PROJECT TITLE:

STUDENT PLACEMENT ANALYTICS

Problem Statement : Develop a **Python-based dashboard** to analyze student placement data (year-wise, branch-wise, and company-wise) and provide insights such as top recruiters and company-wise hiring records with horizontal scrolling support.

Name : Divya KN (4GW23CI012)

Email : divyakn004@gmail.com

Date : 02-09-2025

INDEX :

PROBLEM STATEMENT:

The Placement Cell currently maintains student and company placement data but lacks advanced analytical insights. This makes it difficult to evaluate **year-wise**, **branch-wise**, and **company-wise placement trends**, **identify top recruiters, and review hiring records** effectively. A centralized analytics system is required to transform raw placement data into interactive insights for better decision-making.

Project Overview :

This project develops a **Python-based Student Placement Analytics Dashboard** powered by **Streamlit** and **MySQL**. It provides year-wise, branchwise, and company-wise placement analysis, top recruiter identification, CGPA placement distribution, and horizontally scrollable company-wise hiring records. The interactive dashboard supports filtering by batch, department, and company, enabling the Placement Cell to make data-driven decisions with ease.

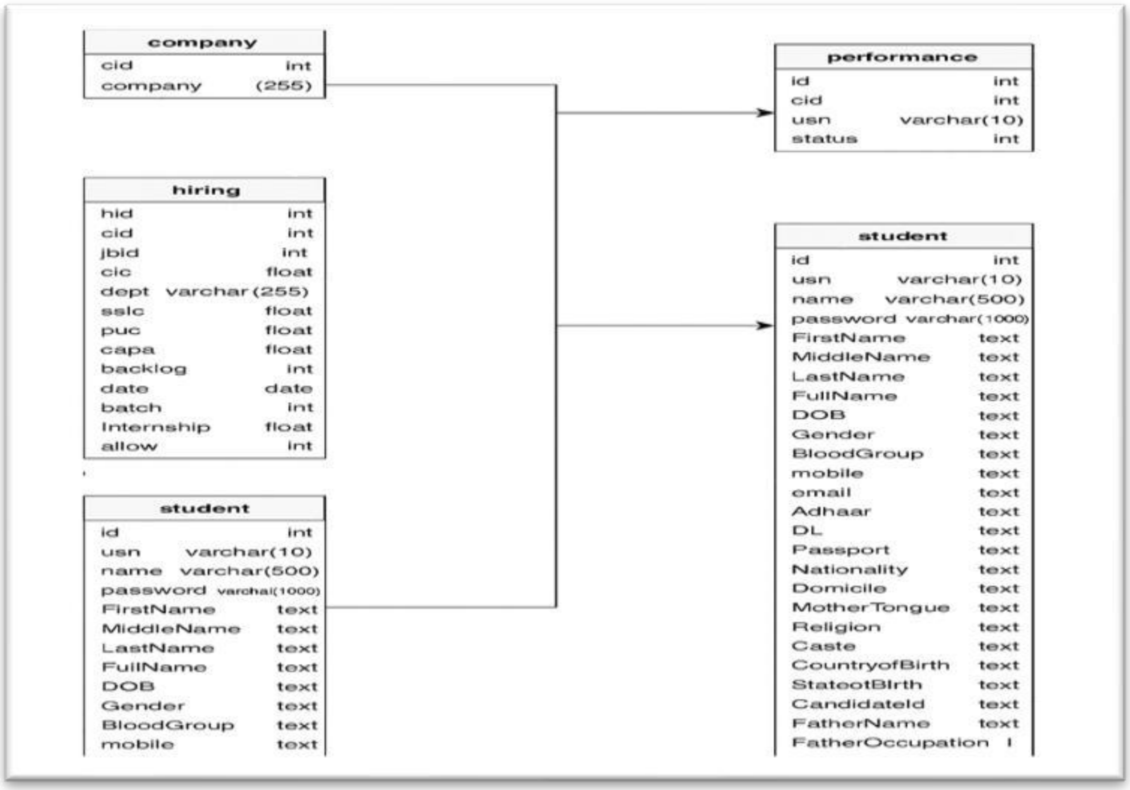
Dataset Description :

The system uses placement-related datasets stored in a MySQL database with the following tables:

1. **student** – Stores student academic details (ID, Name, Batch, Branch, CGPA, etc.).
2. **student1** – Contains extended student info such as email, gender, and contact details.
3. **company** – Records details of recruiting companies (Company ID, Company Name, Domain, etc.).

