Presented By:- Branch:- AI&ML

Team no:A3

Team Members: Dutta Sree Naaga Sai (Team Lead)

K. Vyuhitha

P. Ramya Sri

Ch. V. N. Vyshnavi

Learning Hub AWS-Powered Corporate Training Platform

Project Description:

The "Learning Hub" project develops a scalable corporate training platform, utilizing Flask for backend development and AWS EC2 for hosting. The platform ensures high availability and performance, supporting a growing user base. Amazon RDS manages the database, storing course details and tracking user progress. By leveraging AWS services, the platform offers flexibility, scalability, and secure content delivery. This cloud-native solution demonstrates how AWS can streamline course management, user interactions, and content delivery in a seamless and efficient manner.

Scenario 1: Scalable Web Applications for Online Corporate training

In corporate training scenarios, AWS EC2 provides a scalable infrastructure that adapts to varying levels of user activity. For example, a professional training platform could use EC2 during course enrollments or live training sessions, ensuring the platform remains responsive and accessible. By leveraging Flask for backend development, the platform can efficiently manage user sessions, course catalogs, and progress tracking. This setup allows the training platform to scale seamlessly, accommodating increased user demand without compromising performance or content delivery.

Scenario 2: Efficient Database Management for Course Enrollments

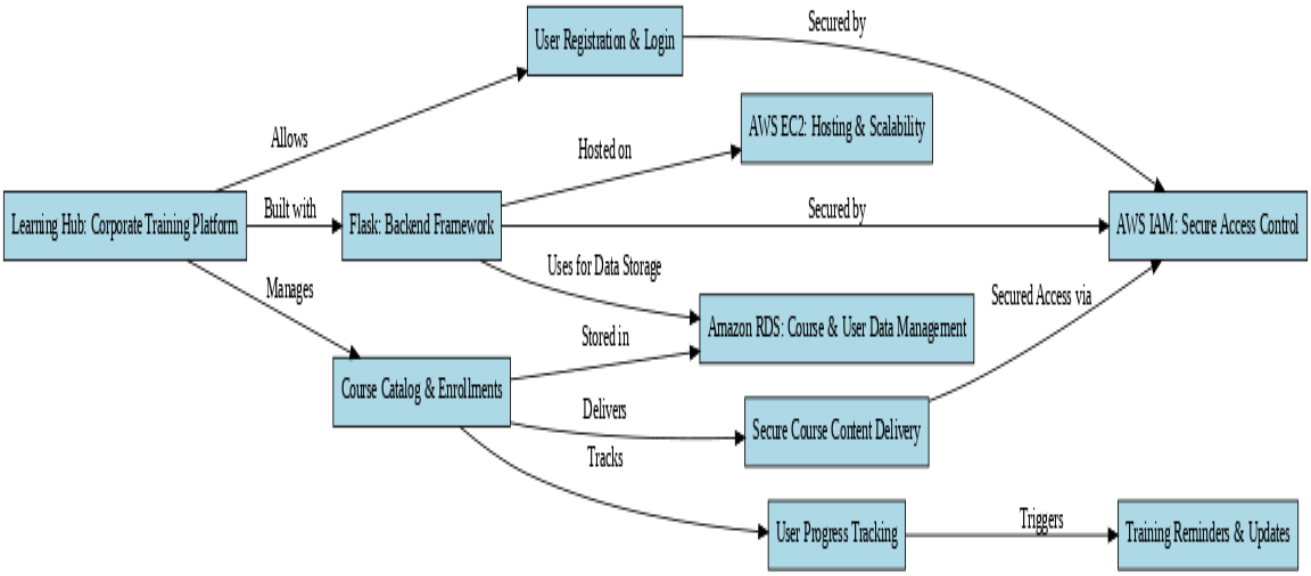
In a corporate training platform, managing course details, user progress, and content access is essential. Amazon RDS offers a managed MySQL database solution that simplifies these operations by providing automated backups, scaling, and high availability. For example, RDS manages user profiles, course enrollments, and progress tracking. By utilizing RDS for database management, the platform ensures consistent performance and data integrity as it

scales to accommodate a growing number of users and expanding course offerings, providing a seamless learning experience.

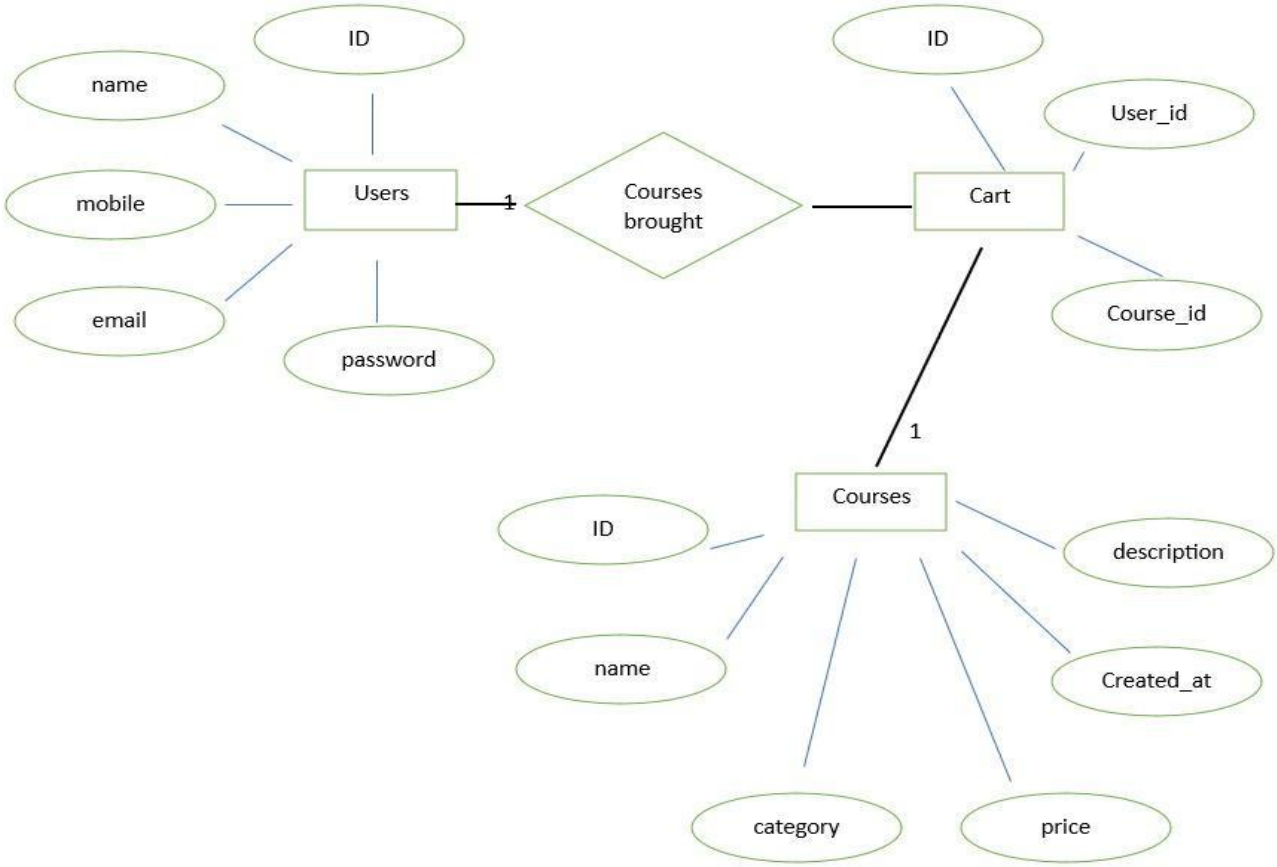
Scenario 3: Secure and Scalable Hosting for Tech Ed Applications

In Tech Ed platform, AWS EC2 provides a secure and scalable environment for hosting applications that manage course registrations, user progress, and content delivery. By integrating AWS IAM (Identity and Access Management) for access control, the platform ensures that only authorized users can access sensitive information such as user profiles and administrative functionalities. EC2's capability to scale dynamically based on user demand allows the training platform to efficiently handle the course enrollments and major course launches. This ensures a smooth and secure experience for all users.

TECHNICAL ARCHITECTURE:



Entity Relationship (ER) Diagram:



Project Flow:

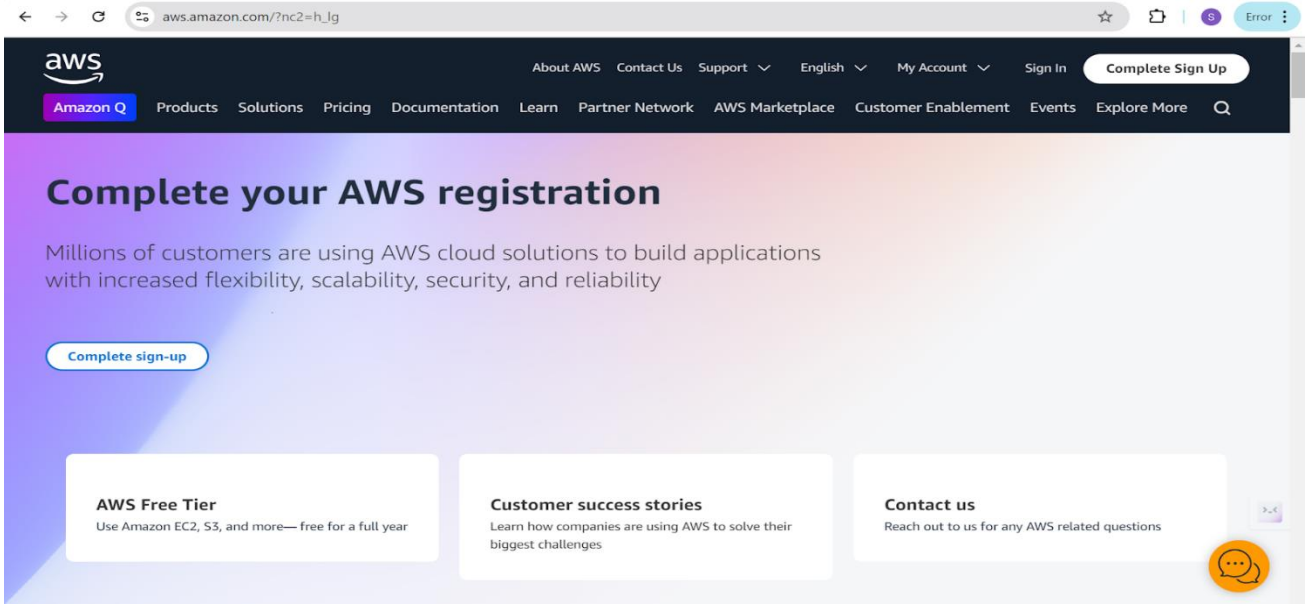
1. AWS Account Setup and Login
    - Set up an AWS account if not already done.
    - Log in to the AWS Management Console to manage resources.
2. RDS Database Creation and Setup
    - Activity 2.1: Create an RDS Instance.
        - Navigate to RDS service in the AWS Console and configure your database instance (select MySQL).
    - Activity 2.2: Install MySQL Workbench.
        - Download and install MySQL Workbench on your local machine.
        - Use the endpoint and credentials of your RDS instance to establish a connection via MySQL Workbench.

