BCSE302L

DATABASE SYSTEMS

DIGITAL ASSIGNMENT

FINAL REPORT

CAMPUS RECRUITMENT MANAGEMENT SYSTEM

BY

AJJAY ADHITHYA V (22BCE1250)

AADARSH RAMAKRISHNA (22BCE1332)

ABSTRACT

The Campus recruitment management system (CRMS) is a comprehensive software designed to optimise and streamline the recruitment process for universities and companies. By integrating the management of student records, company profiles, job postings, applications, interviews and feedback, CRMS aims to facilitate efficient interactions between students, university placement cells and recruiting companies. The system is designed around key entities such as students, companies, job postings, applications, interviews, placement events and feedback, each with well defined attributes and relationships.

CRMS ensures that every student is registered and can apply for multiple job opportunities, while companies can manage job posting & interview schedules. Placement cells oversee recruitment activities, organise events, and generate placement reports using system's powerful data management and analysis features. The backend is powered by Oracle SQL, handling all database operations, while frontend, developed in Python using Tkinter or PyQt, provides intuitive interface for students, companies and administrators.

The system not only automates & organize campus recruitment process but also enhances decision-making with features like Average CGPA calculat-ions, student job matching & comprehensive placement reporting. CRMS is a robust and practical solution that addresses dynamic needs of campus recruitment ensuring transparent & efficient recruitment experience.

AIM:

The aim of this project is to design and implement a Campus Recruitment Management System using Python as frontend and OracleSQL as backend. The system aims to streamline the recruitment process for students, university placement cells, and companies by efficiently managing student records, company profiles, job postings, applications, interview schedule, placement events and feedback.

IDENTIFICATION OF COMPONENTS:

- Entities

→ Student

Manages student data including Personal details and academic information

→ Company

Stores information about companies participating in recruitment process.

→ Job Posting

Captures Job details posted by Companies.

→ Application. (Weak Entity)

Tracks applications made by Students to different Job Postings.

→ Interview Schedule (Weak Entity)

Manages interview dates and times for Student Applications.

→ Placement Cell

Represents the university's placement office managing the recruitment events.

→ Placement Event

Contains details of recruitment events organized by Placement Cell.

→ Feedback (Weak Entity)

Stores feedback given by students about Companies and placement events.

• Relationships

→ Student - Application : Student applies Applications.

→ Company - Job Posting : Company offers Job Postings.

→ Application - Interview Schedule : Application leads to Interview Schedule.

→ Placement_Cell - Placement_Event : Placement_Cell organizes Placement_Event.

→ Student - Interview Schedule : Student attends Interview.

→ Job_Posting - Interview_Schedule : Job organizes Interviews.

→ Student - Feedback : Student provides Feedback.

→ Company - Feedback : Company receives Feedback.

• Attributes

→ Student : Student_ID, Name, Email, Phone, Department, CGPA, Grad_Year.

→ Company : Company_ID, Company_Name, Industry_Type, Contact_Person, Contact_Email, Contact.

→ Job Posting : Job_ID, Position, Job_Description, Salary, Location, Company_ID.

→ Application : Application_ID, Application_Date, Status, Student_ID, Job_ID.

→ Interview_Schedule : Interview_ID, Interview_Date, Interview_time, Status, Student_ID, Job_ID.

→ Placement Cell : Cell_ID, Name, Department, Contact_Email, Contact.

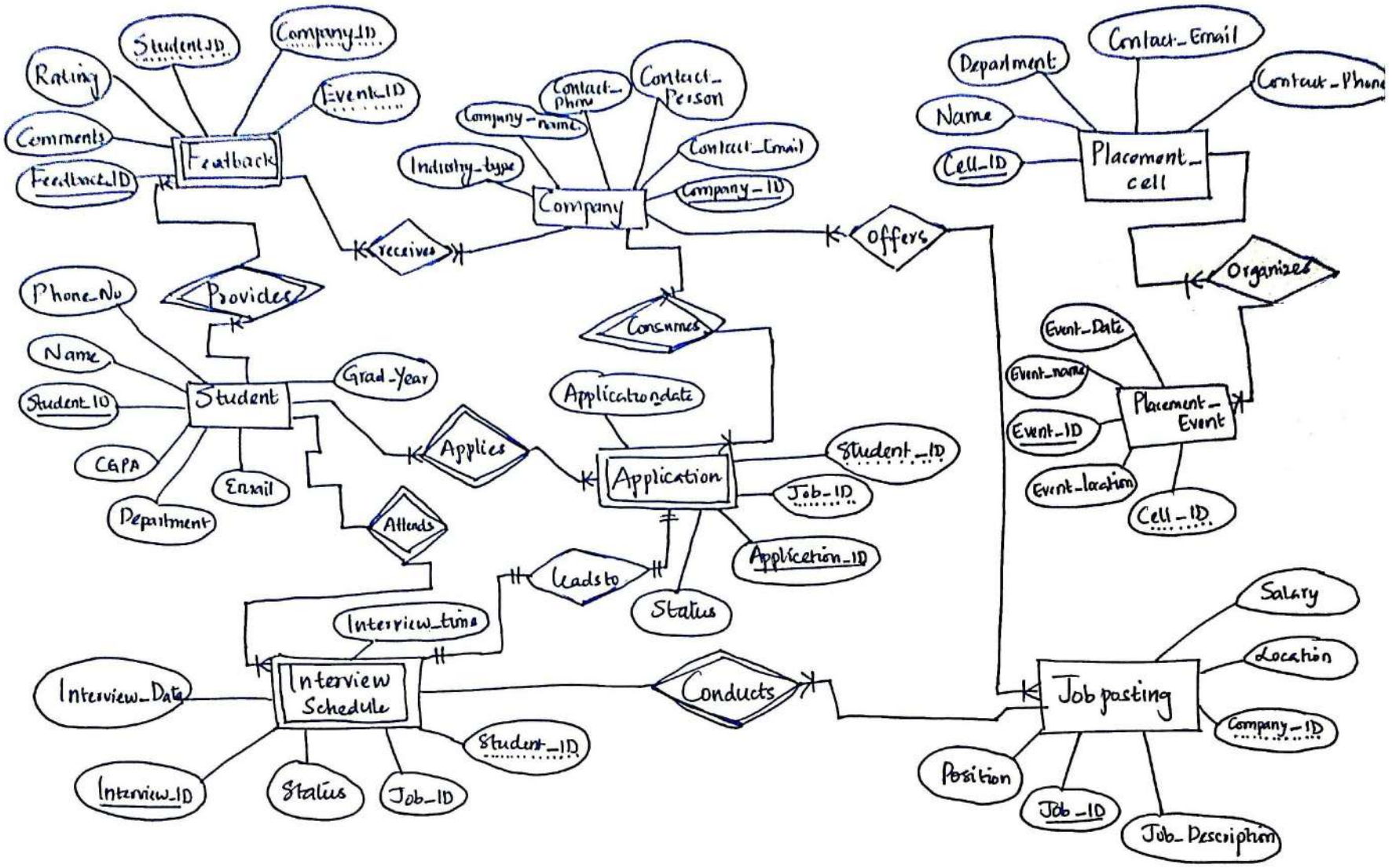
→ Placement Event : Event_ID, Event_Name, Event_Date, Event_Location, Cell_ID.

→ Feedback : Feedback_ID, Comments, Rating, Student_ID, Company_ID, Event_ID.

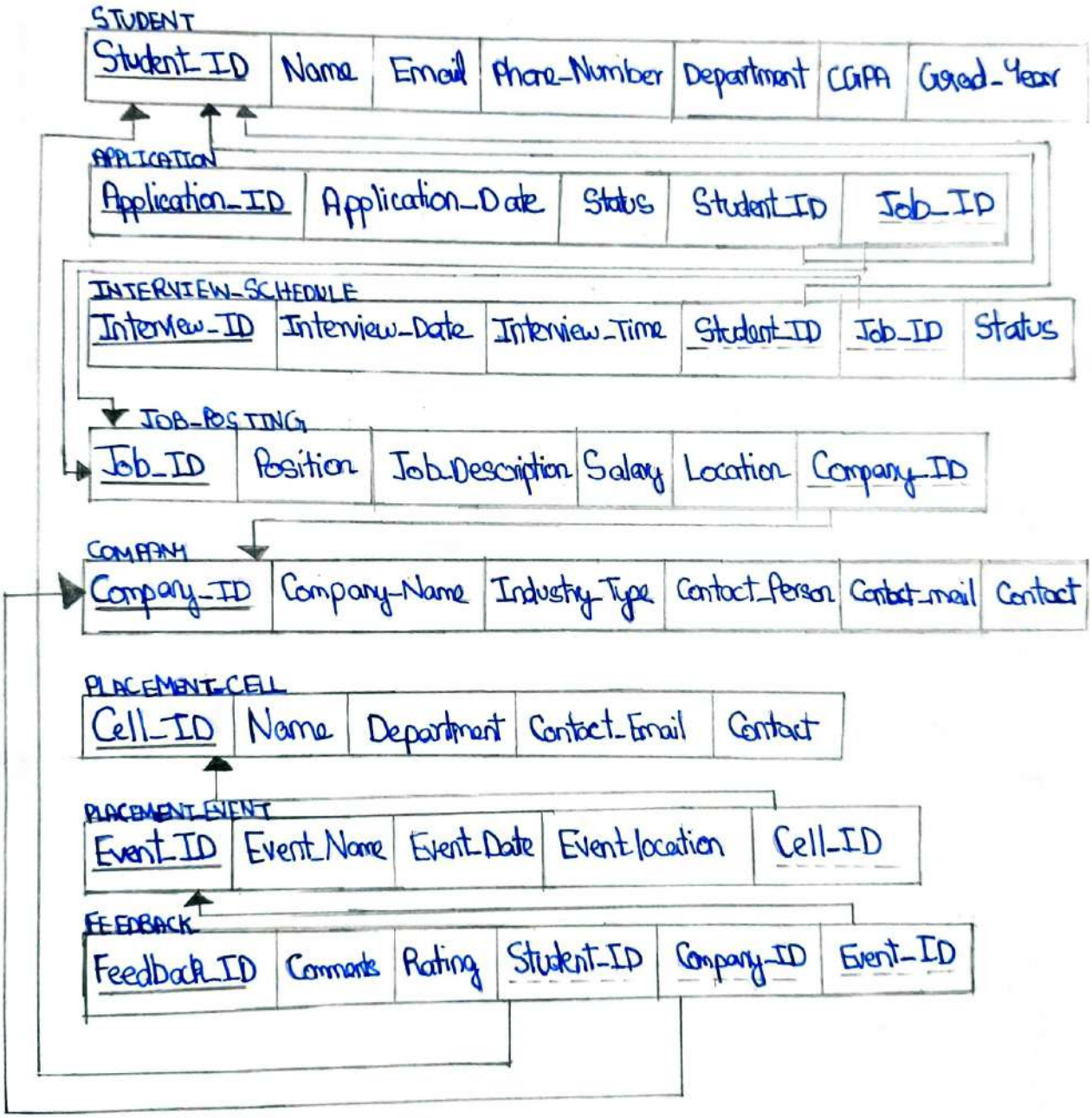
• Constraints denoted :-

→ Primary key - Underlined
→ Foreign Key - Dotted lined
→ Weak Entities - (mentioned)

ER DIAGRAM


Feedback
Rating
StudentID
CompanyID
EventID
Comments
FeedbackID
Company
Company_name
Contact_Phone
Contact_Person
Industry_type
Contact_Email
CompanyID
receives
Provides
Phone_No
Name
Student_ID
Grad_Year
Student
CGPA
Email
Department
Applies
Consumes
Application_date
Application
Student_ID
Job_ID
Application_ID
Status
Leads to
Attends
Interview_time
Interview Schedule
Interview_Date
Interview_ID
Status
Job_ID
Student_ID
Conducts
offers
Department
Name
Cell_ID
Contact_Email
Contact_Phone
Placement_cell
Organizes
Event_Date
Event_name
Event_ID
Event_location
Placement_Event
Cell_ID
Salary
Location
Company_ID
Job posting
Position
Job_ID
Job_Description

Mapping ER Model to Relational Schema



NORMALISATION :

Let us first examine the current tables of the database and then we will convert that into 3NF.

