Asklytics

An LLM for Website Analytics with Sentiment Analysis of Google Analytics 4 (GA-4) & Google Maps Reviews Data

Applied Research Project - COMP 4495

Section: 001

Team Lead

Tamoor Haider Aslam (300367290)

Team Members

Mohamed Nuskhan Mohamed Niyas (300368621)

Date of Submission:

13.04.2025

Introduction & Domain

In the era of data-driven decision-making, organizations heavily rely on website analytics to understand user behavior, enhance performance, and improve digital experiences. Google Analytics 4 (GA4) plays a central role in this process by offering detailed metrics such as bounce rates, user sessions, traffic sources, and engagement patterns. However, for non-technical users, interpreting these analytics can be overwhelming and inaccessible. This project aims to bridge that gap by leveraging conversational AI to make complex analytical insights easily understandable and actionable.

Background & Problem Context

Despite the powerful capabilities of web analytics platforms like GA4, the usability barrier prevents many business stakeholders from deriving full value from their data. To overcome this, the project introduces **Asklytics**, an intelligent AI-powered chatbot built using LangChain and OpenAI's gpt-4o-mini. Asklytics simplifies website data interpretation by allowing users to ask natural language questions and receive context-aware, visual, and sentiment-informed answers in real-time.

Research Objectives & Technical Contribution

The core goal of the research is to build a multi-functional, user-friendly analytics assistant that not only responds to web performance queries but also integrates sentiment analysis and predictive modeling. The key technical components of the system include:

1. LLM-Oriented Agent with Tool Use

Asklytics uses a LangChain agent framework that routes queries through structured tools. These tools allow the model to perform calculations, fetch search results via SerpAPI, and return final responses using a final_answermechanism. This approach ensures transparency, step-by-step reasoning, and modular extensibility.

2. Advanced Prompt Engineering

A custom prompt guides the chatbot to behave like a data analyst, enabling it to:

- Analyze website traffic and engagement trends
- Generate visualizations using Matplotlib or Plotly
- Perform sentiment classification (positive, neutral, negative)

Forecast future performance

3. Streaming with Async Tool Execution

The chatbot supports real-time streaming of responses via an asynchronous callback queue. This ensures fluid, interactive conversations that simulate human-like interactions even for long-running tasks like prediction or visualization rendering.

4. Search-Integrated Responses

Asklytics can pull external insights from the web using SerpAPI, allowing it to supplement internal analytics with the latest contextual information (e.g., market news, trends).

5. Custom Tool Chain & Execution Flow

A configurable and extensible toolset (e.g., add, multiply, serpapi, sentiment classifier) is available to the agent. Each tool runs asynchronously, ensuring smooth multitasking and real-time interactivity during inference.

Framing the Problem

Despite the increasing importance of data-driven decision-making in digital businesses, many users—especially non-technical stakeholders—struggle to effectively interpret and utilize insights from platforms like Google Analytics 4 (GA4). This research addresses several critical gaps in current web analytics practices:

1. **High Barrier to Entry for Non-Technical Users**
   GA4 provides a wealth of metrics—such as bounce rates, session durations, and traffic sources—but interpreting this data often requires technical expertise. The steep learning curve prevents business owners and marketers from independently drawing actionable insights.
2. **Lack of Conversational Interfaces in Analytics Platforms**
   Most web analytics tools present data through static dashboards and complex filters, which can be unintuitive for casual users. There is a notable absence of natural language interfaces that allow users to ask questions and receive human-like, context-aware responses.

3. **Limited Integration of Sentiment Analysis with Web Data**
   Traditional analytics focus on quantitative metrics while ignoring qualitative user feedback from sources like reviews or social media. This project seeks to bridge that gap by combining sentiment analysis with behavioral data, enabling a richer understanding of user perception and its real-time influence on website engagement.
4. **No Real-Time Predictive Intelligence**
   Existing tools are often retrospective, analyzing what has already happened. There is a need for accessible forecasting tools that predict user trends and performance metrics, helping stakeholders make proactive decisions based on emerging patterns.
5. **Fragmented User Experience Across Tools**
   Users often switch between multiple platforms (e.g., analytics dashboards, review monitoring tools, market research sites) to gather insights. This fragmentation slows down decision-making and introduces inconsistencies. Asklytics aims to consolidate these capabilities into a unified, interactive experience.

Existing Tools

Existing research and tools focus heavily on data visualization and automated reporting but rarely incorporate natural language interfaces tailored to specific analytics questions. By leveraging Natural Language Processing (NLP) models, sentiment analysis techniques, and predictive analytics, this project fills a critical gap in website performance analysis and feedback interpretation.

Literature Review

1. This paper highlights how integrating AI and data analytics is revolutionizing competitive intelligence, enabling businesses to gather, analyze, and act on market insights more eCectively. AI-driven technologies like machine learning, natural language processing, and predictive analytics allow organizations to uncover patterns, predict trends, and automate tasks like competitor monitoring and sentiment analysis. Real-time analytics and intuitive visualization tools further enhance decision-making and resource allocation, helping

businesses adapt dynamically to changing market conditions. While the adoption of these technologies presents challenges such as data privacy, integration issues, and skill gaps, the paper provides strategic recommendations to address them. Case studies from various industries demonstrate the benefits of AI-powered competitive intelligence, including driving innovation, improving customer engagement, and optimizing strategies

Jack, Harper. (2024). Leveraging AI and Data Analytics: Revolutionizing Competitive Intelligence for Market Insights. 10.13140/RG.2.2.25148.76166.

