Engineering Branches Analysis

NAME:	SUNNY SINGH
ROLL. NO (USERNAME):	20231CSE0095
EMAIL:	sunny762902@gmail.com
DATE:	17 – JULY - 2025
COLLEGE:	PRESIDENCY UNIVERSITY

CONTENTS

Problem Statement 3

Description.....4

Data & Insights.....5 - 6

Purpose.....7

Plan.....8 - 9

Implementation.....10 - 11

Code & Explanation.....12 - 25

Output, Analysis & Conclusion.....26 – 29

Bibliography.....30 - 31

Problem Statement

In the past few years, engineering education has changed a lot. This is because technology is moving fast, companies need new kinds of skills, and students are starting to choose different fields than before. Today, colleges offer more than a dozen engineering branches, and it's getting harder for students, teachers, and policymakers to figure out which ones are rising in demand, which are falling, and which are staying the same.

Even though we have a lot of raw data — like how many students are enrolling or getting placed in jobs — it's often hard to make sense of it. Most reports are just plain charts or numbers and don't help much when you're trying to answer questions like:

- Which engineering fields are becoming more popular over time?
- Which branches have the best job placement records?
- Are there any branches losing popularity or seeing fewer job offers?
- How can we show all this in a way that's clear and visually appealing?

Right now, we don't have easy-to-use tools that can turn this data into helpful insights. Because of this, students may choose branches without knowing the real trends, and colleges may struggle to update courses or make smart policy changes.

The Core Problem -

There is no intuitive, visual, and analytical platform available for stakeholders to explore the changing trends in engineering branch popularity and demand, based on real-world data from 2019 to 2024.

DESCRIPTION

This project, titled "Data Analytics on Engineering Branch Trends (2019–2024)", is a Flask-based web application designed to analyze, visualize, and interpret enrollment and placement data of engineering branches across multiple years. It offers a data-driven approach to understand which branches are thriving, declining, or stabilizing in popularity and industry demand.

What the Project Does :-

Using an Excel dataset (engineering_btech_trends_2019_2024.xlsx), the system performs the following key functions:

1. **Loads and Preprocesses Data:**
   Reads raw data from Excel, cleans the column names, and ensures correct data types for accurate plotting and analysis.
2. **Performs Analytical Calculations:**
   - Calculates branch-wise growth in student enrollment over time.
   - Identifies the most demanded branches based on placement averages.
   - Detects branches with declining trends using negative year-over-year changes.
   - Highlights least demanded branches by comparing average placements.
3. **Generates Dynamic Visualizations:**
   - Produces line charts and bar graphs using Matplotlib and Seaborn.
   - Plots are saved dynamically with unique filenames to avoid overwriting.
4. **Provides Clear Insights & Conclusions:**
   - Each graph is accompanied by a detailed, auto-generated explanation summarizing the trend.
   - Makes it easier for non-technical users to understand the data.
5. **Displays Results via a Web Interface:**
   - Simple, elegant, dark-themed UI built with HTML, CSS, and Flask templating (Jinja2).
   - Users can navigate from a menu page to explore each type of trend analysis individually.

Data & Insights

This project is driven by a comprehensive Excel dataset containing historical trends in engineering branch enrollments and placement performance from 2019 to 2024. The dataset forms the backbone for all visualizations and conclusions in the application.

Dataset Used :-

- File Name: engineering_btech_trends_2019_2024.xlsx
- Format: Excel (.xlsx)
- Structure:
    - Rows: Each row represents data for a specific year and branch.
    - Columns (example):
        - Year: Integer (e.g., 2019, 2020, ..., 2024)
        - Branch: String (e.g., Computer Science, Mechanical, Civil)
        - Total Students: Number of students enrolled
        - Placed Students: Number of students placed
        - Placement %: Calculated as (Placed / Total) × 100

Note: Column names are stripped of whitespaces during preprocessing to avoid errors.

Key Metrics Extracted : -

The following insights are calculated from the raw data:

1. **Branch Growth Over Years:**
    - The system compares student enrollment numbers across years to find the branch with the steepest upward trend.
    - It uses line plots to visually represent this growth.
2. **Most Demanded Branch:**
    - Based on average placement percentage, the branch with the highest average success rate is identified.

- Helps indicate what the industry is hiring the most.

3. **Least Demanded Branch:**
    - The branch with the lowest average placement rate over the years.
    - Useful to understand which branches are not aligning well with job market demand.
4. **Declining Branch:**
    - A branch showing a consistent decrease in student interest (enrollment) or placement rate.
    - Identified by analyzing negative trends in the time series.

Purpose

The primary purpose of this project is to provide a data-driven, user-friendly platform that helps students, educators, institutions, and policymakers understand the trends in engineering education across various branches from 2019 to 2024.

What Does It Aim to Solve?

1. **Branch Selection Confusion**:
    - Help students choose branches based on real-time demand and growth trends.
2. **Curriculum Improvement**:
    - Enable educators to focus on strengthening or revising underperforming departments.
3. **Policy Planning & Forecasting**:
    - Assist institutions and government bodies in making data-informed decisions on funding, infrastructure, and curriculum modernization.
4. **Transparent Visualization**:
    - Present data in a **clear, visual, and interactive** format so that it's easy to understand at a glance.

Core Goals :-

- Identify **Highest Growth**, **Most Demanded**, **Least Demanded**, and **Declining** engineering branches using real data.
- Show these insights on a web app built with **Flask** (a Python-based tool).
- Create **auto-named graphs** for each insight so that files don't get overwritten by mistake.
- Add smart, **auto-generated conclusions** under each chart to help users understand the meaning behind the numbers.
- Ensure device responsiveness and premium UI using **HTML + CSS**.

Plan

Step-by-Step Plan : -

1. **Dataset Preparation**
    - Collect the Excel file containing data from 2019 to 2024.
    - Clean the data: remove leading/trailing spaces, convert data types, and handle missing values.