Current tables :

① Student table

- Student_ID
- Name
- Email
- Phone_Number
- department
- CGPA
- Graduation_Year
- Application_ID
- Job_ID
- Position
- Salary

This is not in 2NF because, the Application_ID is partially dependent on Student_ID and Job_ID.

(Redundant data and partial dependency)

② Job_posting table

- Position
- Salary
- company_Name
- Industry_type

Has partial dep. on company information

③ Application

- Application_ID
- Application_Date
- Student_ID
- Name (Redundant)
- Email
- Job_ID
- Position (redundant)
- Feedback (transitive)

Has redundancy with Student details.

④ Feedback

- Feedback_ID
- Comments
- Rating
- Student_ID
- Job_ID
- Company_Name

Has transitive dependency with company details

⑤ Interview_Schedule

- Interview_ID
- Interview_Date
- Interview_Time
- Student_ID
- Job_ID
- Company_Name

Company_Name has transitive dependency with company_ID

⑥ Placement_Event :

- Event_ID
- Event_Name
- Event_Date
- Event_location
- Company_Name

Company_Name has transitive dependency with Company_ID

Now, after reiviewing all tables we can say that all the tables are in INF.

Let us convert INF to 2NF by removing all the Partial dependencies.

① Student table (2NF)

- Student_ID (PK)
- Name
- Email
- Phone_Number
- department
- CGPA
- Graduation_Year

Partial dep. are removed, Application and Job details are moved to other table

② Job_Posting (2NF)

- Job_ID (PK)
- Position
- Salary
- Company_ID

③ Application Table (2NF)

- Application_ID (PK)
- Application_Date
- Student_ID (FK)
- Job_ID (FK)

Name, email removed.

④ Interview_Schedule (2NF)

- Interview_ID (PK)
- Interview_Date
- Interview_time
- Student_ID (FK)
- Job_ID (FK)

We removed Company Name.
Still transitive dependency is there

As still transitive dependency exist here, we will convert 2NF to 3NF by removing those.

① Student table
② Job_Posting table
③ Application table

Already in 3NF after 2NF conversion.
So, No change.

④ Interview - Schedule (Converted to 3NF)

- Interview - ID
- Interview - Date
- Interview - time
- Student - ID (FK)
- Job - ID (KK FK)

Now we no longer have trans. dep. on company ID as we removed Company - Name

⑤ Feedback Table (Now converted to 3NF)

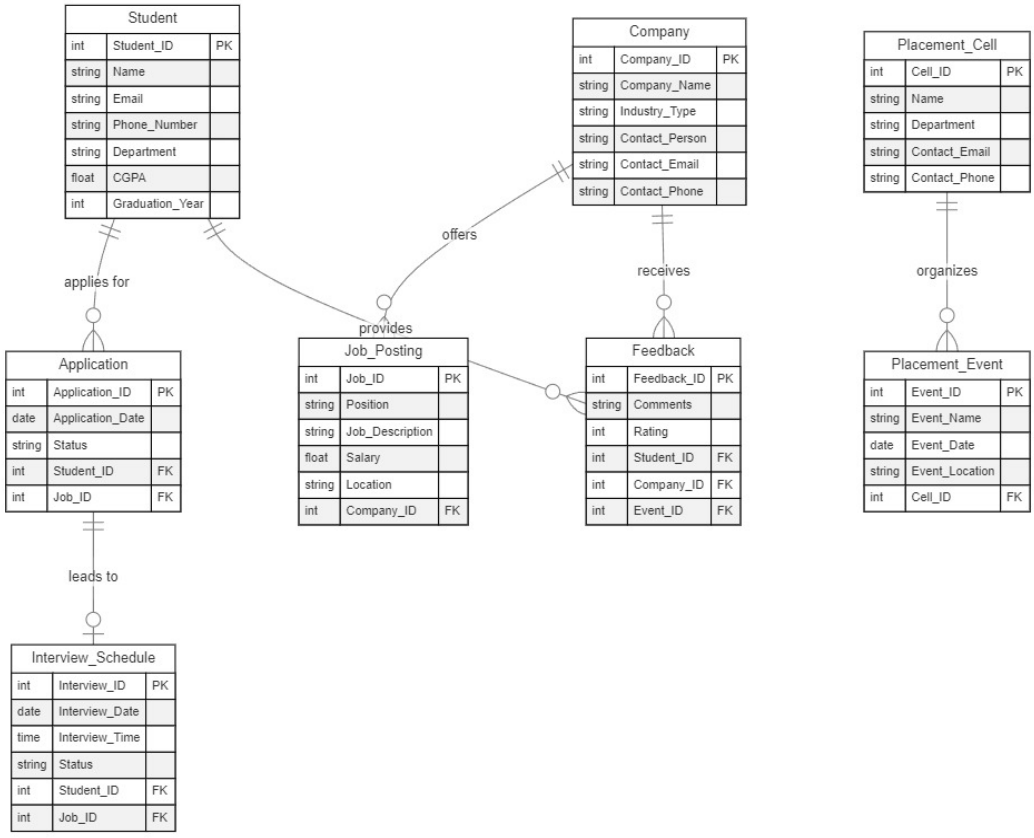
- Comments
- Rating
- Student - ID (FK)
- Job - ID (FK)
- Feedback - ID (PK)

⑥ Placement - Event (converted to 3NF)

- Event - ID (PK)
- Event - Name
- Event - Date
- Event - location
- Company - ID (FK)

* Final list of 3NF tables :

After all the steps, we end up with the follow list of tables :



Summary :

* 1NF to 2NF : 4 tables underwent conversion, removing partial dependency.

* 2NF to 3NF : 3 tables underwent conversion, removing transitive dependency.

CREATION OF TABLES :

```
CREATE TABLE Student (
    Student_ID NUMBER PRIMARY KEY,
    Name VARCHAR2(100),
    Email VARCHAR2(100),
    Phone_Number VARCHAR2(15),
    Department VARCHAR2(50),
    CGPA NUMBER(3,2),
    Graduation_Year NUMBER(4)
);

CREATE TABLE Company (
    Company_ID NUMBER PRIMARY KEY,
    Company_Name VARCHAR2(100),
    Industry_Type VARCHAR2(50),
    Contact_Person VARCHAR2(100),
    Contact_Email VARCHAR2(100),
    Contact_Phone VARCHAR2(15)
);

CREATE TABLE Job_Posting (
    Job_ID NUMBER PRIMARY KEY,
    Position VARCHAR2(100),
    Salary NUMBER,
    Company_ID NUMBER,
    FOREIGN KEY (Company_ID) REFERENCES
Company(Company_ID)
);

CREATE TABLE Application (
    Application_ID NUMBER PRIMARY KEY,
    Application_Date DATE,
    Student_ID NUMBER,
    Job_ID NUMBER,
    Status VARCHAR2(50),
    FOREIGN KEY (Student_ID) REFERENCES
Student(Student_ID),
    FOREIGN KEY (Job_ID) REFERENCES Job_Posting(Job_ID)
);
```

```
CREATE TABLE Interview_Schedule (
    Interview_ID NUMBER PRIMARY KEY,
    Interview_Date DATE,
    Interview_Time VARCHAR2(20),
    Student_ID NUMBER,
    Job_ID NUMBER,
    FOREIGN KEY (Student_ID) REFERENCES
Student(Student_ID),
    FOREIGN KEY (Job_ID) REFERENCES Job_Posting(Job_ID)
);


CREATE TABLE Feedback (
    Feedback_ID NUMBER PRIMARY KEY,
    Comments VARCHAR2(255),
    Rating NUMBER(1),
    Student_ID NUMBER,
    Company_ID NUMBER,
    FOREIGN KEY (Student_ID) REFERENCES
Student(Student_ID),
    FOREIGN KEY (Company_ID) REFERENCES
Company(Company_ID)
);

CREATE TABLE Placement_Event (
    Event_ID NUMBER PRIMARY KEY,
    Event_Name VARCHAR2(100),
    Event_Date DATE,
    Event_Location VARCHAR2(100),
    Company_ID NUMBER,
    FOREIGN KEY (Company_ID) REFERENCES
Company(Company_ID)
);

INSERT INTO Student VALUES (1, 'John Doe',
'johndoe@example.com', '1234567890', 'Computer Science',
3.8, 2024);
INSERT INTO Student VALUES (2, 'Jane Smith',
'janesmith@example.com', '1234567891', 'Mechanical
Engineering', 3.5, 2024);
```

```
INSERT INTO Student VALUES (3, 'Alice Johnson',
'alicej@example.com', '1234567892', 'Electrical
Engineering', 3.7, 2023);
INSERT INTO Student VALUES (4, 'Bob Brown',
'bobb@example.com', '1234567893', 'Civil Engineering',
3.6, 2025);
INSERT INTO Student VALUES (5, 'Charlie Green',
'charlieg@example.com', '1234567894', 'Chemical
Engineering', 3.9, 2024);
INSERT INTO Student VALUES (6, 'David White',
'davidw@example.com', '1234567895', 'Computer Science',
3.85, 2024);
INSERT INTO Student VALUES (7, 'Eva Black',
'evab@example.com', '1234567896', 'Information
Technology', 3.65, 2023);

INSERT INTO Company VALUES (1, 'Tech Innovators', 'IT',
'Mike Brown', 'mikebrown@techinnovators.com',
'9876543210');
INSERT INTO Company VALUES (2, 'Green Solutions',
'Energy', 'Sarah Green', 'sarahgreen@greensolutions.com',
'9876543211');
INSERT INTO Company VALUES (3, 'AutoMakers',
'Automotive', 'David Black', 'davidblack@automakers.com',
'9876543212');
INSERT INTO Company VALUES (4, 'Skyline Constructions',
'Construction', 'Paul White',
'paulwhite@skylineconstructions.com', '9876543213');
INSERT INTO Company VALUES (5, 'BioHealth', 'Healthcare',
'Helen Blue', 'helenblue@biohealth.com', '9876543214');
INSERT INTO Company VALUES (6, 'AgroGrow', 'Agriculture',
'Linda Red', 'lindared@agrogrow.com', '9876543215');
INSERT INTO Company VALUES (7, 'FintechPioneers',
'Finance', 'James Yellow',
'jamesyellow@fintechpioneers.com', '9876543216');

INSERT INTO Job_Posting VALUES (1, 'Software Developer',
60000, 1);
INSERT INTO Job_Posting VALUES (2, 'Energy Analyst',
50000, 2);
```

```
INSERT INTO Job_Posting VALUES (3, 'Mechanical Engineer',
55000, 3);
INSERT INTO Job_Posting VALUES (4, 'Civil Engineer',
52000, 4);
INSERT INTO Job_Posting VALUES (5, 'Healthcare Analyst',
58000, 5);
INSERT INTO Job_Posting VALUES (6, 'Agricultural
Scientist', 53000, 6);
INSERT INTO Job_Posting VALUES (7, 'Financial Analyst',
62000, 7);

INSERT INTO Application VALUES (1, TO_DATE('2024-01-10',
'YYYY-MM-DD'), 1, 1, 'Under Review');
INSERT INTO Application VALUES (2, TO_DATE('2024-01-12',
'YYYY-MM-DD'), 2, 2, 'Interview Scheduled');
INSERT INTO Application VALUES (3, TO_DATE('2024-01-14',
'YYYY-MM-DD'), 3, 3, 'Under Review');
INSERT INTO Application VALUES (4, TO_DATE('2024-01-16',
'YYYY-MM-DD'), 4, 4, 'Rejected');
INSERT INTO Application VALUES (5, TO_DATE('2024-01-18',
'YYYY-MM-DD'), 5, 5, 'Accepted');
INSERT INTO Application VALUES (6, TO_DATE('2024-01-20',
'YYYY-MM-DD'), 6, 6, 'Under Review');
INSERT INTO Application VALUES (7, TO_DATE('2024-01-22',
'YYYY-MM-DD'), 7, 7, 'Interview Scheduled');

INSERT INTO Interview_Schedule VALUES (1, TO_DATE('2024-
02-01', 'YYYY-MM-DD'), '10:00 AM', 1, 1);
INSERT INTO Interview_Schedule VALUES (2, TO_DATE('2024-
02-02', 'YYYY-MM-DD'), '11:00 AM', 2, 2);
INSERT INTO Interview_Schedule VALUES (3, TO_DATE('2024-
02-03', 'YYYY-MM-DD'), '12:00 PM', 3, 3);
INSERT INTO Interview_Schedule VALUES (4, TO_DATE('2024-
02-04', 'YYYY-MM-DD'), '1:00 PM', 4, 4);
INSERT INTO Interview_Schedule VALUES (5, TO_DATE('2024-
02-05', 'YYYY-MM-DD'), '2:00 PM', 5, 5);
INSERT INTO Interview_Schedule VALUES (6, TO_DATE('2024-
02-06', 'YYYY-MM-DD'), '3:00 PM', 6, 6);
INSERT INTO Interview_Schedule VALUES (7, TO_DATE('2024-
02-07', 'YYYY-MM-DD'), '4:00 PM', 7, 7);
```

```
INSERT INTO Feedback VALUES (1, 'Great potential', 5, 1,
1);
INSERT INTO Feedback VALUES (2, 'Needs improvement', 3,
2, 2);
INSERT INTO Feedback VALUES (3, 'Strong technical
skills', 4, 3, 3);
INSERT INTO Feedback VALUES (4, 'Good communication', 5,
4, 4);
INSERT INTO Feedback VALUES (5, 'Excellent fit', 5, 5,
5);
INSERT INTO Feedback VALUES (6, 'Average performance', 3,
6, 6);
INSERT INTO Feedback VALUES (7, 'Very promising', 4, 7,
7);

INSERT INTO Placement_Event VALUES (1, 'Tech Job Fair',
TO_DATE('2024-03-10', 'YYYY-MM-DD'), 'Tech Park', 1);
INSERT INTO Placement_Event VALUES (2, 'Green Energy
Conference', TO_DATE('2024-04-12', 'YYYY-MM-DD'), 'Green
Hall', 2);
INSERT INTO Placement_Event VALUES (3, 'Automotive Career
Expo', TO_DATE('2024-05-14', 'YYYY-MM-DD'), 'Auto
Convention Center', 3);
INSERT INTO Placement_Event VALUES (4, 'Construction
Expo', TO_DATE('2024-06-16', 'YYYY-MM-DD'), 'Skyline
Plaza', 4);

INSERT INTO Placement_Event VALUES (5, 'Healthcare Career
Summit', TO_DATE('2024-07-18', 'YYYY-MM-DD'), 'BioHealth
Arena', 5);

INSERT INTO Placement_Event VALUES (6, 'Agriculture
Conference', TO_DATE('2024-08-20', 'YYYY-MM-DD'),
'AgroGrow Field', 6);

INSERT INTO Placement_Event VALUES (7, 'Fintech Career
Fest', TO_DATE('2024-09-22', 'YYYY-MM-DD'), 'Fintech
Center', 7);
```

```
SELECT * FROM Student;
SELECT * FROM Company;
SELECT * FROM Job_Posting;
SELECT * FROM Application;
SELECT * FROM Interview_Schedule;
SELECT * FROM Feedback;
SELECT * FROM Placement_Event;
```

