**BUAN6390 – Final Post-Project Documentation**

**NOTE:** *Kindly note that some sections may be split into Part 1 and Part 2 to specify which part of our project the information pertains to. While Part 1 and Part 2 use the same exact chat bot, the capabilities built on top of the Chatbot are different in each part.*

* *PART 1 pertains to the csv reading AI (Artificial Intelligence) generative model, and*
* *PART 2 pertains to the web-scraping AI (Artificial Intelligence) generative model.*

**Team Name: *Team 5***

**Client Name: *Fintekera***

# 1. Statement of Group Contribution

**Prepared by:** *DAVID LEO*

|  |  |  |  |
| --- | --- | --- | --- |
| **Group Member** | **Role** | **Level of Contribution**  0=No Contribution  1=Weak Contribution  2=Equal Contribution  3=Strong Contribution | **Statement of Work** |
| Dasari, Lakshmi Sandeep Reddy | Project Manager | 3 | Manages team member responsibilities to ensure that the final deliverable is high quality and on time. Requires leadership and strong interpersonal skills. |
| Gande, Avinash | Developer | 3 | Develops the technical solution as specified in the business requirements. Requires strong programming and problem-solving skills |
| Jacob, Shoun Abraham | Quality Assurance | 3 | Responsible for monitoring, cleaning, validating and profiling the data we are using to ensure that our model is accurate and reliable |
| Leo, David Alfred | Business Analyst | 3 | Acts as the liaison between the Client and the team. Translates client needs into business requirements. Requires strong analytical and communication skills, both verbal and written. |
| Pullenti, Shivani | Systems Architect | 3 | Works closely with the business/project analyst, project manager, developer(s),and other team members to design the technical structure of the project |
| Sayyaparaju, Charitha Srikari | Developer | 3 | Develops the technical solution as specified in the business requirements. Requires strong programming and problem-solving skills. |
| Vemparala, Vaishnavi | Systems Architect | 3 | Works closely with the business/project analyst, project manager, developer(s), and other team members to design the technical structure of the project |

# 2. Post-Project Documentation Overview

**EXECUTIVE SUMMARY**

***Author: SHOUN ABRAHAM JACOB***  ***Editors: DAVID LEO, AVINASH GANDE***

Fintekera provided us with a comprehensive data set that contains information about 99 unique zip codes. In this project, we created a generative AI tool that will provide the user with clear and concise answers to their dataset-related questions, in a narrative format. Such a tool will not only help the user save a significant amount of time but can also be easily used by individuals who are not familiar with methods used to quickly scan through large datasets to find data.

Fintekera also requested a web scraping capability in order to be able to scrape data from the web for this model to chat with. We were able to deliver this capability into our model in “Part 2” of this project, alongside two different API data sources related to the Fintekera’s jobs data objectives. Our model is also capable of answering general questions, through the SERP API using Google homepage, or through the OpenAI API, where one can ask any question of Chat GPT.

**PROJECT OBJECTIVES**

***Author: LAKSHMI SANDEEP REDDY DASARI***  ***Editors: DAVID LEO, SHOUN ABRAHAM JACOB***

Part 1: ZIP Code Specific Questions and Summaries

Business Question/Problem:

"How can we provide users with accurate and easily accessible information about specific ZIP codes, encompassing economic, social, demographic, climate, and crime data?"

Project Objectives:

1. Develop a Generative AI Model for ZIP Code Queries:

* Objective: To create a lightweight AI model that efficiently responds to user queries about specific ZIP codes using relevant data points from the comprehensive dataset.

2. Implement Table-to-Text Generation for Summaries:

* Objective: Utilize Table-to-Text Generation techniques to generate cohesive and informative summaries for each ZIP code, weaving together social, demographic, and economic attributes from the provided Excel file.

3. Ensure CPU Efficiency:

* Objective: Design and optimize the AI model to run efficiently on a CPU, ensuring accessibility for users without high-end computational resources.

Part 2: General Question Answering Model

Business Question/Problem:

"How can we develop a lightweight AI model capable of addressing general questions without the need for a specific dataset, using publicly available online resources?"

Project Objectives:

1. Develop a Generative AI Model for General Questions:

* Objective: Create an AI model capable of answering general questions by leveraging publicly available online resources, demonstrating versatility in obtaining information.

2. Optimize for CPU Execution:

* Objective: Design and optimize the general question answering model for CPU execution, ensuring accessibility and usability across various computational setups.

