MONASH University

FIT3163_S1_2025 Project Proposal & Background Research Summary

---

Due: Monday, 9 June 2025

=Student Names and ID=

Group: S1_DS_21
Sam Chan 33617236
Xin Liu 29154618（Special consideration)
Aryan Chugh 33861986
Harsh Sharma 33116059

Late Submitting Members Summary

- **Jessica: Red Text**
- **Placeholder pages: p.4/ 5-6 / 12-14/ 18-24/ 34-37**

Word Count: 9574 Date of Submission: 09/06/2025

1. Introduction

In today's world of increasingly unpredictable climate patterns and extreme weather events, it is more important than ever for both the general public and decision-makers to access and understand spatial weather data in a meaningful way. Our project, Interactive Visualisation of Spatial Data with Weather Data, aims to close the gap between raw meteorological datasets and human-friendly insights by building a web-based visualisation platform that transforms complex data into interactive, intuitive graphics.

This platform will combine geospatial information with both real-time and historical weather data, offering users a map-based interface to explore environmental patterns across different locations and timeframes. Users will be able to investigate trends, identify anomalies, and draw conclusions based on layered data visualisations such as temperature, humidity, and wind speed. The system is designed to support a wide range of users—from city planners and farmers to researchers, educators, and the general public—regardless of their technical background.

Our team is made up of Data Science students with a strong interest in user-centred design, climate awareness, and interdisciplinary data applications. This project is a response to the challenge that, while datasets from providers like the Bureau of Meteorology (BOM) or NOAA are rich and reliable, their raw formats are often too technical or difficult to interpret. By converting structured spatio-temporal data into dynamic, visual elements, we aim to lower the barriers for non-expert users to extract value from climate data.

More than just a functional tool, the platform is intended to have real-world value. We hope it will help improve public understanding of weather phenomena, support environmental education, and aid practical decision-making in fields such as urban development and agriculture.

This proposal outlines the complete development plan for the system. Section 2 presents a detailed review of related work, tools, and academic research. Section 3 explains the project management strategy, including scope, requirements, roles, timeline, and risk analysis. Section 4 summarises the external design aspects, while Section 5 describes the proposed technologies and development methods. Section 6 outlines our testing approach. Finally, Section 7 concludes the proposal and reflects on expected impact.

Throughout the semester, our team will follow an agile development process based on sprint cycles. We will begin by refining the design and confirming the key needs of target users. Next, we will gradually implement essential features, including weather data integration, interactive map layers, and a timeline slider to view changes over time. Each sprint will involve planning, coding, testing, and team review to ensure steady progress and the ability to adapt to new challenges.

By the end of the semester, we aim to deliver a functional web-based prototype that allows users to explore weather data easily through a visual and interactive map. The platform will include core elements such as layered weather displays and time-based navigation. We will also focus on documenting our development process in detail, using version control and

tracking each feature's progress from planning to testing. The system will be designed to be stable, easy to use, and ready for future improvements or wider application.

2. Background Material/Literature Review

2.1 Introduction to Spatial and Weather Data Visualisation

As climate science and environmental monitoring continue to develop, the amount of data being collected is growing rapidly. This makes it more difficult to understand and use the data without proper tools. Data visualisation plays an important role by turning complex spatial and time-based information into clear and easy-to-understand visuals (Keim et al., 2006).

When it comes to weather data, visualisation tools can help users see how different factors like temperature, wind, and humidity change over time and across different locations (Goodwin et al., 2019). Tools such as interactive maps, time sliders, and layered views make it easier for people without technical backgrounds to explore and understand the data.

2.2 Existing Platforms and Technologies

There are several platforms that currently show weather information through visualisation. For example, Windy and Ventusky provide real-time global weather data with moving map animations. These platforms have strong visual designs, but they mainly focus on showing data rather than allowing deeper interaction. They often don't include features like viewing past weather data, customizing variables, or doing detailed local analysis.