2. **Set Up Flask Web Framework**
    - Use Flask to build a lightweight web interface.
    - Create routes for each type of analysis (e.g., /highest-growth, /most-demanded).

3. **Design a Homepage (Menu)**
    - Create an HTML page that serves as the main dashboard with four navigation buttons:
        - Highest Growth Branch
        - Most Demanded Branch
        - Least Demanded Branch
        - Declining Branch

4. **Data Analysis Logic**
    - For each analysis type, define a function that:
        - Filters or groups the data as needed.
        - Performs statistical operations (e.g., average, trend detection).
        - Generates insights and conclusions based on the result.

5. **Data Visualization**
    - Use Python libraries like matplotlib and seaborn to generate clear, attractive graphs.
    - Save plots with UUID-based filenames to ensure uniqueness.

6. **Dynamic Result Display**
    - Render the analysis results dynamically using HTML templates (.html files).
    - Display the graph and a textual conclusion in each result page.

7. **Styling**
    - Design a premium black & white theme using a custom CSS file.
    - Ensure the layout is mobile responsive with media queries.

8. **Error Handling**
    - Add basic error messages for file read errors, missing data, or route failures.

9. **Final Touches**
    - Add navigation (Back to Menu) on every page.
    - Ensure consistency across all pages (font, layout, image scaling).

Implementation

1) Backend Logic (main_app.py)

- Flask App Initialization:
  Sets up the Flask server with custom routes.
- Route Definitions:
  - / – Loads the main menu.
  - /highest-growth – Displays the branch with the highest trend growth.
  - /most-demanded – Shows the branch most demanded by industry.
  - /least-demanded – Shows the least demanded branch.
  - /declining – Predicts which branch may decline in the future.
- Dynamic Plot Cleanup:
  Each time a route is accessed, the **cleanup_static_folder(**) deletes previously generated plots using **glob.glob()** and **os.remove()** to avoid clutter.

2) Data Analysis Functions (engineering_analysis.py)

Each route calls one of the following:

- **load_data()** – Loads and cleans Excel data.
- **show_highest_growth_trend(**data**)** – Analyzes and plots trends over years.
- **show_most_demanded_branch(**data**)** – Finds the branch with maximum overall demand.
- **show_least_demanded_branch(**data**)** – Detects least chosen branches.
- **show_declining_branch(data) – Uses trendlines to guess future decline.**

All analysis functions:

- Use **pandas** for computation.

- Use **seaborn/matplotlib** to create plots.
- Save the output image with a unique filename using **uuid.uuid4().**

3) HTML Templates

◇ **menu.html**

- Acts as the **homepage**.
- Provides four buttons that link to each branch analysis.
- Uses the Jinja **url_for()** function to load the CSS file dynamically.

◇ **result.html**

- Dynamically displays:
  - Title
  - Image (if generated)
  - Textual insight or conclusion
  - "Back to Menu" button
- Makes use of **Jinja conditionals** to handle missing data gracefully.

4) CSS Styling (style.css)

This file implements a **premium light-black theme** with the following features:

- Uses **modern sans-serif fonts** for a clean and readable look.
- Centers everything on the page for **better focus and readability**
- Adds **rounded boxes**, **soft shadows**, and **hover effects** to give a smooth, modern feel.
- **Mobile-friendly** — looks great even on small screens (like phones under 600px wide).
- Stylish buttons for navigation

The .container class wraps each page section. Buttons are styled as **.menu-btn** and

.back-btn.

The conclusion text is placed inside .conclusion-box for emphasis.

Code & Explanation

➢ engineering_analysis.py

1. Importing Required Libraries

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import uuid
import os
```

EXPLANATION :-

- **pandas (pd)**: For reading and manipulating Excel data.
- **matplotlib.pyplot (plt)**: For plotting graphs.
- **seaborn (sns)**: For enhanced and aesthetic statistical plots.
- **uuid**: To generate unique filenames for saving plots.
- **os**: For file path operations (especially saving images in folders).

2. Data Loading Function

```
def load_data():
    try:
        df = pd.read_excel('engineering_btech_trends_2019_2024.xlsx')
        df.columns = df.columns.str.strip()
        df['Year'] = df['Year'].astype(int)
        sns.set(style="whitegrid")
        return df
    except Exception as e:
        print("Error loading data:", e)
        return None
```

EXPLANATION :-

- **Reads Excel** file: 'engineering_btech_trends_2019_2024.xlsx'.
- **Cleans column names** by removing leading/trailing spaces.
- **Ensures** the Year column is in **integer format**.
- Sets **Seaborn style** to "whitegrid" for better plot appearance.
- **Error handling** is added to gracefully catch file or format issues.

3. Save Plot Function

```python
def save_plot():
    filename = f"plot_{uuid.uuid4().hex}.png"
    path = os.path.join("static", filename)
    plt.savefig(path)
    plt.close()
    return filename
```

EXPLANATION :-

- **Generates a unique filename** using uuid (e.g., plot_a1b2c3d4e5f6.png).
- Saves the plot into a **static/ folder**.
- **Closes the plot** to free memory.
- **Returns the filename** so it can be used in Flask to display the plot.

4. Highest Growth Trend Function