SCREENSHOTS :

```
SQL> select * from Student;

STUDENT_ID NAME            EMAIL                PHONE_NUMBER DEPARTMENT                CGPA GRADUATION_YEAR
---------- --------------- -------------------- ------------ ------------------------- ---- ---------------
         1 John Doe        johndoe@example.com  1234567890   Computer Science          3.8             2024
         2 Jane Smith      janesmith@example.com 1234567891  Mechanical Engineering    3.5             2024
         3 Alice Johnson   alicej@example.com   1234567892   Electrical Engineering    3.7             2023
         4 Bob Brown       bobb@example.com     1234567893   Civil Engineering         3.6             2025
         5 Charlie Green   charlieg@example.com 1234567894   Chemical Engineering      3.9             2024
         6 David White     davidw@example.com   1234567895   Computer Science         3.85             2024
         7 Eva Black       evab@example.com     1234567896   Information Technology    3.65             2023

7 rows selected.
```

```
SQL> select * from Company;

COMPANY_ID COMPANY_NAME          INDUSTRY_TYPE   CONTACT_PERSON CONTACT_EMAIL                    CONTACT_PHONE
---------- --------------------- --------------- -------------- ------------------------------- -------------
         1 Tech Innovators       IT              Mike Brown     mikebrown@techinnovators.com    9876543210
         2 Green Solutions       Energy          Sarah Green    sarahgreen@greensolutions.com   9876543211
         3 AutoMakers            Automotive      David Black    davidblack@automakers.com       9876543212
         4 Skyline Constructions Construction    Paul White     paulwhite@skylineconstructions.com 9876543213
         5 BioHealth             Healthcare      Helen Blue     helenblue@biohealth.com         9876543214
         6 AgroGrow              Agriculture     Linda Red      lindared@agrogrow.com           9876543215
         7 FintechPioneers       Finance         James Yellow   jamesyellow@fintechpioneers.com 9876543216

7 rows selected.
```

```
SQL> select * from Job_posting;

    JOB_ID POSITION                       SALARY     COMPANY_ID
---------- ------------------------------ ---------- ----------
         1 Software Developer             60000               1
         2 Energy Analyst                 50000               2
         3 Mechanical Engineer            55000               3
         4 Civil Engineer                 52000               4
         5 Healthcare Analyst             58000               5
         6 Agricultural Scientist         53000               6
         7 Financial Analyst              62000               7

7 rows selected.

SQL>
```

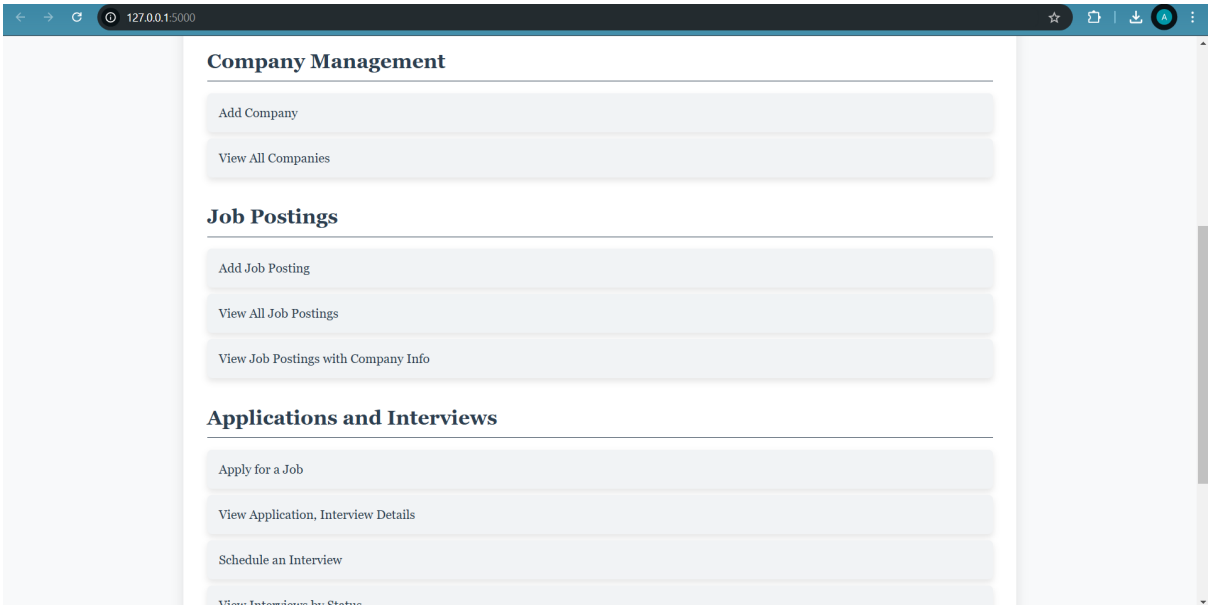
```
SQL> select * from Application;

   APPLICATION_ID APPLICATION_DATE       STUDENT_ID    JOB_ID STATUS
_________________ _____________________ ___________ _________ ____________________
                1 10-JAN-24                       1         1 Under Review
                2 12-JAN-24                       2         2 Interview Scheduled
                3 14-JAN-24                       3         3 Under Review
                4 16-JAN-24                       4         4 Rejected
                5 18-JAN-24                       5         5 Accepted
                6 20-JAN-24                       6         6 Under Review
                7 22-JAN-24                       7         7 Interview Scheduled

7 rows selected.
```