3. Frontend Development and Application Setup
    - Activity 3.1: Build the Frontend.
        - Develop the frontend for the Learning Hub using HTML, CSS, and Flask (Python-based).
        - Ensure the structure supports the Course purchase functionality and user interaction.
4. EC2 Instance Setup
    - Activity 4.1: Launch EC2 Instance.
        - From the AWS Console, launch a Linux-based EC2 instance.
        - SSH into the instance and prepare the environment for hosting the application.
5. Deployment on EC2
    - Activity 5.1: Deploy to EC2.
        - Transfer the developed Flask application to the EC2 instance.
        - Install necessary dependencies (e.g., Flask, MySQL libraries) on EC2.
        - Configure the EC2 instance to connect to the RDS database.
        - Start the Flask application on the instance.

6. Testing and Deployment
    - Activity 6.1: Functional Testing.
        - Test the full application for functionality including frontend interaction, database communication, and overall performance.
        - Run the Flask app with python app.py and access the link provided to verify its correct functioning.
    - Activity 6.2: Deployment.
        - Finalize the deployment in the production environment.
        - Ensure high availability, security, and performance optimization.

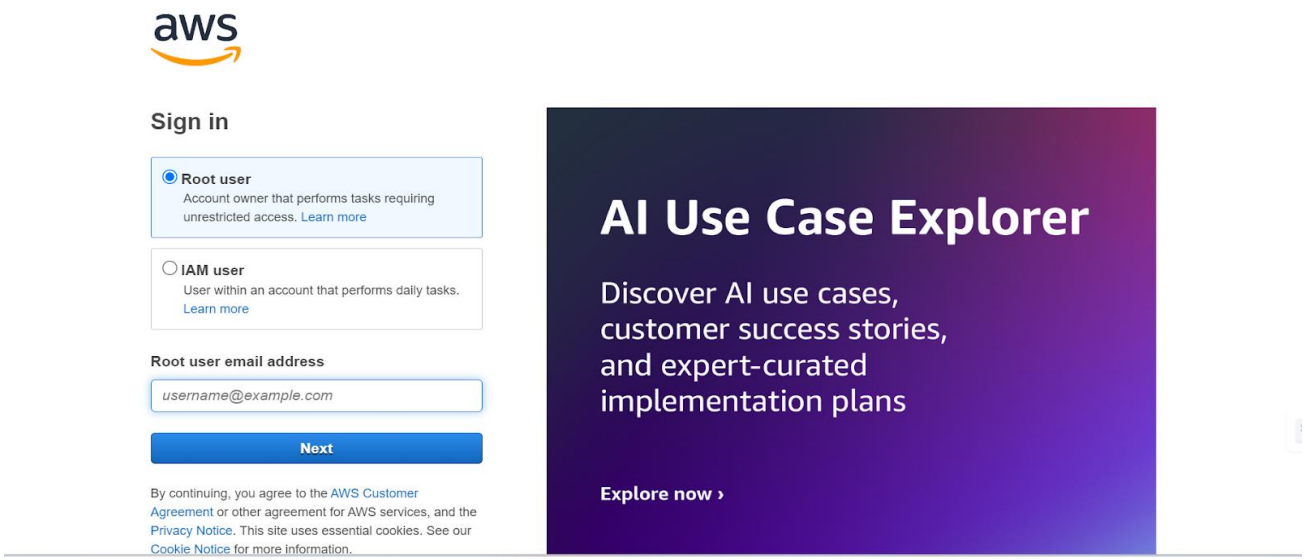
➢ Create AWS Account:-

- Sign up for an AWS account and configure billing settings.



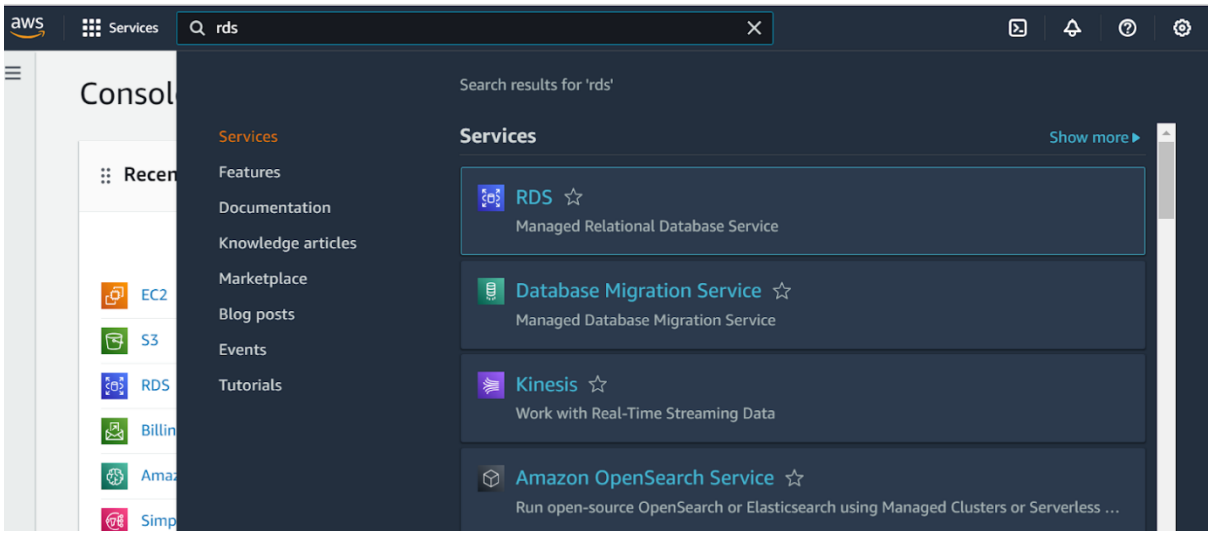
➢ Log in to AWS Management Console

- Access the AWS Management Console using your login credentials.

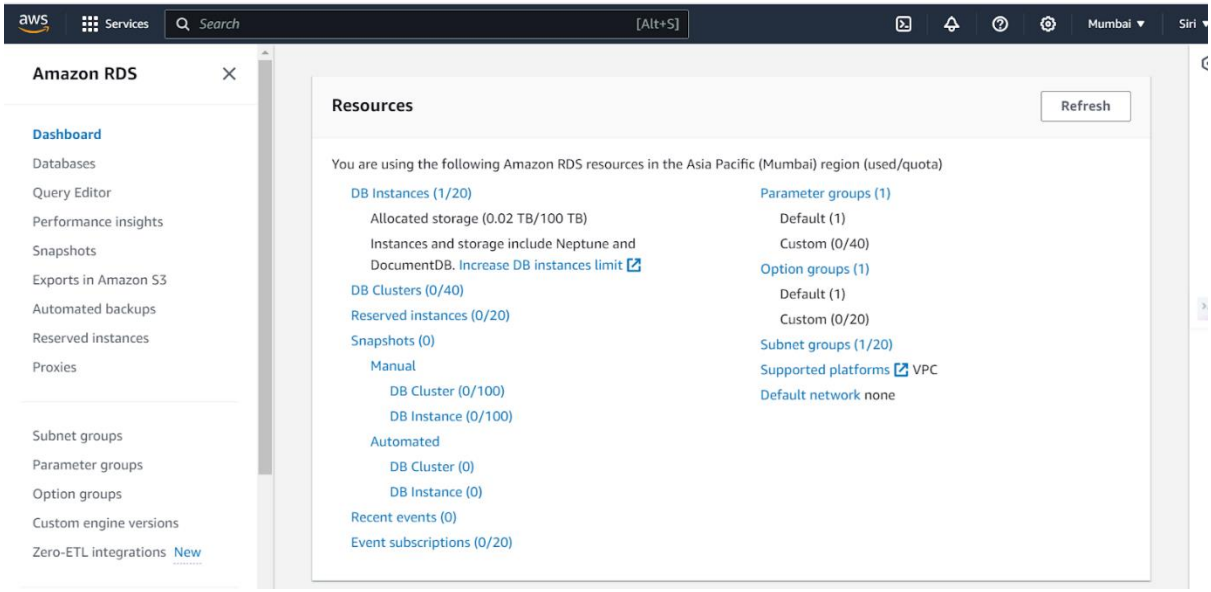


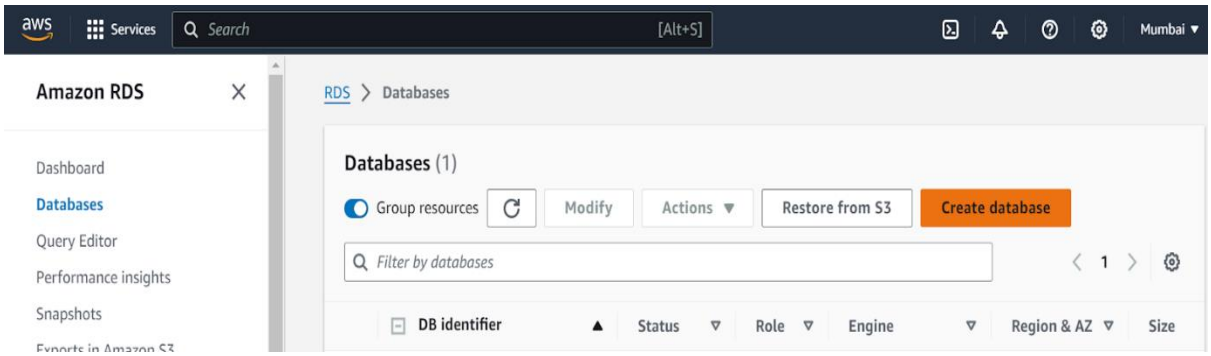
Create an RDS Instance

- Choose the RDS service from the AWS Management Console.



- Select MySQL as the database engine, configure the instance settings (e.g., storage, instance class), and launch the RDS instance.





aws Services Search [Alt+S]

RDS > Create database

Create database Info

Choose a database creation method

- (●) **Standard create**
  You set all of the configuration options, including ones for availability, security, backups, and maintenance.
- ( ) **Easy create**
  Use recommended best-practice configurations. Some configuration options can be changed after the database is created.

aws Services Search [Alt+S]

Engine options

Engine type Info

- ( ) Aurora (MySQL Compatible)
- ( ) Aurora (PostgreSQL Compatible)
- (●) MySQL
- ( ) MariaDB
- ( ) PostgreSQL
- ( ) Oracle

Edition

(●) MySQL Community

Engine version Info

View the engine versions that support the following database features.

▼ Hide filters

[ ] Show versions that support the Multi-AZ DB cluster Info

Create a A Multi-AZ DB cluster with one primary DB instance and two readable standby DB instances. Multi-AZ DB clusters provide up to 2x faster transaction commit latency and automatic failover in typically under 35 seconds.

[ ] Show versions that support the Amazon RDS Optimized Writes Info

Amazon RDS Optimized Writes improves write throughput by up to 2x at no additional cost.

Engine Version

MySQL 8.0.35 ▼

[ ] Enable RDS Extended Support Info

Amazon RDS Extended Support is a paid offering. By selecting this option, you consent to being charged for this offering if you are running your database major version past the RDS end of standard support date for that version. Check the end of standard support date for your major version in the RDS for MySQL documentation.

