TAPMI

(A constituent unit of MAHE, Manipal)

TAPMI

Project Report

MBAT51C02 - Fundamentals of Computing

Authors - Group B1
Mrs. Poojitha K (251620140002)
Mr. Aman Tiwari (251620140008)
Mr. Sriparno Chowdhury (251620140012)
Mr. Swami Mishra (251620140013)
Ms. Shivani Pathania (251620140028)
Mr. Kunjithapatha Chozhan P (251620140032)

September 2025

Contents

Executive summary

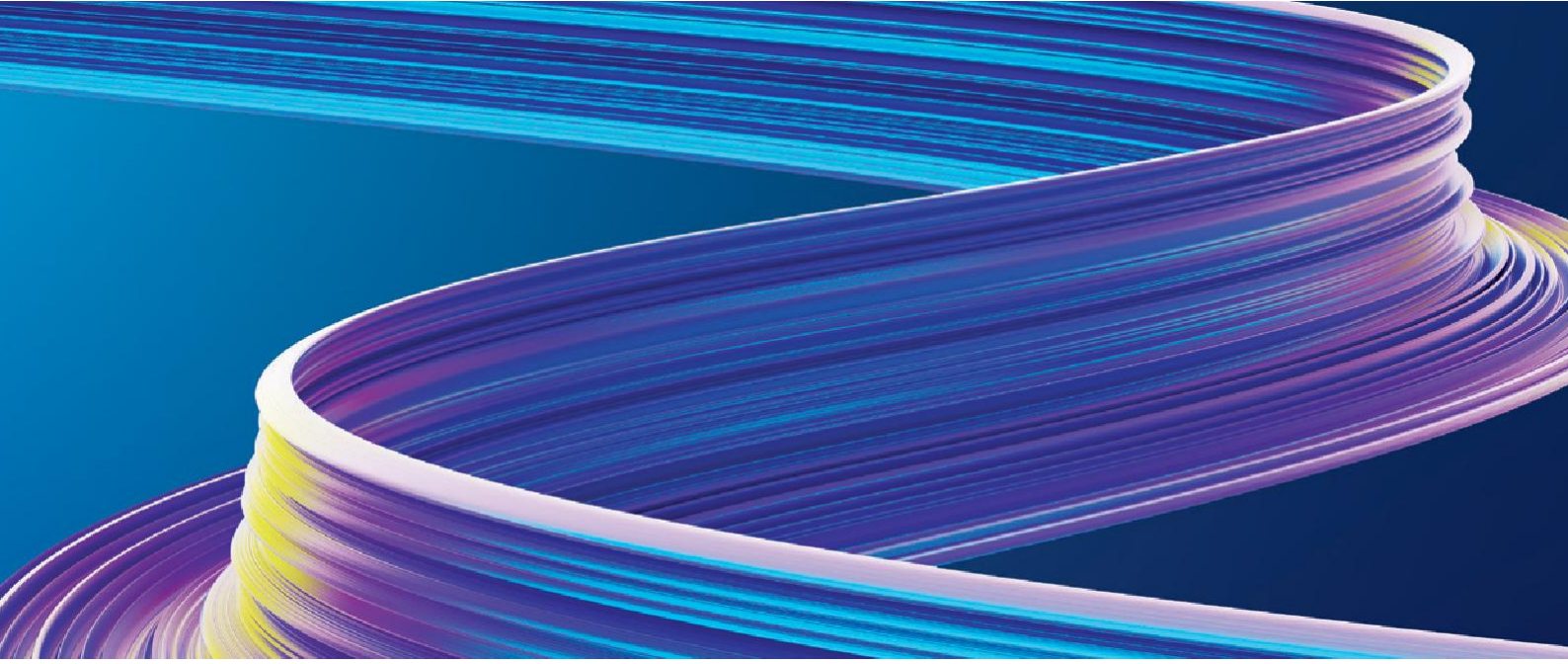
In the contemporary digital economy, characterised by high velocity and intense competition, understanding and anticipating consumer behaviour is a primary determinant of competitive advantage. This project addresses a critical, real-world business problem faced by digital marketers: the inefficient allocation of resources and strategic planning due to a lack of timely, accessible, and interpretable data on consumer search interest. The primary objective was to engineer a comprehensive software solution that transforms raw Google Trends data into actionable strategic insights, enabling data-driven decision-making.