```python
def show_highest_growth_trend(df):
    pivot_df = df.pivot(index='Year', columns='Branch', values='Enrollment').astype(int)
    growth = pivot_df.loc[df['Year'].max()] - pivot_df.loc[df['Year'].min()]
    top_branch = growth.idxmax()

    plt.figure(figsize=(12, 7))
    for branch in pivot_df.columns:
        if branch == top_branch:
            plt.plot(pivot_df.index, pivot_df[branch], marker='o', linestyle='--', linewidth=3,
                     label=f"{branch} (Highest Growth)", color='green')
        else:
            plt.plot(pivot_df.index, pivot_df[branch], marker='o', label=branch, alpha=0.6)

    plt.title(f"Engineering Branch Enrollment Trends ({df['Year'].min()}–{df['Year'].max()})", fontsize=14)
    plt.xlabel("Year")
    plt.ylabel("Enrollment Numbers")
    plt.grid(True)
    plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left')
    plt.ticklabel_format(style='plain', axis='y')
    plt.tight_layout()

    conclusion_text = (
        f"{top_branch} Engineering has shown the highest growth trend in recent years (2019–2024), indicating
        increasing industry relevance.\n"
        f"This growth suggests that {top_branch} is becoming a top choice among engineering aspirants, possibl
        technological demands and attractive career opportunities in this domain."
    )

    filename = save_plot()
    return filename, conclusion_text
```

EXPLANATION :-

- It compares **enrollments** from the **first year** to the **last**.
- Finds the branch with the **highest growth**.

- Creates a **line graph** showing trends for all branches.
- Highlights the **fastest-growing branch** in **green**.
- Adds **labels, title**, and a **summary** explaining the result.
- Saves the graph as an image for later use.

5. Most Demanded Branch Function

```
def show_most_demanded_branch(df):
    branch_demand = df.groupby("Branch")["Students Placed"].mean().reset_index()
    most_demand_branch = branch_demand.loc[branch_demand["Students Placed"].idxmax()]

    plt.figure(figsize=(12, 7))
    sns.barplot(data=branch_demand, x="Branch", y="Students Placed", palette='Greens')
    plt.title("Most Demanded Branch (2019-2024)", fontsize=16)
    plt.axhline(most_demand_branch["Students Placed"], color='black', linestyle='--',
                label=f"Most Demanded: {most_demand_branch['Branch']}")
    plt.xticks(rotation=45)
    plt.legend()
    plt.tight_layout()

    conclusion_text = (
        f"{most_demand_branch['Branch']} is currently the most in-demand engineering branch i
        student placements between 2019 and 2024.\n"
        f"Its clear lead in placement numbers indicates strong and growing industry demand, l
        sectors, software roles, and digital transformation across industries."
    )

    filename = save_plot()
    return filename, conclusion_text
```

EXPLANATION :-

- It calculates the **average placements** for each branch.
- Identifies the **branch with the highest average**.
- Creates a **bar graph** showing placement averages.
- **Highlights** the top branch with a dotted line.
- Adds a short **conclusion** explaining why it's in demand.
- **Saves the graph** as an image.

6. Least Demanded Branch Function

```
def show_least_demanded_branch(df):
    branch_demand = df.groupby("Branch")["Students Placed"].mean().reset_index()
    least_demand_branch = branch_demand.loc[branch_demand["Students Placed"].idxmin()]

    plt.figure(figsize=(12, 7))
    sns.barplot(data=branch_demand, x="Branch", y="Students Placed", palette='Reds')
    plt.title("Least Demanded Branch (2019-2024)", fontsize=16)
    plt.axhline(least_demand_branch["Students Placed"], color='black', linestyle='--',
                label=f"Least Demanded: {least_demand_branch['Branch']}")
    plt.xticks(rotation=45)
    plt.legend()
    plt.tight_layout()

    conclusion_text = (
        f"{least_demand_branch['Branch']} is currently facing the least demand in the indus
        \n"
        f"Its low placement figures suggest limited opportunities and decreasing relevance
    )

    filename = save_plot()
    return filename, conclusion_text
```

EXPLANATION :-

- **Group data by branch**, calculate **average students placed**.
- Identify the **branch with lowest placement** using .idxmin().
- **Barplot** with red colour scheme.
- **Highlight least demanded branch** with a line.
- **Return plot filename and interpretation**.

7. Declining Branch Detection Function

```python
def show_declining_branch(df):
    df['Enrollment'] = pd.to_numeric(df['Enrollment'], errors='coerce').fillna(0).astype(int)
    pivot_df = df.pivot(index='Year', columns='Branch', values='Enrollment').sort_index()
    yearly_change = pivot_df.diff()
    negative_counts = (yearly_change < 0).sum()
    most_declining_branch = negative_counts.idxmax()

    plt.figure(figsize=(12, 7))
    for branch in pivot_df.columns:
        if branch == most_declining_branch:
            plt.plot(pivot_df.index, pivot_df[branch],
                     label=f"{branch} (Most Likely to Decline)", color='red', linewidth=3, linestyle='--', marker='o')
        else:
            plt.plot(pivot_df.index, pivot_df[branch], label=branch, linestyle='-', alpha=0.5)

    plt.title("Branch Most Likely to Decline", fontsize=14)
    plt.xlabel("Year")
    plt.ylabel("Enrollment Count")
    plt.grid(True)
    plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left')
    plt.ticklabel_format(style='plain', axis='y')
    plt.tight_layout()

    conclusion_text = (
        f"{most_declining_branch} Engineering has experienced the most consistent drop in enrollment from {df['Year'].m
        ['Year'].max()}.\n"
        f"This suggests a steady decline in popularity or demand for this branch in recent years."
    )

    filename = save_plot()
    return filename, conclusion_text
```

EXPLANATION :-

- It checks how **enrollment** changed **year by year** for each branch.
- **Counts** how many times each branch had a **drop in enrollment.**
- Picks the branch with the **most frequent drops.**
- Plots a **line graph** showing trends for all branches.
- Highlights the **declining branch** in **red**.
- Adds a summary explaining the drop in interest.
- **Saves** the **graph** as an image.

➢ main_app.py

1. Importing Required Libraries

```
from flask import Flask, render_template, url_for
import pandas as pd
import os
import glob
```

EXPLANATION :-

- **Flask:** Used to create a web application.
- **render_template:** Loads and displays HTML templates (like menu.html or result.html).
- **url_for:** Generates URLs for static files.
- **pandas as pd:** Used to handle Excel or CSV data (your trend dataset).
- os**, glob:** Used for file and folder handling, especially to clean up old plot images.

2. Importing Custom Functions