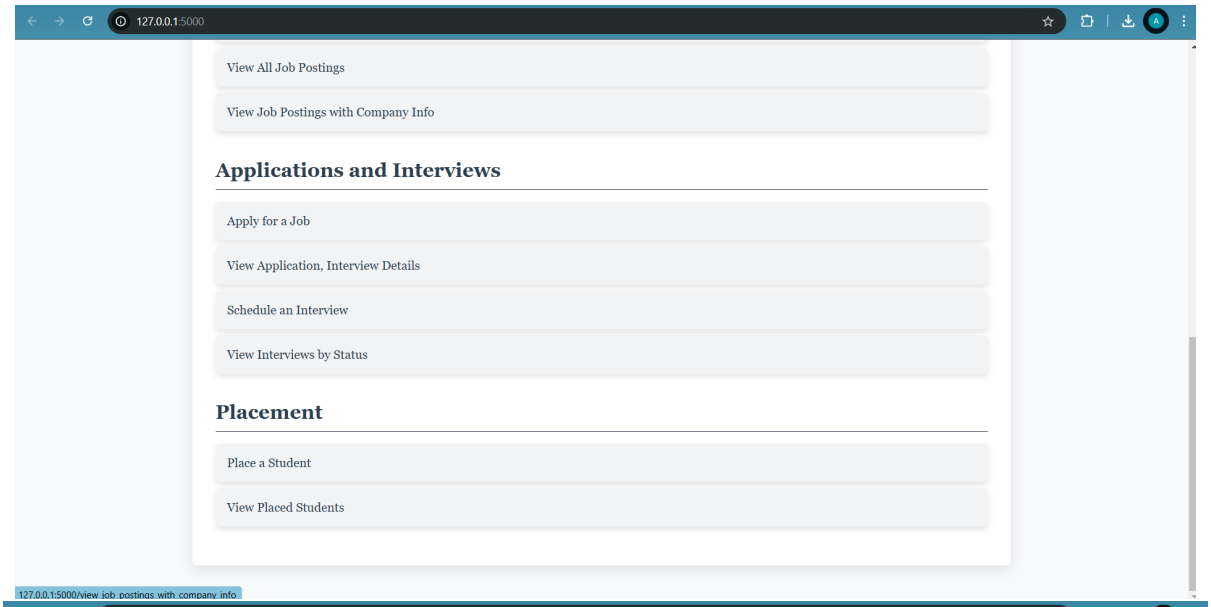
```
SQL> select * from interview_schedule;

   INTERVIEW_ID INTERVIEW_DATE    INTERVIEW_TIME       STUDENT_ID    JOB_ID
________________ _____________________ _____________________ _____________ _________
               1 01-FEB-24         10:00 AM                  1         1
               2 02-FEB-24         11:00 AM                  2         2
               3 03-FEB-24         12:00 PM                  3         3
               4 04-FEB-24         1:00 PM                   4         4
               5 05-FEB-24         2:00 PM                   5         5
               6 06-FEB-24         3:00 PM                   6         6
               7 07-FEB-24         4:00 PM                   7         7

7 rows selected.
```

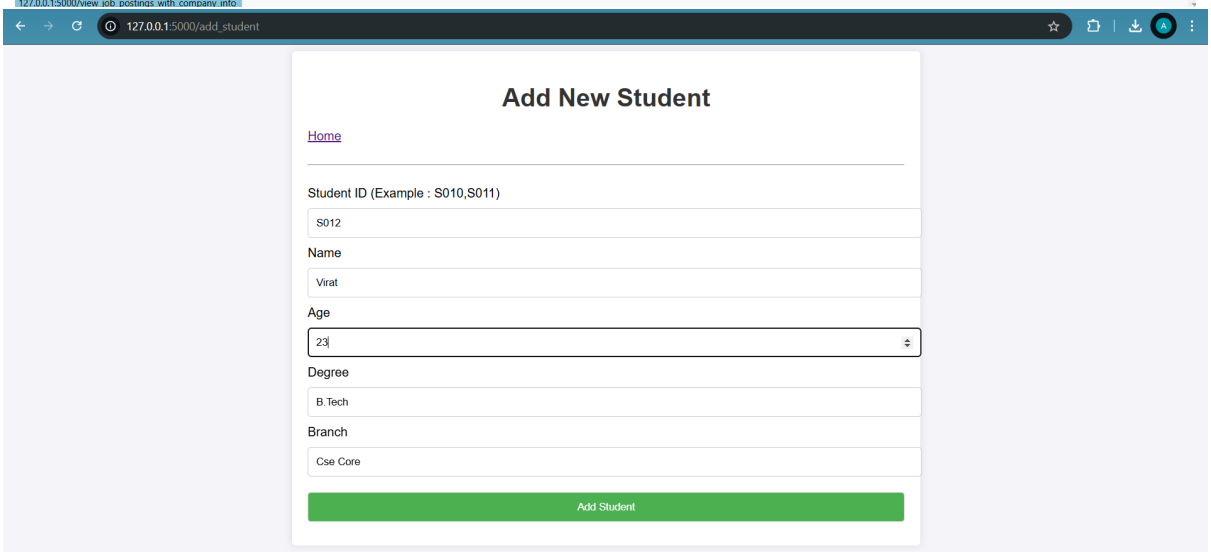
```
SQL> select * from feedback;

   FEEDBACK_ID COMMENTS                       RATING    STUDENT_ID    COMPANY_ID
______________ ____________________________ ________ _____________ _____________
             1 Great potential                     5             1             1
             2 Needs improvement                   3             2             2
             3 Strong technical skills             4             3             3
             4 Good communication                  5             4             4
             5 Excellent fit                       5             5             5
             6 Average performance                 3             6             6
             7 Very promising                      4             7             7

7 rows selected.
```

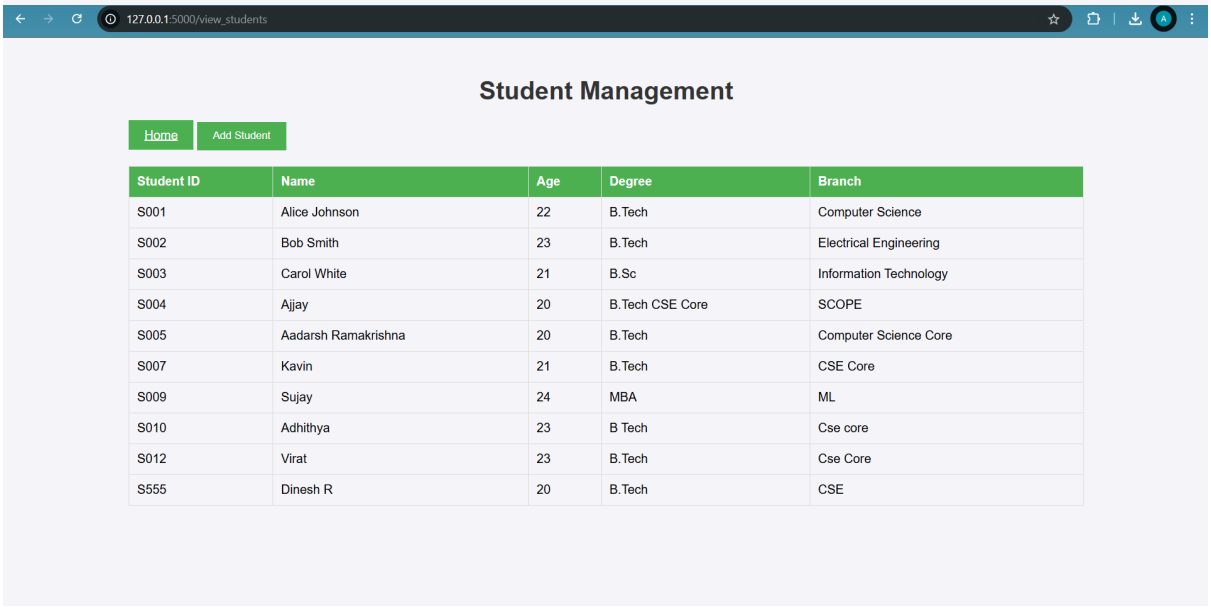
```
SQL> select * from placement_event;

   EVENT_ID EVENT_NAME                 EVENT_DATE    EVENT_LOCATION              COMPANY_ID
___________ _____________________ _____________ _____________________ _____________
          1 Tech Job Fair         10-MAR-24     Tech Park                         1
          2 Green Energy Conference 12-APR-24     Green Hall                        2
          3 Automotive Career Expo  14-MAY-24     Auto Convention Center            3
          4 Construction Expo       16-JUN-24     Skyline Plaza                     4
          5 Healthcare Career Summit 18-JUL-24     BioHealth Arena                   5
          6 Agriculture Conference  20-AUG-24     AgroGrow Field                    6
          7 Fintech Career Fest     22-SEP-24     Fintech Center                    7

7 rows selected.
```

SCREENSHOTS OF THE WEBPAGE :









Student ID	Name	Age	Degree	Branch
S001	Alice Johnson	22	B.Tech	Computer Science
S002	Bob Smith	23	B.Tech	Electrical Engineering
S003	Carol White	21	B.Sc	Information Technology
S004	Ajjay	20	B.Tech CSE Core	SCOPE
S005	Aadarsh Ramakrishna	20	B.Tech	Computer Science Core
S007	Kavin	21	B.Tech	CSE Core
S009	Sujay	24	MBA	ML
S010	Adhithya	23	B Tech	Cse core
S012	Virat	23	B.Tech	Cse Core
S555	Dinesh R	20	B.Tech	CSE

Students with Applications

Home

Student ID	Name	Applications Count
S001	Alice Johnson	1
S002	Bob Smith	2
S003	Carol White	1
S007	Kavin	1
S010	Adhithya	5
S555	Dinesh R	2

Students Without Applications

Home

Student ID	Name	Age	Branch
S004	Ajjay	20	SCOPE
S005	Aadarsh Ramakrishna	20	Computer Science Core
S009	Sujay	24	ML
S012	Virat	23	Cse Core

Add New Company

Home

Company ID (EG : C010,C011)

C013

Company Name

City Bank

Location

Nevada

Sector

Finance

Add Company

127.0.0.1:5000/view_companies

Companies List

Home | Add New Company

Company ID	Company Name	Location	Sector
C001	TechCorp	New York	IT
C002	FinServe	London	Finance
C003	HealthPlus	San Francisco	Healthcare
C004	SuTech	California	IT
C005	Mclaren	Berlin,Germany	Automobile
C009	Accenture	Perungalathur	IT
C010	Google	Nevada	IT
C012	CAMS	Amsterdam	IT
C013	City Bank	Nevada	Finance

127.0.0.1:5000/add_job_posting

Add New Job Posting

Home View All Companies

Job ID (EG: J010,J011)

J012

Job Title

Web Developer

Company ID (EG:C010,C011)

C012

Job Description

Develop and monitor company websites efficiently.

