

Shop Assist: AI-Powered Laptop Chatbot

This document provides a summary of the **Shop Assist** project, a Python Flask web application that functions as a conversational chatbot to help users find and compare laptops. The primary objective is to deliver a seamless, natural, and context-aware chat experience.

Core Components

1. app.py – Flask Backend

Purpose:

The app.py file is the **heart of the server-side logic**. It acts as the bridge between the **frontend UI** and the **chatbot logic**.

Key Responsibilities:

- **Route Handling:**
 - GET / → Serves the main index.html chat page.
 - POST /get response → Accepts user messages, processes them, and returns chatbot replies.
 - **Session Management:**
 - Stores the **chat history** per user session to keep the conversation context-aware.
 - **API Integration:**
 - Sends messages and chat history to conversation.py for processing.
 - Receives processed, human-like responses from the chatbot logic.
 - **Error Handling:**
 - Catches and returns user-friendly error messages if something goes wrong (e.g., data file missing, API issues).
-

2. conversation.py – Chatbot Logic

Purpose:

The conversation.py module acts as the **“brain”** of Shop Assist.

Key Responsibilities:

- **Laptop Data Loading:**
 - Reads laptop.csv into memory using pandas for easy filtering and searching.
 - **User Intent Understanding:**
 - Identifies key information from user messages, such as:
 - **Budget** (e.g., “under ₹50,000”)
 - **Brand preference** (e.g., “Dell” or “HP”)
 - **Usage type** (e.g., “gaming”, “video editing”, “programming”)
 - **Filtering and Comparison:**
 - Matches laptops from the CSV based on the extracted intent.
 - Supports multi-criteria filtering (e.g., “gaming laptop under ₹80,000 with 16GB RAM”).
 - Allows direct **model comparison** if the user specifies two laptops.
 - **AI Response Generation:**
 - Prepares a structured prompt with filtered data and context.
 - Sends the prompt to the **OpenAI API**.
 - Returns a **friendly, coherent, and helpful** reply.
-

3. laptop.csv – Data Source

Purpose:

A **Comma-Separated Values (CSV)** file storing laptop specifications for quick search and filtering.

Typical Data Fields:

- **Brand** – e.g., Dell, HP, Lenovo, Asus
- **Model Name**
- **Price** – in ₹ (or other currencies)
- **RAM Size** – e.g., 8GB, 16GB
- **Processor** – e.g., Intel i5, Ryzen 7
- **Storage** – e.g., 512GB SSD, 1TB HDD
- **Graphics Card** – e.g., NVIDIA GTX 1650
- **Special Features** – e.g., Touchscreen, 144Hz Display

This file acts as the **product database** for the chatbot.

4. index.html – Frontend

Purpose:

The **user interface** for interacting with the chatbot.

Key Features:

- **Chat Interface:**
 - User input box for sending messages.
 - Scrollable chat history showing both user queries and chatbot responses.
 - **Typing Indicator:**
 - Displays “Shop Assist is typing...” while the backend generates a response.
 - **Automatic Scrolling:**
 - Chat always scrolls to the latest message automatically.
 - **Responsive Design:**
 - Built with **Tailwind CSS** for a modern, mobile-friendly look.
 - **Real-Time Feel:**
 - Uses JavaScript (AJAX or Fetch API) to send messages without reloading the page.
-

How the Application Works – Step-by-Step Flow

1. **User Opens Website**
 - Flask serves index.html to the browser.
2. **User Sends a Message**
 - Example: *"Show me gaming laptops under ₹80,000 with at least 16GB RAM"*
 - JavaScript captures the message and sends it to /get_response via an AJAX POST request.
3. **Backend Processes Request** (app.py)
 - Retrieves the current chat history from the session.
 - Passes the new message + history to conversation.py.
4. **Chatbot Logic Runs** (conversation.py)
 - Extracts relevant details (budget, RAM, category).
 - Searches laptop.csv for matching laptops.
 - Formats the filtered results into a structured, easy-to-read summary.

- Sends this summary to the AI API to produce a natural, conversational reply.

5. AI Generates Response

- Example:
*"Here are three gaming laptops under ₹80,000 with 16GB RAM:
1. Asus TUF Gaming F15 – ₹76,990 – Intel i7, NVIDIA RTX 3050.
2. Dell G15 – ₹78,499 – Ryzen 7, NVIDIA RTX 3060.
3. HP Omen – ₹79,999 – Intel i5, NVIDIA GTX 1650.
All have fast SSD storage and high-refresh displays. Which one do you want to compare in detail?"*

6. Response Sent to User

- Flask returns the AI-generated reply to the frontend.
- JavaScript updates the chat window with the new message.

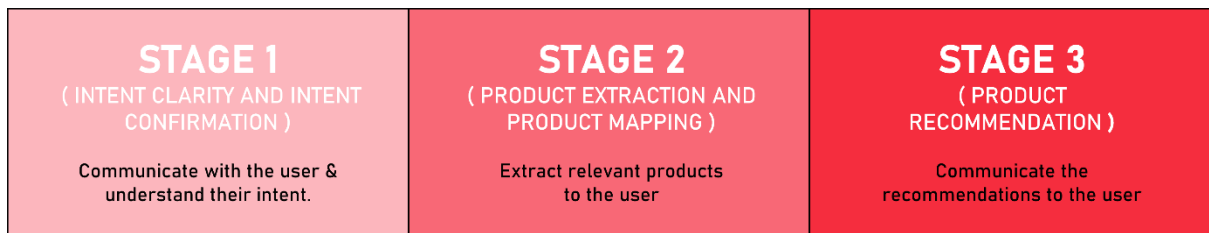
7. Continuous Conversation

- User can ask follow-up questions like:
 - *"Compare Dell G15 and Asus TUF"*
 - *"Which has better battery life?"*
- Chat history ensures the bot remembers the context.

Advantages of the Shop Assist Design

- **Context-Aware:** Remembers conversation history, allowing natural follow-up questions.
- **Multi-Criteria Search:** Supports complex queries like "HP laptops under ₹60,000 for video editing."
- **AI-Enhanced Replies:** Turns raw data into **human-like recommendations**.
- **Responsive UI:** Fast, modern, and mobile-friendly chat interface.
- **Modular Architecture:**
 - Easy to replace CSV with a database.
 - Simple to switch AI models (OpenAI, Gemini, etc.).

Stages of Shop Assist AI



Screen shots of Chat Bot

