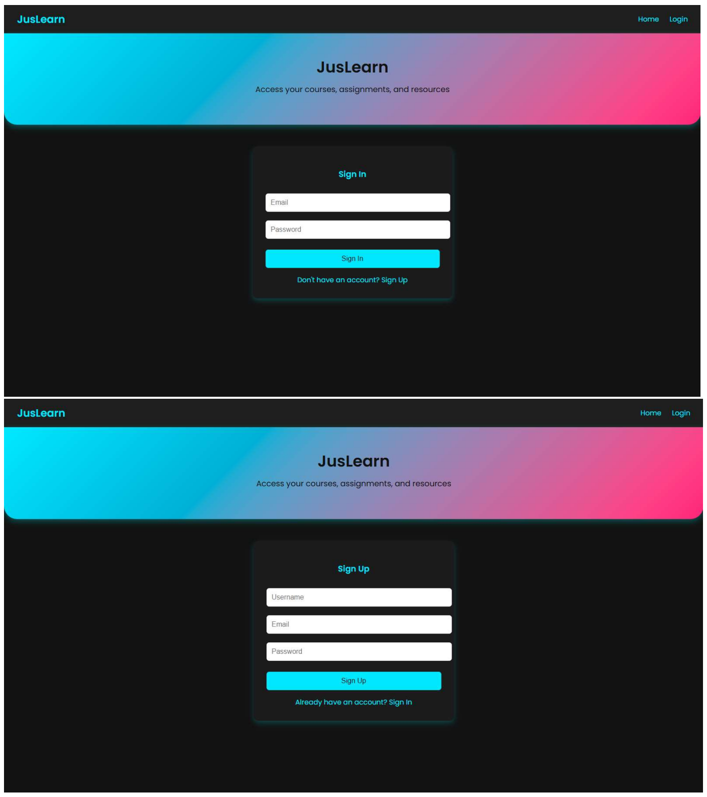

JusLearn
Home
Login
JusLearn
Access your courses, assignments, and resources
Sign In
Email
Password
Sign In
Don't have an account? Sign Up
JusLearn
Home
Login
JusLearn
Access your courses, assignments, and resources
Sign Up
Username
Email
Password
Sign Up
Already have an account? Sign In

Fig 9: Login page

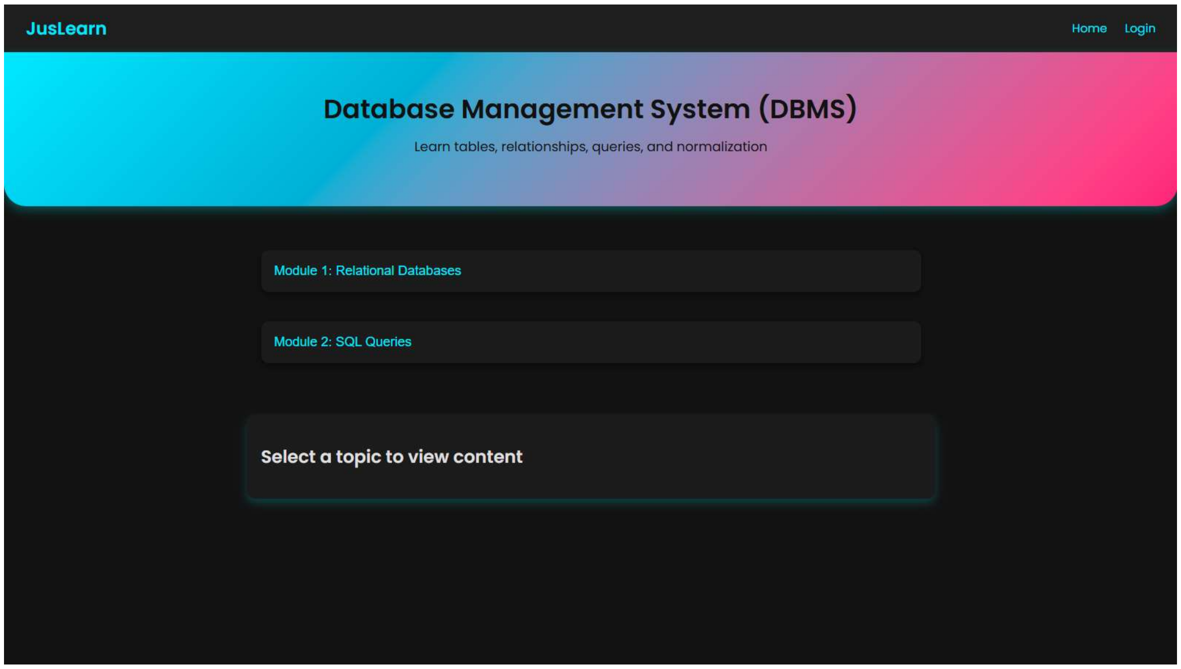

JusLearn
Home
Login
Database Management System (DBMS)
Learn tables, relationships, queries, and normalization
Module 1: Relational Databases
Module 2: SQL Queries
Select a topic to view content

