Shri ganeshay namah

AI Astrologer E-Commerce Chatbot Integration Guide for AstroRemedis.com

Purpose: This guide outlines the development and integration of an AI-powered Astrologer Chatbot for AstroRemedies.com, an e-commerce site selling Rudraksha, gemstones, stone bracelets, yantras, and other astrological items. The chatbot will provide future predictions (based on KP Astrology and Bhrigu Nadi), suggest remedies (Lal Kitab, gemstones, bracelets), and include direct product links for seamless purchases. Initial consultations are free; subsequent ones require recharge/prepaid payments (e.g., wallet system). This is structured for easy pasting into Microsoft Word for your AI developer.

Key Features:

Predictions: 90% accurate (via fine-tuned AI on historical data).

Remedies: Personalized, with e-commerce links (e.g., "Buy Emerald: https://astroremedies.com/product/emerald-5-carat").

Monetization: Free first session; prepaid wallet for more (integrate Stripe/Razorpay).

Integration: Embed as a chat widget on the site.

Disclaimer: Astrology is for entertainment/spiritual guidance. Remedies are suggestions—consult experts. Ensure PCI compliance for payments and data privacy (GDPR/DPDP Act).

Table of Contents

Project Overview

Achieving 90% Prediction Accuracy

E-Commerce Integration and Product Linking

Monetization: Free-to-Paid Model

Astrology Engine: Free/Paid APIs and Language Options

Prerequisites

Step 1: Planning and Features

Step 2: Data Sources for Remedies and Products

Step 3: Technical Setup

Step 4: Core Implementation (Code Examples)

Step 5: AI Integration and Fine-Tuning

Step 6: Website Integration

Step 7: Testing

Step 8: Deployment and Scaling

Step 9: Business and Legal Considerations

Troubleshooting

Resources

1. Project Overview

The chatbot enhances AstroRemedies.com by turning astrological consultations into direct sales opportunities:

User Flow: User asks about future → Bot predicts (KP/Bhrigu) → Suggests remedy → Provides product link (e.g., gemstone/bracelet) → User buys on-site.

E-Commerce Tie-In: Remedies link to inventory (Rudraksha, gemstones, bracelets, yantras).

Monetization: Free first prediction/remedy; recharge (e.g., ₹100 for 5 sessions) for more.

Tech Stack:

Backend: Python + FastAPI (for astrology/payment logic).

Frontend: JavaScript widget (embed on site).

Payments: Stripe/Razorpay for wallet recharge.

Database: PostgreSQL for user sessions/products.

2. Achieving 90% Prediction Accuracy

Dataset: 1000+ verified charts (KP timings, Bhrigu outcomes) from AstroSage/experts.

Backtesting: Validate on historical data (80/20 split).

Fine-Tuning: Hugging Face on astrology-specific data.

Expert Validation: 2-3 astrologers review rules.

Hybrid Model: Rule-based (KP formulas) + AI for remedies/links.

Feedback: Log user purchases/confirmations for retraining.

Benchmark: 70% MVP → 90% post-fine-tuning.

3. E-Commerce Integration and Product Linking

Product Mapping: Link remedies to site inventory.

Example: "Saturn remedy: Blue Sapphire" → Link: https://astroremedies.com/product/blue-sapphire-5-carat.

Dynamic Links: Use backend to query site DB (e.g., WooCommerce API if WordPress).

Inventory Check: Ensure links show stock/price (e.g., "In stock: ₹1500").

Tracking: Use UTM params (e.g., ?utm\_source=chatbot) for analytics.

Product Categories in Bot:

Rudraksha (e.g., 7-Mukhi for Saturn).

Gemstones (e.g., Emerald).

Bracelets (e.g., Crystal for Rahu).

Yantras (e.g., Shani Yantra).

4. Monetization: Free-to-Paid Model

Free Tier: First 1-2 sessions (prediction + remedy link).

Paid Tier: Recharge wallet (₹50-500) for unlimited/priority sessions.

Implementation:

Check user balance in DB before processing.

Integrate Stripe/Razorpay: /recharge endpoint for payments.

Wallet Deduction: ₹20/session post-free.

UI: Bot says, "Free session used. Recharge ₹100 for more?" with payment button.

Revenue Streams:

Consultations (₹51/session).

Product sales via links (20-30% margin).

5. Astrology Engine: Free/Paid APIs and Language Options

Free:

Swiss Ephemeris (Python: pyswisseph for KP/Bhrigu).

AstroSage API (Lal Kitab remedies).

Paid:

rva astrology.com

AstrologyAPI.com

Prokerala

Languages:

Python: Recommended (Swiss Ephemeris).

JS/Node.js: For site integration.