The solution, the **Trend-AI Marketing Dashboard**, is a full-stack web application developed entirely in Python, leveraging the Streamlit framework for rapid and interactive UI development. The system programmatically sources consumer search interest data via the SerpApi commercial API, a strategic decision to ensure high reliability and circumvent the rate-limiting constraints inherent in free scraping methods. A key architectural feature is a dual-mode data pipeline, allowing the application to seamlessly switch between live API calls for custom analysis and a robust offline mode that uses pre-downloaded CSV datasets for guaranteed-to-work demonstrations. This hybrid approach ensures both real-time relevance and high availability.

The core of this project lies in the depth and breadth of its computational analysis, which is categorised into four distinct modules:

- **Descriptive and Diagnostic Analytics:** A comprehensive suite of interactive visualisations built with Plotly provides an overview of historical performance. This includes time-series plots for trend visualisation, geospatial choropleth maps for regional analysis, and seasonal pattern heatmaps for campaign timing. Deeper diagnostic insights are provided through classical time-series decomposition using the statsmodels library, which separates a trend into its underlying trend, seasonal, and residual components.
- **Predictive Analytics:** Meta's Prophet machine learning library is integrated to generate robust 12-month forecasts. The model is fitted on historical data to learn patterns and predict future interest with statistically defined uncertainty intervals, providing a quantitative basis for strategic planning.
- **Generative Analytics:** A state-of-the-art AI "Co-pilot," powered by the Google Gemini API, assists strategy formation. By employing an in-context learning approach, programmatically feeding the AI a real-time, quantitative summary of the analysed data, the module generates highly relevant, data-grounded marketing campaign ideas, bridging the gap between analysis and creative execution.

The application was successfully containerised using a custom Dockerfile and deployed on the Hugging Face Spaces cloud platform. The development process navigated and solved numerous real-world challenges, including API instability and complex cloud deployment permissions. The result is a robust, user-friendly proof-of-concept that demonstrates the transformative potential of combining data engineering, machine learning, and generative AI to solve complex business problems in the marketing domain.



Introduction

Background and Motivation

In the data-rich environment of modern digital marketing, a fundamental paradox exists. While organisations have access to unprecedented volumes of data, many still lack the tools to convert this data into timely, actionable intelligence. Consumer search behaviour, in particular, represents a vast and authentic source of "digital consumer intelligence." Unlike traditional survey data, which is solicited and often latent, search query data is unsolicited, immediate, and captured at a massive scale, providing a direct proxy for consumer curiosity and commercial intent. This project was motivated by the opportunity to harness this data stream, moving beyond simple manual lookups on the Google Trends website to a sophisticated, automated analytical platform.

Problem Statement

Digital marketing professionals require timely, accurate, and easily digestible insights to optimise campaigns. The primary challenges they face include:

- **Inefficient Campaign Timing:** Launching marketing campaigns without understanding the seasonal demand cycles for products or topics.
- **Reactive Market Strategy:** An inability to forecast future interest trends, leading to reactive rather than proactive planning.
- **Complex Data Analysis:** The manual effort to analyze trends for seasonality, growth momentum, and relative popularity is time-consuming and requires specialized skills.
- **Creative Bottlenecks:** A recurring struggle to generate fresh, data-informed marketing campaign ideas that align with current consumer interests.

Project Objectives

1. **To Engineer a Resilient Data Pipeline:**
    - Design and implement a data acquisition system to programmatically source Google Trends data via a reliable, high-availability commercial API.
    - Incorporate a seamless offline fallback mode using local CSV files to ensure high availability for demonstration.

2. **To Develop a Multi-faceted Analytical Engine:**
    - Implement a suite of computational analyses—descriptive, diagnostic, and predictive—to provide a holistic view of consumer trends.
    - Ensure all analytical outputs are rendered as interactive, easily interpretable visualisations.

3. **To Integrate and Deploy a Predictive Model:**
    - Incorporate the Prophet machine learning library to generate statistically sound 12-month forecasts of search interest, complete with uncertainty intervals.

4. **To Implement a Generative AI for Strategy Augmentation:**
    - Integrate the Google Gemini API as an AI "Co-pilot."
    - Develop a "context injection" mechanism to ground the AI's responses in the data being analysed, thus enhancing the relevance and actionability of its outputs.

5. **To Design an Intuitive User Interface:**
    - Build a clean, interactive, user-friendly web dashboard using Streamlit, making sophisticated analysis accessible to non-technical stakeholders.

