Hackathon Project Phases Template that ensures students can complete it efficiently while covering all six phases. The template is structured to capture essential information without being time-consuming.

Hackathon Project Phases Template

Project Title:

DATA QUERY AI: Intelligent data analysis with GOOGLE TAPAS

Team Name:

DYNAMO

Team Members:

- Thakur Chandana
- Mulukanuri Vishva
- G. varshith
- S. sathwika

Phase-1: Brainstorming & Ideation

Objective:

- Google TAPAS addresses the challenge of accessing and analyzing structured tabular data, which traditionally requires technical expertise in SQL, making it difficult for non-technical users to derive insights. By enabling natural language queries, TAPAS makes data analysis more accessible, empowering users to make faster, data-driven decisions without needing technical skills.
- Google TAPAS simplifies data analysis by allowing users to query tabular data in natural language, eliminating the need for technical expertise in SQL or coding. Its impact includes empowering non-technical users, improving decision-making speed, and increasing overall productivity by automating data queries and insights extraction.

Key Points:

1. **Problem Statement:**
   People often struggle to analyze data in tables because it requires technical skills like knowing

SQL. This makes it hard for non-technical users to quickly get useful information from data and make decisions.

2. **Proposed Solution:**
   Google TAPAS allows users to query and analyze tabular data using simple, natural language instead of needing technical skills like SQL. This makes it easy for non-technical users to quickly get insights from data and make informed decisions without requiring any coding knowledge.

3. **Target Users:**

   Business Analysts and Managers who need quick, easy data insights without coding.

   Data-Driven Teams in sales, marketing, and finance for fast analysis.

   Small and Medium Businesses (SMBs) without technical resources.

   Educators and Students learning data analysis without complexity.

4. **Expected Outcome:**
   The project will enable non-technical users to easily query and analyze tabular data using natural language, making data insights accessible without coding. It will improve decision-making speed, enhance productivity, and democratize data access across organizations, empowering a broader audience to make informed, data-driven decisions.

Phase-2: Requirement Analysis

Objective:

- A powerful Python library for data manipulation and analysis, ideal for handling tabular data. A high-performance web framework for building APIs in Python. It enables the creation of efficient and fast RESTful APIs for serving data and processing queries in real-time. An ASGI server used with FastAPI for running Python web applications. A framework for building interactive web applications with Python, used to create user-friendly dashboards and visualizations.

Key Points:

1. **Technical Requirements:** Python ,Transformers for NLP ,Fast API ,Uvicorn ,Streamlit ,Docker ,Pandas.
2. **Functional Requirements**: Natural Language Query Processing , Data Understanding and Mapping , Real-time Data Processing , Data Aggregation and Calculation , Contextual Query Handling , Custom Query Execution , User-Friendly Interface.
3. **Constraints & Challenges**: Building Data Query AI using Google TAPAS involves addressing a variety of challenges related to natural language processing, data integration, scalability, security, and user experience. The complexity of accurately interpreting and querying data in natural language while ensuring high performance, security, and accessibility makes the development of such systems both challenging and resource-intensive.

Phase-3: Project Design

Objective:

- Create the architecture and user flow.

Key Points:

1. **System Architecture Diagram:** (Simple sketch or flowchart)
2. **User Flow:** (How a user will interact with the project)
3. **UI/UX Considerations:** (If applicable, wireframe or basic layout)

Phase-4: Project Planning (Agile Methodologies)

Objective:

Requirements Gathering and Initial Setup
Data Preprocessing and Initial NLP Integration
Basic Query Execution and Aggregation
Advanced Query Interpretation and Contextual Understanding

Backend Development and API Integration
Frontend Development and User Interface
Security, Privacy, and Data Protection

Key Points:

1. **Sprint Planning: 1.**code implementation 2.prerequisites 3.code execution 4.UI design

2. **Task Allocation: 1.varshith**: code building and implementation 2.**chandana**:prerequisites **3.vishva:**code execution and documentation **4.sathwika**:UI design

3. **Timeline & Milestones:** code and testing the execution for first 12 hoursl

---

Phase-5: Project Development

Objective:

Requirements:

FastAPI

Streamlit

Transformers

Uvicorn

Pandas

Docker

Key Points:

