

FIT3162 Test Report

Team: 7

Team members:

- Andrew Duong
- Ming Shern, Siew
- Jordan Kapul

FIT3162 Test Report	1
Testing Approach	6
Test Types	6
End-to-end Testing	6
Integration Testing	6
Unit Testing	7
Useability Testing	7
Agile Testing	7
Code Reviews	8
Integration (API) Tests	9
Test Case 1.1 - Workspaces - Listing workspaces	9
Test Case 1.1.1 - Listing workspaces when none exist	9
Test Case 1.1.2 - Listing workspaces when workspaces have already been created	10
Test Case 1.2 - Workspaces - Creating workspaces	11
Test Case 1.2.1 - Creating workspace with poorly formatted data	11
Test Case 1.2.2 - Creating workspace with correct body data	12
Test Case 1.3 - Workspaces - Updating workspaces	13
Test Case 1.3.1 - Update workspace information correctly	13
Test Case 1.3.2 - Update workspace without providing any updates	14
Test Case 1.3.3 - Update workspace with poorly formatted data	15
Test Case 1.3.4 - Update workspace that does not belong to user	16
Test Case 1.3.5 - Update workspace that does not exist	17
Test Case 1.4 - Workspaces - Getting a workspace	18
Test Case 1.4.1 - Get a workspace correctly	18
Test Case 1.5 - Workspaces - Deleting a workspace	19
Test Case 1.5.1 - Deleting a workspace correctly	19
Test Case 2.1 - User Views - Creating a user view	20
Test Case 2.1.1 - Creating a user view correctly	20

Test Case 2.1.2 - Creating a user view with poorly formatted data	21
Test Case 2.1.3 - Creating a user view for a workspace that does not belong to the user	22
Test Case 2.1.4 - Creating a user view for a workspace that does not exist	23
Test Case 2.2 - User Views - Updating a user view	24
Test Case 2.2.1 - Updating a user view correctly	24
Test Case 2.2.2 - Updating a user view that does not belong to the user	25
Test Case 2.2.3 - Updating a user view with no body data	26
Test Case 2.2.4 - Updating a user view with an invalid combination of workspace and user view	27
Test Case 2.3 - User Views - Listing all user views	28
Test Case 2.3.1 - Listing all user view for a given workspace	28
Test Case 2.4 - User Views - Getting a user view	29
Test Case 2.4.1 - Get a given user view for a given workspace	29
Test Case 2.5 - User Views - Deleting a user view	30
Test Case 2.5.1 - Delete a given user view for a given workspace	30
Test Case 3.1 - User Visualisations - Creating a user visualisation	31
Test Case 3.1.1 - Create a user visualisation for a given workspace and given user view	31
Test Case 3.1.2 - Create a user visualisation for a workspace that does not belong to user	32
Test Case 3.1.3 - Create a user visualisation for an invalid combination of workspace and user view	33
Test Case 3.2 - User Visualisations - Updating a user visualisation	34
Test Case 3.2.1 - Update a user visualisation that does not belong to the user	34
Test Case 3.2.2 - Update a user visualisation correctly	35
Test Case 3.2.3 - Update a user visualisation that does not exist	36
Test Case 3.3 - User Visualisations - Listing all user visualisations	37
Test Case 3.3.1 - List all user visualisations	37
Test Case 3.4 - User Visualisations - Delete a user visualisation	38
Test Case 3.4.1 - Delete a user visualisation	38
Test Case 3.5 - User Visualisations - Fetch data for a user visualisation	39
Test Case 3.5.1 - Fetch measurement data for a user visualisation	39
Test Case 3.6 - Data Preparation	40
Test Case 3.6.1 - Enter one row into the database	40
Test Case 3.6.2 - Enter multiple rows into the database	40

Unit test cases	42
Performance test	57
Bushfire prediction model Efficiency	58
Useability Testing	59
User 1 - Billy (not real name)	59
Bio	59
Observations	59
Debriefing	60
User 2 - Mandy (not real name)	61
Bio	61
Observations	61
Debriefing	62
End-to-end Testing	63
Authentication	63
Login	63
Logout	63
Workspaces	64
Creating a workspace	64
Updating a workspace	64
Deleting a workspace	65
User Views	66
Creating User Views	66
Updating a user view	66
Switching between views	67
Deleting a view	67
Measurement Visualisations	68
Adding a measurement visualisation	68
Updating a measurement visualisation	69
Deleting a measurement visualisation	69

Limitations and Potential Improvements Based on Tests	70
Frontend Performance	70
Predictive model Performance	70
Customisation Options	71
Lack of Direction and Guidance	71
Sharing Work With Others	73
Limitations of the Testing Process	74
Manual vs Automated Testing	74
Continuous Integration	74
Tests Not Carried Out	74
Conclusions	76
Appendix	77

Testing Approach

Test Types

- The team has incorporated all major test types into the project.
 - This includes unit, integration and end-to-end testing.
 - Additionally, the team also carried out useability testing in order to gain real world feedback from potential users.

End-to-end Testing

- Since end-to-end testing allows us to conduct tests from an end user's perspective, we found that these tests were highly important to conduct as they give us the most confidence that the end user will have a working system.
- However, the team decided to conduct these tests manually since automated end-to-end tests are typically very brittle and susceptible to breaking.
 - The fragility of automated end-to-end tests is amplified by the fact that the project is in its very early stages, leading to frequent UI changes which ultimately breaks end-to-end tests.
- The tests were also carried in an exploratory fashion where team members would develop test cases on the fly as they gained a better understanding of the system.
- These tests were then documented in order to reproduce them at a later date.

Integration Testing

- Integration testing also provides the team with confidence that the system is working since these tests involve multiple subsystems working together.
- Integration testing was carried for the application's API by conducting manual API testing with postman.
- These tests allowed us to ascertain whether the API was correctly sanitizing and validating requests.
 - This allowed us to make sure that the API was secure and robust, improving the quality of the application.
 - These tests also allowed us to check the functional correctness of the API.

Unit Testing

- Unit testing was also carried out in order to ensure that individual components were operating correctly.
- Unit testing allowed us to test a wide range of different input data and verify that these components produced the correct results.
- Moreover, unit testing lends itself to easier automation which allowed us to develop automated unit tests which could be run automatically while developing the software.
 - This provided a very fast feedback loop, allowing the team to catch any potential errors early on.

Useability Testing

- Furthermore, the team conducted useability testing by finding potential users who likely represented the target market of the application.
- This allowed the team to gain real world feedback from actual users and pointed out various useability issues with the application.
- These testing sessions involved allowing users to freely use the application without any instruction while the team observed and recorded their behaviour.
- The users were then debriefed to record their reactions and opinion about the application.

Agile Testing

- Testing was not conducted as an afterthought, rather testing was incorporated as early as possible into the development process.
 - Since the team has adopted an agile approach for the project, this allowed the team to incorporate all aspects of the software development lifecycle into each sprint.
 - As such, the team conducted testing side by side with development in order to ensure the robustness and quality of the software being produced.

Code Reviews

- Furthermore, to ensure that the software meets a high quality standard, the team incorporated code reviews whenever code needed to be merged into the main branch.
 - This involved setting up a system where pushing directly to the main branch was prohibited.
 - Instead, team members would have to create feature branches where they could add new features and experimental code while leaving the main branch in a stable, deployable state.
 - In order to merge a team member's code into the main branch, they would have to request a review from at least one other team member.
 - This allowed us to make sure that at least two team members had an understanding of the code, reducing the risk of silos developing within the team.
 - Additionally, this acted as a quality control mechanism where feedback could be provided to team members in order to fix any issues.

Integration (API) Tests

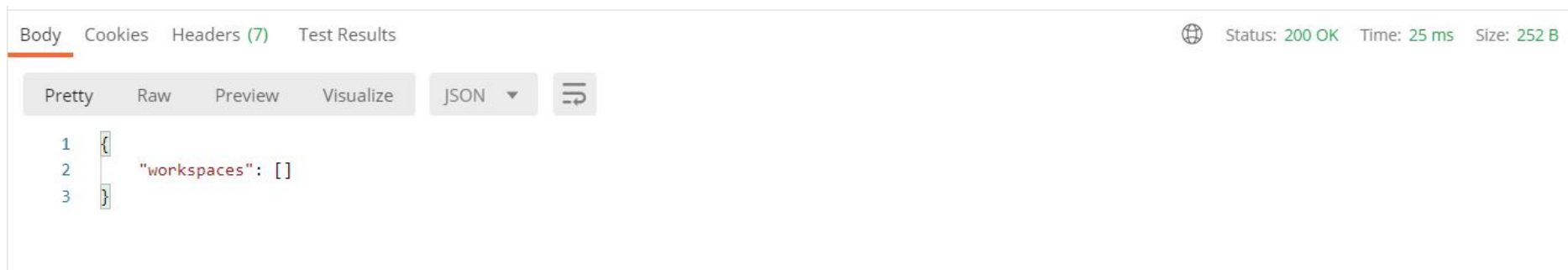
The following test cases have been designed to test the backend API by performing manual API testing using postman.

The path for each request contains the placeholder `{{domain}}` which signifies the domain name of the web application.

Test Case 1.1 - Workspaces - Listing workspaces

Test Case 1.1.1 - Listing workspaces when none exist

- Description: Requesting all workspaces for a user when the user has not created any yet.
- Prerequisites:
 - Make sure no workspaces exist for the user. Delete any if required.
- Request:
 - Path: `http://{{domain}}/api/workspaces`
 - Method: GET
- Expected result:
 - Status 200
 - Empty array being returned
- Actual result:
 - Empty array returned




Test Case 1.1.2 - Listing workspaces when workspaces have already been created

- Description: Requesting all workspaces for a user when the user has created several workspaces.
- Prerequisites:
 - Create a workspace for the user.
- Request:
 - Path: `http://{{domain}}/api/workspaces`
 - Method: GET
- Expected result:
 - Status 200
 - Array of workspaces belonging to this user.
- Actual result:
 - Matches expected result.

Body Cookies Headers (7) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2   "workspaces": [
3     {
4       "workspaceId": 116,
5       "ownerEmail": "jwkap2@student.monash.edu",
6       "workspaceName": "my cool workspace",
7       "workspaceColour": "#4caf50"
8     }
9   ]
10 }
```

 Status: 200 OK Time: 26 ms Size: 377 B

Test Case 1.2 - Workspaces - Creating workspaces

Test Case 1.2.1 - Creating workspace with poorly formatted data

- Description: Create a workspace where the request options are poorly formatted (e.g workspaceName and workspaceColour)
- Prerequisites:
 - None
- Request:
 - Path: `http://{{domain}}/api/workspaces`
 - Method: POST
 - Body:
 - workspaceName: `""`,
 - workspaceColour: `"not a colour"`
- Expected result:
 - Status 400
 - Array of errors:
 - Error saying that workspace name cannot be empty
 - Error saying that workspace colour must be a valid HEX colour
- Actual result:
 - Matches expected result.



The screenshot shows a REST client interface with the 'Body' tab selected. The response is a JSON object containing an 'errors' array. The status bar at the top right indicates 'Status: 400 Bad Request', 'Time: 21 ms', and 'Size: 470 B'. The JSON body is as follows:

```
1 {
2   "errors": [
3     {
4       "value": "",
5       "msg": "workspaceName cannot be empty",
6       "param": "workspaceName",
7       "location": "body"
8     },
9     {
10      "value": "not a colour",
11      "msg": "workspaceColour must be a valid HEX colour",
12      "param": "workspaceColour",
13      "location": "body"
14    }
15  ]
16 }
```

Test Case 1.2.2 - Creating workspace with correct body data

- Description: Create a workspace correctly
- Prerequisites:
 - None
- Request:
 - Path: `http://{{domain}}/api/workspaces`
 - Method: POST
 - Body:
 - `workspaceName: "my new workspace",`
 - `workspaceColour: "#FFF"`
- Expected result:
 - Status 201 created
 - New workspace information is returned.
- Actual result:
 - Matches expected result.

Body Cookies Headers (7) Test Results

Status: 201 Created Time: 154 ms Size: 361 B

Pretty

Raw

Preview

Visualize

JSON



```
1 {  
2   "workspaceId": 118,  
3   "ownerEmail": "jwkap2@student.monash.edu",  
4   "workspaceName": "my new workspace",  
5   "workspaceColour": "#FFF"  
6 }
```

Test Case 1.3 - Workspaces - Updating workspaces

Test Case 1.3.1 - Update workspace information correctly

- Description: Update a workspace correctly
- Prerequisites:
 - Create / get a user workspace and retrieve the workspace id.
- Request:
 - Path: `http://{{domain}}/api/workspaces/:workspaceId`
 - Method: PATCH
 - Body:
 - `workspaceName`: "this name has been changed",
 - `workspaceColour`: "#AAA"
- Expected result:
 - Status 200
 - New workspace information is returned.
- Actual result:
 - Matches expected result.

Body Cookies Headers (7) Test Results

Status: 200 OK Time: 29 ms Size: 366 B

Pretty Raw Preview Visualize JSON

```
1 {
2   "workspaceId": 118,
3   "ownerEmail": "jwkap2@student.monash.edu",
4   "workspaceName": "this name has been changed",
5   "workspaceColour": "#AAA"
6 }
```

Test Case 1.3.2 - Update workspace without providing any updates

- Description: Update a workspace without providing any update values in the body
- Prerequisites:
 - Create / get a user workspace and retrieve the workspace id.
- Request:
 - Path: `http://{{domain}}/api/workspaces/:workspaceId`
 - Method: PATCH
 - Body: `{}`
- Expected result:
 - Status 400
 - Error saying that one of workspace name or workspace colour must be provided
- Actual result:
 - Matches expected result.

Body Cookies Headers (7) Test Results

Status: 400 Bad Request Time: 20 ms Size: 492 B

Pretty Raw Preview Visualize JSON

