PES UNIVERSITY, BANGALORE
Department of Computer Science and Engineering

B.TECH. (CSE)
VI SEMESTER

UE20CS351 – Cloud Computing
Mini-Project Report
on
Project – 3
Deployment of Web App Using AWS Cloud
April - 2023

SUBMITTED BY

1. Utkarsh Bagaria – PES1UG20CS477
2. Tanvi Rajesh – PES1UG20CS461
3. Sunag – PES1UG20CS444
4. Srujan – PES1UG20CS436

VI Semester H Section

PES University

PES UNIVERSITY, BANGALORE
Department of Computer Science and Engineering

Short Description and Scope of the Project

- The aim of this project is to develop a web application, connect it to a database hosted on AWS RDS, and deploy it on an AWS EC2 instance using GitHub actions and Nginx.
- Prior knowledge regarding web development with React, Git/GitHub, AWS, and database fundamentals in SQL was helpful while doing the project.
- The web application should have a basic CRUD (Create, Read, Update and Delete) functionality, and a proper database schema.
- AWS Budgets should be set up to ensure usage within the free tier. This is ensured by using RDS instances like db.t2.micro, db.t3.micro, or db.t4g.
- The deployment should include integration with GitHub actions and be deployed with Nginx.
- We have developed a simple single-page To-Do list Web Application. This was developed using:
    1. Frontend - ReactJS
    2. Backend Routing - NodeJS and ExpressJS
    3. Database - AWS RDS - MySQL
    4. Hosted using - EC2 instances.
- The Web Application has the following functionality:
    1. Users can create tasks.
    2. Users can view the created tasks.
    3. Users can update tasks as completed.
    4. Users can delete tasks.

PES UNIVERSITY, BANGALORE
Department of Computer Science and Engineering

Methodology

The methodology used to host the to-do list web app was:

- Creation of front end using ReactJs. Using bootstrap CSS in Reactjs a working front end was developed.
- An EC2 instance was created with nginx installation to host the webpage. For the EC2 instance to render the webpage the code was pushed to the GitHub repository where a GitHub/workflows/blank.yaml file exists which has an auto-trigger on commit enabled.
- Any commit causes the workflow to run again, deleting and rebuilding the WebApp completely.
- The backend is built using NodeJs and is also pushed to the same GitHub.
- We run the front end and back end on the same instance.
- As the backend needs to be running simultaneously with the frontend we run the front end on port 3000 and redirect it to port 80 of the EC2 instance, while the backend is redirected to port 81 from 3001 of the EC2 instance.
- An SQL database is hosted on AWS RDS. A user profile is created and the host and credentials are added to the NodeJs to allow it to connect to the RDS.
- CRUD operations such as adding a todo, updating a todo and deleting a todo, all will now work with the database. The database ‘CCdatabase’ has a table called todolist which contains the parameters id, listname and completion. Listname is added upon the user adding a todo, upon marking complete completion should turn to 1 from the default 0 value and an auto-assigned id should also be provided to the todo.

The methodology to run the todo-list web app:

- Store the key file (cc-key-pair.pem) which is needed for establishing a connection with the EC2 instance.
- Establish connection on 2 separate terminals using -
  - ssh -i <path_to_key_file> ubuntu@ec2-35-92-36-213.us-west-2.compute.amazonaws.com
  - ssh -i "cc-key-pair.pem" ubuntu@ec2-35-92-36-213.us-west-2.compute.amazonaws.com
- Once the connection is successfully established, we navigate to the folder containing the files to run the frontend and backend respectively.
  - For frontend - cd /var/www/todo/frontend
  - For backend - cd /var/www/todo/backend
- Start the servers
  - Frontend - serve -s build
  - Backend - npm start
- Open browser and type -
  - http://34.218.250.140/

PES UNIVERSITY, BANGALORE
Department of Computer Science and Engineering

Testing

Some of the key observations that we made during testing are-

1. Updating multiple users at quick succession takes time due to the 3-tier architecture that is used to develop this application. There is a very visible lag that is observed in this situation. The website renders the tasks as completed however it takes time for the changes to appear in the database.

2. The timeout for this reverse proxy connection is set to 240 seconds and hence, in cases like updating or deleting multiple tasks in quick succession may result in overloading the system and the result are not updated before timeout.

3. While adding tasks to the database, a task_id is auto-incremented. The task_id numbers persist and retain their order until the table is truncated.

