FIT3162 Code Report

Team: 7
Team members:

- Andrew Duong
- Ming Shern, Siew
- Jordan Kapul

System Architecture

- The system architecture consists of 4 main modules.
- Each of the external modules can only interact with each service via a public interface, hiding each service's internal implementation and therefore allowing each service to change its internal implementation without any concern of causing breaking changes in external systems.
- The front-end application communicates with a Node.js application via a REST API in order to pull data from the database and populate the user interface.
- The Node.js application handles all incoming HTTP requests. The data storage service is responsible for handling data requests to and from the database. The data preparation service is responsible for handling data clean-up and formatting in order to transform raw data into correctly formatted data needed for storage and modelling.
- The data modelling service is responsible for predicting the amount for occurrences of bushfires by running machine learning algorithms on provided historical data. The back-end architecture utilises a centralised relational database in order to share data between all services and allow easy transaction management.

Quality Considerations

Robustness

Data Preparation

- All data entering the database will be cleaned up to ensure it is properly formatted and will not cause any issues regarded to invalid inputs within the system
- Weather data is taken directly from the Bureau of Meteorology's FTP server ensuring that there are minimal points in the path of the data from its source to the database for it to become corrupted or unreadable.
- IF-EISE conditions for situations where dataset format is expected are implemented.
    - For instance, ftp://ftp.bom.gov.au processed a similar format for all csv files such as the first 12 rows and last rows should be skips which can be done with r programming.

Backend

- Any request received from the client is always sanitised and validated in order to ensure that they are properly formatted and that any malicious input is caught before adversely affecting the system.
    - Data sent to the server is validated against various business logic rules which dictate the correct format for the data and raise appropriate errors when required.
    - These errors are then packaged into http responses that allow the client to inspect what went wrong and how to address the problem.
- Furthermore, requests that would require accessing multiple database tables are always carried out by performing atomic transactions.
    - Consequently, queries involving multiple tables are always executed safely by running them within atomic transactions that will always leave the database in a consistent state.

Frontend

- The frontend always makes sure to validate user input before attempting to send requests to the backend server.
    - This allows a quick feedback loop whereby the user is immediately notified of erroneous input, allowing them to quickly and easily change the input without having to wait for a response from the server.
    - Additionally, this limits the network traffic targeted towards the server, eliminating unnecessary traffic and overheads.
- Moreover, the frontend will disable inputs that are not allowed to be used. This improves feedback to the user and avoids the possibility of the user performing actions that are not permitted.

Predictive Modeling

- Exception handling is done to detect error return from openweather api using TRY-EXCEPT. This happens when api has experienced a down server or part of information from api has been removed which made developer unable to access removed resources leading to error messages being returned.
- Robustness for data preparation and predictive modelling will be compromised if the data retrieved from ftp://ftp.bom.gov.au changed its format. Therefore, refactor may be expected in future when bom.gov.au decides to change the format of the file.

Scalability

Data preparation

- The data is sourced directly from the Bureau of Meteorology's FTP server which contains data from every weather station in Australia reaching back as far as the stations have existed providing the system with any data that may be required for the system.
- The server accessed is updated daily with new measurements and will continue to be a valid source for any new values that may be required.

Backend

- The backend is built using Node.js which allows our application to scale to a large number of user requests with ease due to its non-blocking event loop.
- Additionally, to facilitate efficient access to our database, our application pools database connections in order to reuse connections across multiple requests.
    - This allows the application to keep connections open and to immediately serve incoming requests without having the overhead of establishing new connections each time.
- The backend architecture has been designed to allow for the easy addition of new types of measurements and visualisations.
    - This greatly improves the maintainability and extendability of the system.
    - This has been primarily achieved by constructing database queries that can dynamically change the intended table name as well as column names.
        - This allows the system to dynamically alter queries at runtime to target appropriate tables based on the particular measurement or visualisation required.

Frontend

- The frontend has been designed to easily accommodate new measurements and visualisations.
    - This has been accomplished by creating a visualisation factory that can be customised to produce tailored components based on any combination of measurement, location and visualisation type.
- Furthermore, the frontend application employs a sophisticated build pipeline to ensure an optimal frontend build:
    - The build pipeline bundles assets such as JavaScript or CSS files into a single bundle, enabling faster loading times as the web server only needs to send back a few files.
        - Additionally, this allows the team to easily modularise their code by splitting it across multiple different files.
        - This consequently greatly improves the understandability and maintainability of the code as the team can now create locally scoped files that can export the code required.
    - The build pipeline also automatically creates a dependency graph based on what code has actually been used.
        - This allows the build to only include code that is required, eliminating unused code as well as reducing the build's overall bundle size.
- However, as highlighted in the limitations section below, the frontend is very performance heavy as it produces complex visualisations for potentially thousands of data points spread across multiple measurements.
    - This ultimately results in a choppy experience as the browser struggles to render so many complex visualisations.
    - Potential improvements are discussed in the limitations section below.

Predictive Modelling

- The software implemented is also designed to be able to process any given size if its data can be processed within the memory of the device.
- Within the software code itself, it is visible that the amount of time to forecast weather conditions will be constant since it will always forecast next 7 days, no more and no less whereas the predictive modelling may take a long time to compute depending on the number of rows in a dataset it will compute.
- The software should be able to produce the desired output if the dataset within a device does not exceed the amount of available memory.

Portability

Data Preparation

- As the data is inputted directly into the database, it does not have any software requirements and will be compatible with any infrastructure.

Backend

- Since the backend application is built using Node.js, the server can be run on any infrastructure that has Node.js installed.
- Additionally, the environment variables required to correctly configure the application are integrated via a utility tool that correctly assigns the variables based on the operating system.
    - This ultimately means that the application can be built on any operating system without any compatibility concerns.

Frontend

- The frontend application has been built to work across all major, modern browsers.
    - The frontend application employs a build pipeline that converts all modern JavaScript syntax and features into old syntax, allowing the code to be run on any browser, regardless of whether or not it supports the latest features.
    - Additionally, browser specific prefixes are automatically injected into CSS files to ensure compatibility across all browsers.
    - Finally, the frontend utilises a CSS reset which allows consistent styling across all browsers.

Predictive modelling

- If the system meets software and library requirements, predictive modelling will be able to produce functionality as expected.

Security

Data Preparation

- All requests between the database and the data preparation script are secured through inbuilt SQL database measures to ensure that data is authenticated.

Backend

- All requests that require privileged access must contain appropriate access keys retrieved by logging in through the user's browser.
    - These keys are then cross referenced against a third party authentication provider to validate their authenticity and to populate the request with appropriate user data such as the user's email address.
- All API handlers will immediately throw an error and respond to the client with a 403 forbidden if they do not have the appropriate credentials.
- Furthermore, the backend API treats all client data as suspicious and sanitises the client data by escaping any malicious input found.
    - This is done to prevent XSS (cross site scripting) attacks whereby malicious actors can inject code into the client data which can then run on another user's browser.
    - Additionally, sql injection is prevented by escaping any sql identifiers or literals provided by the user.
- Once the user's identity has been established and the data provided has been sanitised, the system will then perform checks against the database to ensure that the user has appropriate credentials to access the desired user data.
    - For example, if bob@gmail.com is attempting to access a workspace that belongs to james@hotmail.com, the backend will catch this by sending back an error to the client.
- Moreover, sensitive environment variables such as the database password are not hardcoded into the code.
    - Alternatively, these environment variables are stored in .env files that reside locally on the machine.
    - It is important to note that these configuration files are NOT committed to the git repository as this would be a major security issue.
- Finally, although errors are always sent back to the client, the backend makes sure to include only appropriate information. This is done to ensure that no sensitive information is leaked to the client.

Frontend

- The frontend application allows the user to log in via a third party authentication provider.
    - This enables the client to store user access tokens within the browser and subsequently make authenticated calls to the backend API.
    - These tokens have automatic expirations set on them, ensuring that new tokens must be generated on a regular basis.

Predictive modelling

- There was no form of security in any part of the predictive modelling since implementation for a secured use of predictive modelling is not part of the project scope.

Limitations and Potential Improvements

Data Preparation

- Currently the data preparation scripts are designed to work directly with the BOM FTP server. This means that in its current state it would not be able to deal with data from any other source. Although, we have designed the code in a way where it would require minimal effort to adapt to another data source.

Backend

- Currently, certain database queries are dynamically constructed using manual string concatenation combined with appropriate escaping mechanisms to avoid sql injection attacks.
    - However, the code currently relies on developers to remember to correctly escape these values.
    - In order to improve this for future work, the team would build a reusable module that would automatically escape values and allow for easier development of dynamic queries.

