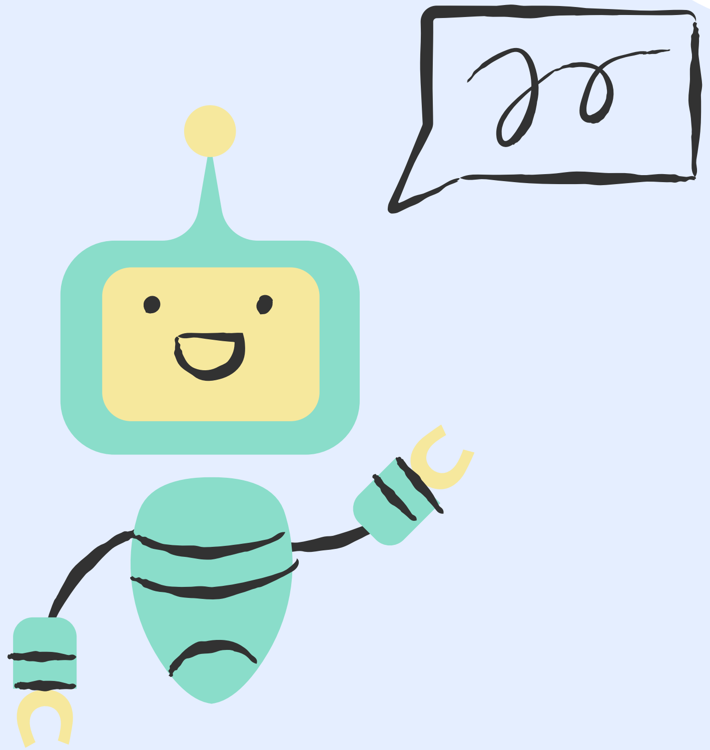
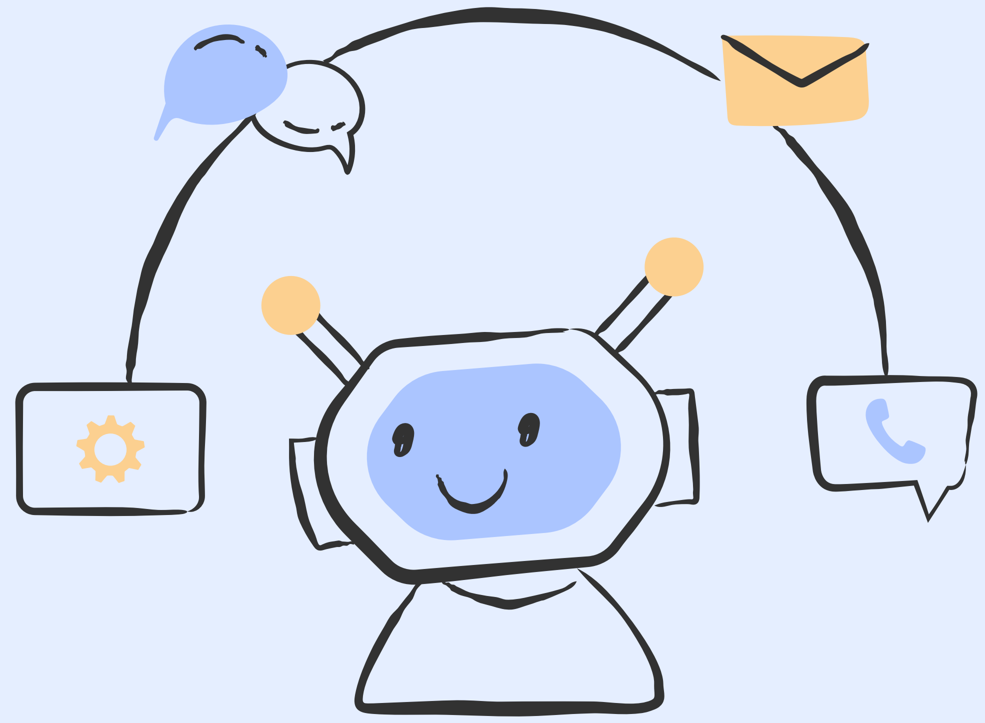
NNDSS Data Pipeline Project

Group 3





Objective

01

Anomaly detection to identify spikes or errors.

02

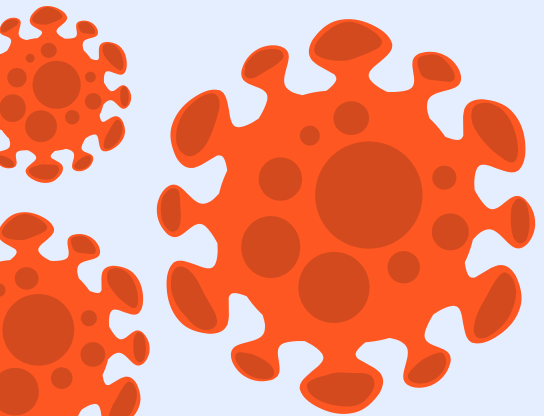
Trend analysis for disease monitoring.

