

9/6/2024

Design Specification

Milestone 2

Table of Contents

Group Members	2
Project: Phenology Data Analytics of Aloe Ferox	3
1. Project Overview	3
2. System Overview	5
2.1. System Architecture.....	5
2.2. Functional Description.....	6
2.3. Design Constraints	6
2.4. API Integrations	7
3. Detailed Design	9
3.1. Component Design	9
3.2. Interface Design	9
3.3. Data Design.....	11
3.4. Power BI Design	18
2.1. Power BI DAX Code and Advanced Implementations.....	23
2.1.1. Setup for DAX Code in Power BI:	23
2.1.2. Detailed Explanation of Each DAX Code:	24
2.1.1. Putting it all together:	31
3. User Interface Design	39
3.1. UI Layout.....	39
3.2. UAX Prototyping.....	40
3.3. User Experience.....	41
3.4. Personas.	44
3.4.1. First Persona:	44
3.4.2. Second Persona:	45
4. Performance and Scalability	46
4.1. Performance Requirements.....	46
4.2. Scalability.....	46
5. Security Design	48
5.1. Security Requirements.....	48
5.2. Compliance	49
6. Testing and Validation	51
6.1. Test Plan.....	51
6.2. Validation Criteria.	53
7. Deployment and Maintenance	56
7.1. Deployment Plan.	56
7.2. Maintenance Plan.	59

Group Members

- Ajay Ramkelawan	577913@student.belgiumcampus.ac.za
- Reino Powell	576713@student.belgiumcampus.ac.za
- Reon Gottsche (Team Lead)	576949@student.belgiumcampus.ac.za
- Elias Modiga	576891@student.belgiumcampus.ac.za
- Andre Nienaber	576207@student.belgiumcampus.ac.za
- Johan Hattingh	574434@student.belgiumcampus.ac.za
- Jamie Frances Kench	577255@student.belgiumcampus.ac.za
- Thabiso Mokoena	1537@student.belgiumcampus.ac.za
- Tshepo Tsibolane	22143@student.belgiumcampus.ac.za

Project: Phenology Data Analytics of Aloe Ferox

Project Title: PID – Plant Identification Programme | Aloe Ferox

Project Date: June 03, 2024

Supervisor: Mr. Dino Giovannoni

Co-Supervisor: Mr. Phillip van Huyssteen.

Project Sponsor: Prof. Craig Peter, Department of Botany, Rhodes University

1. Project Overview

Objective:

- To use the provided dataset to characterise the phenology of *Aloe Ferox* based on the flowering stages of development.
- To use dates, seasonal and location data to identify the characteristics of the phenology.
- To calculate relevant phenological parameters like the average First Bloom Date (FBD), Last Bloom Date (LBD), Flowering Duration (FD), Peak Bloom Date, etc., for *Aloe Ferox* across the dataset in different geographic regions of the country.
- To use additional environmental weather data (e.g., temperature, rainfall, humidity, daylight hours, etc.) to explore potential correlations between these factors and flowering times. This could involve creating scatter plots or using correlation coefficients.
- To create maps visualising the spatial distribution of FBD or LBD across the observed regions. Heat maps or choropleth maps could be used.

Scope:

- Collecting images of *Aloe Ferox* from iNaturalist based on the identifier in the provided spreadsheet.
- Using the provided phenological data, together with weather and geographical data to provide a complete picture of the phenological characteristics of *Aloe Ferox*.
- Developing a suite of analysis tools to understand the temporal, geographic and environmental phenological characteristics of *Aloe Ferox*.
- Obtain weather data for the location of the images from the South African weather service, Google Earth Engine, or other sources.
- Using appropriate maps to superimpose location and other phenological data.
- Perform detailed data analytics to characterise the data's phenology. This could be used to identify the time of year for each stage of flowering, the relationship between various weather data for the various stages of development, and any geographical characteristics.

Key Deliverables:

- Design a web interface with an appropriate dashboard to visualise the Aloe Ferox flowering data and allow users to explore it easily.
- An interface that allows for the selection of the various phenological stages.
- Various types of data analysis providing scatter plots, geographic maps, correlation plots, etc., based on the phenological parameters.
- Calculate and display various phenological parameters as described in the objectives.
- An interface to show images illustrating the various phenological stages of development.

Tools & Techniques:

The following tools and techniques may be required for the execution of the project:

- **Data Analytics:** Developing suitable data analytics on datasets.
- **File I/O:** Importing and exporting of data files.
- **User Interface Design:** Developing appropriate applications with suitable user interfaces.

Stakeholders:

- **Project co-supervisor:** *Mr. Philip van Huyssteen*. Technology Aided Biocontrol Group (TAB), Belgium Campus Itiversity.
- **Project supervisor:** *Mr. Dino Giovannoni*. Technology Aided Biocontrol Group (TAB), Belgium Campus Itiversity.
- **Project sponsor:** *Prof. Craig Peter*. Department of Botany (RUBOT), Rhodes University.

2. System Overview

The system integrates weather and observation data to provide enriched environmental insights into *Aloe ferox* phenology. Using both the **Meteostat** and **iNaturalist** APIs, weather data is retrieved and added to an existing SQL database containing the plant's observation data, enhancing the phenological analysis. Additionally, the system integrates with web applications for real-time data updates using a FastAPI service.

Sections to consider:

- System Architecture
- Functional Description
- Design Constraints
- API Integrations

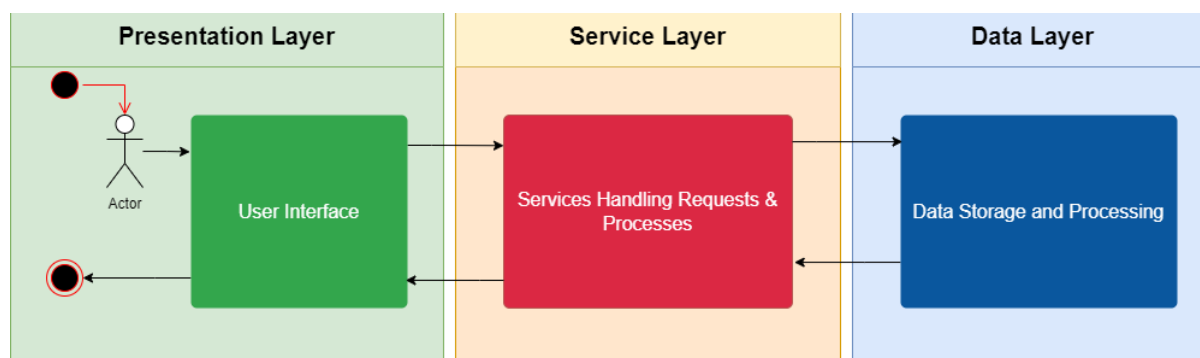
2.1. System Architecture

The system is designed as a modular architecture consisting of several components:

Component	Description
SQL Database	A central data store containing observation records (dates, latitude, longitude), media (images/sounds), location data, and weather data.
Meteostat API	Provides weather data for specific geographic coordinates and dates.
iNaturalist API	Provides phenological observation data for <i>Aloe ferox</i> , including flowering stages such as flowers, buds, and fruits.
Python Backend	Handles data extraction, transformation, and loading (ETL) operations, calling APIs, processing responses, and updating the SQL database.
FastAPI Service	A REST API service exposing endpoints to interact with the system for weather data retrieval and database updates.
Frontend Interface	JavaScript application interacting with FastAPI for real-time updates, visualising data through charts and maps generated by Power BI and other tools. HTML: Structure of the web pages. CSS: Styling the web pages.

The architecture uses a **three-layer model**:

1. **Data Layer:** Stores observation, location, media, and weather data in SQL.
2. **Service Layer:** FastAPI is a middleware that handles HTTP requests for weather data integration.
3. **Presentation Layer:** JavaScript interface interacting with FastAPI for data display and real-time updates.



2.2.Functional Description

The Aloe Ferox Phenology web application allows users to explore and analyse detailed data related to the flowering stages of Aloe Ferox across various geographic regions. The primary functionality includes data visualisation, data filtering, and interaction through a user-friendly interface.

Function	Description
Weather Data Retrieval	Retrieves weather data using the Meteostat API based on observation date and geographic coordinates.
Phenology Data Retrieval	Queries iNaturalist API for phenological data specific to Aloe ferox (flowering, budding, fruiting).
Data Processing	Processes retrieve, transform, and append data to the SQL database in the appropriate columns.
Database Updates	Updates the SQL database by adding new columns for weather data (temperature, precipitation, etc.) and appending the retrieved weather information to relevant rows.
FastAPI Integration	Exposes endpoints to retrieve weather data and update the database, allowing external applications to request real-time data updates.
Error Handling & Logging	Logs API failures, missing data, or processing errors, ensuring the system can recover gracefully.
Data Visualization	Power BI dashboards display enhanced observation data with the newly added weather information for analysis.
Scalability & Performance	Efficient processing of large datasets, handling API requests in batches to avoid timeouts, ensuring system reliability and scalability.

2.3.Design Constraints

The design of the Aloe Ferox Phenology web application is subject to the following constraints:

Constraint	Description
API Rate Limiting	Meteostat and iNaturalist APIs may impose rate limits on requests, requiring batching or throttling of requests to avoid timeouts and penalties.
Weather Data Availability	Meteostat API may not have historical weather data for specific locations or dates, which requires logging the absence of data without affecting other records.
Data Accuracy & Integrity	The system must ensure data integrity by accurately matching weather data with observation records, avoiding duplicate or mismatched entries in the database.
Performance with Large Datasets	The system must efficiently process and update thousands of records without introducing performance bottlenecks.
Modular Design for API Expansion	The architecture should be flexible enough to support additional APIs in the future, allowing for easy integration of more environmental or phenological datasets.

Security Considerations	The FastAPI service must be securely deployed, ensuring that sensitive data (e.g., database credentials) is protected during data processing and API calls.
Data Consistency	The database updates must ensure atomic operations, preventing partial updates that could corrupt the database or lead to inconsistent data.

2.4.API Integrations

The system integrates with two key APIs: **Meteostat** for weather data and **iNaturalist** for phenology data.

API	Purpose	Key Endpoints/Functions	Data Retrieved
Meteostat API	Retrieves weather data (temperature, precipitation, etc.) for a given location and date.	/stations/daily, /stations/hourly	<ul style="list-style-type: none"> - Temperature (average, minimum, maximum) - Precipitation - Snow depth - Wind direction & speed - Pressure - Sunshine duration
iNaturalist API	Retrieves observation and phenological data for Aloe ferox from the iNaturalist platform.	/observations, /phenology	<ul style="list-style-type: none"> - Phenological stages (flowering, budding, fruiting) - Observation timestamps - Location data (latitude, longitude) - User-submitted media (images, sounds)

Meteostat API Integration:

- **Objective:** Retrieve weather data based on each observation's geographic coordinates and date from the SQL database.
- **Process:**
 1. Retrieve coordinates and observation dates from SQL.
 2. Query Meteostat API using the Meteostat Python library.
 3. Append weather data to the corresponding observation records in the SQL database.

iNaturalist API Integration:

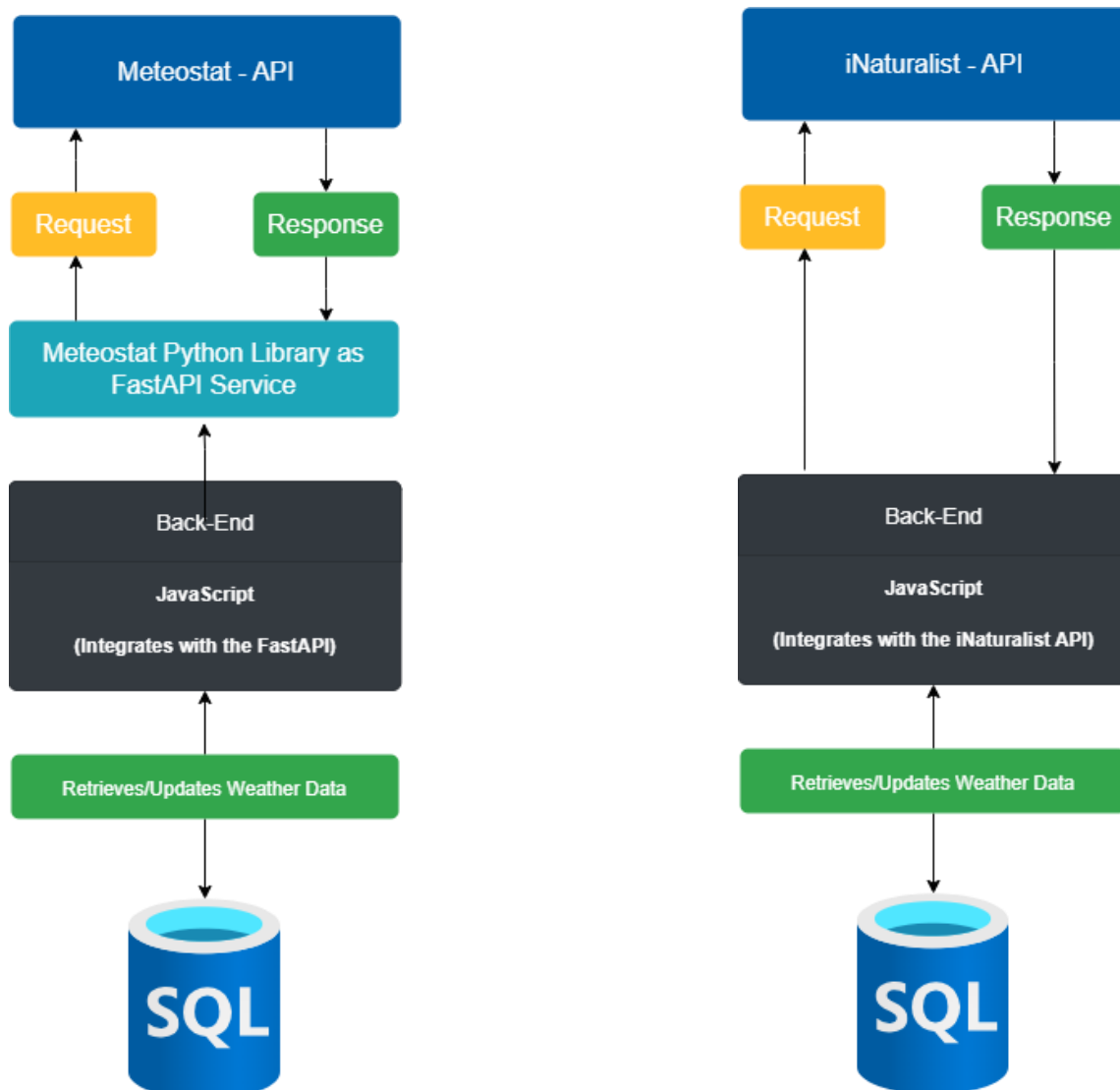
- **Objective:** Retrieve Aloe ferox sightings and phenological stages.
- **Process:**
 1. Query iNaturalist API for all sightings of Aloe ferox.
 2. Extract and process phenological data such as flower stages, fruits, and buds.
 3. Merge this data with existing observation records in the SQL database.

FastAPI Integration:

- **Objective:** Deploy a RESTful service to facilitate interaction between the JavaScript frontend and the backend weather/phenology data processing.
- **Endpoints:**
 - GET /weather-data: Retrieves weather data for specified coordinates and dates.
 - POST /update-db: Updates the SQL database with the retrieved weather data.

Frontend Integration:

- **Objective:** Allow real-time updates from the JavaScript frontend, which interacts with the FastAPI service.
- **Features:**
 1. In real-time, Users can view updated observation data, including weather information.
 2. Data is visualised using tools such as Power BI, showing geographic and temporal trends in Aloe ferox phenology.



3. Detailed Design

This section delves into the finer details of the system architecture, describing the design of each core component, the interfaces between components, how the data is structured, and the design of Power BI visualisations.

3.1.Component Design

The system consists of various key components, each designed with specific responsibilities to ensure seamless data integration from the APIs, processing, and updating the SQL database.