6. **To Deploy a Production-Ready, Containerised Application:**
    - Containerise the application using Docker to ensure a reproducible and isolated environment.
    - Deploy the final application to a public cloud platform (Hugging Face Spaces) for accessibility and demonstration.

Scope and Limitations

In Scope:

- The application will focus exclusively on Google Trends data.
- The analyses will include time-series, geospatial, seasonality, YoY growth, and decomposition.
- The application will support both live API and offline modes using user-provided CSVs.
- The final deliverable is a publicly deployed web application.

Out of Scope:

- Integration with other data sources (e.g., sales data, social media analytics, ad spend).
- User authentication and data persistence (e.g., saving past searches).
- Automated anomaly detection and alerting features.

Methodology & System Design

Technology Stack Justification

The selection of technologies was driven by the principles of analytical power, rapid development, and deployment stability.

Component	Technology	Justification
Core Language	Python	Chosen for its extensive and mature ecosystem of libraries for data science, machine learning, and web development.
Web Framework	Streamlit	Selected for its rapid application development (RAD) capabilities, allowing for the creation of interactive, data-centric UIs with pure Python, thus eliminating the need for separate front-end development (HTML/CSS/JS).
Data API	SerpApi	Chosen over free scraping libraries (pytrends) due to its high reliability, guaranteed uptime, and ability to bypass IP-based rate limiting, which is a critical requirement for a stable, deployed application.
Data Manipulation	Pandas	The industry standard in Python for data manipulation, cleaning, and transformation. Its DataFrame structure is central to all analytical functions.
Visualization	Plotly Express	Selected for its ability to generate a wide range of high-quality, fully interactive charts (line, pie, map, heatmap) with a concise, high-level API.
Forecasting	Prophet (Meta)	A machine learning library specifically designed for business time-series forecasting. It was chosen for its robustness in handling missing data and outliers, its ability to model multiple seasonalities, and its easily interpretable outputs.
Statistical Analysis	statsmodels	A core scientific library in Python. It was used specifically for its robust implementation of classical time-series decomposition, a key diagnostic feature.
Generative AI	Google Gemini API	The gemini-1.5-flash-latest model was selected for its strong performance in natural language understanding and generation, its speed, and its cost-effectiveness for a chat-based application.
Deployment	Docker & Hugging Face Spaces	This combination was chosen to create a reproducible, isolated, and scalable deployment environment. Docker containerization solves dependency and permission issues, while Hugging Face Spaces provides a free and simple platform for hosting and sharing such applications.

System Architecture and Data Flow

The application is architected around a stateful UI that manages the data pipeline.

User Input (Control Plane): The user interacts with sidebar widgets. A central st_searchbox "combo" field, built from a community component, serves as the primary input. The state of this and other widgets is managed using Streamlit's session_state.

Mode Selection (Logic Switch): The application's core logic checks the string value from the st_searchbox. If the string is an exact match to a key in a predefined scenarios dictionary, it enters "Offline Mode." For any other string, it enters "Live API Mode."

Data Acquisition (Data Plane):

- **Offline Mode:** The load_all_offline_data function is triggered. It reads and parses a set of pre-named CSV files from the repository's /data directory using Pandas. This function includes logic to handle the specific formatting quirks of Google Trends CSVs, such as header rows and inconsistent column names.
- **Live API Mode:** The fetch_data_from_serpapi function is triggered. It securely retrieves an API key, constructs a parameter dictionary, and makes an HTTP request to the SerpApi endpoint. It then parses the returned JSON into a standardized dictionary of DataFrames.

