



*Mini project report on*

## **University Placement Management System**

*Submitted in partial fulfilment of the requirements for the award of degree of*

**Bachelor of Technology**

**in**

**Computer Science & Engineering**

**UE22CS351A – DBMS Project**

*Submitted by:*

**Bhavesh Budharaju**                   **PES2UG22CS129**

**Dhanush Suresh Jettipalle**                   **PES2UG22CS175**

Under the guidance of  
**Prof. Mannar Mannan**

Associate Professor

PES University

**AUG - DEC 2024**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING  
PES UNIVERSITY**

(Established under Karnataka Act No. 16 of 2013)

Electronic City, Hosur Road, Bengaluru – 560 100, Karnataka, India



## PES UNIVERSITY

(Established under Karnataka Act No. 16 of 2013)  
Electronic City, Hosur Road, Bengaluru – 560 100, Karnataka, India

## CERTIFICATE

*This is to certify that the mini project entitled*

### **University Placement Management System**

*is a bonafide work carried out by*

**Bhavesh Budharaju**

**PES2UG22CS129**

**Dhanush Suresh Jettipalle**

**PES2UG22CS175**

In partial fulfilment for the completion of fifth semester DBMS Project (UE22CS351A) in the Program of Study - Bachelor of Technology in Computer Science and Engineering under rules and regulations of PES University, Bengaluru during the period AUG. 2024 – DEC. 2024. It is certified that all corrections / suggestions indicated for internal assessment have been incorporated in the report. The project has been approved as it satisfies the 5<sup>th</sup> semester academic requirements in respect of project work.

Signature

Prof. Mannar Mannan

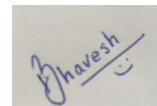
Associate Professor

## **DECLARATION**

We hereby declare that the DBMS Project entitled **University Placement Management System** has been carried out by us under the guidance of **Prof. Mannar Mannan, Associate Professor** and submitted in partial fulfilment of the course requirements for the award of degree of **Bachelor of Technology** in **Computer Science and Engineering** of **PES University, Bengaluru** during the academic semester AUG – DEC 2024.

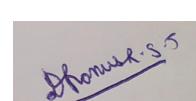
**Bhavesh Budharaju**

**PES2UG22CS129**



**Dhanush Suresh Jettipalle**

**PES2UG22CS175**



## **ABSTRACT**

University Placement Management System is a web-based placement management system, developed using MySQL and Python-Streamlit, that simulates the functionalities of a recruitment platform. Designed as an academic project for a DBMS course, it provides a structured approach to handling student, company, and admin interactions within a placement system. The project implements multi-user authentication, role-specific dashboards, and a real-time email notification feature. By utilizing a modular code structure, this project offers students a hands-on understanding of database operations, web app deployment, and secure data management in a placement context.

## **TABLE OF CONTENTS**

<b>Chapter No.</b>	<b>Title</b>	<b>Page No.</b>
1.	<b>INTRODUCTION</b>	1
2.	<b>PROBLEM DEFINITION WITH USER REQUIREMENT SPECIFICATIONS</b>	2
3.	<b>LIST OF SOFTWARES/TOOLS/PROGRAMMING LANGUAGES USED</b>	4
4.	<b>ER MODEL</b>	5
5.	<b>ER TO RELATIONAL MAPPING</b>	6
6.	<b>DDL STATEMENTS</b>	7
7.	<b>DML STATEMENTS (CRUD OPERATION SCREENSHOTS)</b>	9
8.	<b>QUERIES (JOIN QUERY, AGGREGATE FUNCTION QUERIES AND NESTED QUERY)</b>	26
9.	<b>STORED PROCEDURE, FUNCTIONS AND TRIGGERS</b>	28
10.	<b>FRONT END DEVELOPMENT (FUNCTIONALITIES/FEATURES OF THE APPLICATION)</b>	30
	<b>REFERENCES/BIBLIOGRAPHY</b>	32
	<b>APPENDIX A DEFINITIONS, ACRONYMS AND ABBREVIATIONS</b>	33

## **1. INTRODUCTION**

University Placement Management System addresses the complexity and demands of managing recruitment activities in educational institutions by replicating the core functionalities of a placement management platform. Current manual processes in campus placements can lead to inefficiencies and errors, especially when handling large volumes of student applications and job listings. This Project offers a scalable and organized system where different user roles—students, companies, and admins—can perform role-specific actions securely and effectively. Students can edit their profiles, track their applications, and manage skill information, while companies can post jobs, shortlist candidates, and manage applications. Administrators play a supervisory role, overseeing system metrics and user management, and they receive real-time insights into the system's functioning. This Project is built using MySQL for data storage and retrieval, ensuring robust data handling, while Python-Streamlit provides a responsive, user-friendly interface. The project uses core database management concepts like triggers, stored procedures, and secure password hashing, integrating these into a real-world, application-based project.

## **2a . PROBLEM DEFINITION**

Managing placement activities in educational institutions can be a challenging task, especially when the process relies heavily on manual coordination. Universities often face difficulties in managing student applications, job postings, and tracking candidate progress due to the complexity and sheer volume of data. These manual processes not only lead to inefficiencies, such as missed opportunities or errors in candidate shortlisting, but also lack transparency, which can create confusion among students, companies, and administrators. As a result, many placement cells struggle to streamline operations, making the recruitment process slower and prone to mistakes.

This Project addresses this issue by automating and simplifying the entire placement management process. This web-based system provides a structured approach where students, companies, and administrators can interact efficiently. By offering role-based functionalities and secure data handling, the system reduces the chances of errors and enhances the overall management of recruitment activities. Moreover, the use of modern technologies like MySQL for database management and Python-Streamlit for the interface ensures that the system is scalable, easy to use, and effective in streamlining the entire placement process.

## **2b . USER REQUIREMENT SPECIFICATIONS**

### **FUNCTIONAL REQUIREMENTS :**

- 1.** The University Placement Management platform is designed to cater to three distinct user roles, each with specific requirements. Students need an easy-to-use interface where they can manage their profiles, update their skill sets, and apply for job postings. They also require the ability to track the status of their applications and receive email notifications about updates, such as whether they have been shortlisted or rejected. Additionally, the system must protect sensitive student data.
- 2.** Companies, on the other hand, need a dashboard that allows them to manage their company profiles, post job openings, and view applications from interested students. They should be able to filter and shortlist candidates based on their qualifications and skills.
- 3.** Administrators are responsible for overseeing the entire system. They need features for managing user accounts (both students and companies) and monitoring the overall placement management. Admins should have access to detailed reports and metrics, such as total number of applications, companies, and students.