```
1  {}
2  "errors": [
3    {
4      "msg": "One of workspaceName or workspaceColour must be provided",
5      "param": "_error",
6      "nestedErrors": [
7        {
8          "msg": "Invalid value",
9          "param": "workspaceName",
10         "location": "body"
11        },
12        {
13          "msg": "Invalid value",
14          "param": "workspaceColour",
15          "location": "body"
16        }
17      ]
18    }
19  ]
20  }
```

Test Case 1.3.3 - Update workspace with poorly formatted data

- Description: Update a workspace but provide incorrectly formatted update data
- Prerequisites:
 - Create / get a user workspace and retrieve the workspace id.
- Request:
 - Path: `http://{{domain}}/api/workspaces/:workspaceId`
 - Method: PATCH
 - Body:
 - `workspaceName: ""`,
 - `workspaceColour: "#AAA is this a colour?"`
- Expected result:
 - Status 400
 - Array of errors:
 - Workspace name cannot be empty
 - Workspace colour must be a valid hex colour
- Actual result:
 - Matches expected result.

Body Cookies Headers (7) Test Results

Status: 400 Bad Request Time: 158 ms Size: 480 B

Pretty Raw Preview Visualize JSON

```
1 {
2   "errors": [
3     {
4       "value": "",
5       "msg": "workspaceName cannot be empty",
6       "param": "workspaceName",
7       "location": "body"
8     },
9     {
10      "value": "#AAA is this a colour?",
11      "msg": "workspaceColour must be a valid HEX colour",
12      "param": "workspaceColour",
13      "location": "body"
14    }
15  ]
16 }
```

Test Case 1.3.4 - Update workspace that does not belong to user

- Description: Update a workspace that does not belong to the user
- Prerequisites:
 - Create / get a user workspace that does not belong to the test user and retrieve the workspace id.
- Request:
 - Path: `http://{{domain}}/api/workspaces/:workspaceId`
 - Method: PATCH
 - Body:
 - `workspaceName: "Not my workspace",`
 - `workspaceColour: "#AAA"`
- Expected result:
 - Status 400
 - Error stating that user does not own this workspace
- Actual result:
 - Matches expected result.



The screenshot shows a REST client interface with the following components:

- Body** (selected), Cookies, Headers (7), Test Results
- Status:** 400 Bad Request
- Time:** 162 ms
- Size:** 355 B
- Viewers:** Pretty, Raw, Preview, Visualize
- Format:** JSON
- JSON Body:**

```
1 {
2   "errors": [
3     {
4       "value": "21",
5       "msg": "User does not own this workspace",
6       "param": "workspaceId",
7       "location": "params"
8     }
9   ]
10 }
```


Test Case 1.3.5 - Update workspace that does not exist

- Description: Update a workspace that does not exist
- Prerequisites:
 - Find a workspace id that does not exist
- Request:
 - Path: `http://{{domain}}/api/workspaces/:workspaceId`
 - Method: PATCH
 - Body:
 - `workspaceName: "Not a workspace",`
 - `workspaceColour: "#AAA"`
- Expected result:
 - Status 400
 - Error stating that workspace does not exist
- Actual result:
 - Matches expected result.

Body Cookies Headers (7) Test Results

Status: 400 Bad Request Time: 18 ms Size: 346 B

```
1 {
2   "errors": [
3     {
4       "value": "1",
5       "msg": "Workspace does not exist",
6       "param": "workspaceId",
7       "location": "params"
8     }
9   ]
10 }
```

Test Case 1.4 - Workspaces - Getting a workspace

Test Case 1.4.1 - Get a workspace correctly

- Description: Retrieve a workspace correctly
- Prerequisites:
 - Create / get a user workspace and retrieve the workspace id.
- Request:
 - Path: `http://{{domain}}/api/workspaces/:workspaceId`
 - Method: GET
- Expected result:
 - Status 200
 - Workspace details should be returned.
- Actual result:
 - Matches expected result.

Body Cookies Headers (7) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2   "workspaceId": 118,
3   "ownerEmail": "jwkap2@student.monash.edu",
4   "workspaceName": "my workspace",
5   "workspaceColour": "#BBB"
6 }
```

⌐ Status: 200 OK Time: 19 ms Size: 352 B

Test Case 1.5 - Workspaces - Deleting a workspace

Test Case 1.5.1 - Deleting a workspace correctly

- Description: Delete a user's workspace correctly
- Prerequisites:
 - Create / get a user workspace and retrieve the workspace id.
- Request:
 - Path: `http://{{domain}}/api/workspaces/:workspaceId`
 - Method: DELETE
- Expected result:
 - Status 200
 - Workspace details should be returned.
- Actual result:
 - Matches expected result.

Body Cookies Headers (7) Test Results

Status: 200 OK Time: 14 ms Size: 352 B

Pretty Raw Preview Visualize JSON

```
1 {
2   "workspaceId": 118,
3   "ownerEmail": "jwkap2@student.monash.edu",
4   "workspaceName": "my workspace",
5   "workspaceColour": "#BBB"
6 }
```

Test Case 2.1 - User Views - Creating a user view

Test Case 2.1.1 - Creating a user view correctly

- Description: Creating a user view correctly
- Prerequisites:
 - Create / get a user workspace and retrieve the workspace id.
- Request:
 - Path: `http://{{domain}}/api/workspaces/:workspaceId/views`
 - Method: POST
 - Body:
 - `viewName: "My cool view"`
- Expected result:
 - Status 200
 - User view details should be returned.
- Actual result:
 - Matches expected result.

Body Cookies Headers (7) Test Results

Status: 201 Created Time: 18 ms Size: 315 B

Pretty Raw Preview Visualize

JSON



```
1 {  
2   "viewId": 63,  
3   "workspaceId": 116,  
4   "viewName": "My cool view",  
5   "gridLayout": null  
6 }
```

Test Case 2.1.2 - Creating a user view with poorly formatted data

- Description: Creating a user view with poorly formatted data (e.g viewName)
- Prerequisites:
 - Create / get a user workspace and retrieve the workspace id.
- Request:
 - Path: `http://{{domain}}/api/workspaces/:workspaceId/views`
 - Method: POST
 - Body:
 - `viewName: ""`
- Expected result:
 - Status 400
 - Error stating that view name cannot be empty.
- Actual result:
 - Matches expected result.

Body Cookies Headers (7) Test Results

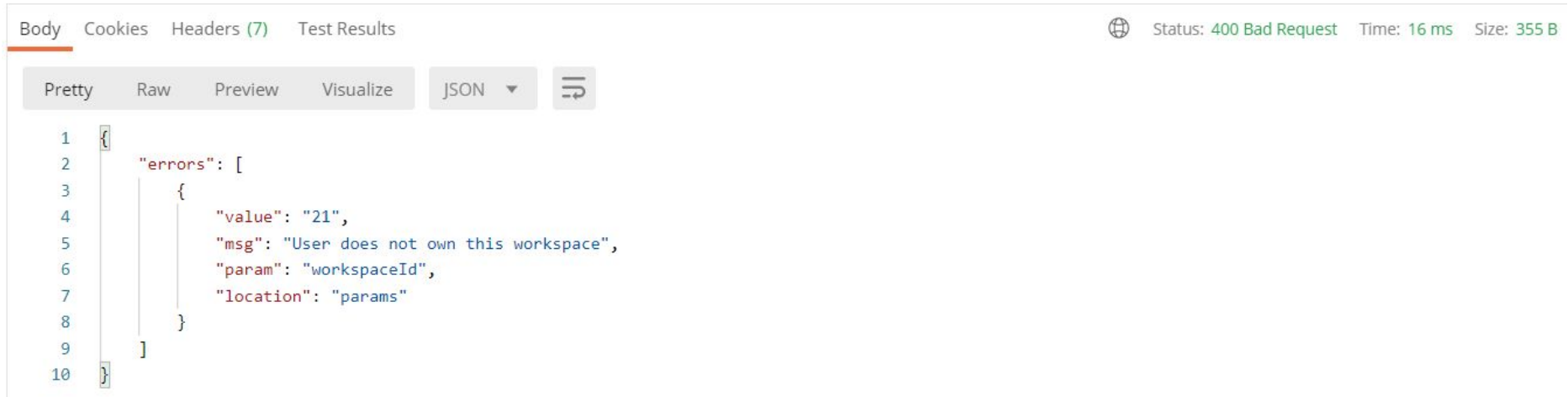
Pretty Raw Preview Visualize JSON

```
1 {
2   "errors": [
3     {
4       "value": "",
5       "msg": "viewName must not be empty",
6       "param": "viewName",
7       "location": "body"
8     }
9   ]
10 }
```

⌐ Status: 400 Bad Request Time: 20 ms Size: 341 B

Test Case 2.1.3 - Creating a user view for a workspace that does not belong to the user

- Description: Creating a user view for a workspace that does not belong to the user
- Prerequisites:
 - Create / get a workspace that does not belong to the user and retrieve the workspace id.
- Request:
 - Path: `http://{{domain}}/api/workspaces/:workspaceId/views`
 - Method: POST
 - Body:
 - `viewName: "I don't own this workspace"`
- Expected result:
 - Status 400
 - Error stating that the user does not own this workspace.
- Actual result:
 - Matches expected result.



Body Cookies Headers (7) Test Results

Status: 400 Bad Request Time: 16 ms Size: 355 B

Pretty Raw Preview Visualize JSON

```
1 {
2   "errors": [
3     {
4       "value": "21",
5       "msg": "User does not own this workspace",
6       "param": "workspaceId",
7       "location": "params"
8     }
9   ]
10 }
```

Test Case 2.1.4 - Creating a user view for a workspace that does not exist

- Description: Creating a user view for a workspace that does not exist
- Prerequisites:
 - Find a workspace id that does not exist
- Request:
 - Path: `http://{{domain}}/api/workspaces/:workspaceId/views`
 - Method: POST
 - Body:
 - `viewName: "This doesn't exist"`
- Expected result:
 - Status 400
 - Error stating that the workspace does not exist.
- Actual result:
 - Matches expected result.

Body Cookies Headers (7) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2   "errors": [
3     {
4       "value": "2111",
5       "msg": "Workspace does not exist",
6       "param": "workspaceId",
7       "location": "params"
8     }
9   ]
10 }
```

⌐ Status: 400 Bad Request Time: 14 ms Size: 349 B

Test Case 2.2 - User Views - Updating a user view

Test Case 2.2.1 - Updating a user view correctly

- Description: Creating a user view for a workspace that does not exist
- Prerequisites:
 - Create / get a workspace and retrieve the workspace id.
 - Create / get a user view that belongs to the above workspace and retrieve the user view id.
- Request:
 - Path: `http://{{domain}}/api/workspaces/:workspaceId/views/:viewId`
 - Method: PATCH
 - Body:
 - `viewName: "A new cool name for my view"`
- Expected result:
 - Status 200
 - The new view details should be returned.
- Actual result:
 - Matches expected result.



Test Case 2.2.2 - Updating a user view that does not belong to the user

- Description: Updating a user view that does not belong to the user
- Prerequisites:
 - Create / get a workspace that does not belong to the user and retrieve the workspace id.
 - Create / get a user view that belongs to the above workspace and retrieve the user view id.
- Request:
 - Path: `http://{{domain}}/api/workspaces/:workspaceId/views/:viewId`
 - Method: PATCH
 - Body:
 - viewName: "I don't own this!"
- Expected result:
 - Status 400
 - Error stating that user does not own the workspace.
- Actual result:
 - Matches expected result.

The screenshot shows a REST client interface with tabs for Body, Cookies, Headers (7), and Test Results. The Body tab is selected, displaying a JSON response in 'Pretty' format. The response is a 400 Bad Request with an error message. The status bar at the top right indicates 'Status: 400 Bad Request', 'Time: 20 ms', and 'Size: 355 B'.

```
{
  "errors": [
    {
      "value": "21",
      "msg": "User does not own this workspace",
      "param": "workspaceId",
      "location": "params"
    }
  ]
}
```

Test Case 2.2.3 - Updating a user view with no body data

- Description: Updating a user view with no body data
- Prerequisites:
 - Create / get a workspace that belongs to the user and retrieve the workspace id.
 - Create / get a user view that belongs to the above workspace and retrieve the user view id.
- Request:
 - Path: `http://{{domain}}/api/workspaces/:workspaceId/views/:viewId`
 - Method: PATCH
 - Body: null
- Expected result:
 - Status 400
 - Error stating that one of view name or grid layout must be provided
- Actual result:
 - Matches expected result.