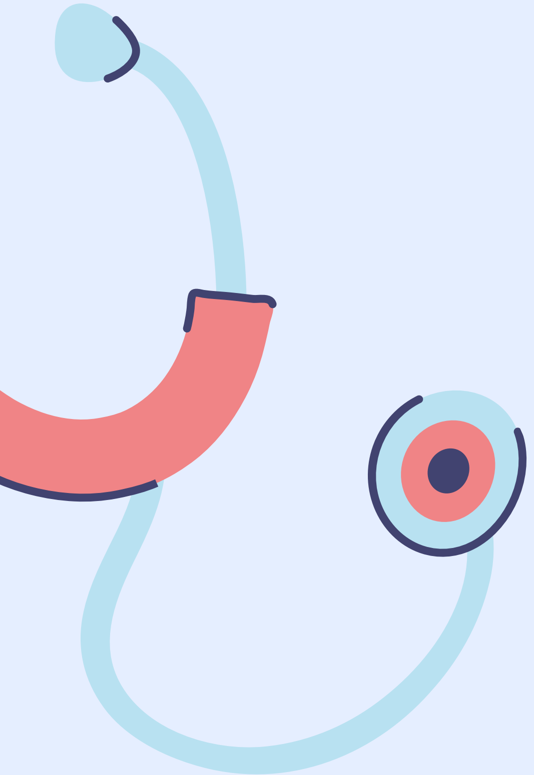
03

Automating processes for efficiency.

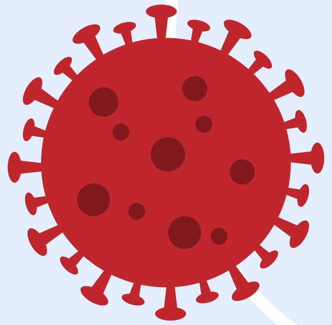
04

Enhancing user interaction with an AI-driven chatbot.



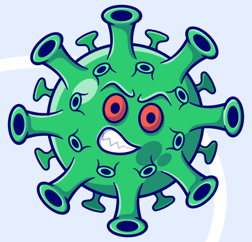


Understanding the data



Dataset

- NNDSS dataset by the CDC which updates every week
- Spans from 2022 onwards, and is organized by weekly reports (MMWR weeks)
- Helps detect disease outbreaks, monitor trends, and guide public health responses.



ER Diagram

Source:

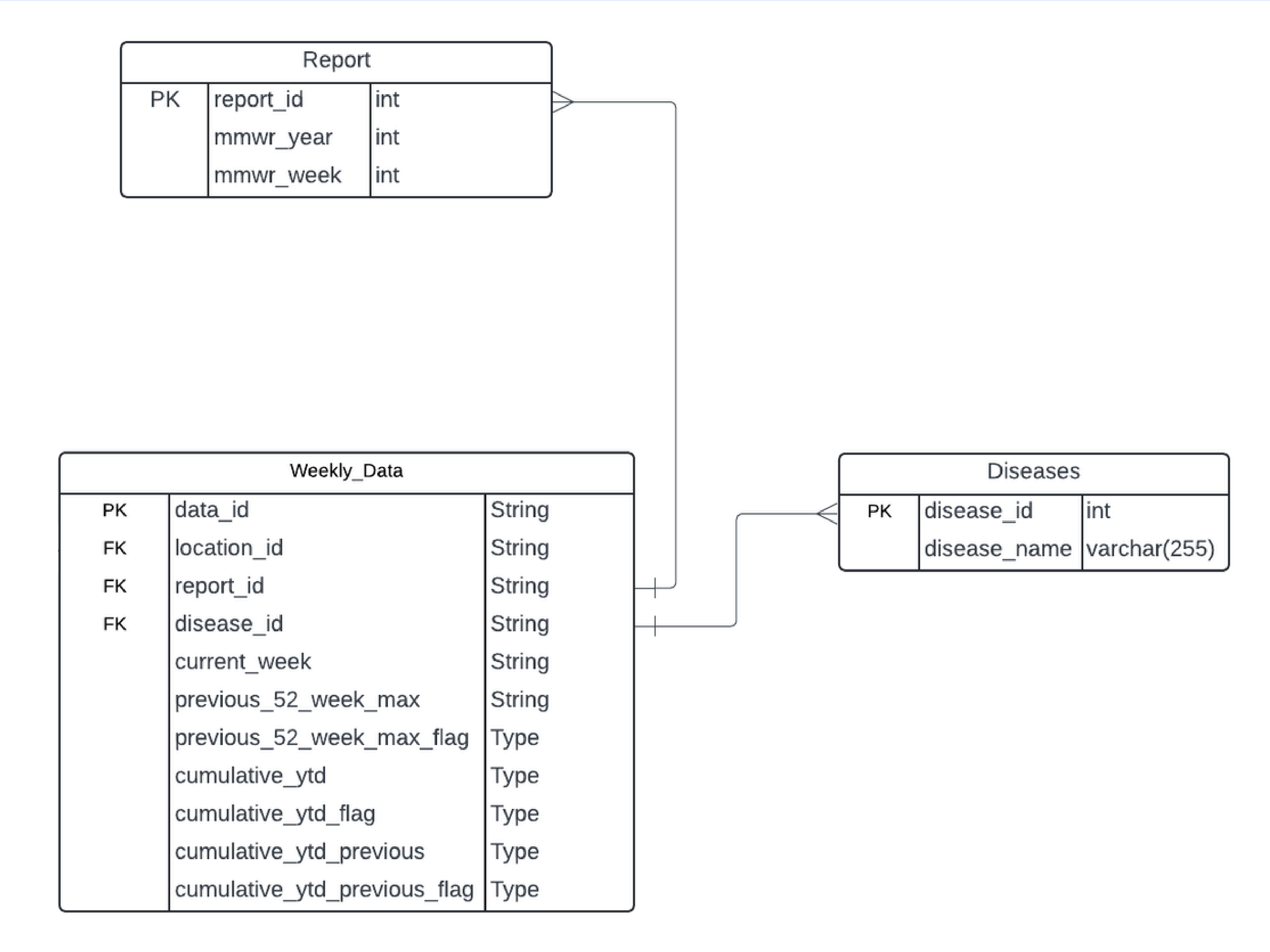
CDC NNDSS API (updates weekly)