Frontend

- Currently, the frontend application will always make new requests to the server to fetch relevant measurement data.
    - However, since this data is primarily composed of static data, an improvement for the future would be to incorporate client side caching of the measurement data.
    - This would most likely involve the use of data storage solutions already available within the browser such as IndexedDB (https://developer.mozilla.org/en-US/docs/Web/API/IndexedDB_API).
    - This locally cached data would then need to be synced with the server by performing synchronisation checks.
- Additionally, the frontend currently suffers from performance issues when loading a lot of measurement data.
    - This occurs since rendering large amounts of data across multiple visualisations is an expensive operation and often blocks the UI.
    - This can be overcome by utilising low level visualisation libraries to build more performant visualisations as opposed to using off the shelf components.
    - Moreover, the frontend could employ virtualisation techniques whereby only the visualisations that are visible to the user get rendered.
- Finally, the frontend would greatly benefit from implementing client side routing which would enable the user to bookmark their views and visualisations for easier access.
    - Currently, all paths from the application url result in the homepage being displayed.
    - Since the frontend is a single page application, implementing client side routing would allow for easier navigation of the site without incurring the overhead of further network requests and page refreshes.

Predictive Modelling

- 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).
- Manually executing a data preparation module and predictive module can be time consuming because repeated tasks are expected.
    - Implementation for automation to compute data preparation module and predictive module which will be able to reduce repetition tasks for the technical user.
- 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 with extensive research for a suitable method to compute prediction and forecasting which can be used to predict bushfire occurrence.
- Finally, Reliance in the involvement of trusted third parties to forecast weather conditions may not produce accurate results.

Other Limitations and Improvements

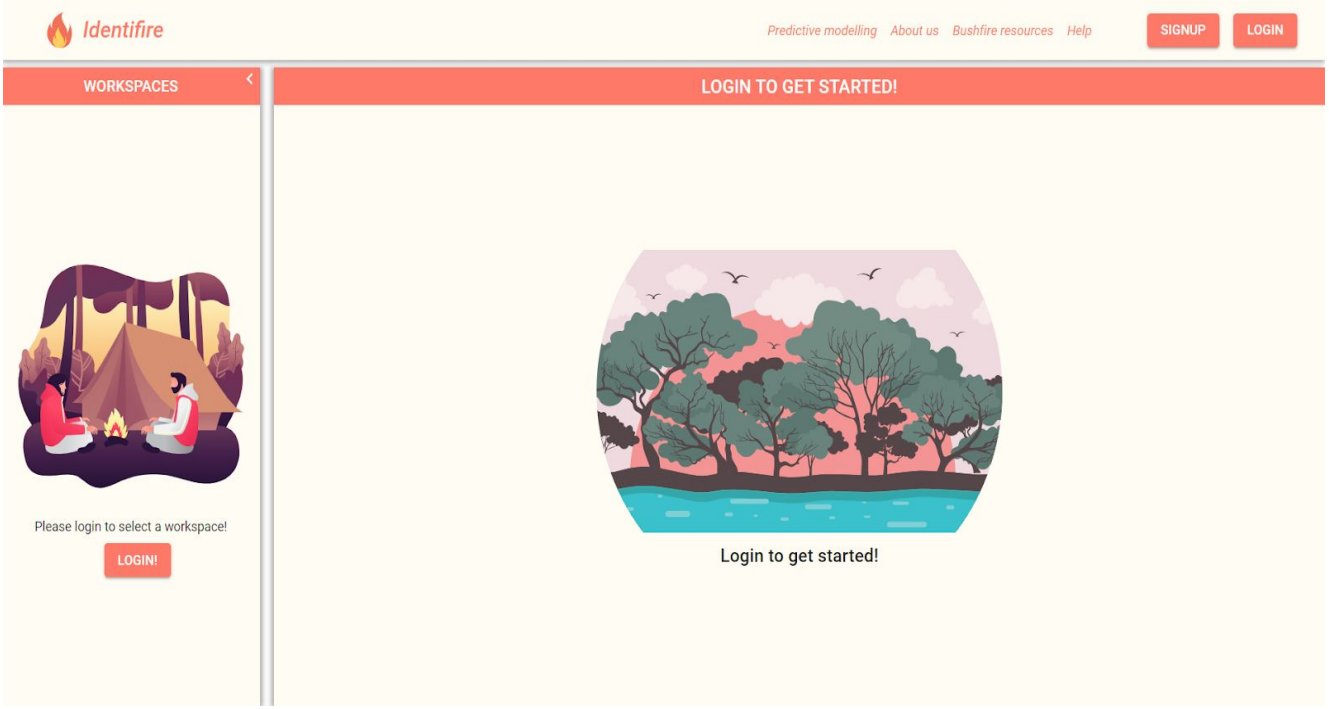
- For a description of limitations and improvements found by conducting tests, please refer to the limitations and improvements section in the test report.

End User Guide

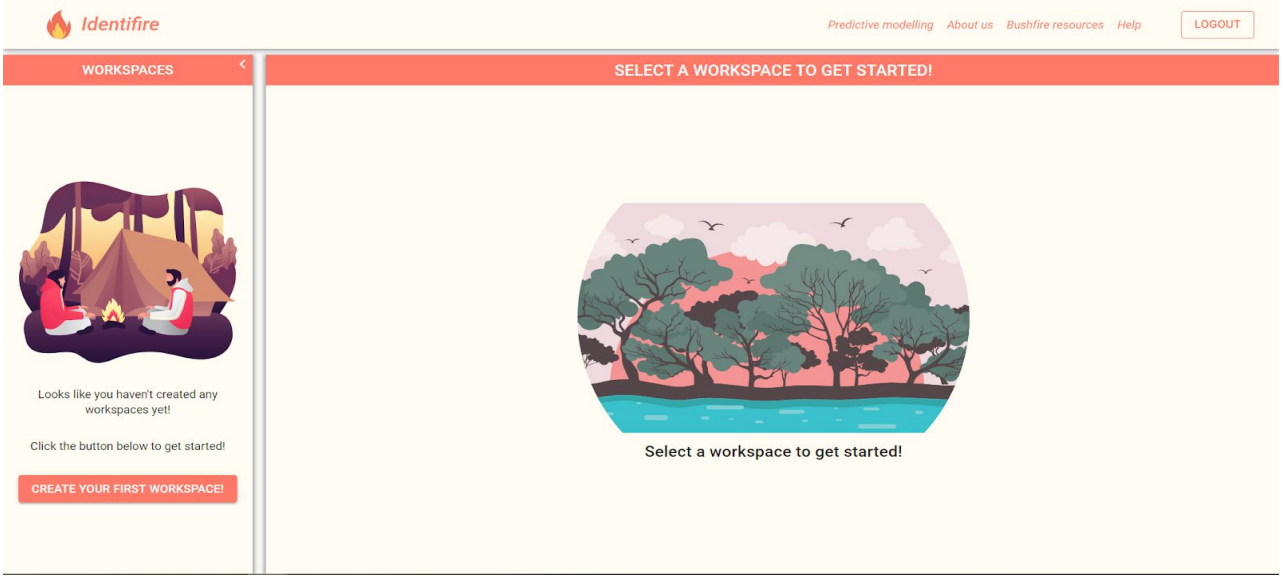
Welcome to identifire! Our application allows students to easily visualise important bushfire data by creating custom measurement dashboards which we call “views”.

Registering An Account

In order to fully utilise the identifire application, you will need to first log in or sign up.



- You can find signup and login buttons at the top right corner of the screen.
    - Alternatively, you can use the login button located within the workspaces drawer (located on the left side of the screen).
    - To easily log in for the future, we recommend logging in with your google account.
- Once you have logged in, you should see the following screen:

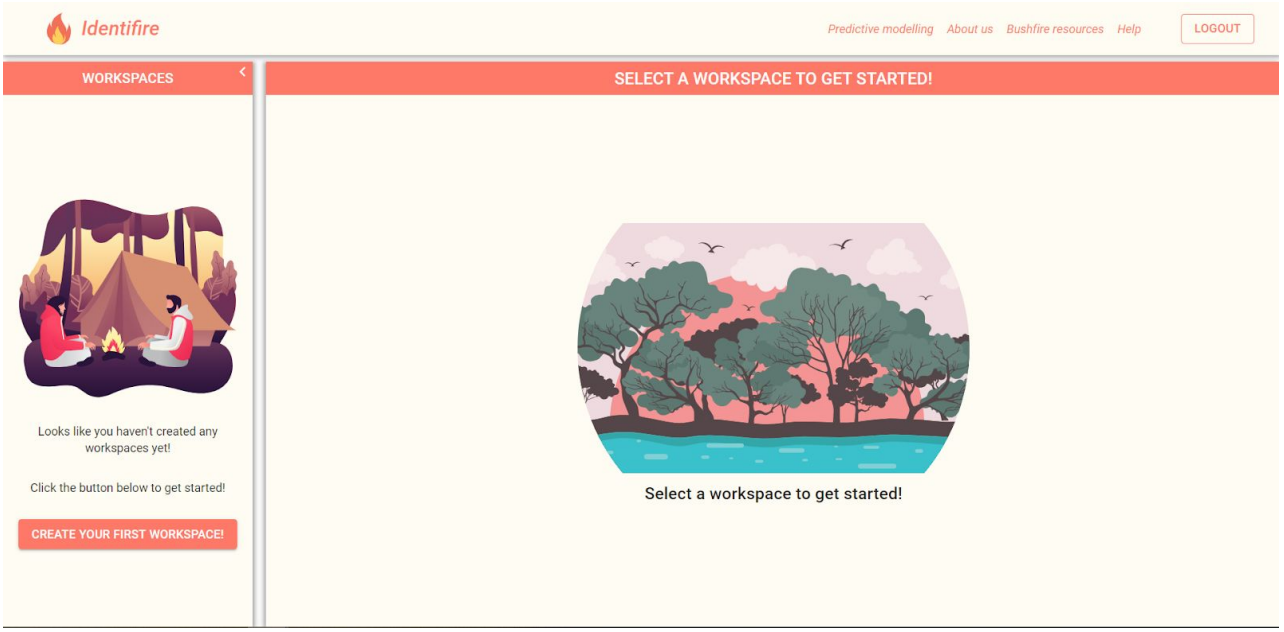


Workspaces

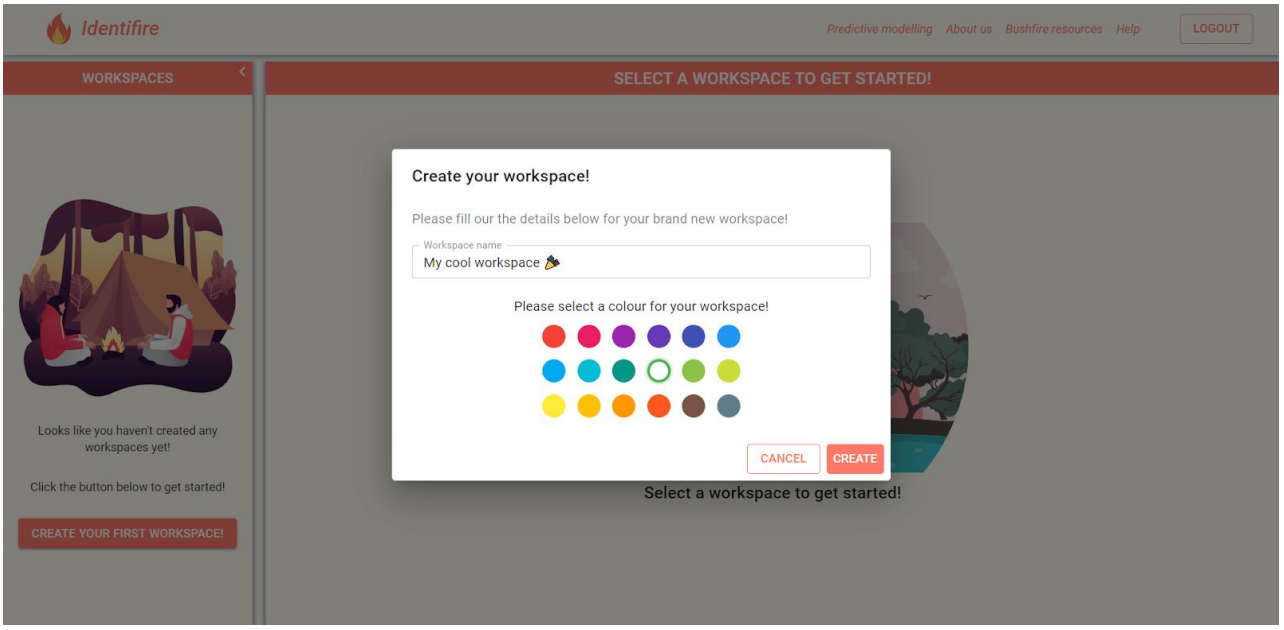
Workspaces allow you to better organise your visualisation projects into separate folders.
Workspaces have a unique name and colour, allowing you to quickly identify your workspaces and perhaps organise them according to colour.

Creating Workspaces

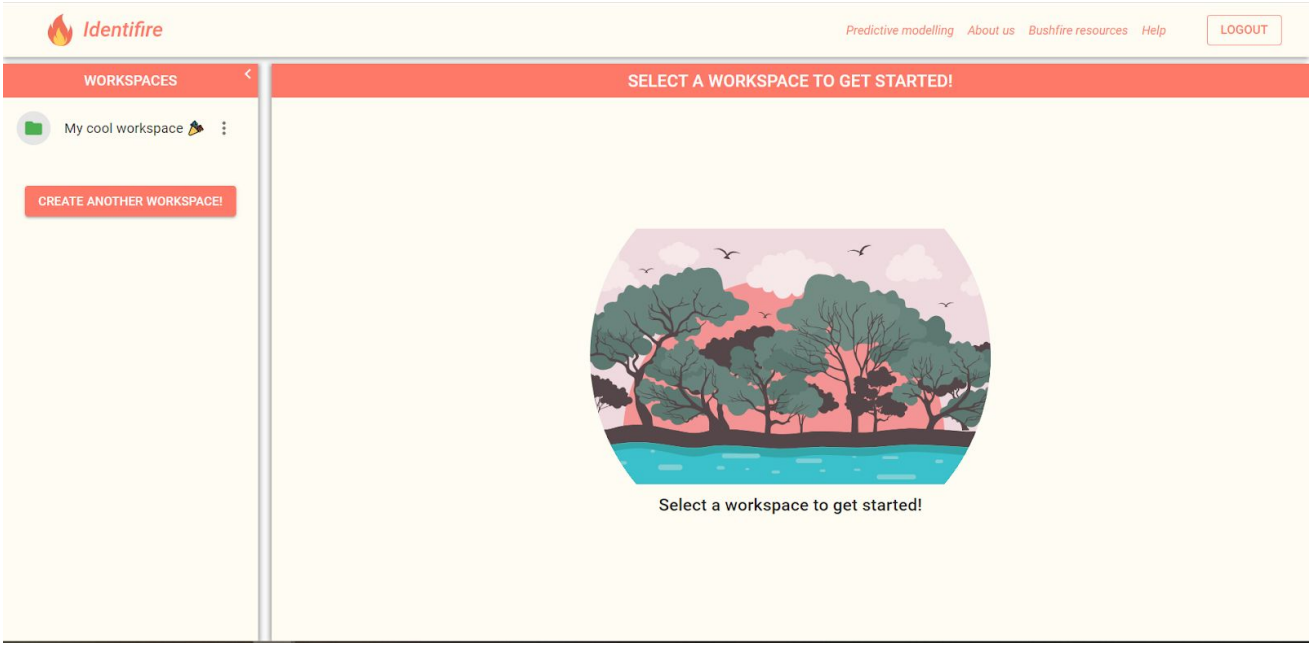
- In order to create your first workspace, click on the “create your first workspace” button located in the workspaces drawer (found on the left side of the screen).