In Australia, the BOM Weather Viewer shows basic weather layers like radar and forecasts. However, its interactive functions are limited. It doesn't let users compare data across time or filter by specific weather conditions.

From a technical point of view, many of these platforms use open weather data sources such as NOAA, BOM, or the OpenWeather API. They also rely on interactive map tools like Leaflet.js or Mapbox (Chaturvedi et al., 2021). While these technologies are powerful, some GIS-based tools used in academic research are difficult to use without technical skills, which makes them less user-friendly for the general public.

2.3 Limitations and Research Gaps in Current Solutions

While the platforms mentioned above are good at showing weather changes, they often focus more on visual design than on ease of use or user interaction. A study by Zhao et al. (2022) pointed out that many weather visualisation tools can be hard to use, especially on small screens or for people without technical knowledge. Problems like too much information on the screen and limited interactive features make it difficult for users to explore the data effectively.

Also, most platforms do not allow users to explore historical weather data in detail. This is a major issue because past data is very important for fields like farming, city planning, and building design.
Professional GIS tools such as ArcGIS or QGIS are very powerful, but they are expensive and hard to learn. This makes them difficult for everyday users like teachers, students, or local government workers to access (Zhou et al., 2020). In addition, many platforms don't have options to customise weather variables, compare different regions, or analyse data across time. These are the kinds of features our project aims to include.

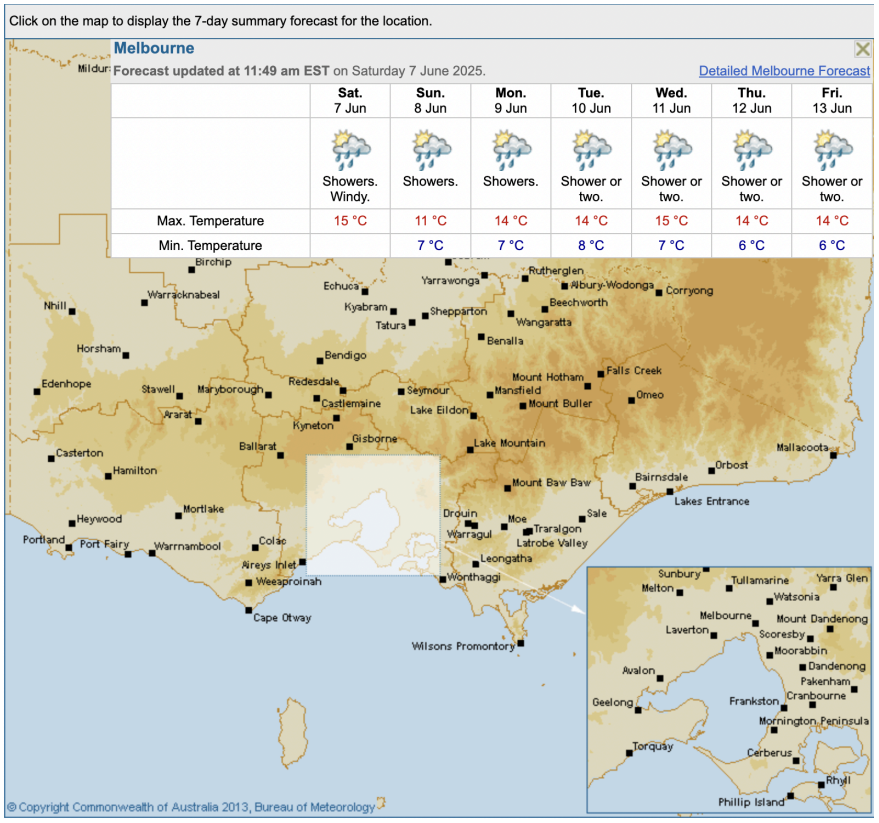
2.4 Basis and Innovations of Our Project

Our project aims to solve the problems found in current platforms by creating a simple, web-based visualisation tool that is easy to access but still provides useful analysis features. Compared to platforms like Windy or BOM, our tool will allow users to filter weather variables, compare data from different time periods, and explore maps in an interactive way. We are especially focused on making the platform easy to use for people without a technical background.