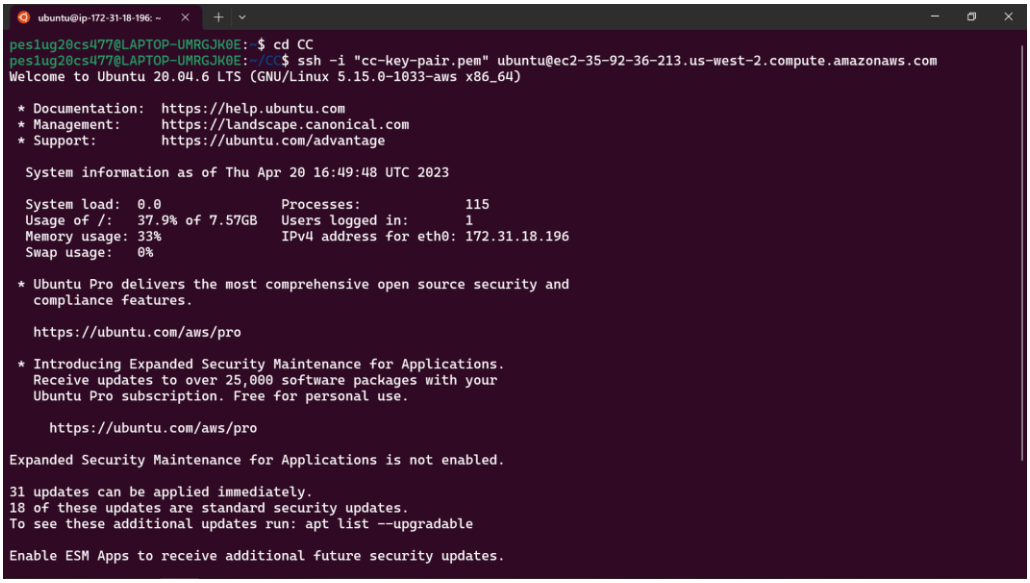
4. The connection to the database at the AWS RDS instance is established using MySQL Workbench. This fails to connect unless the IP address of the system is a part of the inbound rules of the RDS database instance.