2. The rise of big data and advanced analytics has transformed decision-making, enabling organizations to shift from reactive to predictive strategies. Predictive analytics uses machine learning, AI, and statistical models to analyze historical and real-time data, uncover patterns, forecast trends, and optimize decisions. It is widely applied in industries like healthcare, finance, manufacturing, and retail, enhancing outcomes such as improved patient care, fraud prevention, and predictive maintenance. However, challenges such as data quality, integration complexities, and ethical concerns persist. This paper explores the transformative role, applications, challenges, and prospects of predictive analytics, emphasizing its importance in fostering resilience, sustainability, and competitive advantage in a data-driven world.

   Nyoni, Rumbidzai. (2025). Harnessing Data Analytics for Predictive Insights: Advancing Decision- Making with Big Data Innovations. International Journal of Research Publication and Reviews. 6. 2915-2936.

3. Financial institutions are increasingly using data lakes to manage vast and diverse datasets, enabling advanced fraud detection and prevention. These repositories store structured, semi-structured, and unstructured data, facilitating insights into large-scale financial information. By integrating advanced analytics techniques such as machine learning, anomaly detection, predictive modeling, and natural language processing (NLP), data lakes empower institutions to identify patterns, predict fraudulent behavior in real time, and take proactive measures. This paper examines the architecture and technologies behind financial data lakes, their practical applications in fraud prevention, and the challenges of implementation. It also provides a roadmap for successful adoption, emphasizing their role in safeguarding assets and maintaining customer trust.

Arena, Frank & Paul, Justin. (2024). INNOVATIVE ANALYTICS TECHNIQUES FOR FINANCIAL DATA LAKES.

Hypothesis, Assumptions & Benefits

Our hypothesis is that an intuitive, AI-driven chatbot combined with sentiment analysis will empower users to derive meaningful insights without requiring technical expertise. This project has the potential to enhance decision-making, reduce the time spent on manual analysis, and improve customer-centric strategies for organizations.

Summary of the research project

This research project introduces **Asklytics**, an AI-powered conversational assistant designed to democratize access to website analytics for non-technical users.
Leveraging **LangChain** and **OpenAI's gpt-4o-mini**, Asklytics interprets Google Analytics 4 (GA4) metrics through natural language queries, making data-driven insights more intuitive and actionable.

The project addresses the usability gap in traditional analytics platforms by integrating **LLM-driven agents**, **advanced prompt engineering**, **sentiment analysis**, and **predictive modeling**.

Asklytics supports real-time interactions through **asynchronous tool execution**, enabling dynamic responses with visualizations, forecasts, and web-integrated insights using SerpAPI.

Its modular architecture, featuring a custom tool chain for calculations, sentiment classification, and forecasting (ARIMA, Prophet), enables a scalable, transparent, and user-friendly approach to web performance analysis. This project contributes a novel solution that blends conversational AI with analytical intelligence, enhancing decision-making for businesses across technical skill levels.

Changes to the Proposal

Over the course of the Asklytics project development, several changes were made to the original proposal. These changes primarily impacted the **models used**, **technology stack**, and **overall architecture**. Each modification was based on practical challenges, technical constraints, and the evolving needs of the application. Below is a detailed explanation of these changes and the rationale behind them.

1. Changes in Machine Learning Models

Original Plan: Use open-source models (T5, FLAN-T5, DeepSeek R1, Mistral 7B, Phi-3-mini)

Initially, the plan was to utilize open-source language models to perform tasks like sentiment analysis, time-series forecasting, and data-driven query answering. The models explored included:

- **T5 and FLAN-T5** for text summarization and query response
- **DeepSeek R1** for processing unstructured financial news
- **Mistral 7B** for correlating sentiment with stock data
- **Phi-3-mini** for instruction-following and simple reporting

Change: Shift to OpenAI GPT-4 API

The final implementation replaced all the above models with OpenAI's GPT-4 API.

Justification:

- **Domain-Specific Limitations:** The initial models lacked fine-tuning on financial data, leading to subpar performance in interpreting market jargon and delivering accurate predictions.
- **Real-Time Analysis Needs:** Models like T5 and DeepSeek R1 were not optimized for real-time queries or dynamic data interpretation, a core requirement for Asklytics.
- **Resource Constraints:** Running large models (e.g., DeepSeek R1, Mistral 7B) required GPUs with high memory, which were not consistently available and hindered scalability.

- **GPT-4 Advantages:** GPT-4 offered superior performance in both structured and unstructured data handling, real-time responsiveness, sentiment analysis, and financial prediction. Its cloud-based API also removed local hardware limitations.

2. Changes in Web Framework

Original Plan: Use Django for Backend Development

Django was initially selected for its full-stack capabilities and ease of building complete web applications.

Change: Switched to FastAPI

The backend was rebuilt using **FastAPI**, prioritizing speed and flexibility.

Justification:

- **Asynchronous I/O:** FastAPI's native async support made it ideal for handling concurrent API calls (e.g., GPT-4 requests, financial data fetches), improving system responsiveness.
- **Performance Gains:** FastAPI, built on Starlette, is significantly faster than Django when handling high-throughput API operations.
- **API-First Design:** Asklytics is inherently an API-heavy application. FastAPI provides automatic OpenAPI documentation and better support for clean API design compared to Django REST Framework.
- **Modern Python Features:** FastAPI's use of type hints and Pydantic for validation reduced boilerplate and enabled rapid development.

