CLOUD INFOTECH SOLUTIONS
Moosram Bagh, Malakpet, Hyderabad – 36, TS

📧 hr@cloudinfotechsol.com | 📞 8688253560

Student Portal – Login & Registration System

Using Python + AWS Lambda + API Gateway + RDS MySQL + Secrets Manager

---

1️⃣ WHERE this project fits (real-world use)

- Colleges / coaching institutes
- Internship portals
- Any website that needs **secure login & registration**
- Used in **MNC applications, SaaS products, EdTech platforms**

👉 This is a **real production pattern**, not a dummy project.

2️⃣ WHAT this project does (features)

Pages / APIs

- ✅ Register student
- ✅ Login student
- ✅ Home page (after login)
- ✅ Dashboard (student details)

AWS Services Used

CLOUD INFOTECH SOLUTIONS
Moosram Bagh, Malakpet, Hyderabad – 36, TS

📧 hr@cloudinfotechsol.com | 📞 8688253560

Service	Purpose
Lambda (Python)	Backend logic
API Gateway	HTTP APIs
RDS MySQL	Store users
Secrets Manager	Secure DB credentials

3️⃣ WHY these services are used

- **Lambda** → No server management
- **API Gateway** → Secure REST APIs
- **RDS MySQL** → Structured student data
- **Secrets Manager** → No passwords in code (BEST PRACTICE)

Project Summary (for records)

Title:
Student Registration & Login System using Serverless AWS

Description:
Built a serverless student portal using AWS Lambda, API Gateway, RDS MySQL, and Secrets Manager to handle secure registration, login, and dashboard access.

1️⃣3️⃣ Challenges & Solutions

Challenge	Solution

CLOUD INFOTECH SOLUTIONS
Moosram Bagh, Malakpet, Hyderabad – 36, TS

📧 hr@cloudinfotechsol.com | 📞 8688253560

Secure DB credentials	Used AWS Secrets Manager
No server management	Used Lambda
Scalability	Serverless auto-scaling
Data consistency	RDS MySQL

1️⃣4️⃣ Outcomes (what students learn)

- Real AWS architecture
- Backend APIs
- Database integration
- Security best practices
- Interview-ready project

Student Login & Registration – CLICK BY CLICK AWS GUIDE

🔹 STEP 0: Prerequisites (DO THIS FIRST)

✔ AWS Account
✔ Region: **Mumbai (ap-south-1)**
✔ Basic Python knowledge

👉 Login to **AWS Console**

CLOUD INFOTECH SOLUTIONS
Moosram Bagh, Malakpet, Hyderabad – 36, TS

📧 hr@cloudinfotechsol.com | 📞 8688253560

🔷 STEP 1: Create RDS MySQL (DATABASE)

📍 WHERE TO GO

AWS Console → **Services** → **RDS**

🖱️ CLICKS

1. Click **Create database**
2. Choose **Standard create**
3. Engine: **MySQL**
4. Version: Default

⚙️ SETTINGS

- Templates: **Free tier**
- DB instance identifier: `student-db`
- Master username: `admin`
- Master password: `Admin@123`

Connectivity

- Public access: **Yes**

CLOUD INFOTECH SOLUTIONS
Moosram Bagh, Malakpet, Hyderabad – 36, TS

📧 hr@cloudinfotechsol.com | 📞 8688253560

- VPC security group → **Create new**
- New SG name: `student-db-sg`

Final

➡️ Click **Create database**

⏳ Wait 5–7 minutes
👉 Copy **Endpoint** (VERY IMPORTANT)

🔷 STEP 2: Create Database & Table

📍 WHERE

AWS Console → **RDS** → Databases → `student-db`

🖱️ CLICKS

1. Click **Query Editor**
2. Login using DB username/password

🗒️ RUN SQL

```
CREATE DATABASE student_portal;
USE student_portal;

CREATE TABLE users (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100),
    email VARCHAR(100) UNIQUE,
```


```
    password VARCHAR(255)
);
```

🔷 STEP 3: Store DB Credentials (Secrets Manager)

📍 WHERE

AWS Console → **Secrets Manager**

🖱️ CLICKS

1. Click **Store a new secret**
2. Secret type: **Other type**
3. Key/Value:

```
{
  "host": "YOUR_RDS_ENDPOINT",
  "username": "admin",
  "password": "Admin@123",
  "dbname": "student_portal"
}
```