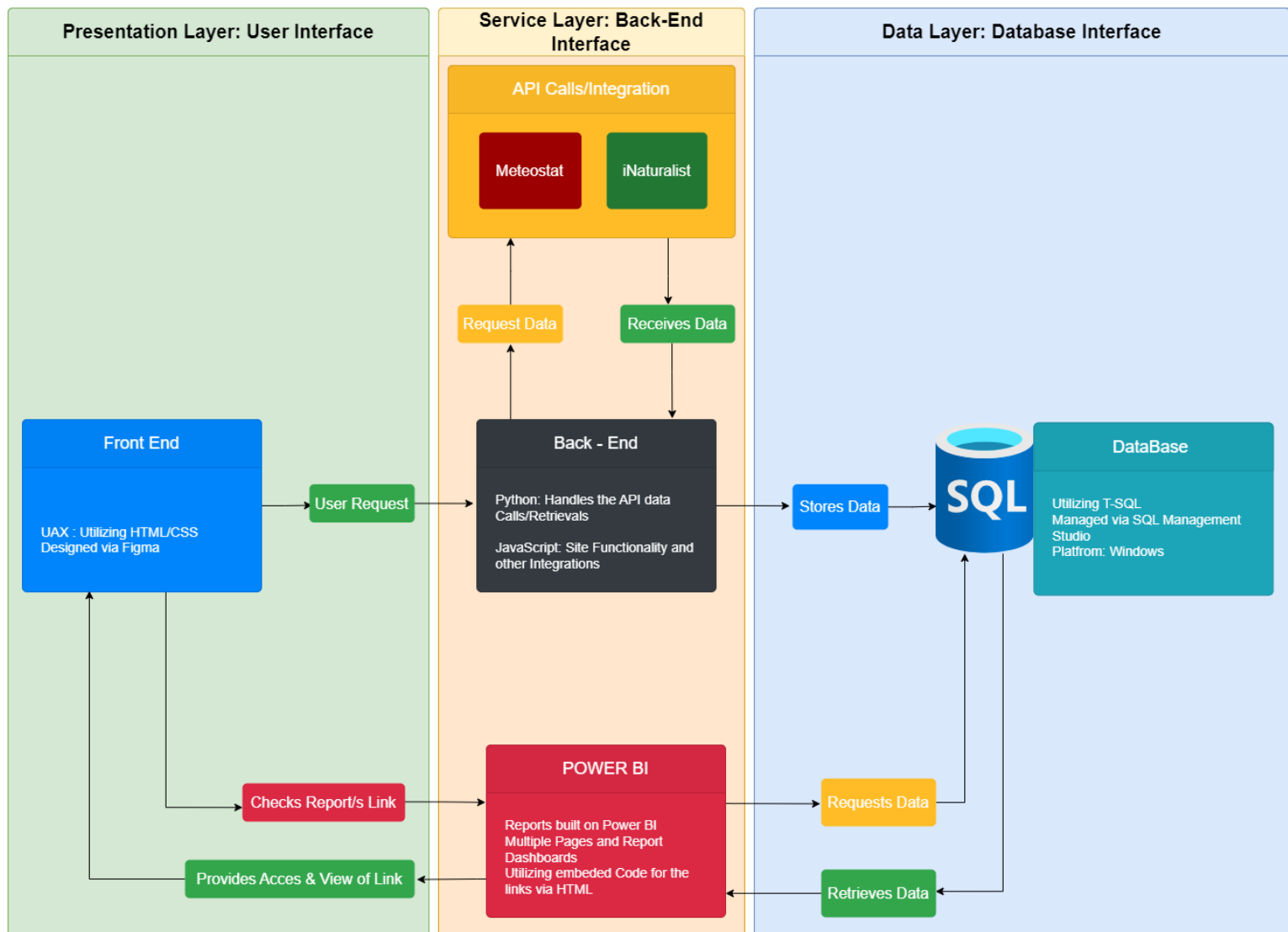
Component	Purpose	Key Methods/Functions
SQL Database	Central data storage for observation data (phenology, weather, locations), ensuring efficient data retrieval, storage, and updates.	- insert_data() - update_data() - query_data()
Meteostat API	Retrieves weather data for specific coordinates and dates, adding environmental context to the phenology data.	- get_weather_by_location() - batch_weather_retrieval()
iNaturalist API	Fetches phenology observations for Aloe ferox, providing real-time data on flowering stages across various locations.	- get_observations() - extract_phenology_data()
Data Processor	Responsible for ETL (extract, transform, load) operations that process API data before updating the database.	- process_weather_data() - process_phenology_data()
FastAPI Service	RESTful API service provides endpoints for real-time interaction between the front end and the back end.	- GET /weather-data - POST /update-db
Frontend Interface	JavaScript-based interface allows users to visualise data, trigger data updates, and view insights in real-time.	- fetch_data() - update_visualization()
Power BI Dashboards	Power BI creates visualisations that represent trends in phenology and weather data, along with spatial and temporal patterns.	- create_visualization() - filter_by_date()

3.2.Interface Design

The system employs multiple interfaces to communicate between layers, enabling seamless data integration and visualisation. The interfaces have been designed to be modular and flexible to support future extensions and API integrations.

Interface	Description	Endpoints/Methods
SQL Database Interface	This interface defines the database's methods for inserting, updating, and querying. It ensures that the weather and observation data are consistently stored and retrieved.	- insert_observation(observation_data) - update_weather(weather_data)
Meteostat API Interface	It provides a set of functions that interact with the Meteostat API. It sends requests to retrieve weather data based on location (latitude/longitude) and observation dates, ensuring data is processed and ready for database insertion.	- get_weather_data(coordinates, date_range) - parse_weather_response()
iNaturalist API Interface	Defines the interaction with iNaturalist to fetch Aloe ferox observations and phenological data. The interface ensures that the correct observation metadata (e.g., timestamps and locations) is fetched and processed.	- get_observation_data() - extract_phenology_details()
FastAPI Service Interface	The API service interface exposes endpoints for the front end to request weather data and trigger database updates. It	- GET /weather-data - POST /update-db

	allows real-time communication and enables future API expansions.	
Frontend (JavaScript) Interface	Defines the communication methods between the FastAPI backend and the JavaScript frontend, ensuring seamless retrieval of data and triggering visualisation updates.	<ul style="list-style-type: none"> - fetch_weather_data() - fetch_observation_data() - update_visualization_with_new_data()
Power BI Interface	Provides a set of configurations to retrieve data from the SQL database and integrate it into Power BI reports. It enables the use of filters, cross-referencing data, and the generation of different visualisations (scatter plots, heat maps, etc.).	<ul style="list-style-type: none"> - refresh_data_source() - apply_filter_by_location() - generate_scatter_plot()



3.3.Data Design

The data design focuses on structuring the data model within the SQL database to support efficient storage, querying, and analysis of phenological observations and weather data.

Data Entity	Description	Fields
Observation Data	Contains records of phenological observations, including the species, location, date, and observed phenology stage (e.g., flowering, budding).	<ul style="list-style-type: none"> - observation_id (Primary Key) - species - phenology_stage - latitude - longitude - observed_on - image_url
Weather Data	Stores the weather data for specific geographic locations (latitude/longitude) and observation dates. This is used to correlate phenological stages with environmental factors.	<ul style="list-style-type: none"> - weather_id (Primary Key) - latitude - longitude - date_observed - temperature_min - temperature_max - precipitation - wind_speed
User Data	Contains information about users who contribute to the iNaturalist platform by uploading observations. This data helps identify contribution patterns and can be used for deeper analysis of observer trends.	<ul style="list-style-type: none"> - user_id (Primary Key) - user_name - login - created_at - location
Location Data	Records the geographic coordinates and the place names associated with observations. This helps generate maps and conduct spatial analysis in Power BI.	<ul style="list-style-type: none"> - location_id (Primary Key) - latitude - longitude - town - county - state - country
Phenology Data	Contains metadata for phenological stages (flowering, budding, etc.) of Aloe ferox, linking the observation to the phenological state. This data is key to understanding the life cycle patterns of the plant.	<ul style="list-style-type: none"> - phenology_id (Primary Key) - observation_id (Foreign Key) - phenology_stage - date_stage_observed

Data Cleanup

Data cleanup involves identifying and correcting errors or inconsistencies in your data. It is essential to ensure data accuracy and reliability before proceeding with database operations.

Step 1: Identify Data Quality Issues

- **Missing Data:** Check for missing values in essential fields.
- **Duplicate Records:** Identify and remove any duplicate records.
- **Inconsistent Formats:** Ensure consistent data entries (e.g., date formats, capitalisation).
- **Incorrect Data:** Look for data entries that do not make logical sense (e.g., a negative value for temperature).

Step 2: Cleaning Data

- **Handling Missing Data:**
 - Imputation: Replace missing values with a calculated value (e.g., mean, median).
 - Deletion: Remove records with missing values if they are not critical.
- **Removing Duplicates:**
 - Use SQL queries to identify duplicate records.
 - Delete duplicates after review.
- **Standardizing Formats:**
 - Ensure dates are in a consistent format.
 - Standardize text fields (e.g., ensuring all country names are in uppercase).
- **Validating Data:**
 - Ensure all latitude values fall within the range of -90 to 90 and longitude within -180 to 180

Data Preparation

Data preparation involves transforming raw data into a suitable format for analysis or database storage, including ensuring the data conforms to the database schema.

Step 1: Data Transformation

- **Data Type Conversion:** Convert data types to match the database schema.
- **Derived Columns:** If needed, create new columns based on existing data. For example, if you need a full location description

Step 2: Data Import

- **Batch Import:** Use SQL Server's BULK INSERT or similar tools to import large datasets efficiently, ensuring the data matches the schema.
- **Validation:** After import, validate the data to ensure it aligns with your database schema and constraints.

Normalisation

Normalisation is organising data within the database to reduce redundancy and improve data integrity. The goal is to ensure that each piece of data is stored in only one place and that relationships between tables are well-defined.

Step 1: First Normal Form (1NF)

- **Ensure Atomicity:** Each column should contain atomic (indivisible) values. For example, avoid storing multiple pieces of data in a single column (e.g., storing city and state together).
- **Unique Rows:** Ensure each table has a primary key uniquely identifying each row.

Step 2: Second Normal Form (2NF)

- **Remove Partial Dependencies:** Ensure that non-key attributes fully depend on the primary key. For example, if you have a Location table, the latitude and longitude should depend on the location_id, not partially on observation_id.

Step 3: Third Normal Form (3NF)

- **Remove Transitive Dependencies:** Ensure that non-key attributes depend only on the primary key. For instance, in the Location table, the place_county_name should not depend on place_state_name; instead, both should depend directly on the location_id.

Step 4: Beyond 3NF (Optional)

- **Boyce-Codd Normal Form (BCNF):** This is a more robust version of 3NF. Ensure that X is a superkey for every functional dependency $X \rightarrow Y$ (i.e., no partial or transitive dependencies).

Understanding the Requirements

Before setting up the database, you must understand the entities (tables) that will store your data, their attributes (columns), and how they relate. Based on the provided requirements, the following entities are identified:

Observation: Stores the primary data of each observation.

Media: Stores URLs of media (images, sounds) associated with observations.

Location: Stores geographic information related to each observation.

Weather: Stores weather-related data for each observation.

Designing the ERD (Entity-Relationship Diagram)

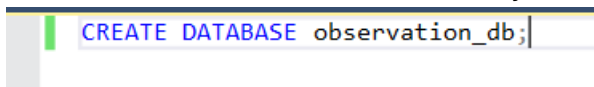
An ERD visually represents the structure of your database. It shows how different entities (tables) are connected. Here is how to design the ERD for our scenario:

- **Entities:**
 - **Observation:** The central entity that other entities relate to.
 - **Media:** Related to the Observation by a foreign key.
 - **Location:** Stores details like latitude, longitude, and place names linked to Observation by a foreign key.
 - **Weather:** Holds weather data linked to the Observation by a foreign key.
- **Relationships:**
 - **Observation to Media:** One-to-Many (One observation can have multiple media items).
 - **Observation to Location:** One-to-one or One-to-Many (Depending on whether each observation is in one or multiple locations).
 - **Observation to Weather:** One-to-One (Each observation corresponds to one weather record).

Setting Up the Database in SQL Server

Step 1: Create the Database

- You first need to create a database to store your tables



```
CREATE DATABASE observation_db;
```

Step 2: Create the Tables

- Create each table with its respective columns and relationships (foreign keys)

```

IF NOT EXISTS (SELECT * FROM sys.databases WHERE name = 'observation_db')
BEGIN
    CREATE DATABASE observation_db;
END;

USE observation_db;

CREATE TABLE Phenology (
    stage_id INT PRIMARY KEY,
    phenology_stage VARCHAR(255) NOT NULL
);

CREATE TABLE Observation (
    observation_id INT IDENTITY(1,1) PRIMARY KEY,
    stage_id INT FOREIGN KEY REFERENCES Phenology(stage_id),
    observed_on DATE NOT NULL,
    time_observed_at TIME NOT NULL,
    time_zone VARCHAR(50) NOT NULL,
    user_login VARCHAR(100) NOT NULL,
    user_name VARCHAR(100) NOT NULL,
    num_identification_agreements INT NOT NULL
);

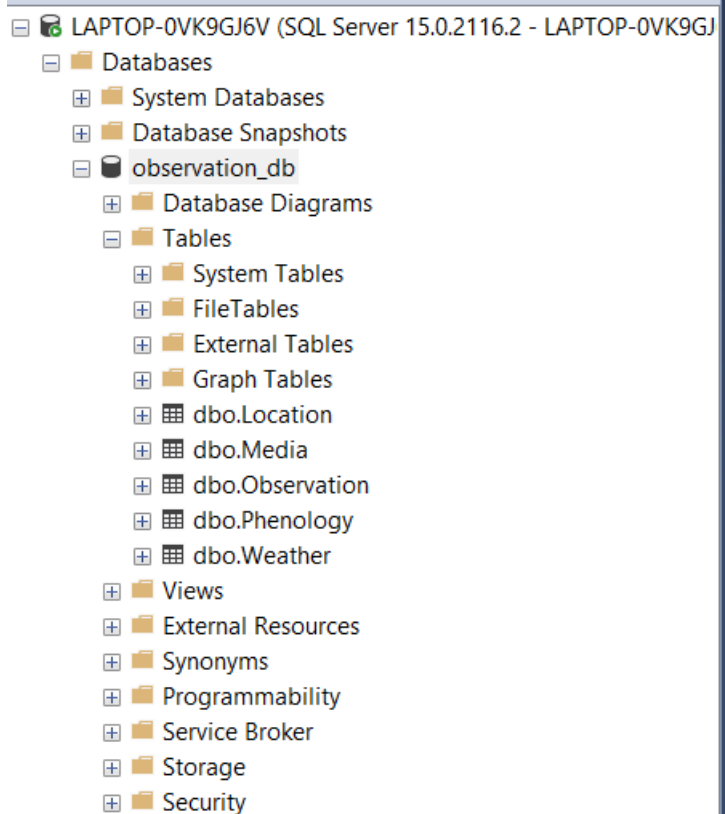
CREATE TABLE Media (
    media_id INT IDENTITY(1,1) PRIMARY KEY,
    observation_id INT FOREIGN KEY REFERENCES Observation(observation_id),
    url VARCHAR(255) NOT NULL
);

CREATE TABLE Location (
    location_id INT IDENTITY(1,1) PRIMARY KEY,
    observation_id INT FOREIGN KEY REFERENCES Observation(observation_id),
    place_guess VARCHAR(255),
    latitude DECIMAL(10, 8),
    longitude DECIMAL(11, 8),
    place_county_name VARCHAR(100),
    place_state_name VARCHAR(100),
    place_country_name VARCHAR(100)
);

CREATE TABLE Weather (
    weather_id INT IDENTITY(1,1) PRIMARY KEY,
    observation_id INT FOREIGN KEY REFERENCES Observation(observation_id),
    tavg DECIMAL(5, 2), -- Average temperature (°C)
    tmin DECIMAL(5, 2), -- Minimum temperature (°C)
    tmax DECIMAL(5, 2), -- Maximum temperature (°C)
    prcp DECIMAL(5, 2), -- Precipitation (mm)
    snow DECIMAL(5, 2), -- Snowfall (mm)
    wdir DECIMAL(5, 2), -- Wind direction (degrees)
    wspd DECIMAL(5, 2), -- Wind speed (m/s)
    wpgt DECIMAL(5, 2), -- Wind gust (m/s)
    pres DECIMAL(7, 2), -- Atmospheric pressure (hPa)
    tsun DECIMAL(7, 2) -- Sunshine duration (minutes)
);

```

After creating the database and tables, you should see something like the following in your object explorer:



Creating a View for Consolidated Data

A view is a virtual table consolidating data from multiple tables into a single table-like structure. Here's how to create a view that combines data from **Observation**, **Media**, **Location**, and **Weather** tables.


```

CREATE VIEW observation_results AS
SELECT
    o.observation_id,
    o.observed_on,
    o.time_observed_at,
    o.time_zone,
    o.user_login,
    o.user_name,
    m.url,
    o.num_identification_agreements,
    l.place_guess,
    l.latitude,
    l.longitude,
    l.place_county_name,
    l.place_state_name,
    l.place_country_name,
    w.tavg,
    w.tmin,
    w.tmax,
    w.prcp,
    w.snow,
    w.wdir,
    w.wspd,
    w.wpgt,
    w.pres,
    w.tsun,
    p.phenology_stage
FROM
    Observation o
LEFT JOIN
    Media m ON o.observation_id = m.observation_id
LEFT JOIN
    Location l ON o.observation_id = l.observation_id
LEFT JOIN
    Weather w ON o.observation_id = w.observation_id
LEFT JOIN
    Phenology p ON p.stage_id = o.stage_id;

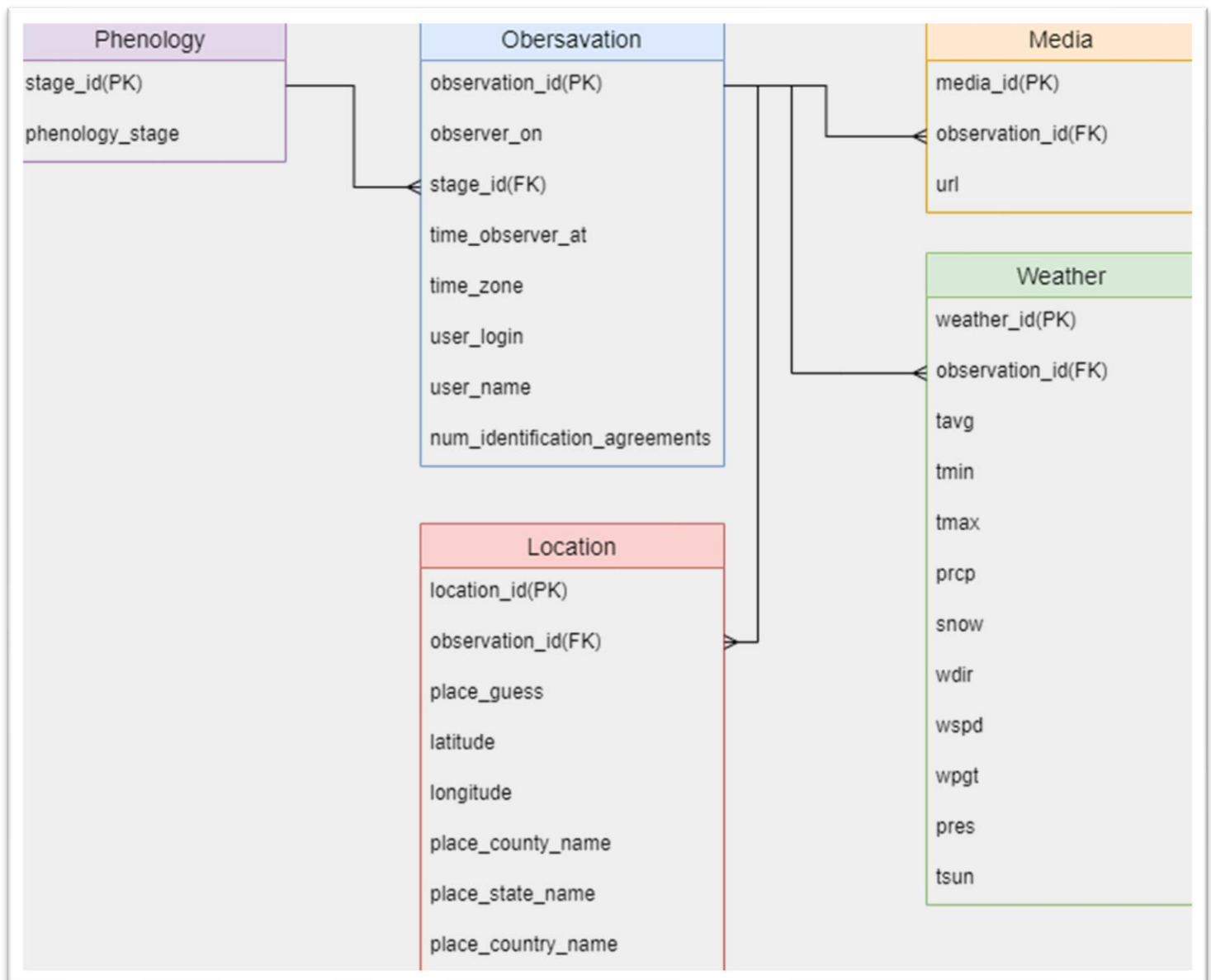
```

After creating the View, it should reflect under the Views directory in your object explorer.

Refining the ERD

After setting up the database and testing with data, refine the ERD to reflect any changes or improvements in your design. Tools like MySQL Workbench, Microsoft Visio, or online ERD tools can help you visualize and edit your ERD.

The following is an initial design of our ERD:

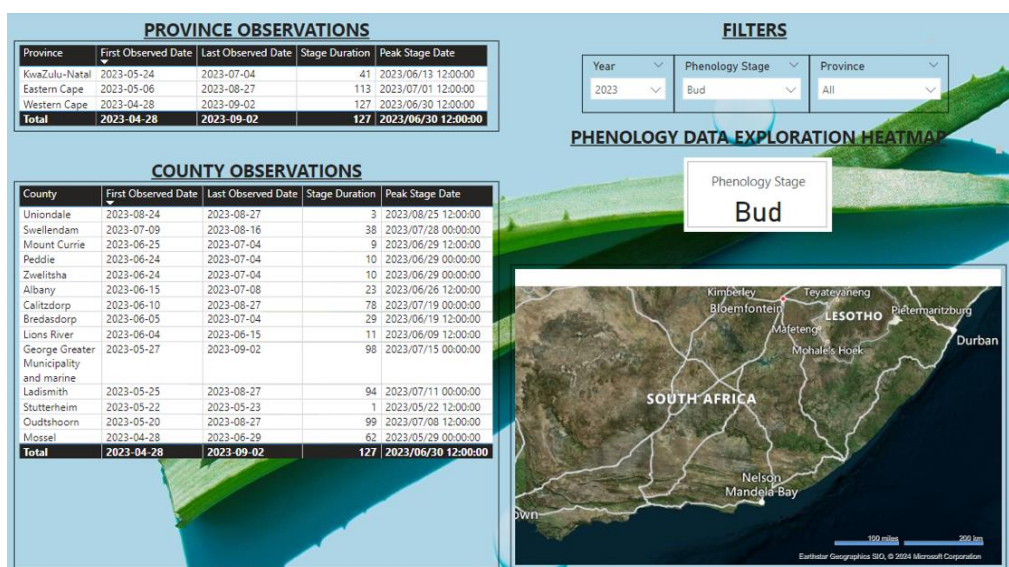
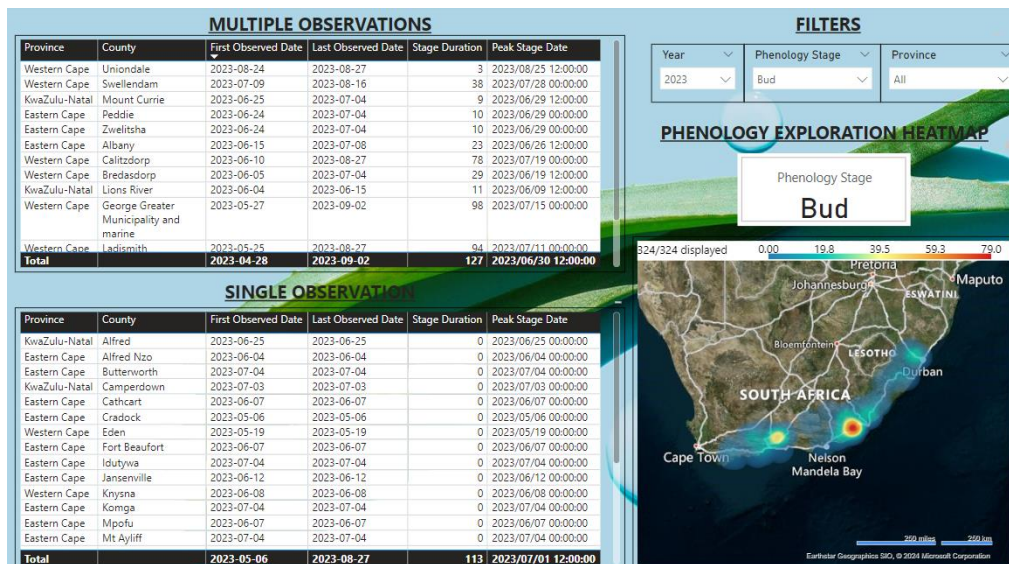
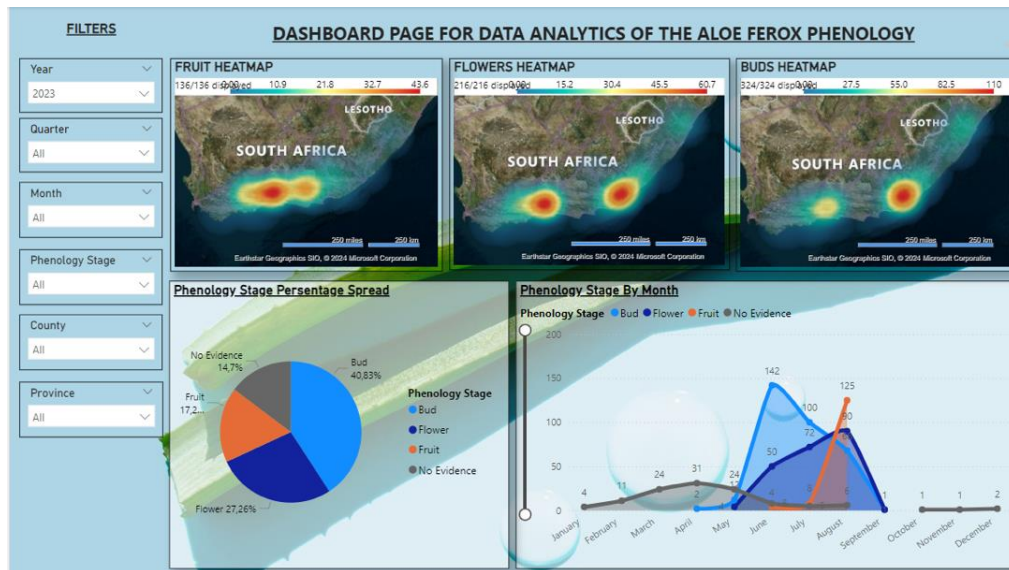


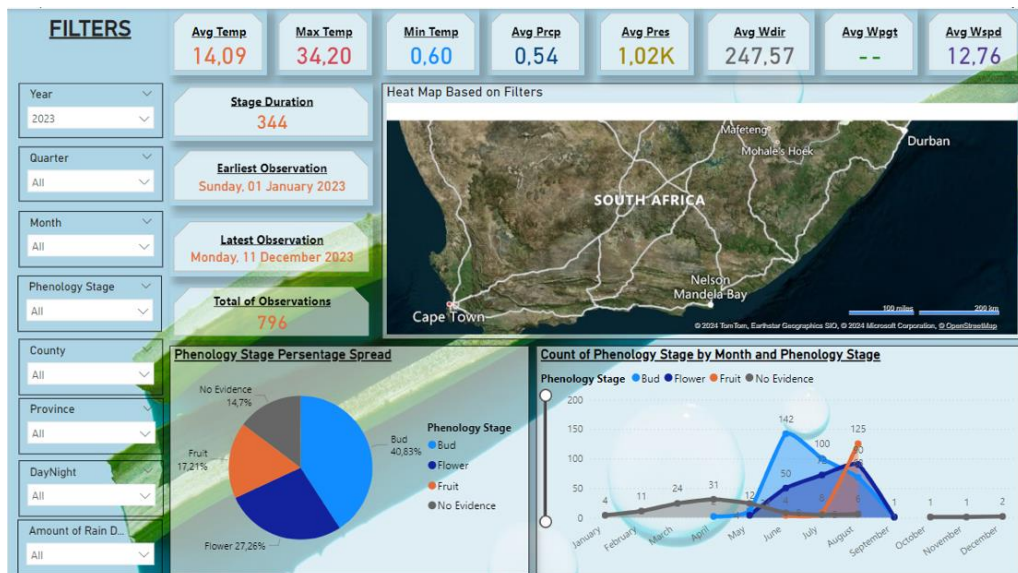
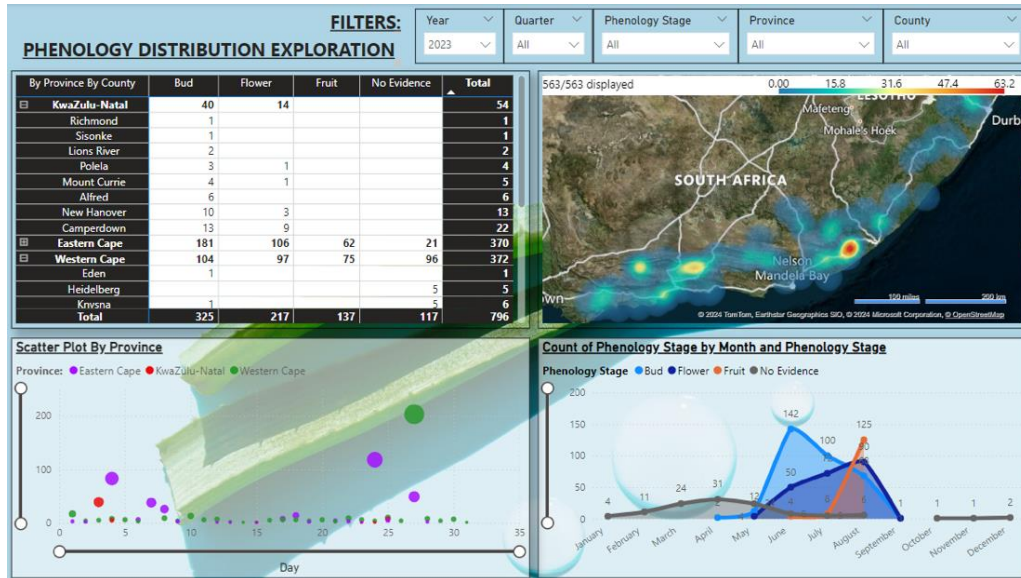
3.4. Power BI Design

Power BI visualizes and explores the relationships between phenological data and environmental factors like temperature, precipitation, and location. The Power BI reports and dashboards design is focused on intuitive data representation through charts, maps, and time-series analyses.

Power BI Feature	Description	Design Elements
Scatter Plot	Visualises correlations between weather factors (e.g., temperature) and phenology stages (e.g., flowering, budding), helping to identify patterns.	<ul style="list-style-type: none"> - X-Axis: Date of Observation - Y-Axis: Temperature (Min/Max) - Color-coding based on the Phenology Stage (Flowering, Budding, Fruiting)
Heatmap	Displays the spatial distribution of phenology stages and weather data, providing a geographic view of where certain stages occur more frequently.	<ul style="list-style-type: none"> - Layers: Geographic boundaries (town, state, country) - Color Gradient: Frequency of flowering or budding across regions
Time-Series Line Graph	Plots trends over time, showing specific phenology stages and weather conditions. This provides a clear picture of how environmental factors affect phenology.	<ul style="list-style-type: none"> - X-Axis: Time (Months, Seasons) - Y-Axis: Number of Observations - Line Colors: Different phenology stages
Dashboard Filters	Allows users to filter data by date, location, or phenology stage, enabling dynamic updates of visualisations based on specific interests or hypotheses.	<ul style="list-style-type: none"> - Filter Controls: Dropdown for Date Range - Toggle Buttons: Filter by Region - Checkbox: Select specific Phenology Stage
Interactive Reports	Enables users to drill down into specific regions or phenology stages by clicking on elements in the charts or maps. Provides real-time interaction with the underlying data.	<ul style="list-style-type: none"> - Clickable Regions: Zoom into specific areas on the map - Detailed Observation Panel: Show detailed data for specific observations on the side panel

We have done the following as a basic dashboard and a few extra pages from POWER-BI from a prototype perspective.