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.

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.

learnhub-8198

The DB instance identifier is case-insensitive, but is stored as all lowercase (as in "mydbinstance"). Constraints: 1 to 60 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 Neutral

Minimum constraints: At least 8 printable ASCII characters. Can't contain any of the following symbols: / ' " @

Confirm master password Info

••••••••

Instance configuration

The DB instance configuration options below are limited to those supported by the engine that you selected above.

DB instance class Info

▼ Hide filters

Show instance classes that support Amazon RDS Optimized Writes Info
Amazon RDS Optimized Writes improves write throughput by up to 2x at no additional cost.

Include previous generation classes

Standard classes (includes m classes)

Memory optimized classes (includes r and x classes)

Burstable classes (includes t classes)

Storage

Storage type Info

Provisioned IOPS SSD (io2) storage volumes are now available.

General Purpose SSD (gp3)
Performance scales independently from storage

Allocated storage Info

20 GiB

Minimum: 20 GiB. Maximum: 6,144 GiB

After you modify the storage for a DB instance, the status of the DB instance will be in storage-optimization. Your instance will remain available as the storage-optimization operation completes. Learn more

▶ **Advanced settings**

Baseline IOPS of 3,000 IOPS and storage throughput of 125 MiBps are included for allocated storage less than 400 GiB.

▶ **Storage autoscaling**

Compute resource

Choose whether to set up a connection to a compute resource for this database. Setting up a connection will automatically change connectivity settings so that the compute resource can connect to this database.

Don't connect to an EC2 compute resource
Don't set up a connection to a compute resource for this database. You can manually set up a connection to a compute resource later.

Connect to an EC2 compute resource
Set up a connection to an EC2 compute resource for this database.

Network type Info

To use dual-stack mode, make sure that you associate an IPv6 CIDR block with a subnet in the VPC you specify.

IPv4
Your resources can communicate only over the IPv4 addressing protocol.

Dual-stack mode
Your resources can communicate over IPv4, IPv6, or both.

Virtual private cloud (VPC) Info

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

vpc-0c9a09cc4c0ef9dd5
3 Subnets, 3 Availability Zones

Only VPCs with a corresponding DB subnet group are listed.

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

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.

VPC security group (firewall) Info

Choose one or more VPC security groups to allow access to your database. Make sure that the security group rules allow the appropriate incoming traffic.

- ( ) Choose existing
  Choose existing VPC security groups
- (●) Create new
  Create new VPC security group

New VPC security group name

new security group

Availability Zone Info

No preference

RDS Proxy

RDS Proxy is a fully managed, highly available database proxy that improves application scalability, resiliency, and security.

- [ ] Create an RDS Proxy Info
  RDS automatically creates an IAM role and a Secrets Manager secret for the proxy. RDS Proxy has additional costs. For more information, see Amazon RDS Proxy pricing.

Database authentication

Database authentication options Info

- (●) Password authentication
  Authenticates using database passwords.
- ( ) Password and IAM database authentication
  Authenticates using the database password and user credentials through AWS IAM users and roles.
- ( ) Password and Kerberos authentication
  Choose a directory in which you want to allow authorized users to authenticate with this DB instance using Kerberos Authentication.

Monitoring

- [ ] Enable Enhanced Monitoring
  Enabling Enhanced Monitoring metrics are useful when you want to see how different processes or threads use the CPU.

▶ Additional configuration

Database options, encryption turned on, backup turned on, backtrack turned off, maintenance, CloudWatch Logs, delete protection turned off.

This billing estimate is based on on-demand usage as described in Amazon RDS Pricing. Estimate does not include costs for backup storage, IOs (if applicable), or data transfer.

Estimate your monthly costs for the DB Instance using the AWS Simple Monthly Calculator.

Estimated monthly costs

The Amazon RDS Free Tier is available to you for 12 months. Each calendar month, the free tier will allow you to use the Amazon RDS resources listed below for free:

- 750 hrs of Amazon RDS in a Single-AZ db.t2.micro, db.t3.micro or db.t4g.micro Instance.
- 20 GB of General Purpose Storage (SSD).
- 20 GB for automated backup storage and any user-initiated DB Snapshots.

Learn more about AWS Free Tier.

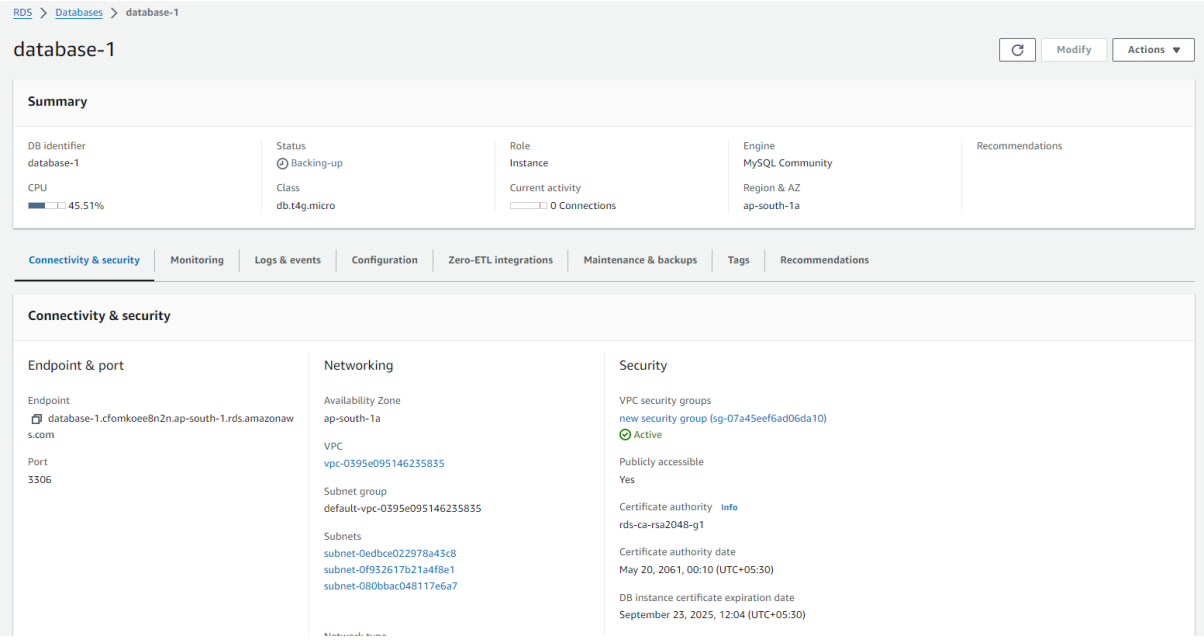
When your free usage expires or if your application use exceeds the free usage tiers, you simply pay standard, pay-as-you-go service rates as described in the Amazon RDS Pricing page.

You are responsible for ensuring that you have all of the necessary rights for any third-party products or services that you use with AWS services.

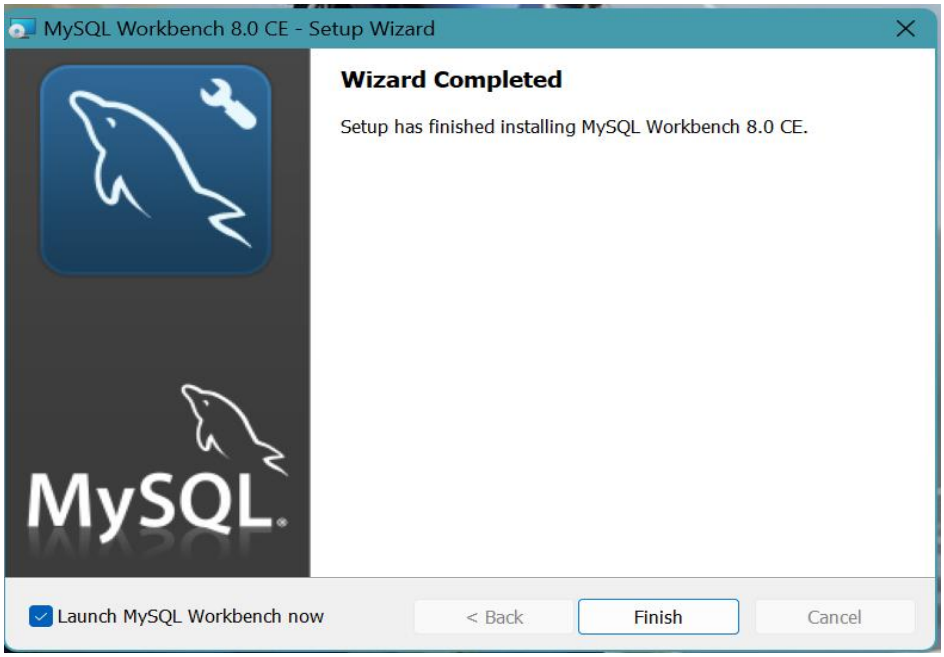
Cancel | Create database

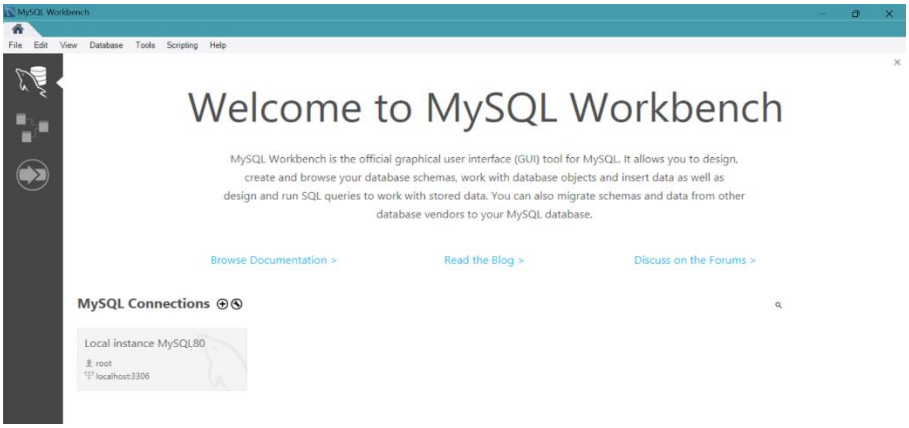
➢ Configure Database Access:-

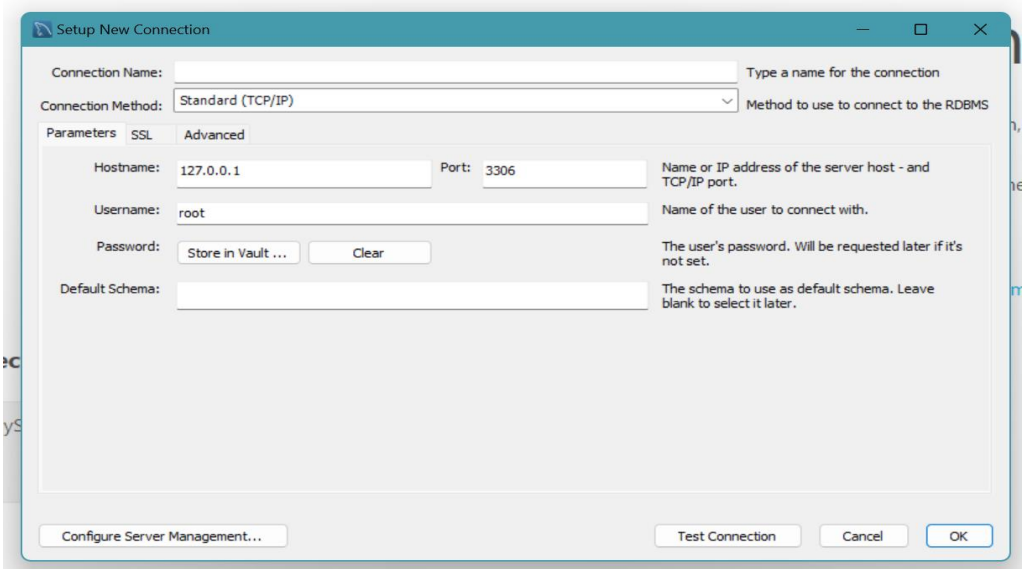
Set up security groups, create database credentials, and configure access policies to ensure secure connectivity to the database.



