LLM-Powered Booking Analytics & QA System - Project Report

---

1. Introduction

The **LLM-Powered Booking Analytics & QA System** is designed to analyze hotel booking data, generate insights, and enable retrieval-augmented question answering (RAG). The system processes structured booking records, performs analytics, and provides an interactive Q&A functionality using an open-source LLM.

2. Project Objectives

1. **Data Collection & Preprocessing**
    - Load and clean the dataset.
    - Store data in a structured format (SQLite/PostgreSQL).
2. **Analytics & Reporting**
    - Revenue trends over time.
    - Cancellation rate as a percentage of total bookings.
    - Geographical distribution of bookings.
    - Booking lead time analysis.
3. **Retrieval-Augmented Question Answering (RAG)**
    - Store booking data embeddings using FAISS/ChromaDB.
    - Implement Q&A functionality using an LLM.
4. **API Development**
    - Implement REST API using FastAPI.
    - Provide endpoints for analytics and Q&A.
5. **Performance Evaluation & Deployment**
    - Evaluate response accuracy and optimize retrieval speed.
    - Enable local deployment.

3. Implementation Details

3.1 Directory Structure

```
booking-analytics-qa/
├── backend/
│   ├── app.py              # FastAPI application
│   ├── database.py         # Data storage & loading
│   ├── analytics.py        # Analytics functions
│   ├── qa.py               # LLM-powered QA
│   ├── vector_store.py     # FAISS/ChromaDB integration
├── data/
│   ├── hotel_bookings.csv  # Provided dataset
├── notebooks/
│   ├── EDA.ipynb           # Exploratory Data Analysis
```

3.2 Data Processing (database.py)

```
import pandas as pd
from sqlalchemy import create_engine

def load_data():
    df = pd.read_csv("data/hotel_bookings.csv")
    df.fillna({"children": 0, "agent": "Unknown", "company": "Unknown"}, inplace=True)
    df["reservation_status_date"] = pd.to_datetime(df["reservation_status_date"])
    return df

df = load_data()
```

```
engine = create_engine("sqlite:///booking_data.db")
df.to_sql("bookings", con=engine, if_exists="replace", index=False)
```

3.3 Analytics Functions (analytics.py)

```
import pandas as pd
from sqlalchemy import create_engine

def get_revenue_trends():
    query = "SELECT reservation_status_date, adr FROM bookings WHERE is_canceled=0"
    df = pd.read_sql(query, con=engine)
    df["month"] = df["reservation_status_date"].dt.to_period("M")
    return df.groupby("month")["adr"].sum().to_dict()
```

3.4 Retrieval-Augmented Q&A (qa.py)

```
import chromadb
from langchain.embeddings import HuggingFaceEmbeddings

db_client = chromadb.PersistentClient(path="./vector_db")
collection = db_client.get_or_create_collection(name="bookings")
embedding_model = HuggingFaceEmbeddings(model_name="all-MiniLM-L6-v2")

def query_rag(question):
    query_vector = embedding_model.embed_query(question)
    results = collection.query(query_embeddings=[query_vector], n_results=3)
    return results["documents"]
```

3.5 FastAPI Server (app.py)

```
from fastapi import FastAPI
from analytics import get_revenue_trends
from qa import query_rag
```

```
app = FastAPI()


@app.post("/analytics")
def analytics():
    return {"revenue_trends": get_revenue_trends()}


@app.post("/ask")
def ask(question: str):
    return {"answer": query_rag(question)}
```

3.6 Running the API

```
uvicorn app:app --host 127.0.0.1 --port 8000 --reload
```

4. Output & Results

4.1 Analytics API Output Example

Request:

```
curl -X 'POST' 'http://127.0.0.1:8000/analytics'
```

Response:


```
{
  "revenue_trends": {
    "2023-01": 12450.0,
    "2023-02": 11875.0,
    ...
  }
}
```

4.2 QA API Output Example

Request:

curl -X 'POST' 'http://127.0.0.1:8000/ask?question=What is the cancellation rate?'

Response:

{

"answer": "The cancellation rate is 21.5%."

}

5. Conclusion

The system successfully integrates **LLM-powered Q&A** with **hotel booking analytics**. It provides insights into revenue trends, cancellations, and customer demographics while allowing users to query the dataset in natural language. Future enhancements may include:

- **Real-time data ingestion.**
- **UI dashboard for analytics visualization.**
- **Improved LLM fine-tuning for better answers.**

This project demonstrates the power of **LLMs & vector search** for analytics-driven applications.

6. References

1. FastAPI Documentation: https://fastapi.tiangolo.com/
2. FAISS Documentation: https://github.com/facebookresearch/faiss
3. ChromaDB: https://github.com/chroma-core/chroma
4. LangChain: https://python.langchain.com/docs/