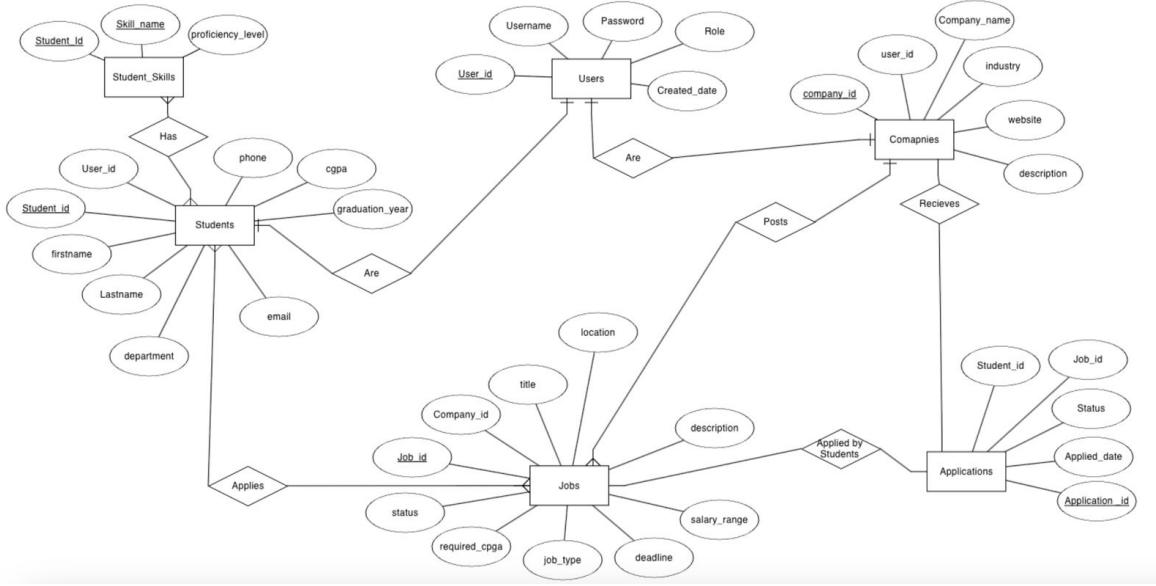
### **NON FUNCTIONAL REQUIREMENTS :**

- 1.** The system should be able to handle a large number of concurrent users (students, companies, and admins) without significant performance degradation.
- 2.** Application load times should be minimal, and queries should be optimized for speed.
- 3.** The system should implement secure authentication, using modern password hashing algorithms (e.g., Bcrypt) for user passwords.
- 4.** The user interface should be responsive, ensuring usability on various devices like desktops, tablets, and mobile phones.
- 5.** The code should be modular and well-documented to facilitate easy maintenance and updates by developers.
- 6.** Regular backups should be made of the database and other critical data to prevent loss in case of a failure.

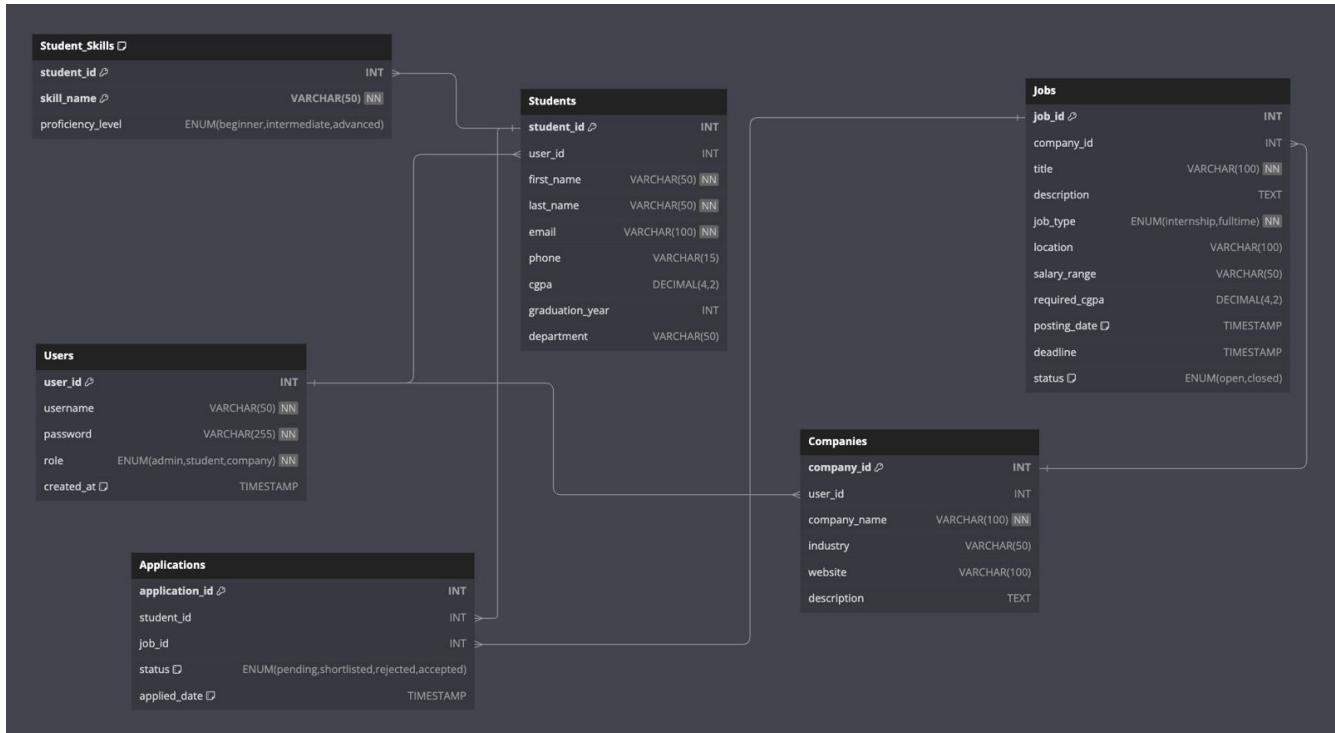
### **3. LIST OF SOFTWARES/TOOLS/PROGRAMMING LANGUAGES USED**

- **Python 3.x:** Core language for application logic
- **Streamlit:** For creating the web-based interface
- **MySQL:** Backend database for data storage and retrieval
- **Bcrypt:** Secure password hashing for user authentication
- **dotenv:** To manage environment variables securely
- **Python MySQL Connector:** To facilitate database interactions from Python
- **SMTPlib and MIME:** Email Notifications
- **Visual Studio Code:** Code Editor
- **Git :** Version Control

## 4. ER MODEL



## 5. ER MODEL TO RELATIONAL MAPPING



## 6. DDL STATEMENTS

- **Create Commands :**

```
```
CREATE TABLE IF NOT EXISTS Users (
    user_id INT PRIMARY KEY AUTO_INCREMENT,
    username VARCHAR(50) UNIQUE NOT NULL,
    password VARCHAR(255) NOT NULL,
    role ENUM('admin', 'student', 'company') NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE IF NOT EXISTS Students (
    student_id INT PRIMARY KEY AUTO_INCREMENT,
    user_id INT,
    first_name VARCHAR(50) NOT NULL,
    last_name VARCHAR(50) NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    phone VARCHAR(15),
    cgpa DECIMAL(4,2),
    graduation_year INT,
    department VARCHAR(50),
    FOREIGN KEY (user_id) REFERENCES Users(user_id)
);

CREATE TABLE IF NOT EXISTS Companies (
    company_id INT PRIMARY KEY AUTO_INCREMENT,
    user_id INT,
    company_name VARCHAR(100) NOT NULL,
    industry VARCHAR(50),
    website VARCHAR(100),
    description TEXT,
    FOREIGN KEY (user_id) REFERENCES Users(user_id)
);

CREATE TABLE IF NOT EXISTS Jobs (
    job_id INT PRIMARY KEY AUTO_INCREMENT,
    company_id INT,
    title VARCHAR(100) NOT NULL,
    description TEXT,
    job_type ENUM('internship', 'fulltime') NOT NULL,
    location VARCHAR(100),
    salary_range VARCHAR(50),
    required_cgpa DECIMAL(4,2),
    posting_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    deadline TIMESTAMP,
    status ENUM('open', 'closed') DEFAULT 'open',
    FOREIGN KEY (company_id) REFERENCES Companies(company_id)
```

```
);

CREATE TABLE IF NOT EXISTS Applications (
    application_id INT PRIMARY KEY AUTO_INCREMENT,
    student_id INT,
    job_id INT,
    status ENUM('pending', 'shortlisted', 'rejected', 'accepted') DEFAULT 'pending',
    applied_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (student_id) REFERENCES Students(student_id),
    FOREIGN KEY (job_id) REFERENCES Jobs(job_id)
);

