📄 Project Report

Project Title: Java Application Deployment with Reverse Proxy on AWS

✅ 1. Introduction

This project demonstrates the deployment of a Java-based Student Registration Web Application on Amazon Web Services (AWS). The goal is to host the application securely with database integration and controlled public access using a reverse proxy. By combining multiple open-source tools and cloud services, the solution ensures scalability, security, and maintainability.

📌 2. Requirements

- Deploy a Java `.war` application on an EC2 instance using Apache Tomcat.
- Launch an Amazon RDS MySQL database, create schema and table for student registration data.
- Configure a second EC2 instance as a reverse proxy (using Apache HTTPD or Nginx).
- Ensure only the reverse proxy EC2 instance is publicly accessible.
- Enable secure communication between application server and database.

🛠️ 3. Technology Stack

Layer	Technology / Service
Cloud Infrastructure	AWS EC2, AWS RDS
Application Server	Apache Tomcat
Reverse Proxy	Nginx (or Apache HTTPD)
Database	MySQL-compatible Amazon RDS
OS	Amazon Linux 2
Java Connector	MySQL Connector/J
App	Java Servlet-based `student.war`

📊 4. System Architecture

Components:

- **EC2-Backend**: Hosts Tomcat and the Java application, listens on port 8080.
- **Amazon RDS**: Stores student registration data.
- **EC2-Proxy**: Reverse proxy publicly accessible, forwards traffic to backend.

Traffic Flow:

```
User → EC2-Proxy (Public IP, port 80)      → forwards to EC2-Backend (Private IP, port
8080)      → EC2-Backend → Amazon RDS (Private connection)
```

Security:

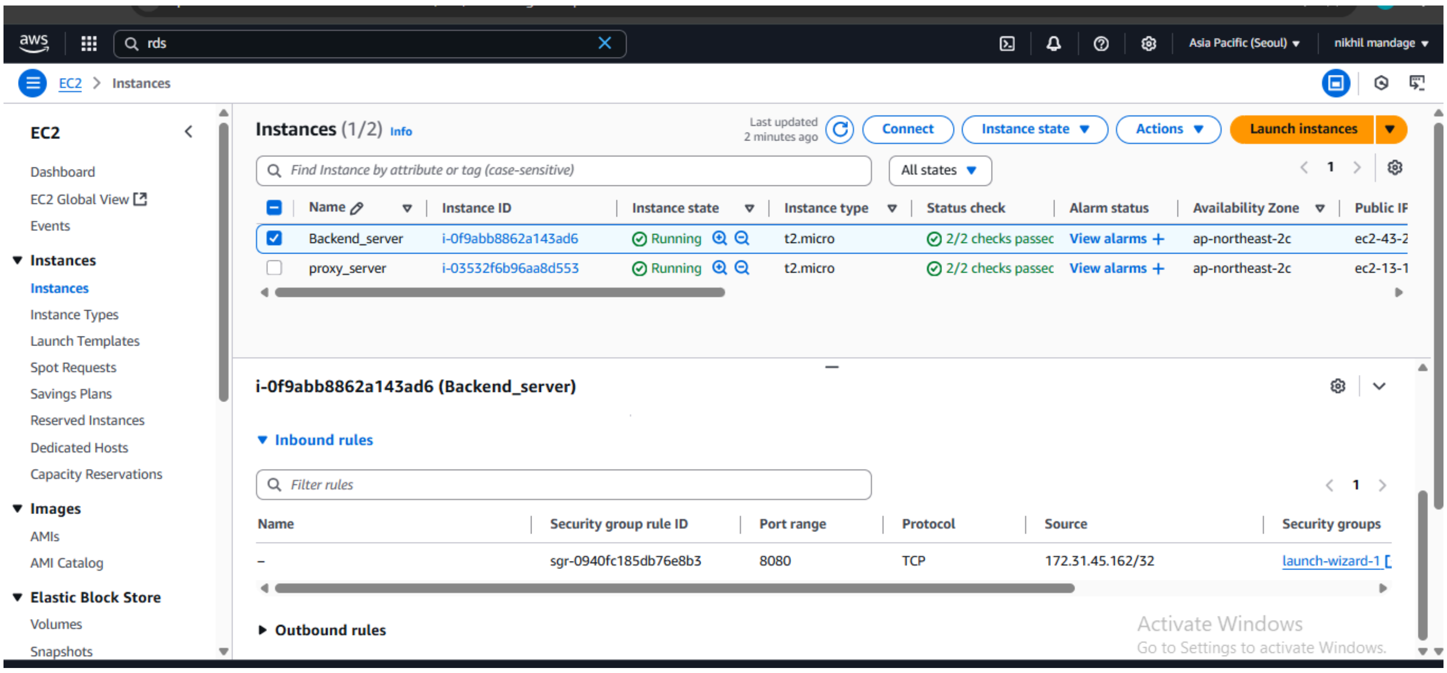
- EC2-Backend: only accepts incoming traffic from EC2-Proxy's private IP.
- EC2-Proxy: only accepts incoming traffic from the internet on port 80.
- RDS: only accepts connections from EC2-Backend.