PES UNIVERSITY, BANGALORE
Department of Computer Science and Engineering

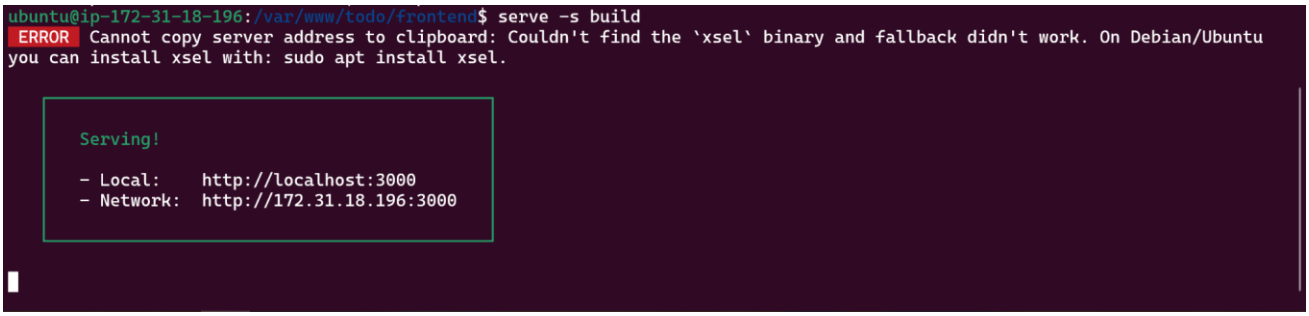
Results and Conclusions

The end result of this project can be seen here

Terminal



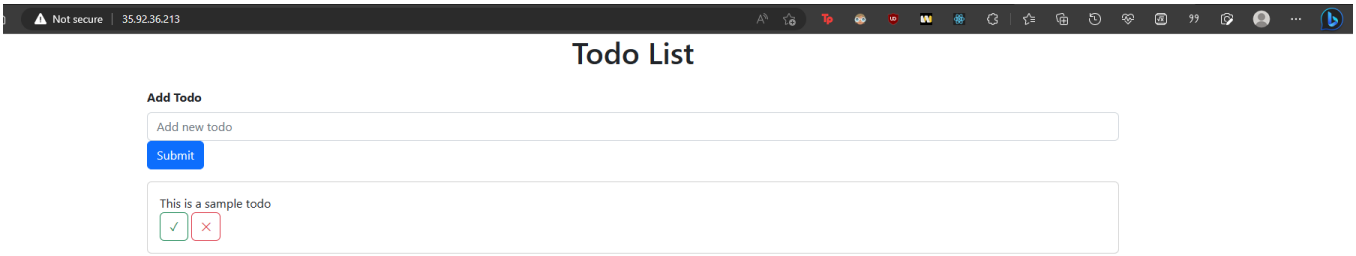
Frontend



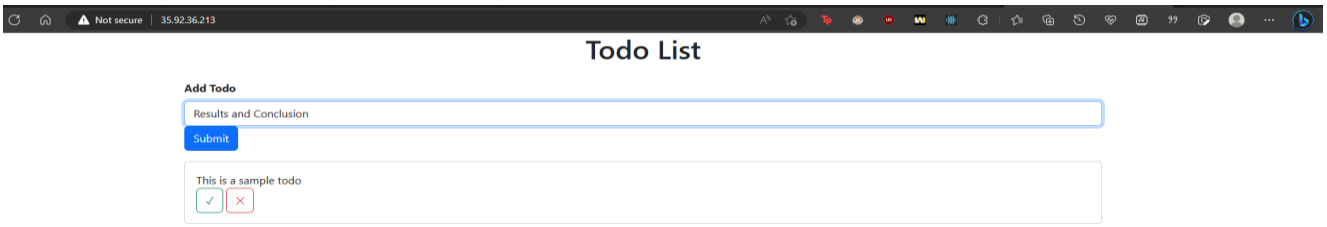
Backend

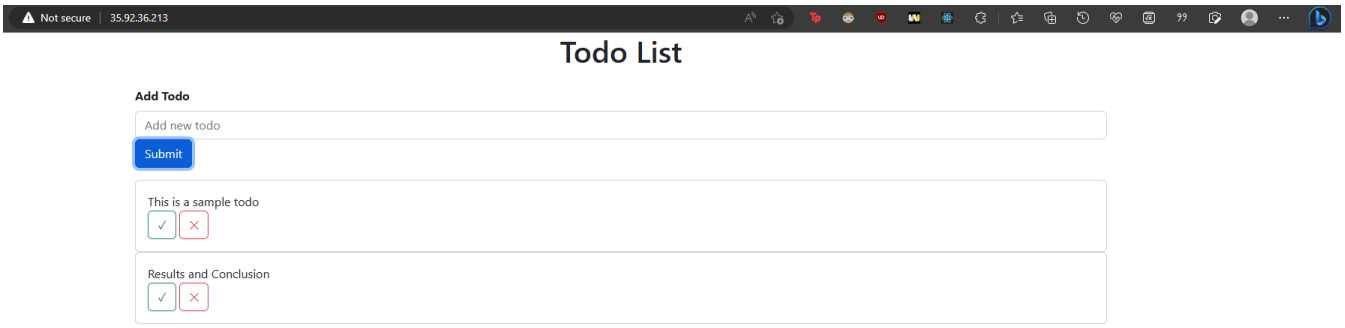
PES UNIVERSITY, BANGALORE
Department of Computer Science and Engineering