CREATE TABLE IF NOT EXISTS Student_Skills (
    student_id INT,
    skill_name VARCHAR(50) NOT NULL,
    proficiency_level ENUM('beginner', 'intermediate', 'advanced'),
    PRIMARY KEY (student_id, skill_name),
    FOREIGN KEY (student_id) REFERENCES Students(student_id)
);
```

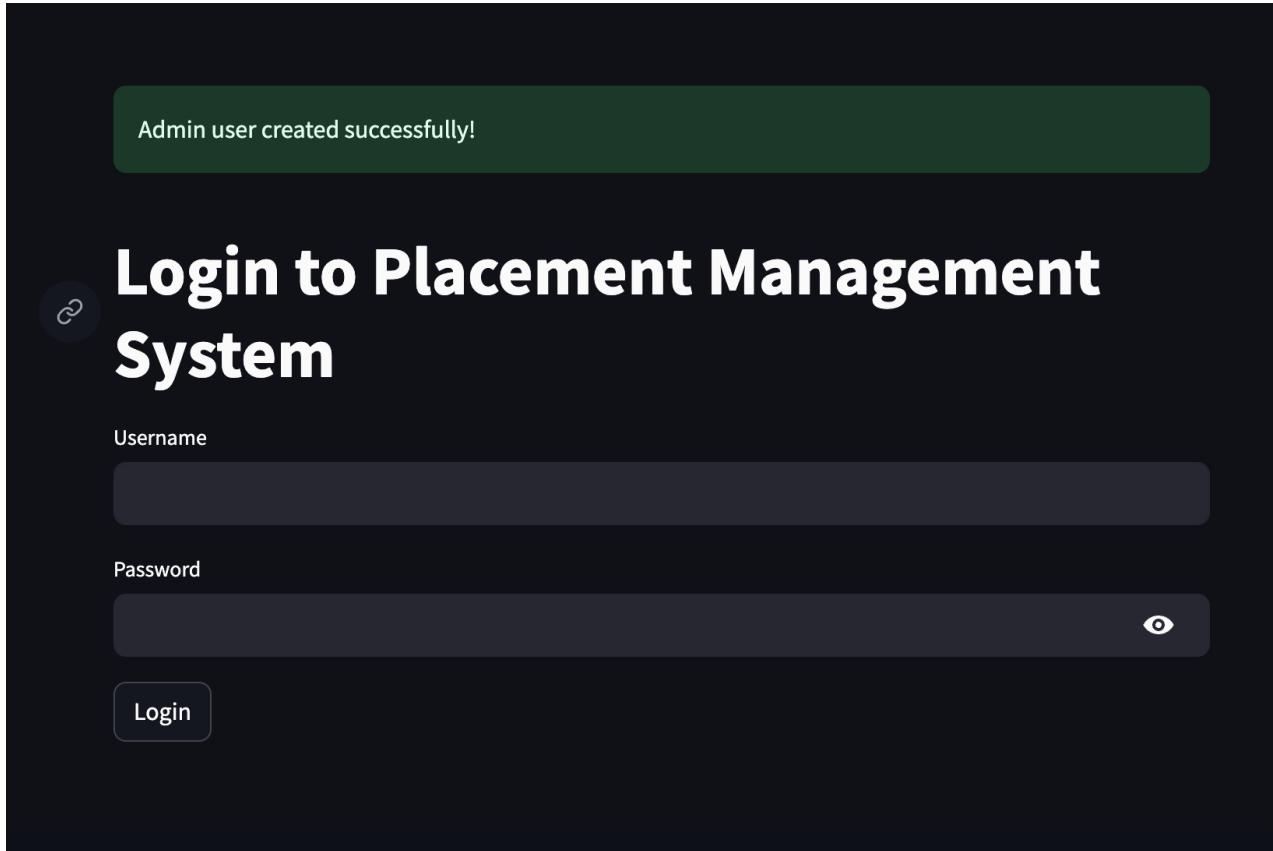
```

When The DB is initialized for the first time , these commands are run and a user with role **admin**

**username : admin**

**Password : admin123** is created.

## 7. DML STATEMENTS(CRUD)



Admin user created on initial initialization of database with credentials

Username : admin

Password : admin123

```
cursor = conn.cursor()

# Check if admin exists
cursor.execute("SELECT COUNT(*) FROM Users WHERE username = 'admin'")
admin_exists = cursor.fetchone()[0]

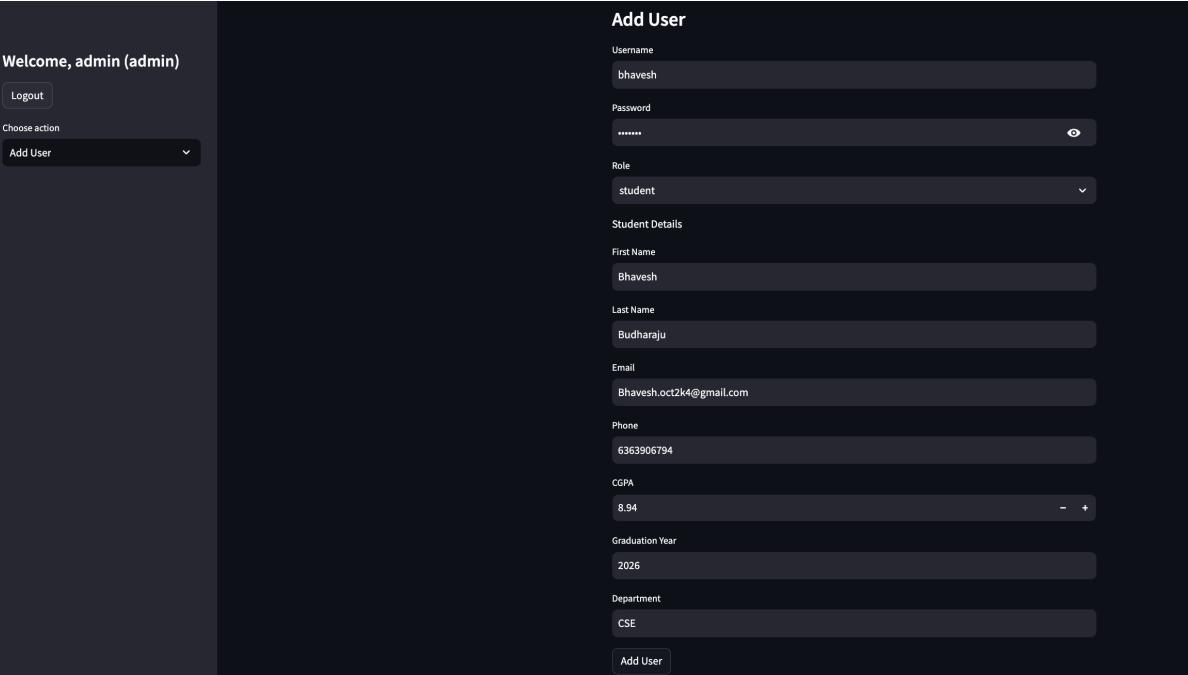
if not admin_exists:
    # Create admin with hashed password
    hashed_password = bcrypt.hashpw('admin123'.encode('utf-8'), bcrypt.gensalt())

    cursor.execute("""
        INSERT INTO Users (username, password, role)
        VALUES (%s, %s, %s)
    """, ('admin', hashed_password.decode('utf-8'), 'admin'))

    conn.commit()
    st.success("Admin user created successfully!")
```