6. Prerequisites

Skills: Python/JS, e-commerce APIs (WooCommerce/Stripe).

Tools: VS Code, Git.

Accounts: Gemini API, Stripe/Razorpay keys, site admin access.

Install: pip install pyswisseph fastapi stripe geopy; npm install razorpay.

7. Step 1: Planning and Features

Features:

Predictions: KP (timing), Bhrigu (events).

Remedies: Lal Kitab + product links.

E-Commerce: Direct buy buttons.

Payments: Wallet recharge.

User Flow:

Query: "Career future?"

Bot: Predicts + remedy + link (free).

Next: "Recharge to continue."

8. Step 2: Data Sources for Remedies and Products

Remedies: remedies.json with links.

{

"gemstones": {

"emerald": {"description": "For Mercury", "link": "https://astroremedies.com/product/emerald-5-carat"}

},

"bracelets": {

"rudraksha-7-mukhi": {"description": "For Saturn", "link": "https://astroremedies.com/product/rudraksha-bracelet-7-mukhi"}

}

}

Products: Query site API (e.g., WooCommerce REST API).

9. Step 3: Technical Setup

Folder: mkdir astro-bot-ecom.

Env: python -m venv venv.

Files: app.py (backend), chat.js (widget), config.py.

DB: PostgreSQL table for wallets: users (id, balance REAL).

Config: Add STRIPE\_KEY, SITE\_API\_URL.

10. Step 4: Core Implementation

Backend (app.py):

from fastapi import FastAPI

import swisseph as swe

from datetime import datetime

import stripe

import json

from geopy.geocoders import Nominatim

app = FastAPI()

stripe.api\_key = STRIPE\_KEY

swe.set\_ephe\_path(EPHE\_PATH)

swe.set\_sid\_mode(swe.SIDM\_KRISHNAMURTI)

def get\_astrology\_data(birth\_date, birth\_time, location): # KP/Bhrigu as before

# ... (from previous code)

return astro\_data

def get\_remedies\_with\_links(afflicted):

with open('remedies.json', 'r') as f:

data = json.load(f)

gem\_link = data['gemstones'].get(afflicted, {}).get('link', '#')

return {'gemstone': f"Buy: {gem\_link}"}

@app.post("/consult")

async def consult(data: dict, user\_id: str):

# Check balance

balance = get\_user\_balance(user\_id) # From DB

sessions\_used = get\_sessions\_used(user\_id)

is\_free = sessions\_used < 1

if not is\_free and balance < 20:

return {'error': 'Recharge required', 'recharge\_link': '/recharge'}

# Compute

astro\_data = get\_astrology\_data(data['birth\_date'], data['time'], data['location'])

remedies = get\_remedies\_with\_links('mercury') # Example

prediction = get\_ai\_prediction(astro\_data, data['query']) # From AI

if not is\_free:

deduct\_balance(user\_id, 20)

return {'prediction': prediction, 'remedies': remedies, 'links': remedies}

@app.post("/recharge")

async def recharge(user\_id: str, amount: int):

payment = stripe.PaymentIntent.create(amount=amount\*100, currency='inr')

# On success: update DB balance

return {'success': True, 'client\_secret': payment.client\_secret}

# Run: uvicorn app:app

Frontend (chat.js):

// Widget HTML as before

async function sendConsult() {

const data = { /\* inputs \*/ };

const res = await fetch('/consult', { method: 'POST', body: JSON.stringify(data) });

const result = await res.json();

if (result.error) {

document.getElementById('response').innerHTML += `<button onclick="recharge()">Recharge ₹100</button>`;

} else {

// Display prediction + <a href="${result.links.gemstone}">Buy Now</a>

}

}

function recharge() {

// Stripe integration

stripe.redirectToCheckout({ sessionId: 'session\_id' });

}

11. Step 5: AI Integration and Fine-Tuning

Prompt: Include "Suggest remedy with product link from AstroRemedies.com."

Fine-Tuning: Add e-commerce examples to dataset.

12. Step 6: Website Integration

Embed chat.js in footer.

API Calls: Point to backend URL.

Analytics: Track link clicks.

13. Step 7: Testing

Unit: Test links, payments.

E2E: Simulate free/paid flow, purchase.

Accuracy: 90% on 100 tests.

14. Step 8: Deployment and Scaling

Heroku/AWS.

Scale: Redis for sessions.

15. Step 9: Business and Legal Considerations

Revenue: 30% from sales uplift.

Legal: PCI for payments; disclaimers.

16. Troubleshooting

Link Errors: Verify site URLs.

Payments: Test Stripe sandbox.

17. Resources

Stripe Docs: stripe.com/docs.

WooCommerce API: woocommerce.com/api.

Eti shree