3. Utilize Online Resources Efficiently:

* Objective: Establish mechanisms to effectively draw upon online resources, such as platforms like huggingface.co, to source accurate and up-to-date information for general questions.

4. Encourage Collaborative Approach:

* Objective: Foster collaboration by remaining open to insights and questions throughout the development process, ensuring a collective effort to achieve project objectives.

Overall Project Objective:

To deliver two lightweight generative AI models, one for ZIP code-specific queries and summaries, and another for general questions, both optimized for CPU execution and designed to provide accurate and accessible information to users.

**STAKEHOLDER ANALYSIS**

***Author: LAKSHMI SANDEEP REDDY DASARI*** ***Editors: AVINASH GANDE***

Our Stakeholder Analysis process was the 3 simple steps below:

1. Identification:

* Identify individuals or groups who are directly or indirectly affected by the project.
* Consider those who have an interest in the project's success or may impact its outcomes.

2. Prioritization:

* Prioritize stakeholders based on their level of influence, interest, and involvement in the project.

3. Assessment:

* Assess the expectations, concerns, and needs of each stakeholder.
* Consider the potential impact of the project on each stakeholder and vice versa.

Identified Stakeholders:

1. Project Team:

* Developer, Business Analysts, Systems Architect, Quality Assurance, project managers.
* Interest: Successful development and implementation of models.
* Influence: Directly involved in the project's day-to-day activities.

2. Management Team:

* Role: Project sponsor (Fintekera).
* Interest: Alignment with organizational goals, successful project completion.
* Influence: Decision-makers and resource allocators.

3. End Users:

* Role: Individuals using AI models for information retrieval.
* Interest: Accurate and accessible information about ZIP codes and general questions.
* Influence: Usage patterns and feedback can impact model improvements.

4. Data Providers:

* Role: Suppliers of data.
* Interest: Effective and ethical use of their data.
* Influence: Data quality and access.

5. General Public:

* Role: Those impacted by AI-generated information.
* Interest: Ethical use of AI, accurate information, accessibility.
* Influence: Public perception can affect the Fintekera’s reputation.

**TECHNOLOGY PLATFORM**

***Author: AVINASH GANDE***  ***Editors: DAVID LEO, VAISHNAVI VEMPARALA***

In the brief for the project, our sponsor had mentioned Python as one of the preferred languages to use. Therefore, we use Python for the creation of both parts of our model. The main python libraries we use include:

*PART 1*

* Tkinter (filedialog, messagebox)
* Pandas (read\_csv)
* Threading
* Langchain (agent\_toolkits, chat\_models, agent\_types)

*PART 2*

* Langchain
* OpenAI
* Requests
* BeautifulSoup
* Urilib

Apart from Python, we also used SERP (Search Engine Results Page) API and Back4app API as data sources to web scrape job-related data.

**DATA COLLECTION AND CLEANING**

***Author: SHOUN ABRAHAM JACOB*** ***Editors: VAISHNAVI VEMPARRALA***

PART 1

* For the ‘csv’ reading capability of the model, we were provided with a comprehensive dataset that included over 30 sheets of data and supporting summaries/visualizations. To make a csv file that could be used as input for our model, we compiled all the sheets containing data into one sheet, using the ZCTA (Zip Codes) column as a primary key. This was done using the *VLOOKUP* function in Excel.
* Since there were several cells without data, we used Python to add ‘Null’ values and clean the data. Other than that, the dataset we received was clean and all the data in the tables were uniform.
* Since we were using GPT 3.5 to develop our model, it is only capable of reading a csv file with fewer than 4000 tokens. Given that the dataset we have has 10,000+ tokens (30 sheets), we split the dataset into 3 csv files (data from 10 sheets in each file) to ensure that all the data can be used in our model.

PART 2

In Part 2 of our model, the data collection can be done in multiple ways. The primary way is by utilizing an API key and the various capabilities of our model to scrape from Google Jobs data, but we have also built the capability to scrape data from Back4app, any link the user provides, or Google itself. The data is then simply transformed into string data, which is the only requirement for our model to be able to chat with the web scraped data. For general question asking using the model, one does not need to collect or clean any data but can use the section labeled General Question Chatbot to simply ask any question and receive answers similar to those given by Chat GPT.