Size:

Rows: 1.12 million, Columns: 16

Data Type:

Structured data in JSON format



Report

PK	report_id	int
	mmwr_year	int
	mmwr_week	int

Weekly_Data

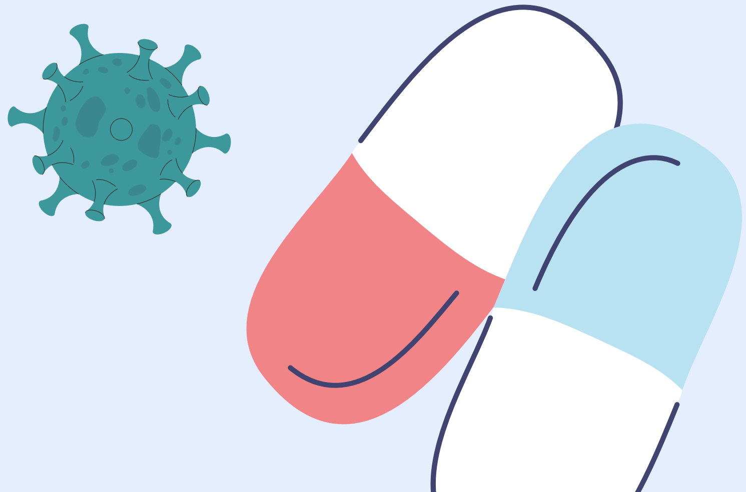
PK	data_id	String
FK	location_id	String
FK	report_id	String
FK	disease_id	String
	current_week	String
	previous_52_week_max	String
	previous_52_week_max_flag	Type
	cumulative_ytd	Type
	cumulative_ytd_flag	Type
	cumulative_ytd_previous	Type
	cumulative_ytd_previous_flag	Type

Diseases

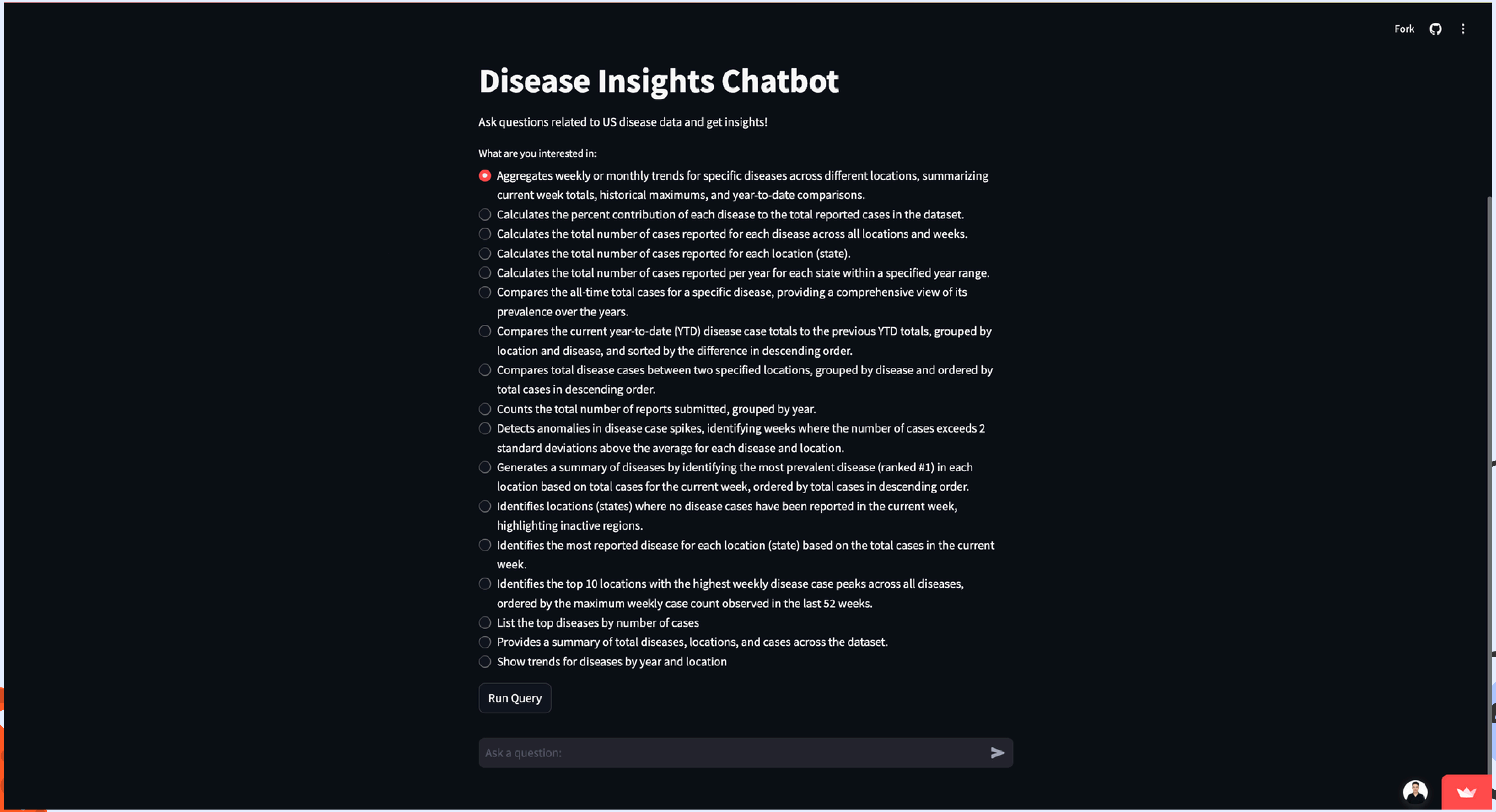
PK	disease_id	int
	disease_name	varchar(255)