(Draw this architecture as a diagram: 3 boxes labeled EC2-Proxy, EC2-Backend, and RDS, with arrows showing the traffic flow.)

⚙ 5. Implementation Steps

5.1 Infrastructure Setup

- Launched two Amazon Linux 2 EC2 instances:
  - `ec2-backend` (private, only accessible via SSH from trusted IPs)
  - `ec2-proxy` (publicly accessible)



Instances (1/2) Info

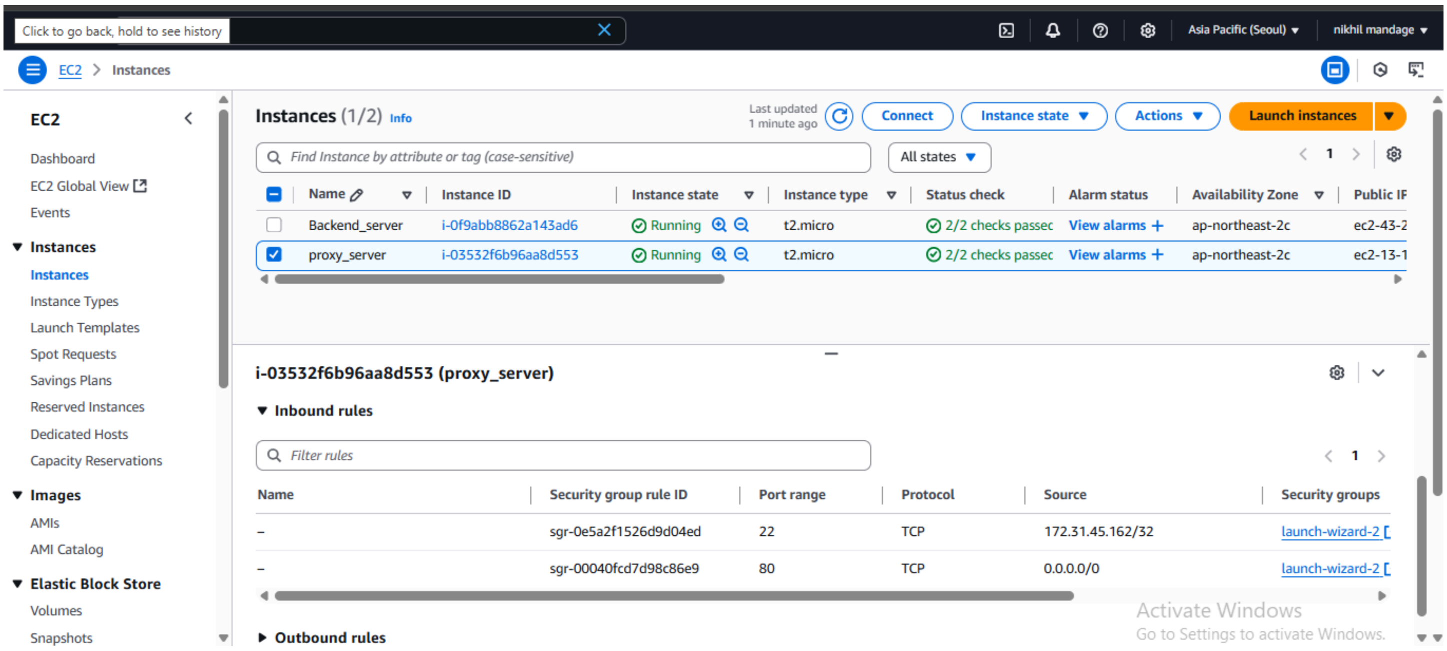
	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IP
☑	Backend_server	i-0f9abb8862a143ad6	Running	t2.micro	2/2 checks passec	View alarms +	ap-northeast-2c	ec2-43-2
☐	proxy_server	i-03532f6b96aa8d553	Running	t2.micro	2/2 checks passec	View alarms +	ap-northeast-2c	ec2-13-1

i-0f9abb8862a143ad6 (Backend_server)