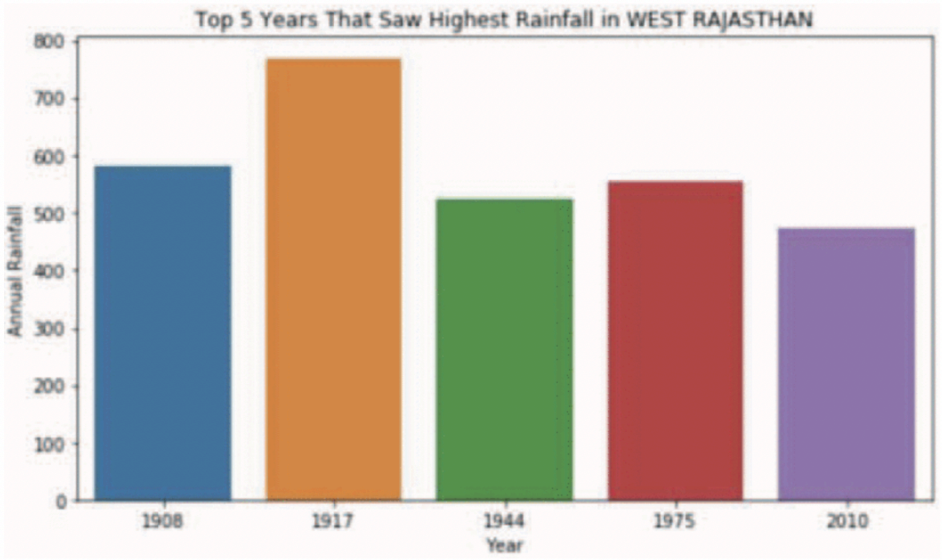
We also plan to use a User-Centered Design (UCD) approach. This means we will talk to users, test prototypes, and collect feedback regularly during development. Many existing platforms do not take this kind of user feedback seriously.

In the end, our goal is to build a weather data visualisation platform that is not just for experts, but also useful in real-world areas like education, farming, public services, and local planning.