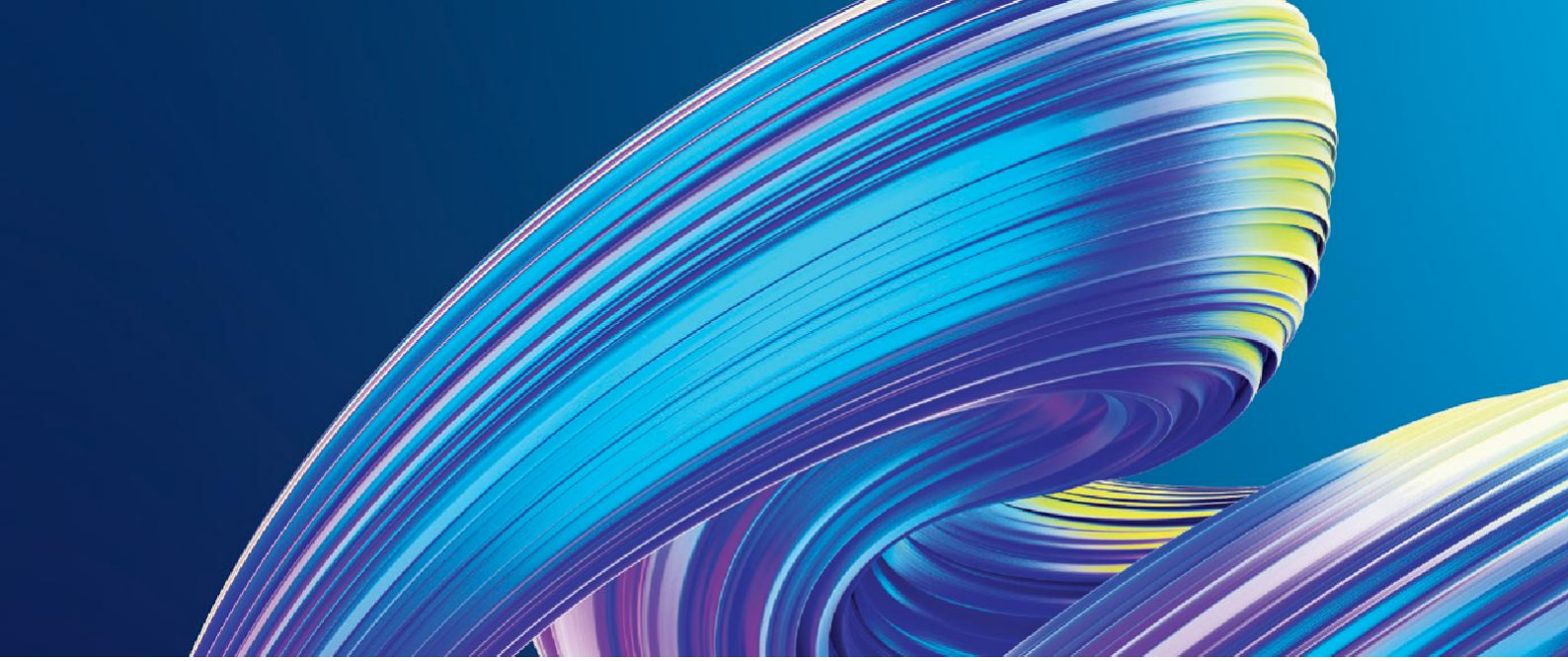
Data Processing & Analysis: Both data paths yield a unified trends_data dictionary. The data is then cleaned (e.g., pd.to_numeric) and passed to the various analytical modules within the dashboard tabs.

Rendering: Each module computes its specific analysis (e.g., resampling for seasonality) and renders the appropriate Plotly or Matplotlib chart to the Streamlit frontend.

Data Acquisition and Preprocessing

A resilient, dual-mode data pipeline was engineered.

- **Live Data Pipeline (SerpApi):** The fetch_data_from_serpapi function serves as the interface to the live data source. It accepts user inputs, constructs a parameter dictionary, and makes a GET request to the google_trends engine of the SerpApi service. The function is designed to robustly parse the nested JSON response, extracting keys such as interest_over_time and interest_by_region. A critical sub-component of this function is a date parsing helper that can handle multiple datetime formats returned by the API (e.g., daily Sep 10, 2025 vs. weekly Sep 10 – 16, 2025), ensuring the creation of a valid DatetimeIndex for the Pandas DataFrame.
- **Offline Data Pipeline (CSV):** The load_all_offline_data function provides a reliable fallback. It takes a scenario configuration, constructs file paths to the required CSVs within the project's /data directory, and uses pandas.read_csv to ingest them. This function contains crucial preprocessing logic to handle the specific format of Google's downloaded CSVs, including skipping header rows (skiprows=...), renaming columns for consistency, and cleaning the data (e.g., converting the string "<1" to a numeric value).



Computing Analysis & Feature Implementation

The analytical core of the dashboard is presented in a multi-tab interface.

The "Trend Analysis" Module (Descriptive & Diagnostic)

This module provides a multi-faceted, uncollapsed view of the historical data.