4. **performance** – Tracks student performance in recruitment rounds (Aptitude, GD, Technical, HR, etc.).
5. **hiring** – Contains final hiring decisions, offer status, and recruitment outcomes.

ER Diagram:



Features Implemented :

1. Year-wise Placement Analysis

a. Placement data is analyzed batch-wise, with the current year automatically detected from the system date.

1. **Branch-wise Placement Analysis**
   a. Placement statistics are compared across different branches to identify department-level performance.
2. **Company-wise Analytics**
   a. Provides details of top recruiters.
   b. Hiring records are displayed in a horizontally scrollable format for easy comparison across companies.
3. **Placement Statistics Overview**
   a. Displays the total number of students placed, not placed, and the overall conversion ratio (eligible vs placed).
4. **Interactive Visualization**
   a. Donut and bar charts present placement distribution in a visually intuitive way.
   b. Trends can be analyzed interactively for better decision-making.
5. **Data Cleaning & Standardization**
   a. Inconsistencies in branch names (e.g., *AI&DS* vs *AI & DS*) are resolved for accurate analytics.

Future Enhancements :

1. **Predictive Modeling**
   a. Implement machine learning models to predict student placement probability based on academic records, skills, and past placement trends.
2. **Salary Insights**

a. Extend the system to include salary-based analytics (highest, average, lowest) for better evaluation of placement quality.

Technical Architecture :

1. Data Layer (Storage & Input)

- **Datasets Used:** Student details, company data, performance records, and hiring information (CSV/Excel format).
- **Data Handling:** Pandas is used for loading, cleaning, merging, and transforming datasets.

2. Processing Layer (Backend Logic)

- **Language:** Python
- **Libraries:** Pandas, NumPy
- **Responsibilities:**

  ○ Data preprocessing (filtering, merging student–company–hiring records). ○ Aggregations (year-wise, branch-wise, company-wise statistics). ○ Preparing datasets for visualization (counts, percentages, KPIs).

3. Analytics & Visualization Layer (Application Layer)

- **Framework:** Streamlit (used to build the dashboard).
- **Visualization Libraries:** Plotly (interactive charts), Matplotlib (basic plots).
- **Features Implemented:**

  ○ Year-wise placement statistics. ○ Branch-wise & company-wise placement analysis.

  ○ Donut charts for placement distribution. ○ Horizontally scrollable company-wise hiring records.

4. Presentation Layer (User Interface)

- **Interface:** Streamlit web-based dashboard.
- **Output:** ○ Interactive visualizations, Scrollable tables for hiring data.

Code Walkthrough :

`db_config.py` : Database Configuration

```
import mysql.connector

def get_connection():    try:        conn = 
mysql.connector.connect(            host="localhost", 
user="root",            password="",          # Your MySQL 
password            database="statistics",  # Your DB name 
            port=3307    # Change to actual port from XAMPP 
        )        return conn    except 
mysql.connector.Error as err:        print(f"MySQL 
connection error: {err}")        return None
```

Explanation:

- Provides a reusable function `get_connection()` to establish connection with **MySQL database**.
- Ensures flexibility to change credentials/port.
- Includes error handling to capture connection failures.

2. `data_loader.py` : Data Loading & Integration

```
import pandas as pd import
mysql.connector from datetime import
datetime from db_config import
get_connection

def load_all_data():
    conn = get_connection()

    student1_df  =  pd.read_sql("SELECT  *  FROM  student1",
conn)
    student2_df = pd.read_sql("SELECT * FROM student", conn)
    company_df = pd.read_sql("SELECT * FROM company",
conn)    performance_df = pd.read_sql("SELECT * FROM
performance", conn)    hiring_df = pd.read_sql("SELECT *
FROM hiring", conn)

    query = """
    SELECT s.usn, s.name, s.dept, s.batch, s.cgpa,
p.status, c.company
    FROM student s
    LEFT JOIN performance p ON s.usn = p.usn
    LEFT JOIN company c ON p.cid = c.cid;
    """
    combined_df = pd.read_sql(query, conn)

    conn.close()
    return student1_df, student2_df, company_df,
performance_df, hiring_df, combined_df
```

Explanation:

- Imports all required tables (`student1`, `student`, `company`, `performance`, `hiring`).