2.5 Project Rationale Summary & Comparison to Studies/Tools

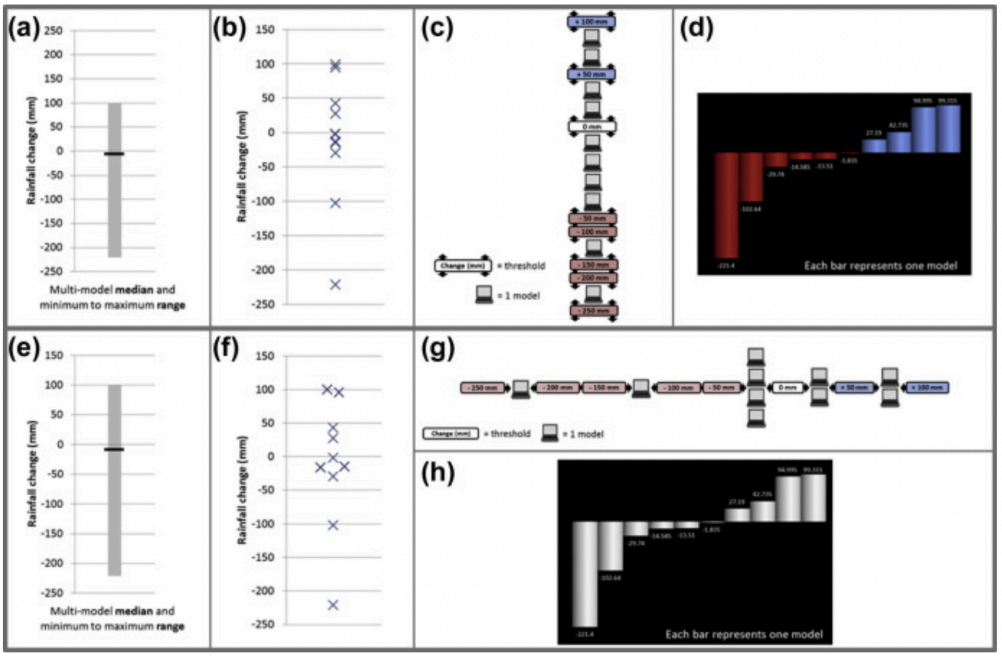


Much of the weather data used for analysis and reporting in Australia is sourced from the Bureau of Meteorology. The BOM provides daily and weekly forecasts for cities and towns across the country, often accompanied by maps and basic statistics. For example, as shown in the graphic above, the 7-day forecast for Melbourne includes daily temperature ranges and weather icons, offering a concise overview. While this format is effective for users seeking quick forecasts, it is less efficient when the goal is to compare trends across multiple regions or over extended periods. Currently, users have to manually select each location and navigate through separate web pages to perform comparisons. Our proposed project aims to streamline this process by creating a platform that enables users to view, compare, and interact with weather data from multiple regions within a single interface. Through the use of interactive maps and timeline controls, we aim to reduce the time and effort needed to draw insights from raw weather data. This allows users to make more informed and efficient decisions based on weather trends.



In a study from the 10th International Conference on Cloud Computing, Data Science & Engineering, researchers highlighted the role that data visualisation plays in understanding datasets. They concluded that visualising rainfall data “has greatly improved the data and improved the understandability” (Joshi et al., 2020), allowing for more accessible and accurate analysis by both novices and experts. An example from their study includes a rainfall graph that clearly shows changes over time, allowing researchers to easily identify the period with the highest rainfall. This immediate visual insight allowed them to link rainfall patterns to seasonal changes and make predictions for future trends. Ultimately, this study demonstrates the importance of effective data visualisation in the field of meteorology. Our project builds on this principle by providing a platform that makes spatial weather trends visually accessible and interactive, ensuring that users can quickly gain insights and apply them to make key decisions.

The importance of data visualisation in supporting analysis and decision-making is further emphasised by many studies. A study published in IEEE Transactions on visualisation and computer graphics investigates the role of visualisation techniques. Researchers highlight the importance in “clarity of 2D maps with respect to spatial perception and conveyance of quantitative information” (Rautenhaus et al., 2018). This aligns well with our project, which aims to prioritise clean and accessible presentations of weather trends to ensure a comfortable but concise user experience. Moreover, the article also talks about how “most interactive visualisation tools lack scalability”. As datasets grow in size and complexity, it is difficult to maintain usability. Our project will address this by adopting a supporting scalable data rendering, which will allow users to explore large datasets across various timeframes without sacrificing performance or clarity.



A journal titled *Climate Risk Management* discusses the uses of meteorological data visualisation and its implications in the real world. The study uses different visualisations to observe monthly rainfall data and across developing countries in Africa, to assess which one is most interpretable for users. After asking respondents to predict the likelihood of rainfall to be over 500m. The study concludes that “responses vary considerably for all visualisations” (Daron et al., 2015), meaning that certain users may be more inclined to understand different types of visualisations. Additionally, “with appropriate tools to process and visualise large-volume multiple-datasets, it is possible to exploit the inherent information” (Haase et al., 2000). This shows that it is necessary to implement a variety of visualisations in the project, rather than allowing users to display data as one type of graph. Furthermore, users should be able to switch between graphs to maximise understanding and draw relevant conclusions.

Previous studies also show weather-related data visualisation as a way of understanding environmental trends and making predictions. A study published in *Environment and Planning* investigated how individuals concerned about climate change respond to different types of meteorological visualisations. Participants were shown a variety of visual representations and their responses were assessed to see if different visualisations affect their perceptions. The researchers found that there was a “decrease in concern about the local natural environment” (Bishop et al., 2013) after participants were exposed to the visualisations. This finding suggests that the type of visualisation can significantly influence how users interpret the impact of environmental changes. This study highlights the importance of user-sensitive visual design, which our project aims to address by presenting weather trends in an intuitive way. By allowing users to interact with a variety of visualisations, our platform hopes to improve understandability with the data. Whether users are forming predictions or simply exploring environmental changes, the project aims to assist with more informed discussions and further studies.