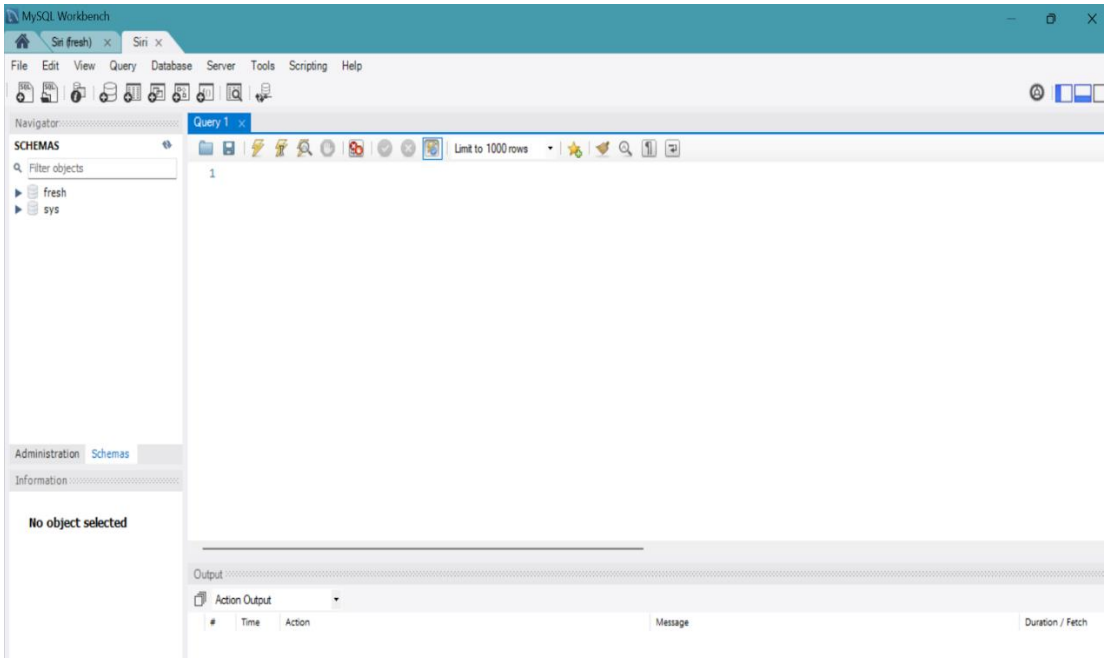
Install MySQL Workbench

- Download and install MySQL Workbench on your local machine for database management.
- Connect to the RDS instance via MySQL Workbench using the endpoint and credentials from AWS
- 





- Give a connection name.
- Copy the endpoint from the RDS database that is created in AWS and paste it in Hostname.
- Write the username and enter the password , then click on Test Connection.
- Once the connection is successful, you'll be welcomed with this interface



➢ Create the Database and the tables which are required.

Create a basic database schema for an e-commerce platform

```
Query 1
Limit to 1000 rows
1 • create database learnhub;
2 • use learnhub;
```

users:

- Stores user information such as name, email, password, and mobile number.
- Each user has a unique ID (id), which serves as the primary key.
- The email column is unique to ensure no duplicate accounts.

```
CREATE TABLE users (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    mobile VARCHAR(15) NOT NULL,
    email VARCHAR(100) NOT NULL UNIQUE,
    password VARCHAR(255) NOT NULL
);
```

Courses:

- The Name column stores the title of the course and is a required field (NOT NULL).
- The Category field indicates the category to which the course belongs, helping users easily navigate and find relevant courses; it is also a required field (NOT NULL).
- The Price column records the cost of the course in a decimal format, ensuring precise financial representation, and is a mandatory field (NOT NULL).
- The Description is an optional column that provides a detailed overview of the course content, objectives, and any prerequisites, helping users make informed purchasing decisions.
- The Created At column automatically logs the timestamp when the course is added to the database, using the current timestamp by default.
- The Updated At column records the timestamp whenever the course information is modified, ensuring that the latest updates are tracked automatically.

```sql
CREATE TABLE courses (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(255) NOT NULL,
    category VARCHAR(100) NOT NULL,
    price DECIMAL(10, 2) NOT NULL,
    description TEXT,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
);
```

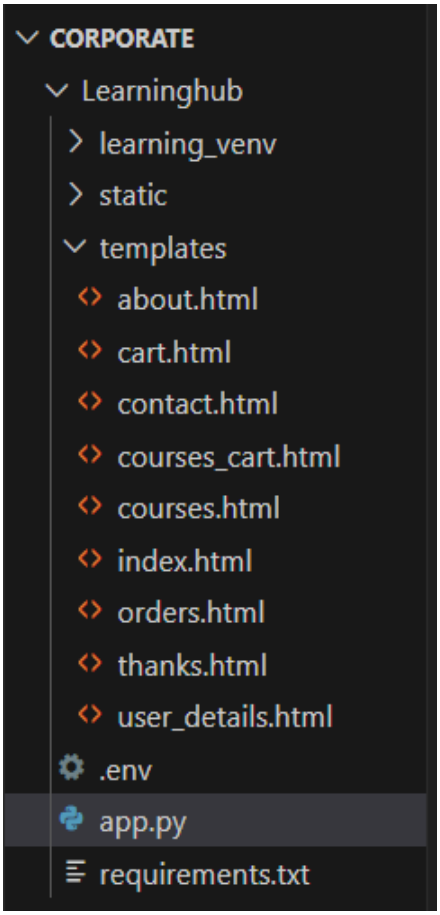
Data Operations:

1. Data Retrieval: We retrieved data from each table to view the stored information, including user details, available items, and orders.

```sql
select * from users;
select * from cart;
select * from courses;
```

➢ Build the Frontend:-

- Develop HTML, CSS, and Python-based Flask application files for Learninghub frontend interface.



Integrate Application with RDS

- Connect app.py (Flask application) to the MySQL RDS database by configuring database connection settings and verifying connectivity.

Description of the code :

1.Flask App Initialization: Initializes a Flask application with secret key for sessions.

```
from flask import Flask, render_template, request, redirect, url_for, flash, session, jsonify
import pymysql
import bcrypt
import logging

app = Flask(__name__)
app.secret_key = "supersecretkey"
```

2.Database Configuration: Configures MySQL RDS with connection pooling for efficient database access.

```
db_config = {
    'host': 'learnhub-8198.c5ukqgu4mk81.ap-south-1.rds.amazonaws.com',
    'user': 'admin',
    'password': 'tanirstanir',
    'database': 'learnhub'
}
```

3.Connection Pool: Uses MySQL connection pooling to handle multiple database connections.

```
# Function to get a database connection
def get_db_connection():
    try:
        connection = pymysql.connect(**db_config)
        logging.info("Connected to the database")
        return connection
    except pymysql.MySQLError as e:
        logging.error(f"Database connection failed: {e}")
        return None
```

4.Home Route: Renders the home page template when the root URL is accessed.

5.Register Route (GET/POST): Handles user registration, inserts user data into the database.

```python
@app.route('/signup', methods=['GET', 'POST'])
def signup():
    if request.method == 'POST':
        name = request.form['signupName']
        mobile = request.form['signupNumber']
        email = request.form['signupEmail']
        password = request.form['signupPassword'].encode('utf-8')
        hashed_password = bcrypt.hashpw(password, bcrypt.gensalt())

        connection = get_db_connection()
        if connection:
            cursor = connection.cursor()
            try:
                cursor.execute("INSERT INTO users (name, mobile, email, password) VALUES (%s, %s, %s,
                                (name, mobile, email, hashed_password))
                connection.commit()
                flash('Account created successfully!', 'success')
                return redirect(url_for('login'))  # Redirect to login after signup
            except pymysql.MySQLError as e:
                flash('Error creating account. Email may already exist.', 'error')
                logging.error(f"Error: {e}")
            finally:
                cursor.close()
                connection.close()

    return render_template('user_details.html')
```

6.Login Route (GET/POST): Authenticates user with email and password, creates session on success.

```python
@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        email = request.form['loginEmail']
        password = request.form['loginPassword'].encode('utf-8')

        connection = get_db_connection()
        if connection:
            cursor = connection.cursor()
            try:
                cursor.execute("SELECT id, password FROM users WHERE email = %s", (email,))
                result = cursor.fetchone()

                if result and bcrypt.checkpw(password, result[1].encode('utf-8')):
                    session['user_id'] = result[0]  # Store user ID in session
                    flash('Login successful!', 'success')
                    print(result[0])
                    return redirect(url_for('courses_cart', user_id=result[0]))

                else:
                    flash('Invalid credentials. Please try again.', 'error')
            finally:
                cursor.close()
                connection.close()

    return render_template('user_details.html')
```

7.Courses_cart: Shows available courses, and adds the course to the cart

```python
@app.route('/courses_cart', methods=['GET'])
def courses_cart():
    connection = get_db_connection()
    courses = []

    if connection:
        cursor = connection.cursor()
        try:
            # Fetch courses from the database
            cursor.execute("SELECT id, name, category, price, description FROM courses")
            courses = cursor.fetchall()  # Fetch all courses as a list of tuples
        except pymysql.MySQLError as e:
            flash('Error fetching courses from the database.', 'error')
            print(f"Error: {e}")
        finally:
            cursor.close()
            connection.close()
    else:
        flash('Could not connect to the database.', 'error')

    # Pass the courses to the template
    return render_template('courses_cart.html', courses=courses)
```

8.Payment Page: Adds selected Courses , to the user's booking details.

```python
def insert_order(user_email, course_name, course_price):
    connection = get_db_connection()
    cursor = connection.cursor()

    try:
        insert_query = """
        INSERT INTO Orders (UserEmail, CourseName, CoursePrice)
        VALUES (%s, %s, %s)
        """
        # Execute query
        cursor.execute(insert_query, (user_email, course_name, course_price))

        # Commit the transaction
        connection.commit()
        print("Order inserted successfully!")
    except Exception as e:
        print(f"Error: {e}")
        connection.rollback()
    finally:

        cursor.close()
        connection.close()
```

9.Thank you route: Shows the message “Thanks for Booking!” and redirects to the home page.

```python
@app.route('/thanks')
def thanks():
    return render_template('thanks.html')

if __name__ == '__main__':
    app.run(debug=True)
```

10.Database Queries: Uses SQL queries to interact with MySQL RDS for room availability, bookings, and customer data.