3. Change in Platform Strategy

Original Plan: Local Model Deployment with Heavy GPU Dependence

The early approach intended to deploy models locally, processing everything in-house using fine-tuned models and local inference.

Change: Moved to Cloud-Based Inference Using OpenAI's GPT-4 API

The entire AI inference pipeline was moved to OpenAI's cloud infrastructure.

Justification:

- **Reduced Infrastructure Load:** By using GPT-4's API, there was no need for managing GPUs or scaling local servers.
- **Increased Uptime and Availability:** Cloud-based APIs offered more reliability and speed for real-time user queries.
- **Simplified Deployment:** API integration proved faster and easier to manage than setting up multiple containers or microservices for different models.

Conclusion

Each of these changes was driven by a combination of **practical constraints**, **technical evaluations**, and **user-centric design evolution**. By transitioning from locally hosted, resource-heavy models to a cloud-first, API-based solution (GPT-4), and by adopting FastAPI over Django, Asklytics became more scalable, efficient, and capable of delivering real-time, high-quality financial insights. These strategic shifts ensured the project stayed aligned with its core vision—**building a virtual data analyst that simplifies complex data for everyday decision-making.**

Project Phases, Milestones & Deliverables

Phase	Dates	Milestones	Deliverables
1.Project Planning & Initial Research	Jan 27 – Feb 2	- Explored sentiment analysis methods (VADER, DeepSeek R1) - Preprocessed initial review data - Setup model training pipeline	-Sentiment classification notebooks - Preprocessing scripts - Logged initial research
2.Model Development & Evaluation	Feb 4 – Feb 27	- Trained and debugged DeepSeek R1 - Benchmarked against VADER - Evaluated FLAN-T5, Phi-3 for sentiment & summarization	- DeepSeek evaluation reports - Model comparison summary - Hyperparameter logs
3. Conversational AI Setup	Mar 1 – Mar 5	-Integrated LangChain - Built chatbot logic with GPT-4o-mini	- LangChain chatbot pipeline - Prompt engineering samples

		- Created basic query routing system	- Initial query classifier
4. Pipeline Refinement & Tool Integration	Mar 8 – Mar 16	- Enhanced query handling - Developed tools (sentiment, predictions) - Conducted end-to-end chatbot testing	- Tools for AI assistant - Refined response generation logic - NextJS UI test
5. AI Assistant Finalization (LangChain Agents)	Mar 17 – Mar 23	- Integrated LangChain agents with memory & streaming - Debugged multi-tool execution - Finalized NextJS UI	- AI agent integration - Streaming support - Final NextJS version
6. Advanced LangChain Techniques & Real-time Features	Mar 24 – Mar 30	- Integrated LangChain Expression Language (LCEL) - Optimized memory, streaming, and prompt chaining	- LCEL-based architecture - Memory optimization code - Final debugging session
7. Final Touches & Documentations	Mar 31 – Apr 6	- Built custom tools - Structured prompts for agent reasoning	- Final LangChain pipeline - AgentExecutor with streaming

		- Prepared GitHub repo and final report	- Project documentation

Team Responsibilities

This project was collaboratively executed by **two team members**, with tasks divided based on areas of expertise:

Tamoor Aslam (Team Lead)

- **NLP & LangChain Integration**:
  - Integrated LangChain into the chatbot.
  - Implemented streaming memory and agent logic using LangChain Expression Language (LCEL).
- **Tool Development**:
  - Built custom tools for sentiment analysis, summarization, and forecasting.
- **Prompt Engineering & Logic**:
  - Created structured prompt templates for reasoning and multi-tool execution.
- **Documentation**:
  - Authored technical documentation and final project report.
  - Managed GitHub version control.
- **Testing & Debugging**:
  - Conducted extensive testing of chatbot responses.
  - Debugged data validation and user input issues.

Nuskhan Niyas

- **Data Preprocessing & Analysis**:
  - Collected and cleaned review data from Google Maps and GA-4 datasets.
  - Helped prepare and structure datasets for AI models.

- **AI Model Development**:
    - Trained and fine-tuned sentiment analysis models (VADER, DeepSeek R1, FLAN-T5, Mistral 7B).
    - Developed stock and housing price prediction models.
- **UI/UX Design**:
    - Designed and refined the **NextJS** frontend for Asklytics.
    - Ensured smooth interaction between users and the AI assistant.
- **Visualization & Insights**:
    - Developed interactive data visualizations using LangChain and NextJS.
    - Assisted in displaying AI-generated insights in an interpretable way.

