Book Recommendation Chatbot using IBM Watsonx Assistant

BuiltwithPythonWebhooksand DatasetIntegration

Presented by:DikshaPatel

Project Overview & Objectives

Ourgoalwastodevelopaconversational chatbotcapableofrecommending books based on user preferences, primarily genre.Thisprojectseamlesslyintegrates IBMWatsonxAssistantwithcustom Python webhooks for dynamic data retrieval.

Conversational AI

Leveraging IBM Watsonx Assistant for natural language understanding and dialog management.

Dynamic Recommendations

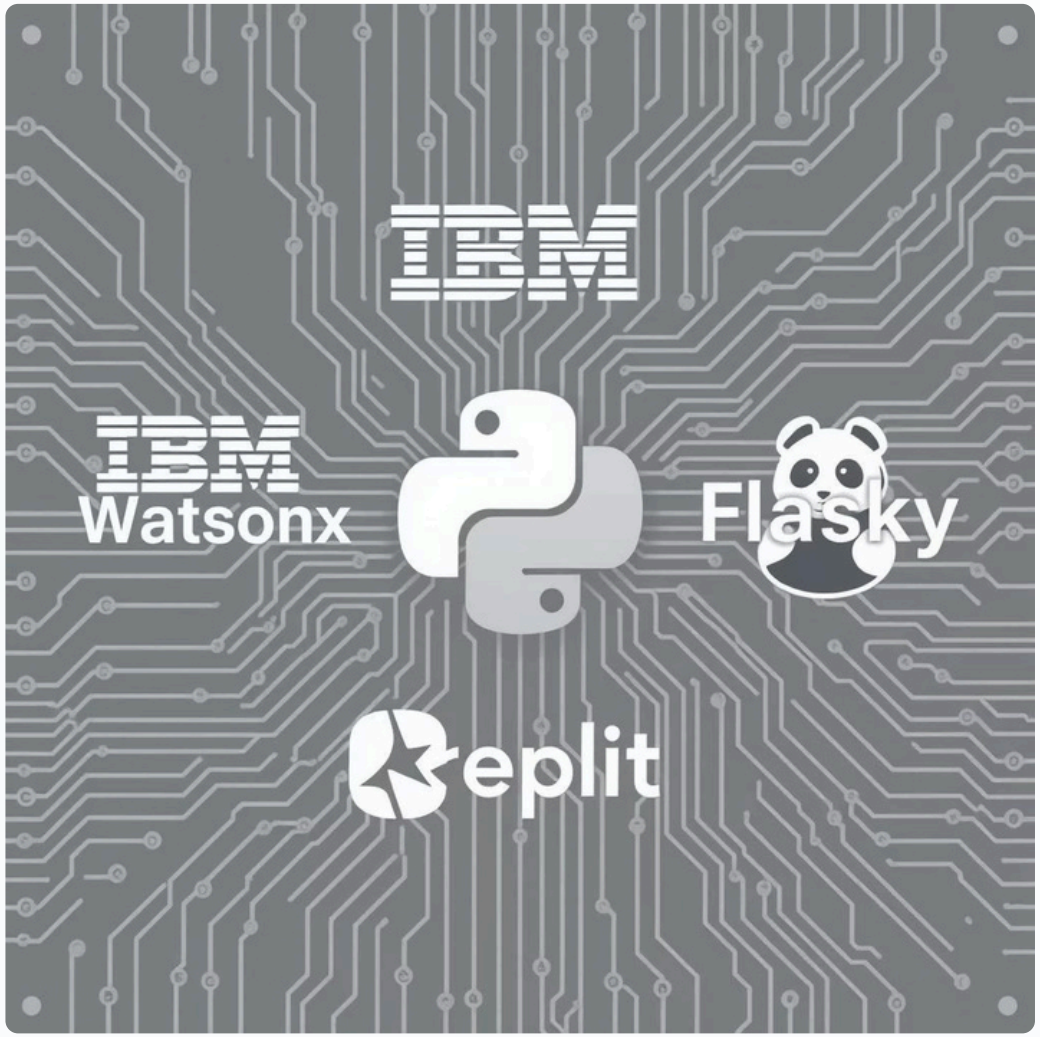
Using Python webhooks to fetch and filter book data in real-time based on user input.