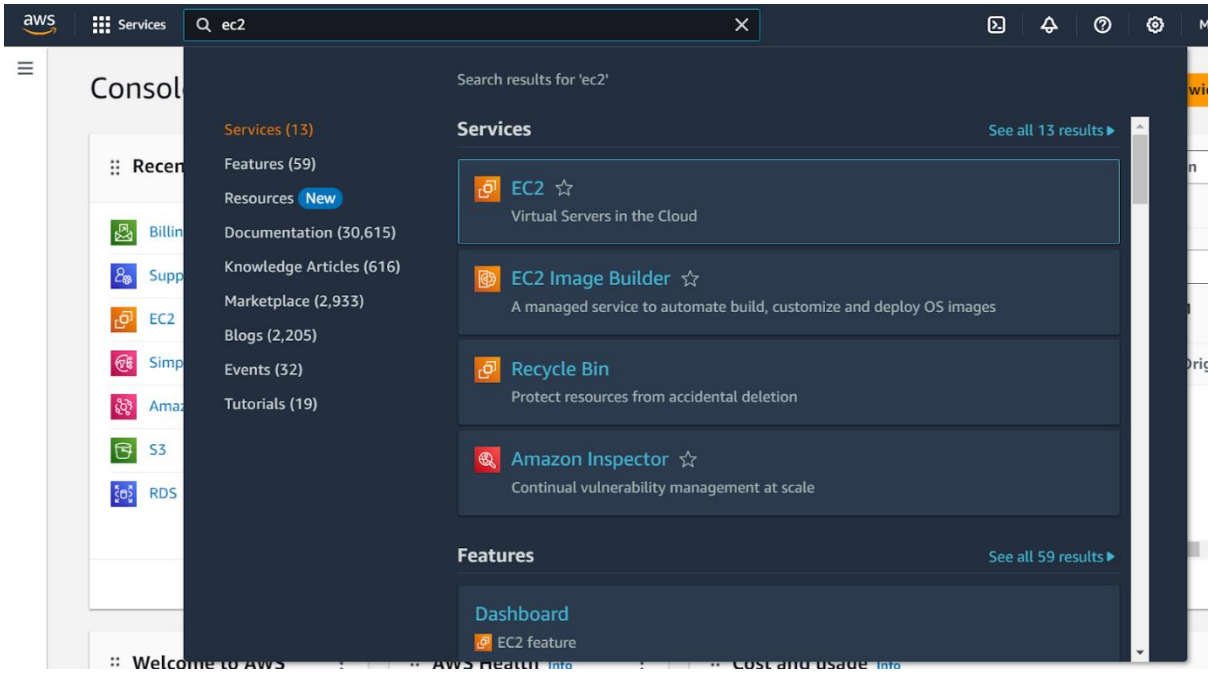
11.Session Management: Uses Flask sessions to store user preferences and booking details for seamless navigation.

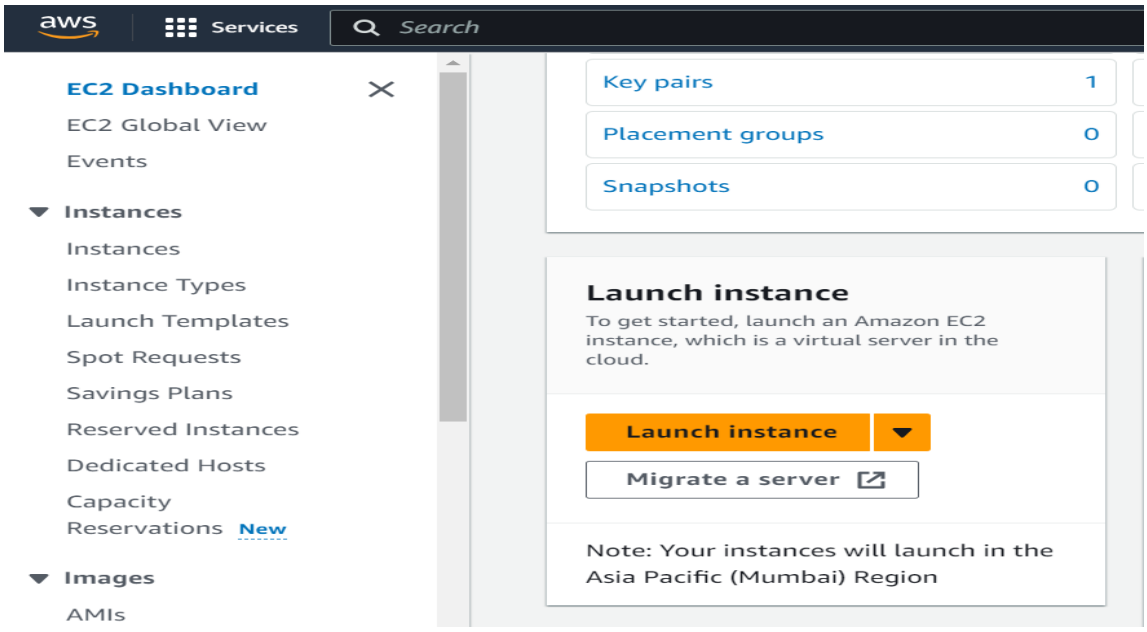
12.Flash Messages: Provides user feedback through flash messages for login, registration, booking confirmation, and updates.

13.Room Data Fetching: Retrieves room types, prices, and availability from the database to display on the booking interface.

➢ Launch EC2 Instance:-

- Choose a Linux-based EC2 instance from the AWS Console to host the FreshBasket application.





EC2 > Instances > Launch an instance

Launch an instance Info

Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

Name and tags Info

Name

learnhub

Add additional tags

Application and OS Images (Amazon Machine Image) Info

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below

Search our full catalog including 1000s of application and OS images

Amazon Linux | macOS | Ubuntu | Windows | Red Hat

Browse more AMIs

Including AMIs from AWS, Marketplace and the Community

Amazon Machine Image (AMI)

Amazon Linux 2023 AMI — Free tier eligible

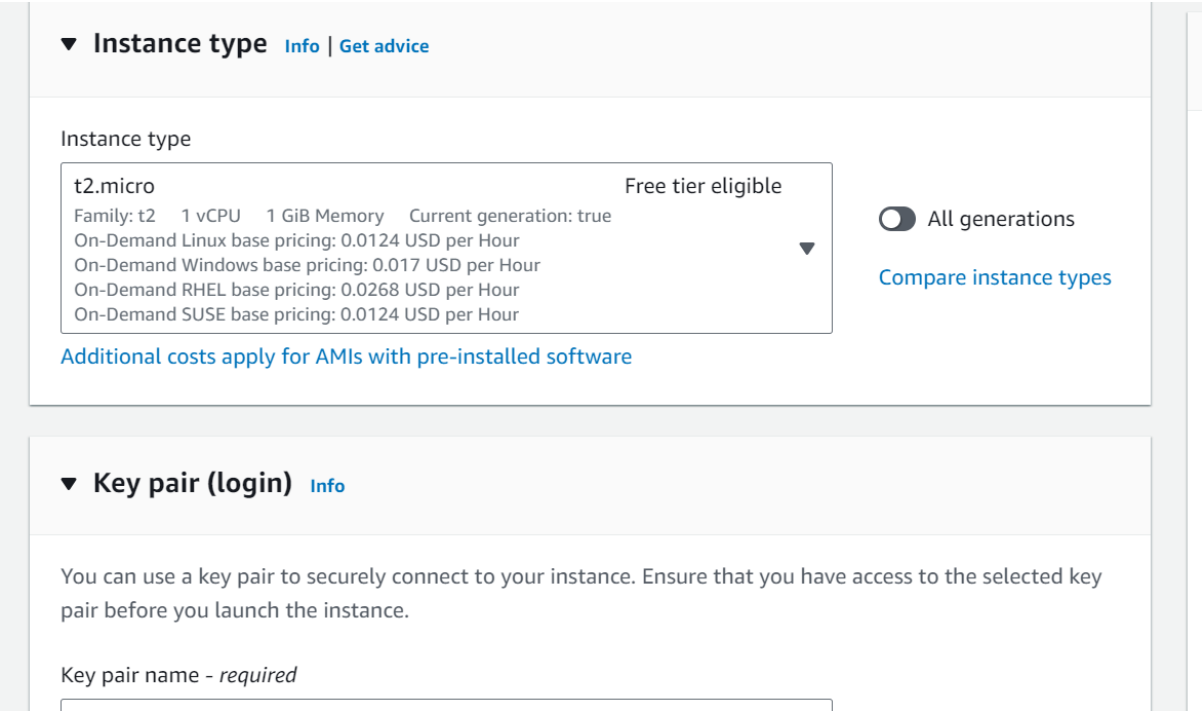
ami-02b49a24cfb95941c (64-bit (x86), uefi-preferred) / ami-04ad8c7fcc828fad4 (64-bit (Arm), uefi)

Virtualization: hvm ENA enabled: true Root device type: ebs

Description

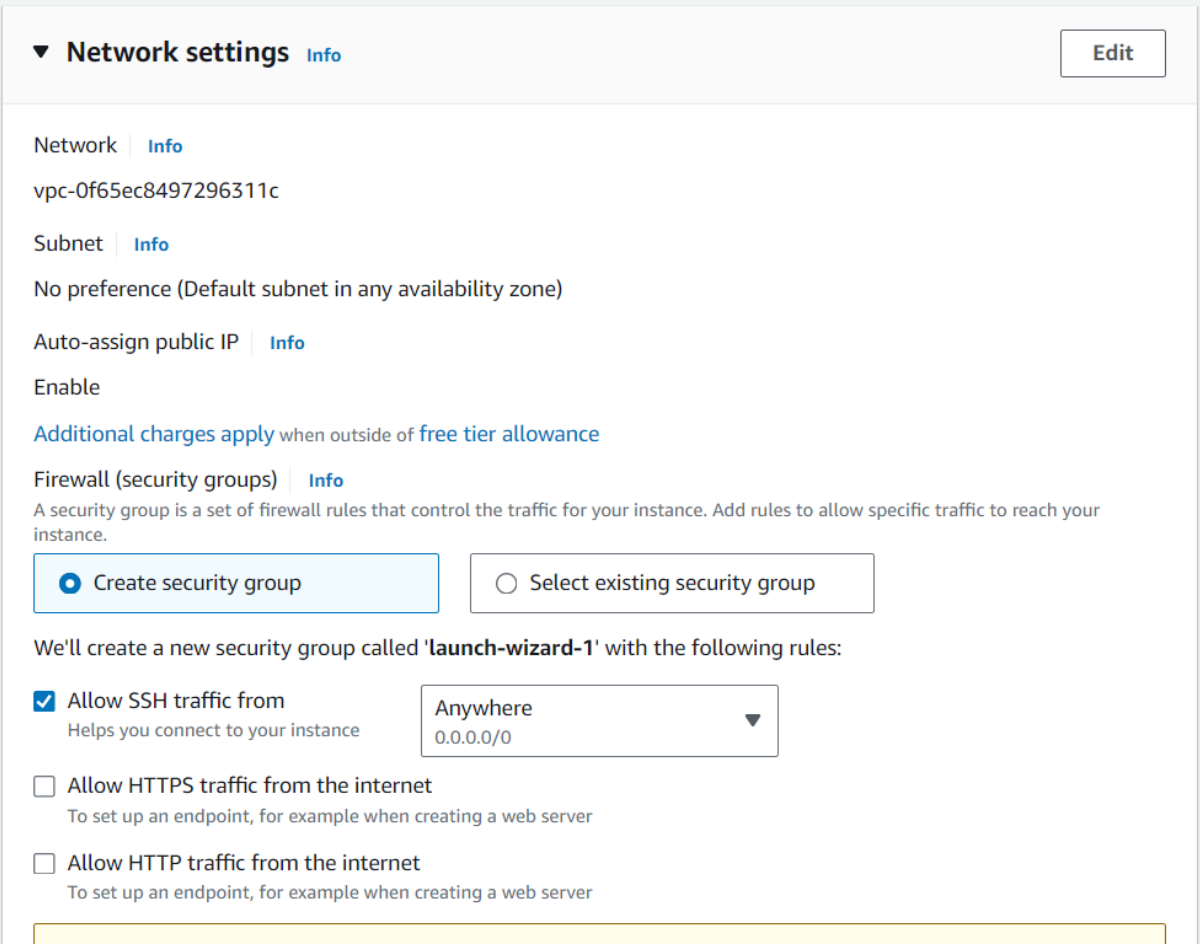
Amazon Linux 2023 is a modern, general purpose Linux-based OS that comes with 5 years of long term support. It is optimized for AWS and designed to provide a secure, stable and high-performance execution environment to develop and run your cloud applications.