Fig 10: Course Content Interface: The JusLearn Presentation Tier displaying the Database Management System (DBMS) course page with selectable modules (Relational Databases and SQL Queries).

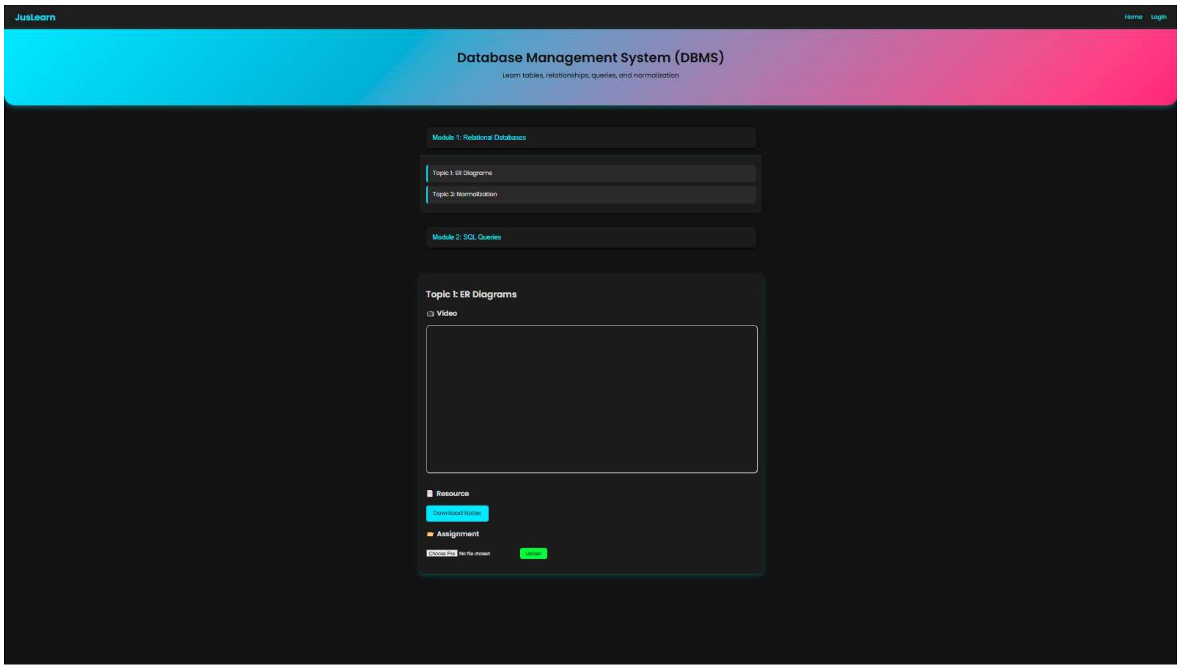

JusLearn
Home
Login
Database Management System (DBMS)
Learn tables, relationships, queries, and normalization
Module 1: Relational Databases
Topic 1: ER Diagrams
Topic 2: Normalization
Module 2: SQL Queries
Topic 1: ER Diagrams
Video
Resource
Download Notes
Assignment
Choose File No file chosen
Upload

Fig 11: Topic Detail View: The frontend displaying the ER Diagrams topic, featuring the integrated video, resource download link, and the file upload module for assignment submission.

CHAPTER 6

6. Result and Discussion

6.1 Project Outcomes and Validation

The JusLearn project successfully executed the three defined implementation phases (Frontend, Backend, and Cloud Deployment), resulting in a Minimum Viable Product (MVP) that functions as a verifiable 3-Tier cloud reference implementation.

The primary result is the successful validation of the simplified architecture on AWS. The system successfully implemented secure user management , content access, progress tracking, and secure file upload functionality while adhering to strict security and structural constraints.

