Technical documentation of my project

1.Main.py:

This Python script uses the Flask framework to create a web application. It connects to a MySQL database using the Flask-MySQLdb extension.

The script first sets up the Flask application and configures the database connection details. It then initializes the MySQL connection.

The home() function is mapped to the root URL ("/") of the web application. When this URL is accessed, the function executes several SQL queries to count the number of students for different student population codes ('ISM', 'CS', 'DSA', 'AIs', 'SE') for the years 2021 and 2020.

Each query is executed using a separate cursor, which fetches the results and then closes. The results of these queries are stored in variables data, data2, ..., data10.

Finally, the function renders an HTML template ("index.html") and passes the data from the queries to this template. The template can then use this data to dynamically generate the webpage.

Also, the Python code uses Flask, to create several HTTP endpoints. Each endpoint corresponds to a different route and is associated with a function that will be executed when the route is accessed. The methods GET and POST are allowed for all routes. An example is :

@app.route('/population/ISM', methods=['GET', 'POST']): This route fetches the first name, last name, email, and the count of courses passed (with a score of 10 or more) for students in the 'ISM' population for a given year. The year is passed as a query parameter.

@app.route('/population/ISM_courses', methods=['GET', 'POST']): This route fetches the course reference and the count of sessions for each course for the 'ISM' population for a given year.

2.Index.html

This is an HTML document that displays a webpage with charts and data related to a Computer Science program.

The <head> section includes metadata about the webpage, a link to the Chart.js library, and CSS styles for the page.

The <body> section contains the main content of the webpage. It starts with a welcome message, followed by two sections displaying data for the years 2021 and 2020. Each

section has a list of Computer Science courses with their respective populations, and a chart visualizing this data. The charts are created using the Chart.js library.

The onclick attribute on the elements is used to call JavaScript functions when the user clicks on them. These functions redirect the user to different pages based on the course and year.

The last script block in the <body> section is a function that updates the text content of a <p> element with the id lastModified every second, displaying the current date.

The {{data[n][0]["count(*)"]}} syntax suggests that this HTML is intended to be used with a templating engine, which will replace these placeholders with actual data.

3.CS, AIs, DSA, ISM, SE html templates

This is an HTML document that displays two tables: one for students and one for courses.

The <head> section includes meta information and CSS styles for the page and tables.

The <body> contains a div with the class "table-container", which contains two divs. Each div contains a table: the first table (id "data-table") displays student information, and the second table (id "data-table2") displays course information.

A JavaScript function getUrlParameter(year) is defined to extract the value of a URL parameter given its name.

4.Student.html:

This is an HTML document that displays a table of student grades per course.

The <head> section includes meta information and CSS styles for the table.

The <body> contains a heading and a table with an id of "data-table". The table has five headers: email, first name, last name, Course, and Grade. The table body is initially empty.

A JavaScript function getUrlParameter(email) is defined to extract the value of a URL parameter given its name. When the document is fully loaded (DOMContentLoaded event), an asynchronous function is executed. This function:

- Selects the table body element.
- Gets the 'email' URL parameter.
- Makes a fetch request to the "/student" endpoint with the email as a query parameter.
- If the request is successful, it converts the response to JSON.

- For each item in the JSON data, it creates a new table row and appends it to the table body. Each row contains the student's email, first name, last name, course code, and grade.
- If there's an error during the fetch request, it logs the error to the console.