**Prerequisites**

**MySQL Database Setup:**

Ensure you have MySQL Server installed and running.

Create a database named school and a table named students with appropriate columns (id, name, age, grade). Refer to the instructions for setting up the database.

**Python Environment:**

Make sure you have Python installed on your system. You can download Python from python.org if not already installed.

**Project Files:**

1. database.py:

Implement MySQL connection and CRUD operations.

2. main.py:

Implement FastAPI application with CRUD endpoints.

3. requirements.txt:

List the Python dependencies (fastapi, uvicorn, mysql-connector-python).

------------------------------------------------------------

**requirements.txt:**

fastapi==0.70.0

uvicorn==0.15.0

mysql-connector-python==8.0.28

**How to Use:**

Create a new file named requirements.txt in your project directory.

Copy the above dependencies into requirements.txt.

Install these dependencies using pip:

pip install -r requirements.txt

By including these dependencies in your requirements.txt file, you ensure that anyone setting up your FastAPI project can easily install the necessary packages with a single command.

------------------------------------------------------------

**Install uvicorn Globally**

UVICORN is an ASGI (Asynchronous Server Gateway Interface) web server implementation tailored for Python.

pip install uvicorn

uvicorn main:app --reload

python -m uvicorn main:app --reload

Once uvicorn starts your FastAPI application successfully, you should see output indicating that the server is running, usually on http://127.0.0.1:8000 by default.

Terminate the Server:

To stop the FastAPI application, press Ctrl + C in the terminal or command prompt where the server is running.

**MySQL Server**

import mysql.connector

from mysql.connector import Error

def create\_connection():

connection = None

try:

connection = mysql.connector.connect(

host="localhost",

user="root",

password="1234",

database="school"

)

print("Connection to MySQL DB successful")

except Error as e:

print(f"The error '{e}' occurred")

return connection

**host**: The hostname or IP address where your MySQL server is running. Typically, if it's on your local machine, you use "localhost".

**user**: The username to authenticate with the MySQL server.

**password**: The password corresponding to the username for authentication.

**database**: The name of the database you want to connect to.

**Create Database and Table:**

CREATE DATABASE school;

USE school;

CREATE TABLE students (

id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(100) NOT NULL,

age INT NOT NULL,

grade VARCHAR(10) NOT NULL

);

**Explanation:**

CREATE DATABASE: This SQL command creates a new database named school.

USE school: Sets the school database as the current working database.

CREATE TABLE students: Defines a table named students within the school database. It includes columns id (auto-incremented integer and primary key), name (string), age (integer), and grade (string).

**CRUD:**

To perform CRUD operations on your FastAPI application using Postman, you'll use HTTP methods (POST, GET, PUT, DELETE) to interact with your API endpoints. Here’s how you can set up and execute these commands in Postman:

**Prerequisites**

Ensure your FastAPI application is running using uvicorn as explained earlier.

**1. Create a Student (POST Request)**

Method: POST

URL: http://127.0.0.1:8000/students/

Headers:

Content-Type: application/json

Accept: application/json

{

"name": "John Doe",

"age": 21,

"grade": "A"

}

Click "Send". You should receive a response confirming the creation of the student.

**2. Get All Students (GET Request)**

Method: GET

URL: http://127.0.0.1:8000/students/

Headers:

Accept: application/json

Click "Send". You should receive a response with a list of all students.

**3. Get a Student by ID (GET Request)**

Method: GET

URL: http://127.0.0.1:8000/students/{student\_id}

Replace {student\_id} with the ID of the student you want to fetch.

Headers:

Accept: application/json

Click "Send". You should receive a response with the student details if the student exists.

**4. Update a Student (PUT Request)**

Method: PUT

URL: http://127.0.0.1:8000/students/{student\_id}

Replace {student\_id} with the ID of the student you want to update.

Headers:

Content-Type: application/json

Accept: application/json

{

"name": "Jane Doe",

"age": 22,

"grade": "A+"

}

**5. Delete a Student (DELETE Request)**

Method: DELETE

URL: http://127.0.0.1:8000/students/{student\_id}

Replace {student\_id} with the ID of the student you want to delete.

Headers:

Accept: application/json

Click "Send". You should receive a response confirming the deletion of the student.

**Notes:**

1. Ensure your FastAPI server (uvicorn main:app --reload) is running while testing with Postman.
2. Adjust the URL (127.0.0.1:8000) and endpoint paths (/students/, /students/{student\_id}) based on your FastAPI application's configuration.
3. Verify each operation's success through Postman's response and status codes (200 OK, 201 Created, 204 No Content, 404 Not Found, etc.).

By following these steps, you can effectively test your FastAPI application's CRUD operations using Postman, ensuring that your API endpoints behave as expected when interacting with the MySQL database.