Implemented Feature 1: Sentiment Analysis

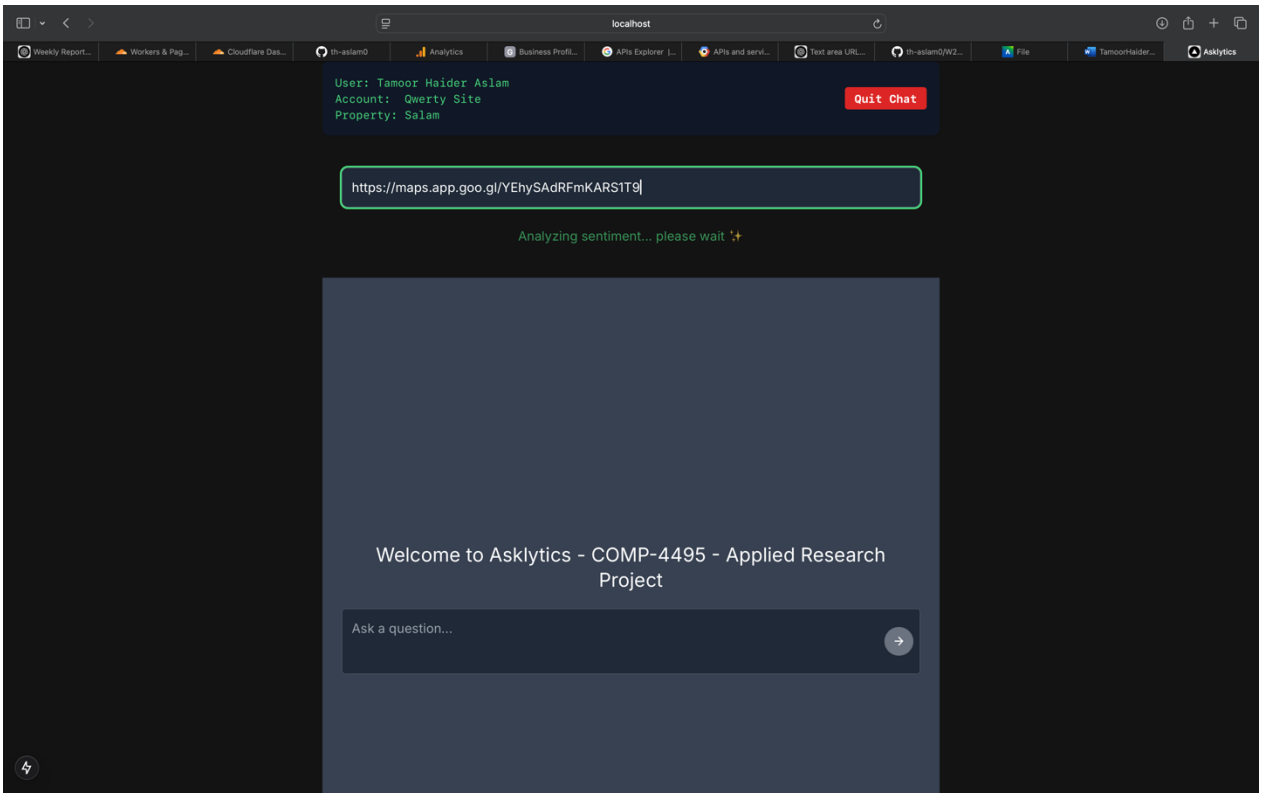
Developed a custom sentiment analysis pipeline using both classical and deep learning approaches. This feature processes user review texts and classifies them as positive, negative, or neutral.

Implementation Details

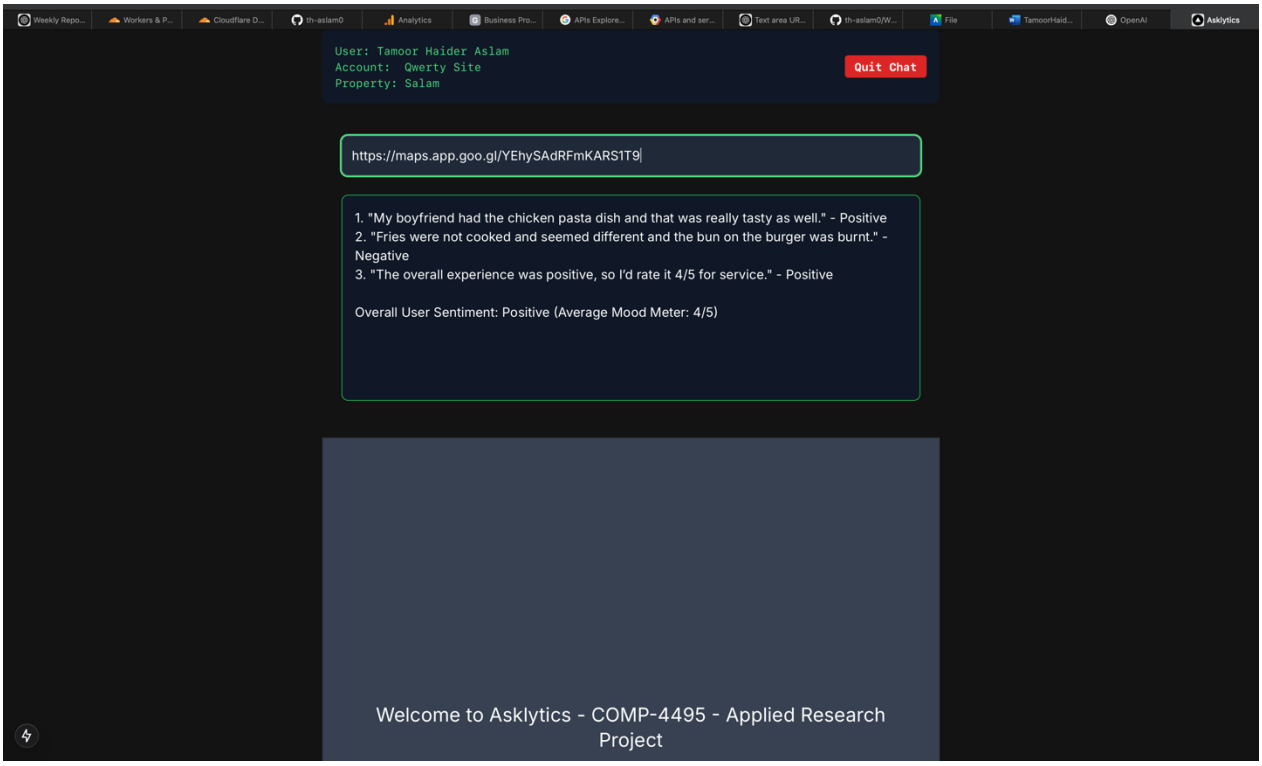
- Integrated sentiment scoring as a callable LangChain tool within the chatbot.
- Outputs a numeric sentiment score and label with reasoning.