1. **Technology Stack Used:** FastAPI ,Pandas ,Transformers ,Streamlit ,Docker
2. **Development Process:**
   Setting Up the Environment:
   install the required libraries for the project:
   Pandas ,Transformers ,FastAPI ,Uvicorn ,Streamlit.
   CSV Dataset Loading
   The next step was to load the dataset (typically a **CSV** file) that the application would query. This dataset contains structured tabular data that the user will interact with asking questions.
   FastAPI Backend Setup:
   The backend of the application was created using **FastAPI**, which provides a fast and efficient framework for building REST APIs in Python. FastAPI was chosen to handle user requests, process the natural language queries, and connect to the TAPAS model for generating responses.
   Streamlit Frontend Setup:
   The frontend of the application was built using Streamlit, which is a popular framework for quickly creating interactive web apps for data science projects. Streamlit allows users to interact with the model without needing any technical knowledge.
   User Interface: The interface features an input field where the user can type natural language queries.
   Query Submission: The frontend sends the query to the FastAPI backend via HTTP POST requests.
   Display Results: The results are returned to the frontend and displayed in an easily interpretable format (e.g., text or tables)
   Connecting Frontend and Backend:
   Running the FastAPI Backend: The backend server was run using Uvicorn. This server listens for incoming POST requests and processes them asynchronously.
   Running the Streamlit Frontend: The frontend was run using **Streamlit**, which automatically generates an interactive interface for the user.
   Testing the Integration:
   After connecting both parts, extensive testing was conducted to ensure the entire system worked seamlessly. This included:
   Verifying that user queries were properly handled by the FastAPI backend.
   Ensuring that the TAPAS model returned correct and relevant answers based on the tabular data.

3. **Challenges & Fixes:**
   Challenge: The TAPAS model is pre-trained for specific tasks, and while it can be fine-tuned, adapting it to highly domain-specific data or specialized queries can be a challenge.
   Solution: Fine-tuning the TAPAS model on domain-specific data or augmenting it with additional training examples can make the system more robust for specialized use cases.

- **Challenge**: Seamlessly integrating the **FastAPI** backend with **Streamlit** frontend can be difficult, especially when handling multiple API requests or when there are issues with the server deployment (such as delays in response times).
  - **Solution**: Proper testing, error handling, and implementing **asynchronous programming** in FastAPI can help mitigate these integration issues. Additionally, maintaining good communication between the frontend and backend and using a robust API architecture will ensure smooth interactions.

Phase-6: Functional & Performance Testing

Objective:

"Data Query AI: Intelligent Data Analysis with Google TAPAS," the project deployment involves several key steps, which we'll outline below. Google TAPAS (Tabular Pre-trained Language Model) is an AI model designed to perform natural language processing (NLP) on tabular data. It enables users to query structured data (like spreadsheets or databases) using natural language, which simplifies data analysis for users who may not be familiar with traditional query languages like SQL.

Key Points:

1. Test Cases Executed: Model Query Understanding
2. Test Case 1: Check if the model can correctly interpret simple queries.
3. Query: "What is the total sales for 2023?"
4. Expected Result: Model should return the correct aggregated sales value for the year 2023.
5. Test Case 2: Test if the model can understand and interpret a multi-step query.
6. Query: "What is the average sales by region in 2023?"
7. Expected Result: Model should return the correct average sales per region for the year 2023.
8. Test Case 3: Check if the model can handle queries with specific column names.
9. Query: "Show the sales for the product 'XYZ' in January 2023."
10. Expected Result: Model should return the sales data for product 'XYZ' in the specified month.

11. b. Data Accuracy and Integrity

12. Test Case 4: Check if the model can retrieve accurate data from the database.

13. Query: “What is the total revenue for all products in Q1 of 2024?”

14. Expected Result: The system should return the correct total revenue for Q1 2024 from the database.

Bug Fixes & Improvements: Bug: "What is the total revenue for 2023?" returns incorrect data. Fix: Debug and refine the query translation process from TAPAS to SQL. Ensure the correct table columns and values are being used in the generated SQL statements. Add more robust error handling for data retrieval, ensuring that fallback data or error messages are provided when issues occur.

Final Validation: After assessing these points, if the project successfully addresses the outlined requirements—accurate data query generation, seamless integration with Google TAPAS, user-friendly design, scalability, and security—you can conclude that the project meets the initial goals for intelligent data analysis with Google TAPAS. If any critical aspects are missing or underperforming, additional work would be needed to address those gaps.

Deployment (if applicable): localhost:8502

Final Submission

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