**Admin Manual: GiftChatBot**

### 1. Introduction

This manual provides administrators with the necessary instructions to manage, maintain, and monitor the GiftChatBot system.

### 2. System Overview

GiftChatBot is an e-commerce assistant built using Rasa. It enables customers to interact with an online store via natural language processing. The system supports:

* Product browsing
* Stock checking
* Cart management
* Order placement
* Feedback collection

The backend is built with Flask and communicates with the chatbot via REST APIs.

### 3. Project Structure

GiftChatBot/  
├── actions/ # Custom actions for Rasa  
├── data/ # NLU, rules, and stories  
├── documents/ # User/Admin manuals  
├── instance/ # Flask instance-specific configs  
├── media/ # Media assets  
├── models/ # Trained Rasa models  
├── mywebapp/ # Web interface (if any)  
├── tests/ # Testing scripts  
├── chatbot\_app.py # Flask API app  
├── config.yml # Rasa config  
├── credentials.yml # Channels (REST, etc.)  
├── domain.yml # Intents, responses, slots, etc.  
├── endpoints.yml # Action server endpoint

### 4. Starting the Chatbot

**Run the Rasa server:**

rasa run --enable-api --cors "\*"

**Run the action server:**

rasa run actions

**Run the Flask web API:**

python chatbot\_app.py

Make sure ports are not blocked by firewalls.

### 5. Adding/Updating FAQs

FAQs are grouped by categories and served dynamically from the Rasa backend. To update them:

* Open nlu.yml
* Add intent examples under faq/your\_topic intents
* Update domain.yml with corresponding utter\_faq/your\_topic responses

Example:

## intent:faq/how\_to\_order  
- How do I place an order?  
- Can I order through the bot?  
  
responses:  
 utter\_faq/how\_to\_order:  
 - text: "To place an order, browse our products, add items to your cart, and proceed to checkout."

### 6. Managing Orders and Products

Order and product management is handled via the Flask API. Endpoints include:

* /products — List products
* /cart\_items — Add/view items in cart
* /orders — Create and view orders

Use Postman or a frontend interface to interact with the endpoints.

### 7. Monitoring Logs

Logs are printed in the terminal during Rasa and Flask runtime. For production, configure logging to files.

### 8. Deployment Considerations

* Use nginx or Apache to serve the frontend securely.
* Use gunicorn or uvicorn to serve Flask in production.
* Setup HTTPS with Let’s Encrypt.
* Run Rasa and actions as services using systemd or Docker.

### 9. Download Admin Manual

Make sure the manual is placed in the /documents/ folder, and Flask serves it via:

@app.route('/download/admin-manual')  
def download\_admin\_manual():  
 return send\_from\_directory('documents', 'Admin\_Manual\_Chatbot.docx', as\_attachment=True)

### 10. Contact

For technical issues, contact the system developer or maintainer.