Salary

75000

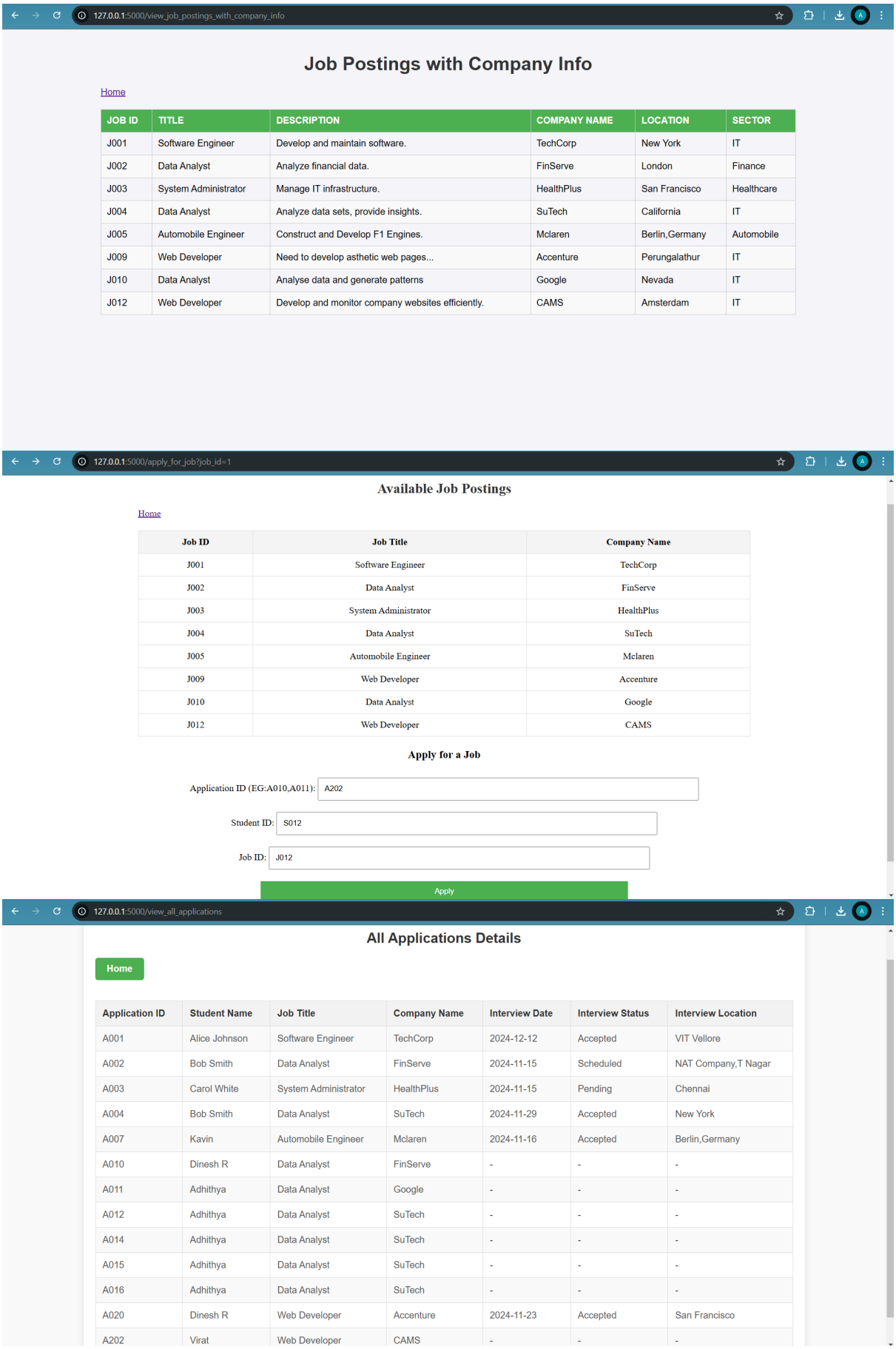
Add Job Posting

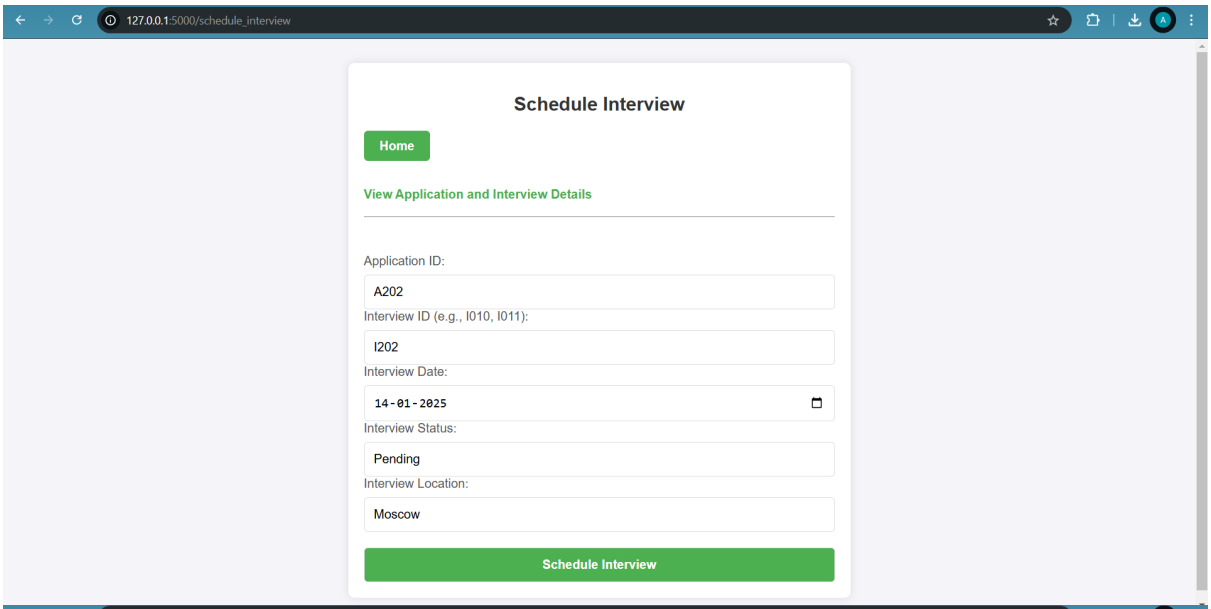
127.0.0.1:5000/view_job_postings

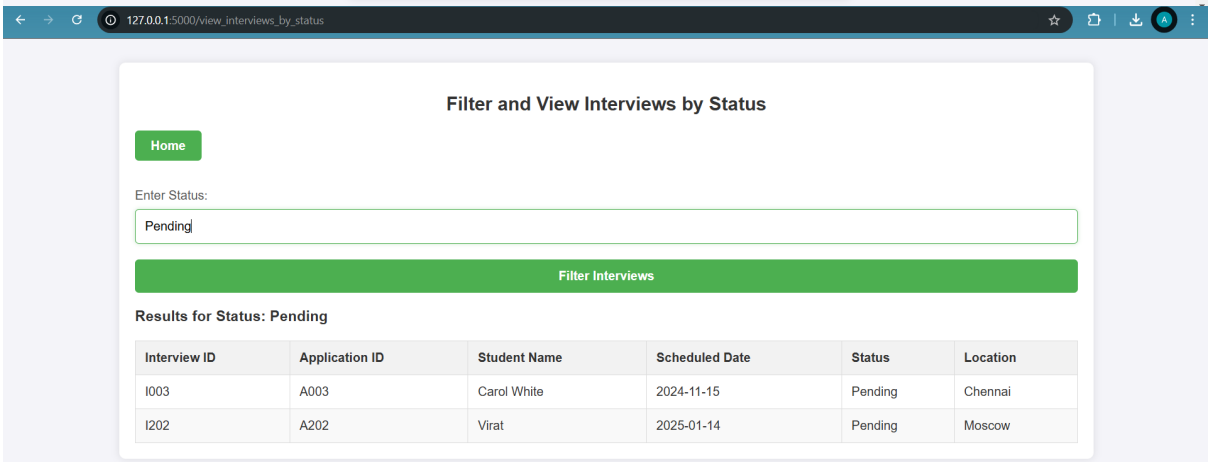
Job Postings

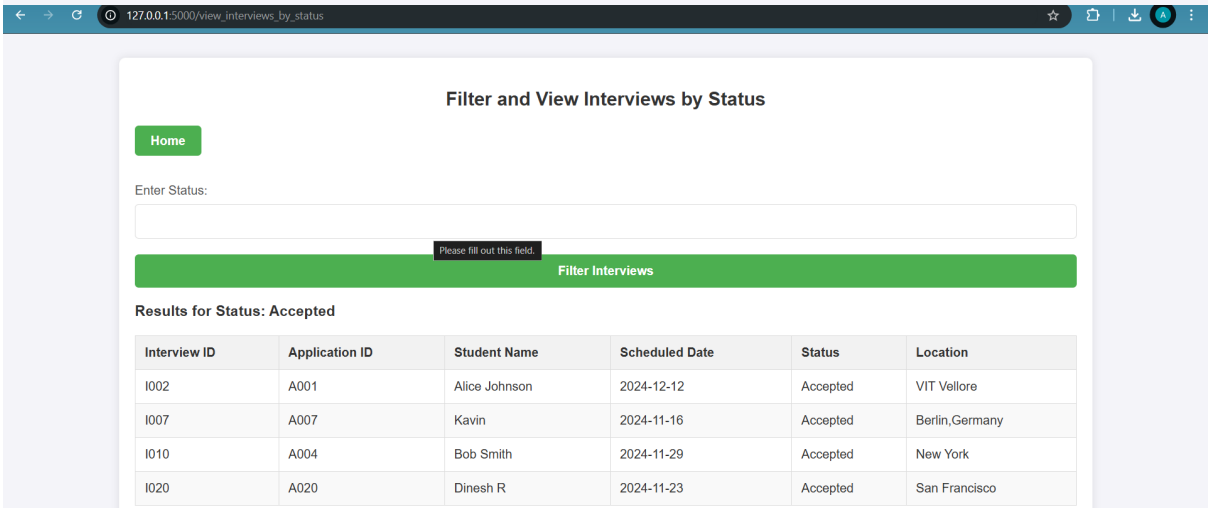
Home | Add New Job Posting

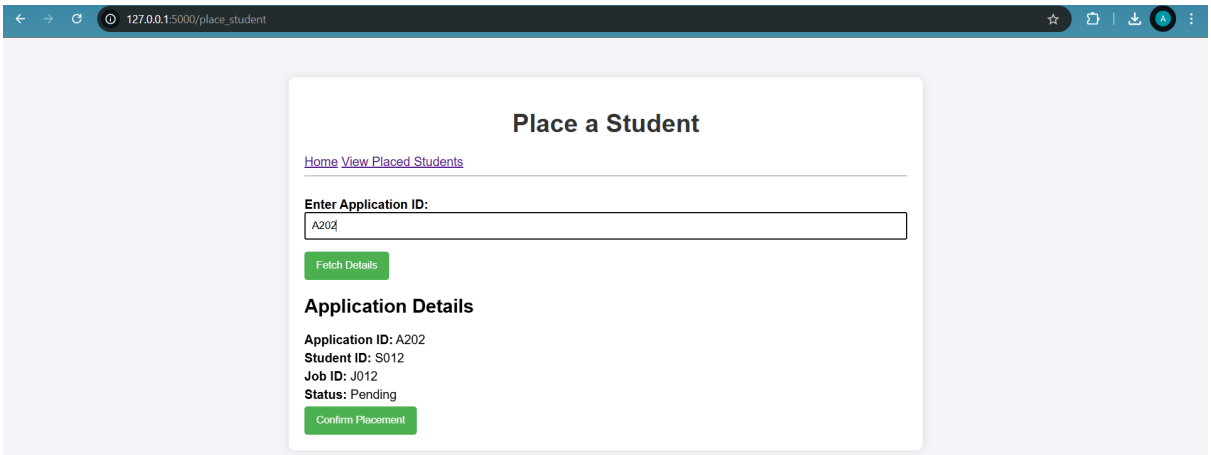
Job ID	Title	Description	Salary	Company ID
J001	Software Engineer	Develop and maintain software.	75000.00	C001
J002	Data Analyst	Analyze financial data.	65000.00	C002
J003	System Administrator	Manage IT infrastructure.	70000.00	C003
J004	Data Analyst	Analyze data sets, provide insights.	50000.00	C004
J005	Automobile Engineer	Construct and Develop F1 Engines.	90000.00	C005
J009	Web Developer	Need to develop asthetic web pages...	12000.00	C009
J010	Data Analyst	Analyse data and generate patterns	50000.00	C010
J012	Web Developer	Develop and monitor company websites efficiently.	75000.00	C012