```
from engineering_analysis import (
    load_data,
    show_highest_growth_trend,
    show_most_demanded_branch,
    show_least_demanded_branch,
    show_declining_branch
)
```

EXPLANATION :-

1. This imports **pre-defined functions** from another Python file called engineering_analysis.py.

2. These functions do the following:

    a. **load_data():** Loads the Excel file.

    b. **show_highest_growth_trend(data):** Analyzes and returns a plot + summary for the branch with highest growth.

    c. Similar logic applies for **most_demanded**, **least_demanded**, and **declining branches.**

3. Flask App Initialization

```
app = Flask(__name__)
```

EXPLANATION :-

- Initializes your Flask web app.

4. Static Folder Cleanup

```
def cleanup_static_folder():
    """Delete old generated plots to avoid clutter."""
    for file in glob.glob("static/plot_*.png"):
        try:
            os.remove(file)
        except Exception as e:
            print(f"Error deleting file {file}: {e}")
```

EXPLANATION :-

- This function **deletes all old plot images** from the static folder starting with plot_.
- Prevents clutter and ensures fresh plots every time.

5. Home Page Route – Menu

```
@app.route('/')
def menu():
    return render_template('menu.html')
```

EXPLANATION :-

- When a user visits the **home page (/)**, it renders the **menu.html template**.
- This page typically shows buttons or links to choose the analysis type (growth, demand, etc.).

6. Highest Growth Branch

```
@app.route('/highest-growth')
def highest_growth():
    cleanup_static_folder()
    data = load_data()
    if data is None:
        return "Data loading failed.", 500
    image, conclusion = show_highest_growth_trend(data)
    return render_template('result.html', title="Highest Growth Branch", image=image, conclusion=conclusion)
```

EXPLANATION :-

- It sets up a webpage that opens when you visit /highest-growth in the browser.
- Before doing anything, it deletes any old graph images from the "static" folder to keep things clean.
- It then loads the engineering data (like enrollment numbers across years).
- If the data can't be loaded, it shows an error message.
- If the data loads successfully, it analyzes which engineering branch had the **highest growth in enrollment** over the years.
- A graph is created that shows this growth trend clearly.
- The graph highlights the fastest-growing branch using a different color and style.
- It also writes a short explanation (conclusion) about why this branch is growing.
- Finally, it shows a result page with the graph, a title ("Highest Growth Branch"), and the conclusion text.

7. Most Demanded Branch

```
@app.route('/most-demanded')
def most_demanded():
    cleanup_static_folder()
    data = load_data()
    if data is None:
        return "Data loading failed.", 500
    image, conclusion = show_most_demanded_branch(data)
    return render_template('result.html', title="Most Demanded Branch", image=image, conclusion=conclusion)
```

EXPLANATION :-

- It creates a webpage that opens when you visit (**/most-demanded)**.
- First, it **cleans up old graph images** from the folder to avoid clutter.
- Then, it **loads the engineering data** (like student placements).
- If data loading fails, it shows an **error message** on the page.
- If the data is loaded correctly, it finds the branch with the **highest average student placements** from 2019 to 2024.

- A **bar graph** is created showing placement stats for all branches.
- The **most demanded branch** (with the most placements) is highlighted.
- A **conclusion is written** explaining why this branch is the most in demand.
- Finally, it shows a result page with the graph, a title ("Most Demanded Branch "), and the conclusion text.

8. Least Demanded Branch

```python
@app.route('/least-demanded')
def least_demanded():
    cleanup_static_folder()
    data = load_data()
    if data is None:
        return "Data loading failed.", 500
    image, conclusion = show_least_demanded_branch(data)
    return render_template('result.html', title="Least Demanded Branch", image=image, conclusion=conclusion)
```

EXPLANATION :-

- It sets up a webpage that runs when you visit (**/least-demanded)** in your browser.
- It starts by **deleting old graph images** to keep the folder clean.
- It **loads engineering data** (like student placement numbers).
- If data loading fails, it shows an **error message** on the page.
- If the data loads successfully, it finds the branch with the **lowest average student placements** from 2019 to 2024.
- It creates a **bar graph** showing the placement numbers for all branches.
- The branch with the **lowest demand** is clearly **highlighted** in the graph.
- A **conclusion is written** explaining why this branch is currently the least in demand.
- Finally, it shows a result page with the graph, a title ("Least Demanded Branch "), and the conclusion text.

9. Declining Branch

```python
@app.route('/declining')
def declining():
    cleanup_static_folder()
    data = load_data()
    if data is None:
        return "Data loading failed.", 500
    image, conclusion = show_declining_branch(data)
    return render_template('result.html', title="Branch Most Likely to Decline", image=image, conclusion=conclusion)
```

EXPLANATION :-

- It sets up a webpage that runs when you visit **/declining** in your browser.
- It begins by **cleaning old graph images** from the folder to avoid saving too many files.
- It then **loads the engineering data**, which includes yearly enrollment numbers.
- If the data fails to load, it shows an **error message** on the page.
- If the data loads correctly, it analyzes the enrollment trends to find the branch that has had the **most frequent drops** in enrollment over the years.
- It generates a **line graph** showing how enrollment has changed for all branches.
- The branch with the most consistent declines is **highlighted** in red.
- A **conclusion is prepared**, explaining why that branch may be losing popularity or demand.
- Finally, it shows a result page with the graph, a title ("Least Demanded Branch "), and the conclusion text.

10. Running the Flask App

```python
if __name__ == '__main__':
    cleanup_static_folder()
    app.run(debug=True, port=5762)
```

EXPLANATION :-

- This line runs **only when the Python file is executed directly**, not when imported as a module.
- It first **cleans up old graph images** by calling cleanup_static_folder().
- Then it **starts the Flask web app** using app.run().
- The app runs in **debug mode**, which helps during development by showing errors clearly.
- It uses **port 5762**, so you can open the app in your browser at http://localhost:5762.

➢ result.html

```
<!DOCTYPE html>
<html>
<head>
    <title>{{ title }}</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='style.css')
}}">
</head>
<body>
    <div class="container">
        <h2>{{ title }}</h2>
        {% if image %}
            <img src="{{ url_for('static', filename=image) }}" alt="Graph">
        {% else %}
            <p class="no-image">No image to display.</p>
        {% endif %}