Data Workflow Recap

1. Data Extraction from website using API calls and Postman
2. Data transformations using BigQuery
3. EDA
4. Set up Google Cloud Schedular
5. Interactive Dashboard using Tableau
6. Trend Analysis
7. Anamoly Detection



Chatbot using Streamlit



Disease Insights Chatbot

Ask questions related to US disease data and get insights!

What are you interested in:

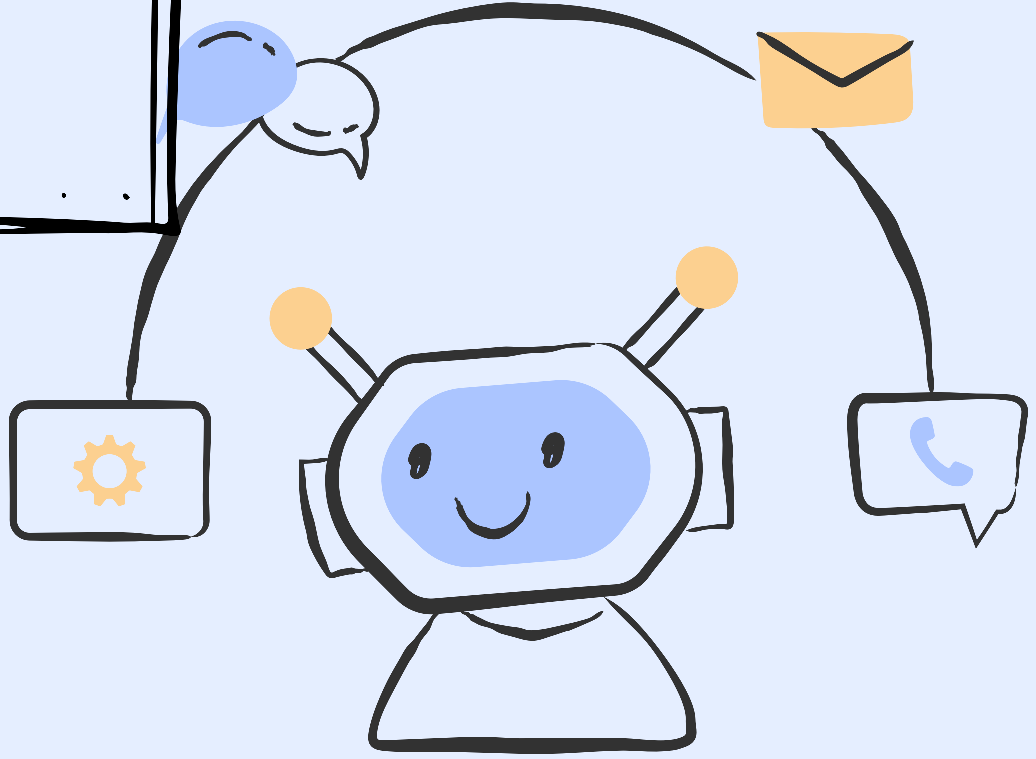
● Aggregates weekly or monthly trends for specific diseases across different locations, summarizing current week totals, historical maximums, and year-to-date comparisons.
○ Calculates the percent contribution of each disease to the total reported cases in the dataset.
○ Calculates the total number of cases reported for each disease across all locations and weeks.
○ Calculates the total number of cases reported for each location (state).
○ Calculates the total number of cases reported per year for each state within a specified year range.
○ Compares the all-time total cases for a specific disease, providing a comprehensive view of its prevalence over the years.
○ Compares the current year-to-date (YTD) disease case totals to the previous YTD totals, grouped by location and disease, and sorted by the difference in descending order.
○ Compares total disease cases between two specified locations, grouped by disease and ordered by total cases in descending order.
○ Counts the total number of reports submitted, grouped by year.
○ Detects anomalies in disease case spikes, identifying weeks where the number of cases exceeds 2 standard deviations above the average for each disease and location.
○ Generates a summary of diseases by identifying the most prevalent disease (ranked #1) in each location based on total cases for the current week, ordered by total cases in descending order.
○ Identifies locations (states) where no disease cases have been reported in the current week, highlighting inactive regions.
○ Identifies the most reported disease for each location (state) based on the total cases in the current week.
○ Identifies the top 10 locations with the highest weekly disease case peaks across all diseases, ordered by the maximum weekly case count observed in the last 52 weeks.
○ List the top diseases by number of cases
○ Provides a summary of total diseases, locations, and cases across the dataset.
○ Show trends for diseases by year and location