4. Click **Next**
5. Secret name: `rds/student/db`

CLOUD INFOTECH SOLUTIONS
Moosram Bagh, Malakpet, Hyderabad – 36, TS

📧 hr@cloudinfotechsol.com | 📞 8688253560

6. Click **Store**

🔷 STEP 4: Create IAM Role for Lambda

📍 WHERE

AWS Console → **IAM** → Roles

🖱️ CLICKS

1. Create role
2. Trusted entity: **AWS service**
3. Use case: **Lambda**

Attach policies

✔ `AWSLambdaBasicExecutionRole`
✔ `SecretsManagerReadWrite`

➡️ Click **Create role**
Role name: `lambda-rds-role`

🔷 STEP 5: Create Lambda Function (Register)

📍 WHERE

CLOUD INFOTECH SOLUTIONS
Moosram Bagh, Malakpet, Hyderabad – 36, TS

📧 hr@cloudinfotechsol.com | 📞 8688253560

AWS Console → **Lambda**

🖱️ CLICKS

1. Create function
2. Author from scratch
3. Function name: `register-user`
4. Runtime: **Python 3.10**
5. Execution role: **Use existing role**
6. Select: `lambda-rds-role`
7. Click **Create**

🗒️ CODE (Paste fully)

```
import json, pymysql, boto3

def get_db():
    secret = boto3.client('secretsmanager', region_name='ap-south-1')
\
        .get_secret_value(SecretId='rds/student/db')
    creds = json.loads(secret['SecretString'])

    return pymysql.connect(
        host=creds['host'],
        user=creds['username'],
        password=creds['password'],
        database=creds['dbname']
```


```
    )

def lambda_handler(event, context):
    data = json.loads(event['body'])
    conn = get_db()
    cur = conn.cursor()

    cur.execute(
        "INSERT INTO users (name,email,password) VALUES (%s,%s,%s)",
        (data['name'], data['email'], data['password'])
    )
    conn.commit()

    return {
        "statusCode": 200,
        "body": json.dumps("User Registered")
    }
```

➡️ Click **Deploy**

🔷 STEP 6: Create Login Lambda

📍 WHERE

Lambda → **Create function**

- Name: `login-user`
- Runtime: Python 3.10

- Role: `lambda-rds-role`

🧾 CODE

```
import json, pymysql, boto3

def lambda_handler(event, context):
    data = json.loads(event['body'])
    conn = pymysql.connect(
        host="YOUR_RDS_ENDPOINT",
        user="admin",
        password="Admin@123",
        database="student_portal"
    )

    cur = conn.cursor()
    cur.execute(
        "SELECT * FROM users WHERE email=%s AND password=%s",
        (data['email'], data['password'])
    )

    user = cur.fetchone()
    if user:
        return {"statusCode": 200, "body": json.dumps("Login
Success")}
    else:
        return {"statusCode": 401, "body": json.dumps("Invalid User")}
```

➡️ Click **Deploy**

CLOUD INFOTECH SOLUTIONS
Moosram Bagh, Malakpet, Hyderabad – 36, TS

📧 hr@cloudinfotechsol.com | 📞 8688253560

🔷 STEP 7: API Gateway (CONNECT EVERYTHING)

📍 WHERE

AWS Console → **API Gateway**

🖱️ CLICKS

1. Create API
2. REST API
3. New API
4. Name: `student-api`
5. Create

Create Resources

Resource	Method	Lambda
/register	POST	register-user
/login	POST	login-user

➡️ Enable **CORS**
➡️ **Deploy API**
➡️ Stage: `prod`

👉 COPY API URL

CLOUD INFOTECH SOLUTIONS
Moosram Bagh, Malakpet, Hyderabad – 36, TS

📧 hr@cloudinfotechsol.com | 📞 8688253560

🔹 STEP 8: Test (FINAL)

POSTMAN / Browser

Register

```
POST /register
{
 "name":"Ali",
 "email":"ali@gmail.com",
 "password":"1234"
}
```

Login

```
POST /login
{
 "email":"ali@gmail.com",
 "password":"1234"
}
```

🎯 FINAL RESULT

✅ Student registered
✅ Login works
✅ AWS Serverless project
✅ Interview ready

CLOUD INFOTECH SOLUTIONS
Moosram Bagh, Malakpet, Hyderabad – 36, TS

📧 hr@cloudinfotechsol.com | 📞 8688253560

🧠 How to Explain in Interview (ONE LINE)

> “I created a serverless student authentication system using AWS Lambda, API Gateway, RDS MySQL, and Secrets Manager with secure credential handling.”

====
Update code if user input required 👍

✅ **Registration**
✅ **Login**
✅ **Search student by email / name**

All code is **student-friendly**, **easy**, and **interview-ready**.