        {% if conclusion %}
        <div class="conclusion-box">
            <h3>Conclusion</h3>
            <p>{{ conclusion }}</p>
        </div>
        {% endif %}

        <a href="/" class="back-btn">← Back to Menu</a>
    </div>
</body>
</html>
```

EXPLANATION :-

1. **Starts a Webpage**
    - It begins by telling the browser that this is an HTML5 document.
    - It sets up the structure for a webpage using standard HTML elements like <head>, <body>, and so on.
2. **Dynamically Sets the Page Title**
    - The code includes a placeholder for a page title (like "Student Report" or "Branch Analysis").
    - This title is filled in automatically when the page is loaded, depending on the data passed in from the backend (like Flask in Python).
3. **Applies a Custom Design (CSS)**
    - A separate styling file is linked to make the page look good (e.g., setting font sizes, colors, layouts).

- This design file is stored in the "static" folder of the project and automatically linked.

4. **Creates a Main Content Box**
    - Inside the webpage, a main container is used to hold everything neatly in the center of the screen.
    - The title is shown again inside the page as a large heading.
5. **Displays an Image (If Available)**
    - If there's an image file provided (like a graph or chart), it shows that image inside the container.
    - If no image is available, it shows a simple message: "No image to display."
6. **Shows a Conclusion Section (Optional)**
    - If a conclusion is provided (like an insight or summary), the code displays it inside a highlighted box.
    - If no conclusion is passed in, this part stays hidden.
7. **Provides a Back Button**
    - At the bottom, there's a button that lets users return to the home page or main menu.
    - It looks like a normal clickable link but styled like a button.

➢ menu.html

```
<!DOCTYPE html>
<html>
<head>
    <title>Engineering Branch Analysis</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='style.css')
}}">
</head>
<body>
    <div class="container">
        <h1>Engineering Branch Analysis</h1>
        <a href="/highest-growth" class="menu-btn">Highest Growth Branch</a>
        <a href="/most-demanded" class="menu-btn">Most Demanded Branch</a>
        <a href="/least-demanded" class="menu-btn">Least Demanded Branch</a>
        <a href="/declining" class="menu-btn">Declining Branch</a>
    </div>
</body>
</html>
```

EXPLANATION :-

1. **Sets the Page Title**
   - The name at the top of the browser tab is set to: “Engineering Branch Analysis”.
2. **Adds a Style File**
   - It loads a design (CSS) file to make everything look nice — like button colors, fonts, layout, etc.
3. **Shows the Main Heading**
   - On the webpage, you’ll see a big title that says: “Engineering Branch Analysis”.
4. **Displays Four Buttons**
   - The page shows four clickable buttons:
     - **Highest Growth Branch** — takes you to a page that shows which branch is growing the fastest.
     - **Most Demanded Branch —** shows which branch has the highest demand.
     - **Least Demanded Branch —** shows the one with the lowest interest or demand.
     - **Declining Branch —** shows which branch is dropping in popularity or jobs.

➢ Style.css

```
body {
    font-family: 'Segoe UI', sans-serif;
    background-color: #1e1e1e;
    color: #f2f2f2;
    margin: 0;
    padding-top: 50px;
    text-align: center;
}
```

EXPLANATION :-

- Sets a dark background and light text.
- Uses a modern font (Segoe UI).

- Removes default margin and adds padding on top.
- Centers all text inside the body.

```
.container {
    max-width: 900px;
    margin: auto;
    background: #2b2b2b;
    padding: 40px;
    border-radius: 16px;
    box-shadow: 0 0 25px rgba(0,0,0,0.4);
}
```

EXPLANATION :-

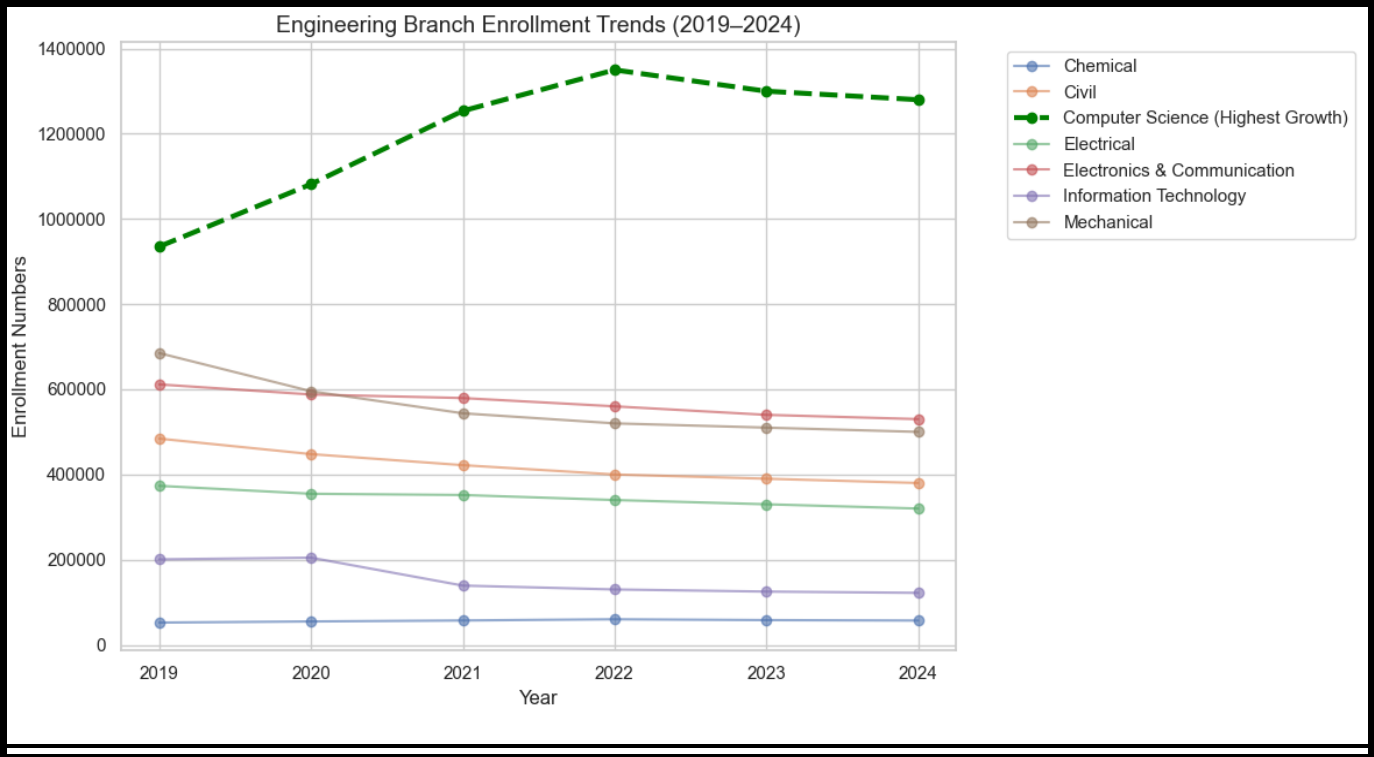
- Creates a centered box with a width up to 900px.
- Has a dark gray background and smooth rounded corners.
- Adds padding inside and shadow outside to make it look elevated.

