11/1/2024

Milestone 3 Progress Report for Phenology Data Analytics System (Aloe Ferox)

Team members:

REON GOTTSCHE
ANDRE NIENABER
JAMIE FRANCES KENCH
REINO POWELL
AJAY RAMAKELAWAN
ELIAS MOGIDA
TSHEPO TSIBOLANE
JOHAN HATTINGH
THABISO MOKOENA

Milestone 3 Progress Report for Phenology Data Analytics System (Aloe Ferox)

1. Purpose and Scope

This document serves to provide progress on the technology stack used and what the current progress of the solution development is according to Milestone 3 for the **Phenology Data Analytics Solution** focused around the Aloe Ferox Phenology.

This system integrates various data sources, including weather and phenological data, to enable detailed analytics and visualization.

Below is a preliminary perspective on various requirements in order to have the solution on a live platform.

2. System Components

Component	Description
Backend Framework	FastAPI (Python 3.9+) for API and data processing.
External APIs	Meteostat API: For retrieving weather data.
Visualization Tool	Power BI for dynamic dashboards and data visualization embedded via iframes.
Frontend Framework	HTML, CSS, and JavaScript (Angular/React for interactivity).
Coding Platforms	Visual Studio Code and PyCharm for development and debugging.

3. Prerequisites

Hardware Requirements:

Requirement	Description
Server	Minimum of 8 GB RAM, 100 GB storage, and a quad-core processor. Recommended: 16 GB RAM, SSD storage, scalable instance.
Client Devices	Modern web browsers (Chrome, Firefox, Safari, Edge) with JavaScript enabled.

Software Requirements:

Requirement	Description
Backend and API	Python 3.9+, FastAPI
Frontend	HTML, CSS, JavaScript (Angular/React)
Visualization	Power BI Desktop and Power BI REST API for embedding

Data sources	Meteostat and iNaturalist APIs, including configured API keys
Coding Platforms	Visual Studio Code, PyCharm

4. Proposed Installation and Configuration Requirements

4.1 Backend Setup

Step	Command/Instruction	Description
Clone the Repository:	git clone [repository_url]	Clones the repository to your local machine.
Setup Virtual Environment:	Run python -m venv env and activate it.	Creates and activates a virtual environment for the project.
Install Dependencies:	pip install -r requirements.txt	Installs all required packages from the requirements.txt file.
Run FastAPI Server:	uvicorn main:app –reload	Starts the FastAPI server locally with auto-reload enabled.

4.3 API Configuration

Step	Instruction	Description
Meteostat API	Configure with a valid API key, then fetch sample weather data for verification.	Ensures Meteostat API is properly set up and tested.

4.4 Power BI Integration

Step	Instruction	Description
Power BI Setup	Create and configure visualizations in Power BI Desktop.	Prepares visual elements for the data analytics system.
Embed Power BI Reports	Use Power BI REST API to embed reports securely with token-based access.	Embeds Power BI reports into the system frontend securely.

4.5 Frontend Setup

Step	Instruction	Description
Install Node.js and Dependencies	Install required libraries and frameworks for the frontend.	Sets up the frontend environment.
UI and Dashboard Testing	Ensure API connections and embedded Power BI visualizations display accurately on the frontend.	Confirms that data is accurately represented in the user interface.
Cross-Browser Compatibility	Test on multiple browsers for UI responsiveness and interactivity.	Ensures a consistent experience across different web browsers.

5. Deployment

Step	Instruction	Description
Deploy on Production Server	Deploy FastAPI on the designated server.	Deploys the backend components.
Set Up Regular Data Updates	Schedule periodic data pulls from APIs.	Keeps the system updated with the latest data from Meteostat and iNaturalist.
Security Setup	Implement HTTPS, secure tokens, and access controls.	Ensures secure access to the application and data.

6. Testing and Validation

Testing Type	Method	Tools	Team	Frequency
Unit Testing	Test individual backend and frontend components.	Pytest, Postman	Developers	Upon code commits, as part of CI pipeline.
Connectivity Testing	Validate connectivity between components (API or external APIs).	Postman, Ping	Developers, QA Team	Weekly in dev, bi-weekly in staging.