# Insertion :

## a. Inserting student from admin



```
hashed_password = hash_password(password)

# Insert into Users table
cursor.execute("INSERT INTO Users (username, password, role) VALUES (%s, %s, %s)",
               (username, hashed_password, role))
conn.commit()
user_id = cursor.lastrowid # Get the last inserted user_id
```

```
if role == "student":
    cursor.execute("""
        INSERT INTO Students (user_id, first_name, last_name, email, phone, cgpa, graduation_year, department)
        VALUES (%s, %s, %s, %s, %s, %s, %s, %s)
    """, (user_id, additional_details['first_name'], additional_details['last_name'],
          additional_details['email'], additional_details['phone'],
          additional_details['cgpa'], additional_details['graduation_year'],
          additional_details['department']))
```

```
mysql> select * from users;
+-----+-----+-----+
| user_id | username | password |
+-----+-----+-----+
| 1 | admin | $2b$12$0o4I8C/mJUaGsJWCIQq6xuHMjJ1mcYmuZxR0/GFzTT8nZgIIGX7le |
| 2 | bhavesh | $2b$12$TxhBXwXlrHog3i271vyN.CQyBcRuP9ifUbpp/pkbtbHHDBbjzP6qK |
+-----+-----+-----+
2 rows in set (0.00 sec)

mysql> select * from students;
+-----+-----+-----+-----+-----+-----+-----+-----+
| student_id | user_id | first_name | last_name | email | phone | cgpa | graduation_year | department |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | 2 | Bhavesh | Budharaju | Bhavesh.oct2k4@gmail.com | 6363906794 | 8.94 | 2026 | CSE |
+-----+-----+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

User bhavesh added successfully!

Welcome, admin (admin)

Logout

Choose action

View All Users

	user_id	username	password	role
0	1	admin	\$2b\$12So04i8C/mJUaGsJWCiQq6xuHMjJ1mcYmuZxR0/GFzTT8nZgIGX7le	admin
1	2	bhavesh	\$2b\$12STXhBXwXlrHog3i271vyyN.CQyBcRuP9ifUbpp/pkbtbHHDBjzP6qK	student

## b. Inserting company from admin

Welcome, admin (admin)

Logout

Choose action

Add User

### Admin Dashboard

#### Add User

Username  
google

Password  
\*\*\*\*\*

Role  
company

Company Details

Company Name  
Google

Industry  
Technology

Website  
www.google.com

Description  
Make life easier with a little help from our products. Google's software engineers develop the next-generation technologies that change how billions of users connect, explore, and interact with information and one another. Our products need to handle information at massive scale, and extend well beyond web search.

Add User

User google added successfully!

```

hashed_password = hash_password(password)

# Insert into Users table
cursor.execute("INSERT INTO Users (username, password, role) VALUES (%s, %s, %s)",
               (username, hashed_password, role))
conn.commit()
user_id = cursor.lastrowid # Get the last inserted user_id

elif role == "company":
    cursor.execute("""
        INSERT INTO Companies (user_id, company_name, industry, website, description)
        VALUES (%s, %s, %s, %s, %s)
    """, (user_id, additional_details['company_name'], additional_details['industry'],
          additional_details['website'], additional_details['description']))

conn.commit()
st.success(f"User {username} added successfully!")

```

The screenshot shows the Admin Dashboard interface. On the left, there's a sidebar with 'Welcome, admin (admin)' and a 'Logout' button. Below it is a dropdown menu titled 'Choose action' with 'View All Users' selected. The main area is titled 'Admin Dashboard' and contains a table with columns: user\_id, username, password, and role. The table has four rows of data. To the right of the table, there's a preview of the MySQL database structure.

	user_id	username	password	role
0	1	admin	\$2b\$12\$0o4i8C/mJuGsJWC1Qq6xuHMjJ1mcYmuZxRO/GFzTT8nZgIGX7le	admin
1	2	bhavesh	\$2b\$12\$TXhBxwXlrHog3i271vyvN.CQyBcRuP9ifUbpp/pkbtbHHDBjzP6qK	student
2	3	dhanush	\$2b\$12\$orPGPCBlnuS5rT4A6VnksODd5rZeikQdoaa1DE24BT9HfT9AcMsK	student
3	4	google	\$2b\$12\$FVtEsOp.JsBHTAMY5RaS./S/WhLyKog/xBG0v3ycxKgyBvnOZ.ym	company

```

mysql> select * from users;
+-----+-----+-----+-----+
| user_id | username | password |
+-----+-----+-----+-----+
| 1 | admin | $2b$12$0o4i8C/mJuGsJWC1Qq6xuHMjJ1mcYmuZxRO/GFzTT8nZgIGX7le |
| 2 | bhavesh | $2b$12$TXhBxwXlrHog3i271vyvN.CQyBcRuP9ifUbpp/pkbtbHHDBjzP6qK |
| 3 | dhanush | $2b$12$orPGPCBlnuS5rT4A6VnksODd5rZeikQdoaa1DE24BT9HfT9AcMsK |
| 4 | google | $2b$12$FVtEsOp.JsBHTAMY5RaS./S/WhLyKog/xBG0v3ycxKgyBvnOZ.ym |
+-----+-----+-----+-----+
4 rows in set (0.00 sec)

mysql> select * from companies;
+-----+-----+-----+-----+
| company_id | user_id | company_name | industry | website | description |
+-----+-----+-----+-----+
| 1 | 4 | Google | Technology | www.google.com | Make life easier with a little help from our products. Google's software engineers develop the next-generation technologies that change how billions of users connect, explore, and interact with information and one another. Our products need to handle information at massive scale, and extend well beyond web search. |
+-----+-----+-----+-----+
1 row in set (0.00 sec)

```

### c. Inserting another admin from admin

Welcome, admin (admin)

[Logout](#)

Choose action

[Add User](#)

## Admin Dashboard

**Add User**

Username	<input type="text" value="admin1"/>
Password	<input type="password" value="....."/> <a href="#">eye</a>
Role	<input type="text" value="admin"/>
<a href="#">Add User</a>	

User admin1 added successfully!

Welcome, admin (admin)

[Logout](#)

Choose action

[View All Users](#)

## Admin Dashboard

	user_id	username	password	role
0	1	admin	\$2b\$12\$o04l8C/mJUaGsJWCIQq6xuHMjJ1mcYmuZxRO/GFzTT8nZgIlGX7le	admin
1	2	bhavesh	\$2b\$12\$TXhBXwXlrHog3i271vyvN.CQyBcRuP9ifUbpp/pkbtbHHDbjzP6qK	student
2	3	dhanush	\$2b\$12\$orPGPCBlnuS5rT4A6VnksODd5r2eikQdoaa1DE24BT99Hft9AcMsK	student
3	4	google	\$2b\$12\$FvtEsOp.JsBhtAMY58RaS./S/WhLyKog/xB60v3ycxKgyBvnOZ.ym	company
4	5	microsoft	\$2b\$12\$ieZI3Vte00CxZku3TGTdUe5wIoIpisZzWMmtzbWKCxUiPcGLbPcsa	company
5	6	admin1	\$2b\$12\$dg4zOhxrbeBVsxzWD47.L0nLwUZlg4VsPLEZKStkLNhsAk9SbuOae	admin

```
mysql> select * from users;
+-----+-----+-----+-----+-----+-----+
| user_id | username | password | role | created_at |
+-----+-----+-----+-----+-----+
| 1 | admin | $2b$12$o04l8C/mJUaGsJWCIQq6xuHMjJ1mcYmuZxRO/GFzTT8nZgIlGX7le | admin | 2024-11-12 09:22:50 |
| 2 | bhavesh | $2b$12$TXhBXwXlrHog3i271vyvN.CQyBcRuP9ifUbpp/pkbtbHHDbjzP6qK | student | 2024-11-12 09:33:22 |
| 3 | dhanush | $2b$12$orPGPCBlnuS5rT4A6VnksODd5r2eikQdoaa1DE24BT99Hft9AcMsK | student | 2024-11-12 09:38:57 |
| 4 | google | $2b$12$FvtEsOp.JsBhtAMY58RaS./S/WhLyKog/xB60v3ycxKgyBvnOZ.ym | company | 2024-11-12 09:43:19 |
| 5 | microsoft | $2b$12$ieZI3Vte00CxZku3TGTdUe5wIoIpisZzWMmtzbWKCxUiPcGLbPcsa | company | 2024-11-12 09:48:38 |
| 6 | admin1 | $2b$12$dg4zOhxrbeBVsxzWD47.L0nLwUZlg4VsPLEZKStkLNhsAk9SbuOae | admin | 2024-11-12 09:49:56 |
+-----+-----+-----+-----+-----+
6 rows in set (0.00 sec)
```