Inbound rules

Name	Security group rule ID	Port range	Protocol	Source	Security groups
–	sgr-0940fc185db76e8b3	8080	TCP	172.31.45.162/32	launch-wizard-1

Outbound rules



Instances (1/2) Info

	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IP
☐	Backend_server	i-0f9abb8862a143ad6	Running	t2.micro	2/2 checks passec	View alarms +	ap-northeast-2c	ec2-43-2
☑	proxy_server	i-03532f6b96aa8d553	Running	t2.micro	2/2 checks passec	View alarms +	ap-northeast-2c	ec2-13-1

i-03532f6b96aa8d553 (proxy_server)

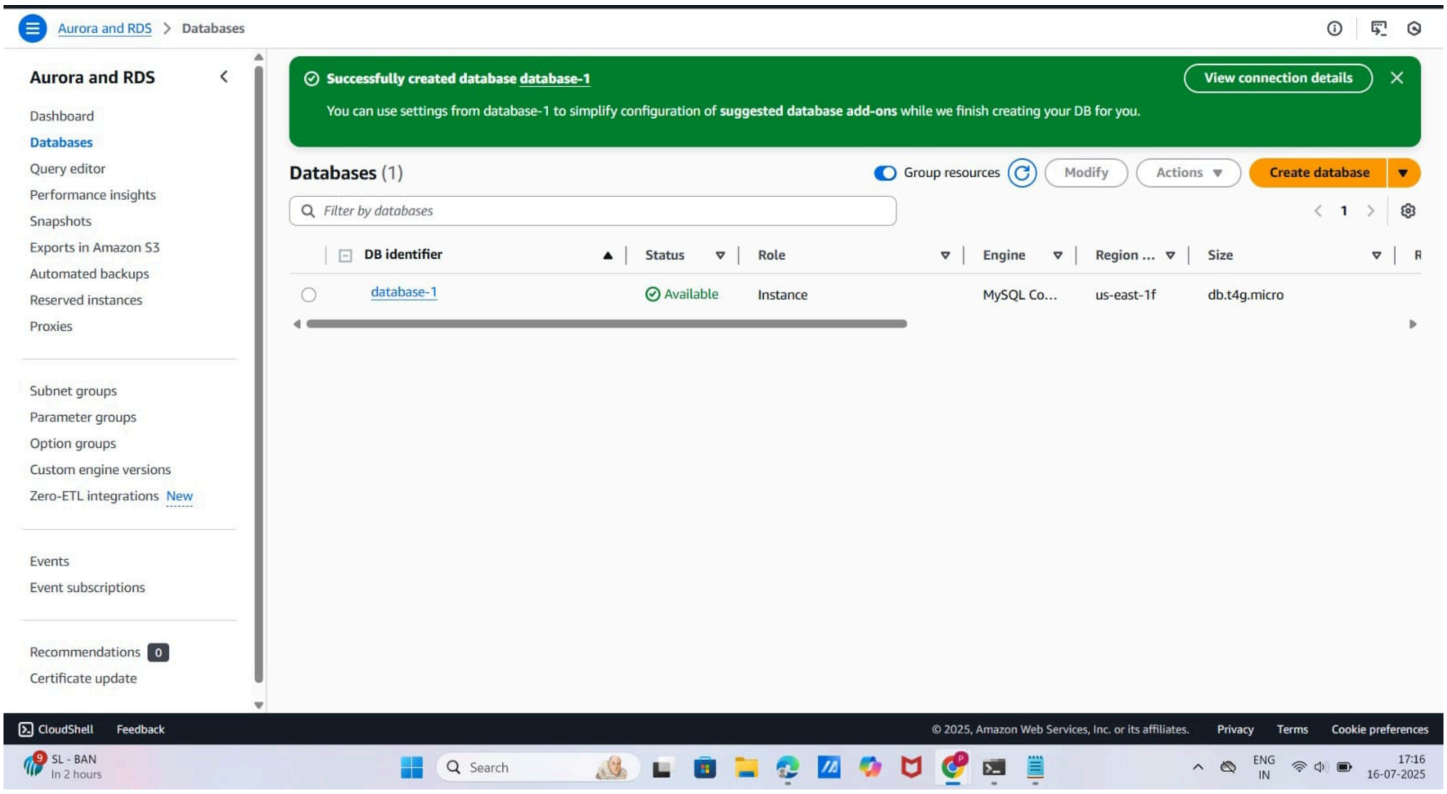
Inbound rules

Name	Security group rule ID	Port range	Protocol	Source	Security groups
–	sgr-0e5a2f1526d9d04ed	22	TCP	172.31.45.162/32	launch-wizard-2
–	sgr-00040fcd7d98c86e9	80	TCP	0.0.0.0/0	launch-wizard-2

