

Hackathon Project Phases Template for the DataQueryAI project.

---

# Hackathon Project Phases Template

## Project Title:

DataQueryAI: Intelligent Data Analysis with Google TAPAS

**Team Name:** PIONEER

## Team Members:

- L. Pranathi
  - B. Shivani
  - B. Kavya
  - G. Saranya
- 

## Phase-1: Brainstorming & Ideation

### Objective:

DataQueryAI aims to provide advanced AI-driven solutions for data analysis, enabling businesses to extract meaningful insights from complex datasets. Its objective is to simplify data querying, automate decision-making processes, and enhance operational efficiency through intelligent data processing. By leveraging machine learning and natural language processing, DataQueryAI helps users interact with data intuitively and effectively.

### Key Points:

#### 1. Problem Statement:

DataQueryAI leverages advanced generative AI models to provide insightful and contextually accurate responses to questions about your data. This innovative system simplifies data analysis by allowing users to upload CSV files and receive detailed answers based on the data's content. DataQueryAI enhances productivity and ensures accurate data interpretation across various scenarios, offering robust solutions tailored to different user needs.

## 2. Proposed Solution:

DataQueryAI empowers business analysts to extract valuable insights from data effortlessly. When a user uploads a CSV file containing sales data, DataQueryAI analyzes the content and generates informative responses to queries. For instance, if an analyst asks for the top-selling products in the last quarter, DataQueryAI can quickly identify and present the relevant information. This feature reduces the time spent on manual data analysis, ensures accuracy in reporting, and enhances decision-making processes.

## 3. Target Users:

- **Businesses and Enterprises** – Companies looking to streamline data analysis and make data-driven decisions more efficiently.
- **Data Analysts and Scientists** – Professionals who need advanced tools for querying and interpreting large datasets.
- **Non-technical Users** – Individuals in organizations without technical expertise who want easy, intuitive access to data insights.
- **IT Teams and Developers** – Teams integrating AI-driven data querying solutions into their existing systems for automation and optimization.

## 4. Expected Outcome:

DataQueryAI empowers business analysts to extract valuable insights from data effortlessly. When a user uploads a CSV file containing sales data, DataQueryAI analyzes the content and generates informative responses to queries. For instance, if an analyst asks for the top-selling products in the last quarter, DataQueryAI can quickly identify and present the relevant information. This feature reduces the time spent on manual data analysis, ensures accuracy in reporting, and enhances decision-making processes.

---

# Phase-2: Requirement Analysis

## Objective:

DataQueryAI needs robust machine learning models, natural language processing (NLP) capabilities, and integration with Google TAPAS for complex table-based data understanding and querying. It requires high-performance infrastructure to handle large datasets and real-time query processing. The system should support intuitive natural language queries, provide accurate insights from structured and unstructured data, and deliver seamless integration with existing databases and cloud platforms, enabling efficient, automated data analysis.

## Key Points:

### 1. Technical Requirements:

- **Programming Language:** Python
- **Back-end:** Pandas, Pipeline from transformers
- **Frontend:** Google collab
- **Database:** Google BigQuery

### 2. Functional Requirements:

- **Data Structuring:** Automatically structure unstructured or semi-structured data into tabular format for processing by Google TAPAS.
- **Real-Time Data Analysis:** Provide real-time processing and fast retrieval of insights from large datasets.
- **Advanced Data Querying:** Support complex queries on table-based data, utilizing TAPAS' table understanding capabilities.
- **User-Friendly Interface:** Offer an intuitive interface that simplifies data exploration and querying for all users, regardless of technical expertise.
- **Scalable Architecture:** Ensure the system can handle large datasets efficiently as data volumes grow.
- **Data Visualization:** Present results in clear, interpretable visual formats (e.g., charts, graphs).
- **Security and Data Privacy:** Ensure secure access, data encryption, and compliance with privacy regulations.
- **Customizable Reporting:** Allow users to generate and customize reports based on query results.