**DATA MODELING**

***Author(s): AVINASH GANDE, DAVID LEO*** ***Editors: SHOUN ABRAHAM JACOB***

PART 1

The script is a basic GUI application for querying a CSV file using Tkinter in Python. Here is an overview of its components and functionalities:

* Imports and Setup:
  + Imports Tkinter, Pandas, and other necessary modules.
  + Sets up the LangChain's create\_pandas\_dataframe\_agent with OpenAI's GPT-3.5 model for querying the DataFrame.
  + An OpenAI API key environment variable is set (placeholder provided).
* Function load\_csv\_file:
  + Opens a file dialog to select a CSV file.
  + Reads the selected CSV file into a Pandas DataFrame (df).
  + Creates a LangChain agent (agent) for querying the DataFrame.
  + Handles file loading errors and displays appropriate messages.
* Function execute\_query:
  + Retrieves the user's query from an entry widget.
  + Runs the query on the DataFrame using the LangChain agent.
  + Displays the result in a text widget or shows an error message if the query fails.
  + Uses threading to run the query without blocking the GUI.
* GUI Setup:
  + Creates the main window (root) with the title "CSV Query Application".
  + Adds a button (load\_button) to load a CSV file.
  + Provides an entry widget (entry) for typing queries.
  + Adds a button (execute\_button) to execute the query.
  + Includes a text widget (text) to display query results.
* Event Loop:
  + Starts the Tkinter event loop to display and operate the GUI.

### Usage:

* Load CSV: Click the "Load CSV File" button to select and load a CSV file.
* Enter Query: Type a query in the entry box.
* Execute Query: Click the "Execute Query" button to run the query and see results in the text area.

### Notes:

* Ensure you have the necessary Python packages (tkinter, pandas, langchain\_experimental, langchain) installed.
* Replace "Insert your openAI API Key here" with a valid OpenAI API key.
* The script uses threading to prevent the GUI from freezing during query execution.
* Error handling is included for file loading and query execution processes.

PART 2

Part two deploys the same model as used in part 1, except for it deploys it on data retrieved from the web scraping capability we built using one of multiple different data sources or taking data directly from Chat GPT via the OpenAIAPI.

* Using Requests and Beautiful Soup, the data is scraped from the website and then transformed into String format.
* We set up multiple capabilities, where the user can bring their own links for link & sub link scraping. This process uses the above libraries to scrape data from the specified links and then simply transforms the data into string format for the model.
* When using SERP API/ Back4app API, one can return results from Google Jobs pages, enter any search into google and then return the home page results, or tap into an extensive jobs database in Back4app API. Similar to the other web scraping processes, the data modeling here is simple, the data is simply transformed into string data after being captured using the various APIs.
* When using the general question asking function, all the data modeling is already done and the code doesn’t need to do anything extra.

**EVALUATION METRICS**

***Author:*** ***SHIVANI PULLENTI*** ***Editors: AVINASH GANDE***

Given the nature of our generative AI model, traditional statistical performance measures are not applicable. However, we assessed the model's outputs with responses generated by humans, providing a practical assessment of its effectiveness. We also did extensive research and tested out multiple different chatbot technologies, and both our experiments and research lead us to conclude that the OpenAI Chatbot is the most effective model available on the market today.

**VISUALIZATION**

***Author: VAISHNAVI VEMPARALA*** ***Editors: DAVID LEO***

This project, focusing on generative AI models and web scraping, does not incorporate any visualization components. The primary objective is to enhance backend functionalities and data processing, emphasizing the development of an efficient AI model and accurate web scraping methods without the need for graphical representations. But we have provided Fintekera with a visual walkthrough of sorts in the form of Jupyter Notebooks.

**INTERPRETATION AND INSIGHTS**

***Author(s): SHOUN ABRAHAM JACOB/ DAVID LEO*** ***Editors: CHARITHA SRIKARI SAYYAPARAJU***

*PART 1*

Although our model returns answers to any question that is input by the user, there are two main points that the user must keep in mind while using it:

* Cleaning the data is crucial to ensuring the smooth functioning of the model. The more you can clean the data (standardize all the data, get rid of empty cells), the better results you will receive from the model. The Open AI model is the best available model on the market as of today, so if answers are not coming through optimally the only thing that can be done is on the side of the Datasource.
* There is a certain syntax that should be used to ensure the proper output from the model. For example, if we look at a sample question: “what are the 5 highest crime\_index\_value. show only the ZCTA,” the underlined part of the statement is an addition to make sure that you get the exact output you are looking for. Although the user can simply phrase a question normally and get the answer, the second portion will ensure that the model will not return any errors.
* More natural language heavy datasets can have answers delivered without taking such precautions.