UI Testing	Test UI functionality and responsiveness on various devices/browsers.	Selenium, BrowserStack	QA Team, Frontend Developers	After each frontend update, bi-weekly.
Load Testing	Simulate user load to test performance and response times.	JMeter, Locust	QA Team, Performance Engineers	Monthly in staging, before major deployments.
User Acceptance Testing (UAT)	Ensure end-user requirements and expectations are met.	Google Sheets, Miro for feedback	End Users, Stakeholders	During staging phase for each release.

Supplementary Testing Instructions

1. Page Load Test
    - Objective: Verify that the webpage loads successfully.
    - Steps:
        - Open the website URL in a web browser.
    - Expected Result: The webpage loads without errors.
2. Power BI Report Display Test
    - Objective: Verify that the Power BI report is displayed correctly.
    - Steps:
        - Open the website URL in a web browser.
        - Check that the Power BI report iframe is visible.
    - Expected Result: The Power BI report is visible and correctly rendered within the iframe.
3. Responsive Design Test

- Objective: Ensure the Power BI report is displayed correctly on different screen sizes.
- Steps:
  - Open the website URL on a desktop browser.
  - Resize the browser window to different widths.
  - Open the website URL on a mobile device.
- Expected Result: The Power BI report remains visible and properly scaled on all screen sizes.

4. URL Functionality Test
   - Objective: Verify that the Power BI report URL is correctly linked and accessible.
   - Steps:
     - Inspect the iframe or embed code to ensure the URL is correctly placed.
     - Open the Power BI report URL directly in a new tab.
   - Expected Result: The URL should link to the correct Power BI report and be accessible.
5. Basic Interaction Test
   - Objective: Ensure that basic interactions with the Power BI report (e.g., filters, clicks) are functional.
   - Steps:
     - Interact with filters or clickable elements within the Power BI report.
   - Expected Result: Interactions should work as expected, reflecting changes in the report.

---

7. Troubleshooting

Issue	Resolution Steps
API Access Failures	Confirm API keys, verify server accessibility, and check API rate limits.

Power BI Embedding Errors	Re-generate access tokens if necessary, verify permissions, and ensure REST API connectivity.
Server Issues	Review server logs, verify configurations, and troubleshoot any misconfigurations or resource issues.

8. Environment Management and Update Strategy

Environment	Purpose	Configuration	Deployment Tools	Update Schedule
Development	Code development and initial testing.	Local setup with FastAPI backend, frontend.	Git, Visual Studio Code, PyCharm	Continuous with code commits.
Staging	Feature validation and testing.	Cloud instance replicating production setup.	Git, Jenkins/GitHub Actions, FastAPI	Bi-weekly after feature integration.
Production	Live environment for end-users.	High-performance server with secure access and FastAPI backend.	Jenkins/GitHub Actions, Monitoring tools	Monthly; urgent patches as needed.

Code Snippets

Navbar component

Testing: Testing the navbar involves verifying its responsiveness, navigation links, and alignment across various screen sizes and devices. Ensure that all links lead to the correct pages, highlight appropriately when active. Additionally, check keyboard accessibility and whether it's fully navigable using only the keyboard.

```
<header>
    <nav>
        <div class="container">
            <div class="logo"><b>Aloe Ferox Insights</b></div>
            <ul>
                <li><a href="./index.html" style="color: orange;">Home</a></li>
                <li><a href="./dashboard.html">Dashboard</a></li>
                <li><a href="./insights.html">Insights</a></li>
                <li><a href="./contact.html">Contact Us</a></li>
                <li><a href="./downloads.html">Downloads</a></li>
            </ul>
        </div>
    </nav>
</header>
```

Iframe implementation for PowerBI

Testing: For embedded Power BI iframes, test the correct loading and responsiveness of the Power BI visuals within the iframe. Verify that interactive elements (like filters, tooltips, and drill-downs) work as expected within the iframe. Ensure that iframes scale properly on different devices, do not introduce excessive load times, and follow security protocols (e.g., embed codes and secure access)

```
<div id="main-section">
    <iframe id = "dashboardMain" title="PRJ371-FINAL-REPORT Thabzi" src="https://app.powerbi.com/view?r=eyJrIjoiMzM1NDk4YW
</div>
```