Architecture	Boot mode	AMI ID	
64-bit (x86)	uefi-preferred	ami-02b49a24cfb95941c	Verified provider

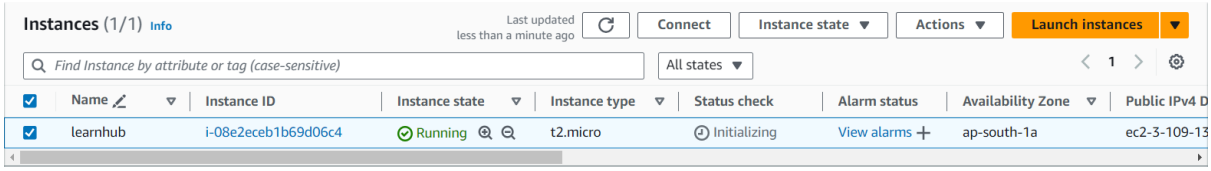


➢ Configure Network Settings:-

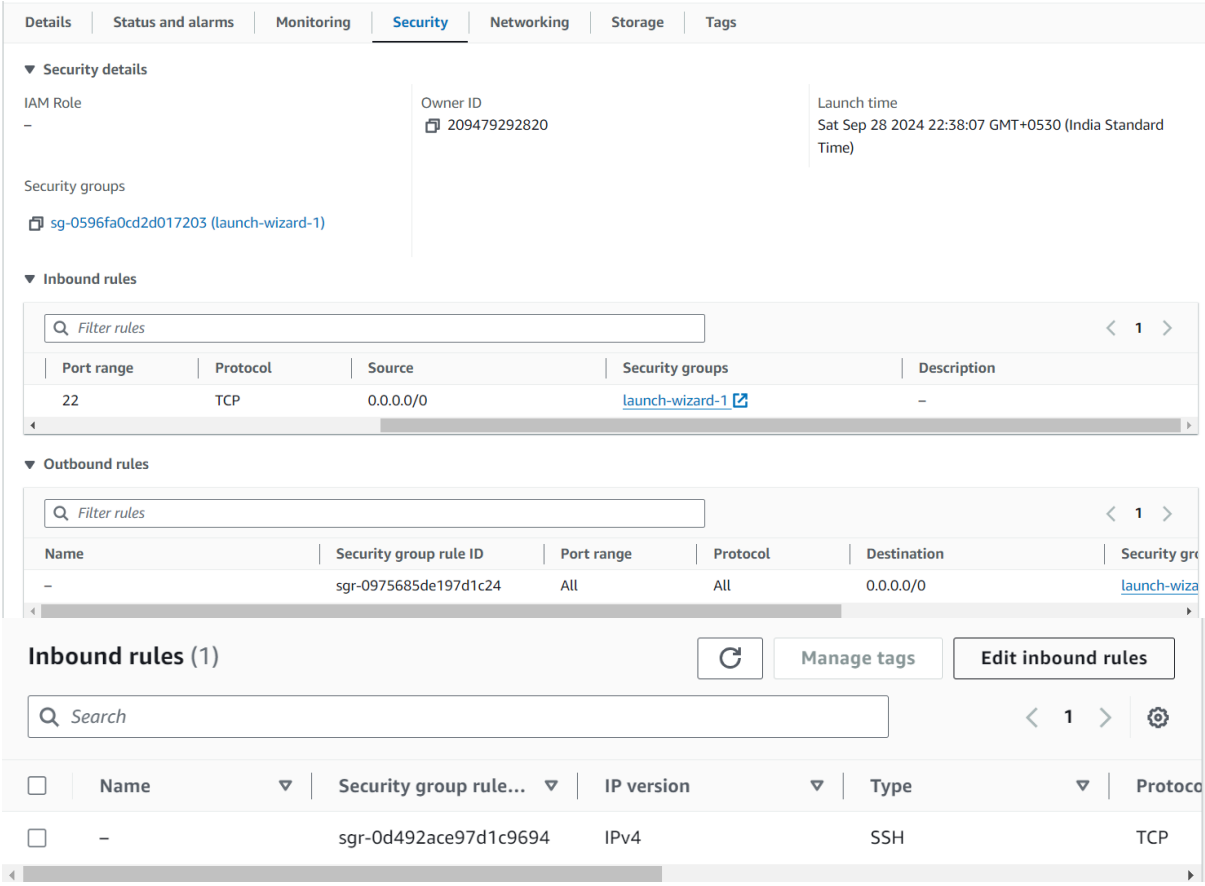
- Set up the security group to allow HTTP, HTTPS, and SSH traffic.

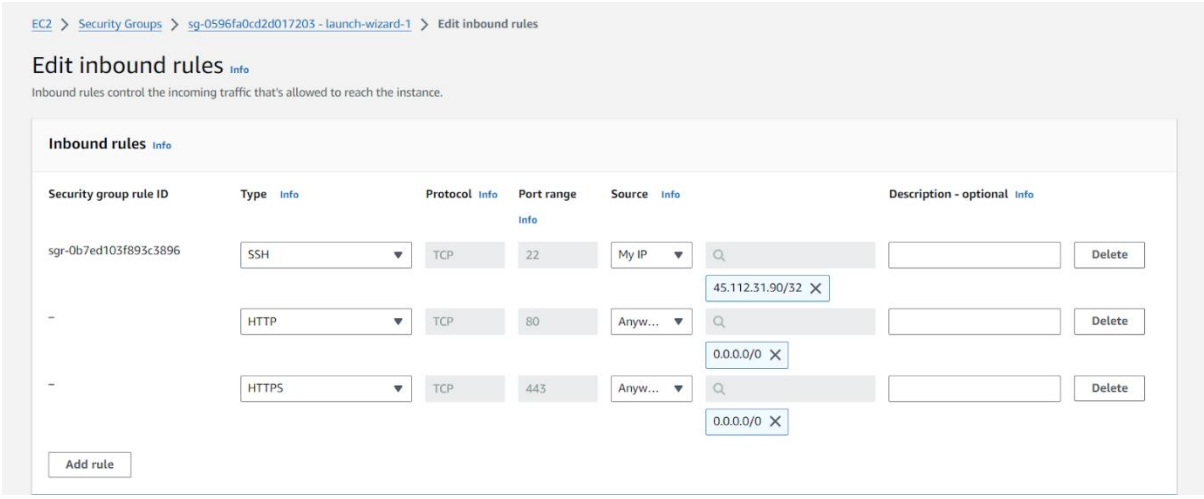


- Create and download the key pair for SSH access.

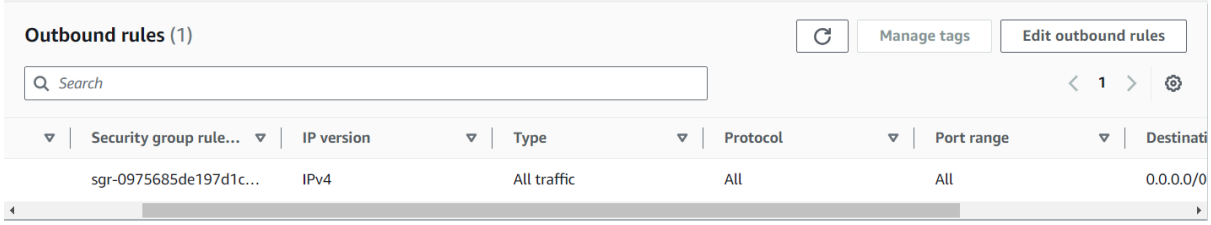


Setting up Inbound and Outbound rules



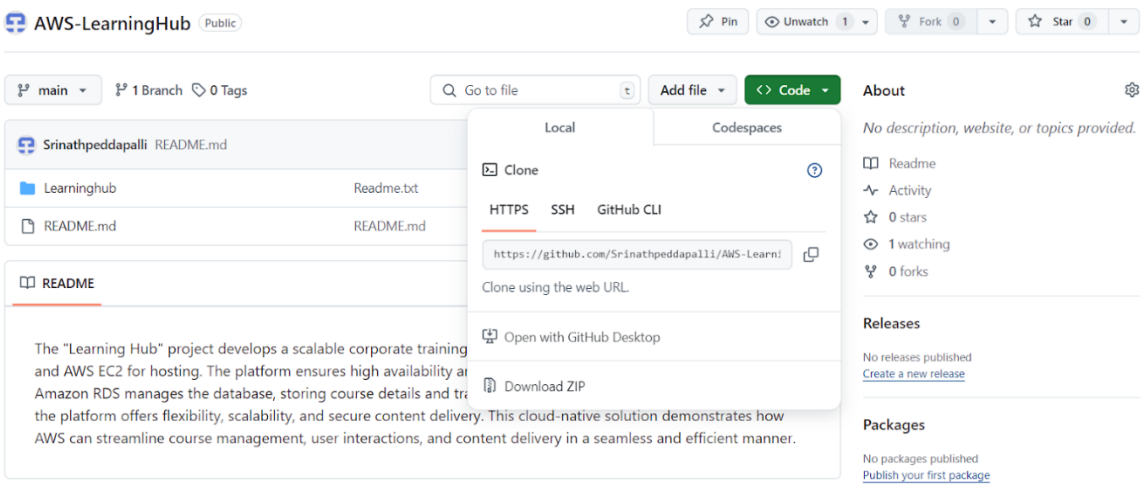


- Add Type : HTTP > Source : Anywhere
- Add Type : HTTPS > Source : Anywhere



➢ Deploy to EC2:-

1. Transfer your application code to the EC2 instance.
2. Set up any necessary environment variables, including database connection strings.
3. Configure the web server to serve your application.
4. Start your application and ensure it's accessible via the EC2 instance's public IP or domain.
5. Run the below commands on ec2 terminal
6. sudo yum update -y
7. sudo yum install python3 -y
8. sudo pip3 install virtualenv
9. python3 -m venv
10. source venv/bin/activate
11. pip install flask
12. git clone https://github.com/Srinathpeddapalli/AWS-LearningHub.git