Before Creating new task

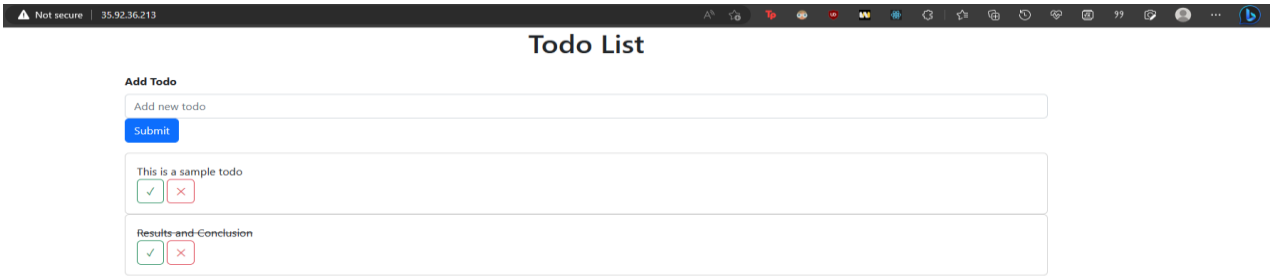


Adding Task



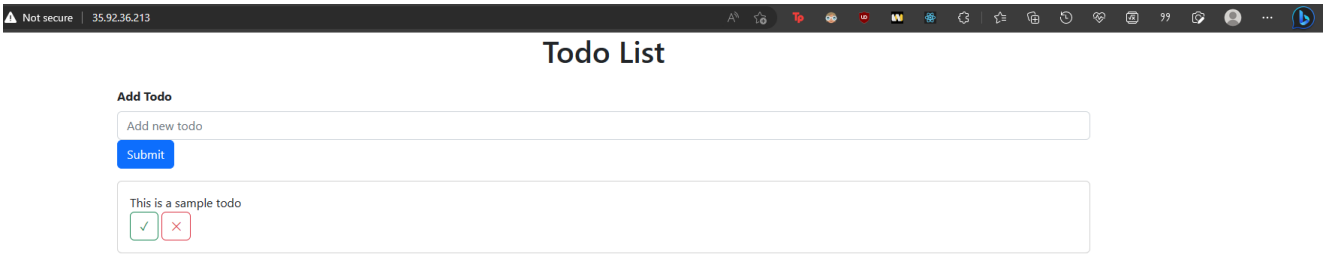


Marking Task as Completed



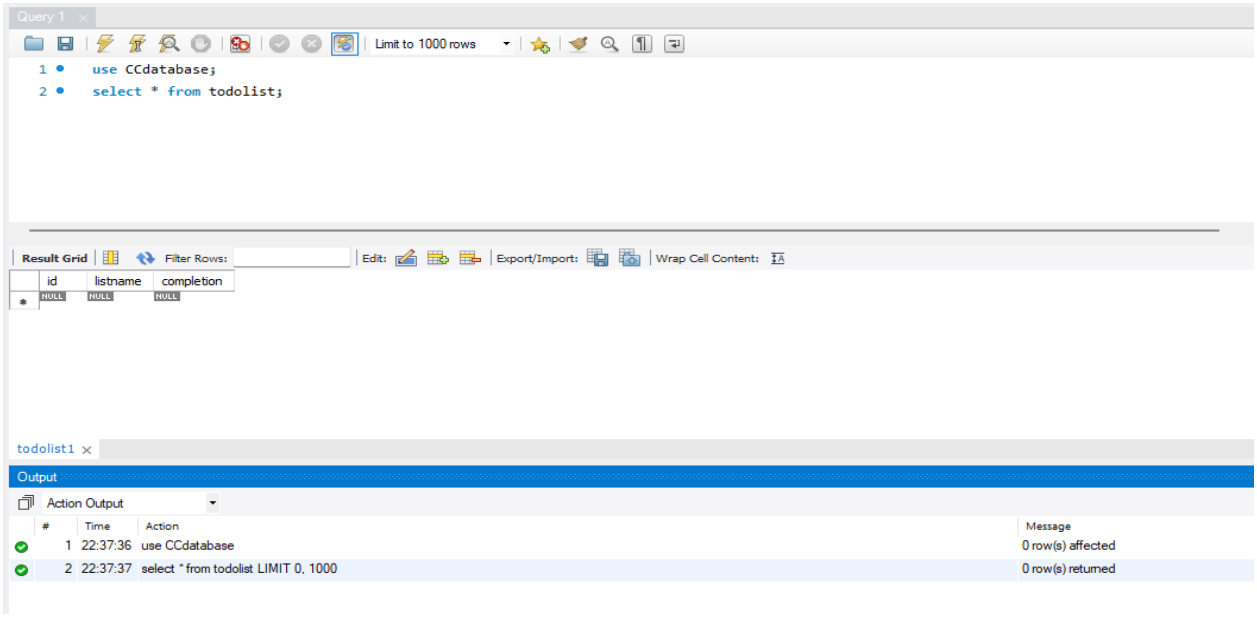
PES UNIVERSITY, BANGALORE
Department of Computer Science and Engineering