Document/Resources download structure

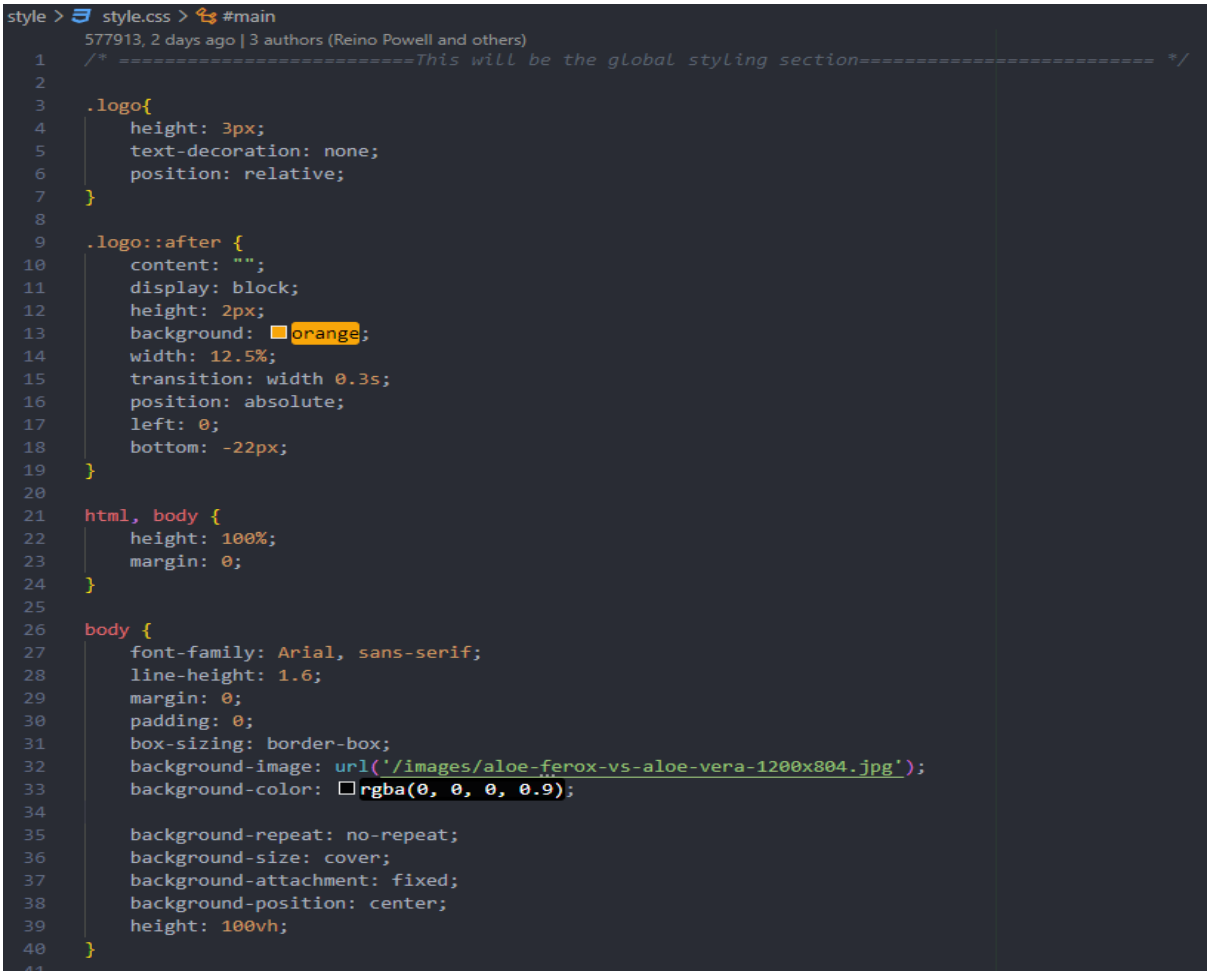
Testing: Testing the download structure includes verifying that links for each document or resource are accurate and that files download without errors. Check that files are correctly named, appropriately formatted, and accessible. Confirm download behavior across different browsers and devices, ensuring the correct MIME types to avoid security prompts

```
<div id="download-structure">
    <div id="download-blob">
        <div>Data Set</div>
        <img id="download-image" src="/images/excel.png" alt="">      You, 2 months ago • Added download page and materials for download
        <br><br>
        <button id="download-btn">
            <a id="download-text-link" href="/resources/Aloe Ferox Dataset.xlsx" download="your-excel-file.xlsx">Download</a>
        </button>
    </div>

    <div id="download-blob">
        <div>Data Visualization</div>
        <img id="download-image" src="/images/pbiimg.png" alt="">
        <br><br>
        <button id="download-btn">
            <a id="download-text-link" href="/resources/Aloe Ferox Data Visualization.pbix" download="your-pbi-file.pbix">Download</a>
        </button>
    </div>
</div>
```