```
h1, h2, h3 {
    color: #ffffff;
    margin-bottom: 20px;
    font-weight: 600;
}
•
.menu-btn, .back-btn {
    display: block;
    margin: 15px auto;
    padding: 14px 24px;
    width: 85%;
    font-size: 16px;
    font-weight: 500;
    background-color: #444;
    color: #fff;
    text-decoration: none;
    border-radius: 10px;
    transition: background-color 0.3s ease, transform 0.2s ease;
    border: none;
    box-shadow: 0 4px 10px rgba(0,0,0,0.2);
}

.menu-btn:hover, .back-btn:hover {
    background-color: #5c5c5c;
    transform: translateY(-2px);
}
```

```
img {
    margin-top: 20px;
    max-width: 100%;
    height: auto;
    border-radius: 12px;
    border: 1px solid #3a3a3a;
}

.conclusion-box {
    margin-top: 30px;
    text-align: left;
    padding: 20px;
    background: #3a3a3a;
    border-left: 5px solid #999;
    border-radius: 12px;
    max-width: 95%;
    margin-left: auto;
    margin-right: auto;
    font-size: 17px;
    line-height: 1.6;
    color: #e0e0e0;
}

.no-image {
    color: #888;
    font-style: italic;
}

@media (max-width: 600px) {
    .container {
        width: 95%;
        padding: 25px;
    }

    .menu-btn, .back-btn {
        width: 100%;
        font-size: 14px;
        padding: 12px;
    }