- **Time-Series Visualization of Relative Search Interest:**
  - **Business Objective:** To provide an immediate, high-level understanding of the performance and narrative of the selected keywords over time.
  - **Technical Methodology:** The core interest_over_time DataFrame, with its DatetimeIndex, is passed directly to plotly.express.line(). Plotly's interactivity allows users to zoom into specific periods and hover to see exact values.
  - **Interpretation:** Users can instantly identify upward/downward trends, periods of high volatility, seasonal spikes, and the relative popularity between compared keywords.
- **Aggregate Interest Analysis (Share of Search):**
  - **Business Objective:** To provide a quick, zero-dimension summary of the overall popularity of each keyword relative to the others in the compared set.
  - **Technical Methodology:** The pandas.DataFrame.sum() method is used to calculate the total aggregate interest score for each keyword column over the entire period. These aggregate scores are then visualized using a plotly.express.pie() chart.
  - **Interpretation:** This chart allows a marketer to immediately identify the "market leader" within the search query set, providing a clear picture of brand dominance or topic popularity.

- **Geospatial Interest Analysis:**
  - **Business Objective:** To identify geographic markets with the highest and lowest consumer interest, informing decisions on ad campaign targeting and regional market prioritization.
  - **Technical Methodology:** The interest_by_region DataFrame is visualized using plotly.express.choropleth(). This function maps the 'Region' names to geographic shapes and uses the 'Interest' value to determine the color intensity on a continuous scale.
  - **Interpretation:** The map provides a powerful visual tool for identifying market hotspots at a glance, while the accompanying sorted data table provides precise quantitative rankings.

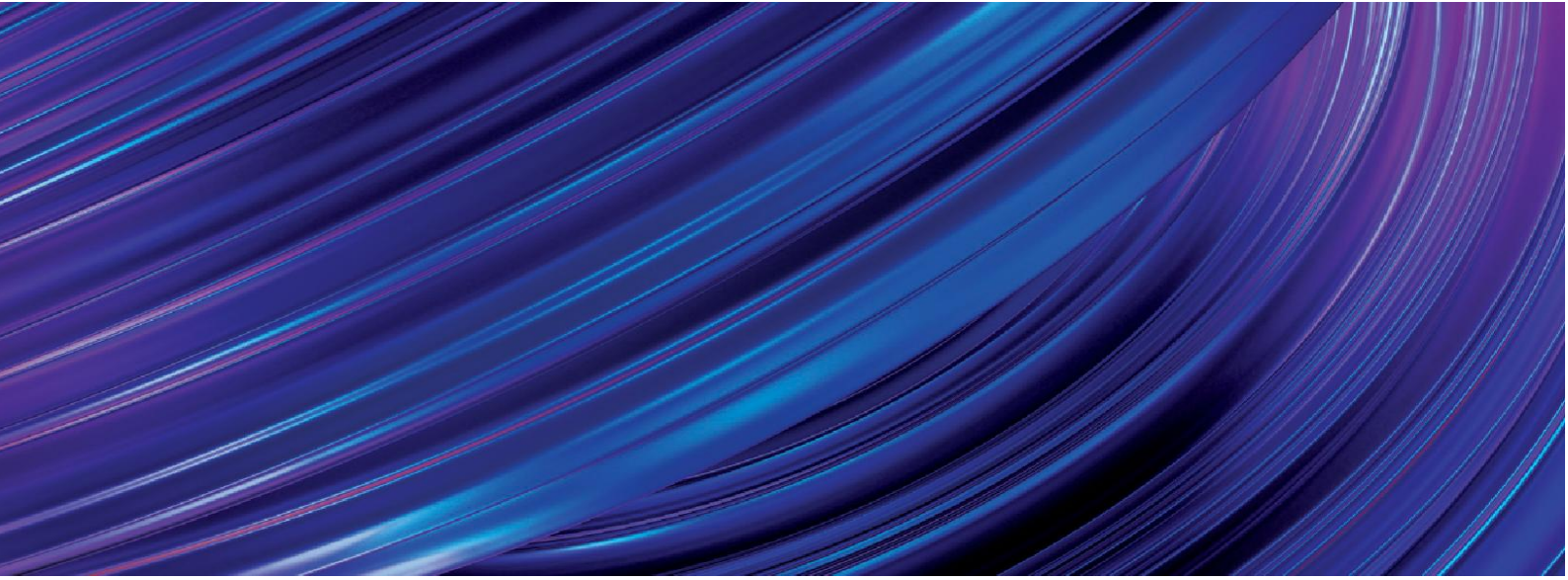
- **Temporal Pattern Analysis (Seasonality):**
  - **Business Objective:** To move beyond simple observation of spikes and quantitatively identify which months consistently represent peak and off-peak seasons for the selected keywords.
  - **Technical Methodology:** The analysis involves several Pandas operations. First, the time-series data is resampled to a monthly frequency using df.resample('M').mean(). This aggregates daily or weekly data into a stable monthly average. The DataFrame is then grouped by the month number (df.groupby(df.index.month)) to calculate the average interest for each month across all years. To enable fair comparison, this result is then normalized to a 0-100 scale using the formula: value_norm=fracvalue−value_minvalue_max−value_mintimes100. Finally, this normalized data is plotted on a plotly.express.imshow() heatmap.
  - **Interpretation:** The heatmap allows marketers to easily identify, for example, that interest in "Nike" consistently peaks in August and December, providing a clear, data-backed directive for campaign timing.
- **Momentum Analysis (Year-over-Year Growth):**
  - **Business Objective:** To quantify the growth or decline of a keyword's momentum, a standard business metric that is more insightful than absolute values alone.
  - **Technical Methodology:** Using the monthly resampled data for a single user-selected keyword, the pandas.DataFrame.pct_change(periods=12) method is applied. This calculates the percentage change from the same month in the prior year. The resulting DataFrame is then styled to color-code positive (green) and negative (red) growth.
  - **Interpretation:** This allows a marketer to answer critical questions like, "Is our brand's growth accelerating this quarter compared to last year?"

