# Project Report

# Extract Structured Data from Invoice JSON

## Introduction

In today's digital landscape, businesses generate substantial amounts of data, with invoices being a critical component for financial record-keeping. However, extracting meaningful and structured information from these invoices can be a cumbersome task due to their inconsistent formats and structures. This project, titled "Extract Structured Data from Invoice JSON," aims to revolutionize this process by leveraging cutting-edge Artificial Intelligence (AI) and machine learning techniques to automate the extraction of key details from invoices and present them in a structured format.

## Objectives

This project focuses on the following primary objectives:

1. **Automated Data Extraction:** Develop a system capable of parsing and extracting structured data from invoices in various formats.
2. **Generative AI Integration:** Utilize Generative AI models like Gemini to enhance the accuracy and reliability of data extraction.
3. **User-Friendly Interface:** Create an efficient and user-friendly interface for users to query and retrieve specific invoice details.
4. **Standardized Data Output:** Ensure the extracted data is accurate, comprehensive, and presented in a consistent JSON format.

## Methodology

To achieve the stated objectives, the project follows a systematic approach:

1. **Data Collection:**
    - Gather a diverse set of sample invoices in various JSON formats. This dataset will be utilized for training and testing the model. The data variety is crucial for ensuring the model's ability to handle real-world invoice variations.
2. **Model Selection:**
    - Employ Gemini, a powerful AI model, for two key functionalities:
        - **Embedding Generation:** Gemini will generate embeddings, which are high-dimensional vector representations of the invoice data. These embeddings capture the semantic meaning and relationships within the data.
        - **Language Model Utilization:** Gemini's language model capabilities will be utilized to interpret the invoice data and extract specific details like invoice number, order number, date, and total due.
3. **Database Integration:**
    - For efficient retrieval of extracted data, the project will integrate with a Chroma DB database. Chroma DB is well-suited for storing and managing structured data like the invoice information extracted by the model.
4. **User Interface Development:**

- Develop an interface using Gradio. This interface will allow users to:
  - Easily submit queries about specific invoice details.
  - Retrieve the corresponding extracted data in a structured JSON format.
  - This interface can be a web application or a command-line tool, depending on the desired user experience.

5. **Validation and Evaluation:**
   - To ensure the accuracy of the extracted data, the system will be validated by comparing its output with manually annotated datasets. Techniques like precision, recall, and F1 score can be used to evaluate the model's performance.

## Code Implementation

The project involves the following code implementation:

```
python
Copy code
import os
import shutil
import logging
import json
import gradio as gr
from dotenv import load_dotenv
from llama_index.core import Settings, SimpleDirectoryReader, StorageContext, VectorStoreIndex
from llama_index.llms.gemini import Gemini
from llama_index.embeddings.gemini import GeminiEmbedding
from llama_index.vector_stores.chroma import ChromaVectorStore
import chromadb

# Set up logging
logging.basicConfig(level=logging.DEBUG, format='%(asctime)s - %(levelname)s - %(message)s')
logger = logging.getLogger(__name__)

# Load environment variables from .env file
load_dotenv()

# Get the Gemini API key from the environment variable
gemini_api_key = os.getenv('GEMINI_API_KEY')

# Initialize models and settings
gemini_embedding_model = GeminiEmbedding(api_key=gemini_api_key, model_name="models/embedding-001")
llm = Gemini(api_key=gemini_api_key, model_name="models/gemini-pro")
Settings.llm = llm
Settings.embed_model = gemini_embedding_model

# Define the upload directory
UPLOAD_DIR = "./data"

def process_document():
    """Load documents, create vector store and index, and return query engine."""
    documents = SimpleDirectoryReader(UPLOAD_DIR).load_data()
    client = chromadb.PersistentClient(path="./chroma_db")
    chroma_collection = client.get_or_create_collection("quickstart")
    vector_store = ChromaVectorStore(chroma_collection=chroma_collection)
    storage_context = StorageContext.from_defaults(vector_store=vector_store)
    index = VectorStoreIndex.from_documents(documents, storage_context=storage_context)
```

```
    return index.as_query_engine()

def upload_file(file):
    """Handle file upload and processing."""
    logger.debug("Upload function called")

    logger.info(f"File object received: {file}")
    logger.debug(f"File name: {file.name}")

    filename = os.path.basename(file.name)
    logger.info(f"Original filename: {filename}")

    # Create the full path for the uploaded file
    file_path = os.path.join(UPLOAD_DIR, filename)
    logger.info(f"Destination file path: {file_path}")

    # Copy the uploaded file to the destination
    logger.debug("Attempting to copy file")
    shutil.copy(file.name, file_path)

    # Verify if the file was actually copied
    if os.path.exists(file_path):
        logger.info(f"File successfully copied to {file_path}")
        file_size = os.path.getsize(file_path)
        logger.info(f"Uploaded file size: {file_size} bytes")
        # Process the uploaded file
        global query_engine
        query_engine = process_document()
        return json.dumps({"message": f"File '{filename}' uploaded successfully to {file_path}. Size: {file_size} bytes. You
can now ask your query."})

def ask_query(query):
    question = (
        f"You are a JSON parser which gives response in extracting and providing information from "
        f"invoice PDF documents in response to user queries. The system accepts user questions "
        f"regarding specific invoice details and returns the relevant data in a standardized JSON format. "
        f"Make sure if the user asks multiple queries, then the output will be in new lines. Do not give "
        f"output in a single line output.\n The user query is: {query}"
    )

    response = query_engine.query(question)
    # Format the response in the desired format
    response_text = str(response)

    # Convert the JSON-like response string into a proper format with new lines
    response_json = response_text.replace("\\n", "\n")

    return response_json

# Create the Gradio interface
with gr.Blocks() as demo:
    gr.Markdown("# PDF Upload and Query System")

    with gr.Tab("Upload File"):
        file_input = gr.File(label="Choose a PDF file to upload")
        upload_output = gr.Textbox(label="Upload Result")
        upload_button = gr.Button("Upload")
```

```
    with gr.Tab("Ask a Query"):
        query_input = gr.Textbox(label="Enter your query")
        query_output = gr.TextArea(label="Query Result", lines=10)
        query_button = gr.Button("Submit Query")

    upload_button.click(upload_file, inputs=file_input, outputs=upload_output)
    query_button.click(ask_query, inputs=query_input, outputs=query_output)

# Launch the interface
demo.launch(debug=True)

logger.info("Gradio interface launched")
```

## Expected Outcomes

This project aims to deliver the following outcomes:

- **Automated Data Extraction:** A robust system capable of automatically extracting key details from invoices in various JSON formats.
- **Improved Accuracy with Generative AI:** Increased accuracy and reliability of data extraction through the integration of Generative AI models like Gemini.
- **User-Friendly Data Access:** A user-friendly interface that empowers users to efficiently query and retrieve invoice details in a structured JSON format.
- **Standardized Data Format:** Consistent and structured JSON output for seamless integration with other systems and applications.

## Conclusion

By leveraging AI and machine learning techniques, this project has the potential to significantly improve the efficiency and accuracy of invoice data extraction. This, in turn, can streamline financial processes and enhance data management capabilities for businesses. The project's emphasis on user-friendliness and standardized output further ensures the practical application of this technology in real-world scenarios.