Chatbot Objectives

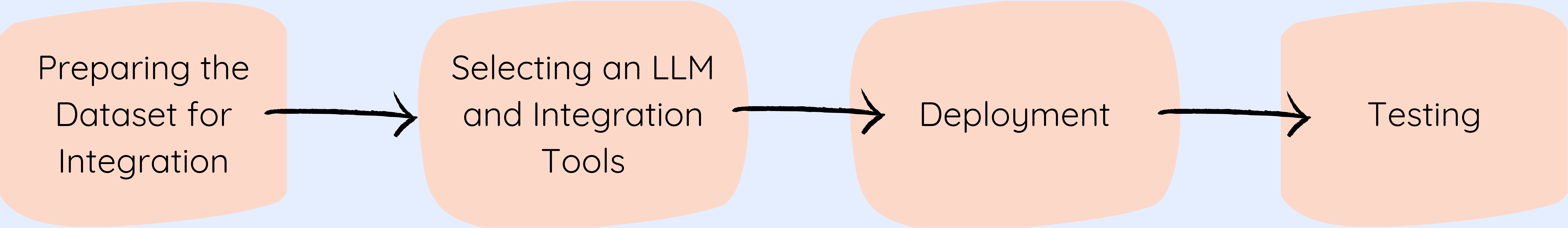
Pre-defined questions for BigQuery queries.

Freeform chat box for user-typed queries.

Fallback mechanism for unrelated questions.