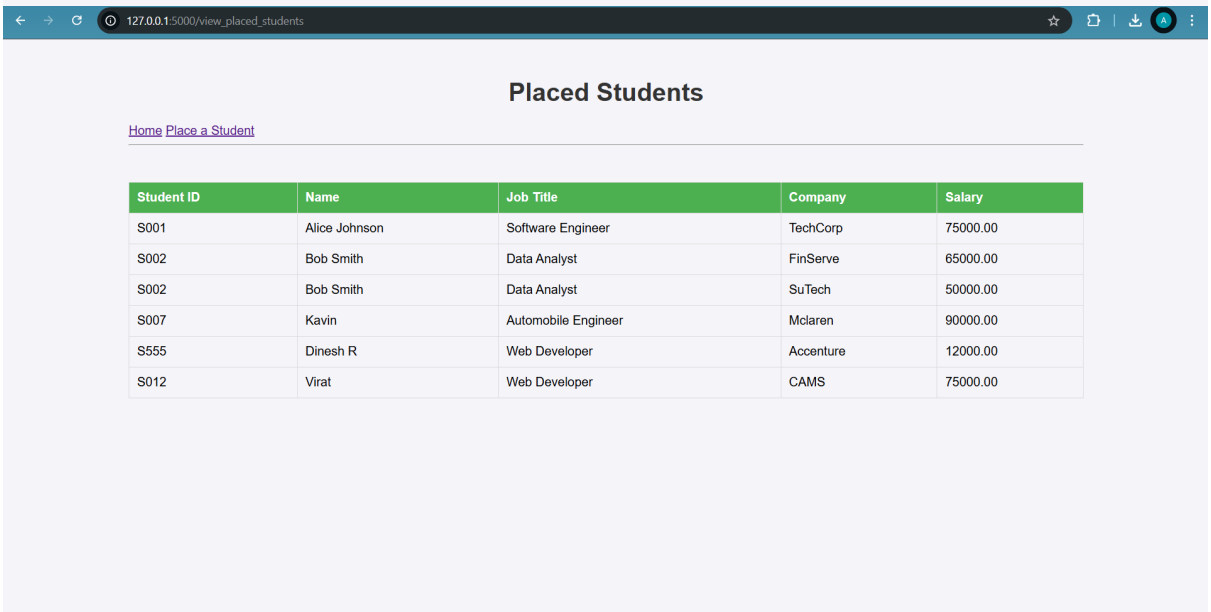

127.0.0.1:5000/view_job_postings_with_company_info
Job Postings with Company Info
Home
JOB ID | TITLE | DESCRIPTION | COMPANY NAME | LOCATION | SECTOR
J001 | Software Engineer | Develop and maintain software. | TechCorp | New York | IT
J002 | Data Analyst | Analyze financial data. | FinServe | London | Finance
J003 | System Administrator | Manage IT infrastructure. | HealthPlus | San Francisco | Healthcare
J004 | Data Analyst | Analyze data sets, provide insights. | SuTech | California | IT
J005 | Automobile Engineer | Construct and Develop F1 Engines. | Mclaren | Berlin,Germany | Automobile
J009 | Web Developer | Need to develop asthetic web pages... | Accenture | Perungalathur | IT
J010 | Data Analyst | Analyse data and generate patterns | Google | Nevada | IT
J012 | Web Developer | Develop and monitor company websites efficiently. | CAMS | Amsterdam | IT
127.0.0.1:5000/apply_for_job?job_id=1
Available Job Postings
Home
Job ID | Job Title | Company Name
J001 | Software Engineer | TechCorp
J002 | Data Analyst | FinServe
J003 | System Administrator | HealthPlus
J004 | Data Analyst | SuTech
J005 | Automobile Engineer | Mclaren
J009 | Web Developer | Accenture
J010 | Data Analyst | Google
J012 | Web Developer | CAMS
Apply for a Job
Application ID (EG:A010,A011): A202
Student ID: S012
Job ID: J012
Apply
127.0.0.1:5000/view_all_applications
All Applications Details
Home
Application ID | Student Name | Job Title | Company Name | Interview Date | Interview Status | Interview Location
A001 | Alice Johnson | Software Engineer | TechCorp | 2024-12-12 | Accepted | VIT Vellore
A002 | Bob Smith | Data Analyst | FinServe | 2024-11-15 | Scheduled | NAT Company,T Nagar
A003 | Carol White | System Administrator | HealthPlus | 2024-11-15 | Pending | Chennai
A004 | Bob Smith | Data Analyst | SuTech | 2024-11-29 | Accepted | New York
A007 | Kavin | Automobile Engineer | Mclaren | 2024-11-16 | Accepted | Berlin,Germany
A010 | Dinesh R | Data Analyst | FinServe | - | - | -
A011 | Adhithya | Data Analyst | Google | - | - | -
A012 | Adhithya | Data Analyst | SuTech | - | - | -
A014 | Adhithya | Data Analyst | SuTech | - | - | -
A015 | Adhithya | Data Analyst | SuTech | - | - | -
A016 | Adhithya | Data Analyst | SuTech | - | - | -
A020 | Dinesh R | Web Developer | Accenture | 2024-11-23 | Accepted | San Francisco
A202 | Virat | Web Developer | CAMS | - | - | -


127.0.0.1:5000/schedule_interview
Schedule Interview
Home
View Application and Interview Details
Application ID:
A202
Interview ID (e.g., I010, I011):
I202
Interview Date:
14-01-2025
Interview Status:
Pending
Interview Location:
Moscow
Schedule Interview


127.0.0.1:5000/view_interviews_by_status
Filter and View Interviews by Status
Home
Enter Status:
Pending
Filter Interviews
Results for Status: Pending
Interview ID | Application ID | Student Name | Scheduled Date | Status | Location
I003 | A003 | Carol White | 2024-11-15 | Pending | Chennai
I202 | A202 | Virat | 2025-01-14 | Pending | Moscow


127.0.0.1:5000/view_interviews_by_status
Filter and View Interviews by Status
Home
Enter Status:
Please fill out this field.
Filter Interviews
Results for Status: Accepted
Interview ID | Application ID | Student Name | Scheduled Date | Status | Location
I002 | A001 | Alice Johnson | 2024-12-12 | Accepted | VIT Vellore
I007 | A007 | Kavin | 2024-11-16 | Accepted | Berlin,Germany
I010 | A004 | Bob Smith | 2024-11-29 | Accepted | New York
I020 | A020 | Dinesh R | 2024-11-23 | Accepted | San Francisco





Student ID	Name	Job Title	Company	Salary
S001	Alice Johnson	Software Engineer	TechCorp	75000.00
S002	Bob Smith	Data Analyst	FinServe	65000.00
S002	Bob Smith	Data Analyst	SuTech	50000.00
S007	Kavin	Automobile Engineer	Mclaren	90000.00
S555	Dinesh R	Web Developer	Accenture	12000.00
S012	Virat	Web Developer	CAMS	75000.00

CODE:

1. app.py

```
from flask import Flask, render_template, request, redirect,
url_for
import db_functions

app = Flask(__name__)

# Home Route
@app.route('/')
def home():
    return render_template('home.html')

# Student Routes
@app.route('/add_student', methods=['GET', 'POST'])
def add_student():
    if request.method == 'POST':
        student_id = request.form['student_id']
        name = request.form['name']
        age = request.form['age']
        degree = request.form['degree']
        branch = request.form['branch']
        db_functions.insert_student(student_id, name, age,
degree, branch)
        return redirect(url_for('view_students'))
    return render_template('add_student.html')

@app.route('/view_students')
def view_students():
    students = db_functions.view_students()
    print(students)  # Debug print
    return render_template('students.html', students=students)

@app.route('/students_with_applications')
def students_with_applications():
    students = db_functions.view_students_with_applications()
    return render_template('students_with_applications.html',
students=students)

@app.route('/students_without_applications')
def students_without_applications():
    students =
db_functions.view_students_without_applications()
```

```
    return
render_template('students_without_applications.html',
students=students)

@app.route('/view_job_application_statistics')
def view_job_application_statistics():
    stats = db_functions.view_jobs_applied_per_student()
    return render_template('job_application_statistics.html',
stats=stats)

@app.route('/view_placed_students')
def view_placed_students():
    students = db_functions.view_placed_students()
    return render_template('placed_students.html',
students=students)

# Company Routes
@app.route('/add_company', methods=['GET', 'POST'])
def add_company():
    if request.method == 'POST':
        company_id = request.form['company_id']
        name = request.form['name']
        location = request.form['location']
        sector = request.form['sector']
        db_functions.insert_company(company_id, name,
location, sector)
        return redirect(url_for('view_companies'))
    return render_template('add_company.html')

@app.route('/view_companies')
def view_companies():
    companies = db_functions.view_companies()
    return render_template('companies.html',
companies=companies)

# Job Posting Routes
@app.route('/add_job_posting', methods=['GET', 'POST'])
def add_job_posting():
    if request.method == 'POST':
        job_id = request.form['job_id']
        title = request.form['title']
        company_id = request.form['company_id']
        description = request.form['description']
        salary=request.form['salary']
        db_functions.insert_job_posting(job_id, title,
company_id, description,salary)
        return redirect(url_for('view_job_postings'))
    return render_template('add_job_posting.html')

@app.route('/view_job_postings')
def view_job_postings():
```

```
    job_postings = db_functions.view_job_postings()
    return render_template('job_postings.html',
job_postings=job_postings)

@app.route('/view_job_postings_with_company_info')
def view_job_postings_with_company_info():
    job_postings =
db_functions.view_job_postings_with_company_info()
    return
render_template('job_postings_with_company_info.html',
job_postings=job_postings)

# Application Routes
'''@app.route('/apply_for_job/<int:job_id>', methods=['GET',
'POST'])
def apply_for_job(job_id):
    if request.method == 'POST':
        student_id = request.form['student_id']
        db_functions.apply_for_job(student_id, job_id)
        return redirect(url_for('view_job_postings'))
    return render_template('apply_for_job.html',
job_id=job_id)'''

@app.route('/apply_for_job', methods=['GET', 'POST'])
def apply_for_job():
    if request.method == 'POST':
        application_id = request.form['application_id']  # Get
APPLICATION_ID from the form
        student_id = request.form['student_id']
        job_id = request.form['job_id']
        db_functions.apply_for_job(application_id, student_id,
job_id)  # Pass all three to the function
        return redirect(url_for('view_all_applications'))

    # Retrieve all available jobs with company name to display
in the table
    job_postings =
db_functions.view_job_postings_with_company_info()
    return render_template('apply_for_job.html',
job_postings=job_postings)

@app.route('/view_all_applications')
def view_all_applications():
    applications = db_functions.view_all_applications()
    return render_template('application_details.html',
applications=applications)

@app.route('/view_application_details/<int:application_id>')
def view_application_details(application_id):
    application_details =
db_functions.view_application_details(application_id)
```

```
    return render_template('application_details.html',
application_details=application_details)

@app.route('/schedule_interview', methods=['GET', 'POST'])
def schedule_interview():
    if request.method == 'POST':
        # Get form data
        application_id = request.form['application_id']
        interview_id = request.form['interview_id']
        scheduled_date = request.form['scheduled_date']
        status = request.form['status']
        location = request.form['location']

        # Schedule interview in the database
        db_functions.schedule_interview(application_id,
interview_id, scheduled_date, status, location)

        # Redirect to the home page after scheduling
        return redirect(url_for('home'))
    return render_template('schedule_interview.html')

@app.route('/view_applications_by_status/<status>')
def view_applications_by_status(status):
    applications =
db_functions.view_applications_by_status(status)
    return render_template('applications_by_status.html',
applications=applications)

@app.route('/view_interviews_by_status', methods=['GET',
'POST'])
def view_interviews_by_status():
    interviews = []
    status = None

    if request.method == 'POST':
        # Get the status from the form
        status = request.form['status']

        # Fetch interviews with the specified status
        interviews =
db_functions.get_interviews_by_status(status)

    # Render the page with the form and the results if any
    return render_template('view_interviews_by_status.html',
interviews=interviews, status=status)

@app.route('/view_scheduled_applications_and_job_postings')
def view_scheduled_applications_and_job_postings():
    scheduled_apps =
db_functions.view_scheduled_applications_and_job_postings()
```

```
    return
render_template('scheduled_applications_and_job_postings.html'
, scheduled_apps=scheduled_apps)

# Specialized View Routes
@app.route('/view_specific_student_details/<student_id>')
def view_specific_student_details(student_id):
    student_details =
db_functions.view_specific_student_details(student_id)
    return render_template('student_profile.html',
student_details=student_details)

@app.route('/place_student', methods=['GET', 'POST'])
def place_student():
    if request.method == 'POST':
        application_id = request.form['application_id']

        # Fetch application details from the database
        application =
db_functions.get_application_details(application_id)

        if application:
            return render_template('place_student.html',
application=application)
        else:
            return render_template('place_student.html',
error="Application ID not found.")

    return render_template('place_student.html')

@app.route('/confirm_placement', methods=['POST'])
def confirm_placement():
    application_id = request.form.get('application_id')
    if application_id:
        # Update the application status to 'Accepted' in the
database
        db_functions.update_application_status(application_id,
'Accepted')
        print(f"Application {application_id} status updated to
'Accepted'")  # Debug message
    else:
        print("Application ID not received")  # Debug message

    return redirect(url_for('place_student'))

# Flask route for update/delete entries
@app.route('/update_delete_entries', methods=['GET', 'POST'])
def update_delete_entries():
    if request.method == 'POST':
        action = request.form.get('action')
        table = request.form.get('table')
```

```
        entry_id = request.form.get('entry_id')

        # Perform the respective action for each table
        if action == 'Update':
            if table == 'student':
                db_functions.update_student(entry_id,
request.form)
            elif table == 'company':
                db_functions.update_company(entry_id,
request.form)
            elif table == 'application':
                db_functions.update_application(entry_id,
request.form)
            elif table == 'interview':
                db_functions.update_interview(entry_id,
request.form)
            elif table == 'job_posting':
                db_functions.update_job_posting(entry_id,
request.form)
        elif action == 'Delete':
            if table == 'student':
                db_functions.delete_student(entry_id)
            elif table == 'company':
                db_functions.delete_company(entry_id)
            elif table == 'application':
                db_functions.delete_application(entry_id)
            elif table == 'interview':
                db_functions.delete_interview(entry_id)
            elif table == 'job_posting':
                db_functions.delete_job_posting(entry_id)

        return redirect(url_for('update_delete_entries'))

    students = db_functions.view_students()
    companies = db_functions.view_companies()
    applications = db_functions.view_all_applications()
    interviews = db_functions.view_all_interviews()
    job_postings = db_functions.view_job_postings()

    return render_template('update_delete_entries.html',
students=students, companies=companies,
applications=applications, interviews=interviews,
job_postings=job_postings)

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

2. db_functions.py