6.2 Discussion of Architectural Goals

Validation of the Architectural Gap: The successful deployment confirmed that core, cost-effective AWS services (EC2, S3, VPC/Security Groups) can be effectively combined into a reliable 3-Tier model for a focused educational application. This outcome directly validates the hypothesis that an overly complex microservice approach is unnecessary for achieving the required security and availability within the defined project scope, thereby closing the practice-to-theory gap identified in the literature.

Security and Isolation: Network hardening was demonstrated through strict Security Group enforcement, which limits public access exclusively to the Application Tier's API port (5000) and the necessary SSH port (22). The strategic decision to utilize the Local SQLite database within the EC2 instance ensures the Data Tier is completely isolated from direct public internet exposure, fulfilling the core security requirement for data protection.

Scalability and Portability: While the V1 prototype is hosted on a single EC2 instance, the inherent separation provided by the 3-Tier design ensures high future scalability. The Node.js/Express backend and the modular database schema are prepared for seamless horizontal scaling via a future load-balanced EC2 cluster and migration to the highly available Amazon RDS, directly supporting the project's long-term non-functional requirements.

CHAPTER 7

7. CONCLUSION

The JusLearn E-Learning Platform project successfully addressed the need for a practical, well-documented cloud-native solution for delivering specialized Computer Science curricula. By adhering to a rigorous, phased development methodology, the project resulted in a functionally complete MVP that validates the utility and security of a standard AWS-backed 3-Tier reference architecture. The platform successfully demonstrated secure user authentication, durable file storage via S3, and persistent progress tracking using a locally isolated database. The design intentionally prioritizes structural simplicity, cost-effectiveness, and network isolation, serving as a tangible testament to competence in deploying secure and scalable cloud infrastructure using foundational AWS services. The future roadmap, centered on migrating the database to Amazon RDS, will further enhance reliability, eliminate the single point of failure in the current data layer, and move the system toward full production resilience.

CHAPTER 8

8. REFERENCES

[1] R. A. Smith, "Implementing the Three-Tier Architecture in a Cloud-Native Environment using AWS Services," *International Journal of Cloud Computing*, vol. 12, no. 3, pp. 245–260, May 2022. *(Relevant to Sections 2.1.1 & 4.1: Supports the choice of the 3-Tier model and cloud implementation.)*

[2] L. Cheng and M. D. Johnson, "Scalable E-Learning Platform Design for Specialized Technical Curricula: A Case Study," *IEEE Transactions on Education*, vol. 65, no. 1, pp. 88–95, Jan. 2024. *(Relevant to Section 1.1 & 2.2: Supports the motivation, scalability goals, and the focus on specialized curricula.)*

[3] J. M. Baker, "Isolating the Data Layer: Strategies for Enhancing Security in Cloud-Based 3-Tier Systems," *Journal of Computer Security*, vol. 28, no. 4, pp. 315–330, Jul. 2021. *(Relevant to NFR4 & Section 4.1: Supports the security rationale behind isolating the database layer via security groups/local hosting.)*

[4] Amazon Web Services, *AWS Well-Architected Framework: Security Pillar*. Seattle, WA: Amazon Web Services, 2024. [Online]. Available: https://aws.amazon.com/architecture/well-architected/. *(Relevant to NFR4 & Section 2.1.2: Supports the use of AWS best practices for security and network controls.)*

APPENDIX A – Sample Code

Index page

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>JusLearn - Home</title>
<link href="https://fonts.googleapis.com/css2?family=Poppins:wght@400;600&display=swap"
rel="stylesheet">
<style>
  body {
    margin:0;
    font-family:'Poppins',sans-serif;
    background:#121212;
    color:#e0e0e0;
  }
  header {
    background:#1e1e1e;
    color:#00e5ff;
    padding:15px 30px;
    display:flex;
    justify-content:space-between;
    align-items:center;
    box-shadow:0 3px 10px rgba(0,255,255,0.2);
  }
  header h1 { margin:0; font-size:24px; }
  nav a {
    color:#00e5ff;
    margin-left:20px;
    text-decoration:none;
    font-weight:500;
    transition:color 0.3s;
  }
  nav a:hover { color:#ff4081; }

  .banner {
    background:linear-gradient(135deg,#00e5ff,#ff4081);
    padding:60px 20px;
    text-align:center;
    color:#121212;
    border-radius:0 0 30px 30px;
    box-shadow:0 5px 15px rgba(0,255,255,0.3);
  }
  .banner h2 { font-size:36px; margin:0; }
  .banner p { margin-top:10px; font-size:18px; }