```
hashed_password = hash_password(password)

# Insert into Users table
cursor.execute("INSERT INTO Users (username, password, role) VALUES (%s, %s, %s)",
               (username, hashed_password, role))
conn.commit()
user_id = cursor.lastrowid # Get the last inserted user_id
```

**What if User is already present in database and redundant user is inserted?**

## Admin Dashboard

**Add User**

Username  
admin1

Password  
\*\*\*\*\*

Role  
admin

**Add User**

Error adding user: 1062 (23000): Duplicate entry 'admin1' for key 'users.username'

### a. Insertion from company – post job

Company Dashboard

Post a Job

Job Title: Data Engineer

Job Description: Work of Distributed Technologies.  
Must have a good knowledge of Kafka and Spark

Job Type: internship

Location: Bangalore

Salary Range: 100000

Minimum CGPA: 8.50

Application Deadline: 2024/11/14

Post Job

```

insert_query = """
    INSERT INTO Jobs (
        company_id, title, description, job_type,
        location, salary_range, required_cgpa, deadline
    ) VALUES (%s, %s, %s, %s, %s, %s, %s, %s)
"""
cursor.execute(insert_query,
               (company_id, title, description, job_type,
                location, salary_range, required_cgpa, deadline))
conn.commit()
st.success("Job posted successfully!")

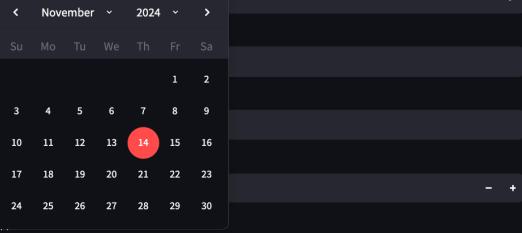
```

If Job Already posted with the same requirements and description

**Post a Job**

Job Title  
Data Engineer

Job Description  
Work of Distributed Technologies.  
Must have a good knowledge of Kafka and Spark

Job Type  


2024/11/14

**Post Job**

A job with the same title, description, job type, and location already exists.

```
check_query = """
    SELECT * FROM Jobs
    WHERE company_id = %s
    AND title = %s
    AND description = %s
    AND job_type = %s
    AND location = %s
"""

cursor.execute(check_query,
               (company_id, title, description, job_type, location))
existing_job = cursor.fetchone()

if existing_job:
    st.warning("A job with the same title, description, job type, and location already exists.")
```

```
mysql> select * from jobs
| -> ;
+-----+-----+-----+-----+-----+-----+-----+
| job_id | company_id | title | description | job_type | location | salary_range | required_cgpa |
| posting_date | deadline | status |
+-----+-----+-----+-----+-----+-----+-----+
| 1 | 1 | Data Engineer | Work of Distributed Technologies .  
Must have a good knowledge of Kafka and Spark | internship | Bangalore | 100000 | 8.50 | 2024-11-12 09:59:17 | 2024-11-14 00:00:00 | open |
+-----+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

## a. Insertion from student – skills

Welcome, bhavesh  
(student)

Logout

CGPA  
8.94

Graduation Year  
2026

Department  
CSE

Update Profile

Skills Management

Add New Skills:

Number of skills to add  
3

Skill 1 Proficiency 1  
python advanced

Skill 2 Proficiency 2  
mysql intermediate

Skill 3 Proficiency 3  
kafka beginner

Add Skills

## Skills Management

Your Current Skills:

- Delete kafka
- Delete mysql
- Delete python

	skill_name	proficiency_level
0	kafka	beginner
1	mysql	intermediate
2	python	advanced

```
[mysql] > select * from student_skills;
+-----+-----+-----+
| student_id | skill_name | proficiency_level |
+-----+-----+-----+
|       1 | kafka      | beginner          |
|       1 | mysql      | intermediate     |
|       1 | python     | advanced          |
+-----+-----+-----+
3 rows in set (0.00 sec)
```

# SELECT ( READ) :

## ADMIN :

### a. View all users

```
def view_all_users():
    conn = get_database_connection()
    if conn:
        cursor = conn.cursor(dictionary=True)
        cursor.execute("SELECT * FROM Users")
        users = cursor.fetchall()
        cursor.close()
        conn.close()
        if users:
            users_df = pd.DataFrame(users)
            st.dataframe(users_df)
        else:
            st.write("No users found.")
    else:
        st.write("Database connection failed.")
```

### Admin Dashboard

	user_id	username	password	role
0	1	admin	\$2b\$12\$o04i8C/mJUaGsJWC1Qq6xuHMj1mcYmuZxRO/GFzTT8nZglGX7le	admin
1	2	bhavesh	\$2b\$12\$TXhBXwXlrHog3i271vyyN.CQyBcRuP9ifubpp/pkbtbHDBjzP6qK	student
2	3	dhanush	\$2b\$12\$orPGPCBlnuS5rT4A6VnksODd5r2eikQdoaa1DE24BT9HfT9acMsK	student
3	4	google	\$2b\$12\$FVtEsOp.JsBHTAMY58RaS./S/WhLyKog/xBG0v3ycxKgyBvnOZ.ym	company
4	5	microsoft	\$2b\$12\$leZl3VteOOCxZku3TGtdUe5w1olpisZzWMmtzbWKCxUlpcGlbPcsa	company
5	6	admin1	\$2b\$12\$dg4zOhxrbeBVszWD47.LOnLwUZlg4VsPLEZKStkLNhsAk9SbUOAe	admin

### b. Statistics

Welcome, admin (admin)

Logout

Choose action

Statistics

### Admin Dashboard

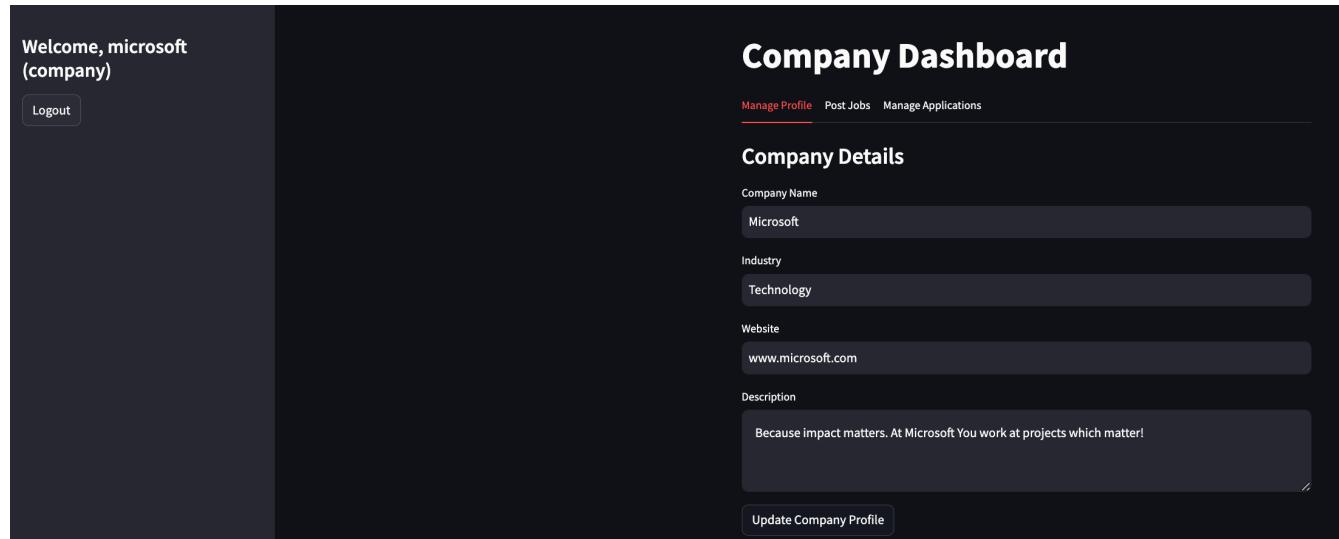
```
{
  "total_students": 2,
  "total_companies": 2,
  "total_jobs": 1
}
```

```
def view_statistics():
    conn = get_database_connection()
    cursor = conn.cursor(dictionary=True)
    cursor.execute("""
        SELECT
            (SELECT COUNT(*) FROM Students) as total_students,
            (SELECT COUNT(*) FROM Companies) as total_companies,
            (SELECT COUNT(*) FROM Jobs) as total_jobs
    """)

    stats = cursor.fetchone() # fetchone since we only expect one row
    st.write(stats)
```