```
import mysql.connector
from mysql.connector import Error

def get_connection():
    try:
        connection = mysql.connector.connect(
            host='localhost',
            database='campus_recruitment',
            user='root',
            password='Declan@23'
        )
        if connection.is_connected():
            return connection
    except Error as e:
        print(f"Error while connecting to MySQL: {e}")
        return None

def close_connection(connection):
    if connection and connection.is_connected():
        connection.close()

# Student Functions
def insert_student(student_id, name, age, degree, branch):
    connection = get_connection()
    if connection:
        cursor = connection.cursor()
        cursor.execute("INSERT INTO STUDENT (student_id, name,
age, degree, branch) VALUES (%s, %s, %s, %s, %s)",
                       (student_id, name, age, degree,
branch))
        connection.commit()
        close_connection(connection)

def view_students():
    connection = get_connection()
    if connection:
        cursor = connection.cursor(dictionary=True)
        cursor.execute("SELECT * FROM STUDENT")
        students = cursor.fetchall()
        close_connection(connection)
        if not students:
            print("No students found in the STUDENT table.")
        return students

def view_students_with_applications():
    connection = get_connection()
    if connection:
        cursor = connection.cursor(dictionary=True)
```

```
        cursor.execute("""
            SELECT S.student_id, S.name, COUNT(A.job_id) AS
applications_count
            FROM STUDENT S
            LEFT JOIN APPLICATION A ON S.student_id =
A.student_id
            GROUP BY S.student_id
            HAVING COUNT(A.job_id) > 0
        """)
        students = cursor.fetchall()
        close_connection(connection)
        return students

def view_students_without_applications():
    connection = get_connection()
    if connection:
        cursor = connection.cursor(dictionary=True)
        cursor.execute("""
            SELECT S.student_id, S.name, S.age, S.branch,
COUNT(A.job_id) AS applications_count
            FROM STUDENT S
            LEFT JOIN APPLICATION A ON S.student_id =
A.student_id
            GROUP BY S.student_id
            HAVING COUNT(A.job_id) = 0
        """)
        students = cursor.fetchall()
        close_connection(connection)
        return students

def view_jobs_applied_per_student():
    connection = get_connection()
    if connection:
        cursor = connection.cursor(dictionary=True)
        cursor.execute("""
            SELECT S.student_id, S.name, COUNT(A.job_id) AS
job_applications
            FROM STUDENT S
            JOIN APPLICATION A ON S.student_id = A.student_id
            GROUP BY S.student_id
        """)
        students = cursor.fetchall()
        close_connection(connection)
        return students

# Company Functions
def insert_company(company_id, name, location, sector):
    connection = get_connection()
    if connection:
        cursor = connection.cursor()
```

```
        cursor.execute("INSERT INTO COMPANY (company_id, name,
location, sector) VALUES (%s, %s, %s, %s)",
                       (company_id, name, location, sector))
        connection.commit()
        close_connection(connection)

def view_companies():
    connection = get_connection()
    if connection:
        cursor = connection.cursor(dictionary=True)
        cursor.execute("SELECT * FROM COMPANY")
        companies = cursor.fetchall()
        close_connection(connection)
        return companies

# Job Posting Functions
def insert_job_posting(job_id, title, company_id, description,
salary):
    connection = get_connection()
    if connection:
        cursor = connection.cursor()
        cursor.execute("INSERT INTO JOB_POSTING (job_id,
title, company_id, description, salary) VALUES (%s, %s, %s,
%s, %s)",
                       (job_id, title, company_id,
description, salary))
        connection.commit()
        close_connection(connection)

def view_job_postings():
    connection = get_connection()
    if connection:
        cursor = connection.cursor(dictionary=True)
        cursor.execute("SELECT * FROM JOB_POSTING")
        jobs = cursor.fetchall()
        close_connection(connection)
        return jobs

def view_job_postings_with_company_info():
    connection = get_connection()
    if connection:
        cursor = connection.cursor(dictionary=True)
        cursor.execute("""
            SELECT JP.job_id, JP.title, JP.description, C.name
AS company_name, C.location, C.sector
            FROM JOB_POSTING JP
            JOIN COMPANY C ON JP.company_id = C.company_id
        """)
        job_postings = cursor.fetchall()
        close_connection(connection)
        return job_postings
```

```
# Application and Interview Management Functions
def apply_for_job(application_id, student_id, job_id):
    connection = get_connection()
    if connection:
        cursor = connection.cursor()
        cursor.execute("""
            INSERT INTO APPLICATION (APPLICATION_ID,
STUDENT_ID, JOB_ID, STATUS)
            VALUES (%s, %s, %s, 'Pending')
        """, (application_id, student_id, job_id))  # Insert
all three fields into the table
        connection.commit()
        close_connection(connection)
def view_all_applications():
    connection = get_connection()
    if connection:
        cursor = connection.cursor(dictionary=True)
        cursor.execute("""
            SELECT A.application_id, A.status, S.name AS
student_name, JP.title AS job_title, C.name AS company_name,
                   IFNULL(I.scheduled_date, '-') AS
scheduled_date,
                   IFNULL(I.status, '-') AS interview_status,
                   IFNULL(I.location, '-') AS location
            FROM APPLICATION A
            JOIN STUDENT S ON A.student_id = S.student_id
            JOIN JOB_POSTING JP ON A.job_id = JP.job_id
            JOIN COMPANY C ON JP.company_id = C.company_id
            LEFT JOIN INTERVIEW I ON A.application_id =
I.application_id
        """)
        applications = cursor.fetchall()
        close_connection(connection)
        return applications

def get_application_details(application_id):
    """
    Fetch application details based on application_id.
    """
    connection = get_connection()
    cursor = connection.cursor(dictionary=True)

    # Query to get application details
    cursor.execute("SELECT * FROM application WHERE
application_id = %s", (application_id,))
    application = cursor.fetchone()