FILTERS

Avg Temp: 14.09, Max Temp: 34.20, Min Temp: 0.60, Avg Prec: 0.54, Avg Pres: 1,02K, Avg Wdir: 247.57, Avg Wpgt: --, Avg Wspd: 12.76

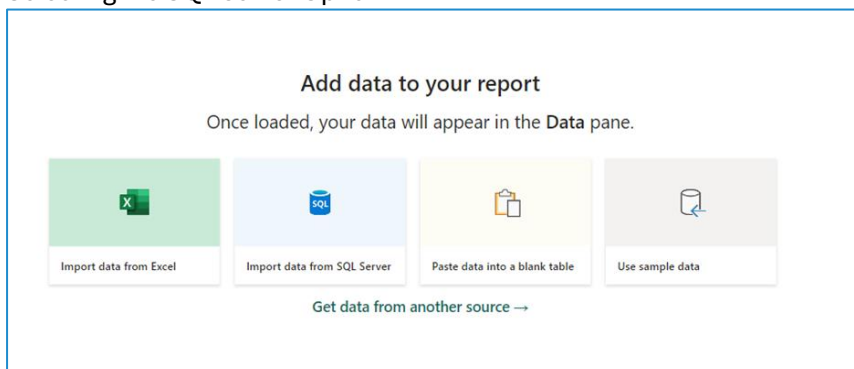
Year: 2023, Quarter: All, Month: All, Phenology Stage: All, County: All, Province: All, Day/Night: All, Amount of Rain D.: All

Stage Duration: 344
Earliest Observation: Sunday, 01 January 2023
Latest Observation: Monday, 11 December 2023
Total of Observations: 796

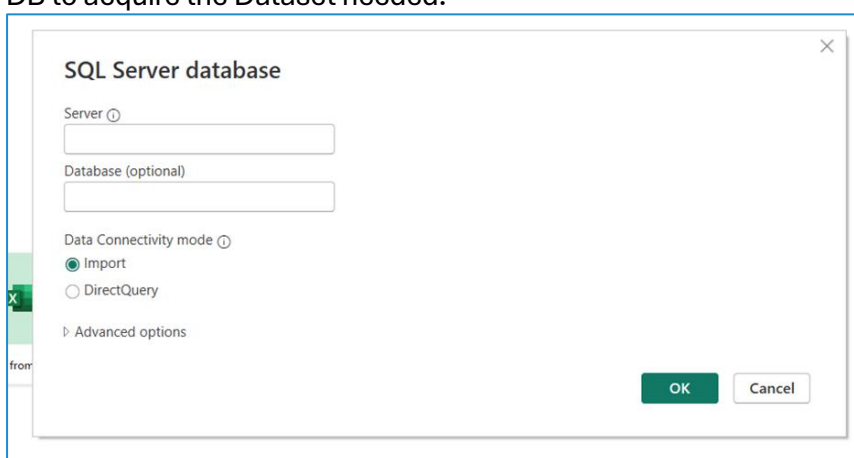
Province	County	Phenology Stage	Stage Duration	Avg Temp	Max Temp	Min Temp	Avg Prec	Avg Wdir	Avg Wspd	Avg Pres	Temp Description
Eastern Cape	Albany	Bud	23		23.40		3.10	246.22	17.41	1 018.77	Cold
Eastern Cape	Albany	Flower	14								Cold
Eastern Cape	Albany	Flower	44	11.21	24.80	5.70	1.20	254.45	19.45	1 016.02	Warm
Eastern Cape	Albany	Fruit	52	12.07	19.80	6.80	0.50	242.00	14.27	1 024.43	Warm
Eastern Cape	Albany	No Evidence	11	18.22	30.90	10.40	0.00	294.60	14.36	1 013.62	Warm
Western Cape	Bredasdorp	Bud	29		15.60	9.20		259.00	22.85	1 024.65	Cold
Western Cape	Bredasdorp	No Evidence	23	21.58	26.20	15.50	0.13	107.00	22.85	1 014.85	Hot
Western Cape	Bredasdorp	No Evidence	88	14.46	24.10	7.90	0.00	191.00	15.94	1 017.27	Warm
Western Cape	Calitzdorp	Bud	78								Cold
Eastern Cape	Cradock	No Evidence	1	14.60	27.80	7.20	0.70	226.50	15.35	1 018.75	Warm
Western Cape	George Greater Municipality and marine	Bud	98		21.20	4.10		188.14	10.11	1 017.93	Cold
Western Cape	George Greater Municipality and marine	Flower	90								Cold
Western Cape	George Greater Municipality and marine	No Evidence	271								Cold
Western Cape	George Greater Municipality and marine	No Evidence	109	14.79	22.00	6.00	1.67	175.09	10.63	1 019.07	Warm
Western Cape	Heidelberg	No Evidence	222								Cold
Western Cape	Ladismith	Bud	94								Cold
Western Cape	Ladismith	Flower	36								Cold
Western Cape	Ladismith	Fruit	35								Cold
Total			342	14.80	30.90	0.60	0.51	232.20	13.74	1 018.98	

Steps of the Import of the SQL SB dataset

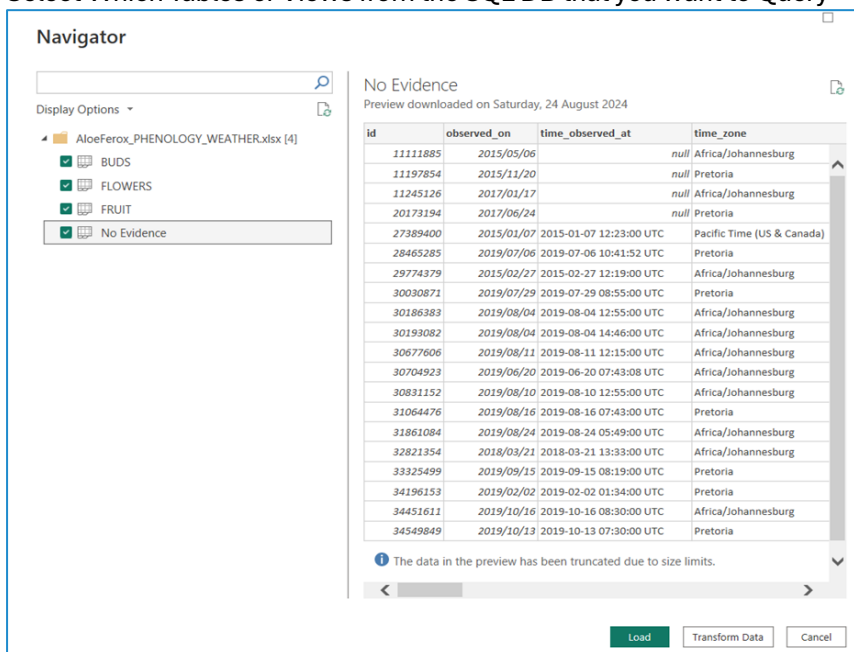
1. Selecting the SQL server Option



2. Pop up where you specify your Server and Database Details to Query the tables of the DB to acquire the Dataset needed.



3. Select Which Tables or Views from the SQL DB that you want to Query



4. Transform Option of the Selected Tables to format the Data even further if required.

[illegible]

5. Recommendations on our Solution that we have implemented beyond this step.

- Combined tables needed to have all data in one table from a Power BI perspective by using the Append Queries Option and selecting the appropriate tables we wanted to use.
- Renaming columns to more appropriate names for easier reporting and readability.
- Change the data types of specific columns to appropriate data types where needed.
- Reordering of Columns where required. See the screenshot below of our Applied Steps in Power BI as an example.

Query Settings

✕

▲

PROPERTIES

Name

BUDS

All Properties

▲

APPLIED STEPS

Source

Navigation

Promoted Headers

Changed Type

Added Custom

Reordered Columns

✕ Renamed Columns

The report design would depend on how the developers/client/s determine their requirements.

2.1. Power BI DAX Code and Advanced Implementations

Depending on the context, in this case, the context would be to display basic weather data linked to a specific Phenomenological site. Where we display the:

- Average Temperature in degrees Celsius
- What the temperature implies “Hot,” “Cold” or “Warm.”
- Show the type of Rain received on a specific date.
- The strength or type of the Wind.
- The time of day.
- And finally, the clarity of that day.

Average Temperature	Minimum Temp	Maximum Temp(tmax)	Maximum Precipitation(prcp)	Wind Direction(wdir)	Wind Speed(wspd)	Pressure(pres)
13,4	11,2	15	32	250	17,4	1020,3
10,4	8,2	15,2	26,9	319	10,2	1027,3
10,8	7,2	12,8	15	314	14,4	1022
10,8	7,2	12,8	15	314	14,4	1022
16,3	13,8	19,4	13	252	23,3	1027
18,1	15,4	22,5	13	225	23,3	1019,4
11,8	10,4	14,2	11,9	351	8,7	1020,3
18,3	13	21,4	11,9	0	13,7	0
9,4	6,4	12,8	9,9	317	15,5	1017
14,3	12,3	16,8	9,9	250	9,8	1021,8
7,6	5,6	11,4	9,9	0	0	0
14,1	10,2	19,2	8,4	84	11,5	0
12,2	3,8	20,4	7,9	347	13,1	1020,1
12,2	3,8	20,4	7,9	347	13,1	1020,1
12,2	3,8	20,4	7,9	347	13,1	1020,1
12,2	3,8	20,4	7,9	347	13,1	1020,1
12,2	3,8	20,4	7,9	347	13,1	1020,1
12,2	3,8	20,4	7,9	347	13,1	1020,1

2.1.1. Setup for DAX Code in Power BI:

Before writing DAX code, ensure you have the necessary data loaded into Power BI, such as weather data, time information, and other relevant fields. Follow these steps to set up the environment:

- **Step 1: Load Data**

Import the dataset into Power BI using Get Data. Ensure that you have fields like:

- [Average Temperature(tavg)]
- [Maximum Precipitation(prcp)]
- [Wind Speed(wspd)]
- [Pressure(pres)]
- [Hour] (from the time or datetime column)

- **Step 2: Create New Columns and Measures**

Use the Data or Report view in Power BI to create new columns or measures:

- **Columns** are calculated for each row in your data model, while **Measures** are dynamic calculations that respond to user interactions and filters in reports.

- **Step 3: Write DAX Code**

Use DAX expressions to create calculated columns or measures. You can enter the DAX code by selecting "Modelling" from the ribbon and then "New Column" or "New Measure."

Now, let's go through a **detailed explanation** of each DAX formula provided earlier.

2.1.2. Detailed Explanation of Each DAX Code:

Degrees Celsius Formatting:

```
DegreesCelsius = [Average Temperature(tavg)] & " °C"
```

- **Purpose:** Converts the numerical value of [Average Temperature(tavg)] to a string format and appends " °C" to display the temperature in Celsius.
- **Type:** Calculated Column
- **Usage:** Useful for visualising temperature data in a more user-friendly format.

DegreesCelsius
12.2 °C
12.2 °C
12.2 °C
12.2 °C
12.2 °C
12.2 °C
12.2 °C
12.2 °C
12.2 °C
15.2 °C
15.2 °C
8.1 °C
10.6 °C
10.6 °C

Temperature Description Categorization:

```
Temperature Description =  
SWITCH(  
    TRUE(),  
    [Average Temperature(tavg)] >= 20, "Hot",  
    [Average Temperature(tavg)] >= 10, "Warm",  
    [Average Temperature(tavg)] > 0, "Cool",  
    "Cold"  
)
```

- **Purpose:** Categorizes temperature values into "Hot," "Warm," "Cool," or "Cold" based on defined thresholds.
- **Type:** Calculated Column
- **Logic:** The SWITCH function evaluates multiple conditions:
 - If the average temperature is 20°C or above, it is labelled as "Hot."
 - If the temperature is between 10°C and 19.9°C, it is "Warm."
 - If it is above 0°C but below 10°C, it is "Cool."

- Otherwise, it is "Cold."

Temperature
Warm
Warm
Warm
Warm
Warm
Warm
Warm
Warm
Warm
Warm
Warm
Cool

Rainfall Description Categorization:

```
Amount of Rain Description =
SWITCH(
  TRUE(),
  [Maximum Precipitation(prcp)] <= 2.5, "Light Rainfall",
  [Maximum Precipitation(prcp)] <= 7.5, "Moderate Rainfall",
  [Maximum Precipitation(prcp)] <= 50, "Heavy Rainfall",
  [Maximum Precipitation(prcp)] <= 100, "Very Heavy Rainfall",
  [Maximum Precipitation(prcp)] > 100, "Extreme Rainfall",
  "No Data"
)
```

- **Purpose:** A textual description of the rainfall amount can be used for visual representation or decision-making processes.
- **Type:** Calculated Column
- **Logic:** Like temperature categorisation, SWITCH is used to label different levels of rainfall:
 - "Light Rainfall" for precipitation ≤ 2.5 mm.
 - "Moderate Rainfall" for precipitation > 2.5 mm and ≤ 7.5 mm.
 - And so on, up to "Extreme Rainfall" for precipitation > 100 mm.

[illegible]

Day or Night Determination:

```
Day_Night =  
IF(  
    [TimeFormatted] >= TIME(4, 0, 0) && [TimeFormatted] <= TIME(18, 0, 0),  
    "Daytime",  
    "Nighttime"  
)
```

- **Purpose:** Determines whether the time is during the day or night.
- **Type:** Calculated Column
- **Logic:** The IF function checks whether the time is between 4 AM and 6 PM:
 - If true, it returns "Daytime."
 - Otherwise, it returns "Nighttime."

[illegible]

Image Keys for Temperature and Day/Night:

```
TempImagekey = [Temperature Description] & ".png"
DNImagekey = [Day_Night] & ".png"
```

- **Purpose:** Concatenates the text descriptions with ".png" to create an image file key that can be used to display relevant images in reports.
- **Type:** Calculated Column
- **Logic:** Simple concatenation of text fields with the ".png" extension.

DNImagekey
Daytime.png
Daytime.png
Daytime.png
Daytime.png
Daytime.png
Daytime.png
Daytime.png
Nighttime.png

Greetings based on Time of Day:

```
Greeting =
SWITCH(
  TRUE(),
  [hour] >= 6 && [hour] <= 12, "Good Morning",
  [hour] > 12 && [hour] <= 17, "Good Afternoon",
  [hour] > 17 && [hour] <= 21, "Good Evening",
  "Good Night"
)
```

- **Purpose:** Provides a greeting based on the time of day.
- **Type:** Calculated Column
- **Logic:** Uses SWITCH to determine the appropriate greeting:
 - Morning: 6 AM to 12 PM.
 - Afternoon: 12 PM to 5 PM.
 - Evening: 5 PM to 9 PM.
 - Night: Otherwise.

Greeting
Good Morning
Good Afternoon
Good Morning
Good Morning
Good Afternoon
Good Morning
Good Morning
Good Night

Wind Speed Description:

```
WindSpeedDescription =
SWITCH(
    TRUE(),
    'FLOWERS'[Wind Speed(wspd)] >= 55, "Severe Gale",
    'FLOWERS'[Wind Speed(wspd)] >= 40, "Very Windy/Strong Winds",
    'FLOWERS'[Wind Speed(wspd)] >= 25, "Windy",
    'FLOWERS'[Wind Speed(wspd)] >= 15, "Moderate Wind",
    "Light Wind"
)
```

- **Purpose:** Categorizes wind speeds into different descriptive labels.
- **Type:** Calculated Column
- **Logic:** Like the previous SWITCH examples, this categorises based on wind speed thresholds.

Cloud Prediction Based on Pressure:

```
CloudyPrediction =
SWITCH(
    TRUE(),
    'FLOWERS'[Pressure(pres)] < 1010, "Cloudy",
    'FLOWERS'[Pressure(pres)] >= 1010 && 'FLOWERS'[Pressure(pres)] <= 1020, "Partly Cloudy"
    'FLOWERS'[Pressure(pres)] > 1020, "Clear"
)
```

- **Purpose:** Predicts cloudiness based on atmospheric pressure.
- **Type:** Calculated Column
- **Logic:** Uses atmospheric pressure ranges to predict cloud conditions.

CloudyPrediction
Clear
Clear
Clear
Clear
Clear
Partly Cloudy
Clear
Cloudy
Partly Cloudy

Weather Icon Determination:

```

WeatherIcon =
VAR WindSpeed = 'FLOWERS'[Wind Speed(wspd)]
VAR MaxPrecipitation = 'FLOWERS'[Maximum Precipitation(prcp)]
VAR RainDescription = 'FLOWERS'[Amount of Rain Description]
VAR DayNightInd = 'FLOWERS'[Day_Night]
VAR CloudyPrediction = 'FLOWERS'[CloudyPrediction]

RETURN
SWITCH(
    TRUE(),
    // Wind and Rain Scenario
    WindSpeed > 0 && MaxPrecipitation > 0, "wind and rain.png",

    // No Wind, Rain Scenario
    WindSpeed = 0 && MaxPrecipitation > 0, SWITCH(
        TRUE(),
        RainDescription = "Light Rainfall", "drizzle.png",
        RainDescription = "Moderate Rainfall" && DayNightInd = "Daytime", "few showers day.png",
        RainDescription = "Moderate Rainfall" && DayNightInd = "Nighttime", "few showers night.png",
        RainDescription = "Heavy Rainfall", "thunderstorms.png"
    ),

    // No Wind, No Rain Scenario
    WindSpeed = 0 && MaxPrecipitation = 0, SWITCH(
        TRUE(),
        CloudyPrediction = "Clear" && DayNightInd = "Daytime", "sun.png",
        CloudyPrediction = "Clear" && DayNightInd = "Nighttime", "moon.png",
        CloudyPrediction = "Partly Cloudy" && DayNightInd = "Daytime", "partly cloudy day.png",
        CloudyPrediction = "Partly Cloudy" && DayNightInd = "Nighttime", "partly cloudy night.png",
        CloudyPrediction = "Cloudy", "cloudy.png"
    ),

    // Wind, No Rain Scenario
    WindSpeed > 0 && MaxPrecipitation = 0, "wind.png",

    // Default
    BLANK()
)

```

Purpose: Determines the appropriate weather icon based on a combination of factors: wind speed, precipitation, cloudiness, and time of day.

- **Type:** Calculated Column
- **Logic:** This is a more complex DAX code that uses SWITCH within SWITCH to handle different scenarios:
 - Scenario 1: Both wind and rain.