```
Body Cookies Headers (7) Test Results
Status: 400 Bad Request Time: 18 ms Size: 472 B

Pretty Raw Preview Visualize JSON
1 {
2   "errors": [
3     {
4       "msg": "One of viewName or gridLayout must be provided",
5       "param": "_error",
6       "nestedErrors": [
7         {
8           "msg": "Invalid value",
9           "param": "viewName",
10          "location": "body"
11        },
12        {
13          "msg": "Invalid value",
14          "param": "gridLayout",
15          "location": "body"
16        }
17      ]
18    }
19  ]
20 }
```

Test Case 2.2.4 - Updating a user view with an invalid combination of workspace and user view

- Description: Updating a user view with a combination of workspace and user view that does not exist
- Prerequisites:
 - Create / get a workspace that belongs to the user and retrieve the workspace id.
 - Get a user view id that does not belong to the workspace above.
- Request:
 - Path: `http://{{domain}}/api/workspaces/:workspaceId/views/:viewId`
 - Method: PATCH
 - Body:
 - `viewName: "invalid combo"`
- Expected result:
 - Status 400
 - Error stating that this is an invalid combination of workspace and user view.
- Actual result:
 - Matches expected result.

Body Cookies Headers (7) Test Results

Status: 400 Bad Request Time: 25 ms Size: 389 B

Pretty Raw Preview Visualize JSON

```
1 {
2   "errors": [
3     {
4       "value": {
5         "workspaceId": "116",
6         "viewId": "630"
7       },
8       "msg": "Invalid combination of workspaceId and viewId",
9       "param": "",
10      "location": "params"
11    }
12  ]
13 }
```

Test Case 2.3 - User Views - Listing all user views

Test Case 2.3.1 - Listing all user view for a given workspace

- Description: Listing all user views for a given workspace
- Prerequisites:
 - Create / get a workspace that belongs to the user and retrieve the workspace id.
 - Create several user views that belong to the workspace above.
- Request:
 - Path: `http://{{domain}}/api/workspaces/:workspaceId/views`
 - Method: GET
 - Body: null
- Expected result:
 - Status 200
 - An array of user views belonging to the workspace
- Actual result:
 - Matches expected result.



The screenshot shows a REST client interface with the following components:

- Body** tab selected, showing a JSON response.
- Status:** 200 OK
- Time:** 36 ms
- Size:** 419 B
- JSON** tab selected, showing the response body.
- Response Body:**

```
{
  "userViews": [
    {
      "viewId": 63,
      "workspaceId": 116,
      "viewName": "A new cool name for my view",
      "gridLayout": null
    },
    {
      "workspaceId": 116,
      "viewName": "Another view!",
      "gridLayout": null,
      "viewId": 64
    }
  ]
}
```

Test Case 2.4 - User Views - Getting a user view

Test Case 2.4.1 - Get a given user view for a given workspace

- Description: Listing all user views for a given workspace
- Prerequisites:
 - Create / get a workspace that belongs to the user and retrieve the workspace id.
 - Create / get a user view that belongs to the above workspace and retrieve the user view id.
- Request:
 - Path: `http://{{domain}}/api/workspaces/:workspaceId/views/:viewId`
 - Method: GET
 - Body: null
- Expected result:
 - Status 200
 - The user view details should be returned.
- Actual result:
 - Matches expected result.

Body Cookies Headers (7) Test Results

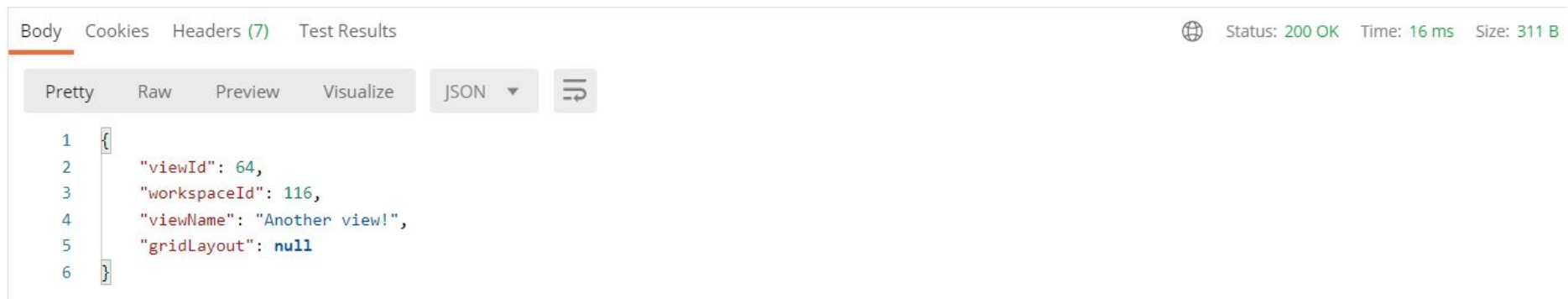
Status: 200 OK Time: 26 ms Size: 325 B

```
1 {
2   "workspaceId": 116,
3   "viewName": "A new cool name for my view",
4   "gridLayout": null,
5   "viewId": 63
6 }
```

Test Case 2.5 - User Views - Deleting a user view

Test Case 2.5.1 - Delete a given user view for a given workspace

- Description: Delete a user view for a given workspace and user view
- Prerequisites:
 - Create / get a workspace that belongs to the user and retrieve the workspace id.
 - Create / get a user view that belongs to the above workspace and retrieve the user view id.
- Request:
 - Path: `http://{{domain}}/api/workspaces/:workspaceId/views/:viewId`
 - Method: DELETE
 - Body: null
- Expected result:
 - Status 200
 - The user view details should be returned.
- Actual result:
 - Matches expected result.



The screenshot shows a REST client interface with tabs for Body, Cookies, Headers (7), and Test Results. The Body tab is selected. Below the tabs are buttons for Pretty, Raw, Preview, and Visualize, followed by a JSON dropdown and a refresh icon. The response body is a JSON object with the following structure:

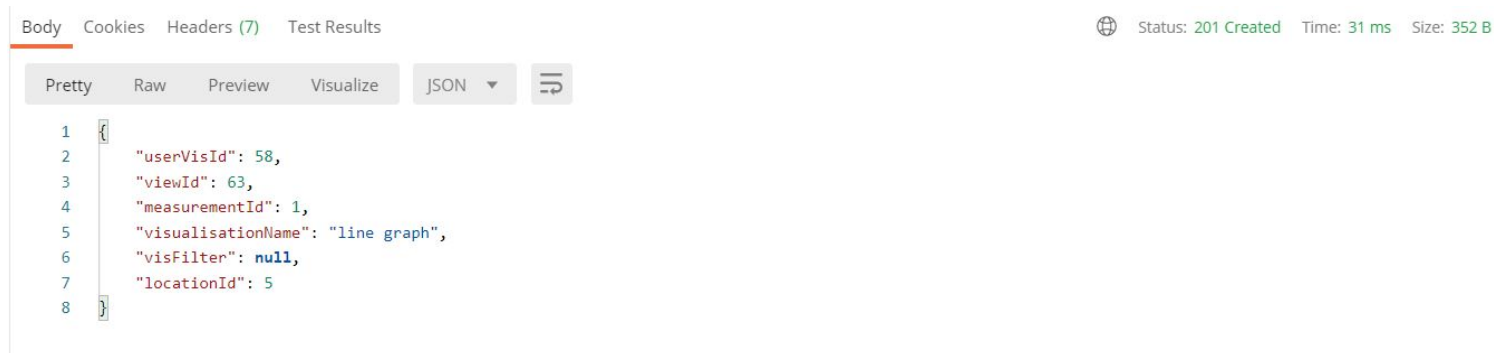
```
1 {
2   "viewId": 64,
3   "workspaceId": 116,
4   "viewName": "Another view!",
5   "gridLayout": null
6 }
```

At the top right of the interface, the status is 200 OK, the time is 16 ms, and the size is 311 B.

Test Case 3.1 - User Visualisations - Creating a user visualisation

Test Case 3.1.1 - Create a user visualisation for a given workspace and given user view

- Description: Create a user visualisation for a given workspace and given user view
- Prerequisites:
 - Create / get a workspace that belongs to the user and retrieve the workspace id.
 - Create / get a user view that belongs to the above workspace and retrieve the user view id.
 - Get a location id that exists
 - Get a measurement id that exists
 - Get a visualisation name that exists
- Request:
 - Path: `http://{{domain}}/api/workspaces/:workspaceId/views/:viewId/vis`
 - Method: POST
 - Body:
 - `locationId: <locationId>`
 - `measurementId: <measurementId>`
 - `visualisationName: <visualisationName>`
- Expected result:
 - Status 200
 - The user visualisation details should be returned.
- Actual result:
 - Matches expected result.



The screenshot shows a REST client interface with tabs for Body, Cookies, Headers (7), and Test Results. The Body tab is selected, displaying a JSON response in a 'Pretty' view. The response is a JSON object with the following fields: `userVisId` (58), `viewId` (63), `measurementId` (1), `visualisationName` (line graph), `visFilter` (null), and `locationId` (5). The status bar at the top right indicates 'Status: 201 Created', 'Time: 31 ms', and 'Size: 352 B'.

```
1 {
2   "userVisId": 58,
3   "viewId": 63,
4   "measurementId": 1,
5   "visualisationName": "line graph",
6   "visFilter": null,
7   "locationId": 5
8 }
```

Status: 201 Created Time: 31 ms Size: 352 B

Test Case 3.1.2 - Create a user visualisation for a workspace that does not belong to user

- Description: Create a user visualisation for a workspace that does not belong to user
- Prerequisites:
 - Create / get a workspace that does not belong to the user and retrieve the workspace id.
 - Create / get a user view that belongs to the above workspace and retrieve the user view id.
 - Get a location id that exists
 - Get a measurement id that exists
 - Get a visualisation name that exists
- Request:
 - Path: `http://{{domain}}/api/workspaces/:workspaceId/views/:viewId/vis`
 - Method: POST
 - Body:
 - `locationId: <locationId>`
 - `measurementId: <measurementId>`
 - `visualisationName: <visualisationName>`
- Expected result:
 - Status 400
 - Error stating that the user does not own this workspace
- Actual result:
 - Matches expected result.

Body Cookies Headers (7) Test Results

Status: 400 Bad Request Time: 160 ms Size: 355 B

Pretty Raw Preview Visualize JSON

```
1 {
2   "errors": [
3     {
4       "value": "21",
5       "msg": "User does not own this workspace",
6       "param": "workspaceId",
7       "location": "params"
8     }
9   ]
10 }
```


Test Case 3.1.3 - Create a user visualisation for an invalid combination of workspace and user view

- Description: Create a user visualisation for a combination of workspace and user view that does not exist
- Prerequisites:
 - Create / get a workspace and retrieve the workspace id.
 - Create / get a user view that does not belong to the above workspace and retrieve the user view id.
 - Get a location id that exists
 - Get a measurement id that exists
 - Get a visualisation name that exists
- Request:
 - Path: `http://{{domain}}/api/workspaces/:workspaceId/views/:viewId/vis`
 - Method: POST
 - Body:
 - `locationId: <locationId>`
 - `measurementId: <measurementId>`
 - `visualisationName: <visualisationName>`
- Expected result:
 - Status 400
 - Error stating that there is an invalid combination of workspace and user view
- Actual result:
 - Matches expected result.



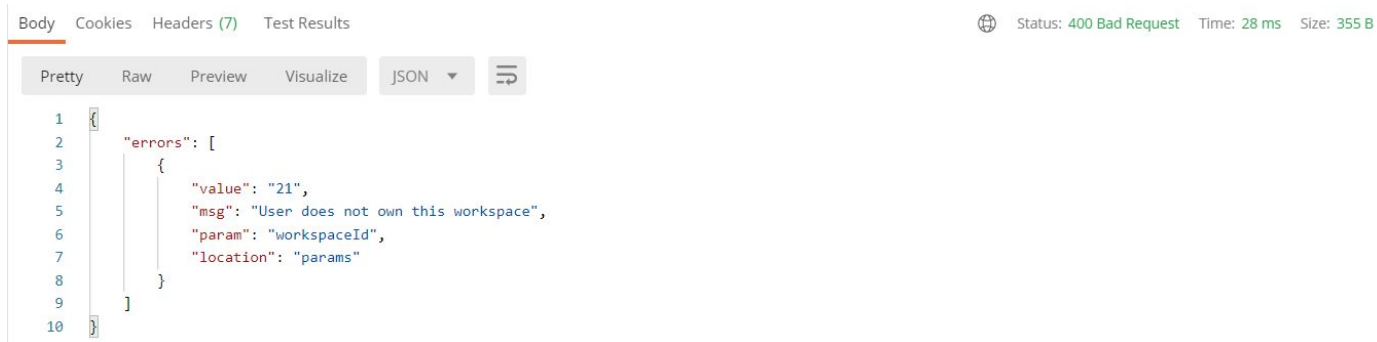
```
Body Cookies Headers (7) Test Results
Status: 400 Bad Request Time: 27 ms Size: 388 B

Pretty Raw Preview Visualize JSON
{
  "errors": [
    {
      "value": {
        "workspaceId": "116",
        "viewId": "64"
      },
      "msg": "Invalid combination of workspaceId and viewId",
      "param": "",
      "location": "params"
    }
  ]
}
```

Test Case 3.2 - User Visualisations - Updating a user visualisation

Test Case 3.2.1 - Update a user visualisation that does not belong to the user

- Description: Update a user visualisation that does not belong to the user
- Prerequisites:
 - Create / get a workspace that does not belong to the user and retrieve the workspace id.
 - Create / get a user view that belongs to the above workspace and retrieve the user view id.
 - Create / get a user visualisation that belongs to the above user view and retrieve the user view id.
 - Get a location id that exists
 - Get a measurement id that exists
 - Get a visualisation name that exists
- Request:
 - Path: `http://{{domain}}/api/workspaces/:workspaceId/views/:viewId/vis/:userVisId`
 - Method: PATCH
 - Body:
 - `locationId: <locationId>`
 - `measurementId: <measurementId>`
 - `visualisationName: <visualisationName>`
- Expected result:
 - Status 400
 - Error stating that the user does not own this workspace
- Actual result:
 - Matches expected result.



The screenshot shows a REST client interface with tabs for Body, Cookies, Headers (7), and Test Results. The Body tab is selected, displaying a JSON response in a code editor. The response is a 400 Bad Request with an error message. The status bar at the top right indicates 'Status: 400 Bad Request', 'Time: 28 ms', and 'Size: 355 B'.

