Explanation of the choice of techniques:

Elicitation Techniques:

- Brainstorming and Prototyping: These techniques were selected due to the project's limitation on involving external parties for requirement gathering. Given Alex's role as a Senior Data Analyst facing challenges in data complexity and quality, brainstorming allows the individual exploration of functionalities needed to streamline data cleaning, pre-processing, and integration. Prototyping aids in visualizing tools that could mitigate these challenges without requiring external involvement or surveys/interviews.

- Analysing Similar Web Apps: Given the absence of direct user interactions, exploring existing web apps allows for insights into functionalities, layouts, and user interactions that could be beneficial for App2's data visualization and analysis capabilities.

Documentation Techniques:

- User Stories for App2 (Web App): User stories were chosen to document App2's requirements as they facilitate a user-centric approach, aligning with Agile methodologies. These stories reflect Alex's frustrations and motivations, such as the need for efficient data analysis tools and insightful data visualization capabilities.

- Natural Language Specification for App1 (REST API): Natural language specification was chosen to define App1's REST API requirements in a structured and detailed format. This method allows clear documentation of functionalities like allowing programmatic access to data (get, update, add, delete) and adherence to RESTful design principles.

Prioritization Technique:

- The “MoSCoW” method was employed to prioritize the documented requirements for App2 (Web App) based on their criticality and time sensitivity. This technique categorizes requirements into four distinct levels:

- Must-Have: Requirements classified under 'must-have' are crucial functionalities necessary for the success of the web app. These directly address Alex's immediate needs, such as an intuitive interface and automated data cleaning tools, pivotal for efficient analysis considering the challenges faced with complex datasets.

- Should-Have: These requirements, while important, are less time-sensitive than the 'must-have' features. They encompass functionalities like customizable visualization options and informative dashboards, aiding comprehensive data interpretation and report communication.

- Could-Have: Identified functionalities under 'could-have' represent additional features that would greatly benefit the project but are not deemed critical for initial deployment. They include advanced analytics or external dataset integration, potentially enhancing the app's capabilities in future iterations.

- Won't-Have-for-Now: Requirements listed here are deemed of lower priority for the current phase of the project. They include advanced machine learning model deployment, considered for future implementations once foundational features are established.