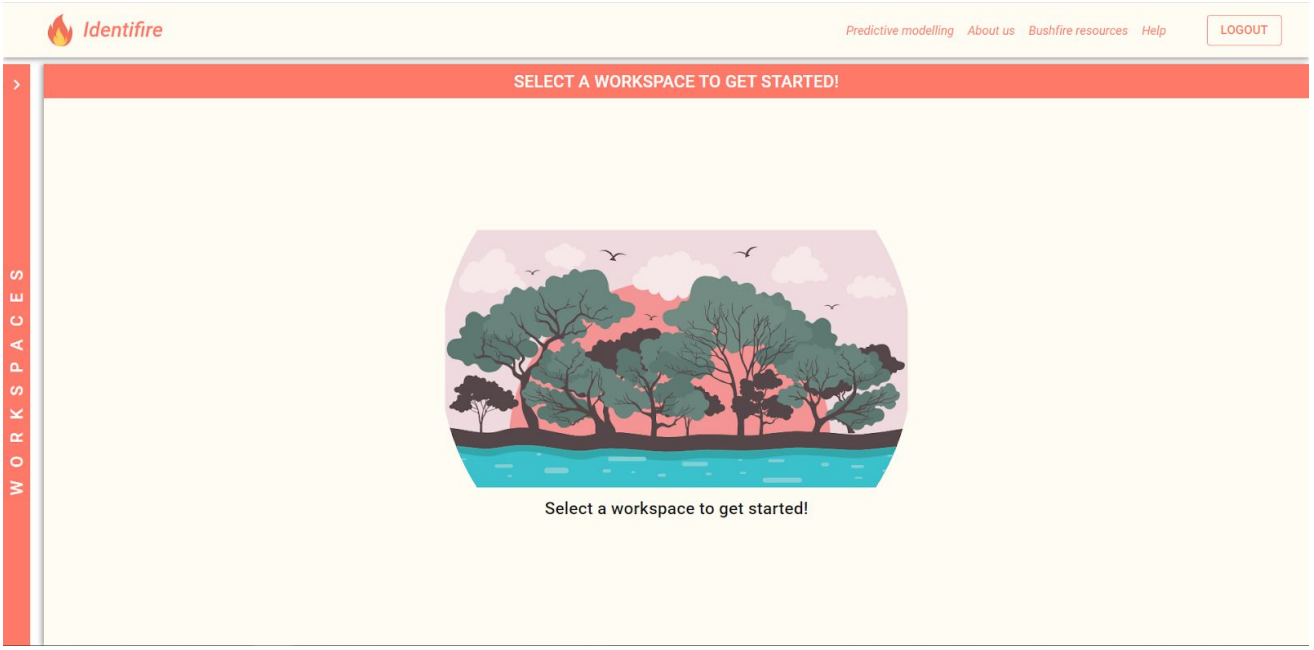


- You can now select a name and colour for your workspace!



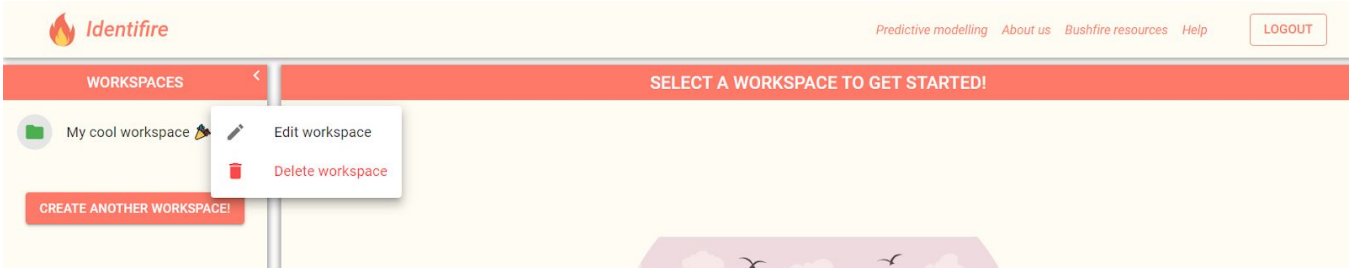
- You can find your workspace located within the workspaces drawer!
    - It's also important to note that you can minimise the workspaces drawer by clicking the left arrow at the top right of the drawer.
    - To open the drawer again, simply select the right facing arrow located at the top of the collapsed workspaces drawer.



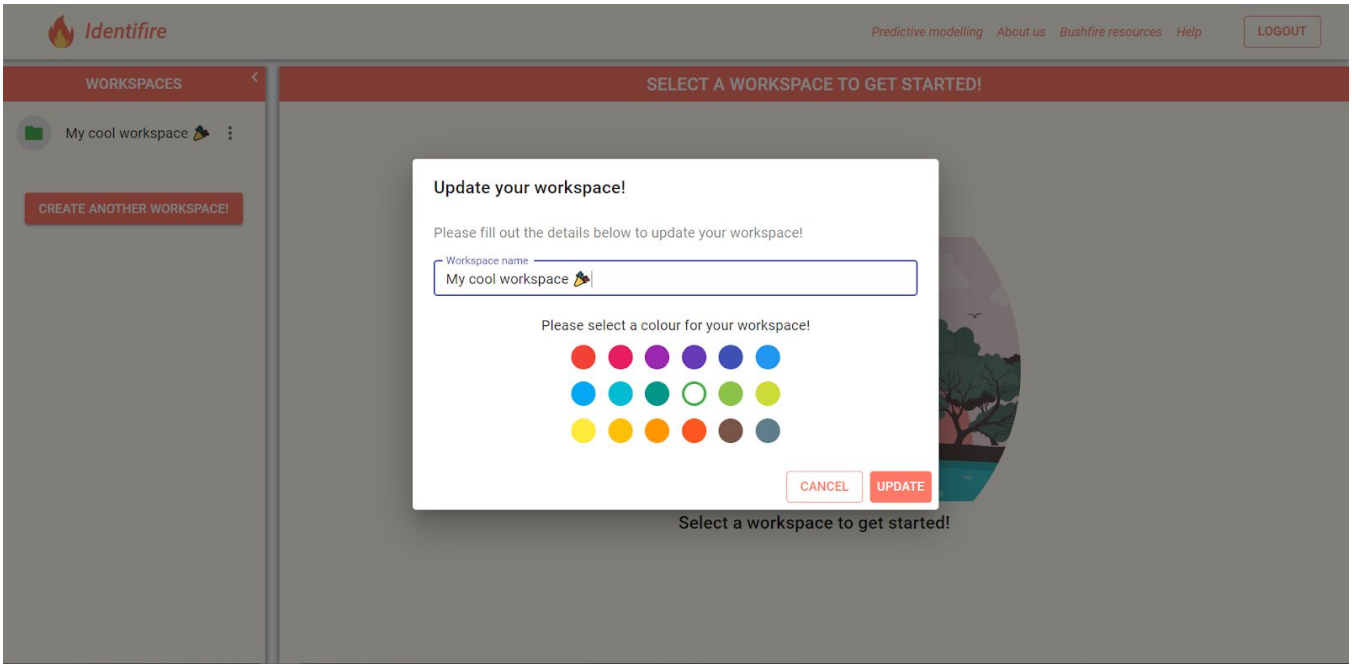


Editing Workspaces

- To edit your workspace details, simply click the three vertical dots located next to the workspace name in the workspaces drawer.
- You will be given the options to edit or delete your workspace, click “Edit workspace”.



- You can now change the name and colour of your workspace!



Deleting Workspaces

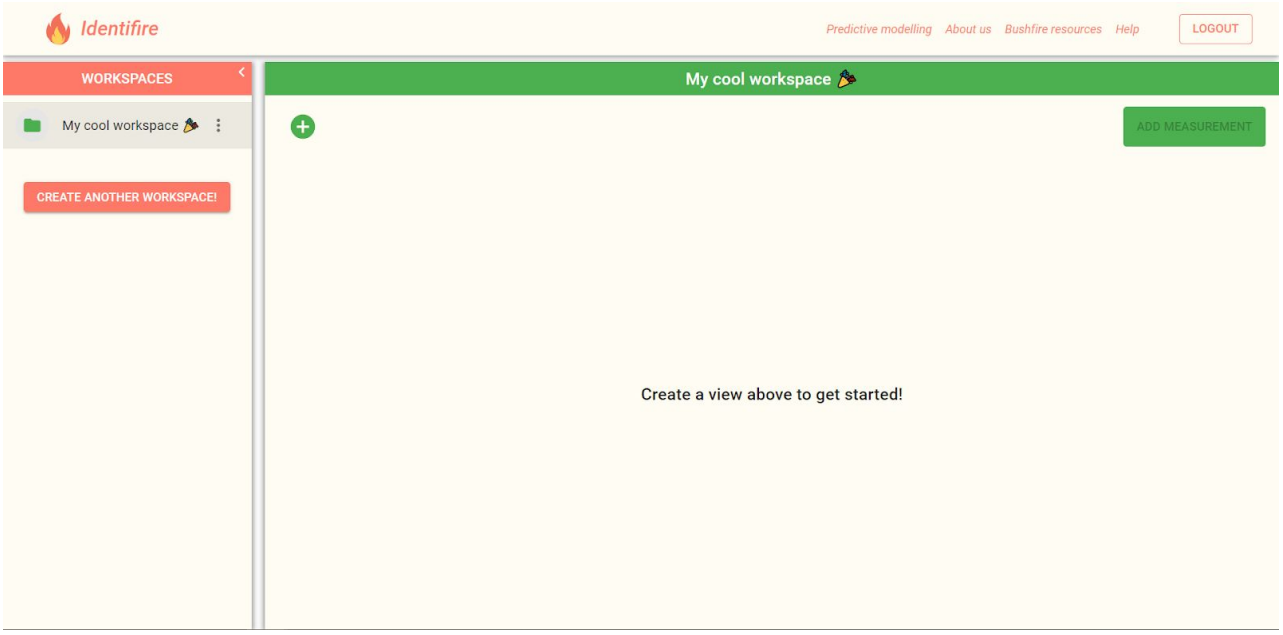
- In order to delete a workspace, simply click the three vertical dots next to the workspace name located in the workspaces drawer and then select “Delete workspace”
- You will then be prompted to confirm the delete as this action **cannot be undone**.