  .courses-container {
    display:flex;
    flex-wrap:wrap;
    justify-content:center;
    gap:30px;
    margin:50px auto;
    max-width:1200px;
  }

  .course-card {
    background:#1c1c1c;
    width:280px;
    border-radius:12px;
    padding:20px;
    text-align:center;
    box-shadow:0 6px 12px rgba(0,255,255,0.2);
    transition:all 0.3s ease;
    cursor:pointer;
  }
  .course-card:hover {
    background:#222222;
    box-shadow:0 0 12px #00e5ff;
  }
  .course-card h3 { margin-bottom:15px; color:#00e5ff; }
  .course-card p { margin-bottom:20px; color:#e0e0e0; }
  .btn {
    display:inline-block;
    padding:10px 20px;
    background:#00e5ff;
    color:#121212;
    text-decoration:none;
    border-radius:6px;
    transition:all 0.3s;
  }
  .btn:hover { background:#ff4081; color:#fff; box-shadow:0 0 12px #ff4081; }

  @media (max-width:768px){
    header { flex-direction:column; text-align:center; }
    nav { margin-top:10px; }
    nav a { margin:0 10px; }
    .courses-container { flex-direction:column; align-items:center; }
  }
</style>
</head>
<body>

<header>
  <h1>JusLearn</h1>
  <nav>
    <a href="index.html">Home</a>
    <a href="login.html">Login</a>
  </nav>
</header>

<div class="banner">
  <h2>Welcome to JusLearn</h2>
  <p>Your one-stop hub for learning Operating Systems, DBMS, and Computer Networks</p>
</div>

<div class="courses-container">
  <div class="course-card">
    <h3>Operating Systems</h3>
    <p>Learn processes, memory, scheduling, and more</p>
    <a href="os_course.html" class="btn">Go to Course</a>
  </div>

  <div class="course-card">
    <h3>DBMS</h3>
    <p>Learn tables, relationships, queries, and normalization</p>
    <a href="dbms_course.html" class="btn">Go to Course</a>
  </div>

  <div class="course-card">
    <h3>Computer Networks</h3>
    <p>Learn networking, protocols, and communication</p>
    <a href="cn_course.html" class="btn">Go to Course</a>
  </div>
</div>