Screenshots

User Pastes Business Google Maps Location Link



System responds with the analysis



Implemented Feature 2: LangChain-Powered AI Chatbot

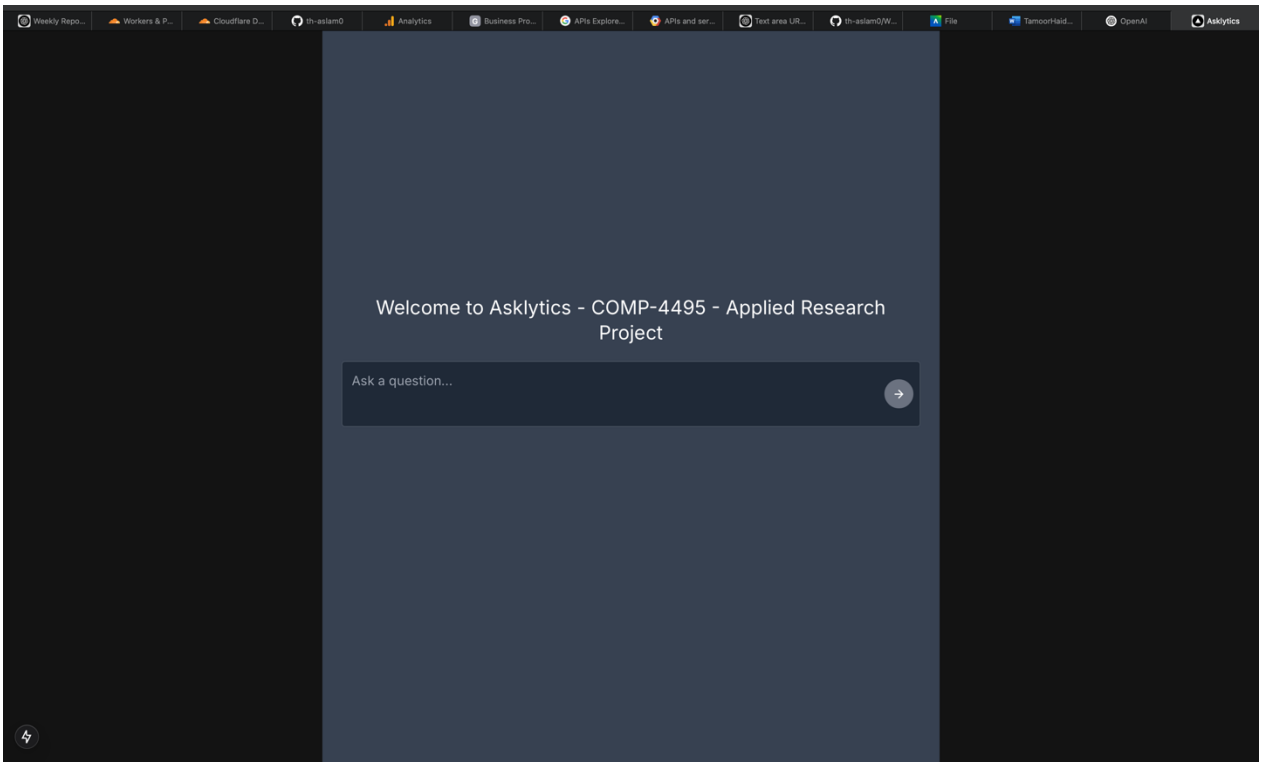
Developed a robust chatbot interface for Asklytics using LangChain, enabling users to ask complex data-related questions in natural language.