Views

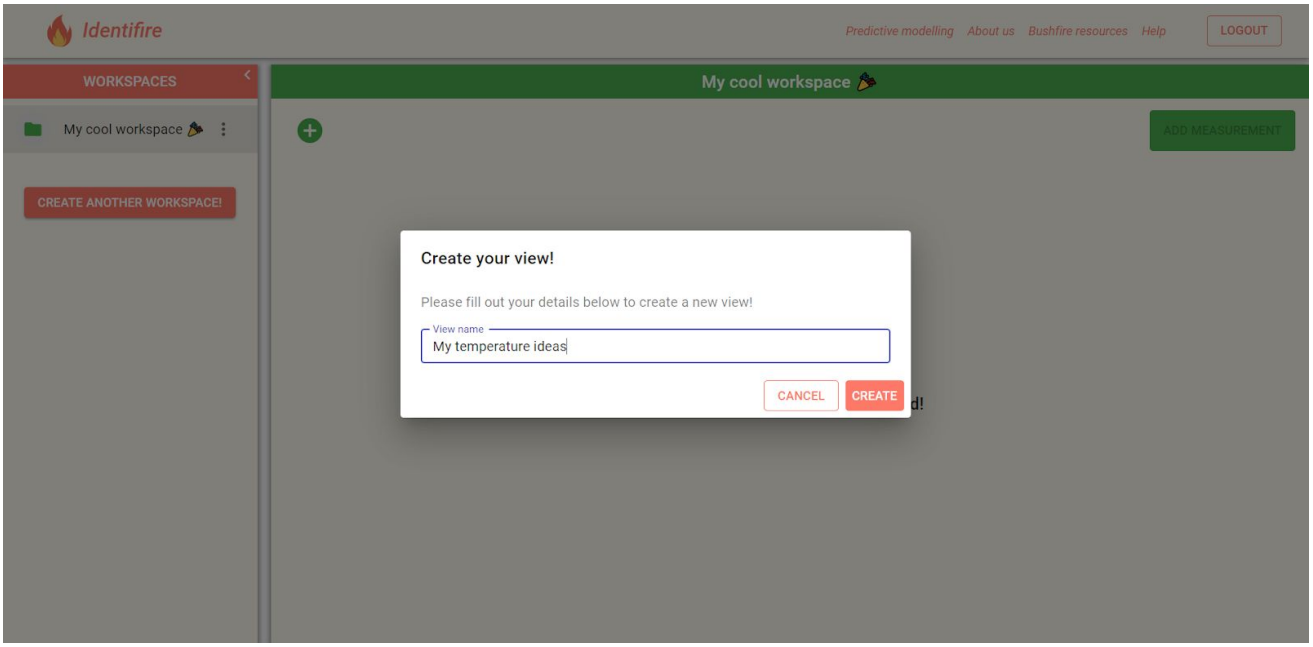
- Views are custom dashboards that allow you to select what measurements you would like to visualise.
- You can easily switch between different views by selecting different view tabs!
- You can also click on predictive modelling to view visualisation of bushfire.

Creating views

- To create a view, you must first select a workspace!
- Simply click on a workspace in the workspaces drawer to select that workspace.
- After selecting a workspace, notice how the right view panel has updated to contain the workspace information.

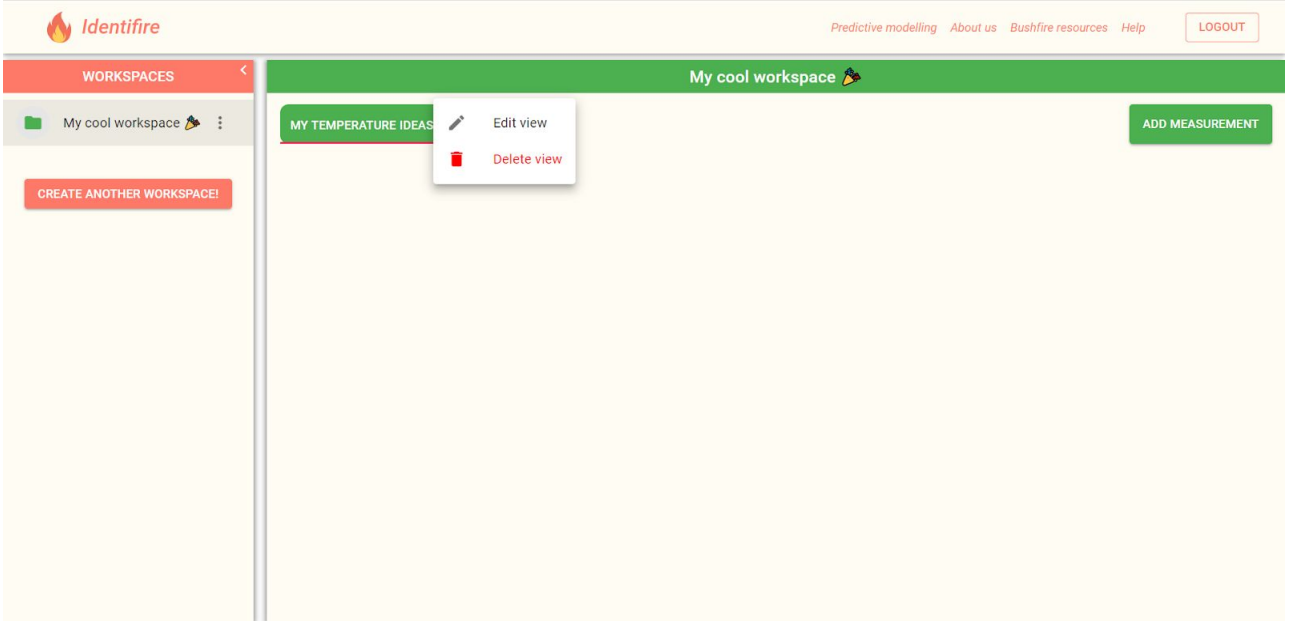


- To add a view to your workspace, click the plus icon located in the top left of the view panel.
- This will open up a dialog that allows you to select a name for your view.



Editing views

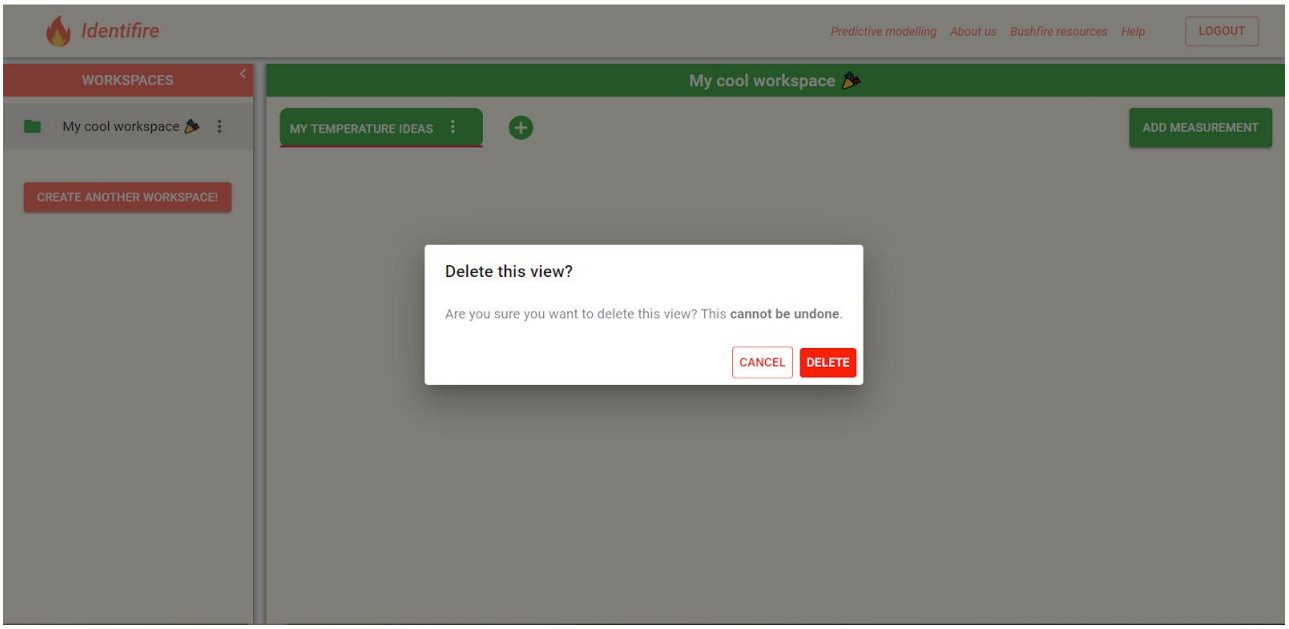
- To edit your view details, simply click the three vertical dots in the view tab to open up an options menu.



