STUDENT MANAGEMENT SYSTEM – AZARCON / CORPUZ

1. Project Overview

The Student Management System is a full-stack web application made to manage student info, subject enrollments, and grade records in a simple and organized way. Instead of focusing too much on UI design, the main goal of this project is to make the user experience more practical—helping reduce mistakes and making the data easier to handle.

The backend was built using Django along with Django REST Framework, which handles the RESTful API that supports all CRUD operations for students, subjects, enrollments, and their grades (activities, quizzes, exams). On the frontend side, it's just plain HTML, CSS, and JavaScript that fetches and displays data from the API. It might not look super fancy, but it's designed to work well and be easy to understand and use.

2. Features Implemented

a. Student Management

- Create, update, and delete student records
- View individual student profiles with all enrolled subjects and grades
- Straightforward layout that prioritizes usability and accuracy

b. Subject Management

- Add and manage subjects with a unique subject code and predefined grading weights
- Grading weights (quiz, activity, exam) are locked once the subject is created to ensure consistent grade calculations across students

c. Enrollment System

- Enroll students into specific subjects using a section-based filter
- Student dropdown auto-fills and locks the corresponding section to prevent mismatched data
- Prevents duplicate enrollments by disabling the submit button if the student is already enrolled in the selected subject
- Fully dynamic form powered by API responses (no hardcoded dropdowns)

d. Grades Management

- Add, edit, and delete individual grade entries for quizzes, activities, and exams
- Automatically creates grade placeholders for all students enrolled in a subject when a new grade (e.g. "Quiz 1") is added
- Validates inputs so no one can enter a score higher than the defined max
- Calculates the student's total grade per subject using the subject's locked weight distribution
- Shows grade category distribution (not actual scores) as a pie chart to help visualize how much each type contributes to the final grade

e. REST API Integration

- All create/read/update/delete operations use the Django REST Framework
- Frontend fetches and submits data using JavaScript's fetch() with zero reliance on Django templates
- API handles all backend logic while the frontend dynamically renders everything using plain JS and DOM manipulation

3. Technologies Used

- **Backend:** Django, Django REST Framework
- **Frontend:** HTML, CSS, JavaScript (vanilla / no framework used)
- **Database:** SQLite (for local development), PostgreSQL (used in deployment)
- **API:** RESTful API between frontend and backend
- **Version Control:** Git, GitHub
- **Deployment:** Render (for hosting backend and database)
- **Environment & Tools:** Python virtual environment, Windows CMD, VSCode

4. Partner Contributions

a. Gerald Harry C. Azarcon

- Worked on the CSS and overall design of the pages.
- Help with layout and making sure the interface looks clean and easy to use.
- Also did testing and gave feedback to improve the functionality.

b. Chris Joven R. Corpuz

- Built the backend with Django and Django REST Framework, made the API endpoints.
- Created the frontend using plain JavaScript to connect with the API and display data.
- Added features like grade calculations, section-based student filtering, and pie charts for grades.
- Deployed the project to Render and set up the environment.
- Did minor CSS tweaks to polish the UI.
- Handle error handling and form validations for better UX.