```
1 {
2   "errors": [
3     {
4       "value": "21",
5       "msg": "User does not own this workspace",
6       "param": "workspaceId",
7       "location": "params"
8     }
9   ]
10 }
```

Test Case 3.2.2 - Update a user visualisation correctly

- Description: Update a user visualisation correctly
- Prerequisites:
 - Create / get a workspace that belongs to the user and retrieve the workspace id.
 - Create / get a user view that belongs to the above workspace and retrieve the user view id.
 - Create / get a user visualisation that belongs to the above user view and retrieve the user view id.
 - Get a location id that exists
 - Get a measurement id that exists
 - Get a visualisation name that exists
- Request:
 - Path: `http://{{domain}}/api/workspaces/:workspaceId/views/:viewId/vis/:userVisId`
 - Method: PATCH
 - Body:
 - `locationId: <locationId>`
 - `measurementId: <measurementId>`
 - `visualisationName: <visualisationName>`
- Expected result:
 - Status 200
 - New user visualisation details returned.
- Actual result:
 - Matches expected result.

Body Cookies Headers (7) Test Results

Pretty Raw Preview Visualize JSON ↕

```
1 {
2   "userVisId": 58,
3   "viewId": 63,
4   "measurementId": 2,
5   "visualisationName": "line graph",
6   "visFilter": null,
7   "locationId": 4
8 }
```

🌐 Status: 200 OK Time: 172 ms Size: 347 B

Test Case 3.2.3 - Update a user visualisation that does not exist

- Description: Update a user visualisation that does not exist
- Prerequisites:
 - Create / get a workspace that belongs to the user and retrieve the workspace id.
 - Create / get a user view that belongs to the above workspace and retrieve the user view id.
 - Get user visualisation that does not belong to the above user view and retrieve the user view id.
 - Get a location id that exists
 - Get a measurement id that exists
 - Get a visualisation name that exists
- Request:
 - Path: `http://{{domain}}/api/workspaces/:workspaceId/views/:viewId/vis/:userVisId`
 - Method: POST
 - Body:
 - `locationId: <locationId>`
 - `measurementId: <measurementId>`
 - `visualisationName: <visualisationName>`
- Expected result:
 - Status 400
 - Error stating invalid combination of workspace, user view and visualisation
- Actual result:
 - Matches expected result.



Body Cookies Headers (7) Test Results

Status: 400 Bad Request Time: 19 ms Size: 417 B

Pretty Raw Preview Visualize JSON

```
1 {
2   "errors": [
3     {
4       "value": {
5         "workspaceId": "116",
6         "viewId": "63",
7         "userVisId": "580"
8       },
9       "msg": "Invalid combination of workspaceId, viewId and userVisId",
10      "param": "",
11      "location": "params"
12    }
13  ]
14 }
```

Test Case 3.3 - User Visualisations - Listing all user visualisations

Test Case 3.3.1 - List all user visualisations

- Description: List all user visualisations
- Prerequisites:
 - Create / get a workspace that belongs to the user and retrieve the workspace id.
 - Create / get a user view that belongs to the above workspace and retrieve the user view id.
- Request:
 - Path: `http://{{domain}}/api/workspaces/:workspaceId/views/:viewId/vis`
 - Method: GET
- Expected result:
 - Status 200
 - Array of all user visualisation belonging to view and workspace
- Actual result:
 - Matches expected result.

Body Cookies Headers (7) Test Results

⌐ Status: 200 OK Time: 35 ms Size: 372 B

Pretty Raw Preview Visualize JSON ↕

```
1 {
2   "userVisualisations": [
3     {
4       "userVisId": 58,
5       "viewId": 63,
6       "measurementId": 2,
7       "visualisationName": "line graph",
8       "visFilter": null,
9       "locationId": 4
10    }
11  ]
12 }
```

Test Case 3.4 - User Visualisations - Delete a user visualisation

Test Case 3.4.1 - Delete a user visualisation

- Description: Delete a user visualisation
- Prerequisites:
 - Create / get a workspace that belongs to the user and retrieve the workspace id.
 - Create / get a user view that belongs to the above workspace and retrieve the user view id.
 - Get user visualisation that belongs to the above user view and retrieve the user view id.
- Request:
 - Path: `http://{{domain}}/api/workspaces/:workspaceId/views/:viewId/vis/:userVisId`
 - Method: DELETE
- Expected result:
 - Status 200
 - The user visualisation details should be returned.
- Actual result:
 - Matches expected result.

Body Cookies Headers (7) Test Results

Status: 200 OK Time: 172 ms Size: 347 B

Pretty Raw Preview Visualize JSON ↺

```
1 {
2   "userVisId": 58,
3   "viewId": 63,
4   "measurementId": 2,
5   "visualisationName": "line graph",
6   "visFilter": null,
7   "locationId": 4
8 }
```

Test Case 3.5 - User Visualisations - Fetch data for a user visualisation

Test Case 3.5.1 - Fetch measurement data for a user visualisation

- Description: Fetch measurement data for a user visualisation
- Prerequisites:
 - Create / get a workspace that belongs to the user and retrieve the workspace id.
 - Create / get a user view that belongs to the above workspace and retrieve the user view id.
 - Get user visualisation that belongs to the above user view and retrieve the user view id.
- Request:
 - Path: `http://{{domain}}/api/workspaces/:workspaceId/views/:viewId/vis/:userVisId/fetch`
 - Method: GET
- Expected result:
 - Status 200
 - An array of all measurement records should be returned.
- Actual result:
 - Matches expected result.



Body Cookies Headers (7) Test Results

Status: 200 OK Time: 432 ms Size: 488.25 KB

Pretty Raw Preview Visualize JSON

```
1 {
2   "records": [
3     {
4       "measurementRecordId": 143463,
5       "measurementId": 1,
6       "startTimestamp": "2014-12-31T13:00:00.000Z",
7       "endTimestamp": "2015-01-01T00:59:00.000Z",
8       "duration": "1 day",
9       "isPredicted": false,
10      "units": "°C",
11      "stationId": 124,
12      "minValue": 12,
13      "maxValue": 25.4,
14      "avgValue": null,
15      "accumulatedValue": null
16    },
17    {
```

Test Case 3.6 - Data Preparation

Test Case 3.6.1 - Enter one row into the database

- Description: Enter one row into the database
- Prerequisites:
 - The server is currently running
 - The database has been created
- Expected result:
 - The database contains the respective values and match the source
 - The console outputs `Done!`
- Actual result:
 - Matches expected result.
 -

Test Case 3.6.2 - Enter multiple rows into the database

- Description: Enter multiple rows into the database
- Prerequisites:
 - The server is currently running
 - The database has been created
- Expected result:
 - The database contains the respective values and match the source
 - The console outputs `Done!`
- Actual result:
 - Matches expected result.

Test Case 3.6.3 - Retrieve data when the source file does not exist

- Description: Retrieve data when the source file does not exist
- Prerequisites:
 - The server is currently running
 - The database has been created
 - The source file does not exist on the server
- Expected result:
 - The console outputs `File with path: \${filePath} does not exist!`
- Actual result:
 - Matches expected result.
 -

Test Case 3.6.4 - Input data that does not meet validation criteria

- Description: Input data that does not meet validation criteria
- Prerequisites:
 - The server is currently running
 - The database has been created
- Expected result:
 - The console outputs the respective SQL error
- Actual result:
 - Matches expected result.

Unit test cases

Test ID: 001

Function being tested: get_past_unix

Test Goal: Check if it can get past 3 days from 10/10/2020 and in melbourne timezone correctly.

Test inputs:

day =1, timezone=34200

day =2, timezone=34200

day =3, timezone=34200

Expected Output:

day =1, 1602230897

day =2, 1602144497

day =3, 1602058097

Actual Output:

day =1, 1602230897

day =2, 1602144497

day =3, 1602058097

Test ID: 002

Function being tested: get_forecast

Test Goal: Check if the forecasting function can compute the daily weather condition given payload correctly from openweather api and nth day.

Test inputs:

File will be a json which consists of payloads of daily forecasting.

File consisting of a format example(too big to copy over this report) of payloads output from web api. [Figure 5]

Expected Output:

Sight difference between actual and expected weather condition.

```
[{'Year': 2020, 'Month': 10, 'Day': 12, 'Max_Temp': 304.41, 'Humidity': 52, 'Wind_Speed': 8.964, 'Rainfall': 0},
{'Year': 2020, 'Month': 10, 'Day': 13, 'Max_Temp': 299.16, 'Humidity': 33, 'Wind_Speed': 19.584000000000003, 'Rainfall': 0},
{'Year': 2020, 'Month': 10, 'Day': 14, 'Max_Temp': 299.07, 'Humidity': 33, 'Wind_Speed': 8.496, 'Rainfall': 0},
{'Year': 2020, 'Month': 10, 'Day': 15, 'Max_Temp': 300.4, 'Humidity': 41, 'Wind_Speed': 11.304, 'Rainfall': 0},
{'Year': 2020, 'Month': 10, 'Day': 16, 'Max_Temp': 303.31, 'Humidity': 58, 'Wind_Speed': 7.452, 'Rainfall': 0},
{'Year': 2020, 'Month': 10, 'Day': 17, 'Max_Temp': 295.21, 'Humidity': 40, 'Wind_Speed': 14.076, 'Rainfall': 0}]
```

Actual Output:

Sight difference between actual and expected weather condition.

