**Project overview**

**Project Components**

1. **Flask Web Application**: Flask serves as the backend framework for handling the form. The form collects user inputs (e.g., name, email, department, etc.), and upon submission, Flask processes the form data and forwards it to Neo4j for storage.
   * **Key Flask Features**:
     + Handles HTTP requests to display the form (GET) and process form submissions (POST).
     + Uses render\_template() or render\_template\_string() to serve the form’s HTML.
     + Extracts form data from request.form and passes it to the database interaction logic.
2. **HTML Form**: The form is a simple HTML interface that collects basic user data like:
   * Name
   * Registration Number
   * Department
   * Email
   * Address
   * Gender (with radio buttons)

Users submit the form, and Flask processes the data.

1. **Neo4j Database**: Neo4j is used to store the form data in the form of nodes and relationships. Each form submission creates a "Student" node with the relevant properties:
   * **Nodes**:
     + Student: Contains properties such as name, regno, department, email, address, and gender.
   * **Relationships**: While the basic implementation may not require relationships, they can be added for more complex data models (e.g., relationships to college, skills, etc.).
2. **Neo4j Driver for Python**: The Neo4j Python driver (neo4j) facilitates the connection between the Flask application and the Neo4j database. It enables creating sessions and running queries to insert nodes and relationships into the graph database.

**Data Flow**

1. **Form Submission**:
   * The user accesses the form at the root URL (/).
   * The form collects user data and sends it to the Flask backend using a POST request upon submission.
2. **Flask Backend**:
   * Flask receives the form data and extracts individual fields.
   * A function is used to insert the form data into the Neo4j database. Each submission creates a new Student node.
3. **Neo4j Storage**:
   * Data is stored as nodes in Neo4j with properties like name, regno, department, etc.
   * If relationships are defined, additional queries can create connections between Student nodes and other nodes (such as College or Year of Passout).