**Hackathon Project Phases Template** for the **DataQueryAI** project.

# **Hackathon Project**

## **Project Title:**

DataQueryAI

## **Team Name:**

QueryBots

## **Team Members:**

* A. Shivani Yadav
* A. Deepthi
* M. Nikitha
* G. Srinija
* B. Bhanu Sri

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

### **Objective:**

### The objective is to initiate a collaborative process of brainstorming and ideation for DataQueryAI, focusing on defining the core functionalities, user needs, and potential enhancements to the platform.

1. **Problem Statement:**

* DataQueryAI uses generative AI to simplify data analysis by allowing users to upload CSV files and receive contextually accurate insights. It empowers business analysts to quickly extract valuable information from complex datasets, improving decision-making processes. This solution reduces manual effort, enhances accuracy.

1. **Proposed Solution:**
   * The proposed system, DataQueryAI, enables users to upload CSV files and perform automated data analysis through a user-friendly Streamlit interface. It leverages the power of generative AI (Gemini) to provide contextually accurate insights by interpreting user queries based on the dataset. The system also includes data visualization features to help users better understand trends and patterns in their data.
2. **Target Users:**
   * **Business Analysts**: Professionals who need to quickly analyze data, identify trends, and make data-driven decisions without the need for complex coding or manual analysis.
   * **Data Scientists**: Users who require quick insights from large datasets and want to use AI to assist in identifying patterns or generating automated reports.
   * **Decision-Makers/Managers**: Executives and managers who need to extract actionable insights from data to inform strategic decisions.
3. **Expected Outcome:**

* The expected outcome of DataQueryAI is to enable users to quickly extract accurate insights from their data, streamlining analysis and decision-making. This will improve productivity by automating data interpretation and reducing manual effort.

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

### **Objective:**

### The objective is to gather and analyze detailed functional and non-functional requirements for DataQueryAI, ensuring alignment with user needs and system expectations.

### **Key Points:**

1. **Technical Requirements:**

* **Programming Language**: Python
* **Backend**: Google Gemini API (for generative AI and natural language processing)
* **Frontend**: Streamlit Web Framework (for UI, file uploads, query input, and visualizations)
* **Data Processing**: Pandas (for handling and processing CSV data)
* **Visualization**: Matplotlib and Seaborn (for dynamic graphing and charting)

1. **Functional Requirements:**
   * Ability to **upload and process CSV files.**
   * Answer **data-related queries using AI.**
   * Generate **actionable insights from the data.**
   * Provide an **intuitive, easy-to-navigate interface for interacting with data,** uploading files, and querying insights.
2. **Constraints & Challenges:**
   * Ensuring real-time updates from **Gemini API**.
   * Handling **API rate limits** and optimizing API calls.
   * Providing a **smooth UI experience** with Streamlit.

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

### **Objective:**

Develop the architecture and user flow of the application.



### **Key Points:**

1. **System Architecture:**
   * User enters csv file and asks query .
   * Query is processed using **Google Gemini API**.
   * AI model fetches and processes the data.
   * The frontend displays **data preview, data summary, data visualisation and AI-Powered Data Insights**.
2. **User Flow:**
   * Step 1: User enters a query (e.g., "Give the total profit").
   * Step 2: The backend **calls the Gemini Flash API** to retrieve vehicle data.
   * Step 3: The app processes the data and **displays results** in an easy-to-read format.
3. **UI/UX Considerations:**
   * **Minimalist, user-friendly interface** for seamless navigation.
   * **Filters for total pieces, sales etc**.
   * **Dark & light mode** for better user experience.

## 

## **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 | Shivani | Google API Key, Python, Streamlit setup | API connection established & working |
| Sprint 1 | Frontend UI Development | 🟡 Medium | 2 hours (Day 1) | End of Day 1 | Srinija | API response format finalized | Basic UI with input fields |
| Sprint 2 | Sales Search & Comparison | 🔴 High | 3 hours (Day 2) | Mid-Day 2 | Deepthi | API response, UI elements ready | Search functionality with filters |
| Sprint 2 | Error Handling & Debugging | 🔴 High | 1.5 hours (Day 2) | Mid-Day 2 | Nikhitha | API logs, UI inputs | Improved API stability |
| Sprint 3 | Testing & UI Enhancements | 🟡 Medium | 1.5 hours (Day 2) | Mid-Day 2 | Shivani | API response, UI layout completed | Responsive UI, better user experience |
| Sprint 3 | Final Presentation & Deployment | 🟢 Low | 1 hour (Day 2) | End of Day 2 | Bhanu Sri | 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:** Streamlit
   * **Backend:** Google Gemini Flash API
   * **Programming Language:** Python
2. **Development Process:**
   * Implement **API key authentication** and **Gemini API integration**.
   * Developing and analyzing the given csv dataset.
   * Optimize **search queries for performance and relevance**.
3. **Challenges & Fixes:**
   * **Challenge:** Delayed API response times.  
      **Fix:** Implement **caching** to store frequently queried results.
   * **Challenge:** Limited API calls per minute.  
      **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 "Give the total profits" | Gives the total profit in number. | ✅ Passed | Shivani |
| TC-002 | Functional Testing | Query "What are the total profits?" | Gives the total profit column. | ✅ Passed | Nikhitha |
| TC-003 | Bug Fixes & Improvements | Fixed incorrect API responses. | Data accuracy should be improved. | ✅ Fixed | Srinija |
| TC-004 | Final Validation | Ensure UI is responsive across devices. | UI should work on mobile & desktop. | ❌ Failed - UI broken on mobile | Deepthi |
| TC-005 | Deployment Testing | Host the app using Streamlit Sharing | App should be accessible online. | 🚀 Deployed | Bhanu Sri |

## **Final Submission**

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