Global Styling sheet for project (snippet)

Testing: Test the global stylesheet by examining its impact on page consistency, verifying that it applies styles uniformly across all pages and elements. Check for proper handling of responsive layouts, text alignment, font sizes, color schemes, and element spacing. Also, ensure there's no CSS code that conflicts with Power BI iframe styling.



Javascript functions (snippet)

Testing: Test JavaScript functions by confirming that they execute as expected, handle edge cases, and manage errors gracefully. This includes checking user interactions or animations for smooth execution across all supported browsers. Also, test for performance impact.

functions > index.js > ...

Reino Powell, 3 months ago | 1 author (Reino Powell)

```
//=========================Javascript file for functionality=========================

//=========================Back to top scroll functionality=========================

window.onscroll = function() {scrollFunction()};

function scrollFunction() {
    const button = document.getElementById("back-to-top");
    if (document.body.scrollTop > 100 || document.documentElement.scrollTop > 100) {
        button.style.display = "block";
    } else {
        button.style.display = "none";
    }
}

function scrollToTop() {
    window.scrollTo({ top: 0, behavior: 'smooth' });
}
```

Reino Powell, 3 months ago • Added javascript scroll function

```
//=========================function=========================
```

Readme file

Testing: This file only provides information about the team as no setup is necessary accept for cloning the project/repo

Readme.txt

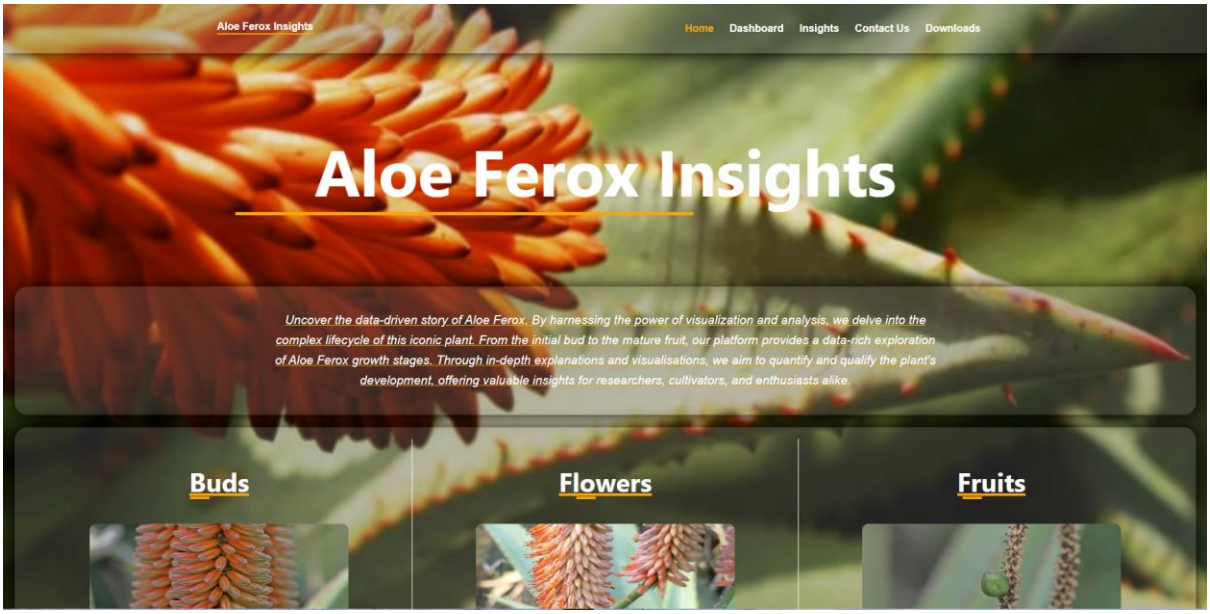
Reino Powell, 3 months ago | 2 authors (You and one other)