- **Statistical Trend Decomposition:**
  - **Business Objective:** To provide a deeper, diagnostic view of a trend by separating the signal from the noise.
  - **Technical Methodology:** The statsmodels.tsa.seasonal.seasonal_decompose function is applied to a selected keyword's time series. The classical time-series decomposition model assumes that a time series Y_t can be expressed as a combination of a trend T_t, a seasonal component S_t, and a residual component R_t. An additive model was employed, Y_t=T_t+S_t+R_t, which is appropriate when the magnitude of the seasonal fluctuation does not vary with the level of the time series.
  - **Interpretation:** The output plots each component separately, allowing a user to distinguish the true long-term growth trajectory (Trend) from the predictable yearly patterns (Seasonality) and unpredictable shocks (Residuals).

- **The "Future Forecast" Module (Predictive Analytics)**
  - **Business Objective:** To provide a quantitative, forward-looking view of potential consumer demand, enabling proactive planning for inventory, budgets, and campaigns.
  - **Technical Methodology:** This module utilizes the Prophet library. Prophet models a time series y(t) as an additive model: y(t)=g(t)+s(t)+h(t)+epsilon_t, where g(t) is the trend component, s(t) is the seasonality component, h(t) represents holidays, and epsilon_t is the error term. The historical data is used to train the model via the .fit() method. The model then generates a 365-day future forecast using .predict().
  - **Interpretation:** The output plot clearly visualizes the historical data, the forecasted values (yhat), and the upper and lower uncertainty intervals, giving the user not just a prediction, but a sense of the model's confidence.

The "AI Co-pilot" Module (Generative Analytics)

- **Business Objective:** To bridge the "insight-to-action" gap by using AI to interpret the quantitative data and generate qualitative, creative marketing ideas.
- **Technical Methodology:** This feature leverages the in-context learning capabilities of Large Language Models. A data_summary string is dynamically created by the Python script, containing statistical descriptions (.describe()) of the time-series data and a list of the top regions. This summary is programmatically inserted into the system prompt that is sent to the Google Gemini API.
- **Interpretation:** This "context injection" grounds the model's response in the specific, factual data being analyzed on the dashboard. This prevents the AI from giving generic advice and instead enables it to provide tailored, data-driven suggestions, such as, "Given that interest in 'Nike' peaks in August, a 'Back to School' campaign on social media would be highly effective."



Deployment & Operationalisation

Containerization with Docker

To ensure a consistent and reproducible runtime environment, the application was containerised using Docker. A custom Dockerfile was created with several key features:

- **Stable Base Image:** It uses the python:3.11-slim base image for known compatibility with the data science libraries.
- **Security Best Practices:** It creates a dedicated, non-root appuser and a writable home directory. This is a critical step that solves the PermissionError issues encountered on restricted cloud platforms by preventing the application from attempting to write to protected system directories.
- **Dependency Management:** It systematically copies the requirements.txt file and uses pip to install all necessary Python packages.
- **Environment Configuration:** It sets necessary environment variables, such as MPLCONFIGDIR, to ensure libraries like Matplotlib function correctly within the container.