Dataset Integration

Integrating a local `books.csv` dataset to serve diverse book recommendations.

Key Technologies Utilised

- **IBMWatsonx Assistant:** Core AIfor chatbotlogic.
- **Python (Flask):** Backend framework for webhook.
- **Pandas:** For efficient dataset handling.
- **Replit:** Hosting and deployment platform.
- **Ngrok/Replit Public URL:** Exposing the webhook securely.
- **books.csv:** The primary book dataset.



System Architecture Flow

Thechatbotoperatesonaclear,sequentialflow,ensuringseamlessinteraction from user query to book recommendation.

1

User Interaction

User queries Watsonx chatbot for book recommendations.

2

Input Collection

Watsonx collects user preferences, such as desired genre.

3

WebhookTrigger

Watsonx sends input to Python Flask backend via webhook.

4

Data Processing

Flask app reads `books.csv` and filters recommendations.

5

Response Display

Watsonx receives and displays recommended books to the user.

The Dataset: `books.csv`

Ourbookrecommendationengineispoweredbyastructured CSV dataset, making it easy to filter and retrieve relevant titles.

title	author	genre
The Hobbit	J.R.R. Tolkien	Fantasy
1984	George Orwell	Dystopian
The Alchemist	Paulo Coelho	Fiction
Pride and Prejudice	Jane Austen	Classic
Dune	Frank Herbert	Science Fiction

Each entry includes the book's title, author, and genre, enabling precise recommendations.

Webhook Backend: Python Flask

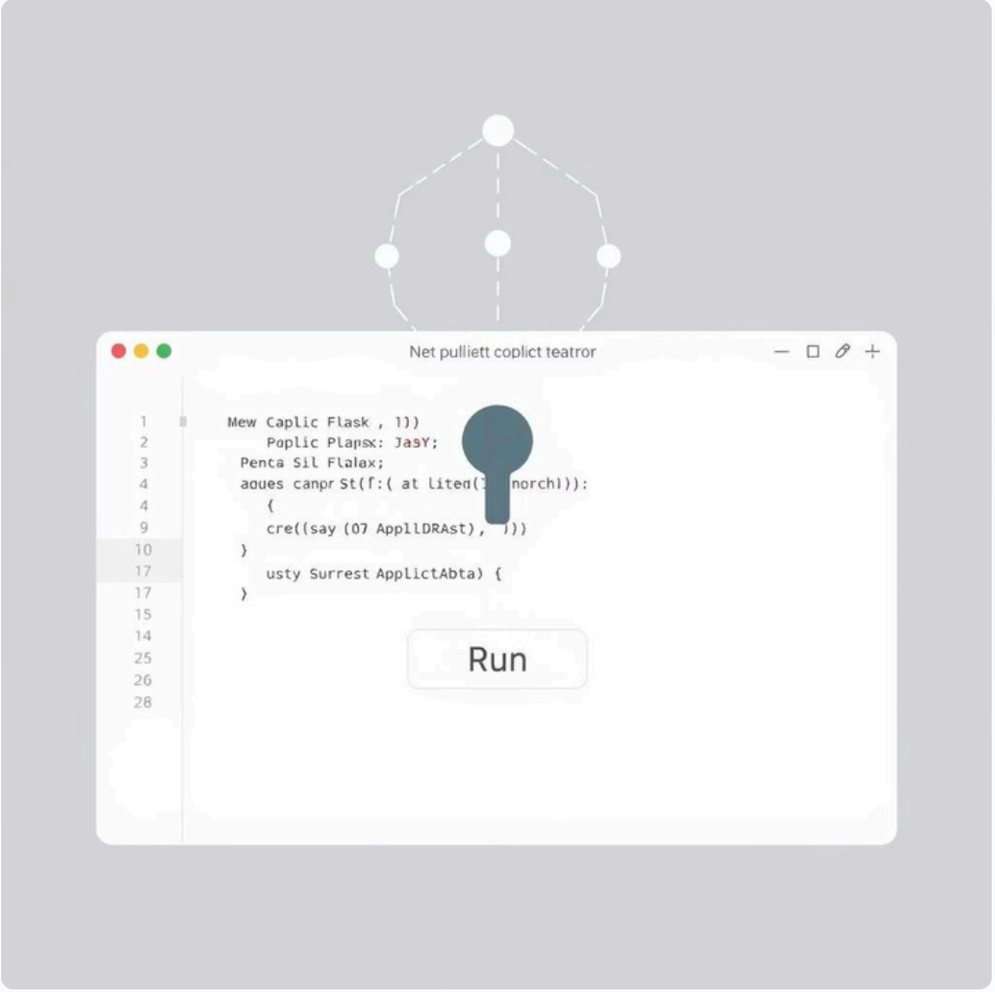
TheFlaskapplicationservesastheintelligentintermediary,processingrequestsfromWatsonx Assistant and delivering tailored book suggestions.

```
@app.route('/webhook', methods=['POST'])
def webhook():
    genre = request.get_json().get('genre', '').lower()
    result = books[books['genre'].str.lower() == genre].head(3)
    if not result.empty:

        recommendations = ""
        for index, row in result.iterrows():
            recommendations += f"" {row['title']} by {row['author']}\n"
        response_text = f"Here are some {genre} books:\n{recommendations}"
    else:
        response_text = "Sorry, I couldn't find any books in that genre."

    return jsonify({"fulfillment_response": {"messages": [{"text": {"text": [response_text]}}]}})
```