Implementation Details

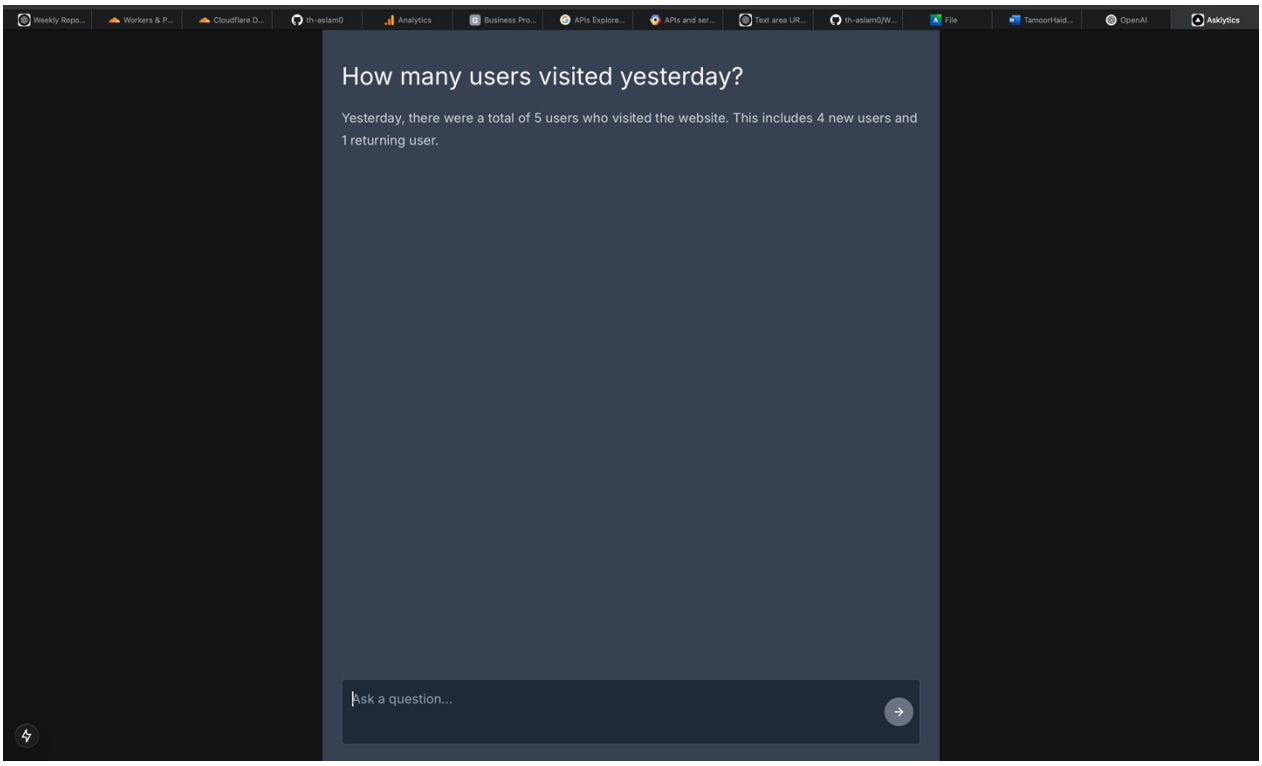
- Integrated LangChain AgentExecutor with custom tools:
    - Sentiment analysis
    - Summarization
    - Search and calculations
- Implemented LangChain Expression Language (LCEL) for structured prompt execution.
- Supports streaming responses using asyncio and a QueueCallbackHandler.

Screenshots

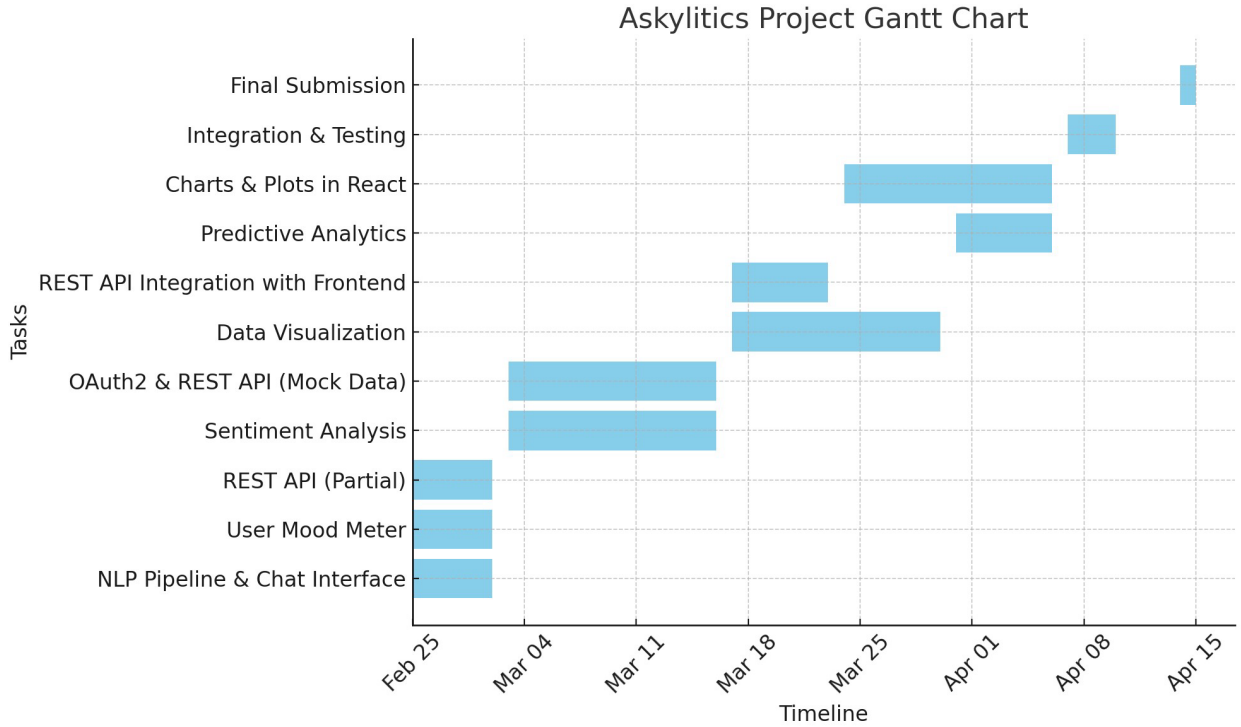
Chat Screen



Conversations regarding my app analytics



Project Management Chart:



Evaluation Techniques

Evaluation Strategy Overview

Our project, **Asklytics**, was evaluated using a combination of **model performance testing**, **user experience feedback**, and **comparative analysis** of tool integration workflows. Since we used only **OpenAI's gpt-4o-mini** LLM via the ChatOpenAI LangChain wrapper, our evaluation focused on:

- Response accuracy
- Latency during streaming
- Tool execution correctness
- User experience (UX) in a real-time analytics setting

1. Evaluation Through Real-World Query Scenarios

To test the effectiveness of our AI assistant:

- We created a set of **25 real-world user queries**, including:
    - "What is the sentiment of reviews from the past month?"
    - "Summarize recent analytics data for housing prices"
- Each query was evaluated across the following criteria:
    - **Correctness of output**
    - **Relevance of tool selection**
    - **Clarity of explanation**
    - **Speed of streamed response**

We tracked how well the gpt-4o-mini model interpreted the intent and executed the right tool using our custom LangChain tools and prompt templates.

