4 Testing

4.1 Unit Testing

Unity Application

Tool: Unity Test Framework

- UI Tests
  - Globe Scaling
    - Pass: Gesture enables and disables zooming mode
    - Fail: Zooming mode can't be enabled or disabled via the UI
  - Globe Rotation
    - Pass: Gesture enables and disables rotation mode
    - Fail: Rotation mode can't be enabled or disabled via the UI
  - Data Overlays
    - Pass: Selecting and deselecting data streams toggles their visibility on the globe
    - Fail: Data streams can't be toggled on and off
  - Application Exit
    - Pass: Pressing on the exit button closes the application
    - Fail: Application can't be exited gracefully from the UI
- Visualization Logic Tests
  - Data Refreshing
    - Pass: Given new data, old overlays are discarded and replaced with newly generated ones
    - Fail: Old overlays are not properly deleted or are not replaced with a correct version
  - Overlay Generation
    - Pass: For each data set, an overlay texture is generated for the globe
    - Fail: Overlays are malformed or missing for any data set
  - Multiple Overlays
    - Pass: Multiple overlays (up to 3) can be overlapped without interfering with each other
    - Fail: Additional overlays obstruct or cover existing ones
- Server Interaction
  - Data Retrieval
    - Pass: Application can make appropriate requests to the server for the data needed by the overlays enabled through the UI
    - Fail: Requests are malformed, unnecessary, or retrieving the wrong data

Data Server

Tool: Postman / Pytest

- External Data Retrieval
  - Data fetching
    - Pass: Application can make appropriate requests to external APIs for the data requested by the Unity client
    - Fail: Requests are malformed, unnecessary, or retrieving the wrong data
  - Query Building
    - Pass: Queries are built with the correct parameters to fetch the necessary data

        - Fail: Queries are missing filters or other necessary information to make the correct request
- Data processing
    - API layer
        - Pass: External data is mutated to fit the visualization client's standard and format
        - Fail: Data is lost or malformed in the mutation

4.2 Interface Testing

Server:

- External API Interface
    - This will be validated and verified using Unit Tests
    - This will be functionally tested using a tool like Postman to make sure requests are compatible with the used external APIs
- Internal API Interface
    - Unit tests will cover the individual API operations
    - Functional tests will be made using a tool like Postman to ensure that the server responds as expected to API calls

Unity Application:

- Server API interface
    - Unit tests will verify that the requests meet the specifications and requirements
    - Functional tests will be made using a tool like Postman to verify that requests are compatible with the server API specification

4.3 Integration Testing

We will test the external API -> HoloLens -> Unity engine path to ensure that our implementation doesn't overwhelm the hardware resources of the HoloLens. The HoloLens framerate, processor/memory/network utilization, and other performance metrics will be viewed and inspected through the Windows Device Portal and the Windows Performance Analyzer. These tools will allow us to ensure that our performance requirements (particularly, those pertaining to framerate and smoothness) are met.

An integration test for the Unity Application will involve verifying that given a mock data stream, the unity application will be able to generate the visualization and respond to various types of UI interactions. This test can be written using the Unity Test Framework in Play Mode.

An integration test for the server will involve verifying that for a request made to its API, it is able to generate the appropriate queries to return a mock stream of external data, mutate it to the appropriate standard, and return the new formatted data to the client.

4.4 System Testing

Functional Requirements :

- The application will be able to scale the three dimensional globe to the user's liking
    - The Unity UI unit tests and the Unity application integration test will verify this requirement is met.
- The application will be able to rotate the globe to give users a full 360 degree view of all geographical based data
    - The Unity UI unit tests and the Unity application integration test will verify this requirement is met.
- This application will have its own API layer that standardizes incoming data streams (Compatibility layer)
    - The Server unit tests and integration test will verify this requirement is met.
- This application will be able to take 3 different types of geographical data streams as input
    - The Unity UI unit tests and the Unity application integration test will verify this requirement is met.
- This application will be able to layer different types of data on the globe visualization for a multi type data visualization
    - The Unity UI unit tests and the Unity application integration test will verify this requirement is met.

4.5 Regression Testing

Unit tests and integration tests will automatically run on a test stage of our CI/CD pipeline, so that new additions are proven to work along existing functionality. This will be done with the Gitlab CI/CD pipelines tool and its merged result functionality which will run as if the changes from the source branch have already been merged into the target branch. Merge requests that fail these tests will not be merged into the project. The unit tests and integration tests will be made in accordance with the requirements, so this approach will verify that overall project requirements are still being met with every addition. Additions that don't include any necessary tests for added functionality will also not be merged.

4.6 Acceptance Testing

Extensive manual testing with the Hololens device will be done by team members and volunteer beta testers to verify that non-functional requirements are being met. The requirements that will be focused on during this testing stage are:

- This application will have a "pick up and go" ease of use where no technical developer is needed for a user to understand how to run the application
- This application will maintain a stable 60 fps through the Hololens visualization (Constraint)
- This application will need to run smoothly utilising the limited Hololens hardware performance (Constraint)

The client will also be asked to interact with the program and give feedback about each feature at different stages of early development and prototyping.

4.7 Security Testing (if applicable)

Not applicable.

4.8 Results

These tests will ensure that our project implementation will comply with the project requirements and meet the client's needs. A thorough testing methodology will allow us to avoid any hidden or edge-case problems that, while not impacting the majority of the user experience, could compromise the reliability and accuracy that our project intends to achieve. Our testing will also help us evaluate the overall usability of our project for the end user.