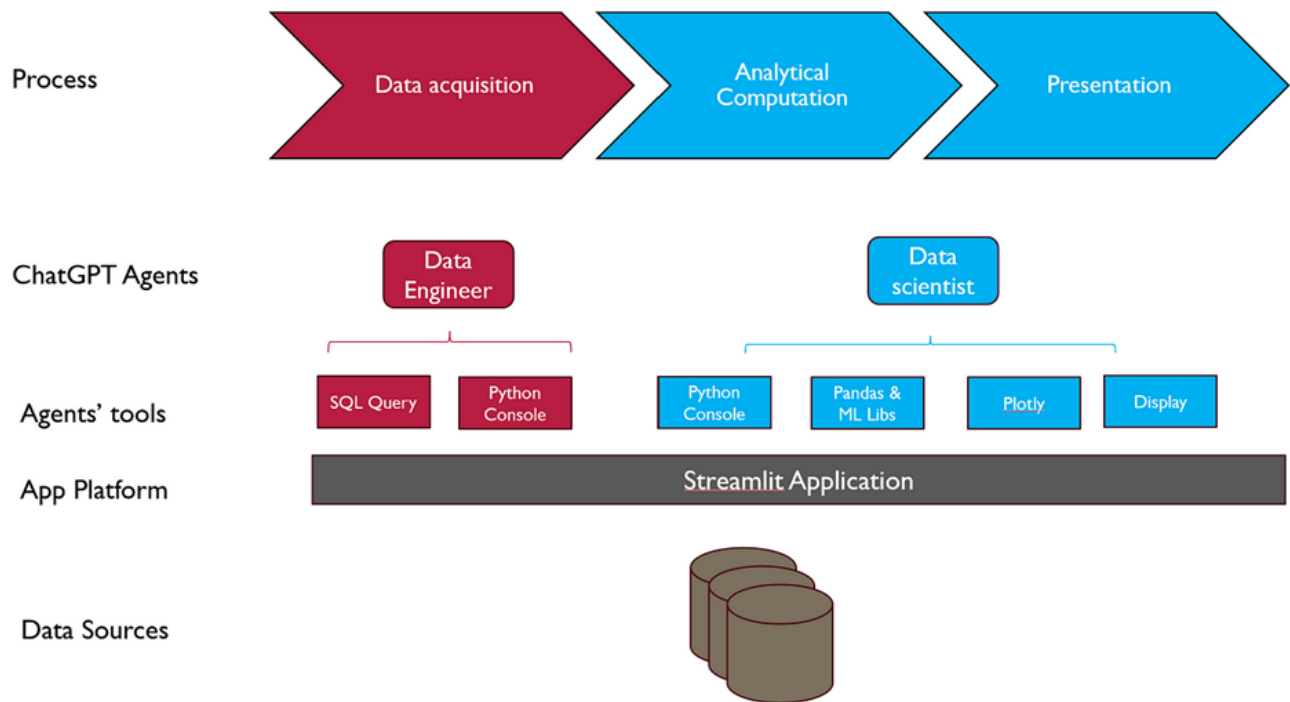
### 3. Constraints & Challenges:

- **Data Quality & Structure:** Ensuring the data fetched from various sources is clean, structured, and in a format compatible with TAPAS for accurate analysis.
  - **Complex Query Interpretation:** Accurately interpreting complex natural language queries, especially when context or data relations are ambiguous.
  - **Scalability:** Handling large volumes of data efficiently without sacrificing performance, especially as datasets grow.
  - **Data Privacy & Security:** Ensuring sensitive data is protected and that the system complies with privacy regulations (e.g., GDPR).
  - **User Experience:** Designing an intuitive interface that accommodates users with varying levels of technical expertise, while maintaining advanced functionalities.
-

## Phase-3: Project Design

### Objective:

Develop the architecture and user flow of the application.



### Key Points:

#### 1. System Architecture:

- **User Interface:** Provides a web or mobile interface for users to input queries and view results via interactive visualizations.
- **API Gateway:** Routes incoming requests from the UI to appropriate services, ensuring security and load balancing.
- **Data Processing Layer:** Executes the queries on databases, data warehouses, or APIs to retrieve relevant data.
- **Database Layer:** Stores the data and serves as the backend, supporting SQL, NoSQL, or graph databases, depending on the use case.

#### 2. User Flow:

The user flow for DataQueryAI can be summarized in the following steps:

- **User Login:** The user logs into the platform via secure authentication (OAuth/JWT).
- **Query Input:** The user types a query in natural language or uses a structured query form.
- **Query Execution:** The query is executed on the relevant data sources or databases.
- **Results Display:** The system processes and visualizes the results in an interactive format (e.g., tables, graphs, or reports).
- **User Feedback/Refinement:** The user can refine the query based on the results or provide feedback for further improvement.

### 3.UI/UX Considerations:







For DataQueryAI, the UI/UX should prioritize simplicity, intuitiveness, and responsiveness. Key considerations include:

- **Natural Language Input:** Make it easy for users to type queries in natural language with auto-suggestions or corrections.
  - **Clear Visualizations:** Display results in interactive formats like tables, charts, or graphs to help users quickly interpret data.
  - **Real-Time Feedback:** Provide instant query processing feedback (e.g., loading indicators or progress bars) for a smoother experience.
  - **Accessible Design:** Ensure the platform is accessible with a clean layout, readable fonts, and contrast for all users.
  - **Personalization:** Offer personalized recommendations based on user behavior and query history.
-

## Phase-4: Project Planning (Agile Methodologies)




### Objective:

Break down development tasks for efficient completion.