- Runs an **SQL join query** to generate a consolidated dataset `combined_df` that merges students with their performance & company data.
- Returns multiple dataframes for downstream analysis in the dashboard.

3. `dashboard.py` – Main Streamlit Dashboard

a. Imports & Status Mapping

```
import streamlit as st import pandas as pd import
plotly.express as px from src.data_loader import
load_all_data from plotly.subplots import make_subplots
import plotly.graph_objects as go

STATUS_MAP = {
    0: "Not Eligible",
    1: "Unable to Clear 1st Round",
    2: "Unable to Clear GD",
    3: "Unable to Clear Technicals",
    4: "Unable to Clear HR",
    9: "Shortlisted",
    10: "Placed"
}
```

Explanation:

- Loads visualization (`plotly`) and dashboard (`streamlit`) libraries.
- Defines a **status mapping dictionary** to convert numeric status codes into readable labels.

b. Helper Functions

```
def map_status(status_code):
    return STATUS_MAP.get(status_code, "Unknown")
```

- Converts raw status codes into human-readable text.

```
def apply_filters(df, batch_filter, dept_filter,
company_filter=None):
# Applies batch, department, and company filters
dynamically
```

- Implements **filtering logic** for interactive selection (batch, department, company).

```
def compute_kpis(student_filtered_df, filtered_df):
    total_students = student_filtered_df['id'].nunique()
total_placed =
filtered_df[filtered_df['status'].isin([9,
10])]['id'].nunique()
    return total_students, total_placed
```

- Computes **key performance indicators (KPIs)** → total students vs placed students.

c. Main Dashboard Flow

```
def main():
    st.set_page_config(page_title="Student Placement
Analysis", layout="wide")    st.title("🎓 Student
Placement Analysis Dashboard")

    @st.cache_data    def
get_all_data():        return
load_all_data()
```

- Sets up **page layout** & **title**.
- Uses `@st.cache_data` to **cache results** and avoid redundant DB queries.

d. Sidebar Filters

```
st.sidebar.header("🔍 Filters")
selected_batch = st.sidebar.selectbox("Select Batch",
["All", "Last 3 Years"] + all_batches)
selected_dept = st.sidebar.selectbox("Select Department",
["All"] + all_departments)
selected_company = st.sidebar.selectbox("Select Company",
["All"] + all_companies)
```

Explanation:

- Interactive **filters** allow users to refine analysis dynamically.
- Supports "All", "Last 3 Years", or specific selection.

e. KPI Metrics

```
col1, col2, col3 = st.columns(3) col1.metric("Total
Students", total_students) col2.metric("Placed
(Shortlisted + Placed)", total_placed)
```
Explanation:

- Displays **summary KPIs** in cards for quick insights.

f. Placement Status Visualization

```
fig_status = px.bar(
status_counts,
    x='Status', y='Percentage', color='Status',
title='Placement Status (%)',
    text=status_counts['Percentage'].map("{:.1f}%".format)
)
st.plotly_chart(fig_status, use_container_width=True)
```

Explanation:

- Creates a **bar chart** showing percentage distribution of placement statuses (Not Eligible, Shortlisted, Placed, etc.).

g. Batch & Branch Analysis

```
fig_batch = px.bar(
batch_stats,
    x='batch', y='count', color='status_text',
barmode='stack',
    title='Batch-wise Placement Status'
)
st.plotly_chart(fig_batch, use_container_width=True)
```

- **Batch-wise** stacked bar chart for placement progress.

```
branch_stats = filtered_df.groupby(['branch',
'status_text']).size().unstack(fill_value=0)
st.bar_chart(branch_stats)
```

- **Branch-wise** distribution of placement outcomes.

h. Top Recruiters

```
def get_top_companies(df):
    placed = df[df['status'].isin([9, 10])]    return
placed['company'].value_counts().head(10)
```

• Identifies **top recruiting companies** by number of placed students.

i. CGPA Placement Analysis (Donut Charts)

```
fig = make_subplots(rows=1, cols=2, specs=[[{'type':
'domain'}, {'type': 'domain'}]],
subplot_titles=['All Students CGPA Distribution', 'Placed
Students CGPA Distribution']) fig.add_trace(go.Pie(...), row=1,
col=1) fig.add_trace(go.Pie(...), row=1, col=2)
st.plotly_chart(fig, use_container_width=True)
```

Explanation:

• Creates **side-by-side donut charts** comparing:

○ Distribution of all students across CGPA ranges. ○ Distribution of **only placed students** across CGPA ranges.

j. Hiring Records (Sticky Table + Download)

```
pivot_df = filtered_df.pivot_table(
index=['id', 'name', 'dept', 'batch'],
columns='company',
values='status_text',    aggfunc='first'
).reset_index()
```

- Creates a **pivot table** of hiring records (students × companies).

```
st.markdown(custom_css, unsafe_allow_html=True)
st.markdown(f'<div class="sticky-
wrapper">{table_html}</div>', unsafe_allow_html=True)
st.download_button(label=" Download CSV",
data=pivot_df.to_csv(index=False).encode('utf-8'), ...)
```

- Uses **custom CSS** to make first 4 columns sticky & horizontally scrollable.
- Adds **CSV export button** for offline analysis.

k. Entry Point

```
if __name__ == "__main__":
    main()
```

• Ensures that dashboard runs only when executed directly.

Setup Instructions :

1. Install Required Software

- Install **Python 3.9+**
- Install **MySQL Server** (e.g., via XAMPP or WAMPP).
- Install a **code editor** (VS Code).

2. Create & Configure Database

1. Open MySQL and create a database:
   ```
   CREATE DATABASE statistics
   ```

2. Create required tables:

    a. `student`
    b. `company`
    c. `hiring`
    d. `Performance`

3. Install Python Dependencies

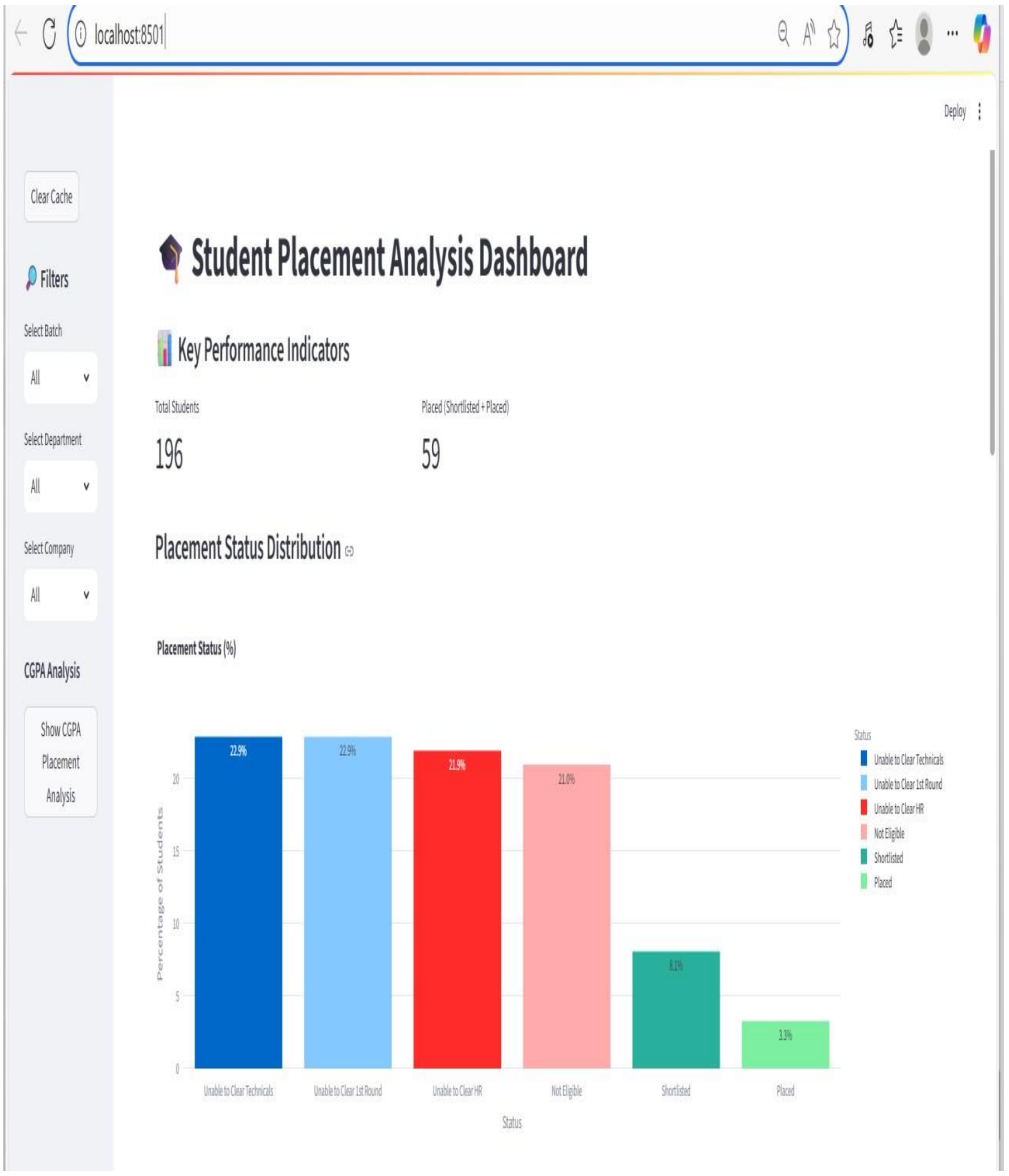
Open terminal in your project folder and run:

```
pip install pandas pymysql streamlit plotly
```

