

BLACK BUCKS ASSIGNMENT-5

PART-1:

AI for Code Generation (GitHub Copilot or ChatGPT)

Task : generate a simple **CRUD app backend** in **Node.js** or **Python (Flask)**.

The prompt/code comment that I gave:

can you generate a simple CRUD app backend in Python (Flask).

The code AI generated:

Sure! Here's a **simple CRUD (Create, Read, Update, Delete) backend app** using **Flask** and **SQLite** as the database. It manages a basic resource like "users".

☐ Features:

- Create a user
- Read all users or a specific user
- Update a user
- Delete a user

☐ Project Structure:

```
crud_app/
├── app.py           # Flask app with all routes
├── database.db      # SQLite database (auto-created)
└── requirements.txt
```

☐ Code: `app.py`

```
from flask import Flask, request, jsonify
import sqlite3

app = Flask(__name__)
```

```

DB_NAME = 'database.db'

# Initialize DB
def init_db():
    conn = sqlite3.connect(DB_NAME)
    c = conn.cursor()
    c.execute('''
        CREATE TABLE IF NOT EXISTS users (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            name TEXT NOT NULL,
            email TEXT UNIQUE NOT NULL
        )
    ''')
    conn.commit()
    conn.close()

init_db()

# Helper: Run SQL query
def query_db(query, args=(), one=False):
    conn = sqlite3.connect(DB_NAME)
    conn.row_factory = sqlite3.Row
    cur = conn.cursor()
    cur.execute(query, args)
    rv = cur.fetchall()
    conn.commit()
    conn.close()
    return (rv[0] if rv else None) if one else rv

# Routes
@app.route('/users', methods=['POST'])
def create_user():
    data = request.get_json()
    try:
        query_db("INSERT INTO users (name, email) VALUES (?, ?)",
            (data['name'], data['email']))
        return jsonify({"message": "User created"}), 201
    except sqlite3.IntegrityError:
        return jsonify({"error": "Email must be unique"}), 400

@app.route('/users', methods=['GET'])
def get_users():
    users = query_db("SELECT * FROM users")
    return jsonify([dict(u) for u in users])

@app.route('/users/<int:user_id>', methods=['GET'])
def get_user(user_id):
    user = query_db("SELECT * FROM users WHERE id = ?", [user_id], one=True)
    if user:
        return jsonify(dict(user))
    return jsonify({"error": "User not found"}), 404

@app.route('/users/<int:user_id>', methods=['PUT'])
def update_user(user_id):
    data = request.get_json()
    user = query_db("SELECT * FROM users WHERE id = ?", [user_id], one=True)
    if not user:

```

```
        return jsonify({"error": "User not found"}), 404
    query_db("UPDATE users SET name = ?, email = ? WHERE id = ?",
(data['name'], data['email'], user_id))
    return jsonify({"message": "User updated"})

@app.route('/users/<int:user_id>', methods=['DELETE'])
def delete_user(user_id):
    query_db("DELETE FROM users WHERE id = ?", [user_id])
    return jsonify({"message": "User deleted"})

if __name__ == '__main__':
    app.run(debug=True)
```

❑ **requirements.txt**

flask

❑ **How to Run**

1. Install Flask:
 2. `pip install -r requirements.txt`
 3. Run the app:
 4. `python app.py`
 5. Test using Postman or curl:
 6. `curl -X POST http://127.0.0.1:5000/users -H "Content-Type: application/json" -d '{"name":"John", "email":"john@example.com"}'`
-

Let me know if you want to add things like JWT auth, pagination, or Swagger docs too!

AI's Role in Code Generation:

I used ChatGPT to help me build a CRUD app with Flask. It gave me clean, ready-to-use code and also walked me through how to set everything up on my system. Whenever I got stuck—like with starting the server or using curl—ChatGPT guided me step-by-step. Honestly, I didn't even need to change the code; it just worked straight out of the box