    h1, h2 {
        font-size: 22px;
    }
}
```

Output, Analysis & Conclusion

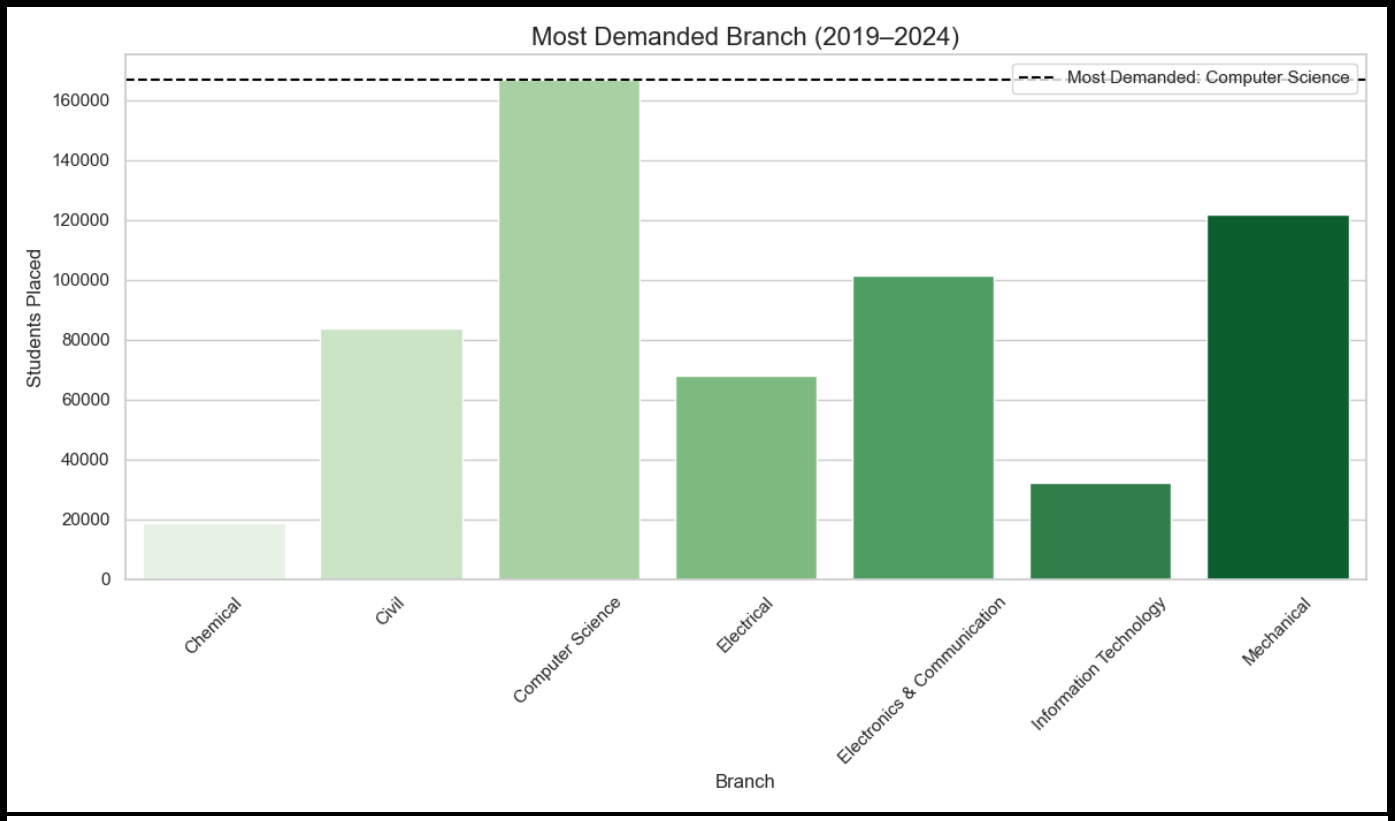
1. **Engineering Branch that has Highest Growth Trends :-**



- **Analysis:**
  - The line graph shows enrollment trends for various engineering branches from 2019 to 2024. **Computer Science** shows a clear upward trend in enrollment, peaking around 2022 and remaining high afterward.
  - In contrast, branches like **Civil**, **Electrical**, and **Mechanical** show a steady decline in student enrollment over the same period.
  - **Electronics & Communication** and **Information Technology** also show slight declines or remain mostly flat.

- **Conclusion:**
  - **Computer Science Engineering** has shown the **highest growth trend** in recent years (2019–2024), indicating rising student interest and industry relevance. Other branches, especially **Civil** and **Electrical**, appear to be declining in popularity.

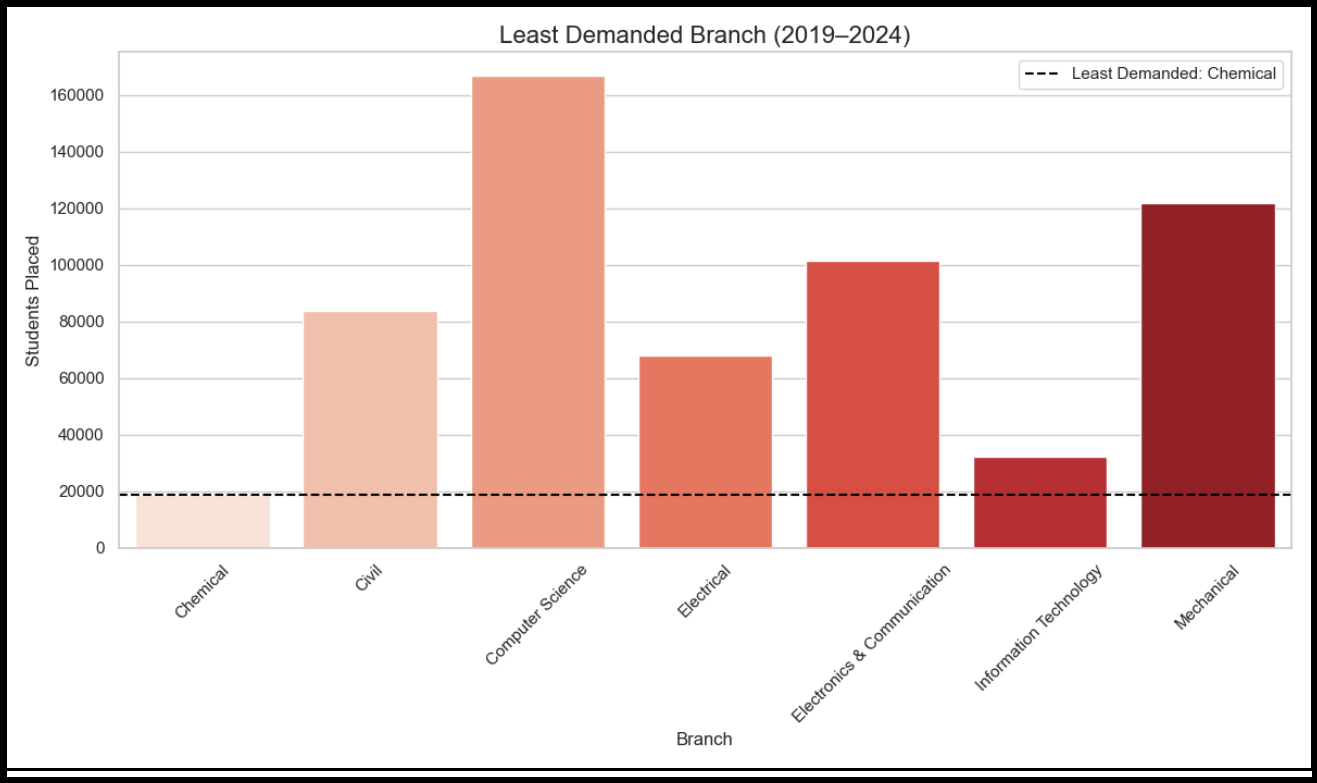
2. **Most Demanded Engineering Branch**:-



- **Analysis:**
  - The chart shows the number of students placed from each engineering branch between 2019 and 2024. **Computer Science** has the highest placements, followed by Mechanical and Electronics & Communication.
  - Branches like Chemical, Civil, and Information Technology have **much lower placement numbers**.

- **Conclusion:**
  - **Computer Science** is **the most in-demand branch** in the industry based on placement data from 2019 to 2024.

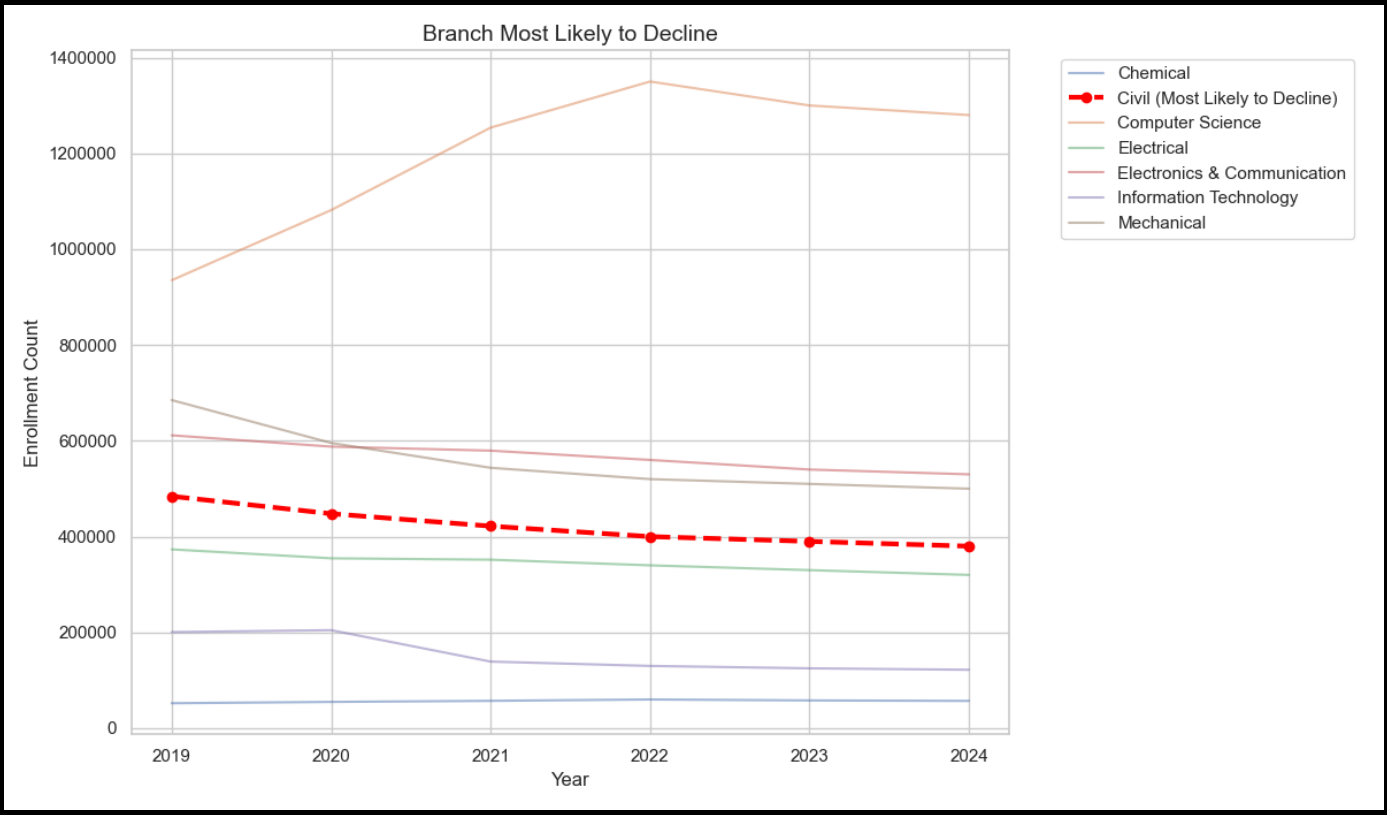
3. **Least Demanded Engineering Branch**:-



- **Analysis:**
  - From the bar chart showing student placements (2019–2024), **Chemical Engineering** has the **lowest number of students placed**, clearly marked as the **least demanded branch**.
  - While branches like Computer Science, Mechanical, and Electronics & Communication show strong placement numbers, Chemical Engineering is significantly lower, with fewer than 20,000 students placed.
  - Other low-performing branches include Information Technology, but even it has higher placements than Chemical.

- **Conclusion:**
  - **Chemical Engineering** is currently facing the **least demand** in the industry based on **placement data** from 2019 to 2024. Its low placement figures suggest **limited opportunities** and **decreasing relevance in the job market**.