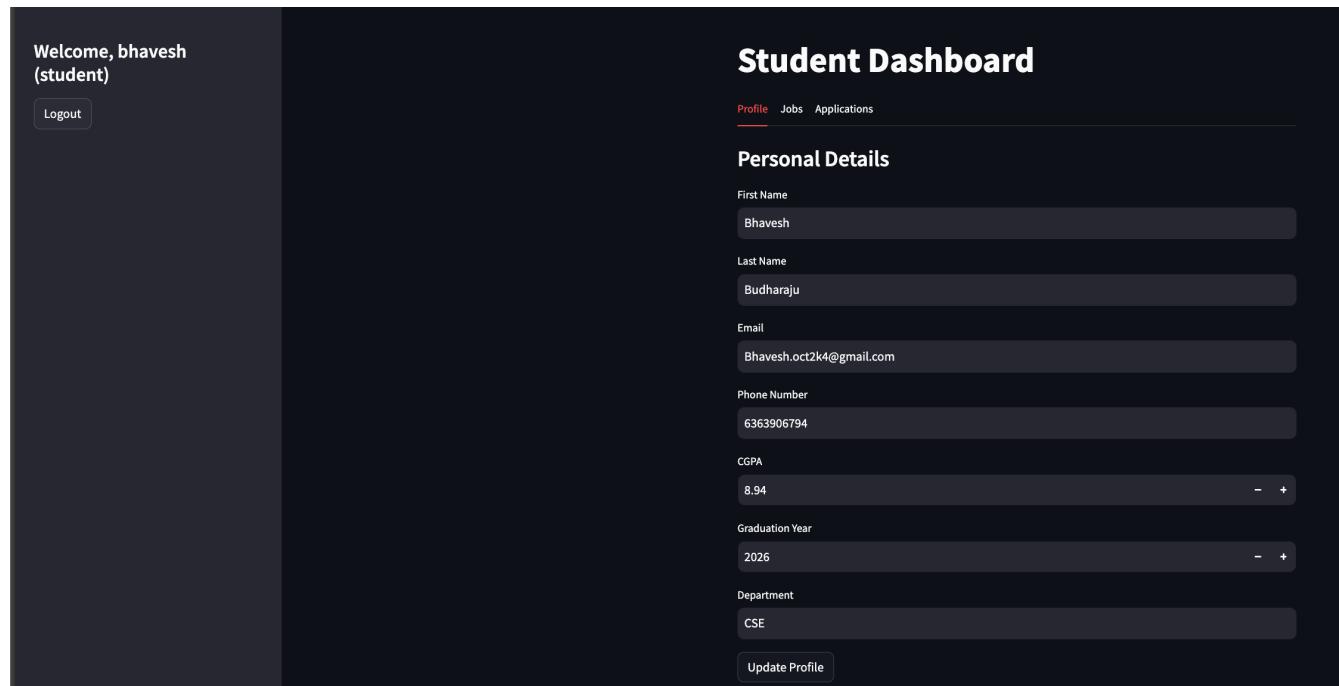
## COMPANY :

```
# Fetch existing details based on the user_id
cursor.execute("SELECT * FROM Companies WHERE user_id = %s", (st.session_state.user_id,))
company = cursor.fetchall() # Use fetchall() to ensure all results are consumed
```



The screenshot shows a company dashboard interface. On the left, there's a sidebar with a dark background and white text. It says "Welcome, microsoft (company)" and has a "Logout" button. The main area is titled "Company Dashboard" and has three tabs: "Manage Profile" (which is active), "Post Jobs", and "Manage Applications". Under "Company Details", there are fields for Company Name (Microsoft), Industry (Technology), Website (www.microsoft.com), and a Description box containing the text "Because impact matters. At Microsoft You work at projects which matter!". At the bottom right of the main area is a "Update Company Profile" button.

## STUDENT :



The screenshot shows a student dashboard interface. On the left, there's a sidebar with a dark background and white text. It says "Welcome, bhavesh (student)" and has a "Logout" button. The main area is titled "Student Dashboard" and has three tabs: "Profile" (which is active), "Jobs", and "Applications". Under "Personal Details", there are fields for First Name (Bhavesh), Last Name (Budharaju), Email (Bhavesh.oct2k4@gmail.com), Phone Number (6363906794), CGPA (8.94 with a slider), Graduation Year (2026 with a slider), Department (CSE), and an "Update Profile" button.

```
# Fetch existing details based on the user_id
cursor.execute("SELECT * FROM Students WHERE user_id = %s", (st.session_state.user_id,))
student = cursor.fetchone()
```

```
student_id = student_data['student_id']
cursor.execute("SELECT * FROM Jobs WHERE status='open'")
jobs = cursor.fetchall()
for job in jobs:
    st.write(job)
    if st.button("Apply", key=job['job_id']):
        cursor.execute("INSERT INTO Applications (student_id, job_id) VALUES (%s, %s)",
                       (student_id, job['job_id']))
        conn.commit()
        st.success(f"Applied for {job['title']}.")

cursor.close()
conn.close()
```

The screenshot shows a dark-themed web application interface. At the top, there is a navigation bar with three items: "Profile", "Jobs", and "Applications". The "Jobs" item is underlined, indicating it is the active page. Below the navigation bar, the title "Available Jobs" is displayed in a large, bold, white font. A detailed job listing is shown below the title:

```
▼ {
  "job_id" : 1
  "company_id" : 1
  "title" : "Data Engineer"
  "description" : "Work of Distributed Technologies .
  Must have a good knowledge of Kafka and Spark"
  "job_type" : "internship"
  "location" : "Bangalore"
  "salary_range" : "100000"
  "required_cgpa" : "Decimal('8.50')"
  "posting_date" : "datetime.datetime(2024, 11, 12, 9, 59, 17)"
  "deadline" : "datetime.datetime(2024, 11, 14, 0, 0)"
  "status" : "open"
}
```

At the bottom left of the job listing is a button labeled "Apply".

## Application Status

```
▼ [ ↴
  ▼ 0 : { ↴
    "application_id" : 1
    "job_title" : "Data Engineer"
    "company_name" : "Google"
    "job_type" : "internship"
    "location" : "Bangalore"
    "application_status" : "pending"
    "applied_date" : "datetime.datetime(2024, 11, 12, 12, 21, 47)"
  }
```

```
cursor.execute("SELECT student_id FROM Students WHERE user_id = %s", (st.session_state.user['user_id'],))
student_data = cursor.fetchone()

if not student_data:
    st.error("Student profile not found.")
    return

student_id = student_data['student_id']
cursor.execute("""
    CALL GetStudentApplicationDetails(%s)
""", (student_id,))
applications = cursor.fetchall()
st.write(applications)
cursor.close()
conn.close()
```

# UPDATE :

## Company :

The screenshot shows a dark-themed company dashboard interface. At the top, there are three navigation links: 'Manage Profile' (in red), 'Post Jobs', and 'Manage Applications'. Below this is a section titled 'Company Details'.

**Company Name:** Google

**Industry:** Technology and Research (highlighted with a red border) | Press Enter to apply

**Website:** www.google.com

**Description:** Make life easier with a little help from our products. Google's software engineers develop the next-generation technologies that change how billions of users connect, explore, and interact with information and one another. Our products need to handle information at massive scale, and extend well beyond web search.

