**Hackathon Project Phases Template** for the **AutoSage App** project.

# **Hackathon Project Phases Template**

## **Project Title:**

**Data Query using Google TAPAS**

## **Team Name:**

**Data Query**

## **Team Members:**

• **M.Likitha Reddy**

**• Vanam. Sathya Sri Asritha**

## **Phase-1: Brainstorming & Ideation**

### **Objective:**

The objective of a data query AI using Google TAPAS is to unable uses to ask natural language questions about tableau data allowing AI to accurately exact information from tables.

### **Key Points:**

### 1. **Understanding TAPAS**

* TAPAS is based on **BERT**, but specialized for table-based question answering.
* It supports **semantic parsing**—allowing users to ask natural language questions on structured data (tables).
* Unlike SQL, it doesn’t require predefined query structures.

### 2. **Input Data Format**

* TAPAS takes input in the form of **tabular data (CSV, Excel, JSON tables)**.
* It requires well-structured column headers and rows.

### 3. **Types of Queries Supported**

* **Aggregation Queries** (Sum, Average, Count, etc.)
* **Comparative Queries** (e.g., "Which country has the highest GDP?")
* **Filtering Queries** (e.g., "Show all employees from New York.")
* **Yes/No Questions** (e.g., "Is the revenue greater than $1M?")
* **Multi-turn Queries** (context-aware queries)

### 4. **Query Execution Process**

* TAPAS processes the natural language question.
* It identifies relevant rows and columns in the table.
* If necessary, it applies **aggregation functions**.
* Returns a direct answer, not an SQL query.

### 5. **Implementation and Usage**

* Available via **Google AI APIs** and **Hugging Face Transformers** (tapas-base, tapas-large models).
* Also available in **Google AI Notebooks** and **Colab**.

### 6. **Advantages of TAPAS**

* **No need for SQL knowledge**—can query with natural language.
* **Pre-trained models**—handles complex table reasoning.
* **Scalable**—works well on large tables.

### 7. **Limitations**

* Limited support for **complex joins** across multiple tables.
* May require **fine-tuning** for domain-specific queries.
* Works best for **structured and well-formatted tables**.

## **Phase-2: Requirement Analysis**

### **Objective:**

### **Key Points:**

**System Requirements**

* Python **3.6+**
* At least **8GB RAM** (16GB+ recommended for large tables)
* (Optional) **GPU** for faster inference

· **Input Data Requirements**

* Structured **table format** (CSV, JSON, Pandas DataFrame)
* Clear **column headers** for better query accuracy

· **Query Format**

* Natural language queries (e.g., "What is the total revenue?")
* Supports **aggregation, comparison, and filtering**

## **Phase-3: Project Design**

### **Objective:**

**Step 1: Load Dependencies**

**Step 2: Load the Model & Tokenizer**

**Step 3: Load Tabular Data**

#### ****Step 4: Accept User Query****

#### ****Step 5: Convert Query & Table to Model Input****

#### **Step 6: Get Predictions**

## 

## **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 | Likitha &Ashritha | Google API Key, Python, Streamlit setup | API connection established & working |
| Sprint 1 | Frontend UI Development | 🟡 Medium | 2 hours (Day 1) | End of Day 1 | Ashritha | API response format finalized | Basic UI with input fields |
| Sprint 2 | **Google AI APIs** and **Hugging Face Transformers** | 🔴 High | 3 hours (Day 2) | Mid-Day 2 | Likitha | API response, UI elements ready | Search functionality with filters |
| Sprint 2 | Error Handling & Debugging | 🔴 High | 1.5 hours (Day 2) | Mid-Day 2 | Ashritha | API logs, UI inputs | Improved API stability |
| Sprint 3 | Testing & UI Enhancements | 🟡 Medium | 1.5 hours (Day 2) | Mid-Day 2 | Likitha | 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 AutoSage App.

### **Key Points:**

1. **Technology Stack Used:**
   * **Frontend:** http
   * **Backend:** Python
   * **Programming Language:** Python
2. **Development Process:**
   * Data analysed
   * Retirevivg data in coloum , row selection wise
   * Optimize **search queries for performance and relevance**.
3. **Challenges & Fixes:**
   * **Challenge:** catching warnings.  
      **Fix:** Implement **caching** to store frequently queried results.
   * **Challenge:** fixing problems&code  
      **Fix:** Optimize queries to fetch **only necessary data**.

## **Phase-6: Functional & Performance Testing**

### **Objective:**

Ensure that the AutoSage App works as expected.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Category** | **Test Scenario** | **Expected Outcome** | **Status** | **Tester** |
| TC-001 | Functional Testing | Query "CRUD frameworks” | Relevant data retireved. | ✅ Passed | Tester 1 |
| TC-002 | Functional Testing | Query "Coloum,row operations" | Retrieved data which is needed. | ✅ Passed | Tester 2 |

## **Final Submission**

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