**Updated Development Roadmap: Data Analyst Client Simulator (DACS)**

This roadmap outlines the planned development phases, building the core functionality first and iteratively adding more complex features. The decision to use Gemini is now a key driver in the technology stack.

**Phase 1: Core Functionality - Scenario & Dataset Generation MVP**

* **Goal:** Implement the basic functionality of generating diverse, domain-based scenarios and corresponding datasets, and displaying them through a simple web interface.
* **Duration (Estimate):** 5-9 weeks (adjusted to account for initial research).
  1. **Project Setup & Environment (Week 1):**
     + Set up your Python development environment (virtual environment, install necessary libraries: Flask, google-generativeai, Pandas, potentially Faker).
     + Initialize a Git repository (GitHub, GitLab, or Bitbucket).
     + Basic Flask application setup (minimal structure, routing).
  2. **Gemini API Exploration & Integration (Weeks 1-2):**
     + **Research and Setup:** Thoroughly explore the Google Gemini API and its capabilities for text generation relevant to scenario creation. Obtain necessary API keys and set up authentication.
     + **Initial Prototyping:** Experiment with different prompts and parameters for Gemini to generate diverse and realistic client scenarios based on various data domains (finance, botany, marketing, etc.). Focus on capturing Business Task and Data Provided.
  3. **Dataset Generation Logic (Weeks 2-3):**
     + **Schema Design from Scenario:** Implement the logic to extract information from the generated scenario (especially "Data Provided") to define the schema (column names and data types) for the dataset.
     + **Data Population with Pandas & Faker:** Write Python code using Pandas and Faker to generate realistic data according to the defined schema.
     + **Implement Data Quality Issues:** Integrate techniques to introduce realistic data quality issues (missing values, outliers, inconsistencies).
  4. **Backend Integration (Weeks 3-4):**
     + Integrate the Gemini-powered scenario generation and dataset generation modules within the Flask application.
     + Create Flask routes to trigger the generation process.
     + Implement logic to pass information between the scenario and dataset generation modules.
     + Implement the logic to generate and make the CSV available for download.
  5. **Basic Front-End Development (Weeks 4-6):**
     + Create a simple HTML template for the title page.
     + Implement the "Generate Scenario" button functionality using JavaScript (minimal).
     + Display the generated scenario components on the page.
     + Provide a link or button for CSV download.
     + Implement basic error handling (pop-up notification).
  6. **Testing & Refinement (Weeks 6-8):**
     + **Manual Testing:** Thoroughly test the application by generating various scenarios across different data domains to ensure:
       - Scenarios are diverse and realistic.
       - Datasets correspond to the scenarios.
       - Data quality issues are present and addressable.
       - The user interface functions as expected.
       - Error handling is working.
       - CSV download works correctly.
     + **Code Review & Refinement:** Review your code for clarity, efficiency, and adherence to good practices. Add comments as needed.
  7. **Documentation & Initial Tracking (Week 9):**
     + Document the codebase, focusing on the core logic for scenario and dataset generation.
     + Create a basic README file explaining how to set up and run the application.
     + Implement basic logging to track when scenarios are generated (to align with future success metrics).

**Phase 2: Refinements and Preparation for Future Features**

* **Goal:** Improve the existing functionality and lay the groundwork for domain selection and dataset size control.
* **Duration (Estimate):** 4-7 weeks.
  1. **Scenario Variety and Realism Enhancement by Domain (Weeks 1-2):**
     + Refine Gemini prompts to generate more nuanced and realistic scenarios specific to different data domains (finance, botany, marketing, etc.).
     + Explore techniques to introduce more contextually relevant data quality issues for each domain.
  2. **Dataset Logic Improvements and Size Control Foundation (Weeks 2-3):**
     + Improve the logic for ensuring stronger consistency between the domain-specific scenario and the generated dataset.
     + Start designing the backend logic and data structures to handle the dataset size control. This might involve exploring Pandas functionalities for creating datasets of varying sizes.
  3. **Planning for Domain Selection UI (Weeks 3-4):**
     + Design the data model and backend logic to categorize and filter scenarios by domain.
     + Start planning the structure of the UI element for domain selection (e.g., dropdown menu, list of categories).
  4. **Front-End UI/UX Improvements (Weeks 4-5):**
     + Focus on improving the user experience based on your usage.
     + Consider basic styling for a more visually appealing interface.
     + Ensure clear instructions or guidance are present.
  5. **Code Refactoring and Modularization (Week 6):**
     + Refactor the codebase to improve its modularity and scalability, preparing for the integration of domain selection and size control.
  6. **Advanced Testing & Error Handling (Week 7):**
     + Implement more robust error handling and logging mechanisms.
     + Write unit tests for critical components of the scenario and dataset generation logic.

**Phase 3: (Future) Implementing Domain Selection and Dataset Size Control**

* **Goal:** Implement the UI elements and backend logic for users to select the data domain and control the dataset size.
* **Duration (Estimate):** 4-6 weeks.
  1. **Front-End Development for Domain Selection and Size Slider (Weeks 1-2):**
     + Implement the UI element for domain selection (e.g., a dropdown menu populated with available domains).
     + Implement the dataset size control (likely a slider that maps to the number of rows).
  2. **Backend Integration for Domain Selection (Weeks 2-3):**
     + Integrate the front-end domain selection with the backend, allowing users to filter scenario generation by domain.
     + Modify the scenario generation logic to prioritize or focus on the selected domain.
  3. **Backend Integration for Dataset Size Control (Weeks 3-4):**
     + Integrate the front-end size slider with the backend.
     + Modify the dataset generation logic to dynamically create datasets with the specified number of rows using Pandas.
  4. **Testing and Refinement (Weeks 5-6):**
     + Thoroughly test the domain selection and dataset size control features, ensuring they function correctly and the generated scenarios and datasets are appropriate for the selected domain and size.

**Phase 4: (Future) Implementing the Interactive AI Client**

* **Goal:** Integrate the functionality for users to interact with the Gemini AI client to ask clarifying questions and adapt the scenario.
* **Duration (Estimate):** 5-8 weeks.
  1. **Gemini API for Interaction:** Explore Gemini's capabilities for conversational AI and its ability to understand and respond to user queries within the context of the generated scenario and data.
  2. **Backend Integration:** Develop the backend logic to handle user input and communicate with the Gemini API for generating responses.
  3. **Front-End Interaction Design:** Design and implement the user interface elements for interacting with the AI client (e.g., a chat window).
  4. **State Management:** Implement logic to maintain the conversation history and context.
  5. **Testing and Refinement:** Thoroughly test the interaction flow and the Gemini client's ability to understand and respond appropriately to data analysis-related questions.

**Phase 5: (Future) Feedback Mechanism and Portfolio Building Support**

* **Goal:** Develop the AI-powered feedback mechanism and features to support portfolio building.
* **Duration (Estimate):** Ongoing, depending on the features prioritized.
  1. **Define Feedback Criteria:** Clearly define the criteria for evaluating the data analyst's work.
  2. **Develop Feedback Logic (Potentially with Gemini):** Explore using Gemini's capabilities to analyze user submissions (code, reports) and provide feedback.
  3. **Backend Integration:** Integrate the feedback mechanism into the application.
  4. **Front-End Presentation:** Design how the feedback will be presented to the user.
  5. **Implement Portfolio Idea Generation Features:** Develop ways for the platform to suggest project ideas based on generated scenarios.
  6. **User Account Management (If Needed):** Implement user registration, login, and potentially progress tracking.
  7. **Deployment and Scaling Considerations:** Plan for deploying the application to a more accessible platform if you intend to share it.