```
[{'Year': 2020, 'Month': 10, 'Day': 12, 'Max_Temp': 304.41, 'Humidity': 52, 'Wind_Speed': 8.96, 'Rainfall': 0},
{'Year': 2020, 'Month': 10, 'Day': 13, 'Max_Temp': 299.16, 'Humidity': 33, 'Wind_Speed': 19.58, 'Rainfall': 0},
{'Year': 2020, 'Month': 10, 'Day': 14, 'Max_Temp': 299.07, 'Humidity': 33, 'Wind_Speed': 8.5, 'Rainfall': 0},
{'Year': 2020, 'Month': 10, 'Day': 15, 'Max_Temp': 300.4, 'Humidity': 41, 'Wind_Speed': 11.30, 'Rainfall': 0},
{'Year': 2020, 'Month': 10, 'Day': 16, 'Max_Temp': 303.31, 'Humidity': 58, 'Wind_Speed': 7.45, 'Rainfall': 0},
{'Year': 2020, 'Month': 10, 'Day': 17, 'Max_Temp': 295.21, 'Humidity': 40, 'Wind_Speed': 14.08, 'Rainfall': 0}]
```

Test ID: 003

Function being tested: calculate_API

Test Goal: Check if API functions can calculate antecedent precipitation index values from weather conditions

Test inputs: File contains weather conditions which will cause the fire danger rating ranges from low to extreme.

Year	Month	Day	Max Temp	Rainfall - Total	Wind Speed - Mean	Humidity
2020	8	31	12.5	1.0	116.64	72
2020	8	27	20.7	0.0	90.72	72
2020	8	24	12.4	2.8	142.56	67
2020	8	22	11.1	4.4	64.80	59
2020	8	23	11.5	25.6	116.64	63

Expected Output:

Correctly compute value of API. Sight difference between actual and expected.

Year	Month	Day	Max Temp	Rainfall - Total	Wind Speed - Mean	Humidity	Expected API
2020	8	31	12.5	1.0	116.64	72	1.000000
2020	8	27	20.7	0.0	90.72	72	0.850000
2020	8	24	12.4	2.8	142.56	67	3.522500
2020	8	22	11.1	4.4	64.80	59	7.394125
2020	8	23	11.5	25.6	116.64	63	31.885006

Actual Output:

Correctly compute value of API. Sight difference between actual and expected.

Year	Month	Day	Max Temp	Rainfall - Total	Wind Speed - Mean	Humidity	API	FFDI	FFDI_Rate
2020	8	31	12.5	1.0	116.64	72	1.000000	2.486371	Low
2020	8	27	20.7	0.0	90.72	72	0.850000	1.523560	Low
2020	8	24	12.4	2.8	142.56	67	3.522500	18.714052	High
2020	8	22	11.1	4.4	64.80	59	7.394125	7.953595	Moderate
2020	8	23	11.5	25.6	116.64	63	31.885006	99.948241	Extreme

Test ID: 004

Function being tested: calculate_FFDI

Test Goal: Check if FFDI functions can calculate FFDI with antecedent precipitation index values from weather conditions

Test inputs: File contains weather conditions which will cause the fire danger rating ranges from low to extreme.

Year	Month	Day	Max Temp	Rainfall - Total	Wind Speed - Mean	Humidity
2020	8	31	12.5	1.0	116.64	72
2020	8	27	20.7	0.0	90.72	72
2020	8	24	12.4	2.8	142.56	67
2020	8	22	11.1	4.4	64.80	59
2020	8	23	11.5	25.6	116.64	63

Expected Output:

Correctly compute value of FFDI based antecedent precipitation index values. Slight difference between actual and expected.

Year	Month	Day	Max Temp	Rainfall - Total	Wind Speed - Mean	Humidity	FFDI
2020	8	31	12.5	1.0	116.64	72	2.49
2020	8	27	20.7	0.0	90.72	72	1.52
2020	8	24	12.4	2.8	142.56	67	18.71
2020	8	22	11.1	4.4	64.80	59	7.95
2020	8	23	11.5	25.6	116.64	63	99.95

Actual Output:

Correctly compute value of FFDI based antecedent precipitation index values. Slight difference between actual and expected.

Year	Month	Day	Max Temp	Rainfall - Total	Wind Speed - Mean	Humidity	API	FFDI	FFDI_Rate
2020	8	31	12.5	1.0	116.64	72	1.000000	2.486371	Low
2020	8	27	20.7	0.0	90.72	72	0.850000	1.523560	Low
2020	8	24	12.4	2.8	142.56	67	3.522500	18.714052	High
2020	8	22	11.1	4.4	64.80	59	7.394125	7.953595	Moderate
2020	8	23	11.5	25.6	116.64	63	31.885006	99.948241	Extreme

Test ID: 005

Function being tested: list_csv_file

Test Goal: Check if list_csv_file function can be used to find all csv files in a given directory.

Test inputs:

Directory contains 3 different csv files.

test_1.csv

test_2.csv

test_3.csv

Expected output: [test_1.csv, test_2.csv, test_3.csv]

Actual output: [test_1.csv, test_2.csv, test_3.csv]

Test ID: 006

Function being tested: collate_csv_file

Test Goal: Check if the collate_csv_file function can be used to find all csv files in a given directory.

Test inputs

Directory contains 3 different csv files.

test_1.csv, test_2.csv, test_3.csv

Each of them has the same format following:

'IDCKWCDE31										
Australian Government Bureau of Meteorology										
South Australia										
Daily Evapotranspiration for ADELAIDE AIRPORT South Australia for January 2009										
Issued at 04:34 GMT on Thursday 22 July 2010										
Â© Copyright Commonwealth of Australia 2009 Bureau of Meteorology (ABN 92 637 533 532)										
Please note Copyright Disclaimer and Privacy Notice < http://www.bom.gov.au/other/copyright.shtml >										
		Evapo-		Pan			Maximum	Minimum	Average	
		Transpiration	Rain	Evaporation	Maximum	Minimum	Relative	Relative	10m Wind	Solar
Station Name	Date	0000-2400	0900-0900	0900-0900	Temperature	Temperature	Humidity	Humidity	Speed	Radiation
		(mm)	(mm)	(mm)	(Â°C)	(Â°C)	(%)	(%)	(m/sec)	(MJ/sq m)
ADELAIDE AIRPORT	1/01/2009	7.1	0	7.4	22.6	17.3	67	38	7.41	31.97
ADELAIDE AIRPORT	2/01/2009	7.8	0	11.4	24.4	13.9	62	22	5.62	34.53
Totals:		257.9	0	330.2						

Expected output

Able to collate them into single files.

Each column will be Identified with a readable text.

Station	Date	Etrans	Rainfall	Epan	Max_Temp	min_Temp	Humidity	Min_hum	Wind_Spe	Rad
ADELAIDE AIRPORT	1/01/2009	7.1	0	7.4	22.6	17.3	67	38	7.41	31.97
ADELAIDE AIRPORT	2/01/2009	7.8	0	11.4	24.4	13.9	62	22	5.62	34.53
ADELAIDE AIRPORT	3/01/2009	7.4	0	9.2	26.3	10.8	75	19	4.09	34.5
ADELAIDE AIRPORT	4/01/2009	7.5	0	8	28.8	13.9	70	16	3.48	32.6
ADELAIDE AIRPORT	5/01/2009	6.9	0	10.2	24.6	12.1	68	30	4.01	34.5
ADELAIDE AIRPORT	6/01/2009	7.3	0	7.4	25.7	13.7	76	26	4.63	34.16

Actual output

Able to collate them into single files.

Each column will be Identified with a readable text.

Station	Date	Etrans	Rainfall	Epan	Max_Temp	min_Temp	Humidity	Min_hum	Wind_Spe	Rad
ADELAIDE AIRPORT	1/01/2009	7.1	0	7.4	22.6	17.3	67	38	7.41	31.97
ADELAIDE AIRPORT	2/01/2009	7.8	0	11.4	24.4	13.9	62	22	5.62	34.53
ADELAIDE AIRPORT	3/01/2009	7.4	0	9.2	26.3	10.8	75	19	4.09	34.5
ADELAIDE AIRPORT	4/01/2009	7.5	0	8	28.8	13.9	70	16	3.48	32.6
ADELAIDE AIRPORT	5/01/2009	6.9	0	10.2	24.6	12.1	68	30	4.01	34.5
ADELAIDE AIRPORT	6/01/2009	7.3	0	7.4	25.7	13.7	76	26	4.63	34.16

Test ID: 007

Function being tested: Imputation

Test Goal: Check if the imputation function can be used to replace all empty entry in any rows with mean value.

Test inputs: A csv file contains empty entry on random rows.

Station	Date	Etrans	Rainfall	Epan	Max_Temp	min_Temp	Humidity	Min_hum	Wind_Speed	Rad
ADELAIDE AIRPORT	1/01/2009	7.1	0	7.4	22.6	17.3	67	38	7.41	31.97
ADELAIDE AIRPORT	2/01/2009	7.8	0	11.4	24.4	13.9	62	22		34.53
ADELAIDE AIRPORT	3/01/2009	7.4	0	9.2	26.3	10.8	75		4.09	34.5
ADELAIDE AIRPORT	4/01/2009	7.5	0	8	28.8	13.9	70	16	3.48	32.6
ADELAIDE AIRPORT	5/01/2009	6.9	0	10.2	24.6	12.1	68	30	4.01	34.5
ADELAIDE AIRPORT	6/01/2009	7.3	0	7.4	25.7	13.7		26	4.63	34.16

Expected output:

Replace entry in empty rows with mean value

Station	Date	Etrans	Rainfall	Epan	Max_Temp	min_Temp	Humidity	Min_hum	Wind_Speed	Rad
ADELAIDE AIRPORT	1/01/2009	7.1	0	7.4	22.6	17.3	67	38	7.41	31.97
ADELAIDE AIRPORT	2/01/2009	7.8	0	11.4	24.4	13.9	62	22	4.72	34.53
ADELAIDE AIRPORT	3/01/2009	7.4	0	9.2	26.3	10.8	75	26	4.09	34.5
ADELAIDE AIRPORT	4/01/2009	7.5	0	8	28.8	13.9	70	16	3.48	32.6
ADELAIDE AIRPORT	5/01/2009	6.9	0	10.2	24.6	12.1	68	30	4.01	34.5
ADELAIDE AIRPORT	6/01/2009	7.3	0	7.4	25.7	13.7	68	26	4.63	34.16

Actual output:

Replace entry in empty rows with mean value

Station	Date	Etrans	Rainfall	Epan	Max_Temp	min_Temp	Humidity	Min_hum	Wind_Speed	Rad
ADELAIDE AIRPORT	1/01/2009	7.1	0	7.4	22.6	17.3	67	38	7.41	31.97
ADELAIDE AIRPORT	2/01/2009	7.8	0	11.4	24.4	13.9	62	22	4.724	34.53
ADELAIDE AIRPORT	3/01/2009	7.4	0	9.2	26.3	10.8	75	26.4	4.09	34.5
ADELAIDE AIRPORT	4/01/2009	7.5	0	8	28.8	13.9	70	16	3.48	32.6
ADELAIDE AIRPORT	5/01/2009	6.9	0	10.2	24.6	12.1	68	30	4.01	34.5
ADELAIDE AIRPORT	6/01/2009	7.3	0	7.4	25.7	13.7	68.4	26	4.63	34.16

Test ID: 008

Function being tested: extract_year

Test Goal: Check if extract_year function can be used to extract date to (year, month and day).

Test inputs: A csv file contains weather conditions of a weather station.

Station	Date	Etrans	Rainfall	Epan	Max_Temp	min_Temp	Humidity	Min_hum	Wind_Speed	Rad
ADELAIDE AIRPORT	1/01/2009	7.1	0	7.4	22.6	17.3	67	38	7.41	31.97
ADELAIDE AIRPORT	2/01/2009	7.8	0	11.4	24.4	13.9	62	22	4.724	34.53
ADELAIDE AIRPORT	3/01/2009	7.4	0	9.2	26.3	10.8	75	26.4	4.09	34.5
ADELAIDE AIRPORT	4/01/2009	7.5	0	8	28.8	13.9	70	16	3.48	32.6
ADELAIDE AIRPORT	5/01/2009	6.9	0	10.2	24.6	12.1	68	30	4.01	34.5
ADELAIDE AIRPORT	6/01/2009	7.3	0	7.4	25.7	13.7	68.4	26	4.63	34.16

Expected output

New column which will be known as year, month and day is created based on the Date column.

Station	Date	Etrans	Rainfall	Epan	Max_Temp	min_Temp	Humidity	Min_hum	Wind_Speed	Rad	Year	Month	Day
ADELAIDE AIRPORT	1/01/2009	7.1	0	7.4	22.6	17.3	67	38	7.41	31.97	2009	1	1
ADELAIDE AIRPORT	2/01/2009	7.8	0	11.4	24.4	13.9	62	22	4.724	34.53	2009	1	2
ADELAIDE AIRPORT	3/01/2009	7.4	0	9.2	26.3	10.8	75	26.4	4.09	34.5	2009	1	3
ADELAIDE AIRPORT	4/01/2009	7.5	0	8	28.8	13.9	70	16	3.48	32.6	2009	1	4
ADELAIDE AIRPORT	5/01/2009	6.9	0	10.2	24.6	12.1	68	30	4.01	34.5	2009	1	5
ADELAIDE AIRPORT	6/01/2009	7.3	0	7.4	25.7	13.7	68.4	26	4.63	34.16	2009	1	6

Actual output

Year, month and day have been created in columns which correspond to the date of given rows.

Station	Date	Etrans	Rainfall	Epan	Max_Temp	min_Temp	Humidity	Min_hum	Wind_Speed	Rad	Year	Month	Day
ADELAIDE AIRPORT	1/01/2009	7.1	0	7.4	22.6	17.3	67	38	7.41	31.97	2009	1	1
ADELAIDE AIRPORT	2/01/2009	7.8	0	11.4	24.4	13.9	62	22	4.724	34.53	2009	1	2
ADELAIDE AIRPORT	3/01/2009	7.4	0	9.2	26.3	10.8	75	26.4	4.09	34.5	2009	1	3
ADELAIDE AIRPORT	4/01/2009	7.5	0	8	28.8	13.9	70	16	3.48	32.6	2009	1	4
ADELAIDE AIRPORT	5/01/2009	6.9	0	10.2	24.6	12.1	68	30	4.01	34.5	2009	1	5
ADELAIDE AIRPORT	6/01/2009	7.3	0	7.4	25.7	13.7	68.4	26	4.63	34.16	2009	1	6

Test ID: 009

Function being tested: transform_windspeed

Test Goal: Check if transform_wind speed function can be used to change m/sec to km/hr with math formula.

Test Input: A csv file contains weather conditions of a weather station.

Station	Date	Etrans	Rainfall	Epan	Max_Temp	min_Temp	Humidity	Min_hum	Wind_Speed	Rad
ADELAIDE AIRPORT	1/01/2009	7.1	0	7.4	22.6	17.3	67	38	7.41	31.97
ADELAIDE AIRPORT	2/01/2009	7.8	0	11.4	24.4	13.9	62	22	4.724	34.53
ADELAIDE AIRPORT	3/01/2009	7.4	0	9.2	26.3	10.8	75	26.4	4.09	34.5
ADELAIDE AIRPORT	4/01/2009	7.5	0	8	28.8	13.9	70	16	3.48	32.6
ADELAIDE AIRPORT	5/01/2009	6.9	0	10.2	24.6	12.1	68	30	4.01	34.5
ADELAIDE AIRPORT	6/01/2009	7.3	0	7.4	25.7	13.7	68.4	26	4.63	34.16

Expected output

New wind speed = $3.6 * \text{original wind speed}$

Station	Date	Etrans	Rainfall	Epan	Max_Temp	min_Temp	Humidity	Min_hum	Wind_Speed	Rad
ADELAIDE AIRPORT	1/01/2009	7.1	0	7.4	22.6	17.3	67	38	26.68	31.97
ADELAIDE AIRPORT	2/01/2009	7.8	0	11.4	24.4	13.9	62	22	17.01	34.53
ADELAIDE AIRPORT	3/01/2009	7.4	0	9.2	26.3	10.8	75	26	14.72	34.5
ADELAIDE AIRPORT	4/01/2009	7.5	0	8	28.8	13.9	70	16	12.53	32.6
ADELAIDE AIRPORT	5/01/2009	6.9	0	10.2	24.6	12.1	68	30	14.44	34.5
ADELAIDE AIRPORT	6/01/2009	7.3	0	7.4	25.7	13.7	68	26	16.67	34.16

Actual output

New wind speed = 3.6 * original wind speed

Station	Date	Etrans	Rainfall	Epan	Max_Temp	min_Temp	Humidity	Min_hum	Wind_Speed	Rad
ADELAIDE AIRPORT	1/01/2009	7.1	0	7.4	22.6	17.3	67	38	26.676	31.97
ADELAIDE AIRPORT	2/01/2009	7.8	0	11.4	24.4	13.9	62	22	17.0064	34.53
ADELAIDE AIRPORT	3/01/2009	7.4	0	9.2	26.3	10.8	75	26.4	14.724	34.5
ADELAIDE AIRPORT	4/01/2009	7.5	0	8	28.8	13.9	70	16	12.528	32.6
ADELAIDE AIRPORT	5/01/2009	6.9	0	10.2	24.6	12.1	68	30	14.436	34.5
ADELAIDE AIRPORT	6/01/2009	7.3	0	7.4	25.7	13.7	68.4	26	16.668	34.16

Test ID: 010

Function being tested: feature_selection

Test Goal: Check if feature_selection function can be used to filter all unused columns.

Test Input: A csv file contains weather conditions of a weather station and Feature consist of { station, date, rainfall, max temperature, humidity and wind speed }

Station	Date	Etrans	Rainfall	Epan	Max_Temp	min_Temp	Humidity	Min_hum	Wind_Speed	Rad
ADELAIDE AIRPORT	1/01/2009	7.1	0	7.4	22.6	17.3	67	38	7.41	31.97
ADELAIDE AIRPORT	2/01/2009	7.8	0	11.4	24.4	13.9	62	22	4.724	34.53
ADELAIDE AIRPORT	3/01/2009	7.4	0	9.2	26.3	10.8	75	26.4	4.09	34.5
ADELAIDE AIRPORT	4/01/2009	7.5	0	8	28.8	13.9	70	16	3.48	32.6
ADELAIDE AIRPORT	5/01/2009	6.9	0	10.2	24.6	12.1	68	30	4.01	34.5
ADELAIDE AIRPORT	6/01/2009	7.3	0	7.4	25.7	13.7	68.4	26	4.63	34.16

Expected output

Columns which are filtered will not be visible to the user.

Station	Date	Rainfall	Max_Temp	Humidity	Wind_Speed
ADELAIDE AIRPORT	1/01/2009	0	22.6	67	7.41
ADELAIDE AIRPORT	2/01/2009	0	24.4	62	4.724
ADELAIDE AIRPORT	3/01/2009	0	26.3	75	4.09
ADELAIDE AIRPORT	4/01/2009	0	28.8	70	3.48
ADELAIDE AIRPORT	5/01/2009	0	24.6	68	4.01
ADELAIDE AIRPORT	6/01/2009	0	25.7	68.4	4.63

Actual output

Columns which are filtered will not be visible to the user.

Station	Date	Max_Temp	Humidity	Wind_Speed	Rainfall
ADELAIDE AIRPORT	1/01/2009	22.6	67	7.41	0
ADELAIDE AIRPORT	2/01/2009	24.4	62	4.724	0
ADELAIDE AIRPORT	3/01/2009	26.3	75	4.09	0
ADELAIDE AIRPORT	4/01/2009	28.8	70	3.48	0
ADELAIDE AIRPORT	5/01/2009	24.6	68	4.01	0
ADELAIDE AIRPORT	6/01/2009	25.7	68.4	4.63	0

Screenshot of successful tests