Chatbot Creation Steps



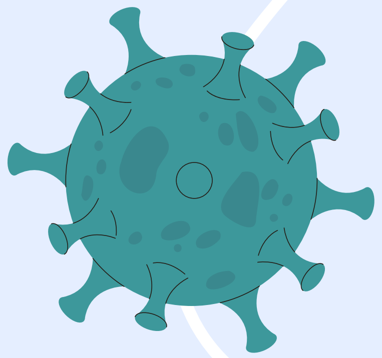
Preparing the Dataset for Integration

Selecting an LLM and Integration Tools

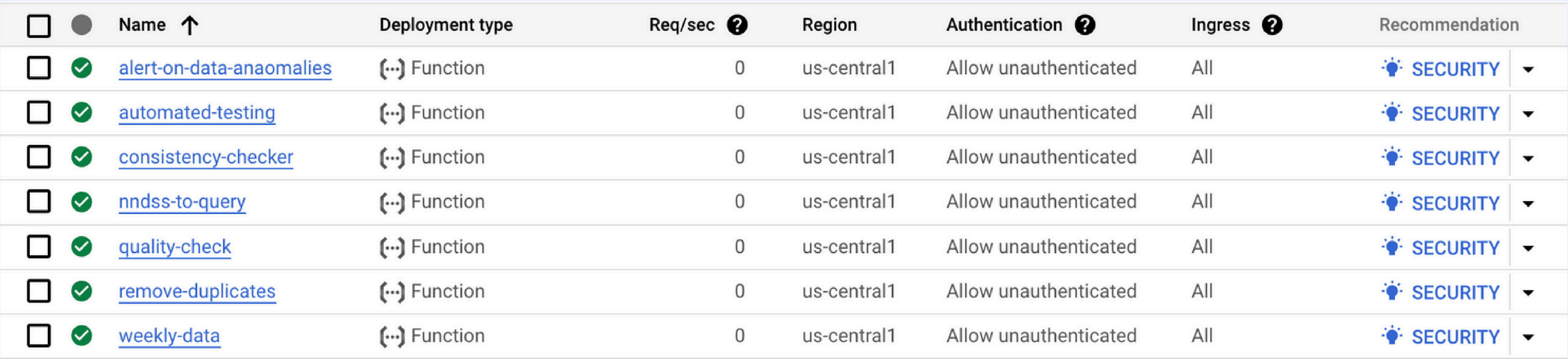
Deployment

Testing

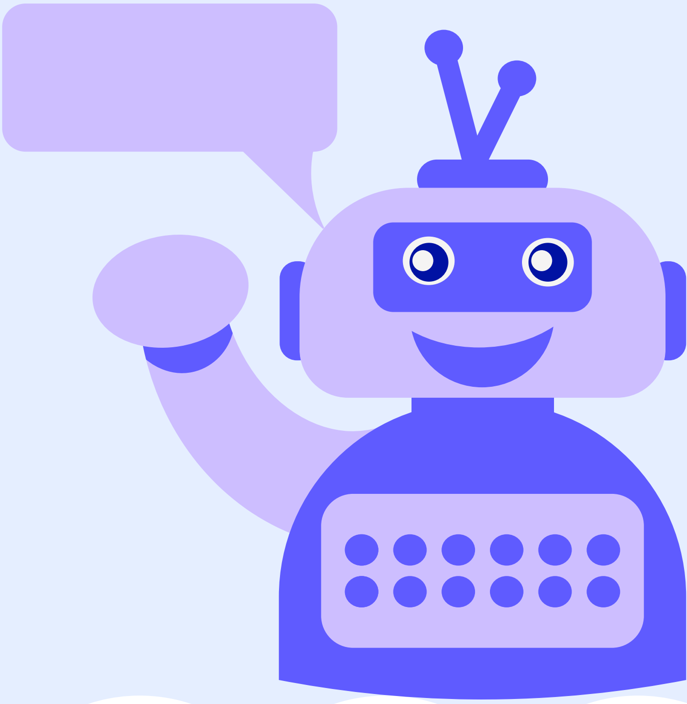
Data Preprocessing



- Ensure the dataset was clean and structured for querying
- We created multiple cloud functions from the start to make sure
- Stored the processed data in a format accessible to the chatbot



□	●	Name ↑	Deployment type	Req/sec ?	Region	Authentication ?	Ingress ?	Recommendation
□	✓	alert-on-data-anaomalies	{...} Function	0	us-central1	Allow unauthenticated	All	SECURITY ▾
□	✓	automated-testing	{...} Function	0	us-central1	Allow unauthenticated	All	SECURITY ▾
□	✓	consistency-checker	{...} Function	0	us-central1	Allow unauthenticated	All	SECURITY ▾
□	✓	nndss-to-query	{...} Function	0	us-central1	Allow unauthenticated	All	SECURITY ▾
□	✓	quality-check	{...} Function	0	us-central1	Allow unauthenticated	All	SECURITY ▾
□	✓	remove-duplicates	{...} Function	0	us-central1	Allow unauthenticated	All	SECURITY ▾
□	✓	weekly-data	{...} Function	0	us-central1	Allow unauthenticated	All	SECURITY ▾



LLM and Integration Tools

The chatbot is a user-friendly interface built with Streamlit that integrates OpenAI's GPT-4 model and Google BigQuery to provide insights into US disease data.

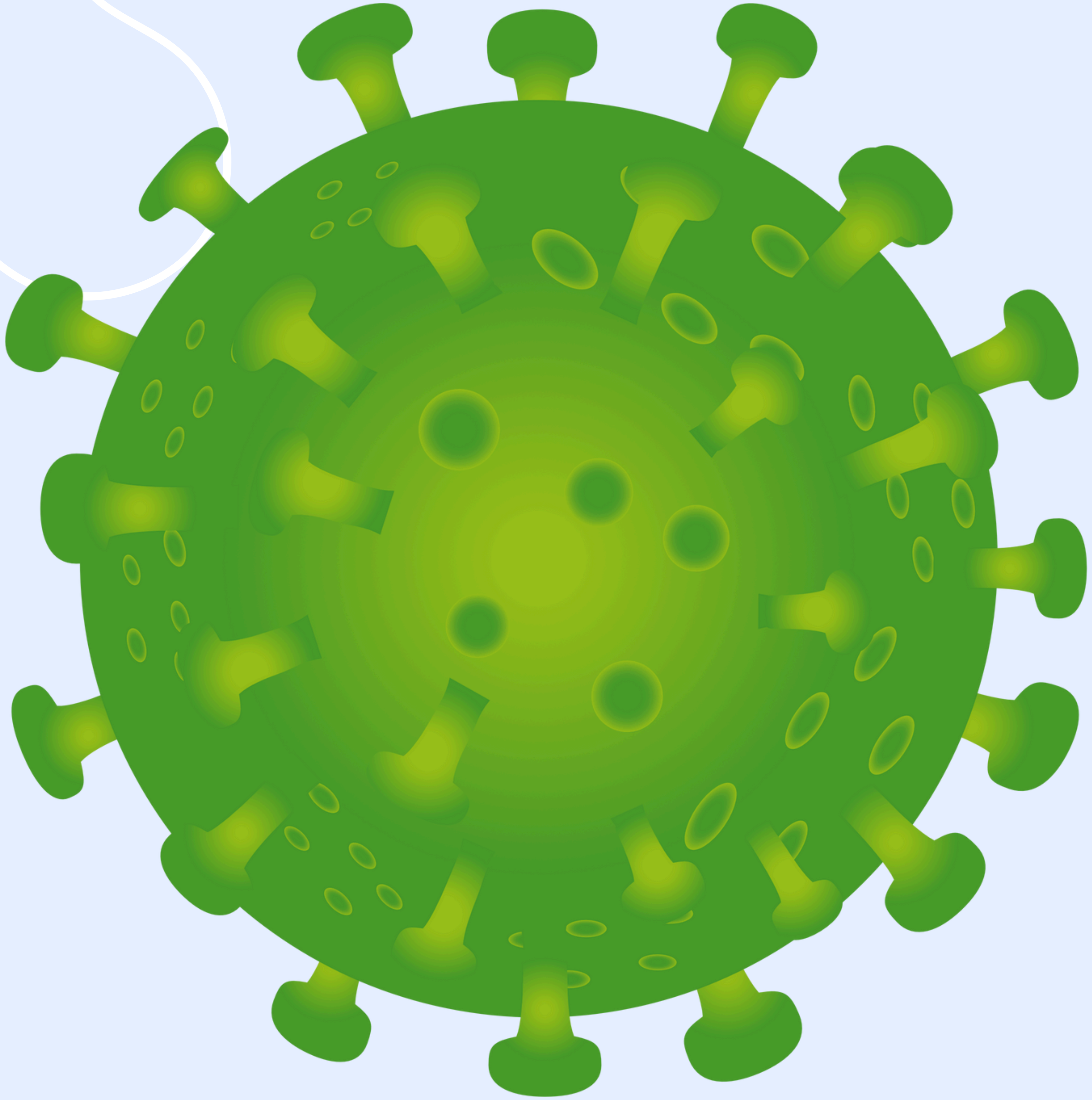
The Github Codespace serves as a cloud-based development platform where the chatbot application is built, tested, and refined.