**Buttons:** Update Company Profile

## Before :

```
+-----+-----+-----+-----+-----+
| 1 | 4 | Google | Technology | www.google.com | Make life easier with a little help from our products. Google's software engineers develop the next-generation technologies that change how billions of users connect, explore, and interact with information and one another. Our products need to handle information at massive scale, and extend well beyond web search. |
+-----+-----+-----+-----+-----+
```

## After:

```
mysql> select * from companies;
+-----+-----+-----+-----+-----+
| company_id | user_id | company_name | industry | website | description
+-----+-----+-----+-----+-----+
| 1 | 4 | Google | Technology and Research | www.google.com | Make life easier with a little help from our products. Google's software engineers develop the next-generation technologies that change how billions of users connect, explore, and interact with information and one another. Our products need to handle information at massive scale, and extend well beyond web search. |
| 2 | 5 | Microsoft | Technology | www.microsoft.com | Because impact matters. At Microsoft You work at projects which matter!
+-----+-----+-----+-----+-----+
```

**Google : Industry changed from ‘Technology’ to ‘Technology and Research’**

# Company Dashboard

Manage Profile Post Jobs Manage Applications

## Manage Applications

Applicant: Bhavesh Budharaju

Job: Data Engineer

Current Status: rejected

Update Status

rejected

Update

Status updated to  
rejected!

```
if st.button("Update", key=f"update_{app['application_id']}"):
    cursor.execute(
        "UPDATE Applications SET status = %s WHERE application_id = %s",
        (new_status, app['application_id']))
    conn.commit()
```

## Student

Profile Jobs Applications

## Personal Details

First Name  
Bhavesh

Last Name  
Budharaju

Email  
Bhavesh.2k4@gmail.com

Phone Number  
6363906794

CGPA  
8.94 - +

Graduation Year  
2026 - +

Department  
CSE

**Update Profile**

Profile updated successfully.

```
mysql> select * from students;
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| student_id | user_id | first_name | last_name | email | phone | cgpa | graduation_year | department |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | 2 | Bhavesh | Budharaju | Bhavesh.oct2k4@gmail.com | 6363906794 | 8.94 | 2026 | CSE |
| 2 | 3 | Dhanush | Suresh Jettipalle | jsdhanush.2004@gmail.com | 6366693334 | 9.30 | 2026 | CSE |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)
```

```
mysql> select * from students;
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| student_id | user_id | first_name | last_name | email | phone | cgpa | graduation_year | department |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | 2 | Bhavesh | Budharaju | Bhavesh.2k4@gmail.com | 6363906794 | 8.94 | 2026 | CSE |
| 2 | 3 | Dhanush | Suresh Jettipalle | jsdhanush.2004@gmail.com | 6366693334 | 9.30 | 2026 | CSE |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)
```

```
if st.button("Update Profile"):
    cursor.execute("""
        UPDATE Students
        SET first_name=%s, last_name=%s, email=%s, phone=%s, cgpa=%s, graduation_year=%s, department=%s
        WHERE user_id=%s
    """, (first_name, last_name, email, phone, cgpa, graduation_year, department, st.session_state.user_id))
    conn.commit()
    st.success("Profile updated successfully.")
```

# DELETE

## Admin:

The screenshot shows the Admin Dashboard interface. On the left, there's a sidebar with "Welcome, admin (admin)" and a "Logout" button. A dropdown menu under "Choose action" has "Delete User" selected. The main area is titled "Admin Dashboard" and has a form for "User ID to delete" with the value "6". A red "Delete User" button is present. Below it, a green success message box says "User ID 6 deleted successfully!".

```
[mysql] select * from users;
+-----+-----+-----+-----+-----+
| user_id | username | password | role | created_at |
+-----+-----+-----+-----+-----+
| 1 | admin | $2b$12$o04I8C/mJUaGsJWC1Qq6xuHMjJ1mcYmuZxRO/GFzTT8nZgIlGX7le | admin | 2024-11-12 09:22:50 |
| 2 | bhavesh | $2b$12$TXhBXwXlrHog3i271vyyN.CQyBcRuP9ifUbpp/pkbtbHHDBjzP6qK | student | 2024-11-12 09:33:22 |
| 3 | dhanush | $2b$12$orPGPCBlnuS5rT4A6VnksODd5r2eikQdoaa1DE24BT99HfT9AcMsK | student | 2024-11-12 09:38:57 |
| 4 | google | $2b$12$FvtEsOp.JsBhtAMY58RaS./S/WnLyKog/xBG0v3yckGkyBvnOZ.ym | company | 2024-11-12 09:43:19 |
| 5 | microsoft | $2b$12$IeZI3Vte0OCxZku3TGTdUe5w1oIpisZzWMmtzbWKCxUIpCGLbPcsa | company | 2024-11-12 09:48:38 |
| 6 | admin1 | $2b$12$dg4z0hxrbbeBVsxzWD47.LOnlwUZlg4VsPLEZKStkLNHsAk9SbUOAe | admin | 2024-11-12 09:49:56 |
+-----+-----+-----+-----+-----+
6 rows in set (0.00 sec)

[mysql] select * from users;
+-----+-----+-----+-----+-----+
| user_id | username | password | role | created_at |
+-----+-----+-----+-----+-----+
| 1 | admin | $2b$12$o04I8C/mJUaGsJWC1Qq6xuHMjJ1mcYmuZxRO/GFzTT8nZgIlGX7le | admin | 2024-11-12 09:22:50 |
| 2 | bhavesh | $2b$12$TXhBXwXlrHog3i271vyyN.CQyBcRuP9ifUbpp/pkbtbHHDBjzP6qK | student | 2024-11-12 09:33:22 |
| 3 | dhanush | $2b$12$orPGPCBlnuS5rT4A6VnksODd5r2eikQdoaa1DE24BT99HfT9AcMsK | student | 2024-11-12 09:38:57 |
| 4 | google | $2b$12$FvtEsOp.JsBhtAMY58RaS./S/WnLyKog/xBG0v3yckGkyBvnOZ.ym | company | 2024-11-12 09:43:19 |
| 5 | microsoft | $2b$12$IeZI3Vte0OCxZku3TGTdUe5w1oIpisZzWMmtzbWKCxUIpCGLbPcsa | company | 2024-11-12 09:48:38 |
+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)
```

```
cursor.execute("SELECT role FROM Users WHERE user_id = %s", (user_id,))
user = cursor.fetchone()

if user:
    role = user['role']
    # Delete from Students or Companies based on role
    if role == 'student':
        cursor.execute("DELETE FROM Students WHERE user_id = %s", (user_id,))
    elif role == 'company':
        cursor.execute("DELETE FROM Companies WHERE user_id = %s", (user_id,))

    # Then delete from Users table
    cursor.execute("DELETE FROM Users WHERE user_id = %s", (user_id,))
    conn.commit()
    st.success(f"User ID {user_id} deleted successfully!")
```

## Student

```
DELETE FROM Student_Skills  
WHERE student_id = %s AND skill_name = %s
```

### Skills Management

Your Current Skills:

- Delete kafka
- Delete mysql
- Delete python

	skill_name	proficiency_level
0	kafka	beginner
1	mysql	intermediate
2	python	advanced

[Remove Selected Skills](#)

### Skills Management

Your Current Skills:

- Delete mysql
- Delete python

	skill_name	proficiency_level
0	mysql	intermediate
1	python	advanced

Before :

```
[mysql]> select * from student_skills;  
+-----+-----+-----+  
| student_id | skill_name | proficiency_level |  
+-----+-----+-----+  
| 1 | kafka | beginner |  
| 1 | mysql | intermediate |  
| 1 | python | advanced |  
+-----+-----+-----+  
3 rows in set (0.00 sec)
```