```
-----  
Ran 4 tests in 0.015s  
OK
```

Predictive modelling unit test

```
==> Testing R file using 'testthat'  
  
v | OK F W S | Context  
v | 12      | UnitTest_Preprocessing
```

```
== Results =====  
Duration: 0.2 s
```

```
OK:      12  
Failed:  0  
warnings: 0  
Skipped: 0
```

```
Keep up the good work.
```

Preprocessing unit test

Performance test

The performance of the predictive modelling was measured by running two different module which are predictive model and forecast model with 3 different size where each size specified is sum of the size of random different number of dataset to be forecasted and predicted(for example, size of ADELAIDE AIRPORT.csv + size of AIREYS INLET + ... = 15 mb) and comparing the time taken to compute by each respective module. These predictions were obtained on a local computer with 8GB of Ram, an i7-9750H CPU and a NVIDIA Geforce RTX 2060 graphics card.

Size(mb)	Time(sec)
15	216.55
60	311.4618
120	595.437

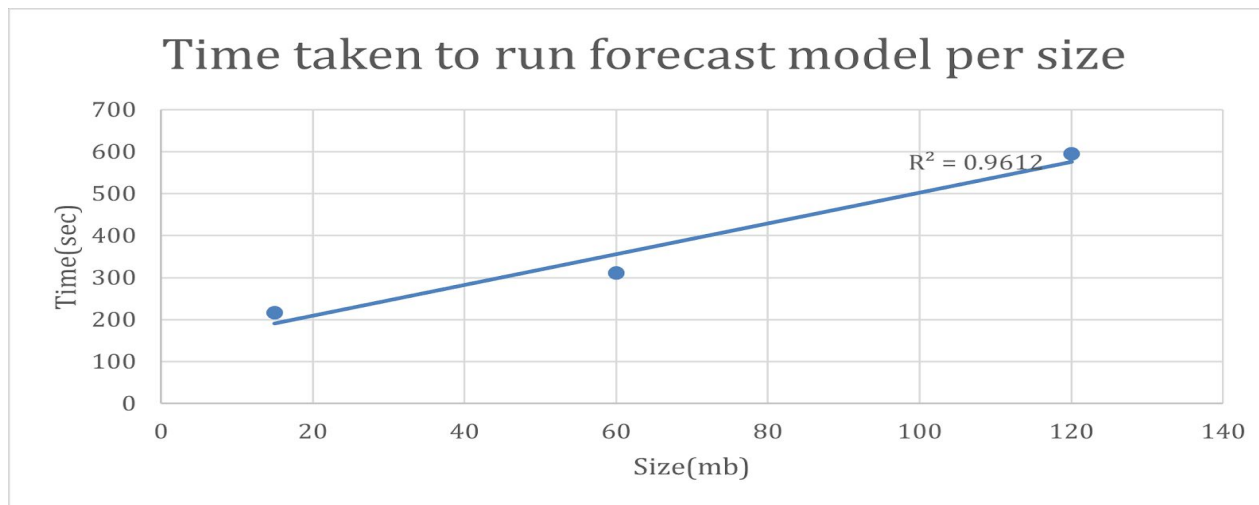


Figure 2: Weather forecasting model Efficiency

The performance for running forecasting model with accordance to time taken to run it and size of dataset it will compute, shows that as size of datasets increase so does the time taken to compute forecasting model. The correlation of determination for time taken to run forecast model per size shows 96.12% of time taken to run forecast model can be explained by size of the dataset it will need to compute. Thus, time taken to run the forecast model has a strong correlation with size of the dataset.

Size(mb)	Time(sec)
15	347.75
60	643.611
120	1284.832

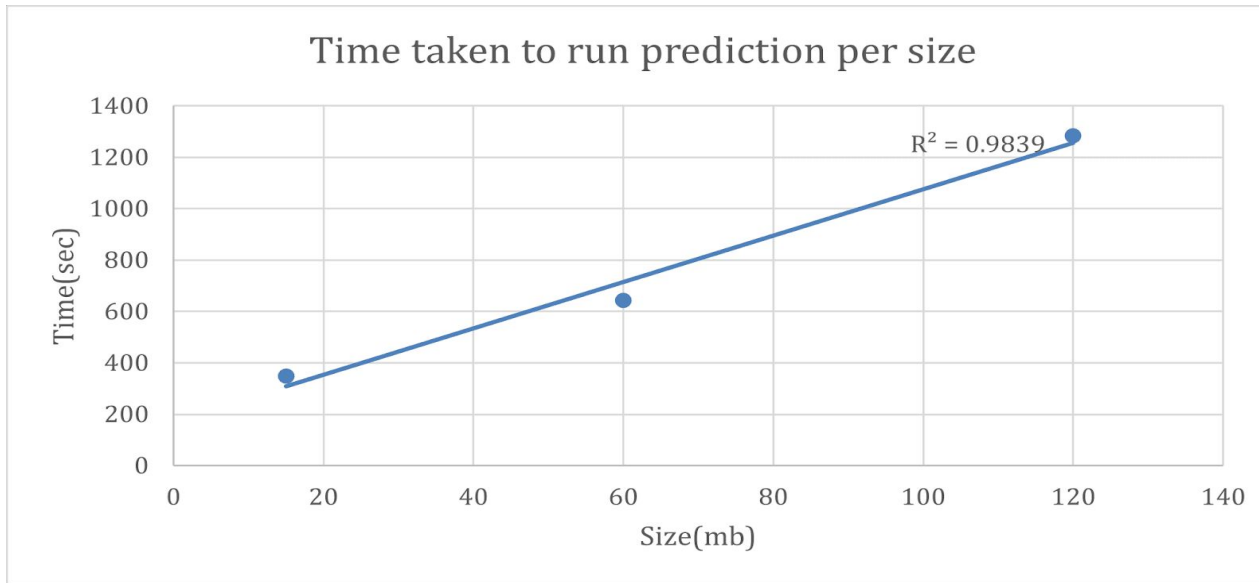


Figure 3: Bushfire prediction model Efficiency

The performance for running prediction model with accordance to time taken to run it and size of dataset it will compute, shows that as size of datasets increase so does the time taken to compute prediction model. The correlation of determination for time taken to run prediction model per size shows 98.39% of time taken to run prediction model can be explained by size of the dataset it will need to compute. Thus, time taken to run prediction models has a strong correlation with size of the dataset.

Usability Testing

- Useability testing was carried out by selecting a range of users who likely represent the application's target market.
- These users were then allowed to freely use the application without any instruction from the team while the team observed and recorded both their behaviour and reactions.
- Finally, the team conducted a debriefing with the potential users and conducted a short open discussion.

User 1 - Billy (not real name)

Bio

- Age: 15
- Occupation: student
- Very tech savvy
- Hasn't really ever thought about bushfire research

Observations

- Billy immediately understood the structure and layout of the page as it aligns with his existing mental model of websites.
 - He was very comfortable navigating the site, collapsing the workspaces drawer and logging in.
- Billy stated that he was very pleased to see a google login option as this is his preferred login method.
- Without any instruction, Billy was able to understand what the different icon buttons represented as he is very familiar with these icons from previous web applications. For example, the pencil icon signifies editing and the three vertical dots are used to open up an options menu.
- Adding measurements came very naturally to Billy, he was able to easily follow the on screen instructions and add measurements.
 - However, he found that while adding measurements, the app would lag when the interactive location map was loading the local government areas.
 - He stated that the screen was frozen and that he couldn't do anything.
 - Since Billy is tech savvy, he was very efficient and quick when selecting options which made the lag even more evident.
- Billy stated that he was very pleased with the ability to both scroll through the dropdown list and search for options in a clean and easy way.

- However, when adding locations via the interactive map, panning has been disabled which makes it difficult to find the smaller local government areas on the map.
- Billy found that switching between view tabs was very natural and easy to understand.
- However, Billy acknowledged that he would have preferred some way of reorganising the order of visualisation within the view as currently he would have to delete visualisations and then add them back to reorder them.

Debriefing

- Billy was overall happy with the application and felt that it was a very modern and clean user interface.
- However, he stated that there were significant performance issues with the application, causing it to lag when loading measurements or locations on the interactive map.
- Additionally, he felt that switching between views was slow since the app would freeze up when loading the new view.
- He acknowledged that he probably wouldn't use the app outside of an academic environment but he would be very happy to use it for a school project.
 - However, he found that he didn't really have a solid understanding of bushfire research and so couldn't fully appreciate the measurements offered.
- Billy also brought up the issue of not being able to filter measurements to his liking. He would like to have been able to exclude certain data points by having a filter option for each measurement.
- Finally, he pointed out that being able to move measurements around would be very handy.

User 2 - Mandy (not real name)

Bio

- Age: 10
- Occupation: student
- Moderately tech savvy
- Has been keen to look into bushfire research

Observations

- Mandy was immediately comfortable with the layout and structure of the application as it resembles many applications she uses currently.
 - However, she was slightly confused when she clicked on the login button and it redirected her to a third party authentication system.
 - She found that it was scary and that she didn't know what was going on, she was afraid it was a scam.
 - She did not notice the google login but instead opted for a username / password sign up.
- Mandy was particularly excited to see that she could change the colour of her workspace and that it was reflected in the view, she felt that she owned her view and could customise it however she wanted.
- It took a while for Mandy to understand the difference between a view and a workspace.
 - She was confused as to why we needed a view and how to add one.
 - Eventually, she saw the plus icon and decided to click it which allowed her to add a view.
- She then immediately understood how to add a measurement
 - However, she didn't know what the different types of graphs looked like and couldn't decide which one to choose.
 - Moreover, she had no idea which location to pick as they all looked the same to her.
- After creating a measurement, she was upset that she could not change the colours of the visualisations themselves.
- Additionally, she wanted to print out her view to show to everyone but couldn't find a print button.

Debriefing

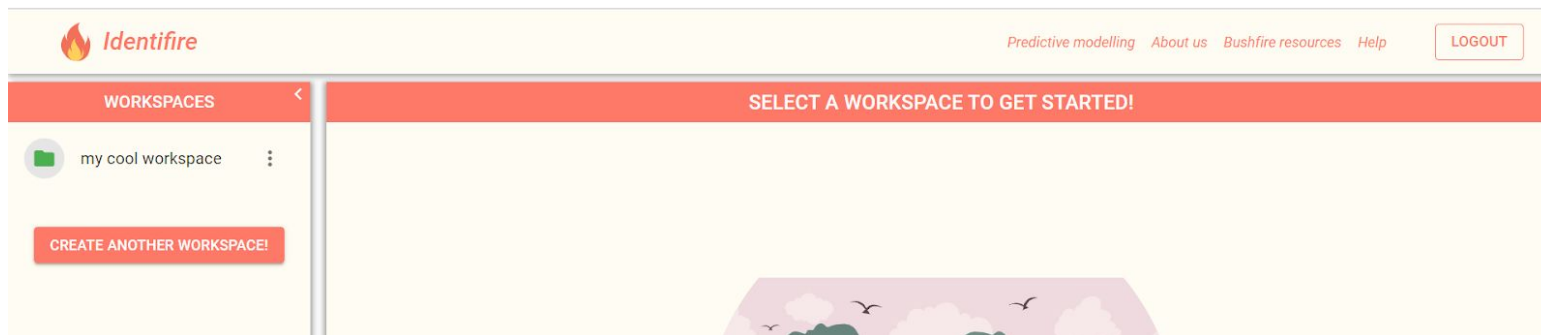