Meteomatics is a company that provides advanced software and services for accessing satellite-based weather data. Meteomatics uses the latest technologies to deliver comprehensive weather solutions through multiple platforms, including an advanced weather API, interactive weather maps, and even weather drones which collect atmospheric data. These tools offer users access to a wide range of meteorological variables and historical trends. However, a key limitation of Meteomatics' current offerings is the lack of support for user-provided data inputs. While users can visualise large-scale atmospheric datasets from Meteomatics' services, there is no facility to upload and integrate their own weather-related dataset, for example CSV data export or manually collected observations. This restricts the tool's flexibility and prevents deeper customisation by individual users or researchers. Our project proposes to fill this gap by enabling users to upload their own datasets, which can then be easily integrated into the platform's visual interface. With just a few clicks, users will be able to upload a file and immediately view it on an interactive map. This promotes data exploration and makes the platform much more versatile and appealing for both everyday users and small-scale data collectors such as farmers or researchers.

Another weather visualisation platform is Visual Crossing, which provides access to historical and forecasted weather data through dashboards and spreadsheets. While its features are powerful, especially for developers and data analysts, a significant drawback is that the platform does not cater effectively to less experienced users. The interface can be difficult for those without a strong background in data interpretation or meteorology. For example, users must often configure detailed API queries or work with various data formats manually. This complexity can be a barrier for educators and students, or for general public users who are looking for quick insights rather than technical control. Furthermore, a South Africa based study concluded that it is important to "provide guidance for novices through an interactive environment where they can keep track of where they are" (Smuts et al., 2015) when using visualisation tools. Our project addresses this by focusing on ease of use, implementing visual aids such as time sliders and comparison modes. By simplifying how data is accessed and visualised, our platform aims to make weather analysis more inclusive and interactive.

In summary, our project addresses key gaps in current weather visualisation tools by improving data interactivity, supporting multi-source integration, and prioritising user accessibility.

3. Project Management Plan

3.1 Project Overview

This project focuses on the development of an interactive web-based visualisation platform for exploring Australian weather data. The platform will allow users to compare temperature, wind speed, and humidity across different regions and time periods through an intuitive interface. A full summary of the project's objectives and anticipated impact is provided in the conclusion.

The development process will follow a structured plan, progressing through key phases: requirements gathering and validation, detailed design, iterative implementation, testing, and

final deployment. Major milestones include delivery of a working prototype, integration of dynamic visualisation components, and completion of usability and performance testing. Throughout the project, regular team reviews and feedback loops will ensure continuous alignment with user needs and technical goals.

3.2 Project Scope

In the scope of the project, we will be defining the boundaries and priorities for the development of the interactive visualisation platform. It outlines what is included in the project deliverables, what is explicitly excluded, and identifies high and low priority items.

In Scope Items	Out of Scope Items

- Responsive web interface for data visualisation
- Integration of Australian weather datasets
  - e.g. Temperature, wind speed, humidity
- Interactive visualisation tools
  - e.g. Sliders, filters, comparison modes
- Backend services - for data querying and processing
- User authentication (incl. Role-based access)
- Deployment on a public hosting platform (e.g. Github)
- Basic testing and performance benchmarking
- Accessibility and cross-browser compatibility
	- Real-time data ingestion (or live data streaming)
	- Mobile app version
	- Advanced user analytics/dashboard (especially for administrators)
	- Offline usage
	- Local storage
	- Multilingual support

Priority Items

High

- Interactive charts and comparisons for temperature, humidity, and wind speed
- Clean, user-friendly web interface
- Accurate and efficient weather data processing

Medium

- Map-based visualisation features
- Role-based access or basic user login

Low

- Dark/Light mode toggle
- Exportable graphs or reports

3.2.1 Product Characteristics and Requirements

The following section will now outline the functional and non-functional requirements for the visualisation platform. These were gathered through an analysis of similar platforms (e.g. BOM, Ventusky, Windy.com), user preference research, team brainstorming session, and

informal feedback from peers and mentors - to gain an insight from typical users. Particular attention was given to providing functionality that supports exploratory data analysis and intuitive interaction.