4. Engineering Branch that Most Likely to Decline :-



- **Analysis:**
  - All branches **except Computer Science** show **falling enrollment** from 2019 to 2024.
  - **Civil Engineering drops steadily** from about **480,000** to **380,000 students**.
  - While Mechanical and Electronics & Communication also decline, their decreases are less uniform and start from much higher levels.
  - Smaller drops also occur in Electrical, Information Technology, and Chemical Engineering.

- **Conclusion:**

  Civil Engineering is **most likely to decline** in demand in the **near future**, given its **continuous and significant drop in enrollment** over recent years.

Bibliography / References

1. **Pandas – Python Data Analysis Library**
   - **Source:** https://pandas.pydata.org/
   - **Purpose**: For reading, cleaning, and analyzing Excel data (enrollment and placement trends).
2. **Matplotlib – Python Plotting Library**
   - **Source:** https://matplotlib.org/stable/index.html
   - **Purpose:** To generate line charts and save them as PNG files.
3. **Seaborn – Statistical Data Visualization Library**
   - **Source:** https://seaborn.pydata.org/
   - **Purpose:** For visually appealing bar plots and styling.
4. **Flask – Python Web Framework**
   - **Source:** https://flask.palletsprojects.com/
   - **Purpose:** To build the web interface and create routes for visualizing the analysis.
5. **Jinja2 – Templating Engine for Python/Flask**
   - **Source:** https://jinja.palletsprojects.com/
   - **Purpose:** To dynamically render HTML templates with data (e.g., titles, plots, and conclusions).
6. **UUID – Python Built-in Library**
   - **Source:** https://docs.python.org/3/library/uuid.html
   - **Purpose:** To generate unique filenames for each saved graph.

7. **HTML5 & CSS3 (Manual Design)**

- **Sources:**
    - https://developer.mozilla.org/en-US/docs/Web/HTML
    - https://developer.mozilla.org/en-US/docs/Web/CSS
- **Purpose:** To design the front-end user interface and ensure mobile responsiveness.

Additional Learning References :-

1. Real Python - https://realpython.com/
    - For Flask tutorials and tips on organizing Python web applications.
2. Stack Overflow
    - For error resolution and understanding delicate behaviors in Pandas and Matplotlib.