- Mandy really liked the customisation options available and wanted to see even more implemented.
- However, she found the experience confusing and would have greatly benefited from some visual cues and hand holding in order to guide her in the right direction.
- Mandy said that she would love to use the app for a school project and would like to share her visualisations with all her friends and possibly even work on them together.

End-to-end Testing

Authentication

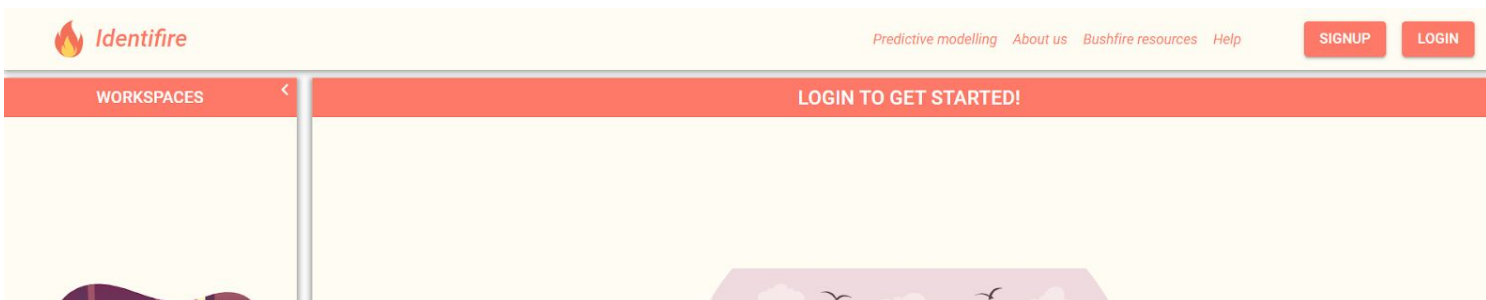
Login

1. Click the login button in the application header
2. Login with preferred method
3. Verify that you are logged in by checking that the signup and login buttons have been replaced by a logout button
 - a. Additionally, if you have any workspaces, they should now be visible.



Logout

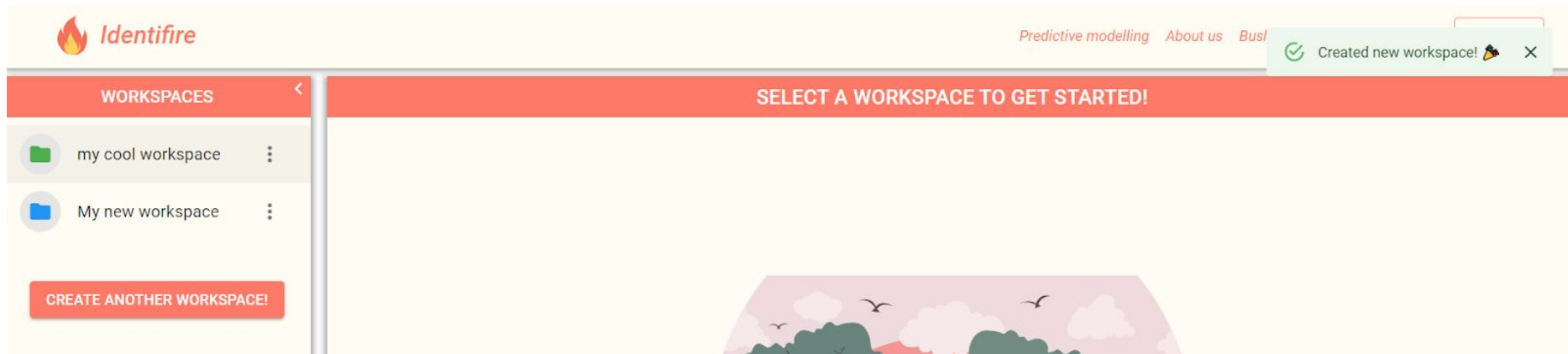
1. Make sure you are logged in.
2. Click the logout button.
3. Verify that you are logged out by checking that the logout button has been replaced by the signup and login buttons.
 - a. Additionally, any workspace or view you had open should no longer be visible.



Workspaces

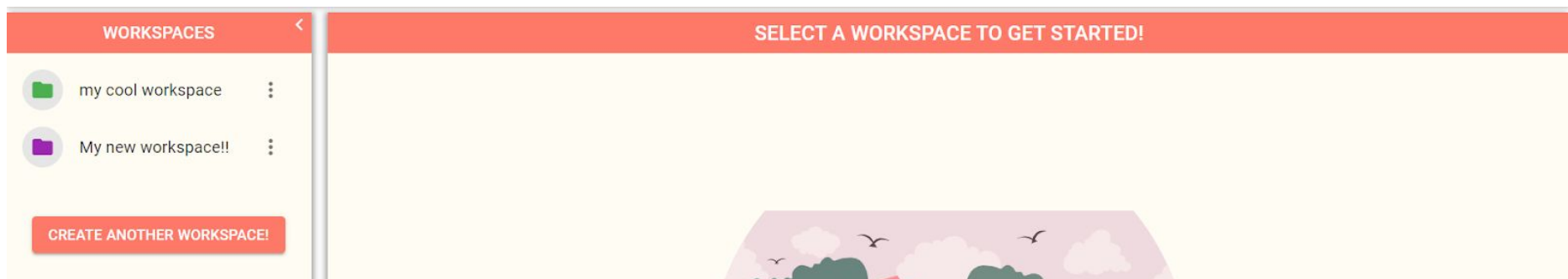
Creating a workspace

1. Click on the create another workspace button to add a new workspace.
2. Enter name and colour for your new workspace.
3. Verify that the workspace has been added to the workspaces drawer and has the correct colour and name.



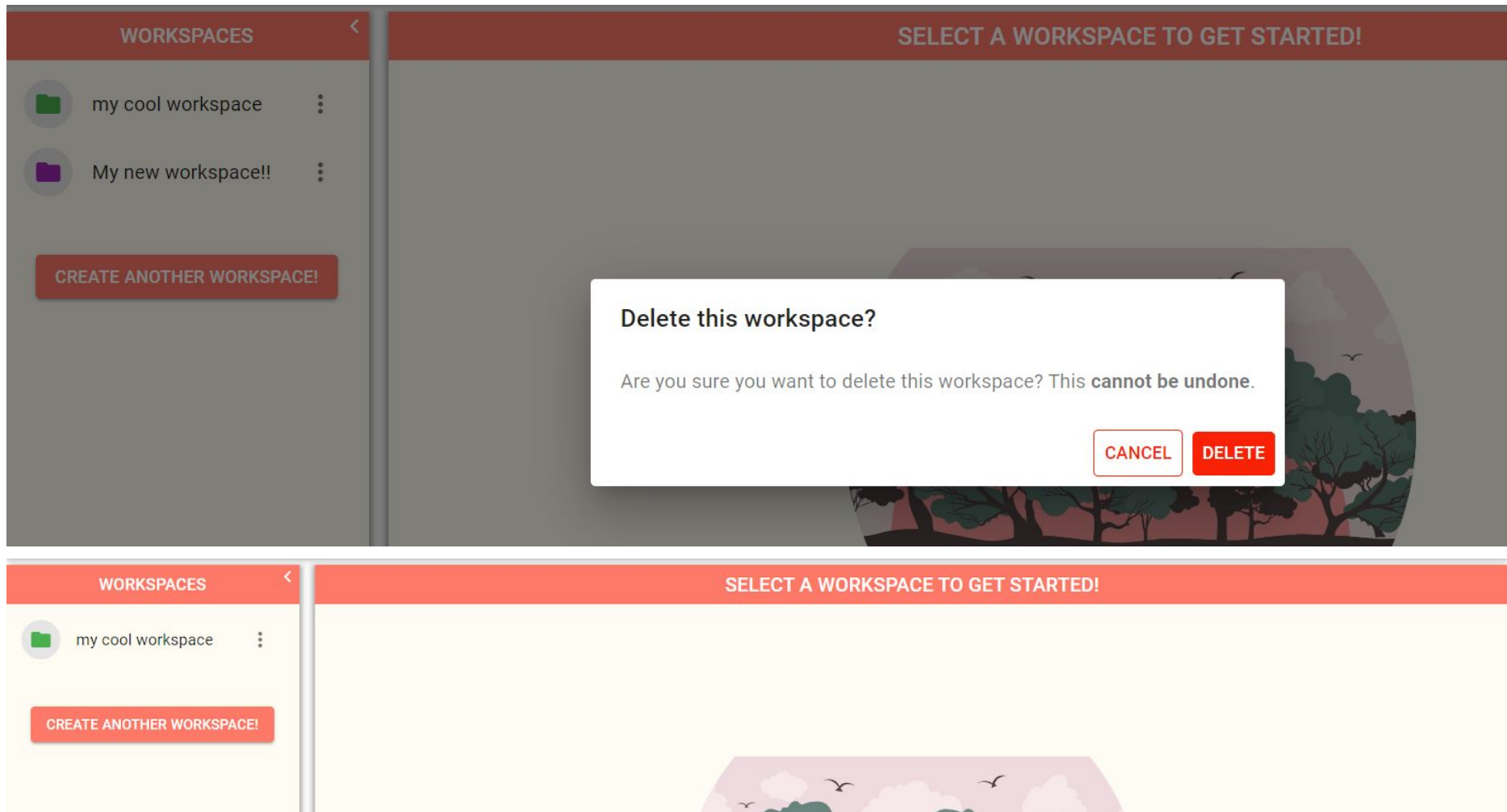
Updating a workspace

1. Make sure you have a workspace created.
2. Click the three vertical dots next to the workspace and click edit workspace.
3. Update the workspace name and colour.
4. Verify that these changes have been reflected in the workspaces drawer.



Deleting a workspace

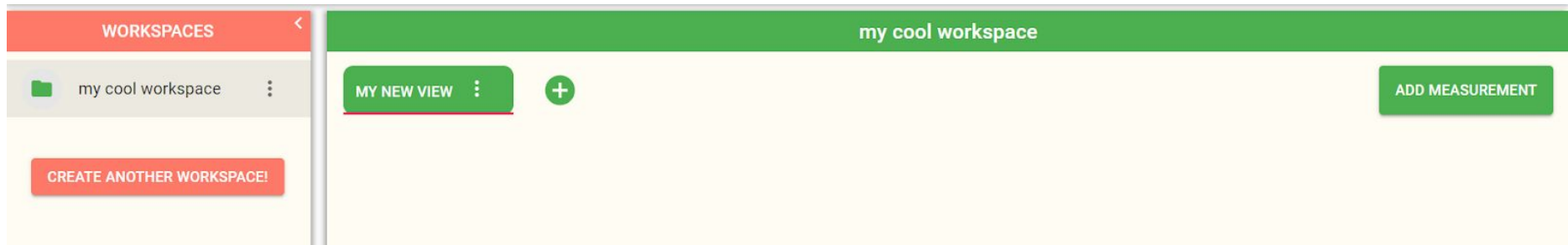
1. Make sure you have a workspace created.
2. Click the three vertical dots next to the workspace name.
3. Select delete workspace.
4. Confirm the deletion.
5. Verify that the workspace has been removed from the workspaces drawer.



User Views

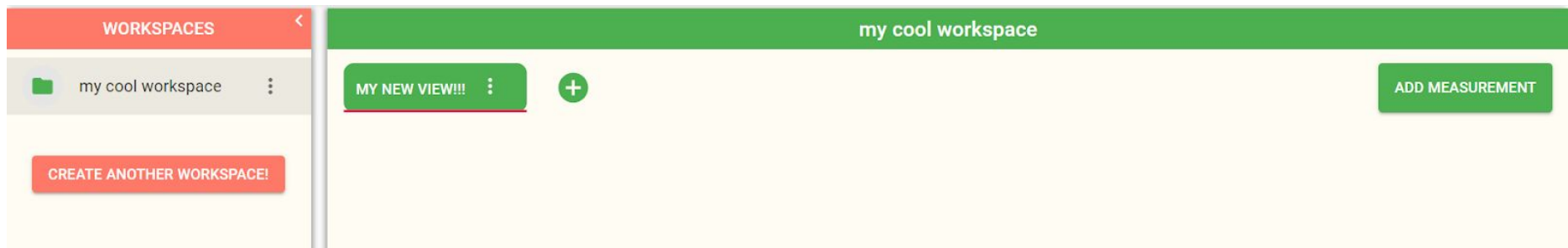
Creating User Views

1. Make sure you have a workspace created.
2. Select the workspace in the workspaces drawer.
3. Click the plus symbol to add a new view.
4. Verify that the view has been added to the tab bar.



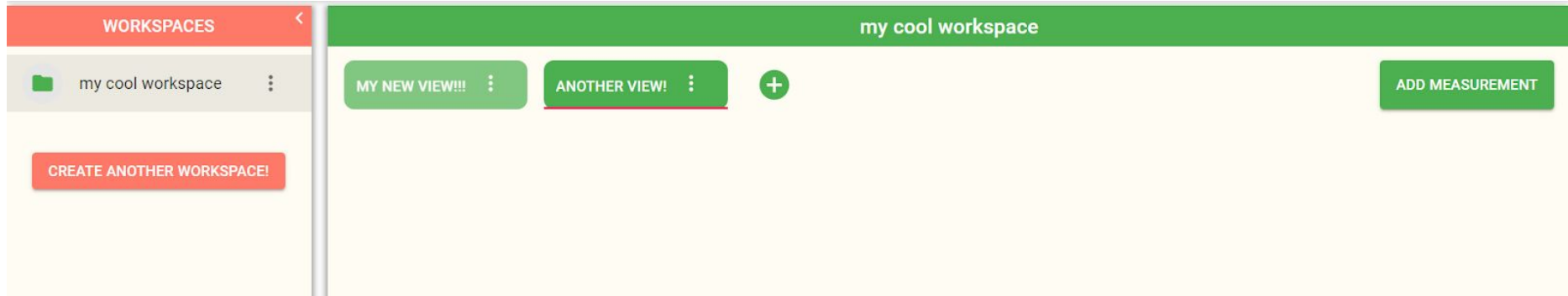
Updating a user view

1. Make sure you have a view created
2. Click the three vertical dots next to the view name and select edit view.
3. Change the name of the view and select update.
4. Verify that the name of the view has now changed.



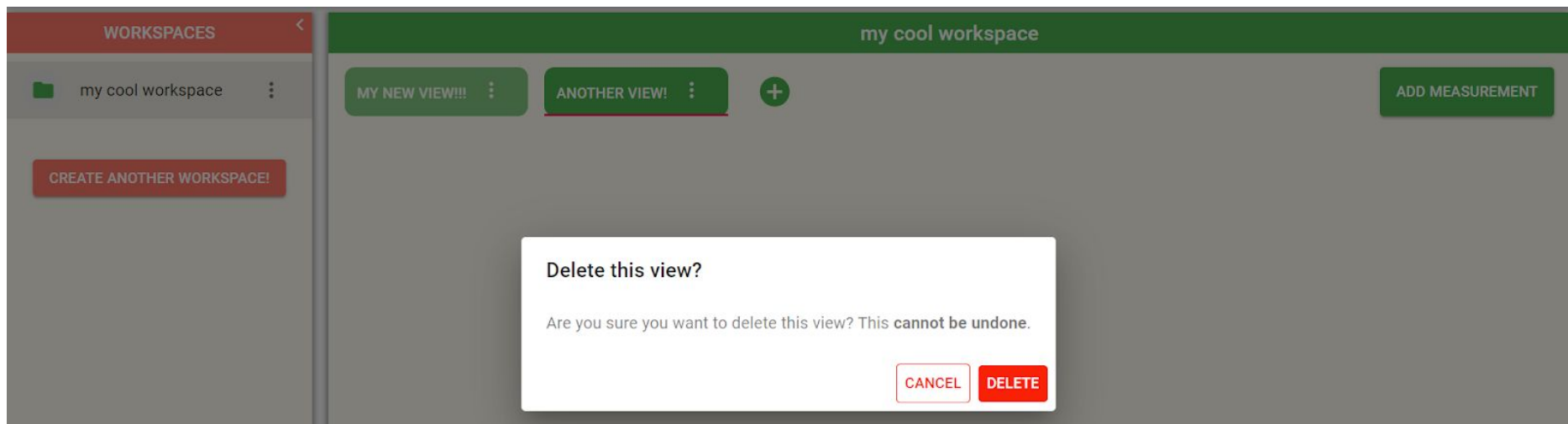
Switching between views

1. Create at least two views
2. Switch back and forth between them, making sure that different views are getting selected.



Deleting a view

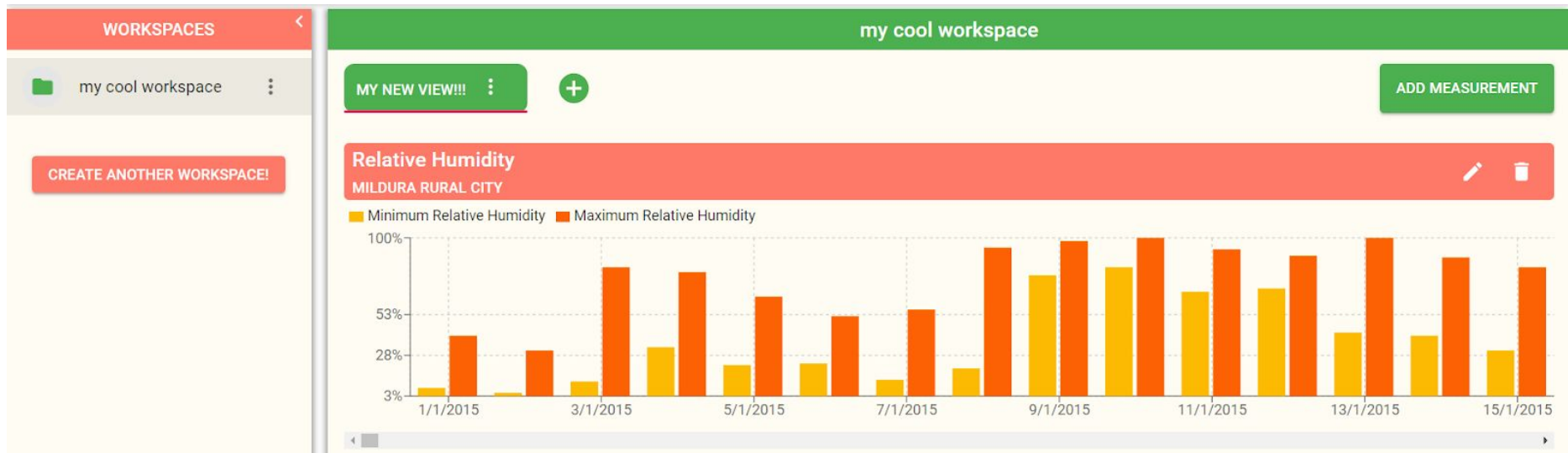
1. Make sure you have a view created.
2. Click the three vertical dots next to the view name and select delete view.
3. Verify that the view has been removed from the tab bar.



Measurement Visualisations

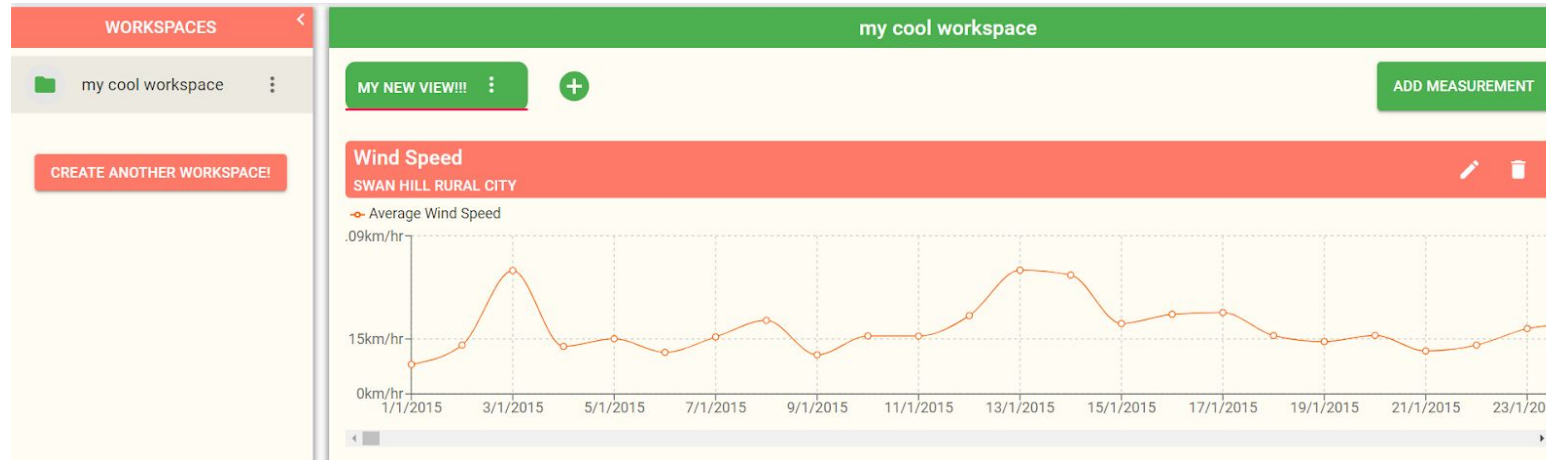
Adding a measurement visualisation

1. Make sure you have selected a view.
2. Click the add measurement button.
3. Select a measurement, visualisation and location from the options provided.
4. Click save.
5. Verify that the measurement was correctly added to the view (same measurement, location and visualisation type).



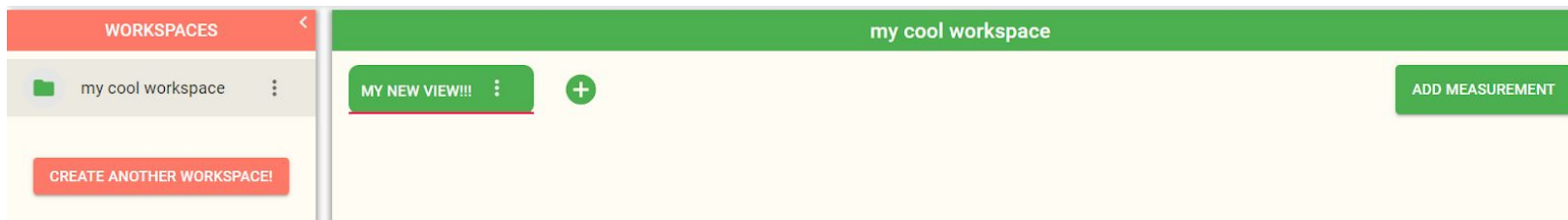
Updating a measurement visualisation

1. Make sure you have added a visualisation to a view.
2. Select the edit icon in the header of the visualisation.
3. Change the measurement type, visualisation and location.
4. Click save.
5. Verify that these changes were reflected in the view.



Deleting a measurement visualisation

1. Make sure you have created a measurement visualisation in a view
2. Click the delete icon in the header of the visualisation
3. Confirm the delete.
4. Verify that the visualisation has been removed from the view.



Limitations and Potential Improvements Based on Tests

Frontend Performance

- Firstly, many users have noticed significant performance issues when using the frontend application when loading locations on the interactive map or loading in measurement visualisations.
 - They have pointed out that the application “freezes” and does not allow them to continue working.
 - This has been a significant pain point for users and requires immediate attention.
- It seems that the core of this issue is coming from rendering complex visualisations which tend to block the UI when painting to the screen.
- To solve this issue, the team will invest time in building custom visualisation solutions with much lower level libraries to ensure that performance is optimal.
- Additionally, animating and producing many visualisations with React.js may be causing performance issues.
 - To address this, the team can explore options to use React.js for the main parts of the application while building visualisations outside of react to gain better performance.

Predictive model Performance

- Bushfire prediction model and forecasting model shows that it will be relatively unbearable to wait longer for computation to end in the foreseeable future because the size of the dataset will expand which can cause long computation time and may cause distress for technical users.
 - May consider either replacing bushfire prediction with a more efficient model or implement a slightly more efficient method to compute prediction and forecasting which can be used to predict bushfire occurrence.
- Weather forecast models lack the capability to forecast beyond 7 days and openweather api will not allow prediction beyond 7 days. The length of forecasting is really short and users may want to know anything that happens a month from the day they inspect the web app.
 - May consider replacing openweather api with a Long short-term memory (LSTM) model which can be used to forecast weather conditions in a long period of time(monthly, yearly, etc).

Customisation Options

- Users have stated that they are very happy with the idea of fully customising their workspaces, views and visualisations.
 - However, they have found that more can be done to customise their projects.
 - Particularly, users would like to be able to move measurements around in order to create the exact layout that they want.
 - Moreover, users would like to be able to customise visualisations further by changing colours and perhaps changing other styling as well.
 - Finally, users have pointed out that there currently is no option to filter measurements in order to exclude unwanted data points.
- To solve these issues, the team has come up with the following solutions:
 - The team can implement a grid system for user views where users can drag and drop their measurement visualisations around the screen.
 - Additionally, users should be able to resize their measurements in order to fully customise the layout of their view.
 - Since the styling of views and visualisations is extremely dynamic and created at runtime, the team has suggested that we refactor the application to use a CSS-in-JS library to create a maintainable system that allows users to store their styling preferences in the database and then update the UI at runtime.