- Scenario 2: No wind but rain (with nested conditions for different rain descriptions and day/night).
- Scenario 3: No wind, no rain (with nested conditions for cloud predictions and day/night).
- Scenario 4: Wind but no rain.

WeatherIcon
wind and rain.png
wind and rain.png
wind and rain.png
wind and rain.png
wind and rain.png
wind and rain.png
wind and rain.png
wind and rain.png
wind and rain.png
wind and rain.png
wind and rain.png
thunderstorms.png

Formatted Time:

```
TimeFormatted = TIME(FLOWERS[Hour], 0,0)
```

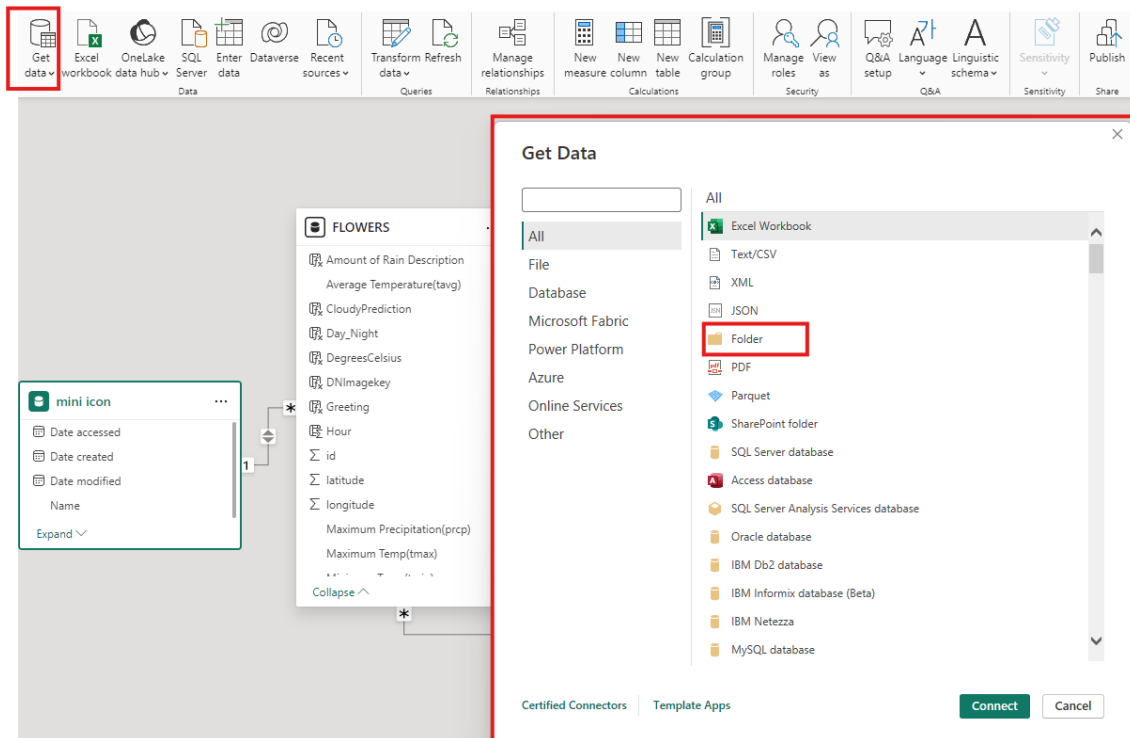
- **Purpose:** Converts the [Hour] field into a time format for further calculations.
- **Type:** Calculated Column
- **Logic:** The TIME function converts the [Hour] value to a time format with minutes and seconds set to 0.

TimeFormatted
09:00:00
14:00:00
10:00:00
10:00:00
14:00:00
11:00:00
11:00:00
00:00:00
08:00:00
09:00:00

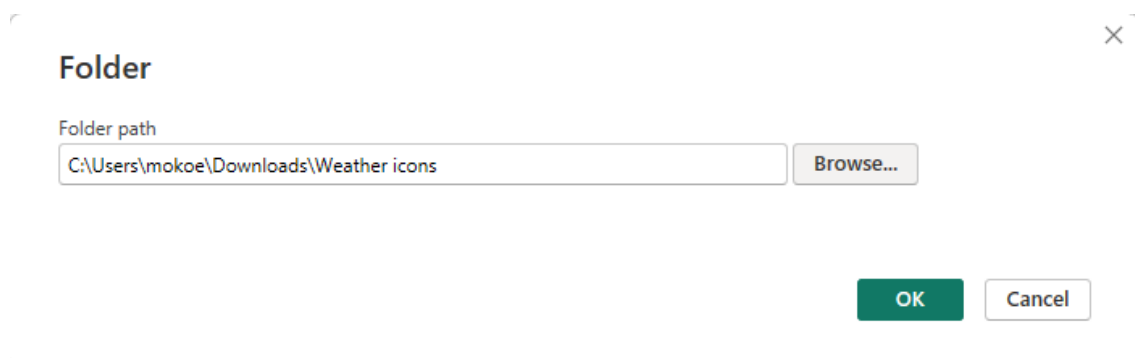
2.1.1. Putting it all together:

Before setting up the visuals, we need to link the specific weather scenario with the relevant images; we need to source the appropriate images and link them to the matching weather conditions. See below:

We need first to upload the necessary images. So, we want to get the data via the get data option on the top left screen (see red outline):



We go to “Folder” and click on it. You will be prompted to give the folder with the necessary items. Once done, click “Ok.”



You will see the below screen where you need to transform or load the data and pay attention to the name:

C:\Users\mokoe\Downloads\Weather icons

Content	Name	Extension	Date accessed	Date modified	Date created	Attributes	Folder Path
Binary	blizzard.png	.png	2024/08/28 20:16:46	2024/08/27 16:42:19	2024/08/27 16:42:19	Record	C:\Users\mokoe\Downloads
Binary	mild cloud.gif	.gif	2024/08/28 20:21:52	2024/08/27 23:31:22	2024/08/27 23:31:22	Record	C:\Users\mokoe\Downloads
Binary	overcast night.png	.png	2024/08/28 20:21:52	2024/08/27 16:43:42	2024/08/27 16:43:42	Record	C:\Users\mokoe\Downloads
Binary	overcast.png	.png	2024/08/28 20:21:52	2024/08/27 16:43:16	2024/08/27 16:43:16	Record	C:\Users\mokoe\Downloads
Binary	partly cloud.png	.png	2024/08/28 22:21:04	2024/08/27 23:39:33	2024/08/27 23:39:33	Record	C:\Users\mokoe\Downloads
Binary	partly cloudy.png	.png	2024/08/28 20:21:52	2024/08/27 16:40:03	2024/08/27 16:40:03	Record	C:\Users\mokoe\Downloads
Binary	Partly_cloudy.png	.png	2024/08/28 22:20:59	2024/08/27 23:53:49	2024/08/27 23:53:48	Record	C:\Users\mokoe\Downloads
Binary	rain.png	.png	2024/08/28 22:21:17	2024/08/27 16:40:40	2024/08/27 16:40:40	Record	C:\Users\mokoe\Downloads
Binary	showers.gif	.gif	2024/08/28 22:21:11	2024/08/27 23:37:32	2024/08/27 23:37:31	Record	C:\Users\mokoe\Downloads
Binary	snow.png	.png	2024/08/28 20:16:46	2024/08/27 16:40:21	2024/08/27 16:40:21	Record	C:\Users\mokoe\Downloads
Binary	sunny.gif	.gif	2024/08/28 20:21:52	2024/08/27 23:28:46	2024/08/27 23:28:45	Record	C:\Users\mokoe\Downloads
Binary	Sunny.png	.png	2024/08/28 20:16:35	2024/08/27 23:54:30	2024/08/27 23:54:30	Record	C:\Users\mokoe\Downloads
Binary	thunderstorms.gif	.gif	2024/08/29 09:45:49	2024/08/27 23:33:03	2024/08/27 23:33:03	Record	C:\Users\mokoe\Downloads
Binary	windy.png	.png	2024/08/28 20:21:54	2024/08/27 16:41:06	2024/08/27 16:41:06	Record	C:\Users\mokoe\Downloads
Binary	Daytime.png	.png	2024/08/29 18:02:34	2024/08/28 20:24:39	2024/08/28 00:02:13	Record	C:\Users\mokoe\Downloads
Binary	Nighttime.png	.png	2024/08/29 18:02:34	2024/08/28 20:25:02	2024/08/28 00:03:09	Record	C:\Users\mokoe\Downloads
Binary	cloudy.png	.png	2024/08/30 22:41:59	2024/08/28 22:35:15	2024/08/28 22:35:15	Record	C:\Users\mokoe\Downloads
Binary	drizzle.png	.png	2024/08/29 18:02:34	2024/08/28 22:52:30	2024/08/28 22:52:30	Record	C:\Users\mokoe\Downloads
Binary	few showers day.png	.png	2024/08/30 22:41:59	2024/08/28 22:36:08	2024/08/28 22:36:07	Record	C:\Users\mokoe\Downloads
Binary	few showers night.png	.png	2024/08/29 18:02:34	2024/08/28 22:36:44	2024/08/28 22:36:44	Record	C:\Users\mokoe\Downloads

The data in the preview has been truncated due to size limits.

Combine

Load

Transform Data

Cancel

Go to the Relationships tab or Model View on the left and drag the fields with which you want to form a relationship. Set the Cardinality to “Many to one” and Cross -filter direction to “Both” for the [WeatherIcon] field on the necessary table (in this case, it’s the Flowers table) and the [Name] field in the Mini Icon table:

There are pending changes in your queries that haven't been applied.

Relationships

mini icon

- Date accessed
- Date created
- Date modified
- Name

Flowers

- Amount of Rain Description
- Average Temperature
- CloudyPrediction
- Day_Night
- DegreesCelsius
- DNImageKey
- Greeting
- Hour
- id
- latitude
- longitude
- Maximum Precipitation
- Maximum Temp(tmax)

Edit relationship

Select tables and columns that are related.

From table: FLOWERS

	user_login	user_name	WeatherIcon	Wind Direction...	Wind Speed(...	WindSpeedD...
www.i...	craigpeter	Craig Peter	cloudy.png	0	0	Light Wind
www.i...	craigpeter	Craig Peter	cloudy.png	0	0	Light Wind
www.i...	craigpeter	Craig Peter	cloudy.png	0	0	Light Wind

To table: mini icon

cessed	Date created	Date modified	Extension	Folder Path	mini_base64	Name
3/29 0...	2024/08/28 2...	2024/08/28 2...	.png	C:\Users\mok...	data:image/p...	cloudy.png
3/29 1...	2024/08/28 2...	2024/08/28 2...	.png	C:\Users\mok...	data:image/p...	drizzle.png
3/29 0...	2024/08/28 2...	2024/08/28 2...	.png	C:\Users\mok...	data:image/p...	few showers ...

Cardinality: Many to one (*:1)

Cross-filter direction: Both

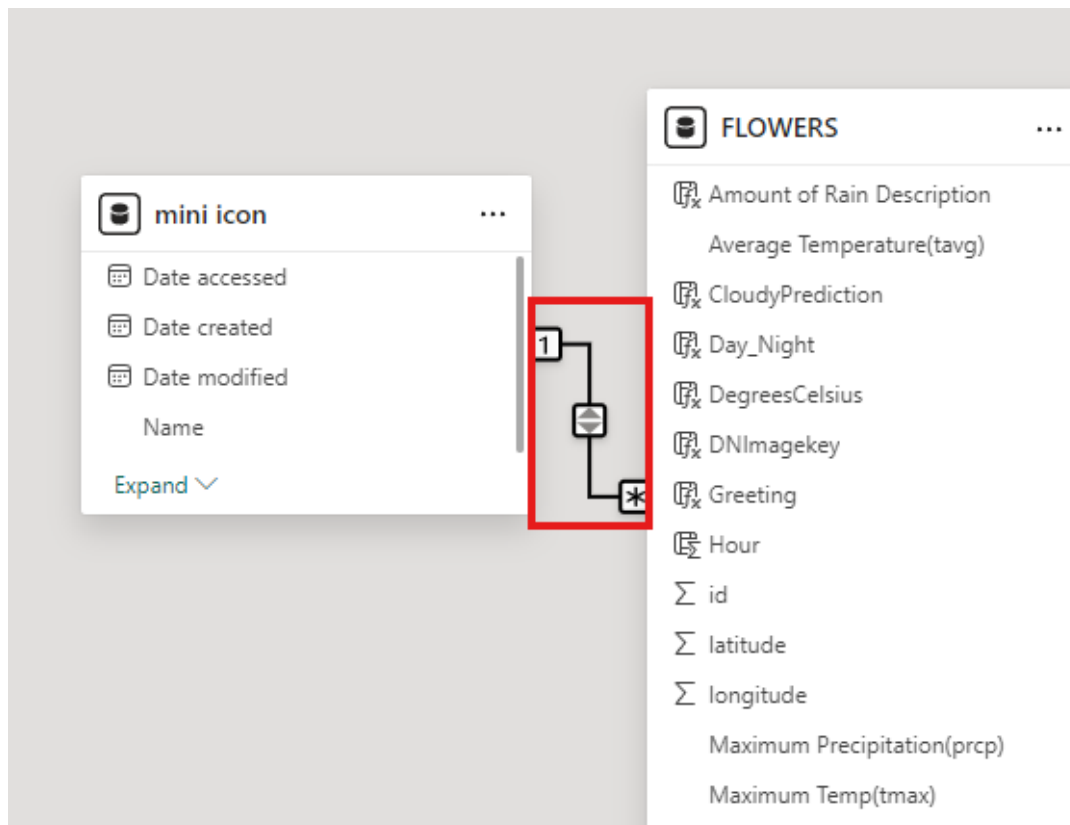
☒ Make this relationship active

☐ Apply security filter in both directions

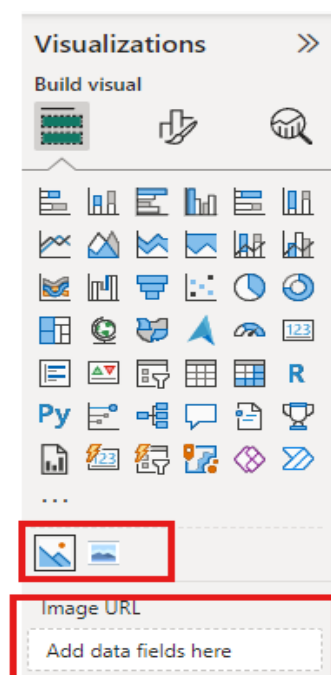
☐ Assume referential integrity

Save Cancel

Once the necessary relationships are created, the picture will be linked to the specific weather condition in our original table (Flowers table):



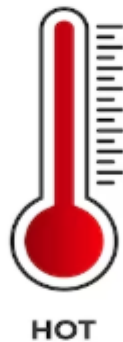
Once those are set up, we can use the Power BI “Image” or “Simple Image” visualisation tools to add pictures. Please select one of them and add the image path on the Mini Icon table. This will be linked to the weather conditions to display the necessary image according to the selected data.



See the below example:

- P-PRJ371

Temperature and Conditions Visuals:



- **Temperature Gauge:**
 - The gauge visually labelled "HOT" is likely derived from the Temperature Description DAX measure. It calculates whether the temperature is "Hot," "Warm," "Cool," or "Cold" based on the [Average Temperature(tavg)] field.

DegreesCelsius
0 °C
0 °C
14.1 °C
15.8 °C
0 °C
13 °C
14.8 °C
12.6 °C
0 °C
9.9 °C
12.3 °C
0 °C
20.1 °C
14.1 °C

- **Degrees Celsius and Cloudy Prediction Table:**
 - The table showing "[DegreesCelsius]" and "[CloudyPrediction]" is based on the [DegreesCelsius] and [CloudyPrediction] DAX measures. The [DegreesCelsius] measure formats the temperature with "°C," while [CloudyPrediction] determines if it is "Cloudy," "Partly Cloudy," or "Clear" based on the [Pressure] value.

