**Detailed Development Roadmap for Phase 1 - Core Functionality (MVP)**

This roadmap includes specific actions, estimated durations, and a clear view of task dependencies. We'll focus on getting the core features working first, with integrated testing and documentation.

* **Overall Goal:** Implement the basic functionality of generating scenarios and corresponding datasets, and displaying them through a simple web interface.
* **Overall Duration (Estimate):** 9 Weeks (Starting January 15th, 2025)

**Phase 1 Detailed Breakdown:**

* **Week 1: Environment Setup (January 15th - January 22nd)**
  + **Task 1.1: Set Up Python Environment:**
    - *Action:*
      * Install Python 3.10+
      * Create virtual environment (venv)
      * Activate the virtual environment.
    - *Documentation:* Record the python version in your Technology Choices document. Add the virtual environment creation in the README.md.
    - *Testing:* Check Python version in your terminal with python --version (or python3 --version) and check that the virtual environment is active with the appropriate command.
    - *Subtasks:*
    - Install Python
    - Create virtual environment
    - Activate virtual environment
    - Verify Python version
    - Check virtual environment
  + *Dependencies:* None
  + **Task 1.2: Install Dependencies:**
    - *Action:* Install required libraries: flask, google-generativeai, pandas, and faker. Make sure to specify the exact versions.
    - *Documentation:* In your "Technology Choices" ClickUp Doc, add the versions of each package and their corresponding installation commands.
    - *Testing:* Test if libraries are installed by importing them to app.py
    - *Subtasks:*
      * Install flask
      * Install google-generativeai
      * Install Pandas
      * Install Faker
      * Verify installed versions
    - *Dependencies:* Task 1.1
  + **Task 1.3: Initialize Git Repository:**
    - *Action:* Create a GitHub repository and initialize a local git repository, and make your first commit with all files.
    - *Documentation:* Add your remote repo URL to the "Resources" section of your PRD in ClickUp.
    - *Testing:* Test your connection by pushing to your remote branch, and pulling down for the first time.
    - *Subtasks:*
    - Initialize git repository
    - add initial files
    - Make first commit
    - Setup remote repository.
    - *Dependencies:* Task 1.1
  + **Task 1.4 : Set up Flask Structure**
    - *Action:* Create basic app.py file, the templates folder, and the static folder.
    - *Documentation:* Write a simple description of each file and the folder they belong to, in the README.md.
    - *Testing:* Verify that the Flask application works correctly by visiting your local host.
    - *Subtasks:*
      * Create app.py
      * Create templates folder
      * Create static folder
      * Create basic flask setup
      * Visit http://127.0.0.1:5000/
    - *Dependencies:* Task 1.1
* **Week 2: Gemini API Integration (January 22nd - January 29th)**
  + **Task 2.1: Explore Gemini API:**
    - *Action:* Go through the Gemini documentation and experiment with the API using simple prompts.
    - *Documentation:* In your ClickUp "Research Notes" document, make notes of the documentation, API endpoints, responses, and everything that is relevant for your project. Make notes on the available models and which one would fit your needs best.
    - *Testing:* Test different APIs and generate responses, log their output and your opinions.
    - *Subtasks:*
    - Review Gemini API documentation
    - Obtain a Google Cloud API Key
    - Test text generation.
    - Explore different models.
  + *Dependencies:* Task 1.2
  + **Task 2.2: Implement Gemini API Call:**
    - *Action:* Implement your first Gemini API call using the python library and test that it can generate basic text from a simple request.
    - *Documentation:* Include a code snippet of your basic tests in the "Research Notes" document. Document all steps in your "Gemini API Documentation" in ClickUp.
    - *Testing:* Check that you obtain a response from the Gemini API. Log all information about your tests.
    - *Subtasks:*
    - Implement the Gemini API library.
    - Call Gemini API.
    - *Dependencies:* Task 2.1
* **Week 3 - 4: Logic Implementation (January 29th - February 12th)**
  + **Task 3.1: Scenario Generation Function:**
    - *Action:* Create the logic to generate your data with your prompt templates, add parsing and error handling for your api calls.
    - *Documentation:* Add code comments to explain the logic in your python file, also add all the different prompt templates to your ClickUp doc.