Outbound rules

  -

- Launched an Amazon RDS MySQL-compatible instance.


Aurora and RDS > Databases

Successfully created database database-1

You can use settings from database-1 to simplify configuration of **suggested database add-ons** while we finish creating your DB for you.

Databases (1)

DB identifier	Status	Role	Engine	Region ...	Size
database-1	Available	Instance	MySQL Co...	us-east-1f	db.t4g.micro

© 2025, Amazon Web Services, Inc. or its affiliates.

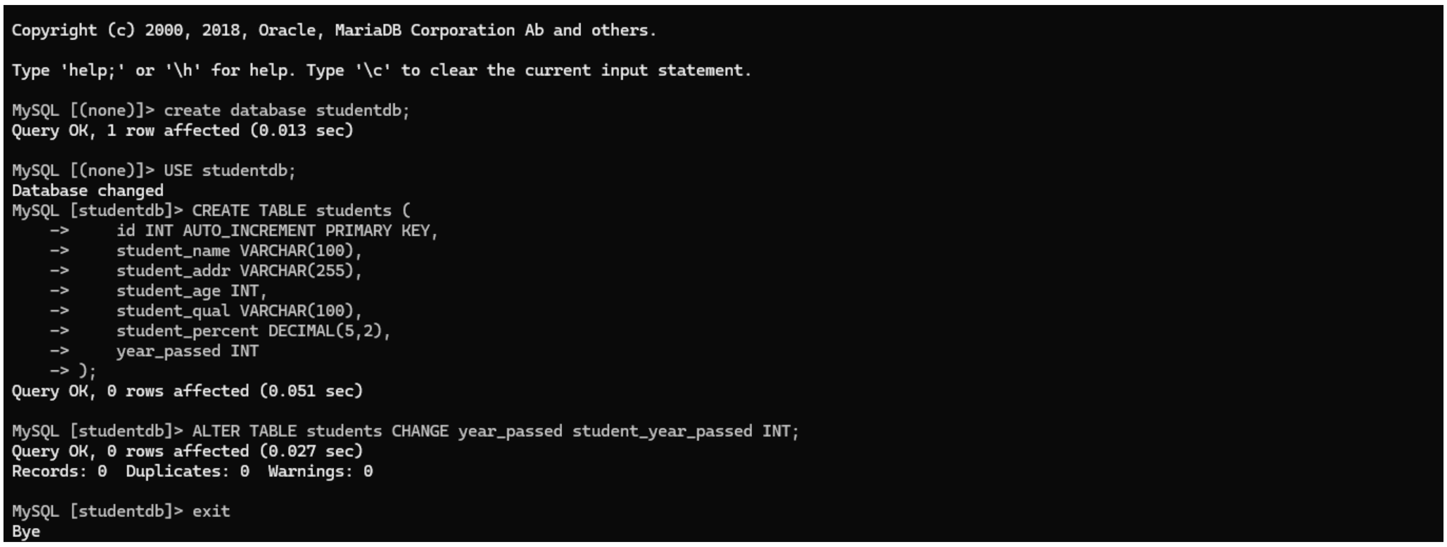
5.2 Application Deployment

- Installed Java and Apache Tomcat on `ec2-backend`.
- Downloaded `student.war` and deployed to Tomcat's webapps directory.
- Verified the application runs internally at `http://<backend-private-ip>:8080/student`.