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

```
    return application

def update_application_status(application_id, status):
    """
    Update the status of an application and its associated
interviews.
    """
    connection = get_connection()
    cursor = connection.cursor()

    # Update the status of the application
    cursor.execute("UPDATE application SET status = %s WHERE
application_id = %s", (status, application_id))

    # Update the status of the interviews linked to this
application
    cursor.execute("UPDATE interview SET status = %s WHERE
application_id = %s", (status, application_id))

    connection.commit()

    cursor.close()
    connection.close()

def view_application_details(application_id):
    connection = get_connection()
    if connection:
        cursor = connection.cursor(dictionary=True)
        cursor.execute("""
            SELECT A.application_id, A.status, S.name AS
student_name, JP.title AS job_title, C.name AS company_name,
                   I.scheduled_date, I.status AS
interview_status, I.location
            FROM APPLICATION A
            JOIN STUDENT S ON A.student_id = S.student_id
            JOIN JOB_POSTING JP ON A.job_id = JP.job_id
            JOIN COMPANY C ON JP.company_id = C.company_id
            LEFT JOIN INTERVIEW I ON A.application_id =
I.application_id
            WHERE A.application_id = %s
        """, (application_id,))
        details = cursor.fetchone()
        close_connection(connection)
        return details

def schedule_interview(application_id, interview_id,
scheduled_date, status, location):
    connection = get_connection()
    if connection:
        cursor = connection.cursor()
        cursor.execute("""
```

```
            INSERT INTO INTERVIEW (INTERVIEW_ID,
APPLICATION_ID, SCHEDULED_DATE, STATUS, LOCATION)
            VALUES (%s, %s, %s, %s, %s)
        """, (interview_id, application_id, scheduled_date,
status, location))
        connection.commit()
        close_connection(connection)

def view_applications_by_status(status):
    connection = get_connection()
    if connection:
        cursor = connection.cursor(dictionary=True)
        cursor.execute("SELECT * FROM APPLICATION WHERE status
= %s", (status,))
        applications = cursor.fetchall()
        close_connection(connection)
        return applications
def view_all_interviews():
    connection = get_connection()
    if connection:
        cursor = connection.cursor(dictionary=True)
        cursor.execute("SELECT * FROM INTERVIEW")
        interviews = cursor.fetchall()
        close_connection(connection)
        return interviews
    return []

def get_interviews_by_status(status):
    connection = get_connection()
    interviews = []
    if connection:
        cursor = connection.cursor(dictionary=True)
        cursor.execute("""
            SELECT INTERVIEW.INTERVIEW_ID,
INTERVIEW.APPLICATION_ID, INTERVIEW.SCHEDULED_DATE,
                   INTERVIEW.STATUS, INTERVIEW.LOCATION,
STUDENT.NAME AS STUDENT_NAME
            FROM INTERVIEW
            JOIN APPLICATION ON INTERVIEW.APPLICATION_ID =
APPLICATION.APPLICATION_ID
            JOIN STUDENT ON APPLICATION.STUDENT_ID =
STUDENT.STUDENT_ID
            WHERE INTERVIEW.STATUS = %s
        """, (status,))
        interviews = cursor.fetchall()
        close_connection(connection)
    return interviews

def delete_entry(table, entry_id):
    connection = get_connection()
    if connection:
```

```
        cursor = connection.cursor()
        cursor.execute(f"DELETE FROM {table.upper()} WHERE
{table.upper()}_ID = %s", (entry_id,))
        connection.commit()
        close_connection(connection)

def update_entry(table, entry_id, new_data):
    connection = get_connection()
    if connection:
        cursor = connection.cursor()
        # Constructing dynamic update query
        update_query = f"UPDATE {table.upper()} SET " + ",
".join([f"{key} = %s" for key in new_data.keys()]) + \
                       f" WHERE {table.upper()}_ID = %s"
        values = list(new_data.values()) + [entry_id]
        cursor.execute(update_query, values)
        connection.commit()
        close_connection(connection)

def view_scheduled_applications_and_job_postings():
    connection = get_connection()
    if connection:
        cursor = connection.cursor(dictionary=True)
        cursor.execute("""
            SELECT A.application_id, S.name AS student_name,
JP.title AS job_title, I.date_time, I.location
            FROM APPLICATION A
            JOIN STUDENT S ON A.student_id = S.student_id
            JOIN JOB_POSTING JP ON A.job_id = JP.job_id
            JOIN INTERVIEW I ON A.application_id =
I.application_id
        """)
        scheduled_interviews = cursor.fetchall()
        close_connection(connection)
        return scheduled_interviews

def update_student_status_after_acceptance(application_id):
    connection = get_connection()
    if connection:
        cursor = connection.cursor()
        cursor.execute("UPDATE APPLICATION SET status =
'Accepted' WHERE application_id = %s", (application_id,))
        connection.commit()
        close_connection(connection)

def view_placed_students():
    connection = get_connection()
    if connection:
        cursor = connection.cursor(dictionary=True)
        cursor.execute("""
```

```
            SELECT S.student_id, S.name, JP.title AS
job_title, JP.salary, C.name AS company_name
            FROM STUDENT S
            JOIN APPLICATION A ON S.student_id = A.student_id
            JOIN JOB_POSTING JP ON A.job_id = JP.job_id
            JOIN COMPANY C ON JP.company_id = C.company_id
            WHERE A.status = 'Accepted'
        """)
        placed_students = cursor.fetchall()
        close_connection(connection)
        return placed_students

def delete_student(student_id):
    connection = get_connection()
    if connection:
        cursor = connection.cursor()
        delete_query = "DELETE FROM STUDENT WHERE STUDENT_ID =
%s"
        cursor.execute(delete_query, (student_id,))
        connection.commit()
        close_connection(connection)

def delete_company(company_id):
    connection = get_connection()
    if connection:
        cursor = connection.cursor()
        delete_query = "DELETE FROM COMPANY WHERE COMPANY_ID =
%s"
        cursor.execute(delete_query, (company_id,))
        connection.commit()
        close_connection(connection)

def delete_application(application_id):
    connection = get_connection()
    if connection:
        cursor = connection.cursor()
        delete_query = "DELETE FROM APPLICATION WHERE
APPLICATION_ID = %s"
        cursor.execute(delete_query, (application_id,))
        connection.commit()
        close_connection(connection)

def delete_interview(interview_id):
    connection = get_connection()
    if connection:
        cursor = connection.cursor()
        delete_query = "DELETE FROM INTERVIEW WHERE
INTERVIEW_ID = %s"
        cursor.execute(delete_query, (interview_id,))
        connection.commit()
        close_connection(connection)
```

```
def delete_job_posting(job_id):
    connection = get_connection()
    if connection:
        cursor = connection.cursor()
        delete_query = "DELETE FROM JOB_POSTING WHERE JOB_ID =
%s"
        cursor.execute(delete_query, (job_id,))
        connection.commit()
        close_connection(connection)

def update_student(student_id, form_data):
    connection = get_connection()
    if connection:
        cursor = connection.cursor()
        update_query = """
            UPDATE STUDENT SET NAME = %s, AGE = %s, DEGREE =
%s, BRANCH = %s
            WHERE STUDENT_ID = %s
        """
        cursor.execute(update_query, (
            form_data.get('name'),
            form_data.get('age'),
            form_data.get('degree'),
            form_data.get('branch'),
            student_id
        ))
        connection.commit()
        close_connection(connection)

def update_company(company_id, form_data):
    connection = get_connection()
    if connection:
        cursor = connection.cursor()
        update_query = """
            UPDATE COMPANY SET NAME = %s, LOCATION = %s,
SECTOR = %s
            WHERE COMPANY_ID = %s
        """
        cursor.execute(update_query, (
            form_data.get('name'),
            form_data.get('location'),
            form_data.get('sector'),
            company_id
        ))
        connection.commit()
        close_connection(connection)

def update_application(application_id, form_data):
    connection = get_connection()
    if connection:
```

```
        cursor = connection.cursor()
        update_query = """
            UPDATE APPLICATION SET STUDENT_ID = %s, JOB_ID =
%s, STATUS = %s
            WHERE APPLICATION_ID = %s
        """
        cursor.execute(update_query, (
            form_data.get('student_id'),
            form_data.get('job_id'),
            form_data.get('status'),
            application_id
        ))
        connection.commit()
        close_connection(connection)

def update_interview(interview_id, form_data):
    connection = get_connection()
    if connection:
        cursor = connection.cursor()
        update_query = """
            UPDATE INTERVIEW SET APPLICATION_ID = %s, DATE =
%s, STATUS = %s
            WHERE INTERVIEW_ID = %s
        """
        cursor.execute(update_query, (
            form_data.get('application_id'),
            form_data.get('date'),
            form_data.get('status'),
            interview_id
        ))
        connection.commit()
        close_connection(connection)

def update_job_posting(job_id, form_data):
    connection = get_connection()
    if connection:
        cursor = connection.cursor()
        update_query = """
            UPDATE JOB_POSTING SET TITLE = %s, COMPANY_ID =
%s, LOCATION = %s, DESCRIPTION = %s
            WHERE JOB_ID = %s
        """
        cursor.execute(update_query, (
            form_data.get('title'),
            form_data.get('company_id'),
            form_data.get('location'),
            form_data.get('description'),
            job_id
        ))
        connection.commit()
        close_connection(connection)
```

3. home.html

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>Campus Recruitment System</title>
    <style>
        body {
            font-family: 'Georgia', serif;
            background-color: #f8f9fa;
            margin: 0;
            padding: 0;
            color: #2c3e50;
            line-height: 1.6;
        }
        .container {
            width: 75%;
            max-width: 1000px;
            margin: 50px auto;
            padding: 30px;
            background-color: #ffffff;
            box-shadow: 0 5px 15px rgba(0, 0, 0, 0.1);
            border-radius: 6px;
            border-top: 4px solid #2c3e50;
        }
        h1 {
            color: #2c3e50;
            text-align: center;
            font-size: 2.4em;
            margin-bottom: 10px;
            font-weight: 700;
        }
        h2 {
            font-size: 1.6em;
            color: #2c3e50;
            margin-top: 30px;
            margin-bottom: 15px;
            padding-bottom: 4px;
            border-bottom: 1px solid #2c3e50;
        }
        .section {
            margin: 20px 0;
        }
        a {
            text-decoration: none;
            color: #2c3e50;
            font-size: 1em;
            display: block;
```

```
            margin: 8px 0;
            padding: 12px 15px;
            border-radius: 6px;
            font-weight: 500;
            transition: background-color 0.3s, color 0.3s,
box-shadow 0.3s;
            background-color: #f1f3f5;
            box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
        }
        a:hover {
            background-color: #e1e6ea;
            color: #1a242f;
            box-shadow: 0 6px 12px rgba(0, 0, 0, 0.15);
        }
        .button {
            display: inline-block;
            padding: 10px 24px;
            background-color: #2c3e50;
            color: white;
            border: none;
            cursor: pointer;
            text-align: center;
            font-size: 1em;
            border-radius: 4px;
            transition: background-color 0.3s;
            font-weight: 600;
        }
        .button:hover {
            background-color: #1a242f;
        }
    </style>
</head>
<body>
    <div class="container">
        <h1>Campus Recruitment and Management System</h1>
        <!-- Student Management Links -->
        <div class="section">
            <h2>Student Management</h2>
            <a href="{{ url_for('add_student') }}">Add
Student</a>
            <a href="{{ url_for('view_students') }}">View All
Students</a>
            <a href="{{ url_for('students_with_applications')
}}">View Students With Applications</a>
            <a href="{{
url_for('students_without_applications') }}">View Students
Without Applications</a>
        </div>

        <!-- Company Management Links -->
        <div class="section">
```

```
            <h2>Company Management</h2>
            <a href="{{ url_for('add_company') }}">Add
Company</a>
            <a href="{{ url_for('view_companies') }}">View All
Companies</a>
        </div>

        <!-- Job Posting Links -->
        <div class="section">
            <h2>Job Postings</h2>
            <a href="{{ url_for('add_job_posting') }}">Add Job
Posting</a>
            <a href="{{ url_for('view_job_postings') }}">View
All Job Postings</a>
            <a href="{{
url_for('view_job_postings_with_company_info') }}">View Job
Postings with Company Info</a>
        </div>

        <!-- Application and Interview Links -->
        <div class="section">
            <h2>Applications and Interviews</h2>
            <a href="{{ url_for('apply_for_job', job_id=1)
}}">Apply for a Job</a>
            <a href="{{ url_for('view_all_applications',
application_id=1) }}">View Application, Interview Details</a>
            <a href="{{ url_for('schedule_interview')
}}">Schedule an Interview</a>
            <a href="{{ url_for('view_interviews_by_status')
}}">View Interviews by Status</a>
        </div>

        <!-- Placement Links -->
        <div class="section">
            <h2>Placement</h2>
            <a href="/place_student">Place a Student</a>
            <a href="{{ url_for('view_placed_students')
}}">View Placed Students</a>
        </div>
    </div>
</body>
</html>
```

CONCLUSION :

The Campus Recruitment Management System project provided an opportunity to apply database management concepts to a practical scenario, offering a streamlined solution for handling recruitment processes on a campus. Beginning with the identification of entities, attributes, and relationships, we designed an Entity-Relationship (ER) diagram to visually represent the database structure. This step ensured clarity and logical consistency in capturing the system's requirements.

Following the design phase, the ER diagram was mapped to a relational schema, converting it into a database structure ready for implementation. To enhance data integrity and eliminate redundancy, we normalized the database to the Third Normal Form (3NF), ensuring a robust and efficient data model.

With the normalized schema as a foundation, we built the project, integrating Oracle SQL for creating, querying, and managing the database. The result is a fully functional Campus Recruitment Management System capable of efficiently storing and retrieving recruitment-related information.

This project enhanced our understanding of database design, normalization principles, and SQL implementation, while also honing our problem-solving and teamwork skills. The experience gained will be invaluable for future endeavors in database-driven application development.