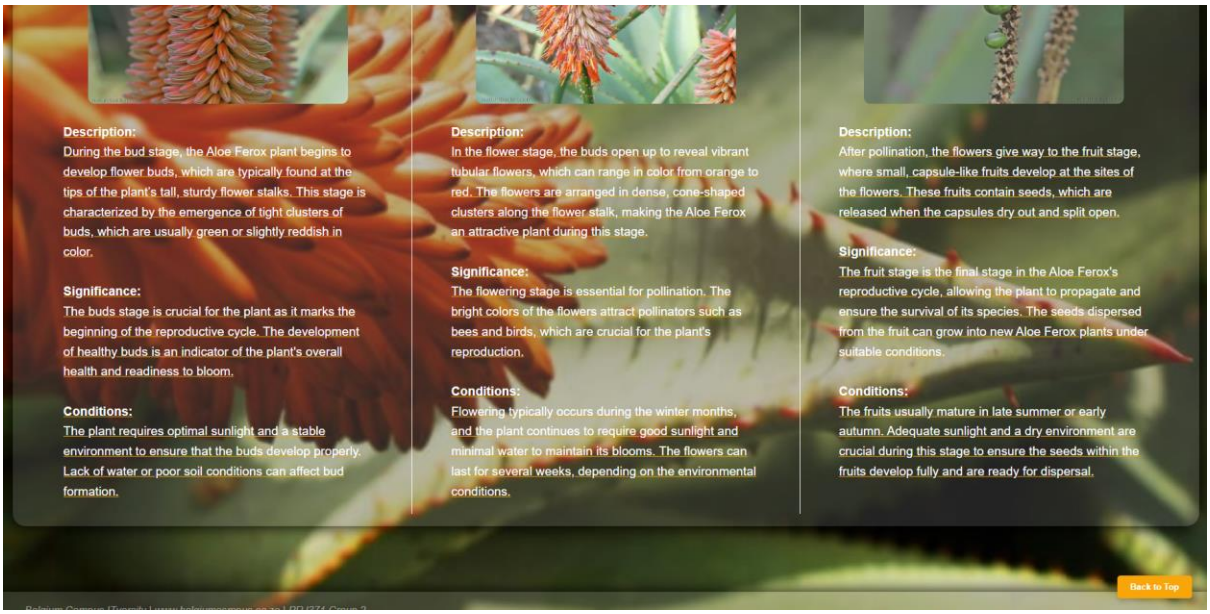
```
Data Analytics Project (PRJ371)

- Ajay Ramkelawan       | 577913@student.belgiumcampus.ac.za
- Reino Powell          | 576713@student.belgiumcampus.ac.za
- Reon Gottsche         | 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
```