Removing Task

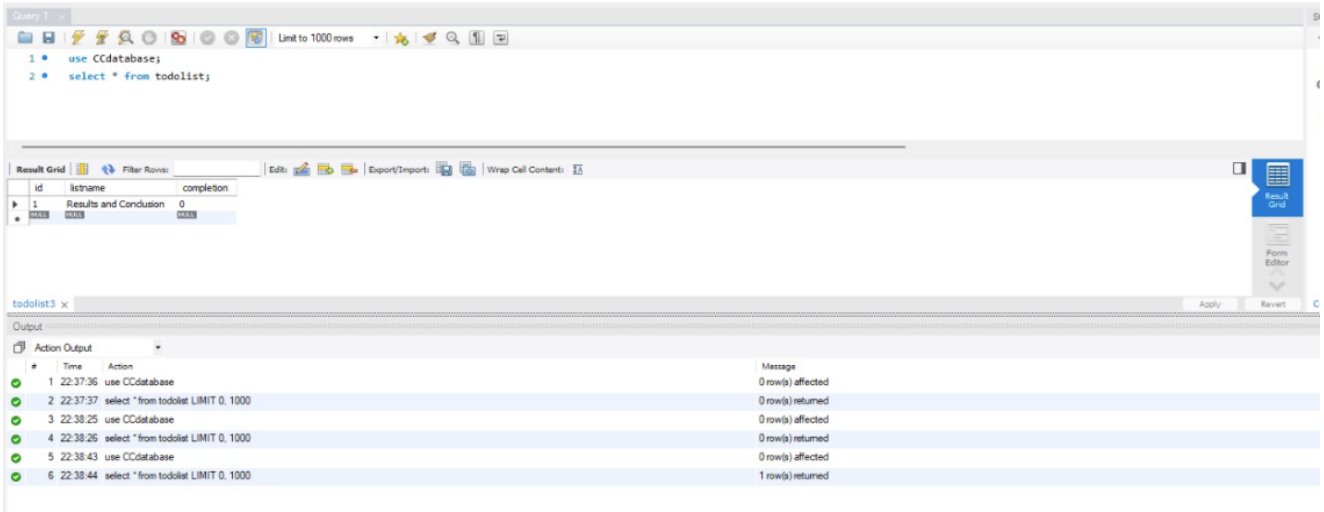


Database

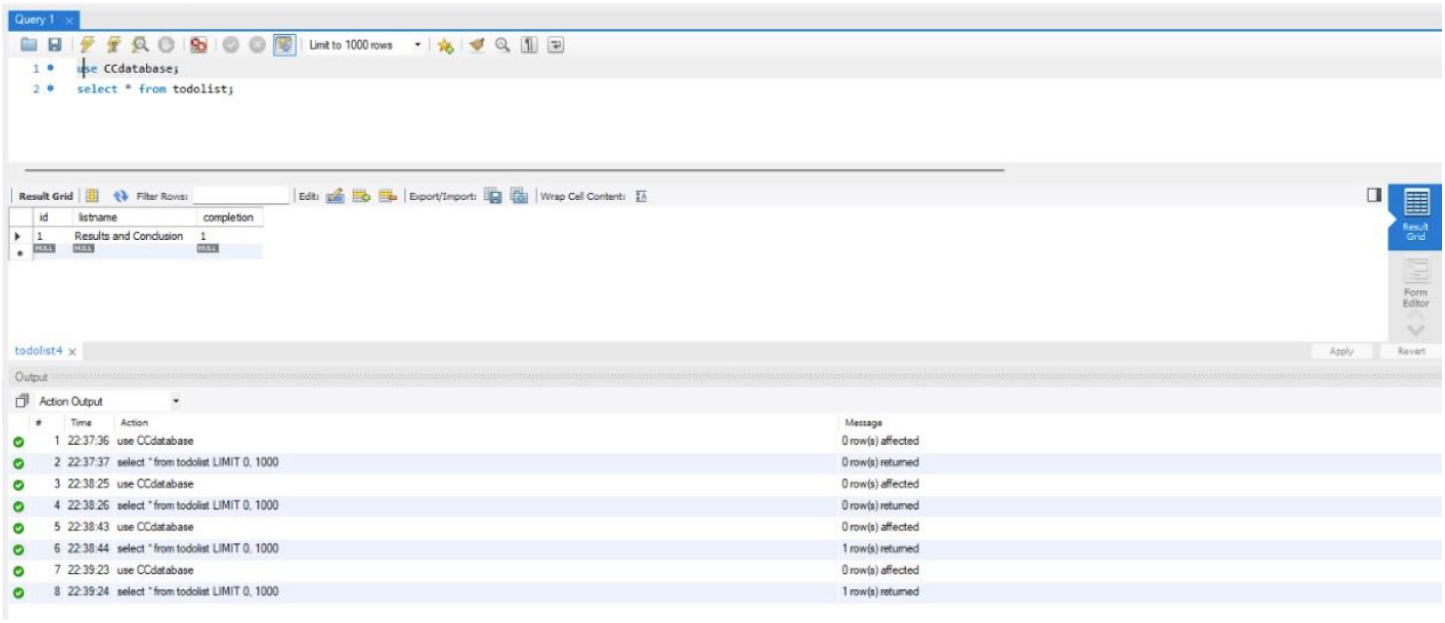
Before Adding Task



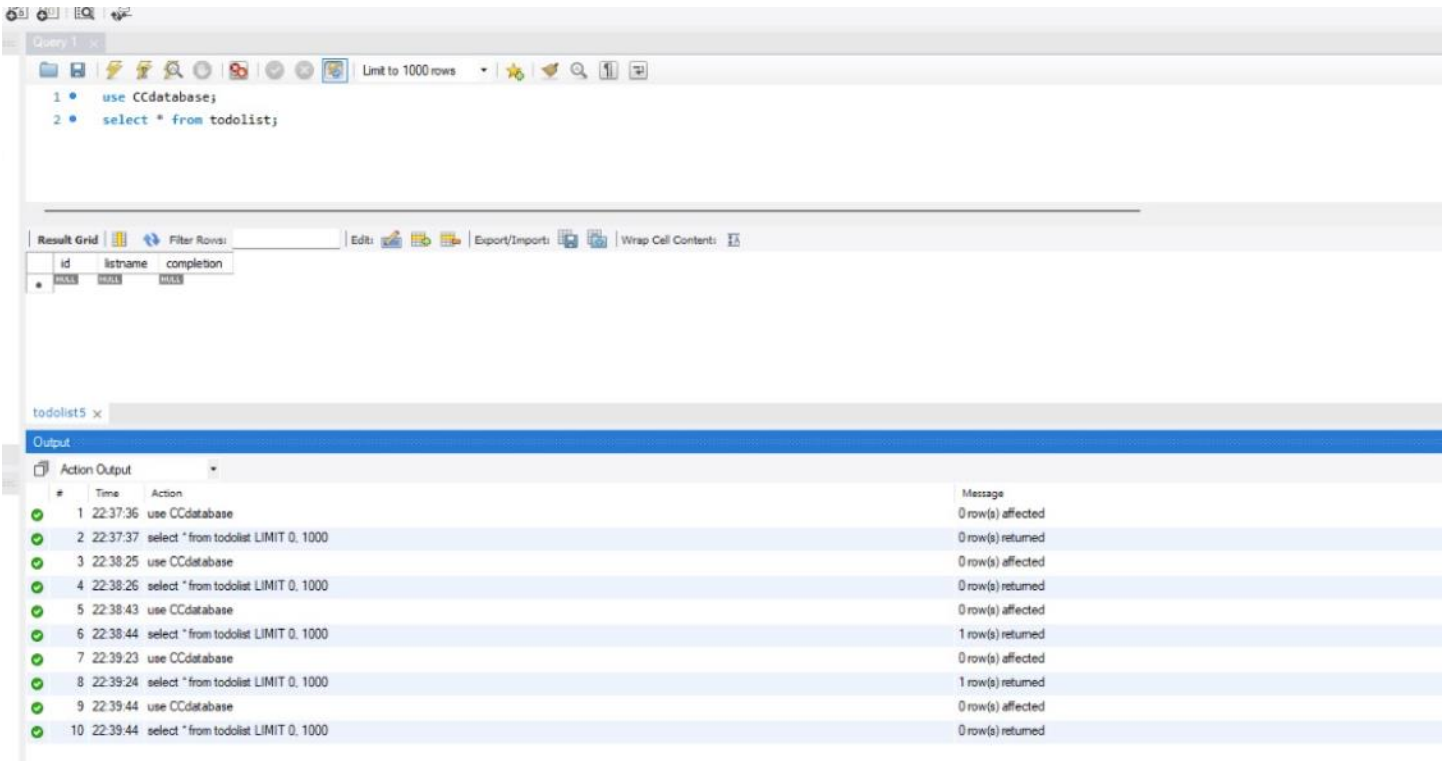
Task Created



Task Updated as completed.



Task Deleted



Through this project, we improved our understanding of developing and hosting a complete web application using AWS. We also learnt how to build a deployment pipeline using deployment tools like GitHub Actions.