Sprint	Task	Priority	Duration	Deadline	Assigned To	Dependencies	Expected Outcome
Sprint 1	Environment Setup & API Integration	 High	6 hours (Day 1)	End of Day 1	L.Pranathi	Google API Key, Python, Streamlit setup	API connection established & working
Sprint 1	Frontend UI Development	 Medium	2 hours (Day 1)	End of Day 1	B.Shivani	API response format finalized	Basic UI with input fields
Sprint 2	Logo Search & Comparison	 High	3 hours (Day 2)	Mid-Day 2	B.Kavya	API response, UI elements ready	Search functionality with filters
Sprint 2	Error Handling & Debugging	 High	1.5 hours (Day 2)	Mid-Day 2	L.Pranathi, G.Saranya	API logs, UI inputs	Improved API stability
Sprint 3	Testing & UI Enhancements	 Medium	1.5 hours (Day 2)	Mid-Day 2	B.Shivani, B.Kavya	API response, UI layout completed	Responsive UI, better user experience
Sprint 3	Final Presentation & Deployment	 Low	1 hour (Day 2)	End of Day 2	Entire Team	Working prototype	Demo-ready project

### Sprint Planning with Priorities



#### Sprint 1 – Setup & Integration (Day 1)

- ( **High Priority**) Set up the **environment** & install dependencies.
- ( **High Priority**) Integrate **Google Gemini API**.
- ( **Medium Priority**) Build a **basic UI** with input fields.

#### Sprint 2 – Core Features & Debugging (Day 2)

- ( **High Priority**) Implement **search & comparison functionalities**. ( **High Priority**) Debug API issues & handle **errors in queries**.

#### Sprint 3 – Testing, Enhancements & Submission (Day 2)

- ( **Medium Priority**) Test API responses, refine UI, & fix UI bugs.
- ( **Low Priority**) Final **demo preparation & deployment**.

# Phase-5: Project Development

## Objective:

Implement core features of the DataQueryAI intelligent Data Analysis with Google TAPAS.

## Key Points:

### 1. Technology Stack Used:

- **Frontend:** Google collab
- **Backend:** Pandas, Pipeline from transformers
- **Programming Language:** Python

### 2. Development Process:

- **Requirement Gathering:** Understand user needs and data sources, defining the types of queries and security needs.
- **Architecture Design:** Build a scalable system with an API gateway, and database layers .
- **Development:** Implement core features, including query processing, data retrieval, and interactive visualizations.
- **Testing:** Conduct unit, integration, and user acceptance testing to ensure functionality and performance.
- **Deployment:** Launch on a cloud platform, ensuring scalability and security.
- **Maintenance & Updates:** Continuously improve the system with new features, bug fixes, and optimizations based on user feedback.

### 3. Challenges & Fixes:

- **Challenge:** Handling ambiguous queries.  
**Fix:** Implement context-driven clarification and contextual NLP models.
  - **Challenge:** Query performance and scalability  
**Fix:** Use query optimization techniques and distributed data processing for efficient scaling.
-

## Phase-6: Functional & Performance Testing

### Objective:

Ensure that the DataQueryAI intelligent DataAnalysis with Google TAPAS works as expected.

Test Case ID	Category	Test Scenario	Expected Outcome	Status	Tester
TC-001	Functional Testing	Query "Best budget cars under ₹10 lakh"	Relevant budget cars should be displayed.	✓ Passed	Tester 1
TC-002	Functional Testing	Query "Motorcycle maintenance tips for winter"	Seasonal tips should be provided.	✓ Passed	Tester 2
TC-003	Performance Testing	API response time under 500ms	API should return results quickly.	⚠ Needs Optimization	Tester 3
TC-004	Bug Fixes & Improvements	Fixed incorrect API responses.	Data accuracy should be improved.	✓ Fixed	Developer
TC-005	Final Validation	Ensure UI is responsive across devices.	UI should work on mobile & desktop.	✗ Failed - UI broken on mobile	Tester 2
TC-006	Deployment Testing	Host the app using Streamlit Sharing	App should be accessible online.	🔗 Deployed	DevOps

---

## Final Submission

1. **Project Report Based on the templates**
2. **Demo Video (3-5 Minutes)**
3. **GitHub/Code Repository Link**
4. **Presentation**