🔹 INPUT FORMAT (VERY IMPORTANT)

👉 **All input comes from API Gateway → JSON body**

Example (user input):

```
{
  "email": "ali@gmail.com",
  "password": "1234"
}
```

🔹 COMMON DB CONNECTION (USE IN ALL LAMBDAS)

CLOUD INFOTECH SOLUTIONS
Moosram Bagh, Malakpet, Hyderabad – 36, TS

📧 hr@cloudinfotechsol.com | 📞 8688253560

📌 **Create one file or copy in all Lambda functions**

```
import json
import pymysql
import boto3

def get_db_connection():
    client = boto3.client("secretsmanager", region_name="ap-south-1")
    secret = client.get_secret_value(SecretId="rds/student/db")
    creds = json.loads(secret["SecretString"])

    return pymysql.connect(
        host=creds["host"],
        user=creds["username"],
        password=creds["password"],
        database=creds["dbname"],
        cursorclass=pymysql.cursors.DictCursor
    )
```

🔷 1️⃣ REGISTRATION CODE (TAKES USER INPUT)

📌 **Lambda name:** `register-user`

```
def lambda_handler(event, context):
    body = json.loads(event["body"])

    name = body["name"]
    email = body["email"]
    password = body["password"]
```


```
    conn = get_db_connection()
    cursor = conn.cursor()

    try:
        cursor.execute(
            "INSERT INTO users (name, email, password) VALUES (%s, %s,
%s)",
            (name, email, password)
        )
        conn.commit()

        return {
            "statusCode": 200,
            "body": json.dumps({"message": "Registration successful"})
        }

    except Exception as e:
        return {
            "statusCode": 400,
            "body": json.dumps({"message": "User already exists"})
        }
```

✅ Input Example

```
{
  "name": "Ali Khan",
  "email": "ali@gmail.com",
  "password": "1234"
}
```

CLOUD INFOTECH SOLUTIONS
Moosram Bagh, Malakpet, Hyderabad – 36, TS

📧 hr@cloudinfotechsol.com | 📞 8688253560

🔷 2️⃣ LOGIN CODE (TAKES USER INPUT)

📌 **Lambda name:** `login-user`

```
def lambda_handler(event, context):
    body = json.loads(event["body"])

    email = body["email"]
    password = body["password"]

    conn = get_db_connection()
    cursor = conn.cursor()

    cursor.execute(
        "SELECT id, name, email FROM users WHERE email=%s AND
password=%s",
        (email, password)
    )

    user = cursor.fetchone()

    if user:
        return {
            "statusCode": 200,
            "body": json.dumps({
                "message": "Login successful",
                "user": user
            })
        }
    else:
        return {
            "statusCode": 401,
            "body": json.dumps({"message": "Invalid credentials"})
```


```
        }
```

✅ Input Example

```
{
  "email": "ali@gmail.com",
  "password": "1234"
}
```

🔷 3️⃣ SEARCH STUDENT (BY NAME OR EMAIL)

📌 **Lambda name:** `search-user`
📌 **Method:** POST

```
def lambda_handler(event, context):
    body = json.loads(event["body"])
    keyword = body["search"]

    conn = get_db_connection()
    cursor = conn.cursor()

    cursor.execute(
        "SELECT id, name, email FROM users WHERE name LIKE %s OR email
LIKE %s",
        (f"%{keyword}%", f"%{keyword}%")
    )

    results = cursor.fetchall()
```

CLOUD INFOTECH SOLUTIONS
Moosram Bagh, Malakpet, Hyderabad – 36, TS

📧 hr@cloudinfotechsol.com | 📞 8688253560


```
    return {
        "statusCode": 200,
        "body": json.dumps(results)
    }
```

✅ Input Example

```
{
  "search": "ali"
}
```

🔹 API GATEWAY ROUTES (FINAL)

Method	Path	Lambda
POST	`/register`	register-user
POST	`/login`	login-user
POST	`/search`	search-user

🔹 DATABASE TABLE (FINAL)

```
CREATE TABLE users (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100),
    email VARCHAR(100) UNIQUE,
    password VARCHAR(255)
);
```

CLOUD INFOTECH SOLUTIONS
Moosram Bagh, Malakpet, Hyderabad – 36, TS

📧 hr@cloudinfotechsol.com | 📞 8688253560

🔹 HOW STUDENT EXPLAINS IN INTERVIEW (READY SENTENCE)

Short:

> "This project allows users to register, login, and search students using serverless AWS services."

Technical:

> "User input is sent via API Gateway in JSON format. AWS Lambda processes the request, securely connects to RDS MySQL using Secrets Manager, and performs registration, authentication, and search operations."

🔹 PROJECT OUTCOME

✅ Understand API input/output
✅ CRUD operations
✅ AWS security best practice
✅ Real-world backend logic