 - Moreover, the team will implement filtering options for each type of measurement.
 - These filters can be added to the create measurement dialog as well as the update measurement dialog so that users can filter their measurements with ease.
 - These filters can also be customised to match the structure and nature of the different types of measurements, visualisations and locations.

Lack of Direction and Guidance

- Many users felt quite lost when using the application since they have no prior knowledge or experience in bushfire related research.
 - Additionally, younger students found it very difficult to understand what to do and why they should choose certain visualisations or locations.
- To solve this, the team will incorporate many more visual cues and interactive guides within the application that can guide students in creating workspaces and views.
- Additionally, these guides can depict the impact of choosing certain measurements or visualisations by giving visual cues as to their significance.
 - This will allow students to gain a better understanding of why they would pick certain measurements or visualisations over others.
 - These guides can also act as introductions to bushfire research and show how different types of measurements affect the likelihood and severity of bushfires.
- Similarly, guides could be put in place to help students choose which locations to add to their measurements.
 - This could be in the form of colour coding of different locations, signifying which locations are more prone to bushfire conditions.

Sharing Work With Others

- Many users have also pointed out that they would like to share their work with others and possibly even collaborate on the same project together.
- To solve this, the team has come up with the following suggestions:
 - The application could implement some sort of printing or exporting mechanism where users can export their views to save them in a different format.
 - Additionally, users can obtain shareable links for their workspaces and views which will allow other students to collaborate on the same project.

Limitations of the Testing Process

Manual vs Automated Testing

- The team has conducted significant manual testing to verify the functional correctness and other quality characteristics of the system.
 - However, manual testing involves significantly more time to conduct than automated testing.
 - Moreover, manual testing necessitates that a team member perform the tests, using up time that could be spent elsewhere.
 - Finally, manual testing is inherently susceptible to human error which renders the test results less trustworthy than a rigorous automated test suite that can be run identically every time.
- To address this, the team should consider incorporating more automated tests into the testing process in order to have more reproducible and faster results, leading to quicker feedback loops to solve any issues.

Continuous Integration

- Another benefit of incorporating more automated tests is that the team can set up automated testing pipelines before code gets merged into the main branch as well as before the code gets deployed to production.
 - Currently, the team is performing manual reviews and manual testing before code gets merged into the main branch which involves a lot of time and is susceptible to human error.
- To solve this issue, the team should consider setting up an automated continuous integration pipeline which can automatically verify the functional correctness of the code and perform regression testing.
 - This will allow the team to quickly and easily ascertain whether there are any issues that need to be addressed.

Tests Not Carried Out

- As a result of team discussions, it was agreed that testing every aspect of the software would be extremely difficult and time consuming.
 - Moreover, most of these supplementary tests would be testing non critical sections of the application or parts of the application that change frequently, causing the tests to break.
 - Therefore, it was decided that only critical functionality would be tested to maximise the balance between testing and development.
 - This gave the team confidence that all critical aspects of the application were working and provided the team with more time to focus on fixing other issues.
- Furthermore, some tests would inherently verify aspects that were required by other tests.
 - For example, within the integration testing, getting a visualization or deleting a visualisation involved exactly the same validation and sanitization checks under the hood.
 - Therefore, only one type of test was thoroughly checked while the others were just tested on a basic level.
 - This was acceptable since by testing one thoroughly, all the other tests were automatically verified as well.
- Finally, tests involved with verifying the accuracy of weather forecasting model and bushfire predictive model were not carried out because verifying accuracy for third party api and McArthur's Forest Fire Danger Index were not part of project scope but a supplementary utility which the team will use to advance the project.
 - Identifire project is designed to be a single platform that is accessible to any users with different backgrounds to understand awareness regarding recent changes in weather conditions and bushfire.
 - Thus, the team has decided to accept the risk regarding the data modelling module not being accurate and put major effort into usability testing and performance testing.

Conclusion

To conclude, all major test types have been performed to ensure the application runs to the expected standard . This includes unit, integration and end-to-end testing. In addition, useability testing was utilised to gain real world feedback from potential users. From a development point of view, the team also incorporated code reviews to ensure that any new functionality was up to the standard of the rest of the application and it did not have any adverse side effects. Through the testing, flaws and bugs within the system were able to be identified and rectified further improving the quality and user experience of our application.

Appendix

Figure 1: weather condition provided by openweather api.

```
{  
  "lat":20.12,  
  "Lon":33.66,  
  "timezone":"AU/Melbourne",  
  "timezone_offset":-18000,  
  "daily": [  
    {  
      "dt":1595268000,  
      "sunrise":1595243663,  
      "sunset":1595296278,  
      "temp": {  
        "day":298.82,  
        "min":293.25,  
        "max":301.9,  
        "night":293.25,  
        "eve":299.72,  
        "morn":293.48  
      },  
    },  
  ],  
}
```

```
"feels_like":{
  "day":300.06,
  "night":292.46,
  "eve":300.87,
  "morn":293.75
},
"pressure":1014,
"humidity":82,
"dew_point":295.52,
"wind_speed":5.22,
"wind_deg":146,
"weather":[
  {
    "id":502,
    "main":"Rain",
    "description":"heavy intensity rain",
    "icon":"10d"
  }
],
"clouds":97,
```

```
"pop":1,
```

```
"rain":12.57,
```

```
"uvi":10.64
```

```
},
```

```
"..."
```

```
}
```

```
}
```