Functional Requirements

- Display geospatial data through relevant visualisations
- Allow the user to manually input their own data or upload data from reliable datasets
- Display real time data from reliable sources
    - But also allow users to visualise/download data from different timelines
- Allow users to select one or more layers for visualisation
- Allow users to submit feedback about potential issues/improvements in the system
- Let users export visualisations in different formats
- Give the option to log-in or create accounts with 2FA to save and retrieve progress

Non-Functional Requirements

- System UI designed should comply with WCAG accessibility standard including visual aid support
- System should ensure platform compatibility across desktop and mobile devices

3.2.2 Product User Acceptance Criteria

To ensure the system meets both functional goals and user expectations, we have defined a comprehensive set of user acceptance criteria. These standards represent the minimum threshold for successful delivery and will be used to evaluate the platform during final testing (Week 11–12). The criteria are based on early-stage interviews with potential users (e.g., university students, non-technical peers), competitive analysis, and sprint reviews.

Functional Acceptance Criteria

No.	Criteria	Description
F1	Interactive Visualisation	Users can view spatial weather data (temperature, humidity, wind speed) over time and across regions via maps and charts.
F2	Multi-Source Data Support	System integrates BOM and OpenWeatherMap APIs for real-time data, and allows CSV upload from users.
F3	Filtering & Comparison	Users can filter by date, location, and weather type, and compare different regions visually.

F4	Time-Based Analysis	Timeline slider works correctly to reveal historical weather patterns.
F5	Data Export	Users can export map/chart visualisations as PNG or CSV files.

Usability and Experience Criteria

No.	Criteria	Description
U1	Ease of Use	A new user without technical background can complete basic tasks (filtering, uploading, viewing data) in under 5 minutes.
U2	Responsiveness	Platform functions correctly on desktops and mobile browsers (Chrome, Firefox, Safari).
U3	Interface Feedback	Users receive clear confirmation or error messages after major actions (e.g. upload success/fail, invalid input).
U4	Accessibility	UI adheres to WCAG 2.1 AA: includes alt text, keyboard navigation, sufficient contrast, readable fonts.

Technical & Security Criteria

No.	Criteria	Description
T1	Authentication	2FA-enabled login system is functional and secure.
T2	System Stability	Application runs without crashes or fatal errors

		across 3 supported browsers.
T3	Code & Documentation	All core components are version-controlled in GitHub with clean commit history and complete documentation.

User Feedback & Testing Criteria

No.	Criteria	Description
UF1	Feedback Loop	Users can submit feedback via an embedded form or GitHub Issues link.
UF2	Testing Coverage	Internal QA and 3 external test users complete assigned task checklists.
UF3	Bug Resolution	All critical bugs discovered during testing are resolved before submission.

Final acceptance of the product will be granted if at least 90% of the above criteria are met and validated through: peer-based usability testing, team QA review, and GitHub Issue log and checklist verification.

3.3 Project Organisation

3.3.1 Process Model

For the project we are adopting an agile methodology with iterative and incremental development. This will be particularly useful as our project involves an evolving understanding of the user's needs, continuous refinement of our UI/UX elements and the integration of diverse and dynamic components such as data, visualisation, and user interaction.

An agile methodology and life cycle approach will ensure we have short feedback loops, continuous improvement, and flexible planning. This works well for our group dynamic and the project's needs.

Development of the project will be split into four structured sprints, with each one focussing on the delivery of key functional units:

- Data Integration and basic visualisation

- Interactive UI and layer toggling
- User accounts, feedback, and data uploading
- Optimisation, testing, and export features

For each sprint, as a team we will plan, develop, do internal demos and review. Feedback from all team members (and peers as test users) will be used to guide each subsequent iteration.