5.3 Database Setup

- Created schema `studentdb`.
- Created table:

```
CREATE TABLE students (  id INT AUTO_INCREMENT PRIMARY KEY,  name VARCHAR(100),  email
VARCHAR(100),  course VARCHAR(50));
```



```
Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MySQL [(none)]> create database studentdb;
Query OK, 1 row affected (0.013 sec)

MySQL [(none)]> USE studentdb;
Database changed
MySQL [studentdb]> CREATE TABLE students (
    ->     id INT AUTO_INCREMENT PRIMARY KEY,
    ->     student_name VARCHAR(100),
    ->     student_addr VARCHAR(255),
    ->     student_age INT,
    ->     student_qual VARCHAR(100),
    ->     student_percent DECIMAL(5,2),
    ->     year_passed INT
    -> );
Query OK, 0 rows affected (0.051 sec)

MySQL [studentdb]> ALTER TABLE students CHANGE year_passed student_year_passed INT;
Query OK, 0 rows affected (0.027 sec)
Records: 0  Duplicates: 0  Warnings: 0

MySQL [studentdb]> exit
Bye
```

- Configured RDS security group to allow connections only from `ec2-backend`.

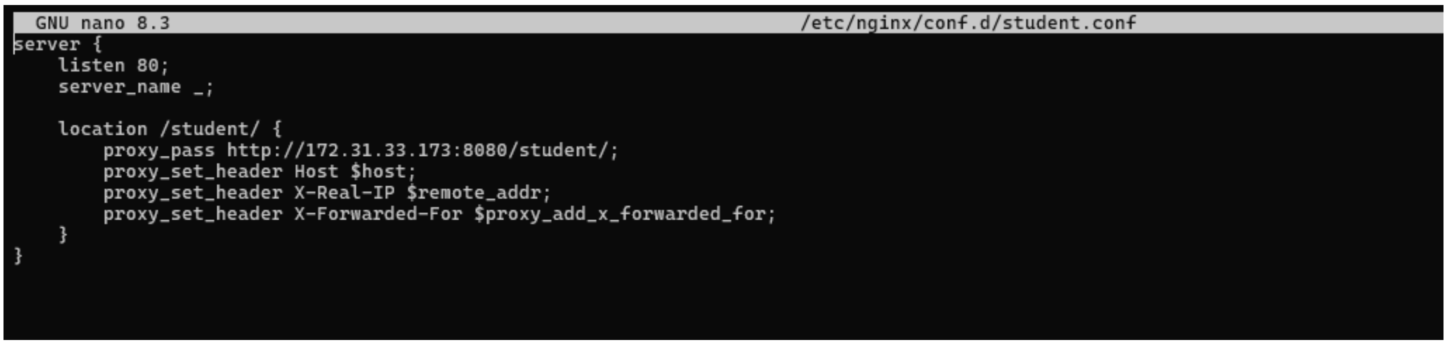
5.4 Database Connector

- Downloaded `mysql-connector.jar`.
- Placed it into Tomcat's `lib` directory.
- Restarted Tomcat to enable JDBC connectivity.

5.5 Reverse Proxy Configuration

- Installed Nginx on `ec2-proxy`.
- Configured Nginx:

```
server {    listen 80;    location /student {        proxy_pass http://<backend-private-ip>:8080/student;    }}
```



```
  GNU nano 8.3                                         /etc/nginx/conf.d/student.conf
server {
    listen 80;
    server_name _;

    location /student/ {
        proxy_pass http://172.31.33.173:8080/student/;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    }
}
```

- Updated backend security group to allow access only from proxy's private IP.

5.6 Access Setup

- Ensured the application is only reachable via:

```
http://<Proxy-Instance-Public-IP>/student
```