Cloud Hosting on Hugging Face Spaces

Hugging Face Spaces was selected as the cloud hosting platform. It was chosen for its built-in, secure "Repository secrets" manager, and its generous free tier for hosting Docker containers.

Security Considerations

A primary security consideration was the management of API keys for SerpApi and the Google Gemini API. These were never hardcoded in the application. For local development, they were stored in a .streamlit/secrets.toml file, which was explicitly excluded from version control via a .gitignore file. For production deployment, the keys were stored as encrypted "Repository secrets" in the Hugging Face Space settings. The application code was written to read these secrets from the environment variables injected by the platform, ensuring no sensitive credentials were ever exposed in the public codebase.

Results & Discussion

The Final Deployed Application: A User Journey

The result of this project is a fully functional and publicly accessible web application. A typical workflow is as follows:

- **Launch:** The application loads instantly with a default, pre-configured analysis of "Nike vs. Adidas" powered by reliable offline data, guaranteeing an error-free first impression.
- **Exploration:** The user can immediately explore the full suite of descriptive, diagnostic, and predictive analyses for this default scenario.
- **Customisation:** The user can then type their own custom keywords (e.g., their brand vs. a competitor) into the sidebar's "combo" field.
- **Live Analysis:** The application intelligently detects the custom input and automatically performs a live API call to SerpApi to fetch the latest data.
- **AI Strategy:** After reviewing the live data, the user can navigate to the AI Co-pilot tab and ask for strategic recommendations, which the AI will base on the data it has just been shown.

The application is live and accessible at:
https://huggingface.co/spaces/swamimishra/trend-ai-dashboard

Key Challenges and Engineering Resolutions

The development process was highly iterative and emblematic of real-world software engineering.

- **API Unreliability:** The initial reliance on pytrends was abandoned due to persistent 429 errors, leading to the strategic adoption of the more stable SerpApi.
- **Deployment Environment Complexity:** Deploying to Hugging Face's Docker-based environment revealed critical PermissionError and StreamlitSecretNotFoundError issues. These were systematically debugged. The final solution involved creating a robust, custom Dockerfile that sets up a non-root appuser with a writable home directory.
- **Data Inconsistency:** The application had to be made robust against various data formats returned by the API and CSVs, such as weekly date ranges (Sep 6 – 12, 2020), non-numeric values (<1), and inconsistent column headers. This was solved by adding defensive data parsing and cleaning logic in the data loading functions.

Discussion of Project Limitations

- **Data Proxy:** Google Trends data is a powerful proxy for consumer interest, but it is not a direct measure of sales or conversions. Its insights should be used to inform strategy, not as a standalone metric of business performance.
- **Data Source Dependency:** The application's live mode is entirely dependent on the availability and terms of service of the SerpApi. Any changes or outages in their service would impact the application.
- **Model Limitations:** The Prophet forecast is based on historical patterns and cannot predict the impact of unprecedented "black swan" events (e.g., a viral marketing moment, a global crisis).

Conclusion

This project successfully demonstrates the application of Python and its data science ecosystem to solve a tangible business problem. It progressed from a simple data scraping concept to a full-stack web application featuring a sophisticated analytical backend, a machine learning forecasting module, and a cutting-edge generative AI assistant. The iterative process of overcoming deployment and data challenges was a critical learning experience, highlighting the importance of robust error handling and adaptable engineering solutions. The final Trend-AI Marketing Dashboard serves as a powerful proof-of-concept for how modern computing tools can democratize data analysis and drive smarter, faster, and more creative marketing strategies.

Recommendations for Future Work

The platform is built to be extensible. A future roadmap could include:

- **Competitive Intelligence:** Adding features to specifically track a brand's "share of search" against a defined set of competitors over time, with automated charting of gains and losses.
- **Anomaly Detection & Alerting:** Implementing a backend statistical process to automatically detect significant trend "breakouts" and send email alerts to the marketer.
- **Data Integration:** Connecting to other marketing APIs, such as Google Ads or social media platforms, to allow users to overlay campaign cost and performance data directly onto the trend chart for direct ROI analysis.

September 2025