Test Design Document

To be used by the Testing Team (and Dev Team) in tandem with the Test Matrix to describe the design of tests.

Manual Tests

How to Use This Document

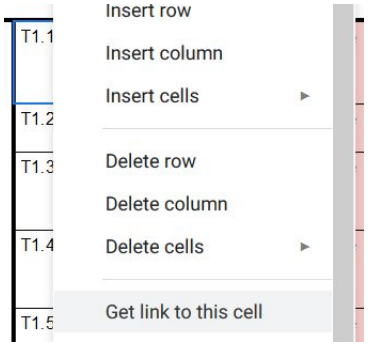
Template

Test ID#:
Test Name:
Test Designer:
Test Location: *(e.g. Example.test.tsx)*
Associate Files: *(e.g. test.csv)*
Test Description:

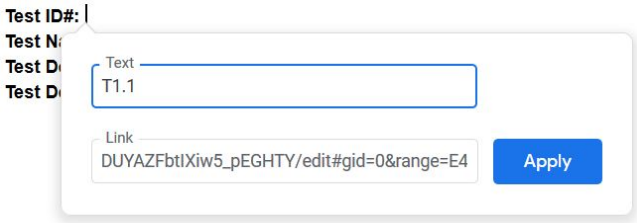
- *What requirement does this test cover?*
- *What is the purpose of this test?*
- *How is this test carried out (manual, automated, etc.)?*
- *etc.*

Linking to Test Matrix

1. Go to Test Matrix.
2. Right Click on corresponding ‘Test Case ID#’.



3. Select ‘Get link to this cell’.
4. Return to location of test in the Test Design Doc.
5. Right Click beside ‘Test ID#’ field and select ‘Link’ (or use Ctrl+K).
6. Put the ‘Test Case ID#’ in the ‘text’ field and paste the link from the Test Matrix in the ‘link’ field. Press the ‘Apply’ button.



7. … And *Voila!* You have a link to the specific cell in the Test Matrix.

Test ID#: T1.1
Test Name: Upload Incompatible File Type
Test Designer: Eileen van Heerde
Test Description: Some example description

Test Files

Test File Layout

Each Reactjs component will have a corresponding test file (identified by the '.test.tsx' extension) located in the ***__tests__*** folder within the *src* directory.

```
--'src'
  |
  |--'__tests__'
  |   |    Column.test.tsx
  |   |    Data.test.tsx
  |   |    Filter.test.tsx
  |   |    ParserComponent.test.tsx
  |   |    ParserInterface.test.tsx
  |   |    TimelineComponent.test.tsx
  |   |    TimelineInterface.test.tsx
  |   |
  |
  |--'components'
  |   |    Column.ts
  |   |    Data.ts
  |   |    Filter.ts
  |   |    ParserComponent.tsx
  |   |    ParserInterface.ts
  |   |    TimelineComponent.tsx
  |   |    TimelineInterface.ts
  |   |
```

Test Organization Within the Test Files

Within each test file, similar tests will be grouped together within `describe(){};` blocks where `it(){};` blocks will contain the actual test implementation.

```
describe('Rendering tests', () => {

        it('renders <ParserComponent /> that accepts CSV files', () => {
                ...
        });

        it('renders <ParserComponent /> that accepts TL files', () => {
                ...
        });

});
```

Tests

R1 Tests

Test ID#: T1.1

Test Name:
Test Designer:
Test Location: *(e.g. Example.test.tsx)*
Associate Files: *(e.g. test.csv)*
Test Description:

- *What requirement does this test cover?*
- *What is the purpose of this test?*
- *Test Outline:*
    - *Step 1*
    - *Step 2*
    - *Etc. (Last step should outline expected behaviour)*
- *How is this test carried out (manual, automated, etc.)?*
- *Etc.*

Test ID#: T1.2

Test Name:
Test Designer:
Test Location: *(e.g. Example.test.tsx)*
Associate Files: *(e.g. test.csv)*
Test Description:

- *What requirement does this test cover?*
- *What is the purpose of this test?*
- *Test Outline:*
    - *Step 1*
    - *Step 2*
    - *Etc. (Last step should outline expected behaviour)*
- *How is this test carried out (manual, automated, etc.)?*
- *Etc.*

Test ID#: T1.3

Test Name: Handling .csv file with multiple date formats
Test Designer: Eileen van Heerde
Test Location: ParserComponent.test.tsx
Associated Files: *TODO (create multiDateTest.csv)*
Test Description:

- This test focuses on making sure that we handle the specific case of a user potentially uploading a .csv file with multiple/inconsistent date formats according to design specifications.

- Test Outline:
    - Render a ParserComponent that accepts .csv files
    - Simulate an onChange input event that passes in a test csv file
    - ParserComponent state should be updated so that uploaded data is sorted by date and added to the ‘data’ value .
- This will be an automated test.

Test ID#: T1.4

Test Name: Handling .csv files with no temporal field
Test Designer: Eileen van Heerde
Test Location: ParserComponent.test.tsx
Associate Files: *TODO (create noDateTest.csv)*
Test Description:

- This test checks that we properly handle .csv files uploaded with no Date/Time/Temporal data field.
- Test Outline:
    - Render ParserComponent that accepts .csv files
    - Simulate an onChange input event that passes in *noDateTest.csv*
    - Error should be thrown and user should be informed that only temporal data can be uploaded.
- This test will be automated.