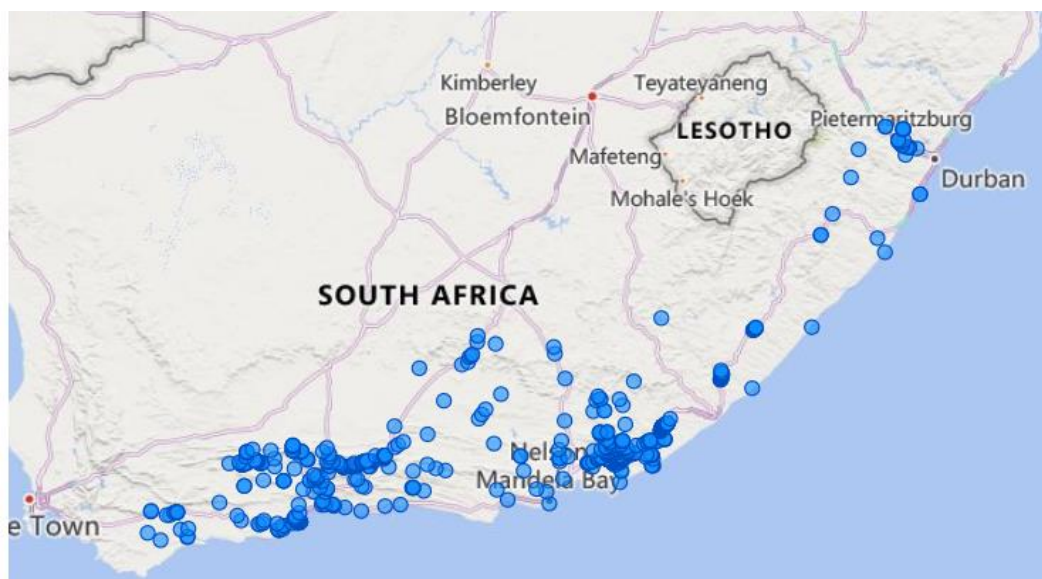
Rain Description Table:

Amount of Rain Description

Light Rainfall
Light Rainfall
Heavy Rainfall
Light Rainfall
Light Rainfall
Light Rainfall
Light Rainfall
Light Rainfall
Light Rainfall
Light Rainfall
Light Rainfall
Light Rainfall
Light Rainfall

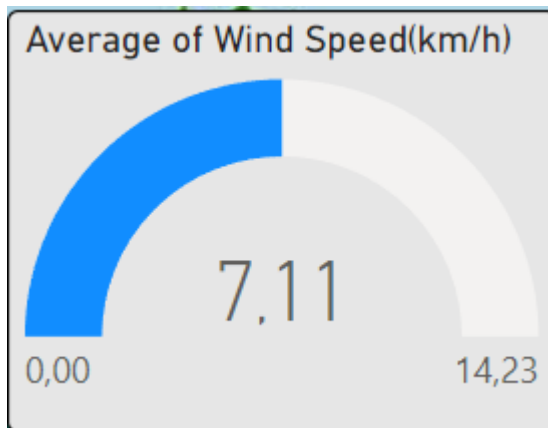
- The **"Amount of Rain Description"** column in the table uses the Amount of Rain Description DAX measure to classify the rainfall intensity (e.g., "Light Rainfall," "Moderate Rainfall," etc.) based on the [Maximum Precipitation(prcp)] value.

Location Visual:



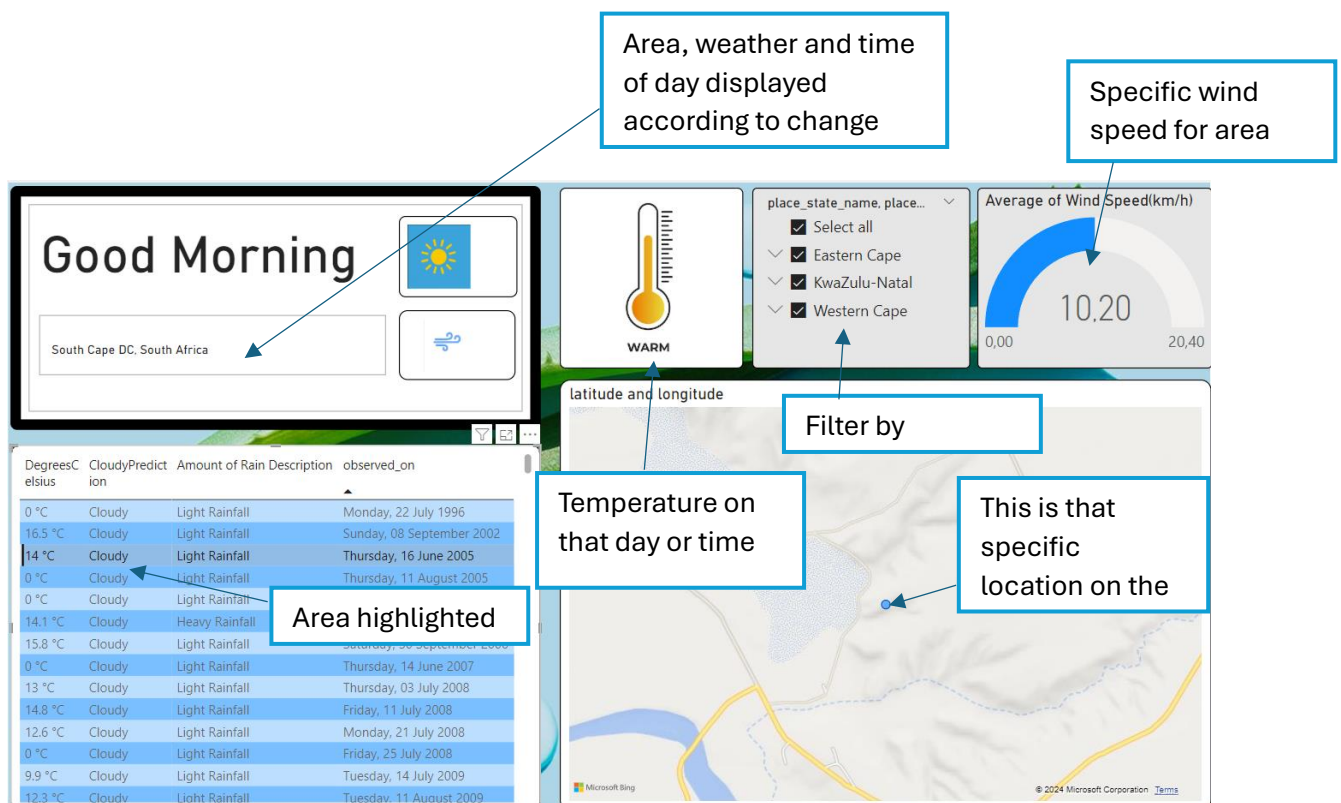
- The **Map Visual** displaying various points in South Africa is likely driven by the **latitude and longitude** data. At the same time, filtering might be applied based on the weather data conditions (like temperature, precipitation, etc.) calculated using the above DAX measures.

Wind Speed Visual:



- The **Gauge for Wind Speed** (Average of Wind Speed(km/h)) reflects the wind speed data and its dynamic interpretation can be linked to the [WindSpeedDescription] DAX measure. It categorises wind conditions (e.g., "Severe Gale," "Very Windy/Strong Winds") based on [Wind Speed(wspd)].

Dynamic data according to location:



Summarising the above:

- Regional Weather Monitoring:**
This report serves as a weather monitoring dashboard for specific regions in South Africa, like the Eastern Cape, KwaZulu-Natal, and the Western Cape. It provides insights into weather conditions, such as temperature, wind speed, rainfall, and cloud cover.
- Real-Time Weather Updates and Forecasts:**

The greeting message and weather icons (e.g., "Good Afternoon" and a "moon" icon for nighttime) suggest that the report is providing real-time updates on the weather, allowing users to understand current conditions quickly. This could be particularly useful for planning daily activities or travel.

- **Analysis of Historical Weather Data:**

The table displaying historical weather data (e.g., the "[observed_on]" column) shows that the report also analyses long-term weather trends. This data could help identify patterns such as recurring periods of heavy rainfall or consistent temperature trends.

- **Decision-making tool for Agriculture and Tourism:**

Detailed rainfall, wind speed, and temperature information are crucial for agricultural planning. Farmers could use this data to decide when to plant and harvest based on predicted rainfall and temperature conditions. Likewise, this report could be useful in the tourism industry, providing data that helps tourists plan trips to natural reserves like Addo Elephant National Park and ensures favourable weather conditions during their visit.

- **Emergency Preparedness and Disaster Management:**

The report can be useful for disaster management agencies by categorizing weather conditions such as "heavy rainfall," "severe gale," or "extreme rainfall." By providing timely alerts and actionable insights, it can help them prepare for and respond to weather-related emergencies.

- **Environmental and Climate Research:**

The report could also be valuable for environmental researchers studying climate change patterns. The data on temperature changes, rainfall variability, and wind speed provides insights into how the climate is evolving in these regions.

- **Public Awareness and Safety:**

The simple visual indicators, like weather icons and greeting messages, make the data more accessible to the public. This serves as a public awareness tool, informing residents of potential weather hazards and enabling them to take necessary precautions.

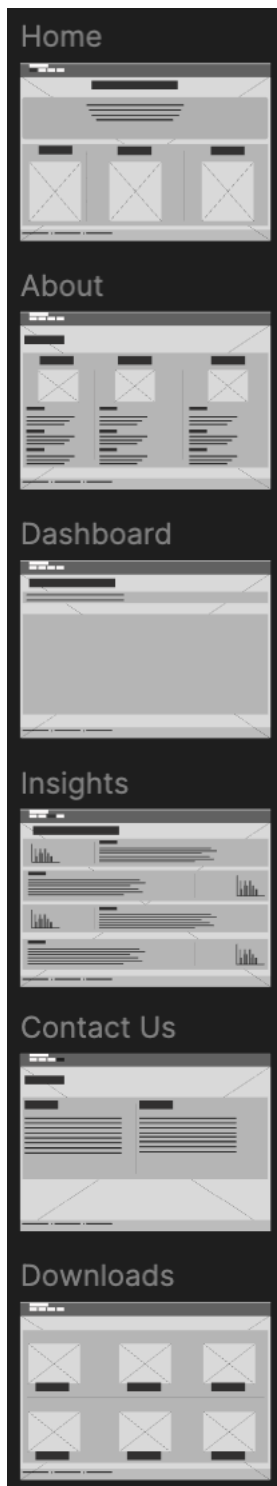
- **Visual Data-Driven Insights:**

The combination of tables, gauges, and maps allows users to explore the data interactively. For example, the map shows weather conditions across various geographical points, which helps in spatial analysis and understanding how weather varies within the country.

3. User Interface Design

3.1. UI Layout

LOW FIDELITY:

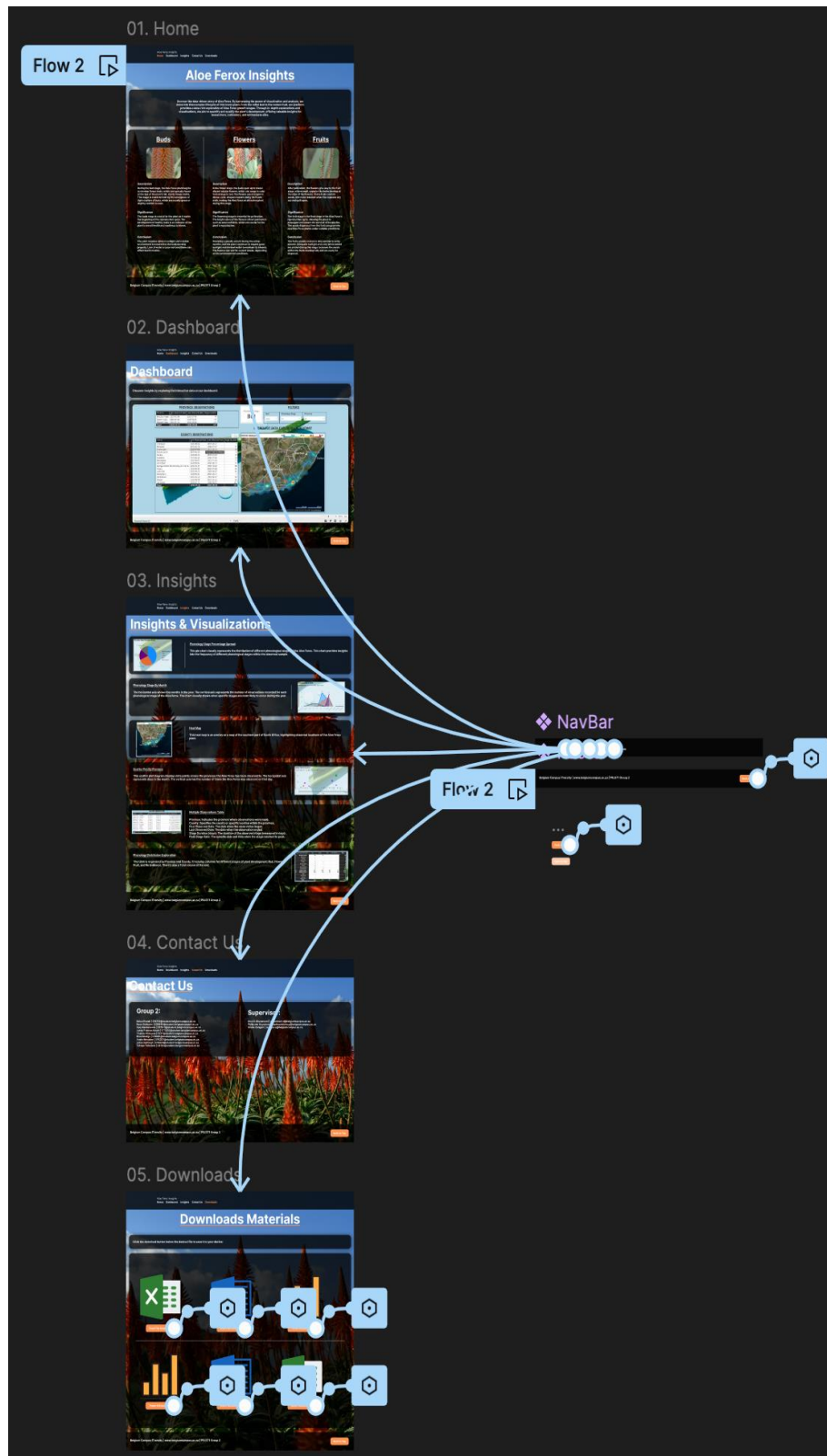


HIGH FIDELITY:



3.2.UAX Prototyping

NAVIGATION BETWEEN PAGES



Link to Figma designs:

<https://www.figma.com/design/4jhem5mKZUNTSihrhsGiHc/PRJ371?node-id=0-1&node-type=CANVAS&t=TNaHY8SjEiH4wOPD-0>

Details from High Fidelity:**01. Home:**

The landing page contains a phenology and information about Aloe Ferox and its three phases.

02. Dashboard:

Filter and display various information about the plant, such as province, timeline observation, weather, and phenology state, in different data illustrations provided by Power BI.

03. Insights:

Explanation of each data illustration on the Dashboard page.

04. Contact Us:

Contact information of each member in the group, along with the supervisor's contact information.

05. Downloads:

Download plant data in 3 formats: Excel, Power BI and Word.

3.3. User Experience.

Front-end prototyping focuses on designing and testing the web application's user interface and user experience (UI/UX). Here, we will detail how the front end looks, interacts, and functions, including the user interface components and visualisations.

Wireframes and Mock-ups:

A tool called Figma was used to create wireframes and visual mock-ups. These outline what the app will look like in low-fidelity and high-fidelity when done.

- **Home Page:** This is where most of the information on the three stages of aloe ferox—the buds, flowers, and fruit stages—will be.
- **Dashboard Page:** The dashboard page is where the user will have an interactive dashboard component to explore the data with visualisations.
- **Insights Page:** The insights page will contain key visualisations and graphs explaining their meaning and purpose in relation to the provided data.
- **Contact Us Page:** The Contact Us page contains details for all the team members and supervisors who oversee the team.
- **Downloads Page:** The downloads page is a repository containing all the project's data material. Therefore, it allows users to access the data themselves and provides them with a convenient way of downloading the data on their machines in a predetermined format.

User Flow:

Overall User Flow:

- Home Page → (Introduction to Aloe Ferox and navigation to other sections)
- Dashboard Page → (Interactive data exploration)
- Insights Page → (Understanding the visualisations)
- Downloads Page → (Downloading data in different formats)
- Contact Us Page → (Reaching out to the team)

Entry Point - Home Page:

- **Objective:** Introduce users to Aloe Ferox and provide basic information about its three development stages: Buds, Flowers, and Fruit.
- **User Interaction:** Upon landing on the home page, users will see an overview of the Aloe Ferox phenological stages, including images and text. They can navigate through tabs or links to learn more about each stage (Buds, Flowers, Fruit). A navigation bar directs users to key pages like Dashboard, Insights, Downloads, and Contact Us.
- **Flow:** After understanding the plant's stages, users can click a call-to-action (CTA) button to explore detailed data on the Dashboard Page.

Dashboard Page:

- **Objective:** Allow users to explore and interact with the phenological data through visualisations.
- **User Interaction:** Users will arrive here to interact with the data, where they can select phenological stages or filters. Visualisations such as heat maps and tables will update based on user inputs. There will be options to switch between various graphs and maps, giving users the freedom to analyse specific regions or periods.
- **Flow:** Users can continue exploring different visualisations or move on to the Insights Page to better understand specific graphs.