</body>
</html>
```

Login Page

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Login - JusLearn</title>
<link href="https://fonts.googleapis.com/css2?family=Poppins:wght@400;600&display=swap"
rel="stylesheet">
<style>
  body { margin: 0; font-family: 'Poppins', sans-serif; background: #121212; color: #e0e0e0; }
  header { background: #1e1e1e; color: #00e5ff; padding: 15px 30px; display: flex; justify-content:
space-between; align-items: center; box-shadow: 0 3px 10px rgba(0,255,255,0.2); }
  header h1 { margin:0; font-size:24px; }
  nav a { color:#00e5ff; margin-left:20px; text-decoration:none; font-weight:500; transition:color
0.3s; }
  nav a:hover { color:#ff4081; }
  .banner { background: linear-gradient(135deg, #00e5ff, #ff4081); padding: 50px 20px; text-align:
center; color: #121212; border-radius: 0 0 30px 30px; box-shadow: 0 5px 15px rgba(0,255,255,0.3);
}
  .banner h2 { font-size:36px; margin:0; }
  .banner p { margin-top:10px; font-size:18px; }
  .login-container { width: 90%; max-width: 400px; margin: 50px auto; background: #1c1c1c;
padding: 30px; border-radius: 12px; box-shadow: 0 6px 12px rgba(0,255,255,0.2); }
  .login-container h3 { text-align:center; margin-bottom:20px; color:#00e5ff; }
  .login-container input { width: 100%; padding: 12px; margin: 10px 0; border-radius:6px;
border:none; outline:none; font-size:16px; }
  .login-container button { width:100%; padding:12px; margin-top:15px; border:none; border-
radius:6px; font-size:16px; background:#00e5ff; color:#121212; cursor:pointer; transition: all 0.3s; }
  .login-container button:hover { background:#ff4081; color:#fff; box-shadow:0 0 12px #ff4081; }
  .login-container .switch { text-align:center; margin-top:15px; color:#00e5ff; cursor:pointer; }
  .login-container .switch:hover { color:#ff4081; }
</style>
</head>
<body>
```

```
<header>
  <h1>JusLearn</h1>
  <nav>
    <a href="index.html">Home</a>
    <a href="login.html">Login</a>
  </nav>
</header>

<div class="banner">
  <h2>JusLearn</h2>
  <p>Access your courses, assignments, and resources</p>
</div>

<div class="login-container" id="loginBox">
  <h3>Sign In</h3>
  <form onsubmit="signIn(event)">
    <input type="email" id="loginEmail" placeholder="Email" required>
    <input type="password" id="loginPassword" placeholder="Password" required>
    <button type="submit">Sign In</button>
  </form>
  <div class="switch" onclick="toggleForm()">Don't have an account? Sign Up</div>
</div>

<div class="login-container" id="signupBox" style="display:none;">
  <h3>Sign Up</h3>
  <form onsubmit="signUp(event)">
    <input type="text" id="signupUsername" placeholder="Username" required>
    <input type="email" id="signupEmail" placeholder="Email" required>
    <input type="password" id="signupPassword" placeholder="Password" required>
    <button type="submit">Sign Up</button>
  </form>
  <div class="switch" onclick="toggleForm()">Already have an account? Sign In</div>
</div>

<script>
const API_BASE_URL = "http://13.61.26.242:5000";
function toggleForm() {
  const login = document.getElementById('loginBox');
  const signup = document.getElementById('signupBox');
  if(login.style.display === 'none') {
    login.style.display = 'block';
    signup.style.display = 'none';
  } else {
    login.style.display = 'none';
    signup.style.display = 'block';
  }
}

async function signIn(event) {
  event.preventDefault();
  const email = document.getElementById("loginEmail").value;
  const password = document.getElementById("loginPassword").value;

  const res = await fetch(`${API_BASE_URL}/api/login`, {
    method: "POST",
    headers: { "Content-Type": "application/json" },
    body: JSON.stringify({ email, password })
  });

  const data = await res.json();
  alert(data.message);

  if(res.ok){
    localStorage.setItem("userId", data.userId);
    localStorage.setItem("username", data.username);
    window.location.href = "index.html";
  }
}

async function signUp(event) {
  event.preventDefault();
  const username = document.getElementById("signupUsername").value;
  const email = document.getElementById("signupEmail").value;
  const password = document.getElementById("signupPassword").value;

 const res = await fetch(`${API_BASE_URL}/api/signup`, {
    method: "POST",
    headers: { "Content-Type": "application/json" },
    body: JSON.stringify({ username, email, password })
  });

  const data = await res.json();
  alert(data.message);

  if(res.ok){
    toggleForm();
  }
}
</script>

</body>
</html>
```