Test ID#: T1.5

Test Name:
Test Designer:
Test Location: *(e.g. Example.test.tsx)*
Associate Files: *(e.g. test.csv)*
Test Description:

- *What requirement does this test cover?*
- *What is the purpose of this test?*
- *Test Outline:*
    - *Step 1*
    - *Step 2*
    - *Etc. (Last step should outline expected behaviour)*
- *How is this test carried out (manual, automated, etc.)?*
- *Etc.*

Test ID#: T1.6

Test Name:
Test Designer:
Test Location: *(e.g. Example.test.tsx)*
Associate Files: *(e.g. test.csv)*
Test Description:

- *What requirement does this test cover?*
- *What is the purpose of this test?*

- *Test Outline:*
    - *Step 1*
    - *Step 2*
    - *Etc. (Last step should outline expected behaviour)*
- *How is this test carried out (manual, automated, etc.)?*
- *Etc.*

Test ID#: T1.7

Test Name:
Test Designer: Eileen van Heerde
Test Location: ParserComponent.test.tsx
Associate Files:
Test Description:

- *What requirement does this test cover?*
- *What is the purpose of this test?*
- *Test Outline:*
    - *Step 1*
    - *Step 2*
    - *Etc. (Last step should outline expected behaviour)*
- *How is this test carried out (manual, automated, etc.)?*
- *Etc.*

Test ID#: T1.8

Test Name:
Test Designer: Eileen van Heerde
Test Location: ParserComponent.test.tsx
Associate Files:
Test Description:

- *What requirement does this test cover?*
- *What is the purpose of this test?*
- *Test Outline:*
    - *Step 1*
    - *Step 2*
    - *Etc. (Last step should outline expected behaviour)*
- *How is this test carried out (manual, automated, etc.)?*
- *Etc.*

R2 Tests

Test ID#: T2.1

Test Name:
Test Designer:
Test Location: *(e.g. Example.test.tsx)*
Associate Files: *(e.g. test.csv)*
Test Description:

- *What requirement does this test cover?*
- *What is the purpose of this test?*
- *Test Outline:*
    - *Step 1*
    - *Step 2*
    - *Etc. (Last step should outline expected behaviour)*
- *How is this test carried out (manual, automated, etc.)?*
- *Etc.*

Test ID#: T2.2

Test Name:
Test Designer:
Test Location: *(e.g. Example.test.tsx)*
Associate Files: *(e.g. test.csv)*
Test Description:

- *What requirement does this test cover?*
- *What is the purpose of this test?*
- *Test Outline:*
    - *Step 1*
    - *Step 2*
    - *Etc. (Last step should outline expected behaviour)*
- *How is this test carried out (manual, automated, etc.)?*
- *Etc.*

Test ID#: T2.3

Test Name:
Test Designer:
Test Location: *(e.g. Example.test.tsx)*
Associate Files: *(e.g. test.csv)*
Test Description:

- *What requirement does this test cover?*
- *What is the purpose of this test?*
- *Test Outline:*
    - *Step 1*
    - *Step 2*
    - *Etc. (Last step should outline expected behaviour)*

- *How is this test carried out (manual, automated, etc.)?*
- *Etc.*

Test ID#: T2.4

Test Name:
Test Designer:
Test Location: *(e.g. Example.test.tsx)*
Associate Files: *(e.g. test.csv)*
Test Description:

- *What requirement does this test cover?*
- *What is the purpose of this test?*
- *Test Outline:*
    - *Step 1*
    - *Step 2*
    - *Etc. (Last step should outline expected behaviour)*
- *How is this test carried out (manual, automated, etc.)?*
- *Etc.*

Test ID#: T2.5

Test Name:
Test Designer:
Test Location: *(e.g. Example.test.tsx)*
Associate Files: *(e.g. test.csv)*
Test Description:

- *What requirement does this test cover?*
- *What is the purpose of this test?*
- *Test Outline:*
    - *Step 1*
    - *Step 2*
    - *Etc. (Last step should outline expected behaviour)*
- *How is this test carried out (manual, automated, etc.)?*
- *Etc.*

R3 Tests

R4 Tests

R5 Tests

R6 Tests

R7 Tests

R8 Tests

R9 Tests

R10 Tests

R11 Tests

R12 Tests

R13 Tests

R14 Tests

Unit Tests

These are tests for the methods described in the Design Documents. Tests that verify that pre/post conditions are met should be written for all of the methods listed below. Should primarily be written by developers, but testers are also encouraged to write them.

ParserComponent

1. constructor()
2. componentDidMount()
3. render()
4. isValid()
5. sortDate
6. inferTypes()
7. parse()
8. parseCsv()
9. parseTl()

Column

1. constructor()
2. show()
3. rescale()

Data

1. constructor()

Filter

1. constructor()
2. redefineRange()
3. addPredicate()
4. removePredicate()

Timeline

1. constructor()
2. componentDidMount()
3. render()
4. drawTimeline
5. ttOver()
6. ttUpdatePos()
7. ttMove()
8. ttLeave()
9. updateChart()

10. updateBars()
11. moveChart()
12. dragStarted()
13. dragged()
14. dragEnded()

Manual Tests

TODO (List some tests that might potentially need to be done manually)