|  |
| --- |
| 1. Creating Database   import sqlite3  conn = sqlite3.connect('students.db')  c = conn.cursor()  c.execute('''  CREATE TABLE students (  student\_id INTEGER PRIMARY KEY,  first\_name TEXT,  last\_name TEXT,  dob TEXT,  amount\_due REAL  )  ''')  conn.commit()  conn.close() |
| 2. Creating Basic Flask App  from flask import Flask  app = Flask(\_\_name\_\_)  @app.route('/')  def hello\_world():  return 'Hello, world!'  if \_\_name\_\_ == '\_\_main\_\_':  app.run(debug=True)  Graphical user interface, text, application, chat or text message, website  Description automatically generated |
| For Create  from flask import request, jsonify  @app.route('/students', methods=['POST'])  def add\_student():  data = request.get\_json()  new\_student = Student(  first\_name=data['first\_name'],  last\_name=data['last\_name'],  dob=data['dob'],  amount\_due=data['amount\_due']  )  db.session.add(new\_student)  db.session.commit()  return jsonify({'message': 'Student added successfully'}) |
| For Read  @app.route('/students/<int:id>', methods=['GET'])  def get\_student(id):  student = Student.query.filter\_by(student\_id=id).first()  if student is None:  return jsonify({'message': 'Student not found'})  else:  return jsonify({  'student\_id': student.student\_id,  'first\_name': student.first\_name,  'last\_name': student.last\_name,  'dob': student.dob,  'amount\_due': student.amount\_due  }) |
| For Update  @app.route('/students/<int:id>', methods=['PUT'])  def update\_student(id):  student = Student.query.filter\_by(student\_id=id).first()  if student is None:  return jsonify({'message': 'Student not found'})  else:  data = request.get\_json()  student.first\_name = data['first\_name']  student.last\_name = data['last\_name']  student.dob = data['dob']  student.amount\_due = data['amount\_due']  db.session.commit()  return jsonify({'message': 'Student updated successfully'}) |
| For Delete  @app.route('/students/<int:id>', methods=['DELETE'])  def delete\_student(id):  student = Student.query.filter\_by(student\_id=id).first()  if student is None:  return jsonify({'message': 'Student not found'})  else:  db.session.delete(student)  db.session.commit()  return jsonify({'message': 'Student deleted successfully'}) |
| For Show all records  @app.route('/students', methods=['GET'])  def get\_all\_students():  students = Student.query.all()  output = []  for student in students:  student\_data = {  'student\_id': student.student\_id,  'first\_name': student.first\_name,  'last\_name': student.last\_name,  'dob': student.dob,  'amount\_due': student.amount\_due  }  output.append(student\_data)  return jsonify({'students': output}) |
| Presentation (Postman/Website)  <!DOCTYPE html>  <html>  <head>  <title>Student Records</title>  </head>  <body>  <h1>Student Records</h1>  <ul id="students-list"></ul>  <script>  // Retrieve all student records using fetch  fetch('http://localhost:5000/students')  .then(response => response.json())  .then(data => {  // Loop through the list of student records and add them to the HTML page  data.forEach(student => {  const listItem = document.createElement('li');  listItem.innerText = `${student.first\_name} ${student.last\_name} (${student.amount\_due})`;  document.getElementById('students-list').appendChild(listItem);  });  });  </script>  </body>  </html>  This HTML page retrieves all student records from the API endpoint at http://localhost:5000/students using fetch, and displays them in an unordered list on the HTML page. You can modify this code to make requests to other API endpoints as needed. |
| Put your app in git repo |
| **Collaboration plan and work division**  Pritesh-Creating Database  Creating Basic Flask App  For Create  For Read  For Update  Rutvik - For Delete  For Show all records  Presentation (Postman/Website)  Put your app in git repo  Collaboration plan and work division |