Insights Page:

- **Objective:** Provide users with key insights and explanations about the visualisations they see on the dashboard.
- **User Interaction:** Each visualisation is described in detail, explaining its representation and relevance to Aloe Ferox's phenology. Users can then navigate back to the Dashboard or move to the Downloads page for further exploration.
- **Flow:** After viewing insights, users can proceed to the Downloads Page to access the raw data or reach out via the Contact Us Page.

Downloads Page:

- **Objective:** Provide access to downloadable project data in different formats (e.g., Excel, Word, Power BI files).
- **User Interaction:** Users can select and download various datasets related to the project, including processed data or raw environmental data for their analysis. The page will offer clear options for downloading the data in user-friendly formats (e.g., Excel, Word, Power BI reports). Users can also find supporting documents like reports or project summaries here.

- **Flow:** After downloading the data, users may want to provide feedback or contact the team for further clarification, which leads them to the Contact Us Page.

Contact Us Page:

- **Objective:** Provide users with contact details for team members and supervisors overseeing the project.
- **User Interaction:** Users can find the project teams and supervisors' names and emails.
- **Flow:** After contacting the team or submitting inquiries, users may return to the Home Page or further explore the Dashboard and Insights sections.


Interactive components:

- Navigation Bar:
 - .1. **Function:** A fixed navigation bar at the top of every page to guide users through different site sections (Home, Dashboard, Insights, Downloads, Contact Us).
 - .2. **Interactive Features:** Users can hover over or click on each item to navigate between pages.
 - .3. **User Interaction:** Users can smoothly transition between pages with a single click.
- Interactive Visualisations (Dashboard & Insights Pages):
 - .1. **Function:** Data visualisations that respond to user inputs and provide detailed insights.
 - .2. **Interactive Features:** Zoom and pan functionality for maps, allowing users to focus on specific areas. Dynamic chart updates when users change the filters (e.g., changing the geographic area or phenological stage filters updates the maps or tables).
 - .3. **User Interaction:** Users can explore the data through interactive graphs, filter for specific information or zoom in on specific regions.

3.4. Personas.

Personas are designed to help developers and designers better understand the user experience by personifying the individuals who will use the system the most. Below is a semi-fictional person who is targeted to use this system more than anyone else. Going by the title Botanist, these personas will make more use of this system to gain various information for their work.

3.4.1. First Persona:








Job Title
Naturalist

Age
35 to 44 years

Highest Level of Education
Master's degree (e.g. MA, MSc)

Social Networks

Industry
Agriculture

Organization Size
11-50 employees

Botanist

Preferred Method of Communication

- Phone
- Email
- Face-To-face
- Text Messaging
- Social Media

Tools They Need to Do Their Job

- Notebook/Tablet
- Data Analysis Software
- Climate Modeling Software
- GPS Device
- Digital Camera
- Hand Lenses
- Soil Test kits
- Geographic Information Systems (GIS)

Job Responsibilities

Conducting Research, Analyzing Data, Conservation and Environmental Management, Publication and Communication, Policy and Advocacy and Publication and Communication

Their Job Is Measured By

Research Contributions, Fieldwork and Data Collection, Applied Work in Industry and Collaboration and Outreach

Reports to

Project Director

Goals or Objectives

Environmental Monitoring and Climate Change Research, Scientific Discovery and Research, Conservation of Plant Species and Ecosystems, and Medicinal and Industrial Applications

They Gain Information By

Field Research and Observation, Laboratory Research, Literature Review and Scientific Publications, Collaborating with Other Scientists and Experts and Technology and Computational Tools

Biggest Challenges

- Resources
- Climate Change
- Pollinator Decline
- Habitat Loss and Deforestation
- Balancing Conservation and Economic Development
- Adapting Agriculture to Global Challenges

3.4.2. Second Persona:

	Preferred Method of Communication <ul style="list-style-type: none"> • Phone • Email • Text Messaging • Social Media • Face-To-face 	Job Responsibilities <ul style="list-style-type: none"> • Conservation and Environmental Management • Agriculture and Crop Improvement • Conducting Research • Analyzing Data
Name Ms Botanist	They Gain Information By <ul style="list-style-type: none"> • Field Research and Observation • Laboratory Research • Literature Review and Scientific Publications • Collaborating with Other Scientists and Experts • Technology and Computational Tools • Ethnobotanical • Studies and Experiments • Applied Research 	Reports to <ul style="list-style-type: none"> • Government Agencies • Botanical Gardens • Arboretums • Museums and Private Sector (Agriculture, Pharmaceuticals, Environmental Consulting)
Job Title Biostatistician	Goals or Objectives <ul style="list-style-type: none"> • Scientific Discovery and Research • Sustainable Agriculture and Food Security • Medicinal and Industrial Applications 	Their Job Is Measured By <ul style="list-style-type: none"> • Research Contributions • Fieldwork and Data Collection • Applied Work in Industry • Collaboration and Outreach
Age 25 to 34 years		Biggest Challenges <p>Resources</p> <p>Habitat Loss and Deforestation</p> <p>Climate Change</p> <p>Balancing Conservation and Economic Development</p> <p>Pollinator Decline</p>
Highest Level of Education Bachelor's degree (e.g. BA,		Tools They Need to Do Their Job <ul style="list-style-type: none"> • GPS Devices • Email • Field Notebooks/Tablets • Climate Modeling Software • Geographic Information Systems (GIS) • Data Analysis Software • Herbarium Management Software • Remote Sensing Technology • Drones • Digital Imaging Tools • Soil and Water Monitoring Sensors • Digital Cameras • Hand Lenses
Social Networks     		
Industry Agriculture		
Organization Size 1-10 employees		

4. Performance and Scalability

The system must meet specific performance requirements to efficiently and timely retrieve, process, and visualize phenological data. Scalability considerations ensure the system can handle increased data volumes, additional users, and future expansions.

4.1. Performance Requirements.

The system's performance is crucial to providing users a smooth, real-time experience when interacting with the data and visualisations. Below are the key performance requirements across different components:

Category	Performance Requirement	Target/Metric
API Response Time	To ensure seamless data updates, the system must fetch weather data from the Meteostat API and phenological data from iNaturalist within a reasonable time frame.	- Average response time: < 1 second - Max response time: 3 seconds
Database Query Speed	Queries to the SQL database must be optimised to retrieve large datasets, including phenology observations and weather data, without significant delays.	- Query response time: < 2 seconds for 95% of queries - Maximum response time: 5 seconds
Data Processing Latency	The ETL (extract, transform, load) process for integrating new weather and phenology data into the database must be executed efficiently to support real-time updates.	- Latency for batch processing: < 5 minutes for batches of up to 10,000 records
Power BI Dashboard Refresh	Power BI dashboards should update in near real-time, reflecting the latest data from the database and API sources.	- Data refresh interval: Every 10 minutes (configurable) - Dashboard loading time: < 5 seconds
Frontend Responsiveness	The user interface must respond to user actions (e.g., data filtering, interaction with visualisations) with minimal delay to enhance user experience.	- Response to UI actions: < 1 second - Page load time: < 2 seconds
Concurrency Handling	The system should support multiple concurrent users querying data, running visualisations, and updating the database without performance degradation.	- Support for 50 concurrent users - Zero downtime during concurrent database access

The system will apply caching mechanisms, efficient API handling, database indexing, and query optimisation techniques to meet these performance goals.

4.2. Scalability.

Scalability ensures the system can handle increasing data volumes, growing user numbers, and additional functionalities without significant performance degradation. The following aspects have been considered to achieve scalability:

Scalability Dimension	Strategy/Approach	Details
Database Scalability	Vertical and Horizontal Scaling	<ul style="list-style-type: none"> - Vertical scaling: Initially, we will ensure that the SQL server has enough CPU, memory, and storage to handle larger data volumes as required. - Horizontal scaling: When the system needs to handle an even greater load, we will distribute the database across multiple servers, partitioning data for efficient querying.
API Integration Scaling	Rate-Limiting and Asynchronous Processing	<ul style="list-style-type: none"> - Rate-limiting: Implement rate-limiting to ensure API calls are distributed over time, avoiding request bottlenecks. - Asynchronous Processing: Use asynchronous programming models to manage multiple API requests concurrently and efficiently process larger volumes of data.
Data Volume Scaling	Batch Processing and Data Archiving	<ul style="list-style-type: none"> - Batch processing: As datasets grow (e.g., more phenological observations), ETL processes will be optimised to handle batch processing, updating data in chunks. - Data archiving: Older datasets that are not frequently accessed will be archived to reduce the load on active databases.
Power BI Scaling	DirectQuery and Incremental Refreshes	<ul style="list-style-type: none"> - DirectQuery: For large datasets, Power BI will be configured to use DirectQuery, pulling only relevant data during user interactions rather than loading entire datasets. - Incremental refreshes: Use incremental refreshes to ensure only the latest data is queried, avoiding reloading historical data constantly.
Frontend Scalability	CDN and Load Balancing	<ul style="list-style-type: none"> - CDN (Content Delivery Network): Host static assets (e.g., JavaScript, CSS) on a CDN for faster loading times across geographies. - Load balancing: Distribute incoming traffic across multiple server instances to handle more concurrent users without degrading performance.
User Scalability	Session Management and Distributed Processing	<ul style="list-style-type: none"> - Session management: Efficient session handling will ensure multiple users can interact with the system simultaneously without overloading server resources. - Distributed processing: Spread the processing load across multiple backend services to ensure high availability and responsiveness.
API Service Scalability	Microservices Architecture	<ul style="list-style-type: none"> - Microservices: If required, the FastAPI service can be split into smaller, independent services (e.g., separate services for weather data and phenology data), improving scaling capabilities.

5. Security Design

The security design aims to protect the system from unauthorised access and data breaches and ensure compliance with relevant legal and regulatory frameworks. This section outlines key security requirements and compliance measures for the system.

5.1. Security Requirements

To safeguard sensitive data (e.g., phenology and weather data, user information) and ensure the integrity of the system, the following security requirements will be implemented:

Security Category	Description	Requirements
Authentication	Ensuring only authorised users have access to the system and its data.	<ul style="list-style-type: none"> - User authentication: Implement OAuth 2.0 for user authentication (via Google, GitHub, or other platforms). - Multi-factor authentication (MFA): Enable MFA for additional security.
Authorisation	Role-based access control (RBAC) to restrict system functionalities and data access based on user roles (e.g., admin, data analyst, public users).	<ul style="list-style-type: none"> - RBAC: Implement role-based access control where admin users can edit, add, and delete data, while public users can only view and query data.
Data Encryption	Data should be protected at rest and in transit to prevent unauthorised access.	<ul style="list-style-type: none"> - Data at rest: Use AES-256 encryption for sensitive data stored in the SQL database (e.g., user information). - Data in transit: Use HTTPS/SSL to encrypt data transmitted over the network.
API Security	Securing external API integrations (e.g., Meteostat, iNaturalist) to prevent unauthorised access and rate-limiting attacks.	<ul style="list-style-type: none"> - API keys: All API requests will use secure, environment-variable-stored API keys. - Rate limiting: Implement rate limiting for external API requests to prevent abuse.
Database Security	Prevent unauthorised access and ensure data integrity in the SQL database.	<ul style="list-style-type: none"> - SQL injection prevention: Use parameterised queries and input validation to prevent SQL injection attacks. - Database encryption: Encrypt sensitive fields in the database.
Network Security	Protect the network infrastructure from unauthorised access and attacks.	<ul style="list-style-type: none"> - Firewalls: Configure firewalls to restrict traffic to the API server, database, and other sensitive components. - IP whitelisting: Enable IP whitelisting for admin-level access to the system.
Data Integrity	Ensure that the data remains accurate and consistent throughout its lifecycle.	<ul style="list-style-type: none"> - Audit logs: Maintain audit trails for critical actions such as data insertions, deletions, and updates. - Version control: Implement version control for datasets to track changes and updates.
Vulnerability Management	Regularly identifying, assessing, and mitigating vulnerabilities in the system.	<ul style="list-style-type: none"> - Automated scanning: Automated vulnerability scanning tools detect potential security flaws (e.g., OWASP Top 10 vulnerabilities).

		<ul style="list-style-type: none"> - Patching: Regularly update dependencies and apply patches to address security vulnerabilities.
Session Management	Protect against session hijacking and ensure secure session handling for logged-in users.	<ul style="list-style-type: none"> - Session expiration: Implement session timeouts for idle users. - Secure cookies: Use secure, HTTP-only cookies to store session information.
Backup and Recovery	Ensure data availability and integrity in the case of system failure or cyberattacks.	<ul style="list-style-type: none"> - Regular backups: Perform daily database backups to a secure location. - Disaster recovery plan: Develop and test a disaster recovery plan to ensure system restoration in case of attack.

5.2.Compliance

The system must adhere to various legal, regulatory, and industry standards to ensure data privacy, security, and integrity. Compliance with these standards will help mitigate risks and ensure trust from users and stakeholders.

Compliance Standard	Description	Implementation Strategy
GDPR (General Data Protection Regulation)	If the system handles personal data from users based in the European Union, GDPR compliance must be ensured to protect user privacy.	<ul style="list-style-type: none"> - Data minimisation: Collect only necessary user data and ensure users' rights to access, rectify, and delete their data. - User consent: Obtain user consent before collecting personal data. - Data breach notification: Develop a procedure for notifying relevant authorities and affected users within 72 hours of a data breach.
POPIA (Protection of Personal Information Act)	For South African users, ensure compliance with POPIA to protect user data and privacy.	<ul style="list-style-type: none"> - Data protection officer (DPO): Appoint a DPO to ensure the system complies with POPIA's data protection principles. - Privacy policy: Provide a clear privacy policy outlining how user data will be handled.
OWASP (Open Web Application Security Project)	Use OWASP best practices to ensure the web application is free from common vulnerabilities (e.g., SQL injection, cross-site scripting).	<ul style="list-style-type: none"> - OWASP Top 10: Implement security controls to mitigate OWASP Top 10 risks, including input validation, secure authentication, and regular vulnerability assessments.
PCI DSS (Payment Card Industry Data Security Standard)	If the system involves financial transactions (e.g., subscription fees), ensure compliance with PCI DSS standards for secure handling of payment data.	<ul style="list-style-type: none"> - Secure payment gateways: Use third-party PCI DSS-compliant payment processors (e.g., Stripe, PayPal) to handle transactions. - Encryption: Encrypt all sensitive financial data (e.g., credit card information) using industry-standard encryption algorithms.

HIPAA (Health Insurance Portability and Accountability Act)	In case the system involves health data, HIPAA compliance ensures the protection of sensitive health-related information.	<ul style="list-style-type: none"> - Data encryption: Encrypt health-related data both in transit and at rest. - Access control: Implement strict access controls to ensure only authorised personnel can access health data.
API Compliance	Ensure that the third-party APIs integrated into the system comply with their respective terms of service and data protection policies.	<ul style="list-style-type: none"> - API terms adherence: Ensure the system adheres to the rate limits, data usage, and security requirements outlined in terms of service for third-party APIs (e.g., Meteostat, iNaturalist).
Data Retention Policies	Implement data retention policies in line with legal and business requirements.	<ul style="list-style-type: none"> - Data retention schedules: Define retention periods for different data types (e.g., user data, observation records) and implement automated data deletion or archiving processes after the retention period.

6. Testing and Validation

The testing and validation process ensures that the system meets functional and non-functional requirements, adheres to security standards, and performs as expected. This section outlines the test plan and validation criteria for the system.

6.1. Test Plan.

The test plan details the testing strategy, scope, objectives, schedule, and test environments. The system will undergo several testing phases, including unit testing, integration testing, system testing, performance testing, and user acceptance testing (UAT).

Test Type	Description	Objective	Tools/Methods
Unit Testing	Testing individual components or modules to ensure they function correctly in isolation.	Ensure that each function or module (e.g., API calls, database queries, data processing) works as intended.	- Mocha - Chai - Jest
Integration Testing	Testing the interactions between integrated modules (e.g., API integrations with weather, Spotify, and Twitter APIs).	Ensure that integrated components interact correctly and data flows as expected.	- Postman - Newman - Supertest
System Testing	End-to-end testing of the entire system to ensure it meets functional and non-functional requirements.	Verify that the system works as a whole, covering all user stories, business logic, and UI functionality.	- Selenium - Cypress
Performance Testing	Evaluating system performance under normal and peak load conditions.	Ensure the system responds efficiently and handles the expected user load without degradation in performance.	- JMeter - Locust
Security Testing	Testing for vulnerabilities such as SQL injection, cross-site scripting (XSS), and authorisation flaws.	Identify and address potential security risks and vulnerabilities that may compromise the system's security.	- OWASP ZAP - Burp Suite
User Acceptance Testing (UAT)	Testing the system with end-users (e.g., data analysts, botanists, stakeholders) to verify that it meets their needs and is ready for deployment.	Ensure the system aligns with the requirements of end-users and stakeholders, confirming it is fit for purpose.	- UAT scenarios and test cases
Regression Testing	Ensuring that new changes (e.g., features or bug fixes) do not negatively affect existing functionality.	Verify that the system continues to function correctly after updates or changes.	- Automated test suites
Load Testing	Simulating high-volume traffic to test how the system handles stress and concurrency.	Ensure the system can handle many simultaneous users or requests without crashing or significant performance degradation.	- LoadRunner - Gatling

Database Testing	Testing the integrity of data transactions, CRUD operations, and data migration to ensure the database functions correctly.	Ensure the correct operation of database interactions, data consistency, and integrity during read/write operations, including bulk imports/exports.	- SQL query testing - Data Validators
API Testing	Testing the functionality, reliability, and security of API endpoints integrated within the system.	Validate that external APIs (e.g., iNaturalist, Meteostat) and internal API calls return expected data and handle errors gracefully.	- Postman - Newman
Compatibility Testing	Ensuring the system works across different devices, operating systems, browsers, and platforms.	Ensure the web interface and dashboards are responsive and fully functional across major browsers (e.g., Chrome, Firefox, Safari) and operating systems (Windows, macOS, Linux).	- BrowserStack
Backup and Recovery Testing	Testing the backup and recovery process to ensure the system can be restored during data loss or corruption.	Verify the ability to recover from backup without data loss and ensure the backup mechanisms work effectively.	- Manual/automated recovery scenarios

Test Objectives

The primary objectives are to:

- Ensure all functional and non-functional requirements are met.
- Validate data integrity, performance, security, and usability.
- Identify and resolve defects before deployment.