    - *Testing:* Call this function with different templates and verify the output you receive.
    - *Subtasks:*
    - Create different prompt templates.
    - Create generate\_scenario function.
    - Add parsing logic
    - Add error handling.
    - *Dependencies:* Task 2.2
  + **Task 4.1: Dataset Generation Function:**
    - *Action:* Implement the logic to create different data schemas, that will generate realistic data. Include logic for outliers, inconsistent formatting and missing data.
    - *Documentation:* Document your data schemas in your "Technology Choices" doc. Add comments to all your code.
    - *Testing:* Run your generate\_dataset() function several times, and verify that all parameters and different schemas are working properly.
    - *Subtasks:*
    - Create data schemas
    - Implement generate\_dataset() function
    - Add outlier logic
    - Add missing data logic
    - Implement inconsistent formatting.
    - *Dependencies:* Task 1.1
* **Week 5: Backend Integration (February 12th - February 19th)**
  + **Task 5.1: Flask Routes and Orchestration:**
    - *Action:* Create Flask routes /generate\_scenario and /download\_data, and implement the create\_scenario\_and\_data() orchestration function.
    - *Documentation:* Document the purpose of each route in code comments. Document the purpose of each function and its different parameters.
    - *Testing:* Test that the /generate\_scenario and /download\_data endpoints are correctly working.
    - *Subtasks:*
    - Create Flask /generate\_scenario route
    - Create Flask /download\_data route
    - Create create\_scenario\_and\_data() function
    - *Dependencies:* Task 3.1, Task 4.1
* **Week 6: Basic Front-End Implementation (February 19th - February 26th)**
  + **Task 6.1: Basic Layout**  
    \* *Action:* Create the index.html file, add basic CSS with style.css.  
    \* *Documentation:* Document all the HTML tags in the code with comments and their purpose. Add comments to all styles in style.css  
    \* *Testing:* Check that your basic layout shows up correctly.  
    \* *Subtasks:*
    - Create index.html file.
    - Design basic layout
    - link the style sheet  
      \* *Dependencies:* Task 1.1
  + **Task 7.1: JavaScript Interaction**
    - *Action:* Add javascript functionality to trigger scenario generation, display text, and download the csv.
    - *Documentation:* Add comments explaining the logic of the javascript code.
    - *Testing:* Verify the button for scenario generation and its corresponding download link.
    - *Subtasks:*
      * Add javascript for scenario generation
      * Add javascript for data display
      * Create CSV download link
      * Add error handling using javascript alerts
    - *Dependencies:* Task 5.1, Task 6.1
* **Week 7 - 8: Testing and Refinement (February 26th - March 12th)**  
  \* **Task 8.1 : Testing Phase 1**
  + *Action:* Test all different components of the application as a whole, and individually. Look for bugs and ways to improve the code.
  + *Documentation:* Take notes of all your findings, and create bug reports for every issue that arises. Make a lessons learned log to indicate all the new information and improvements that you have found during testing.
  + *Testing:* Manually test all different scenarios and data generations. Check different domains, and inputs. Test all your interactions with the UI and the responses of the application.  
    \* *Subtasks:*  
    \* Generate multiple scenarios and review  
    \* Inspect generated datasets  
    \* Verify data alignment
  + Test button functionality
  + Test download link  
    \* Test error handling  
    \* Fix any issues that arise.
  + *Dependencies:* Task 7.1
* **Week 9: Documentation and Tracking (March 5th-March 12th)**
  + **Task 9.1: Document and log:**
    - *Action:* Add all documentation to the Python code using code comments, finalize the README.md, and implement logging to record the application's process.
    - *Documentation:* Create a README.md, with all necessary instructions and a description of each component. Add detailed code comments to your python files, and log every important event of the application execution.
    - *Testing:* Verify that the comments are useful and that the README.md is clear, also make sure that logging is working as intended, logging the execution of the app and the errors correctly.
    - *Subtasks:*
    - Add comments to Python code
    - Create README.md file
    - Implement basic logging
    - *Dependencies:* Task 8.1