- Within the options menu, select “Edit view”.
- This will allow you to rename your view!

Deleting views

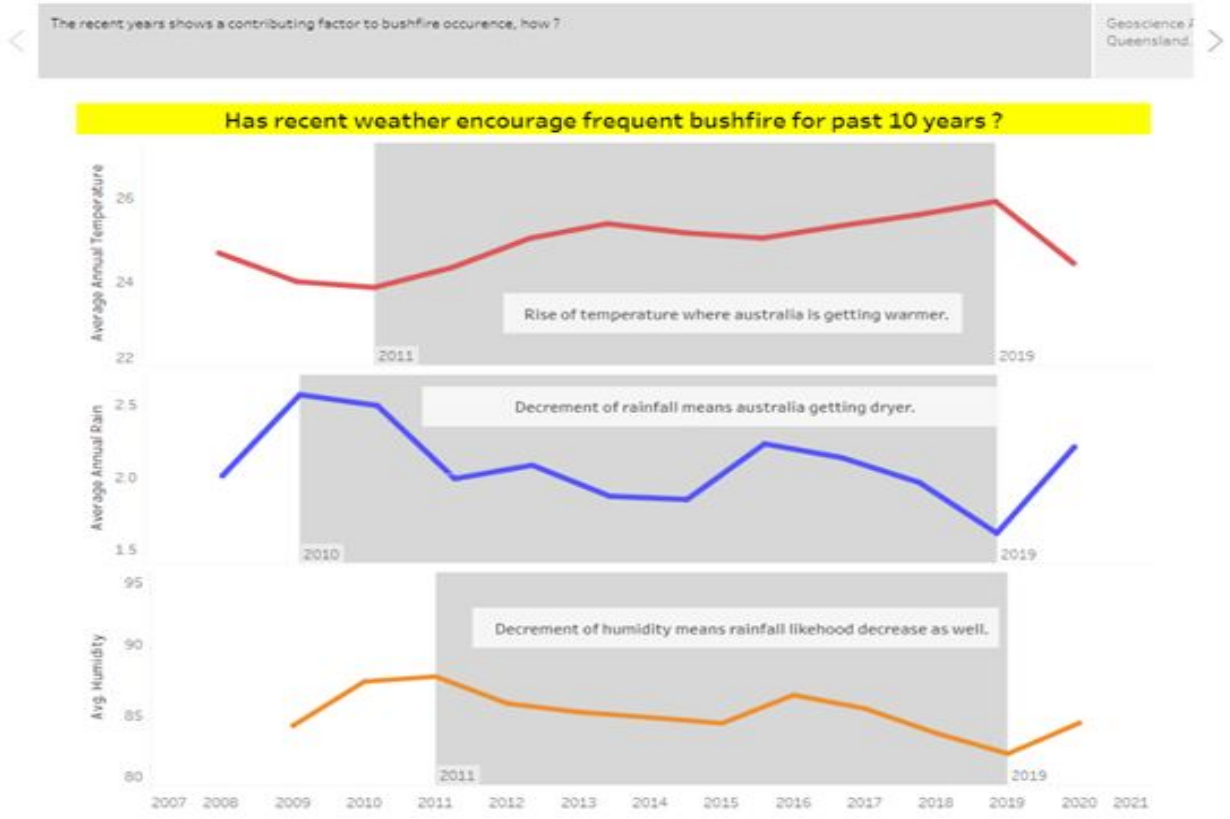
- To delete a view, simply click the three vertical dots in the view tab and select “Delete view”.
- You will then be prompted to confirm the deletion since this action **cannot be undone**.



View bushfire prediction visualization.

- To view bushfire prediction, simply click the predictive modelling.
- You will then be prompted to a new web page which will guide you through what you need to know about bushfire with help of different visualisations.

Factor and prediction of bushfire occurence.

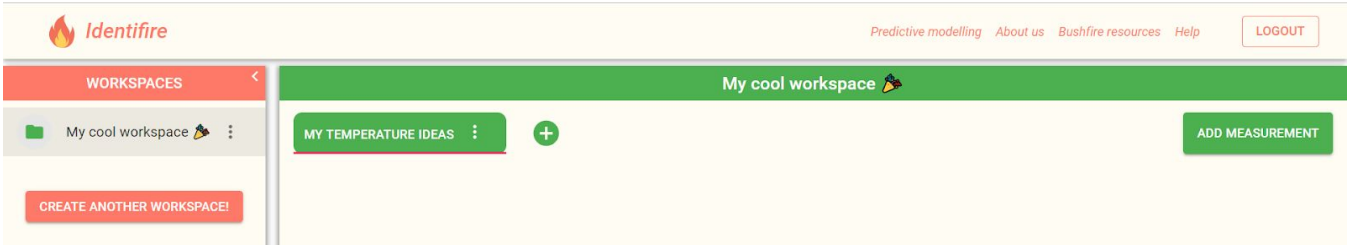


Measurement Visualisations

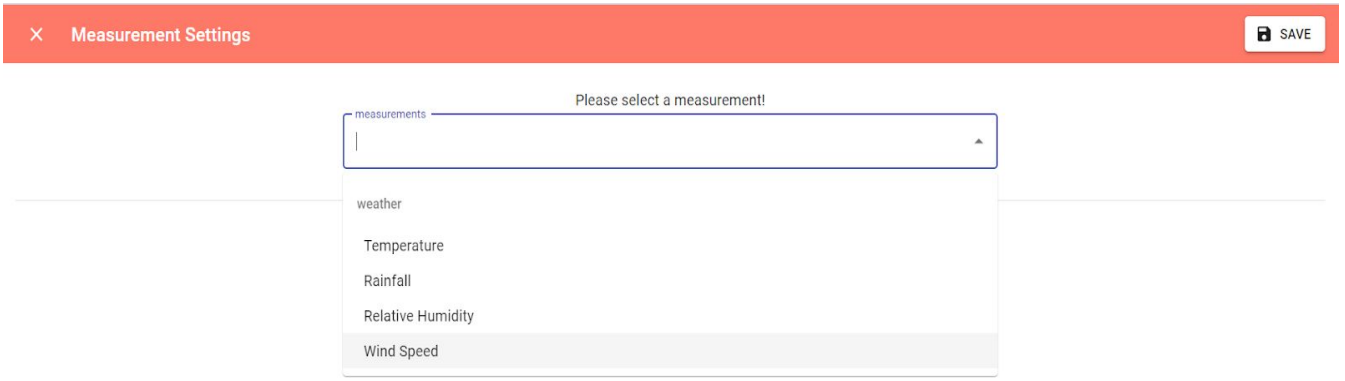
- Within your views, you can create visualisations to better analyse and understand important bushfire data.

Creating Visualisations

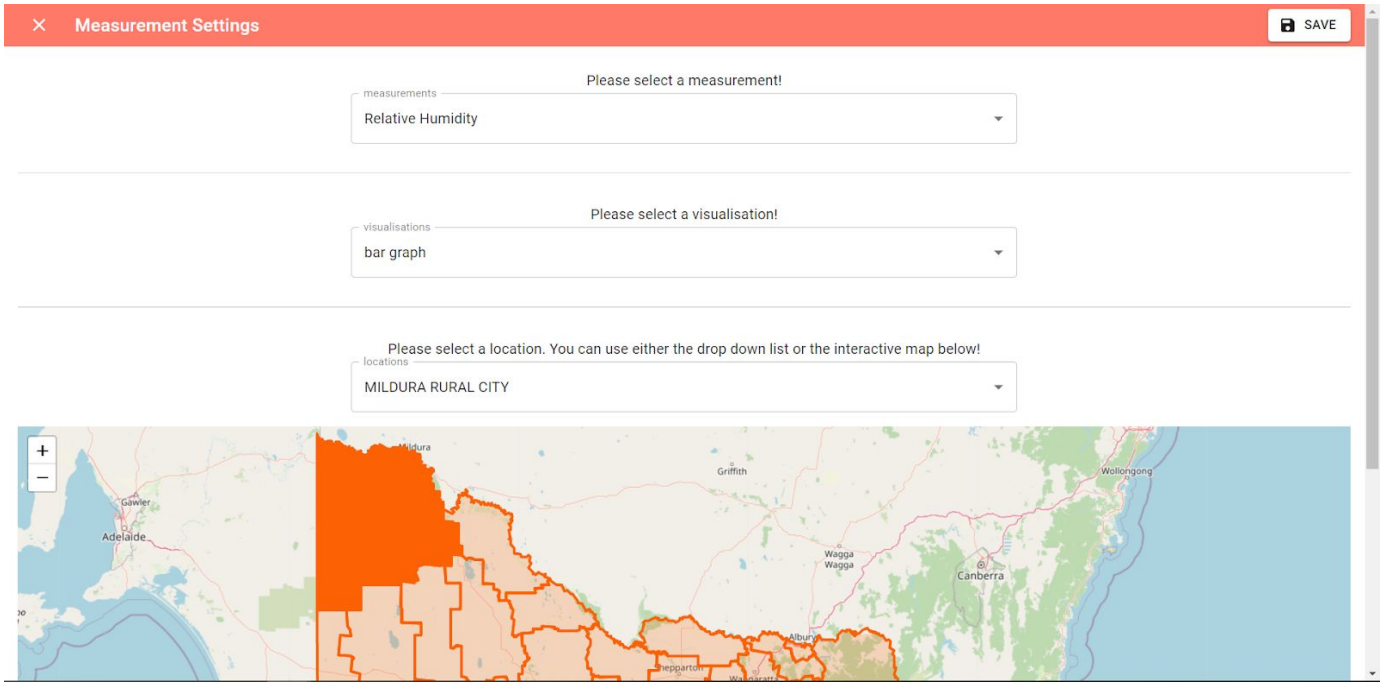
- To add a measurement visualisation to your view, simply select the view you would like to add measurements to by selecting its name in the tab bar located at the top of the view panel.
- Once a view is selected, the “add measurement” button becomes enabled and allows you to add a measurement visualisation to your view!