4.Run the Dashboard *python -m streamlit run src/dashboard.py*

This will launch the dashboard in your **browser** (default: `http://localhost:8501`).

Screenshots : Overall Placement Analysis



Student Placement Analysis Dashboard

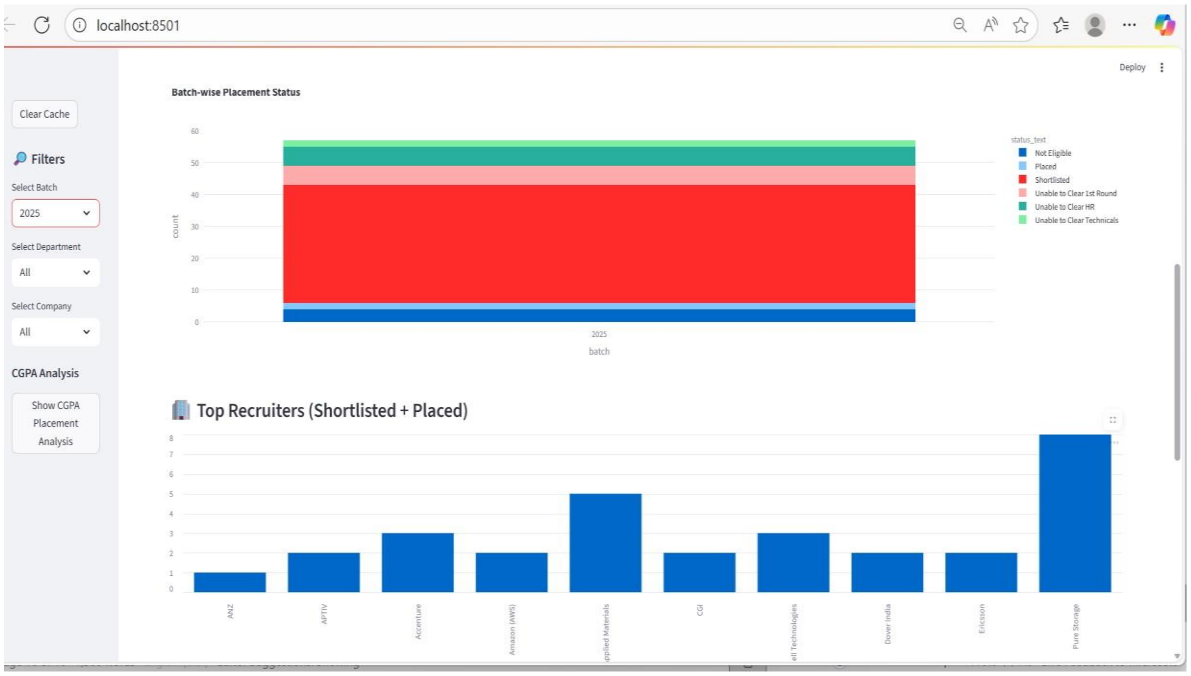
Key Performance Indicators

Total Students
196

Placed (Shortlisted + Placed)
59

Placement Status Distribution

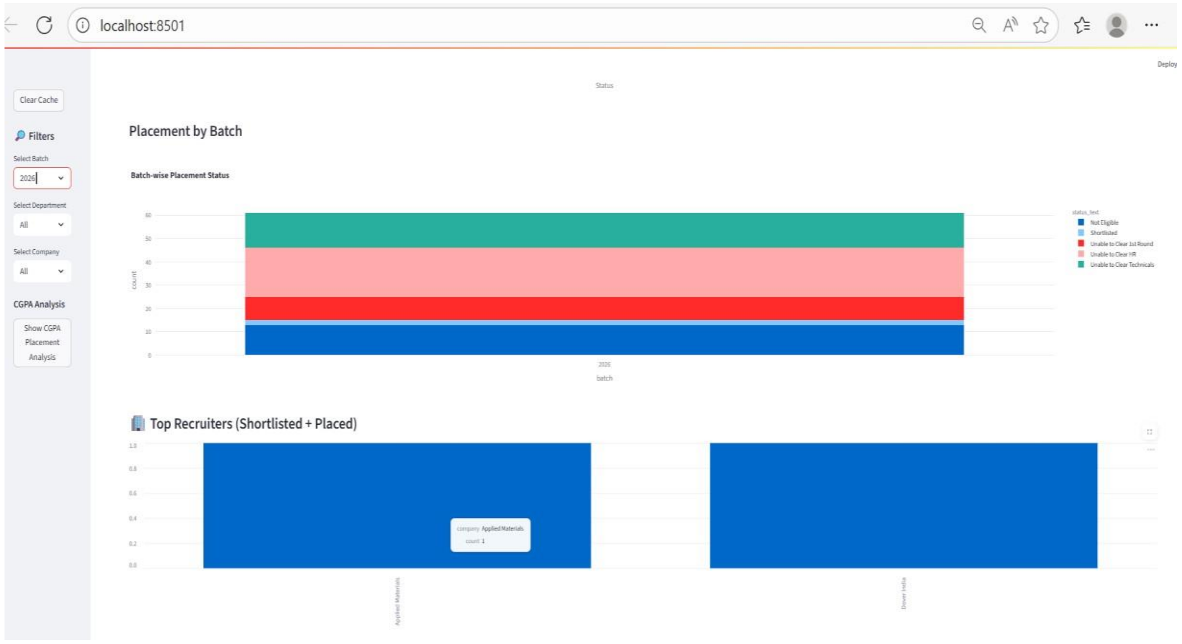
Placement By Batch and Top Recruiters



Select Batch: 2025

Select Department: All

Select Company: All

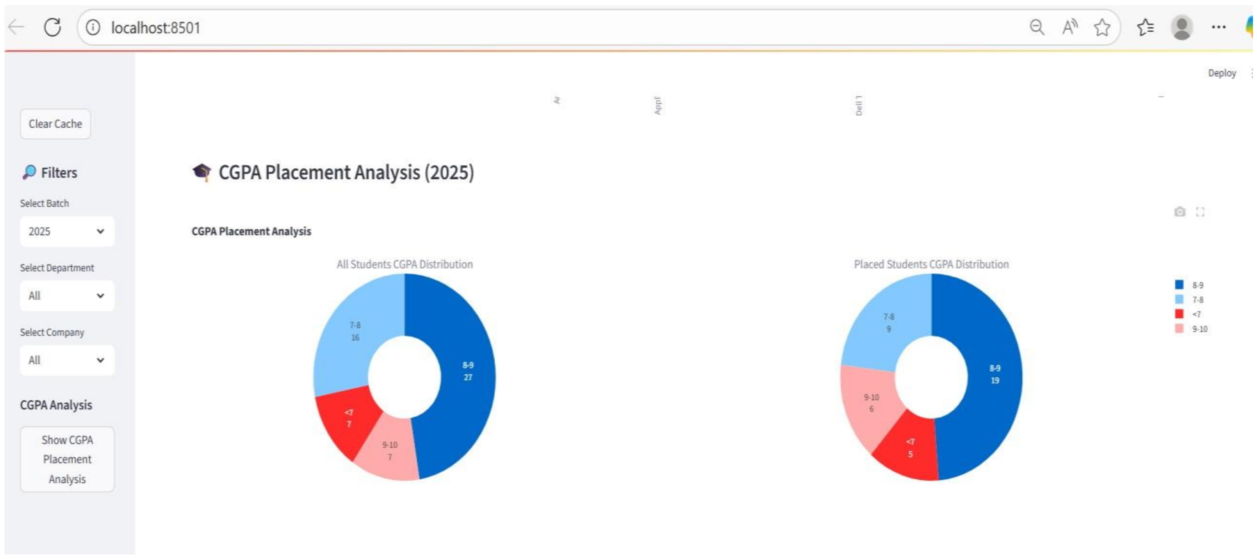