Test Scope

- **In-Scope:** Unit, integration, system, user acceptance, performance, and regression testing.
- **Out-of-Scope:** Testing external data sources and non-core features.

Test Environment

Testing will be conducted using virtual machines, software tools (PowerBI, Python/JavaScript, MySQL), and real and synthetic data. There will be development, staging, and live environments.

- **Development environment:** Initial testing by developers (unit and integration testing).
- **Staging environment:** Pre-deployment testing (system and UAT).
- **Production environment:** Live deployment tests (performance, security).

Test Strategy and Approach

- **Unit Testing:** Focus on individual components conducted by developers.
- **Integration Testing:** Developers and testers validate data flow between the front end, back end, and database.
- **System Testing:** QA will conduct end-to-end testing of the entire application.

- **User Acceptance Testing (UAT):** Ensures user needs are met, conducted by stakeholders and experts.
- **Performance and Load Testing:** Evaluates the system under stress.
- **Regression Testing:** Ensures that new updates don't break existing functionality.

Test Execution Plan

Test cases will cover all requirements, with detailed logs on outcomes. The phases include unit, integration, system, UAT, performance, and regression testing, each with its responsible team.

Roles and Responsibilities

- **Project Manager:** Oversees testing.
- **QA Lead:** Manages planning, execution, and reports.
- **Developers and Test Engineers:** Handle coding, bug fixes, and test execution.
- **Stakeholders:** Approve UAT results.

Defect Management

Defects are logged, prioritised, and resolved based on severity levels (critical, high, medium, low), following a triaging, fixing, and verifying cycle.

Test Reporting and Metrics

Daily reports will track test execution, defect density, and coverage. UAT sign-off marks approval from stakeholders.

Risks and Mitigations

Potential risks include incomplete requirements, code freeze delays, and unstable environments, with strategies in place to mitigate these risks.

Test Closure

A final report will summarise the testing phase, and the team will sign off once all objectives are met.

6.2. Validation Criteria.

The validation criteria ensure the system meets the functional, performance, and security requirements defined in the specifications. The following criteria will be used to validate the system:

Validation Category	Description	Validation Metrics
Functional Requirements	Validating that all functional requirements are met per the system design and user stories.	<ul style="list-style-type: none">- User stories completion: All defined user stories must be successfully implemented.- Feature testing: All major features (e.g., data display, API integrations, reports) work as expected.
Performance Requirements	Ensuring the system meets specified performance metrics, including response time, load handling, and concurrency.	<ul style="list-style-type: none">- Response time: The system must have an average response time of <1 second for data queries.- Scalability: The system must handle at least 1,000 concurrent users without failure.

Security Requirements	Verifying that the system is secure and adheres to security best practices to protect data and resources.	<ul style="list-style-type: none"> - Authentication: Only authorised users must be able to access the system. - Data protection: All sensitive data must be encrypted, and no critical vulnerabilities (e.g., SQL injection, XSS) should exist.
Usability	Validating that the system is user-friendly and intuitive for end-users, stakeholders, and analysts.	<ul style="list-style-type: none"> - User satisfaction: 90% of users in UAT must report satisfaction with the system's ease of use and navigation. - Interface accessibility: The UI must comply with accessibility standards (WCAG).
Data Accuracy and Integrity	Ensuring the data is accurate, reliable, and correctly handled throughout the system's processes (e.g., data input, retrieval, and analysis).	<ul style="list-style-type: none"> - Data validation: Data retrieved from APIs (iNaturalist, Meteostat) must be accurate and consistent. - Database integrity: All CRUD operations on the database must work without data corruption.
Compliance	Ensuring the system complies with legal, regulatory, and industry standards (e.g., GDPR, POPIA).	<ul style="list-style-type: none"> - Compliance audit: The system must pass an external compliance audit and meet all legal standards.
Cross-platform Compatibility	Ensuring the system is accessible and fully functional across multiple devices, browsers, and platforms.	<ul style="list-style-type: none"> - Browser compatibility: The system must work on all major browsers (Chrome, Firefox, Safari, Edge). - Mobile responsiveness: The system must be fully functional on mobile and tablet devices.
API Reliability	Validating the reliability and consistency of third-party APIs integrated into the system.	<ul style="list-style-type: none"> - API uptime: APIs must have an uptime of >99%. - Error handling: APIs must return clear, actionable error messages when issues arise.
Backup and Recovery	Validating that the system's backup and recovery procedures work effectively in case of failure.	<ul style="list-style-type: none"> - Recovery time: The system must be restored within 2 hours in case of failure. - Backup frequency: Daily backups must occur without failure.
Regression Testing	Ensuring that updates or changes to the system do not negatively impact existing functionality.	<ul style="list-style-type: none"> - Regression pass rate: 100% of previous features and functionality must continue to work after new updates or patches.

Validation Process:

1. **Unit Testing and Code Review:** Developers perform unit tests, and peer reviews validate the functionality of individual components.
2. **Integration Testing:** Test integration points, including API interactions and database operations.
3. **User Acceptance Testing (UAT):** End-users, such as data scientists and botanists, validate that the system meets their functional needs.

4. **Performance Validation:** Test under real-world conditions to verify performance metrics (e.g., load handling, response times).
5. **Security Validation:** Conduct vulnerability assessments and ensure compliance with security requirements.

Functionality Testing:

- **Objective:** Verify that all interactive components (e.g., navigation bar, filters, buttons) function as intended.
- **Tests:** Test the navigation bar to ensure all links direct users to the correct pages. Test the data filters to confirm they update visualisations when selections are made in real-time. Verify that the interactive visualisations (hover tooltips, zoom/pan) respond correctly to user interactions. Test the download button to ensure files are downloaded in the correct format without errors.

Usability Testing:

- **Objective:** Assess the user interface's ease of use and intuitiveness, ensuring users can navigate and interact with the application smoothly.
- **Tests:** Conduct user tests where participants are asked to complete tasks (e.g., explore the Dashboard, download a dataset). Gather feedback on the clarity of the navigation structure and whether users can easily understand how to filter and analyse the data. Identify potential pain points or confusing elements in the UI (e.g., unclear filter labels or overly complex chart interactions).

Responsiveness Testing:

- **Objective:** Ensure the interface works seamlessly across different devices (mobile, tablet, desktop) and screen sizes.
- **Tests:** Test the interface on various screen resolutions to ensure that components like the navigation bar, charts, and forms resize correctly. Ensure the data visualisations remain interactive and fully functional on touchscreens and smaller displays. Test the responsive layout to confirm that all UI elements adjust well when switching between landscape and portrait orientations on mobile devices.

Performance Testing:

- **Objective:** Test the speed and responsiveness of the user interface, ensuring a smooth user experience even with large datasets.
- **Tests:** Measure the load time for pages like the Dashboard Page, where visualisations may require extensive data processing. Test the performance of real-time data updates when using filters on the dashboard (e.g., how quickly the charts update when new data is selected). Check for any lag or slowdown when interacting with large-scale maps and datasets.

Cross-browser Testing:

- **Objective:** Ensure the UI works consistently across web browsers (e.g., Chrome, Firefox, Safari, Edge).
- **Tests:** Test the functionality of interactive elements (filters, buttons, charts) in each browser. Ensure that visualisations render correctly in all supported browsers without layout or performance issues.

7. Deployment and Maintenance

7.1. Deployment Plan.

The deployment plan outlines the steps needed to set up, deploy, and ensure the initial functionality of the Aloe Ferox Phenology web application.

Stage	Details
Project Setup	<ul style="list-style-type: none"> - Initialize a new project directory. - Set up version control using Git.
Frontend Development	<ul style="list-style-type: none"> - Create the basic HTML structure. - Style the application using CSS. - Implement JavaScript to handle user interactions, such as scroll abilities.
Backend Development	<ul style="list-style-type: none"> - Develop a FastAPI service to fetch weather and iNaturalist data and update the SQL server. - Use JavaScript to interact with FastAPI for data requests.
Database Integration	<ul style="list-style-type: none"> - Set up MySQL database. - Connect the backend to the MySQL database. - Implement CRUD operations to manage tasks and records.
Testing & Validation	<ul style="list-style-type: none"> - Test the application locally to ensure functionality. - Gather user feedback for improvements. - Make necessary adjustments based on feedback.
Project Structure	<ul style="list-style-type: none"> - Organize the project with the following folders: Functions, Images, Pages, Resources, Style, and Readme. - Each folder should contain corresponding files for logic, images, downloadable resources, styling, and documentation.
Deployment Instructions	<ul style="list-style-type: none"> - Clone the repository from GitHub using the provided link. - Open in Visual Studio Code: Right-click on index.html and select "Open with Live Server". - Access in browser: Interact with the application at http://127.0.0.1:5500/pages/index.html.

Usage Instructions

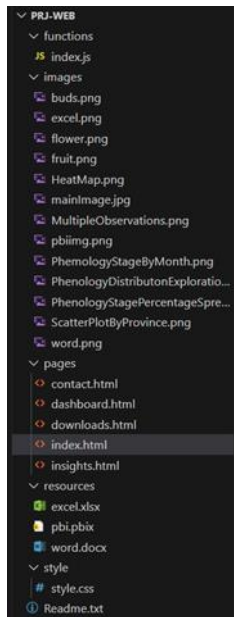
Upon launching the application:

- Home Page: View the key stages of Aloe Ferox development (Buds, Flowers, and Fruit).
- Dashboard Page: Interact with a dynamic dashboard, filter phenological parameters, and view real-time visualisations (graphs, heatmaps).
- Insights Page: Access detailed data patterns and insights.
- Downloads Page: Download project data in Excel, CSV, and Power BI formats.
- Contact Us Page: Reach the project team for feedback or queries.

Project Structure

The technologies used to implement the project are HTML, CSS and JavaScript.

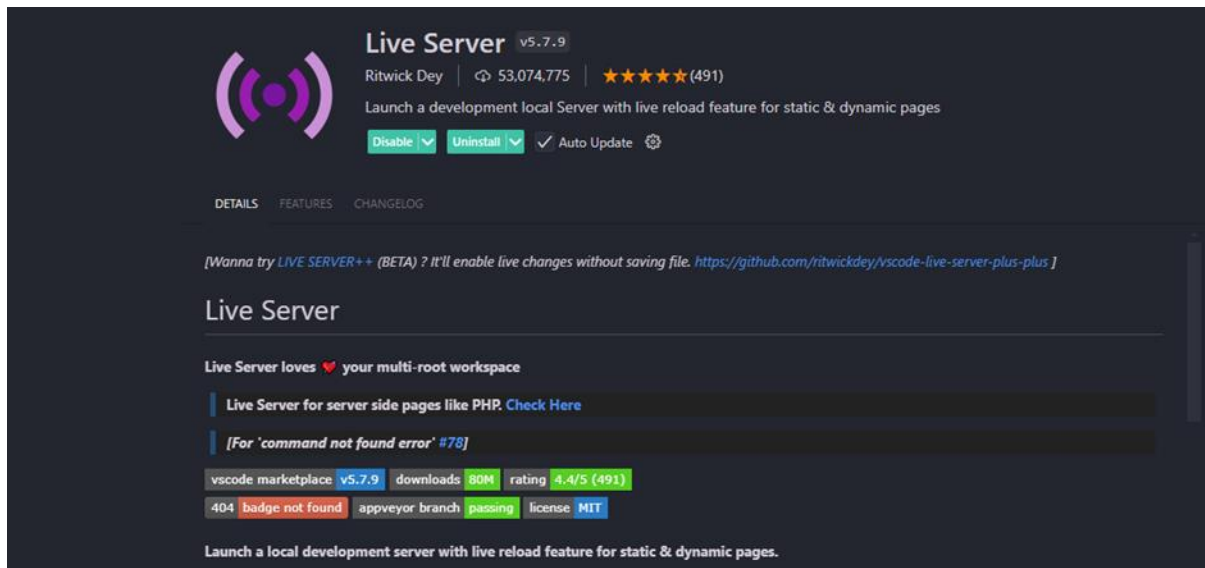
The project structure is as seen in the image below:



- **Functions:** The functions will contain files with any logic used in the JavaScript application.
- **Images:** This folder will contain all the image resources for the web application and can be accessed via the path “/images/flower.png.”
- **Pages:** This is where the pages for the web application will be located
- **Resources:** Resources will contain all the downloadable files and resources that the user will be able to download and have access to
- **Style:** The styling section will contain the global style sheet for the application
- **Readme:** The readme will contain setup information and general team information

Setup Instructions

Pre-requirements: You will need a GitHub account with your Belgium Campus email, and you will need to install Visual Studio Code. Then, you must install the *Live Server* extension in Visual Studio Code.



Step 1: Clone the repository from GitHub: <https://github.com/Reino-p/PRJ-WEB.git>

Step 2: Open the code in Visual Studio Code.

Step 3: Right-click on *index.html*.

Step 4: Select *Open with Live Server* from the menu.

Step 5: A browser should pop up with the URL: <http://127.0.0.1:5500/pages/index.html>

Step 6: You can now interact with the web application.

Usage Instructions

The Aloe Ferox Phenology web application is designed to help users explore detailed data on the flowering stages of Aloe Ferox across different geographic regions.

Upon landing on the Home Page, you'll be introduced to the three key stages of Aloe Ferox development—Buds, Flowers, and Fruit—through images and descriptions.

From there, navigate to the Dashboard Page, where you can interact with a dynamic dashboard. Use the filters to select specific phenological parameters and view real-time visualisations such as tables, graphs or heatmaps.

The Insights Page provides detailed explanations of these visualisations, offering context and insights into the data patterns.

On the Downloads Page, you can access project data and download it in various formats (e.g., Excel, CSV, Power BI) for further analysis.

The Contact Us Page offers a way to reach the project team if you have questions or feedback.

7.2.Maintenance Plan.

The maintenance plan ensures the long-term stability, performance, and usability of the web application, database, and data visualisations. Following this deployment and maintenance plan will make the project robust, scalable, and easy to maintain for developers and end-users.

Aspect	Details
Development Tools	<ul style="list-style-type: none"> - Git: Version control for managing updates and collaboration. - Visual Studio Code: This maintains JavaScript/HTML/CSS. - SQL Management Studio: This manages and maintains the MySQL database. - Power BI: This is used to maintain and update reports and visualisations.
Database Maintenance	<ul style="list-style-type: none"> - Regular Backups: Schedule automatic database backups to prevent data loss. - Indexing: Optimize frequently queried columns with indexes to improve query performance. - Normalization: Ensure the database adheres to normalisation principles to minimise redundancy and improve efficiency.
UI Maintainability	<ul style="list-style-type: none"> - Regular Updates: Keep HTML5, CSS, and JavaScript libraries updated with the latest versions for security patches and feature enhancements. - Code Quality: Maintain clean, well-documented code with tools like ESLint (JavaScript) and Stylelint (CSS). - Performance Optimization: Minify HTML, CSS, and JavaScript files to reduce load times and improve performance.
Bug Fixes & Enhancements	<ul style="list-style-type: none"> - Issue Tracking: Use GitHub Issues to track bugs and feature requests. - Regular Updates: Schedule regular maintenance updates to address bugs, security issues, and requested features.
Documentation	<ul style="list-style-type: none"> - Code Documentation: Maintain comprehensive internal documentation for the codebase to facilitate future development. - User Documentation: Provide clear guides and FAQs for users to navigate and understand the application.
Power BI Maintenance	<ul style="list-style-type: none"> - Report Updates: Periodically refresh Power BI reports with the latest data. - Optimize Visuals: Ensure visualisations remain relevant and are optimised for performance in real-time dashboards.