Users can interact by selecting predefined queries or typing freeform questions about our disease data.

Designed to answer user queries and run predefined data queries, it offers both structured results and GPT-4-powered analyses

Deployment

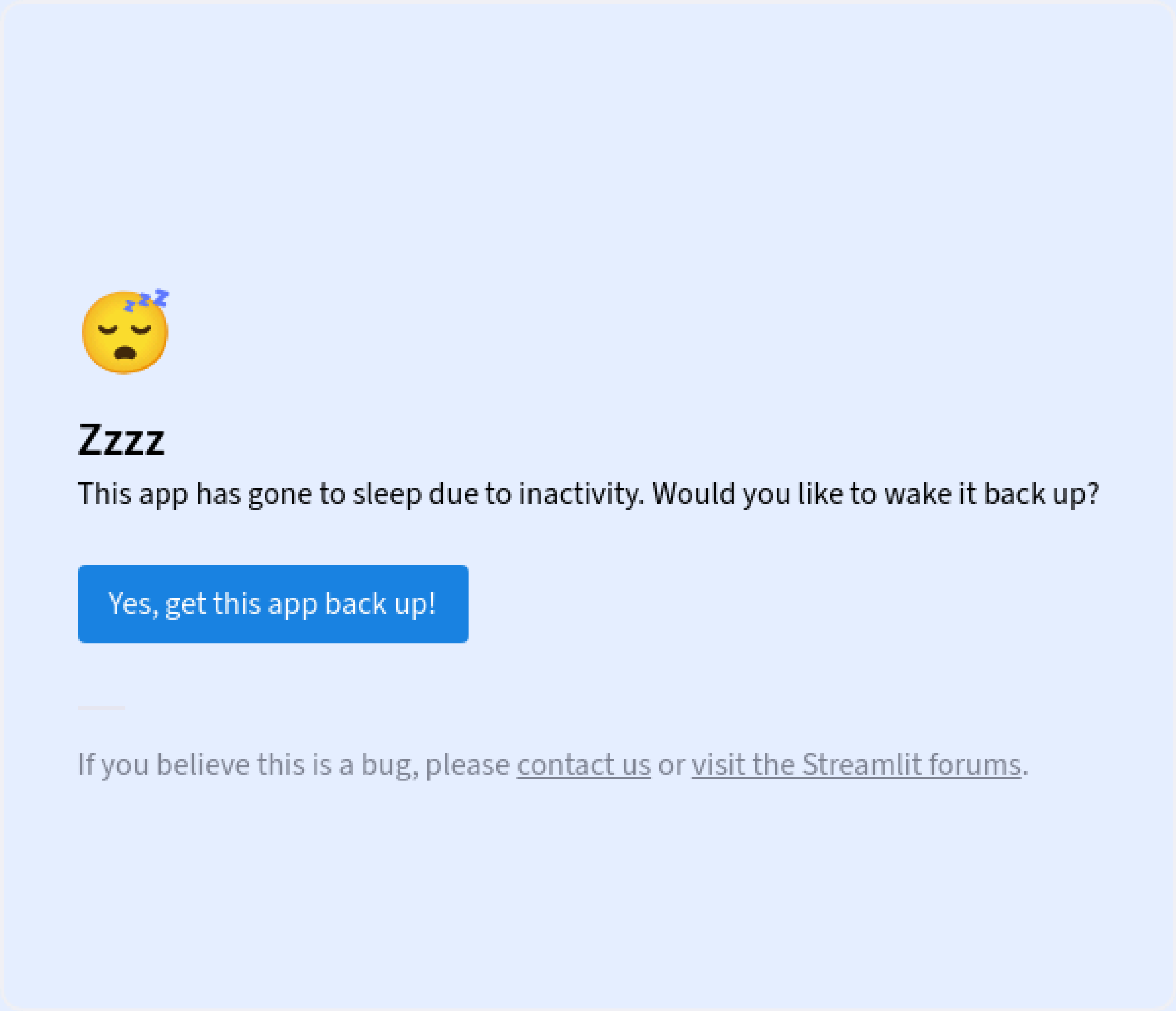
- Users must first provide their OpenAI API key to ensure secure interaction with OpenAI services.
- The app connects to BigQuery using a service account from a provided secrets file to query datasets stored in our GCP project.
- The users can choose some predetermined queries to run, the chatbot will run the chosen query and then provide the result table and a brief analysis.
- The users can also ask the chatbot questions about the database and it will construct the queries and provide the analysis.