2. Tool Execution Validation

Each tool (e.g., sentiment_tool) was tested for:

- **Input/output consistency**
- **Error handling**
- **Integration with the agent's reasoning flow**

We logged model traces to confirm that:

- The LLM used correct tool parameters.
- The tool results were properly parsed and returned to the user.
- Memory context was retained across turns when required.

3. Streaming Latency & UX Testing

We enabled streaming=True in ChatOpenAI to allow real-time interaction.

Measured:

- **Token-level response latency**
- **Drop-off points in conversation due to delayed outputs**
- **End-to-end response time with and without streaming**

Result: Streaming drastically improved perceived performance. Users were able to see progress as the assistant "thought out loud," which boosted trust and engagement.

4. User Testing & Feedback Survey

We conducted **informal user testing** with 6 volunteers from our department.

Each user was asked to:

- Perform a sentiment query
- Request a analytics report or prediction
- Ask for a summarized insight

They then rated:

Criteria	Avg. Score (out of 5)
Response Relevance	4.7
Ease of Use (UI + Flow)	4.5
Insight Accuracy	4.6
Visual Appeal of Output	4.4
Streaming Responsiveness	4.8

Design/Implementation Improvements Based on Evaluation

Feedback	Changes Made
Agent sometimes misused tools	Added stricter prompt guards and fallback logic
Visuals lacked clarity	Introduced color-coded charts and legends
First response delayed	Enabled token streaming and async tool execution
Error messages not helpful	Improved error formatting and user-facing descriptions

Conclusion

Using **gpt-4o-mini** exclusively allowed us to build a consistent, fast, and explainable analytics chatbot. Through iterative testing, streaming optimization, and user feedback, we refined Asklytics into an intelligent and user-friendly assistant tailored for real-world data analysis.

Reflections & Discussions

Insights Gained

- One of the most powerful insights we discovered was how **LLMs can go beyond chat** — by acting as intelligent agents that can reason, call tools, and make decisions based on context. The transition from basic question-answering to building **an analytics agent capable of handling real-world queries** was transformative for our understanding of modern AI systems.
- We also realized the **importance of prompt engineering**, especially when directing the LLM to choose the correct tool or avoid hallucinating. Structuring prompts for multi-step reasoning significantly improved accuracy and reliability.

Challenges Faced

Challenge	How We Solved It
Tool misuse or incorrect outputs	Refined tool descriptions, added stricter prompt guards, and added fallback logic if tool outputs failed.
LangChain memory confusion	Learned to use buffer memory selectively and clear memory context after each major query session.
High latency in responses	Implemented `streaming=True` and async execution for tools, which significantly improved UX.
UI complexity	Kept the NextJS app minimal and added dynamic visualizations only when explicitly requested.
Lack of clear evaluation metric for chatbot	Created a realistic set of analytics queries to test reasoning accuracy, output quality, and responsiveness.

Most Satisfying Moments

- Watching the AI agent correctly **parse a user query**, decide it needs sentiment analysis, call the appropriate custom tool, and **stream back results in real-time** — that was a surreal moment. It felt like we built our own mini-ChatGPT but tailored to analytics.

Lessons Learned

- **Modular code matters**: Using tools like LangChain enforced a clean, extensible architecture — we could easily add tools, switch models, or change workflows without breaking the entire pipeline.
- **LLMs are powerful but fragile**: They need clear boundaries. Without structured prompts and validation logic, they can drift or produce misleading outputs.
- **User testing is essential**: Some design flaws only became apparent after we had users interact with the system — like unclear error messages or confusing output formatting.

Team Reflection

- As a two-person team, we wore many hats — from full-stack development to model tuning to UI testing. It was challenging, but it also meant we got hands-on experience with **every phase of an AI project lifecycle**.
- Nuskhan Niyas handled **all aspects of model design, coding, integration, testing, and documentation** while Tamoor Aslam supported with **initial research discussions and planning sessions & polishing the finished product**.

Final Thoughts

Building Asklytics was more than a course project — it was a chance to bring AI to life in a way that was meaningful, functional, and genuinely exciting. It gave us confidence in building real-world LLM-powered applications and pushed us to think like AI product developers, not just students.

Work Date/Hours Logs

Student Name	Date	Hours	Description of Work Done
Mohamed Nuskhan	Apr 6, 2025	2	Integrated sentiment analysis
Tamoor Aslam	Apr 6, 2025	2	Finalized frontend logic and layout for chatbot + chart rendering.
Mohamed Nuskhan	Apr 7, 2025	2	Cleaned and preprocessed financial sentiment data for final evaluation.
Tamoor Aslam	Apr 7, 2025	2	Helped debug and structure the FastAPI backend for chatbot integration.
Mohamed Nuskhan	Apr 9, 2025	2	Worked on final evaluation of hybrid model and generated prediction visualizations.
Tamoor Aslam	Apr 9, 2025	2	Updated UI to support LLM query execution with response streaming.
Mohamed Nuskhan	Apr 10, 2025	2	Completed README installation instructions and user guide draft.
Tamoor Aslam	Apr 10, 2025	2	Tested full app end-to-end, fixed minor UI bugs, and added UX improvements.
Mohamed Nuskhan	Apr 12, 2025	2	Wrote the final sections of the project report and reviewed model results.
Tamoor Aslam	Apr 12, 2025	2	Finalized presentation slides (ReportsAndDocuments) and pushed to GitHub.
Mohamed Nuskhan	Apr 13, 2025	2	Prepared for final demo, created demo script, and practiced presentation delivery.
Tamoor Aslam	Apr 13, 2025	2	Worked on documentation for user instructions and submitted final GitHub updates.