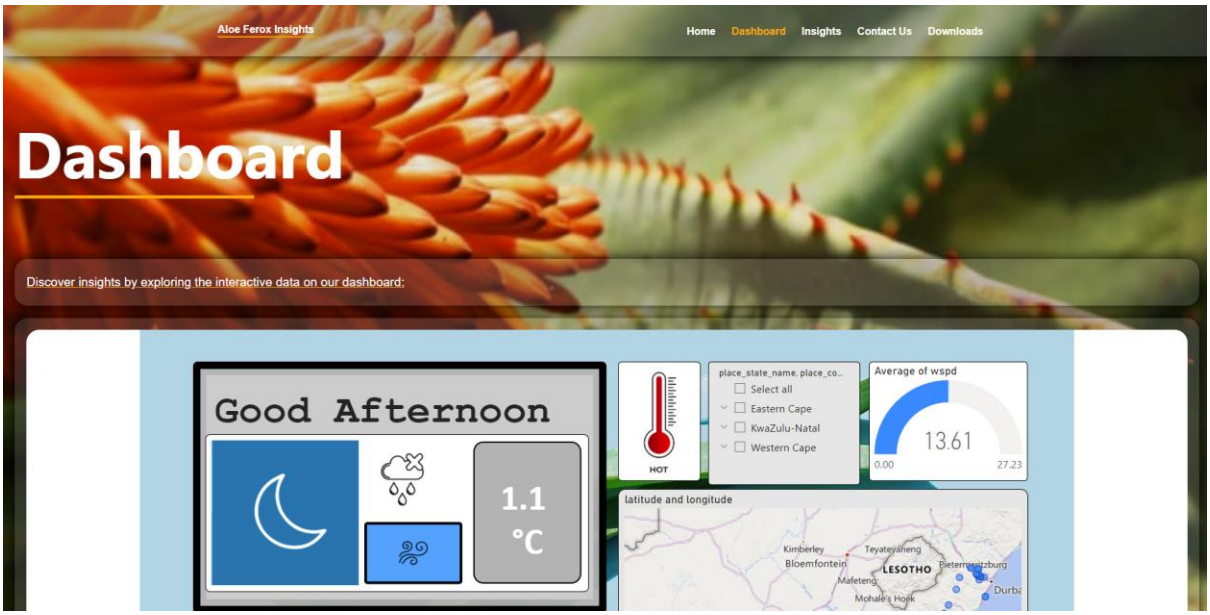
Screenshots of System

Home Page:

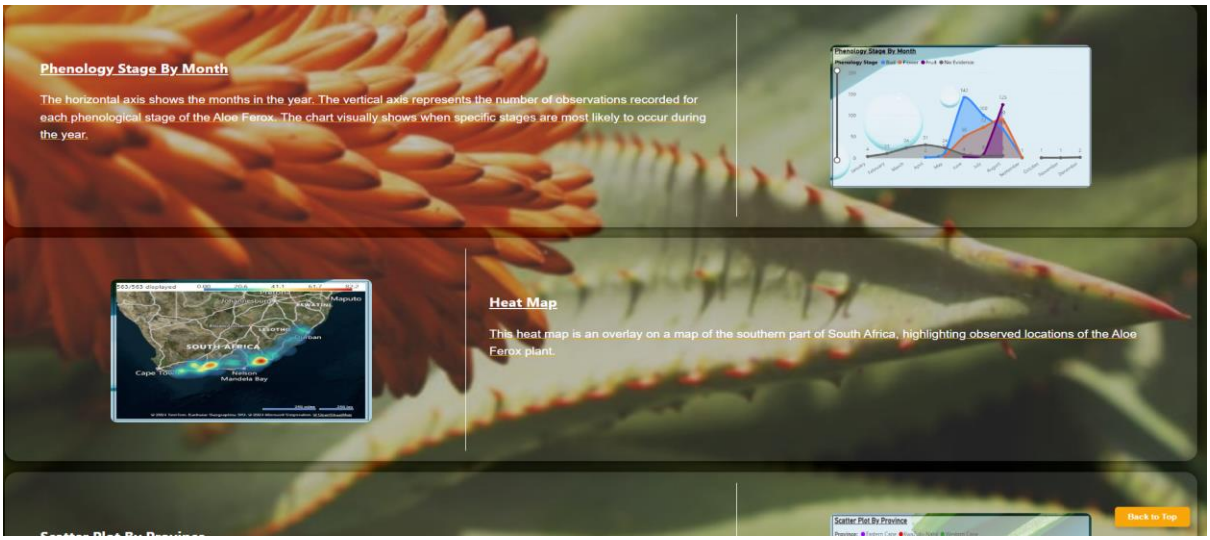




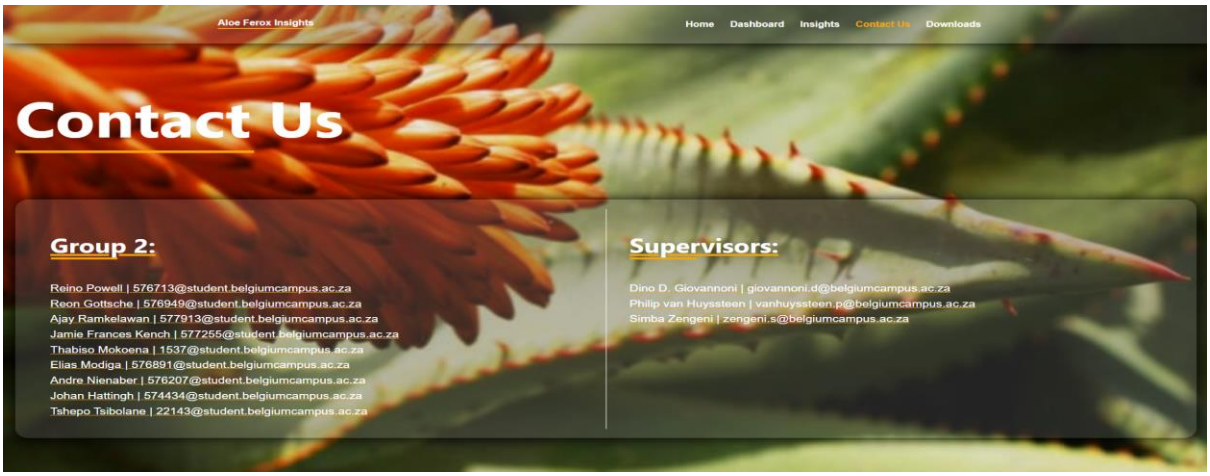
Dashboard Page:



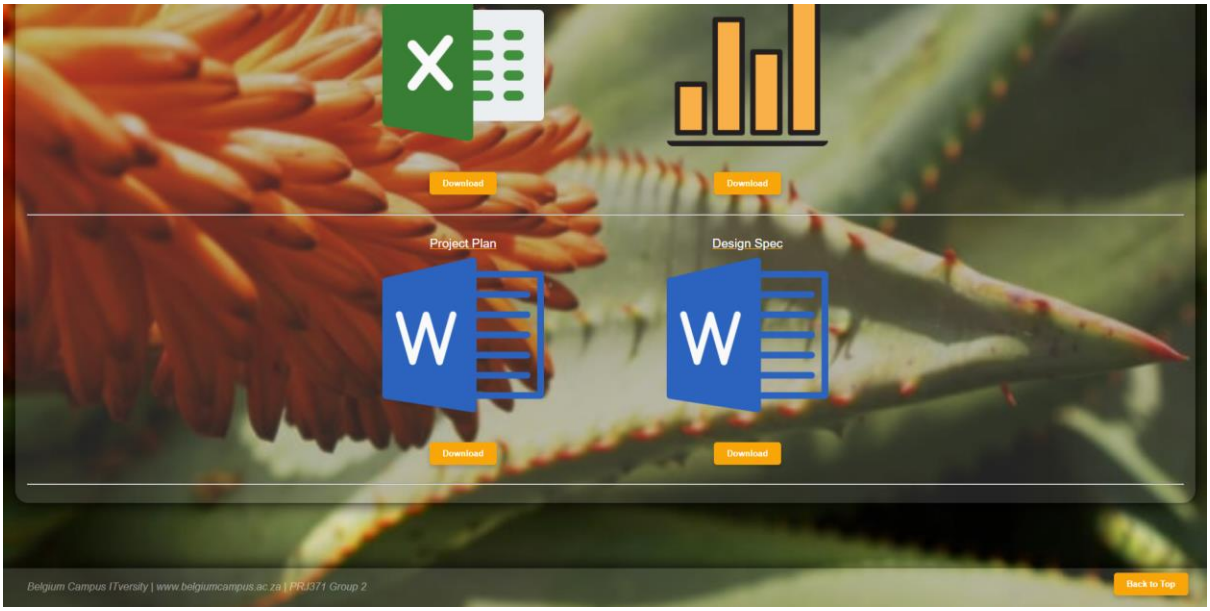
Insights Page:



Contact Page:



Document/Resource Download Page:



Testing for UI:

Testing the entire UI of the system involved a comprehensive approach that ensures each component functions smoothly and cohesively within the overall interface. We started by defining the main user workflows and navigation paths, then created test cases that reflect typical user interactions.

Key actions included verifying visual consistency, layout responsiveness, and accessibility across various devices and browsers. Functional testing checks whether buttons, links, and

interactive elements work as intended and handle interactions gracefully, including edge cases.

To ensure seamless user experience, we conducted usability testing to assess intuitive navigation and clarity. Additionally, we tested the performance by evaluating load times, responsiveness to user actions, and efficient handling of embedded content (e.g., Power BI iframes).

Note some useful information:

Github repo link (it is public)

http://127.0.0.1:5500/pages/index.html (for dev server on local machine)