**🧾 🔍 INVOICE GENERATOR PROJECT – COMPLETE EXPLANATION**

**🔸 1. init\_db() – Creates the database and tables**

python

CopyEdit

def init\_db():

conn = sqlite3.connect('invoice.db') # Connect to SQLite database file

cursor = conn.cursor() # Create a cursor to run SQL queries

**💾 Tables created:**

* **customers**: Stores name, contact, address
* **invoices**: Stores customer ID, date, and total
* **items**: Stores product details per invoice

python

CopyEdit

cursor.execute('''CREATE TABLE IF NOT EXISTS customers (...)''')

cursor.execute('''CREATE TABLE IF NOT EXISTS invoices (...)''')

cursor.execute('''CREATE TABLE IF NOT EXISTS items (...)''')

python

CopyEdit

conn.commit() # Save changes

conn.close() # Close connection

**🔸 2. Flask App Setup**

python

CopyEdit

app = Flask(\_\_name\_\_)

* Creates the web application using Flask

**🔸 3. Home Route /**

python

CopyEdit

@app.route('/')

def home():

return "Welcome to the Invoice Generator! Go to /create to add invoice."

* Shows a simple message with instructions

**🔸 4. Route /create – Shows the invoice form**

python

CopyEdit

@app.route('/create', methods=['GET'])

def create\_invoice():

return render\_template('create\_invoice.html')

* Loads the form HTML (create\_invoice.html) from the templates/ folder

**🔸 5. HTML Form (create\_invoice.html)**

* Takes:
  + Customer info
  + One or more product rows (with JS "Add Another Item" feature)
* On submission, sends POST request to /submit

Example input fields:

html

CopyEdit

<input name="item\_name[]" ...>

<input name="qty[]" ...>

<input name="price[]" ...>

**🔸 6. Route /submit – Handles form submission**

python

CopyEdit

@app.route('/submit', methods=['POST'])

def submit\_invoice():

**✅ Step-by-step inside /submit:**

**🟢 1. Get user input**

python

CopyEdit

name = request.form['name']

contact = request.form['contact']

address = request.form['address']

item\_names = request.form.getlist('item\_name[]')

qtys = request.form.getlist('qty[]')

prices = request.form.getlist('price[]')

**🟢 2. Prepare item details and total amount**

python

CopyEdit

items = []

total\_amount = 0

for name\_i, qty\_i, price\_i in zip(item\_names, qtys, prices):

...

items.append({...})

total\_amount += total

**🟢 3. Insert customer into customers table**

python

CopyEdit

cursor.execute("INSERT INTO customers (...) VALUES (?, ?, ?)", (name, contact, address))

customer\_id = cursor.lastrowid # Get the ID of the newly inserted customer

**🟢 4. Insert invoice into invoices table**

python

CopyEdit

date = datetime.now().strftime("%Y-%m-%d %H:%M:%S")

cursor.execute("INSERT INTO invoices (...) VALUES (?, ?, ?)", (customer\_id, date, total\_amount))

invoice\_id = cursor.lastrowid # Get invoice ID

**🟢 5. Insert each item into items table**

python

CopyEdit

for item in items:

cursor.execute("INSERT INTO items (...) VALUES (?, ?, ?, ?, ?)",

(invoice\_id, item['item\_name'], item['qty'], item['price'], item['total']))

**🟢 6. Commit and close**

python

CopyEdit

conn.commit()

conn.close()

**🟢 7. Show invoice summary using HTML**

python

CopyEdit

return render\_template('invoice\_summary.html',

name=name,

contact=contact,

address=address,

date=date,

items=items,

total\_amount=total\_amount)

**🔸 7. Invoice Summary Page (invoice\_summary.html)**

* Receives all values from the backend
* Displays:
  + Customer details
  + Table of items
  + Total amount

Example:

html

CopyEdit

<p><strong>Name:</strong> {{ name }}</p>

{% for item in items %}

<tr>

<td>{{ item.item\_name }}</td>

...

</tr>

{% endfor %}

**🧠 Final Concepts You Learned:**

| **Concept** | **What You Did** |
| --- | --- |
| Flask routes | Built GET and POST endpoints |
| HTML form handling | Sent form data with multiple items |
| SQLite integration | Inserted and retrieved relational data |
| Template rendering | Passed data into summary page |
| Foreign Keys | Linked customer → invoice → items |
| Python logic | Looped through input and calculated totals |

**✅ Your Project is a Full Working App!**

You built a:

* Multi-table database app
* With a web frontend
* That collects data and renders output
* Saved and tracked in a relational DB