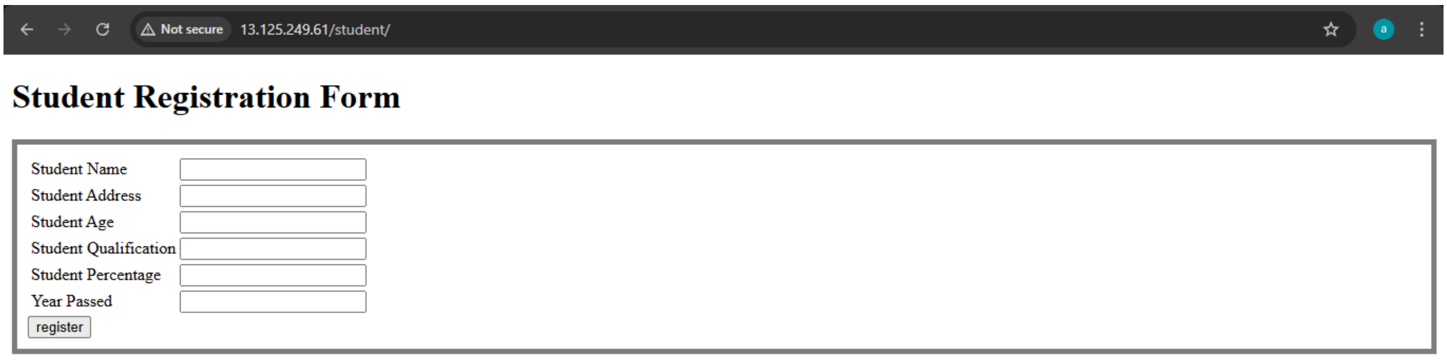
- Direct access to backend EC2 instance on port 8080 is blocked.

📷 6. Result

- Successfully accessed the deployed application via reverse proxy.
- Inserted student registration data.
- Verified data is stored in RDS using MySQL client.

Screenshots included:

1. Application page: `http://<Proxy-Instance-Public-IP>/student`



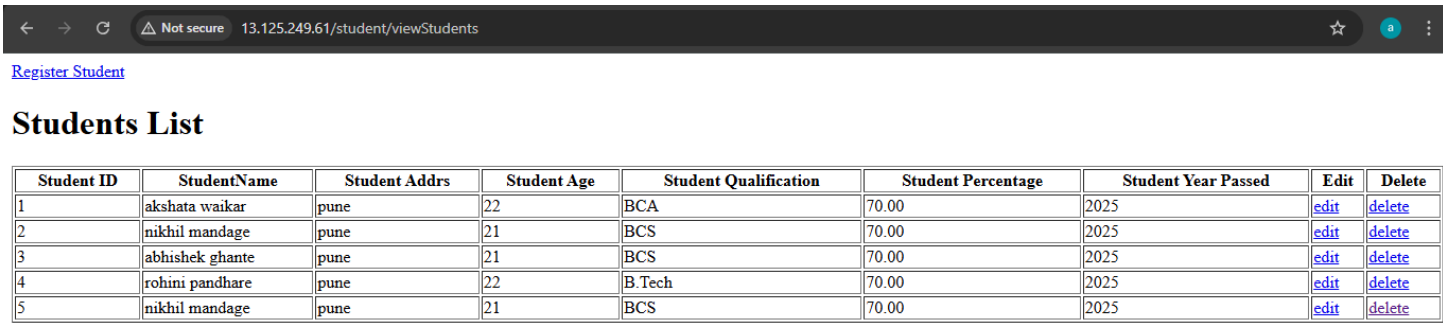
Not secure 13.125.249.61/student/

Student Registration Form

Student Name	
Student Address	
Student Age	
Student Qualification	
Student Percentage	
Year Passed	

register

2 . RDS database table showing stored data.



Not secure 13.125.249.61/student/viewStudents

Register Student

Students List

Student ID	StudentName	Student Addrs	Student Age	Student Qualification	Student Percentage	Student Year Passed	Edit	Delete
1	akshata waikar	pune	22	BCA	70.00	2025	edit	delete
2	nikhil mandage	pune	21	BCS	70.00	2025	edit	delete
3	abhishek ghante	pune	21	BCS	70.00	2025	edit	delete
4	rohini pandhare	pune	22	B.Tech	70.00	2025	edit	delete
5	nikhil mandage	pune	21	BCS	70.00	2025	edit	delete

📝 7. Conclusion

This project demonstrates secure deployment of a Java-based web application on AWS.By using:

- EC2 for compute,
- Amazon RDS for managed database,
- Tomcat as servlet container,
- and Nginx as reverse proxy,

we achieved a scalable, maintainable, and secure architecture. The use of a reverse proxy ensures only a single public entry point, improving security and flexibility.

Challenges faced:

- Configuring security groups correctly to restrict direct backend access.
- Handling connectivity between Tomcat and RDS (resolved by correct VPC security group setup and JDBC connector).

Overall, the deployment meets the objectives, successfully hosting the application with secure access and proper database integration.

name - nikhil mandage

Cloud & DevOps Intern

Cravita Technologies