Assignment: Build a Flask-based Student Platform

We'd like you to build a **Flask application** (Python) that implements a simple **Student Platform** with two modules: **Students** and **Applications**.

---

1. Modules & Data Models

Students Module

Each student should have the following fields:

- **Name** (string, required)
- **Email** (string, required)
- **Phone** (string, required)
- **Highest Intake** (string)
- **Highest Status** (string)

Applications Module

Each application belongs to a student and should have the following fields:

- **University Name** (string, required)
- **Program Name** (string, required)
- **Intake** (string, required, format: `Jan 2026`, `Feb 2026`, etc.)
- **Status** (string, required) – possible values:
    1. Building Application
    2. Application Submitted to University
    3. Offer Received
    4. Offer Accepted by Student
    5. Visa Approved
    6. Dropped

Note: The statuses have a weightage in the above order, where **Visa Approved** is the highest.

---

2. Logic for Highest Status & Highest Intake

- Whenever an **application is created or updated**, recalculate the student's:
    - **Highest Status** → based on the maximum weightage among all applications.
    - **Highest Intake** → taken from the intake of the application that has the highest status.

Example:

- A student has 5 applications.
- If one of them has a status of "Offer Accepted by Student" (higher than others), then the student's **highest status** is also "Offer Accepted by Student".
- The **Highest Intake** = the intake value of that same application.
- If two applications have the same highest status, then the application with the closest intake will be considered the highest. For example, a student has two applications, one for the Jan 2026 Intake and the other for the Sep 2026 intake, both in the 'Offer Received' status. In this case, the Highest Intake will be Jan 2026.

---

3. APIs to Implement

1. **CRUD operations for Students**
   - Create, Read, Update, and Delete student records.
2. **CRUD operations for Application**
   - Create, Read, Update, and Delete applications for a given student.
3. **API to fetch the Highest Status & Highest Intake for a student**
   - Input: Student ID
   - Output: `{ "highest_status": "...", "highest_intake": "..." }`
4.

---

4. Technical Requirements

- **Framework**: Flask (Python)
- **Database**: PostgreSQL or MongoDB (your choice)
- Properly structure your project (models, routes, services).
- Ensure all the mentioned fields are **mandatory** (except `Highest Intake` in Students).
- Provide **sample API request/response examples** in your submission (e.g., via Postman collection or README).
- Implement Docker setup also.
- Add test cases properly
- Add API documentation
- The complete project should be production-ready.

Deliverables

1. Flask project source code (on GitHub or zip).
2. README with setup instructions.
3. Postman collection (mandatory).
4. Test Cases (mandatory)
5. The project should be production-ready.