*PART 2*

Our model is built to have multiple capabilities, the insights into these capabilities for Part 2 are as follows. There are three basic capabilities built in Part 2, web scraping specified links, web scraping APIs, and using the model itself to ask either only your data a question, OpenAI a question (returning the results equal to the query of Chat GPT) or using both your data and OpenAI to answer the user's question.

* In terms of the general question answering capability, the model’s settings are setup to answer questions of the web scraped data in one section of our model, but using the General Questions Chatbot section in our model, one can ask questions and have results returned that are the same as if you had asked both ChatGPT and your data source question.

The key takeaway for Fintekera regarding Part 2 is that given our model and data sources, Fintekera now has the tools to have a dynamic web scraping model which can create and query a large database scraped from the web or query the data and ChatGPT itself at the same time.

**VALIDATION**

***Author: DAVID LEO*** ***Editors: LAKSHMI SANDEEP REDDY DASARI****.*

*PART 1*

For our first model, we validated our output (the answers from our model) using the following methods:

* For all the questions we ran in through our model, we used Python code to ensure that the answers we were getting were accurate.
* We input other csv files with different kinds of data into our model and asked it questions to make sure that we were still getting accurate answers. The results were favorable.

*PART 2*

For part 2, we validated the results by iteratively testing the model on various levels of parsed data, in order to simplify the process and allow the user to be able to simply show up and query the data rather than having to specify parameters unique to each individual website before scraping their desired links/our data sources provided. We also tested the general question chatbot extensively to ensure that it could indeed answer general questions.

**ETHICS AND PRIVACY**

***Author: SHIVANI PULLENTI*** ***Editors: CHARITHA SRIKARI SAYYAPARAJU***

* The only privacy consideration that is applicable to our project relates to the dataset provided by Fintekera.
* All our group members signed NDAs prior to receiving the dataset.
* The main precaution we took was to ensure that we did not share the file using any unsecure platforms, and if shared, it was only with team members via outlook, a secure platform.

**TRANSITION PLAN**

***Author: VAISHNAVI VEMPARALA*** ***Editors: CHARITHA SRIKARI SAYYAPARAJU***

**Access to Project Resources:** The project, including the code files for Part 1 and Part 2, along with the necessary documentation, will be securely uploaded to a dedicated GitHub repository. Clients will be granted exclusive access via a provided link, ensuring full permissions to view, download, and interact with these files. Jupyter Notebook files will also be emailed to the client.

**Client Requirements:** Clients are not required to obtain any additional items or licenses for accessing the GitHub repository. The provided files, including Jupyter notebooks containing the code, will be available without any restrictions. The client must create a free OpenAI account, SERP API account, and Back4app account in order to use the various sections of the model. Instructions regarding this are below in the login section.

**Project Transition Process:** To facilitate a smooth transition, the following steps will be taken:

* **Delivery of Access Credentials:** The client will receive credentials and a link to access the GitHub repository.
* **Access to Code and Documentation:** All necessary code files and documentation will be available in the repository for immediate access.
* **Guidance on File Utilization:** Detailed guidelines on how to utilize the Jupyter notebooks and other provided files will be included, ensuring the client can effectively leverage the project's resources.
* **Support for Transition:** Our team will be available to assist with any queries or support needed during the transition phase to ensure a seamless handover.

This plan is designed to ensure that the client can effectively integrate and utilize the project deliverables with minimal disruption and maximum efficiency.

# 3. Getting Started

**LOGGING-IN CREDENTIALS (IF NEEDED)**

***Author: DAVID LEO*** ***Editors: VAISHNAVI VEMPARALA***

The user must have an OpenAI API key to run both models, a SERP API key and a Back4app API key to run the part 2 section of our model (if not using their own desired links). Other than that, there or no other login credentials required to use our model.

These API keys can be generated for free at the below links:

<https://openai.com/blog/openai-api>

<https://serpapi.com/google-jobs-results>

<https://www.back4app.com/database/back4app/occupations-and-job-titles/get-started/python/rest-api/requests#customize-your-query>

**TIPS FOR WHAT TO AVOID**

***Author: DAVID LEO*** ***Editors: VAISHNAVI VEMPARALA***

When using the link/sub link scraping section of Part 2 of the model, the user must specify the depth parameter they desire to stop the model from scraping indefinitely. Instructions are given directly in the code that describe how to understand the depth parameter for no one to accidentally slip over this issue. Also, if one wishes to use the Back4app API, they need to specify the limit parameter to specify the amount of job description data they wish to download from the database.