- Utilises **Pandas** to read and filter `books.csv`.
- Extracts **genre** from Watsonx request.
- Returns top 2-3 book recommendations based on genre.

Hosting the Webhook on Replit

Replitprovidesan efficientandaccessibleplatformforhostingthePythonFlaskwebhook, making it publicly available for WatsonxAssistant toaccess.



- **Platform:** Replit for seamless deployment.
- **Server:** Flask app runs on port 5000.
- **Public Exposure:** Replit's public domain exposes the webhook URL.
- **Integration:** The generated URL is then configured in Watsonx Assistant settings.

Watsonx Assistant Integration

ConfiguringWatsonxAssistantiscrucialforenablingthechatbottounderstanduser intents, extract entities, and interact with our webhook.

Project Setup

Create a new Watsonx Assistant project.

Intents & Entities

Define intents (e.g., `#recommend_book`) and entities (e.g., `@genre`).

Dialog Nodes

Design dialog flows to capture user preferences.

Webhook Configuration

Integrate the Replit-hosted webhook URL in Watsonx settings.

Parameter Passing

Use `$genre` to send genre preference to the webhook.

Sample User Interaction

Experiencea typicalexchangewiththebookrecommendationchatbot, demonstrating its abilityto deliver relevant suggestionsdynamically.

> **User:** "Can you suggest some fantasy books?"
>
> **Bot (via webhook):**
>
> - "Here are some fantasy books:
> - " The Hobbit by J.R.R. Tolkien
> - " Eragon by Christopher Paolini
> - " A Game of Thrones by George R.R. Martin"

Challenges & Solutions

Challenges Faced

- Steeplearningcurvewith IBM Watsonx integration complexities.
- Ensuring secure exposure of the Replit server.
- Efficient parsing and dynamic filtering of the dataset.
- Rigorous testing and debugging of webhook responses.

Solutions Implemented

- UtilisedReplit'sbuilt-inpublic URL for easy and secure exposure.
- Streamlined dataset structure for optimal filtering per for mance.
- Developed robust fallback mechanisms for unsupported genres.
- Employed extensive logging and print statements for effective debugging.

Conclusion

- Successfully built a working book recommendation bot
- Combined AI chatbot (Watsonx) with real-time Python logic
- Demonstrated webhook-based interaction


Thank
You