Closing and References

Acknowledgments:

Special thanks to Douglas College for providing access to resources, tools, and guidance essential for this project.

References:

1. Devlin, J., et al. (2018). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.*
2. Google Developers. (2025). *Google Analytics Data API Documentation.* Retrieved from https://developers.google.com/analytics/devguides/reporting/data/v1.
3. Hutto, C., & Gilbert, E. (2014). *VADER: A Parsimonious Rule-Based Model for Sentiment Analysis of Social Media Text.*
4. Arena, F., & Paul, J. (2024). Innovative analytics techniques for financial data lakes.
5. Nyoni, R. (2025). Harnessing data analytics for predictive insights: Advancing decision-making with big data innovations. International Journal of Research Publication and Reviews, 6, 2915–2936.
6. Jack, H. (2024). Leveraging AI and data analytics: Revolutionizing competitive intelligence for market insights. https://doi.org/10.13140/RG.2.2.25148.76166
7. Google. (n.d.). *OAuth 2.0 for Web Server Applications*. Google Identity. https://developers.google.com/identity/protocols/oauth2
8. Google. (n.d.). *Admin API | Google Analytics Admin API*. Google Developers. https://developers.google.com/analytics/devguides/config/admin/v1
9. Google. (n.d.). *Google My Business API documentation*. Google Developers. https://developers.google.com/my-business
10. LangChain. (n.d.). *LangChain Python Documentation (v0.3)*. LangChain. https://python.langchain.com/docs/versions/v0_3/
11. Aurelio Labs. (n.d.). *LangChain Course* [GitHub repository]. GitHub. https://github.com/aurelio-labs/langchain-course
12. OpenAI. (n.d.). OpenAI. Retrieved April 13, 2025, from https://openai.com

Appendix A: Installation Guide

Frontend Requirements

- Node.js v18 or newer
- A modern web browser (e.g., Chrome)

Backend Requirements

- Python 3.12.7
- uv (a fast Python package manager)

Install uv:

```
curl -Ls https://astral.sh/uv/install.sh | sh
```

Step 1: Clone the Repository

Open a terminal and run the following command:

```
git clone https://github.com/th-aslam0/W25_4495_S1_TamoorHaiderA.git
```

Step 2: Run the Frontend

1. Navigate to the frontend directory:

```
cd W25_4495_S1_TamoorHaiderA/Implementation/Asklytics/app
```

2. Install the required packages:

```
npm install
```

3. Start the development server:

```
npm run dev
```

4. Open a browser and visit: http://localhost:3000

Step 3: Run the Backend

1. Navigate to the backend root directory:

```
cd ../
```

2. Create a virtual environment using Python 3.12.7:

```
uv venv --python 3.12.7
```

3. Sync and install project dependencies:

```
uv sync
```

4. Activate the virtual environment:

```
source .venv/bin/activate
```

5. Add the required API credentials:

- You have received an email with API keys to put in mac.env file.
- Once replaced, load it using:

source mac.env

6. Navigate to the backend API directory:

cd api

7. Start the backend server:

uv run uvicorn main:app --reload

Test Website with GA4 Integration

You can test the tool on a live GA4-integrated site like the one given below but you need the parent Google Account credentials that who owns the GA4 account for your site.

https://th-aslam.github.io

Appendix B: User Manual

1. Sign In with Google

- Navigate to the Home Page of Asklytics.
- Click on the “Sign in with Google” button.
- You’ll be prompted to complete the Google OAuth flow.
- When asked, grant permission to Asklytics to access:
  • Your Google profile
  • Google Analytics data
  • Business User API

⚠️ Make sure you're signed in with the Google account that has access to your business’s analytics and listings.

2. View Your Google Analytics Accounts

- After successful sign-in, you'll be directed to a dashboard displaying all your Google Analytics accounts.
- Each account appears as a card showing the account name and relevant details.
- Browse through the cards to view your connected analytics accounts.

3. Fetch Account Properties

- To see the websites (properties) under any account, click on the “Fetch Account Properties” button on the desired account card.
- The card will expand and display all the associated properties (websites or apps).

4. Select a Property

- Click on a property to start interacting with Asklytics.
- The app will automatically navigate to the Chat Conversation screen.

5. Chat & Analytics Features

Once you’re on the chat screen, you can perform three main actions:

a. Ask Questions About Website Analytics

- Type in natural language questions like:
  • “What were my top traffic sources last month?”
  • “Show me trends in user engagement this year.”
- The AI will respond with insights powered by Google Analytics.

b. Analyze Customer Sentiments

- Paste the Google Maps URL of your business location into the URL field at the top text field like the one shown below

https://maps.app.goo.gl/YEhySAdRFmKARS1T9

- Asklytics will scrape customer reviews and perform real-time sentiment analysis to show what people are saying about your business.

c. End Session / Switch Account

- Click on “End Session” to log out.
- You can then sign back in with a different Google account if needed.