**Functional Requirements:**

* Timeout Mechanism:
* Define the timeout duration for the function calls. This could be a configurable parameter.
* Implement a mechanism to track the execution time of the function calls.
* Failure Detection:
* Define what constitutes a failure. For example, if the function call exceeds the timeout duration or encounters an error.
* Implement logic to detect failures based on the defined criteria.
* Error Handling:
* Determine how errors will be handled when a failure is detected. This could include logging errors, displaying error messages to users, or retrying the operation.

**Non-Functional Requirements:**

* Performance:
* Ensure that the timeout mechanism does not introduce significant overhead or impact the performance of the application.
* Reliability:
* Ensure that the timeout mechanism is reliable and accurately detects failures in function calls.
* Customization:
* Provide options for customization, such as allowing users to configure timeout durations or error handling strategies.

**Constraints:**

* Google Apps Script Limitations:
* Consider any limitations or constraints imposed by the Google Apps Script environment, such as execution time limits or rate limits.
* User Experience:
* Balance the need for timely failure detection with the impact on the user experience. For example, avoid setting excessively short timeout durations that may frustrate users.

**Use Cases:**

***Scenario 1: Normal Execution:***

User triggers a function call, and it completes within the specified timeout duration without encountering any errors.

***Scenario 2: Timeout Occurs:***

User triggers a function call, but it exceeds the timeout duration. The system detects the timeout and handles it appropriately (e.g., displaying an error message).

***Scenario 3: Error Occurs:***

User triggers a function call, and it encounters an error before the timeout duration elapses. The system detects the error and handles it according to the defined error handling strategy.

**Dependencies:**

* External Services:
* Identify any external services or APIs that your Google Apps Script interacts with, as they may introduce additional factors affecting timeout handling and failure detection.

**Security Considerations:**

* Data Privacy:
* Ensure that timeout handling mechanisms do not compromise the security or privacy of user data.

**Testing:**

* Unit Testing:
* Develop unit tests to verify the behavior of the timeout handling mechanism under different scenarios, including normal execution, timeouts, and errors.
* Integration Testing:
* Perform integration testing to validate the interaction between the timeout handling mechanism and other components of the application.
* By conducting a thorough requirements analysis, you can ensure that the implementation of timeout handling meets the needs of your application while considering relevant constraints and considerations.