Select Batch: 2026

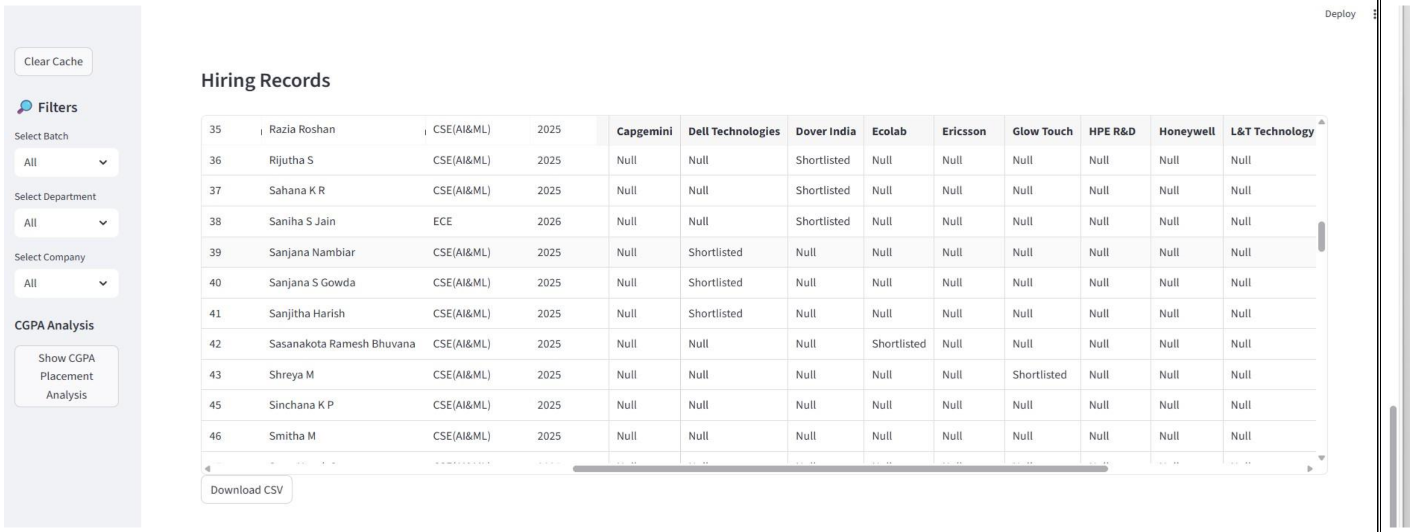
Select Department: All

Select Company: All

CGPA Placement Analysis



Displaying Hiring Records



Hiring Records

				Capgemini	Dell Technologies	Dover India	Ecolab	Ericsson	Glow Touch	HPE R&D	Honeywell	L&T Technology
35	Razia Roshan	CSE(AI&ML)	2025									
36	Rijutha S	CSE(AI&ML)	2025	Null	Null	Shortlisted	Null	Null	Null	Null	Null	Null
37	Sahana K R	CSE(AI&ML)	2025	Null	Null	Shortlisted	Null	Null	Null	Null	Null	Null
38	Saniha S Jain	ECE	2026	Null	Null	Shortlisted	Null	Null	Null	Null	Null	Null
39	Sanjana Nambiar	CSE(AI&ML)	2025	Null	Shortlisted	Null	Null	Null	Null	Null	Null	Null
40	Sanjana S Gowda	CSE(AI&ML)	2025	Null	Shortlisted	Null	Null	Null	Null	Null	Null	Null
41	Sanjitha Harish	CSE(AI&ML)	2025	Null	Shortlisted	Null	Null	Null	Null	Null	Null	Null
42	Sasanakota Ramesh Bhuvana	CSE(AI&ML)	2025	Null	Null	Null	Shortlisted	Null	Null	Null	Null	Null
43	Shreya M	CSE(AI&ML)	2025	Null	Null	Null	Null	Null	Shortlisted	Null	Null	Null
45	Sinchana K P	CSE(AI&ML)	2025	Null	Null	Null	Null	Null	Null	Null	Null	Null
46	Smitha M	CSE(AI&ML)	2025	Null	Null	Null	Null	Null	Null	Null	Null	Null

Download CSV