Testing

- Testing and improving the chatbot involved validating both its querying capabilities and user interaction flow.
- Tests were conducted to ensure the BigQuery connection was stable, and mappings for disease and location IDs worked as intended.
- User feedback was incorporated to enhance the UI, such as displaying query results in a structured table format and limiting rows to the top 10 for readability.

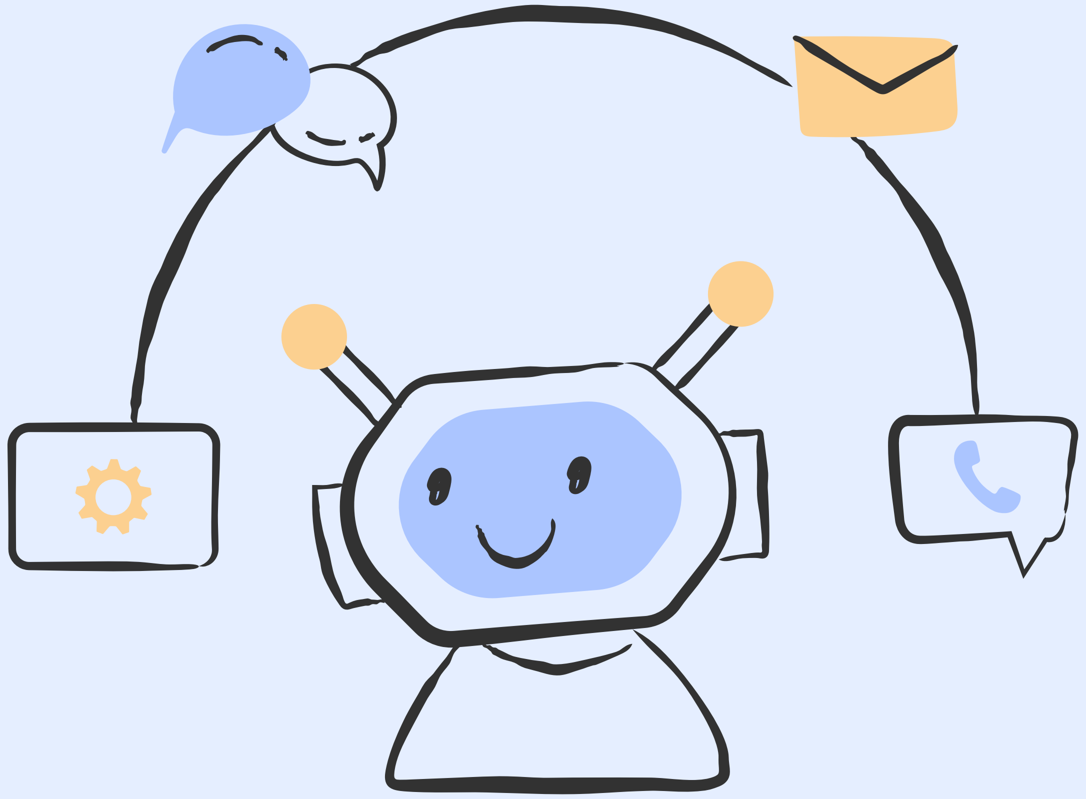
ChatBot Demo



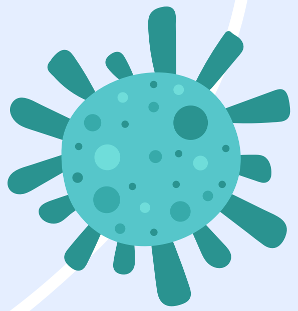
Zzzz

This app has gone to sleep due to inactivity. Would you like to wake it back up?

If you believe this is a bug, please contact us or visit the Streamlit forums.



Challenges

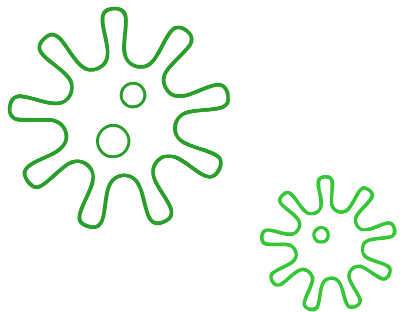


1 **Automation Scalability:** Ensuring automated processes scale seamlessly with growing data and workload demands.

2 **API Key Handling:** Security constraints required users to manually enter the OpenAI API key for each session.

3 **User Experience:** Enhancing the chatbot's understanding of diverse user queries and providing meaningful responses.

4 **Query Understanding:** The chatbot struggled to interpret user inquiries and generate relevant queries.



Business Applications

1 **Public Health Monitoring:** Detect outbreaks early and allocate healthcare resources effectively.

2 **Data Quality Assurance:** Ensure accurate and compliant health data reporting.

3 **Operational Efficiency:** Automate data workflows to save time and reduce manual effort.

4 **AI-Driven Insights:** Provide actionable insights through an intuitive chatbot for decision-making.

Thank You