# 4. Implementation Details

Our model is setup to run in Python with minimal maintenance. For Part 1, one can simply open the Jupyter Notebook provided or run the code from the GitHub link. For Part 2, because the model has multiple different capabilities, one must find the capability they desire to employ using the headers provided in the GitHub and Jupyter notebooks before utilizing the model. After finding the relevant section, the model can be used by simply clicking run or specifying the required parameters, depending on the section.

Also, one must have an API key specified in the areas of the model listed as “API KEY” which they obtain from creating login credentials using the links above.

**SYSTEMS ARCHITECTURE (AS MUCH AS NEEDED)**

***Author: AVINASH GANDE*** ***Editors: VAISHNAVI VEMPARALA***

* **tkinter** and **filedialog**: These are modules from the Tkinter package, a standard GUI toolkit in Python. **tkinter** is used for creating the GUI components, while **filedialog** allows for opening file dialog windows to select files.
* **pandas**: This is a powerful data manipulation and analysis library in Python, used here for reading and handling CSV files.
* **messagebox**: Part of Tkinter, used to display informational, warning, and error messages to the user.
* **threading**: This module is used to run tasks in separate threads, which prevents the GUI from becoming unresponsive during long-running operations.
* **langchain\_experimental.agents.agent\_toolkits** and **langchain.chat\_models**: These are from the LangChain library, which integrates with OpenAI's models. They are used to create an agent that can query Pandas DataFrame using natural language processing.
* **os**: Used for setting the environment variable for the OpenAI API key.
* **os.environ["OPENAI\_API\_KEY"]**: Sets the OpenAI API key as an environment variable. This key should be replaced with an actual API key.

**ACCESS REQUIREMENTS (IF APPLICABLE)**

***Author: VAISHNAVI VEMPARALA*** ***Editors: DAVID LEO***

There are no specific access requirements for this project, as all necessary materials and information will be readily available through our designated channels.

* **GitHub Repository Access:** Clients will receive a link to a specific GitHub repository. This repository contains all the relevant project files.
* **Permission Levels:** Upon receiving the link, clients will be granted appropriate permissions to view, download, and interact with these files as needed for the project.
* However, Users must possess an OpenAI API key for both models, along with SERP and Back4app API keys for the second part of our model. No additional login credentials are necessary for usage.

**CUSTOMIZED FUNCTIONALITIES (IF APPLICABLE)**

***Author:*** ***VAISHNAVI VEMPARALA*** ***Editors: LAKSHMI SANDEEP REDDY DASARI***

In relation to the queries about custom functionalities, modifications to existing code, themes, or plugins, and details of file names and locations of such changes, we confirm that our project does not encompass any such elements. All components utilized in our project are original and have not required alterations to existing frameworks or the introduction of external scripts or code. Therefore, there are no references or details pertaining to such changes applicable to our project.

**FILE STRUCTURE (IF APPLICABLE)**

***Author: SHOUN ABRAHAM JACOB*** ***Editors: AVINASH GANDE***

For the first model, since we used GPT 3.5, we were not able to run our original dataset through it successfully. We were limited to being able to use datasets with fewer than 4000 tokens (while the original had 10,000+ tokens), so we had to split the dataset into 10 parts. All 10 parts can be found in the GitHub suppository.

**DATABASE SCHEMA (IF APPLICABLE)**

***Author:*** *N/A* ***Editors:*** *N/A*

This section is not applicable to our project

# 5. **Issues and Concerns**

* Fundamentally, our model has accomplished Fintekera’s requirements, although it is not perfect. Because of the paywall requirements of OpenAI, we are not capable of running queries against the client’s entire dataset for Part 1 due to constraints imposed by OpenAI to push one towards a payment plan.
* Secondly, SERP API, the primary data source we selected for Part 2, also has a paywall as well that kept us from being able to run the full list of job titles the client provided thus creating a massive jobs database that the client could query from in a turnkey fashion. As we stated in our presentation and slides, if we could go back, we would allocate more time to finding data sources and capabilities without these limitations.
* The good news is that these shortcomings can be overcome by climbing the paywalls imposed, and if Fintekera chooses, they will be able to then use SERPAPI and our chatbot to create a massive database from Google Job postings data or Google itself using the capabilities in the model. Also, using the general question asking capability in the General Chatbot section of the model, one can receive any answer they desire from Chat GPT via the OpenAI API. Therefore, this section can be useful if Fintekera wishes to use this to answer job related questions instead of creating their own large jobs database.