After :

```
[mysql]> select * from student_skills;  
+-----+-----+-----+  
| student_id | skill_name | proficiency_level |  
+-----+-----+-----+  
| 1 | mysql | intermediate |  
| 1 | python | advanced |  
+-----+-----+-----+
```

## 8. QUERIES (JOIN QUERY, AGGREGATE FUNCTION QUERIES AND NESTED QUERY)

JOIN :

```
# Fetch applications
cursor.execute("""
    SELECT
        a.application_id,
        s.first_name,
        s.last_name,
        s.student_id,
        j.title,
        a.status
    FROM Applications a
    JOIN Students s ON a.student_id = s.student_id
    JOIN Jobs j ON a.job_id = j.job_id
    WHERE j.company_id = %s
""", (company_id,))
```

The screenshot shows a dark-themed dashboard titled "Company Dashboard". At the top, there are three navigation links: "Manage Profile", "Post Jobs", and "Manage Applications", with "Manage Applications" being underlined. Below this, the title "Manage Applications" is displayed. There are two entries listed:

- Applicant:** Bhavesh Budharaju  
**Job:** Data Engineer  
**Current Status:** rejected
- Applicant:** Bhavesh Budharaju  
**Job:** Data Engineer  
**Current Status:** pending

For each entry, there is a "Update Status" dropdown and an "Update" button. The first entry's dropdown shows "rejected" and the second one shows "pending".

## Aggregate Function Queries:

```
SELECT COUNT(*) |  
FROM Student_Skills  
WHERE student_id = %s AND skill_name = %s
```

## Skills Management

Your Current Skills:

Delete mysql  
 Delete python

	skill_name	proficiency_level
0	mysql	intermediate
1	python	advanced

## Nested Query :

```
SELECT  
    (SELECT COUNT(*) FROM Students) as total_students,  
    (SELECT COUNT(*) FROM Companies) as total_companies,  
    (SELECT COUNT(*) FROM Jobs) as total_jobs  
....)
```

## Admin Dashboard

```
▼ {  
  "total_students" : 2  
  "total_companies" : 2  
  "total_jobs" : 1  
}
```

## 9. STORED PROCEDURE, FUNCTIONS AND TRIGGERS

```
TRIGGER_SCRIPT = """
CREATE TRIGGER before_application_insert
BEFORE INSERT ON Applications
FOR EACH ROW
BEGIN
    DECLARE student_cgpa DECIMAL(4,2);
    DECLARE job_min_cgpa DECIMAL(4,2);

    SELECT cgpa INTO student_cgpa
    FROM Students
    WHERE student_id = NEW.student_id;

    SELECT required_cgpa INTO job_min_cgpa
    FROM Jobs
    WHERE job_id = NEW.job_id;

    IF student_cgpa < job_min_cgpa THEN
        SET NEW.status = 'rejected';
    END IF;
END;
....
```

### Company Dashboard

Manage Profile Post Jobs Manage Applications

#### Post a Job

Job Title  
Intern

Job Description  
SDE

Job Type  
internship

Location  
Bangalore

Salary Range  
75000

Minimum CGPA  
9.00

Application Deadline  
2024/11/12

Post Job

{  
 "job\_id" : 2  
 "company\_id" : 2  
 "title" : "Intern"  
 "description" : "SDE"  
 "job\_type" : "internship"  
 "location" : "Bangalore"  
 "salary\_range" : "75000"  
 "required\_cgpa" : "Decimal('9.00')"  
 "posting\_date" : "datetime.datetime(2024, 11, 12, 12, 50, 55)"  
 "deadline" : "datetime.datetime(2024, 11, 12, 0, 0)"  
 "status" : "open"  
}  
  

Apply

Applied for Intern.

2 : {  
 "application\_id" : 3  
 "job\_title" : "Intern"  
 "company\_name" : "Microsoft"  
 "job\_type" : "internship"  
 "location" : "Bangalore"  
 "application\_status" : "rejected"  
 "applied\_date" : "datetime.datetime(2024, 11, 12, 12, 51, 23)"  
}

If the students gpa is less than the min gpa required for the job, the status is automatically set to rejected

```
STORED_PROC_SCRIPT = """"
CREATE PROCEDURE GetStudentApplicationDetails(IN p_student_id INT)
BEGIN
    SELECT
        a.application_id,
        j.title AS job_title,
        c.company_name,
        j.job_type,
        j.location,
        a.status AS application_status,
        a.applied_date
    FROM Applications a
    JOIN Jobs j ON a.job_id = j.job_id
    JOIN Companies c ON j.company_id = c.company_id
    WHERE a.student_id = p_student_id;
END;
....
```

# Company Dashboard

Manage Profile Post Jobs Manage Applications

## Manage Applications

<b>Applicant:</b> Bhavesh Budharaju <b>Job:</b> Data Engineer <b>Current Status:</b> rejected	<b>Update Status</b> rejected <input type="button" value="▼"/> <input type="button" value="Update"/>
<b>Applicant:</b> Bhavesh Budharaju <b>Job:</b> Data Engineer <b>Current Status:</b> pending	<b>Update Status</b> pending <input type="button" value="▼"/> <input type="button" value="Update"/>

## 10. FRONT END DEVELOPMENT (FUNCTIONALITIES/FEATURES OF THE APPLICATION)

Email :

Welcome, admin (admin)

Logout

Choose action

Add User

Add User

Username  
bhavesh

Password  
\*\*\*\*\*

Role  
student

Student Details

First Name  
Bhavesh

Last Name  
Budharaju

Email  
Bhavesh.oct2k4@gmail.com

Phone  
6363906794

CGPA  
8.94

Graduation Year  
2026

Department  
CSE

Add User

Application Status Update for Data Engineer Inbox x



bhavesh.oct2k4@gmail.com

to me ▾

Dear Student,

Your application status for the job 'Data Engineer' has been updated to: rejected.

Best Regards,  
Placement System

```
load_dotenv()
EMAIL=os.getenv('EMAIL')
PASSWORD=os.getenv('EMAIL_PASSWORD')
SMTP_SERVER = "smtp.gmail.com"
SMTP_PORT = 587

Codeium: Refactor | Explain | Generate Docstring | X
def send_application_status_email(student_id, job_title, status):
    to_email = get_student_email(student_id)
    if not to_email:
        return

    subject = f"Application Status Update for {job_title}"
    body = f"Dear Student,\n\nYour application status for the job '{job_title}' has been updated to: {status}.\n\nBest Regards,\nPlacement System"

    msg = MIMEMultipart()
    msg['From'] = EMAIL
    msg["To"] = to_email
    msg["Subject"] = subject
    msg.attach(MIMEText(body, "plain"))

    try:
        with smtplib.SMTP(SMTP_SERVER, SMTP_PORT) as server:
            server.starttls()
            server.login(EMAIL, PASSWORD)
            server.sendmail(EMAIL, to_email, msg.as_string())
    except Exception as e:
        print(f"Failed to send email: {e}")
```

## **REFERENCES/BIBLIOGRAPHY**

- <https://dev.mysql.com/doc/>
- <https://docs.streamlit.io/>
- <https://docs.python.org/>
- <https://dev.mysql.com/doc/connector-python/en/>
- <https://www.npmjs.com/package/bcrypt>
- <https://docs.python.org/3/library/smtplib.html>
- <https://github.com/Bhavesh2k4/SQLHire>
- <https://www.dotenv.org/docs/>

## **APPENDIX A DEFINITIONS, ACRONYMS AND ABBREVIATIONS**

### **Definitions:**

- User Authentication: The process of verifying the identity of a user to ensure secure access to the system.
- Stored Procedure: Stored procedures are a set of SQL statements that are stored in a database and can be executed as a single unit.
- Trigger: A database mechanism that automatically executes a predefined action when a specific database event occurs.

### **Acronyms:**

- SQL: Structured Query Language
- DBMS: Database Management System
- CRUD: Create, Read, Update, Delete Operations
- SMTP : Simple Mail Transfer Protocol