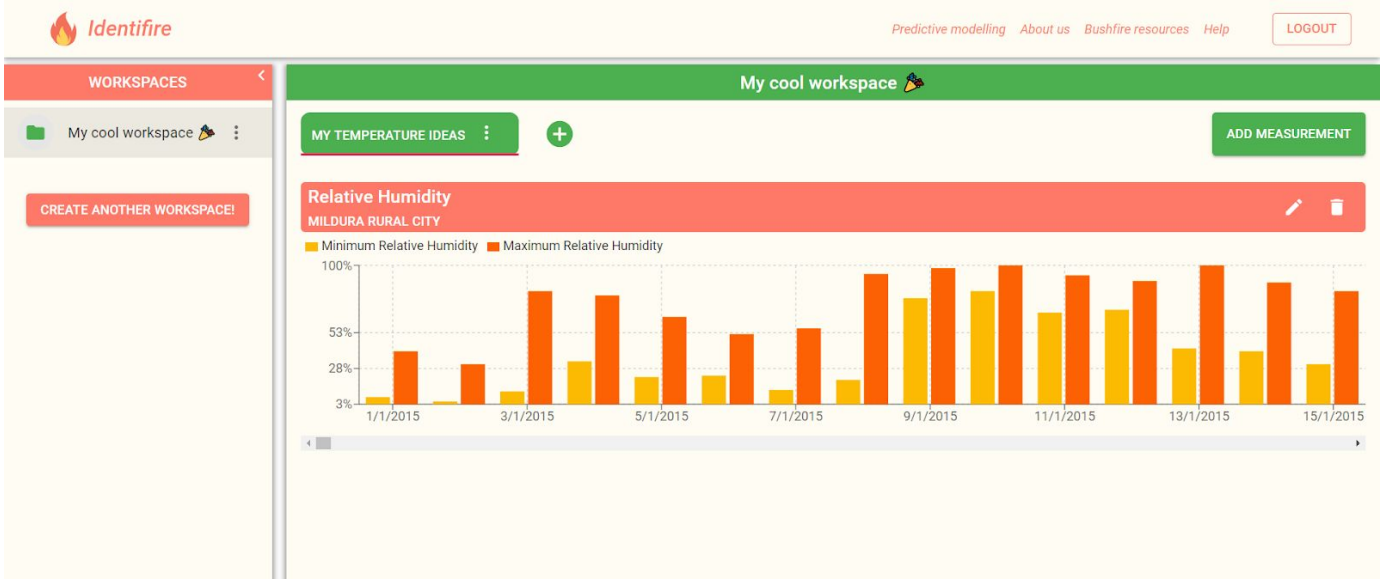
- Click add measurement to get started!
- This will open up a dialog which allows you to select various options:
    - Which measurement you would like to visualise.
    - Which visualisation to use.
    - Which location to visualise the measurement for.
- First, select a measurement using the dropdown provided.
    - Note that you can either select a measurement from the dropdown list or narrow your search by typing the measurement name in the search bar.



- Once you have selected a measurement, you can now select a visualisation type as well as the location!
- The visualisation and location dropdowns work exactly the same as the measurement drop down.
    - That is, you can either scroll through the drop down to find your option or you can search for an option to narrow down your choices!
- However, in addition to a dropdown, you can also select your location by choosing a location within the interactive map provided!
- When you are done, simply click the save button at the top right of the screen to add your visualisation to your view!

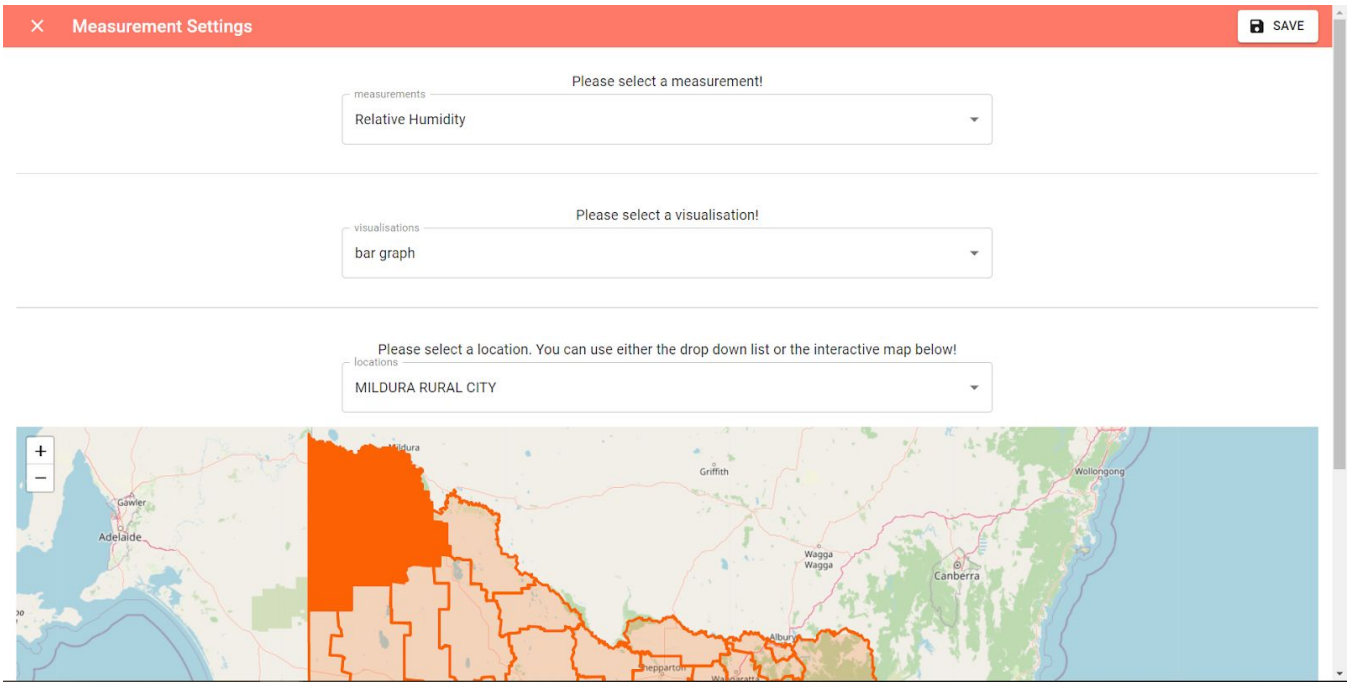


- You can now explore your new visualisation within your view!
- Scroll through the data and hover over desired data points to get more information!