13. cd your-flask-app

14.python3 app.py

➢ Functional Testing:-

- Test the Learning Hub application for functionality, including database interactions and frontend features.
- Run the Flask app python app.py
- It will give you the link

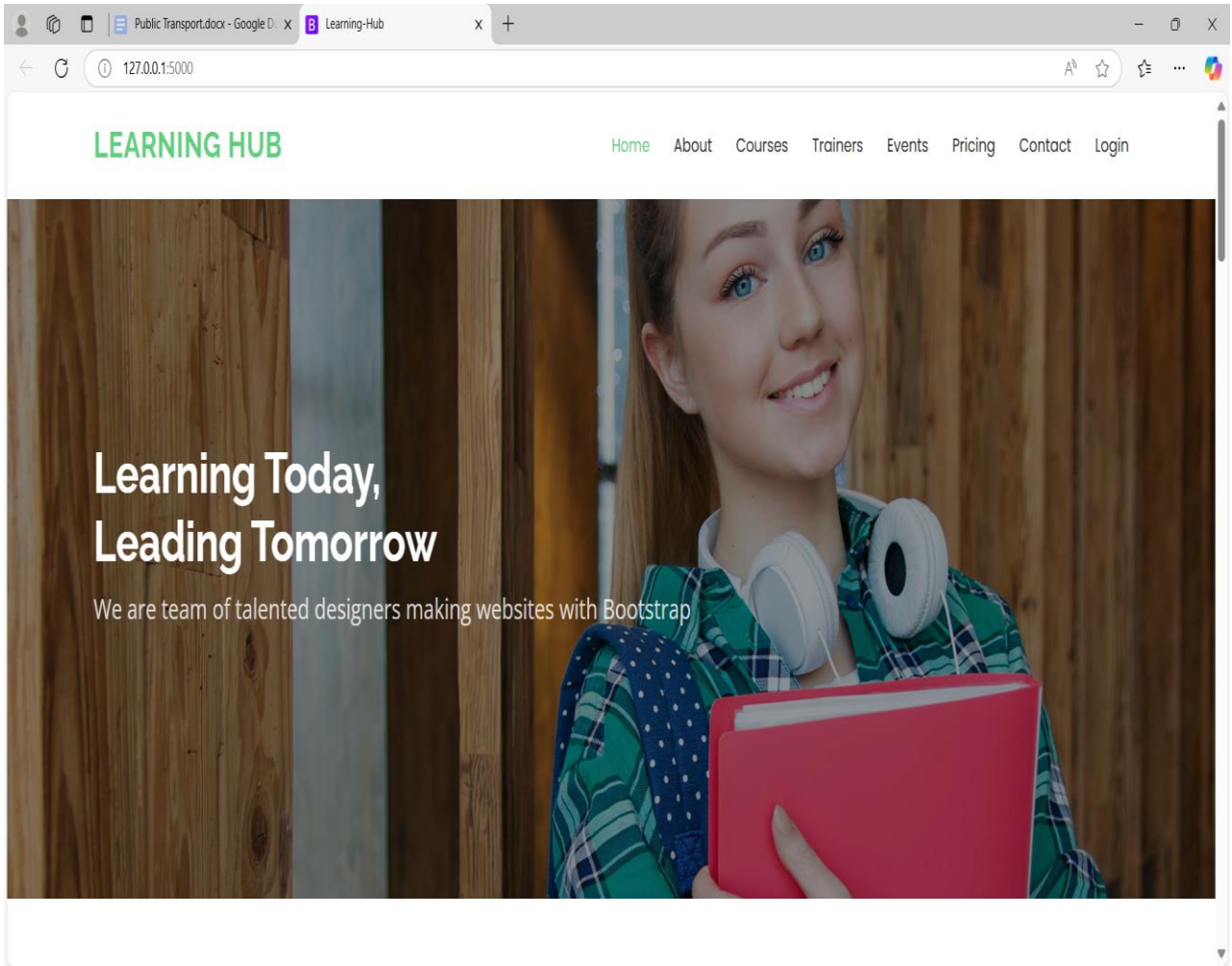
```
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with watchdog (windowsapi)
 * Debugger is active!
 * Debugger PIN: 849-618-898
```

➢ Deployment:-

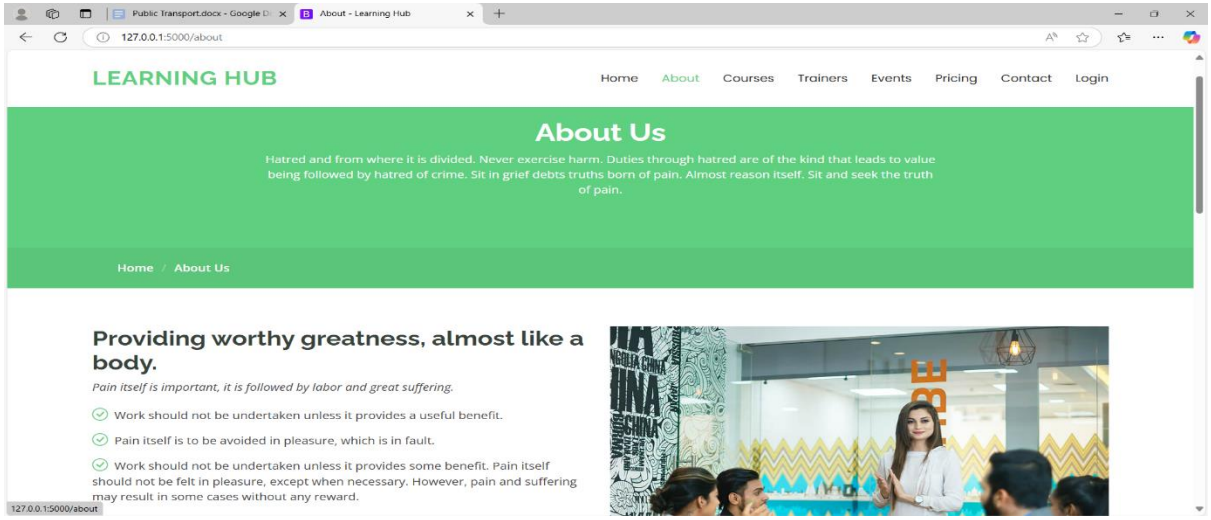
- Deploy the application in a production environment, ensuring high availability and performance.

Click on the link above and it will take you to the webpage:

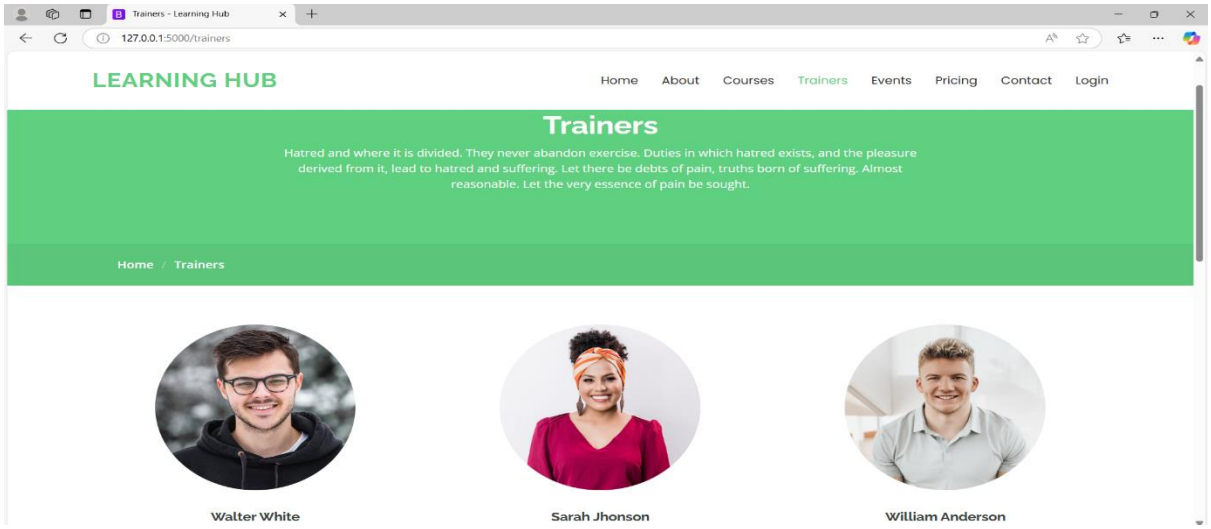
Home page:



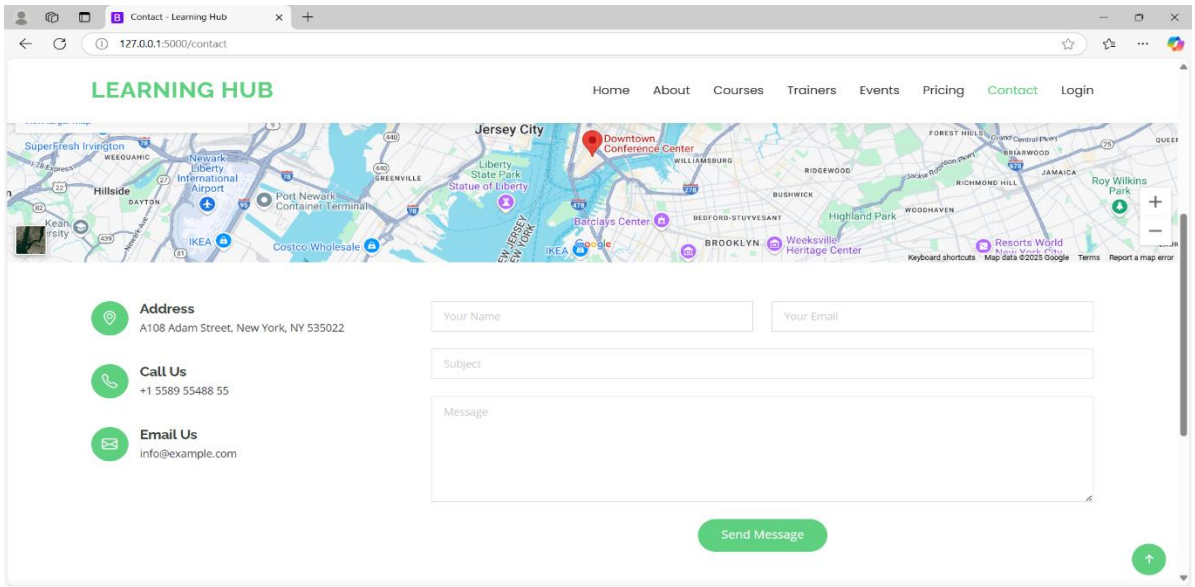
About:



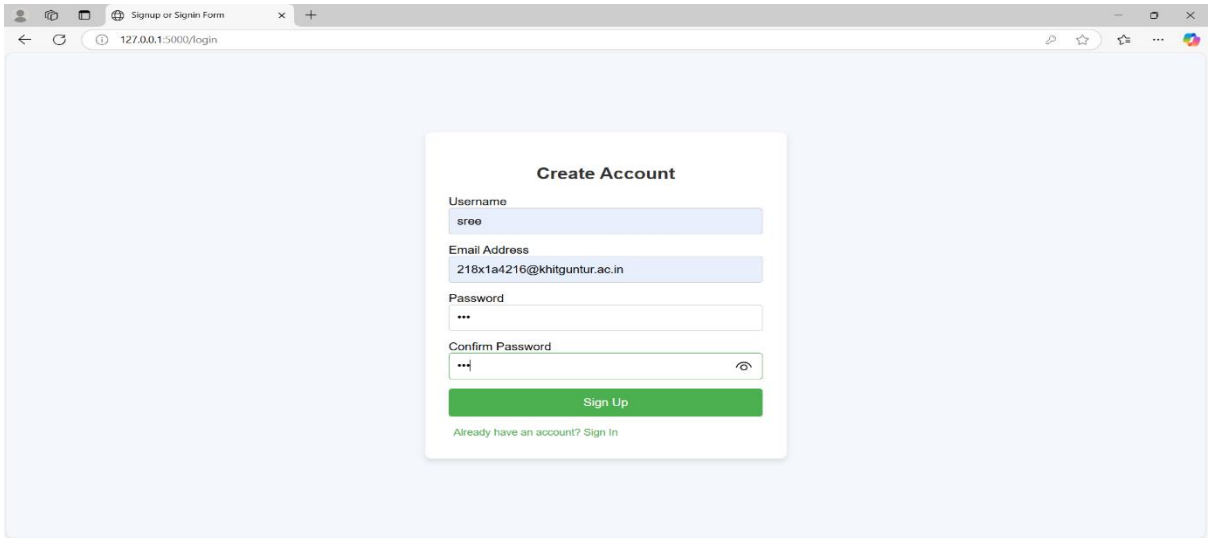
Trainers:



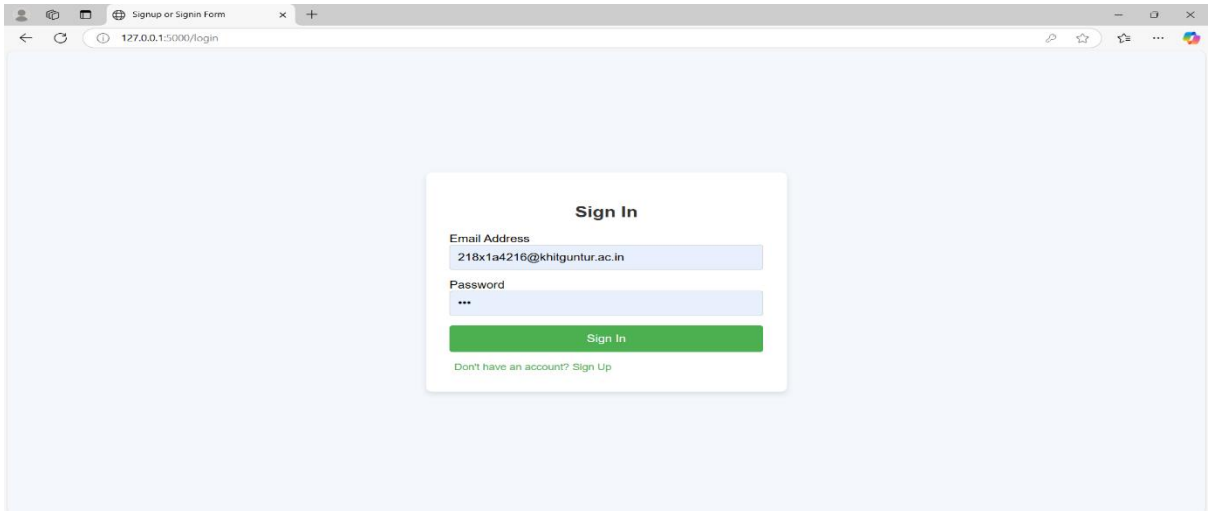
Contact:



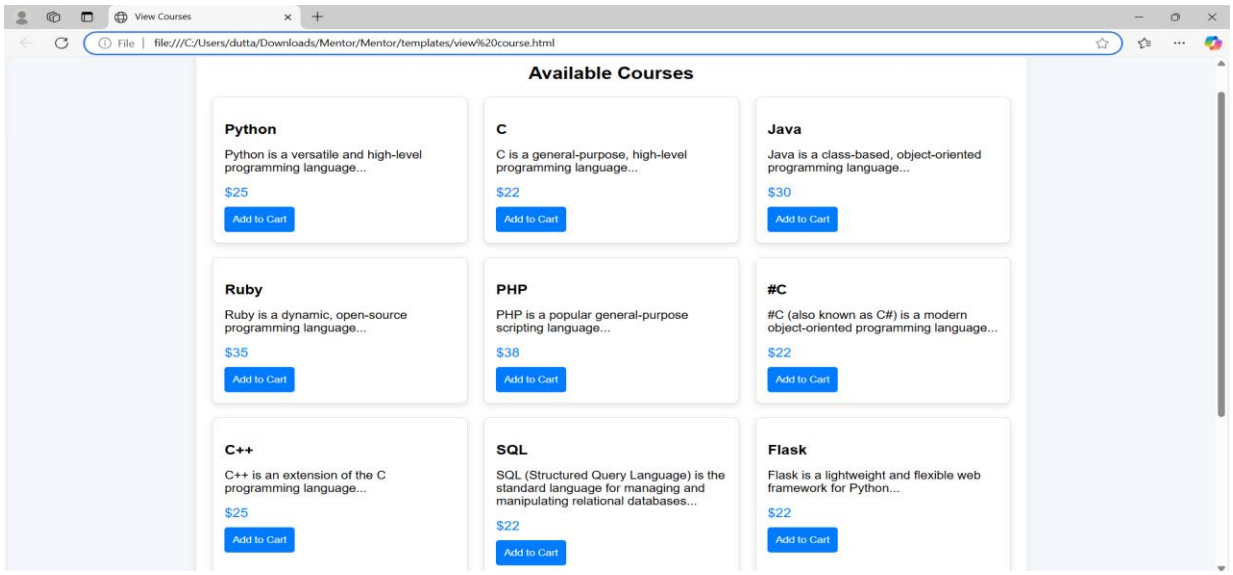
Signup:



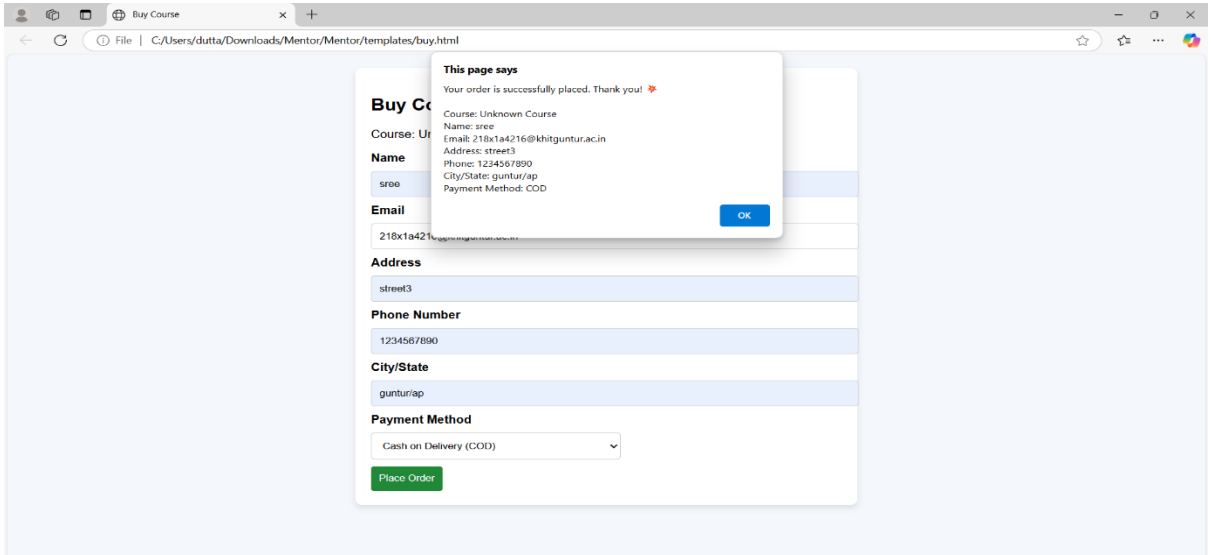
Signin:



Courses:



Cart:



Check the updations in the mysql database.

MySql Database updations:

1. **Users table :**

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

id	name	mobile	email	password
1	Srinath	8885560331	srinathchary30@gmail.com	$2b$12$A.PT9/ySv2DvKVv/.9asqOIH6mFiR7hb...
2	sai kumar	7878787878	sai@gmail.com	$2b$12$0v0o/je/4wns6gWMJ.ri1uRijUbmaCyhI...
3	shiva	8989898989	shiva@gmail.com	$2b$12$iuzWeVpl0/xuguTiBLmR8OcYErxmRdML...
4	ajay	8787878787	ajay@gmail.com	$2b$12$KqgvH3FQSy71hSo4XnkgNu3Mv1UHM...
5	vijaya	8585858585	vijaya@gmail.com	$2b$12$5ogi1WcFInRlP3JB3TQ/D.WXWJ29VeS...

2. **cart table :**

Result Grid | Filter Rows:

id	user_id	course_id
1	2	1
2	3	1
NULL	NULL	NULL

In the your orders section,it will display the purchased course of the particular user

```sql
SELECT
    cart.id AS cart_id,
    users.id AS user_id,
    courses.name AS course_name,
    courses.category AS course_category,
    courses.price AS course_price
FROM cart
JOIN users ON cart.user_id = users.id
JOIN courses ON cart.course_id = courses.id;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

cart_id	user_id	course_name	course_category	course_price
1	2	Website Design	Web Development	99.00
2	3	Website Design	Web Development	99.00

Performance Monitoring

- Set up AWS CloudWatch for monitoring EC2 and RDS performance metrics.
- Implement alerts and notifications for critical performance thresholds.

Optimization

- Optimize the server and database configurations based on monitoring results, including adjusting instance types and query optimization.

Conclusion:

The Learning Hub platform project effectively demonstrates how AWS services can be utilized to create a scalable and efficient system. By combining Flask with Amazon RDS, the platform ensures smooth handling of course management, user progress tracking, and content access. Hosting on AWS EC2 allows for seamless deployment and remote management, with tools like MobaXterm simplifying administrative tasks. Each step, from database setup to backend integration, was designed with performance and reliability in mind. Additionally, AWS CloudWatch provides real-time monitoring to ensure uninterrupted operation. Overall, the project highlights how AWS can support robust and scalable corporate training solutions.