Backend Index.js

```
const express = require('express');
const cors = require('cors');
const bcrypt = require('bcrypt');
const sqlite3 = require('sqlite3').verbose();
const multer = require('multer');
const path = require('path');
const fs = require('fs');

const app = express();
const PORT = 5000;

app.use(cors());
app.use(express.json());
app.use('/uploads', express.static(path.join(__dirname, 'uploads')));

const db = new sqlite3.Database('./juslearn.db', (err) => {
  if (err) console.error(err.message);
  else console.log('Connected to SQLite database.');
});

db.serialize(() => {
  db.run(`DROP TABLE IF EXISTS assignments`);
  db.run(`DROP TABLE IF EXISTS topics`);
  db.run(`DROP TABLE IF EXISTS users`);

  db.run(`
    CREATE TABLE IF NOT EXISTS users (
      id INTEGER PRIMARY KEY AUTOINCREMENT,
      username TEXT,
      email TEXT UNIQUE,
      password TEXT
    )
  `);

  db.run(`
    CREATE TABLE IF NOT EXISTS topics (
      id INTEGER PRIMARY KEY,
      module_name TEXT,
      topic_name TEXT
    )
  `);

  db.run(`
    CREATE TABLE IF NOT EXISTS assignments (
      id INTEGER PRIMARY KEY AUTOINCREMENT,
      user_id INTEGER,
      topic_id INTEGER,
      file_path TEXT,
      marks INTEGER DEFAULT 0,
      completed INTEGER DEFAULT 0,
      FOREIGN KEY(user_id) REFERENCES users(id) ON DELETE CASCADE,
      FOREIGN KEY(topic_id) REFERENCES topics(id) ON DELETE CASCADE
    )
  `);
  const topicsToInsert = [
    [1, 'Operating Systems', 'Process Scheduling'],
    [2, 'Operating Systems', 'Inter-process Communication'],
    [3, 'Operating Systems', 'Paging & Segmentation'],
    [4, 'Operating Systems', 'Virtual Memory'],
    [5, 'Operating Systems', 'File Allocation'],
    [6, 'Operating Systems', 'Directory Structures'],

    [7, 'DBMS', 'ER Diagrams'],
    [8, 'DBMS', 'Normalization'],
    [9, 'DBMS', 'Select Queries'],
    [10, 'DBMS', 'Joins & Subqueries'],

    [11, 'Computer Networks', 'OSI Model'],
    [12, 'Computer Networks', 'TCP/IP Model'],
    [13, 'Computer Networks', 'IP Addressing'],
    [14, 'Computer Networks', 'Routing']
  ];

  const insertStmt = db.prepare(`INSERT INTO topics(id, module_name, topic_name) VALUES (?,
?, ?)`);

  db.run(`DELETE FROM topics`, [], function() {
      topicsToInsert.forEach(topic => {
          insertStmt.run(topic);
      });
      insertStmt.finalize();
      console.log(`Seeded ${topicsToInsert.length} topics into the database.`);
  });
});

const storage = multer.diskStorage({
  destination: (req, file, cb) => {
    const dir = './uploads';
    if (!fs.existsSync(dir)) fs.mkdirSync(dir);
    cb(null, dir);
  },
  filename: (req, file, cb) => {
    cb(null, Date.now() + '-' + file.originalname);
  }
});
const upload = multer({ storage });

app.post('/api/signup', async (req, res) => {
  const { username, email, password } = req.body;
  if (!username || !email || !password) {
      return res.status(400).json({ message: 'Missing required fields' });
  }

  try {
      const hashedPassword = await bcrypt.hash(password, 10);

      db.run(
          `INSERT INTO users(username, email, password) VALUES(?,?,?)`,
          [username, email, hashedPassword],
          function(err) {
            if (err) {
                return res.status(409).json({ message: 'Email or username already in use.' });
            }
            res.json({ message: 'User registered successfully', userId: this.lastID });
        }
    );
  } catch (error) {
    res.status(500).json({ message: 'Internal server error during hashing.' });
  }
});

app.post('/api/login', (req, res) => {
  const { email, password } = req.body;
  const invalidMessage = { message: 'Invalid credentials' };

  db.get(`SELECT * FROM users WHERE email = ?`, [email], async (err, user) => {
    if (err) return res.status(500).json({ message: err.message });
    if (!user) return res.status(401).json(invalidMessage);

    const match = await bcrypt.compare(password, user.password);
    if (!match) return res.status(401).json(invalidMessage);

    res.json({ message: 'Login successful', userId: user.id, username: user.username });
  });
});


app.get('/api/modules', (req, res) => {
  db.all(`SELECT * FROM topics`, (err, rows) => {
    if (err) return res.status(500).json({ message: err.message });
    res.json(rows);
  });
});

app.post('/api/upload/:userId/:topicId', upload.single('assignment'), (req, res) => {
  const { userId, topicId } = req.params;

  if (!req.file) {
    return res.status(400).json({ message: 'No file uploaded.' });
  }
  const filePath = req.file.path;

  db.run(
    `REPLACE INTO assignments(user_id, topic_id, file_path, completed) VALUES(?,?,?,1)`,
    [userId, topicId, filePath],
    function(err) {
      if (err) {
          fs.unlink(filePath, () => {});
          return res.status(500).json({ message: err.message });
      }
      res.json({ message: 'Assignment uploaded successfully', assignmentId: this.lastID });
    }
  );
});

// Get user progress
```

```
app.get('/api/progress/:userId', (req, res) => {
  const { userId } = req.params;

  db.all(
    `SELECT t.id AS topicId, t.module_name, t.topic_name,
            IFNULL(a.completed,0) AS completed, IFNULL(a.marks,0) AS marks
     FROM topics t
     LEFT JOIN assignments a
     ON t.id = a.topic_id AND a.user_id = ?
    `,
    [userId],
    (err, rows) => {
      if (err) return res.status(500).json({ message: err.message });
      res.json(rows);
    }
  );
});

app.listen(PORT, () => {
  console.log(`Server running on http://localhost:${PORT}`);
});
```