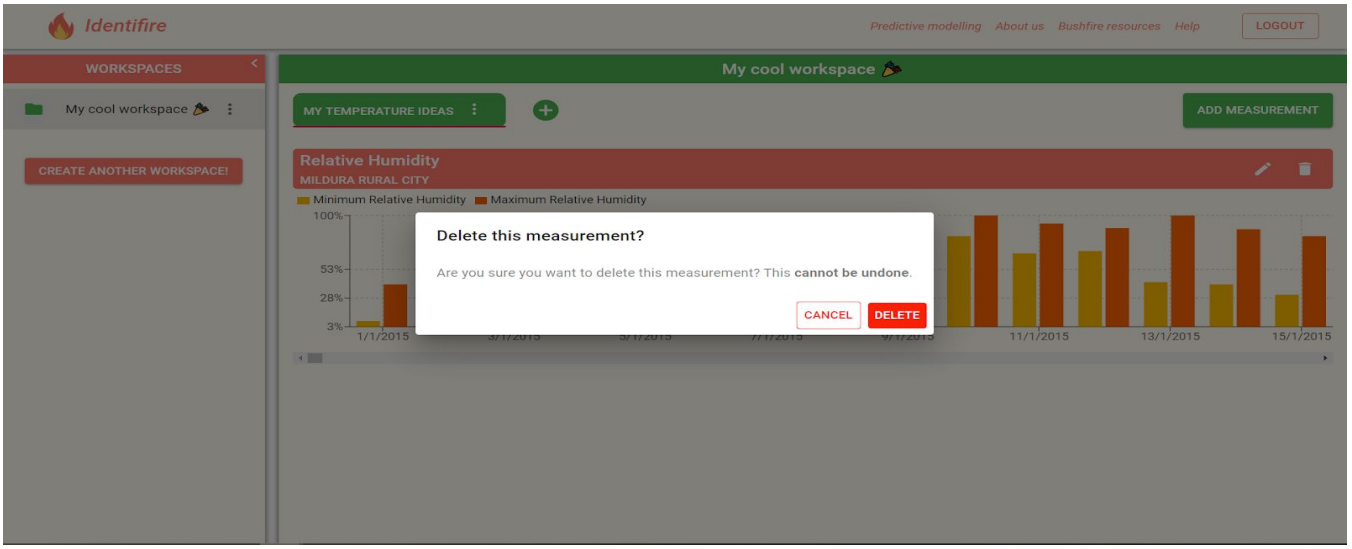
Editing Measurement Visualisations

- To edit a measurement visualisation, simply select the pencil icon located in the header bar of the visualisation.
- This will open up a dialog that is very similar to when you created the visualisation, allowing you to edit its settings.



Deleting Measurement Visualisations

- To delete a measurement visualisation, simply select the delete icon located in the header bar of the visualisation.
- You will then be prompted to confirm your deletion as this action **cannot be undone**.



Technical User Guide

Overview of Application Architecture

- The application has been broken down into several different modules which are housed within different directories within the main project repository.
- These directories form a monorepo where all of the code for the project as a whole is included in a single repository.
- The directories are as follows:
    - backend
        - This directory contains all the code required to run the Node.js web server and backend API.
    - data-prep
        - This directory contains all the data preparation scripts that were required to clean and prepare raw data required for the application.
        - These scripts also populate the database with the cleaned data.
    - database
        - This directory contains some scripts required to initialise the database and also provides instructions of how to populate the database with the latest data.
    - frontend
        - This directory contains all the code necessary to build the frontend application.
    - predictive-modelling
        - This directory contains all the code necessary to run predictive modelling on the raw data.
        - It also contains instructions on where to find larger datasets that could not be pushed to the repository due to their size.

Software Requirements

Software

- In order to run and build both the backend and frontend of the application, you must install **Node.js** on your local machine.
    - Please note that our application uses **version 14** of Node.js which can be downloaded from the Node.js website (https://nodejs.org/en/download/)
- In order to connect to the database locally, you must also install PostgreSQL on your local machine.
    - Please note that our application uses **version 12** of PostgresSQL which can be download from the postgres website (https://www.postgresql.org/download/)
    - Note that it is important to add psql to your machine's PATH in order to run commands to populate the database from dump files.
        - Here is a link that provided instructions on how to add psql to your PATH on windows: https://sqlbackupandftp.com/blog/setting-windows-path-for-postgres-tools
- The **anaconda** with **python interpreter version 3.8 64-bit** is needed to run the program and install required libraries.
    - - https://www.anaconda.com/products/individual
- RScript latest version - https://cran.r-project.org/bin/windows/base/
- RStudio latest version - https://rstudio.com/products/rstudio/download/#download
- Tableau desktop latest version - https://www.tableau.com/products/desktop/download

Libraries

The following libraries are not part of the standard python library and need to be downloaded separately.

External Library:

- pandas- easy-to-use data structures and data analysis tools for the Python programming language.
- requests & urllib3- allows developers to send and receive HTTP requests.
- xlrd,& openpyxl – allows developers to read/write data in Excel spreadsheets.

Starting the Backend Server

- The backend server requires a configuration file to be provided in the config folder found at the path backend/config.
    - This file must be named local.env and contain the appropriate environment variables to start the server, including your username and password for the PostgreSQL database.
    - A sample.env file has been provided to show the necessary environment variables.
    - However, a local.env file can be obtained by contacting a member of the team.
    - These files contain sensitive credentials which means they cannot be committed to the github repository.
- In order to start the backend server, simply navigate to the backend folder from the root directory:
    - `>> cd backend`
- Then run the following commands:
    - `>> npm install`
        - This will install all dependencies on your machine.
    - `>> npm run start:local`
        - This will start the backend server and will run on localhost:5000.

Creating a Frontend Build

- In order to create a frontend build, simply navigate to the frontend folder from the root directory:
    - `>> cd frontend`
- Then run the following commands:
    - `>> npm install`
        - This will install all dependencies on your machine.
    - `>> npm run build`
        - This will create a build directory within the frontend folder that can be served up by the backend server.
- However, you may opt to start a local development server instead which will allow you to edit the frontend code and immediately see the results reflected in the browser.
    - Run the following command from the frontend folder instead of running the build command (you still need to run the install command):
        - `>> npm run start`
    - **IMPORTANT: YOU MUST RUN THE BUILD COMMAND BEFORE DEPLOYING.**
    - The build command will build an optimised build which can be served efficiently by the backend server.
    - The start command is only intended for development purposes.

Restoring the Database

- First, follow the instructions above in order to install PostgreSQL locally on your machine.
- Make sure to create a default user with a username and password.
- In order to restore the database, you will need to obtain the latest database dump file from the following link: https://drive.google.com/drive/folders/1WgEzHfdlG9BTztFBqT2zLoqcup4Lfq0D?usp=sharing
    - The latest version of the dump file will have the largest number appended to the end. For example, identifire_6.dump is more recent than identifire_5.dump.
- Once you have downloaded the dump file and put it into a directory.
- Simply navigate to that directory and run the following command:
    - `>> psql -U your_username database_name < dump_file_name.dump`
    - Make sure to replace the values above with your own credentials and file names.

Preprocessing for prediction on bushfire

1. Install RScript free version - https://cran.r-project.org/bin/windows/base/
2. Install RStudio Desktop free version - https://rstudio.com/products/rstudio/download/#download
3. Open Preprocessing.r in Rstudio by clicking the file tab then open file options.
4. click 'Source' button which is next to the run button can be located on the right side.
5. When asked whether to install the package, click yes and it will run as it should.

Perform unit test for Preprocessing script

1. Install RScript free version - https://cran.r-project.org/bin/windows/base/
2. Install RStudio Desktop free version - https://rstudio.com/products/rstudio/download/#download
3. Open UnitTest_Preprocessing.r in Rstudio by clicking the file tab then open file options.
4. click the 'Run Tests' button which can be located on the right side.
5. When asked whether to install the package, click yes and it will run as it should.

Run Predictive modelling

1. Install Anaconda Python 3.8 64-Bit free version - https://www.anaconda.com/products/individual
    - Scroll down to see for Anaconda Python 3.8 64-Bit setup.
2. Open up Anaconda prompt after installation.
3. type cd on Anaconda prompt, for example cd C:\Project\Identifire\predictive-modelling for windows.
4. type run_prediction.bat on Anaconda prompt.
5. go to dataset folder which can be found within Identifire folder.
6. go to the ‘output’ folder.
7. bushfire_prediction.csv can be found here and it can be used to visualize in tableau.

Test essential function in predictive modelling

1. Install Anaconda Python 3.8 64-Bit free version - https://www.anaconda.com/products/individual
    - Scroll down to see for Anaconda Python 3.8 64-Bit setup
2. Open up Anaconda prompt after installation
3. type cd on Anaconda prompt, for example cd C:\Project\Identifire\predictive-modelling for windows
4. type run_prediction_test.bat on Anaconda prompt to run the predictive modelling.

Use the visualization for predictive modelling

1. Install Tableau desktop for staff/student - https://www.tableau.com/products/desktop/download
    - click the 'STUDENT OR TEACHER? GET A FREE 1-YEAR LICENSE. LEARN MORE'
2. Open up Tableau desktop app.
3. Load VisualizationBushfire.twb into Tableau desktop.
4. Tableau will show all visualization in making, feel free to do anything on the tableau.
5. Alternatively, you can skip step 1 to 4 if you only interested with visualization: https://public.tableau.com/profile/samael#!/vizhome/VisualizationBushfire/Bushfire_Story