The project management tools we will be using are:

- GitHub - Task management and tracking (as well as centralised hosting)
- Figma - UI prototypes and design
- Slack/Discord/Instagram/Zoom - Internal team communication
- Google Docs/Sheets - Collaborative documentation and requirement tracking

3.3.2 Project Responsibilities

Our team for this project consists of four members. Each member will be defined a role to ensure smooth execution and accountability. Where necessary, team members will support each other across tasks, but the core responsibility will have a clear lead.

Team Member	Role	Responsibility
Sam Chan	Data Integration Lead	- Lead development of data ingestion pipelines for historical and real-time weather datasets - Implement data cleaning, validation, and storage procedures - Ensure compatibility and consistency across different data formats and sources - Contribute to decisions on data and performance trad-offs
Xin Liu	Frontend, UI/UX, Testing	- Implement interactive visualisation components (e.g. charts, maps, sliders) - Design intuitive and responsive user interface using design tools and libraries - Conduct testing for functionality, usability, responsiveness, and accessibility - Ensure cross-browser compatibility and mobile responsiveness - Work with the data integration lead and backend developer to ensure seamless project transitions
Aryan Chugh	Project Manager	- Oversee overall project execution and ensure alignment with objectives and

		milestones - Facilitate sprint planning, retrospectives, and check-ins - Manage internal team communication and resolve roadblocks - Define and monitor project acceptance criteria and quality standards - Ensure timely submission of deliverables and version-controlled documentation - Track and mitigate project risks and scope changes
Harsh Sharma	Backend Developer	- Implement the APIs to serve processed weather data - Integrate backend logic with frontend components - Ensure secure user authentication with 2FA - Optimise server-side performance for real-time responsiveness - Write and maintain API documentation for frontend-backend integration - Conduct unit and integration testing for server-side components - Collaborate with data and frontend teams to ensure end-to-end data flow consistency

As a team we are accepting to meet fortnightly at a minimum for sprint standups and to assign any new tasks. These tasks will be tracked using GitHub, with each issue being linked to specific features, requirements, or a bug.

3.4 Management Process

This section outlines the risk management strategy, stakeholder communication plan, and project monitoring and control mechanisms. These components are critical for ensuring that the project progresses smoothly, risks are anticipated and mitigated, and deliverables meet expected standards of quality and timing.

3.4.1 Risk Management

We will follow a proactive and continuous process throughout the project lifecycle in terms of risk management. As a team we will identify potential risks during sprints and their planning sessions, analyse them based on the likelihood and impact, and implement appropriate mitigation or contingency strategies.

Risk Management Process:

1. Risk Identification
    - We will conduct this at the beginning of the project and revisit it at the start of each sprint
    - Our sources will be previous case studies, team and member experience, and technical constraints
2. Risk Analysis
    a. Each risk will be assessed by its likelihood and its impact
    b. Prioritisation will be done using a qualitative risk matrix
3. Risk Response Planning
    a. We will assign mitigation strategies for each high-priority risk
    b. Contingency actions for risks that cannot be prevented will be developed and defined
4. Monitoring and Review
    a. A centralised risk register will be used to track risks
    b. Status' reviewed weekly and updated when necessary

Some key risk categories will be project management risks; timeline delays, team member unavailability, technical risks; integration issues, API failures, and, external risks; third-party data source downtime, library deprecations.

3.4.2 Stakeholder Analysis and Communication Plan

Stakeholder Analysis:

Stakeholder	**Role**	**Interest/Influence**	**Involvement**
Project Team	Developers, Designers	High	Daily collaboration, responsible for all deliverables
Tutor	Supervisor / Guide	Medium	Reviews progress during class check-ins, provides feedback
Unit Coordinator	Approver	High	Reviews final submission and assesses deliverables
Test Users	Informal evaluators	Low-Medium	Help test usability and accessibility in final sprint

Communication Plan:

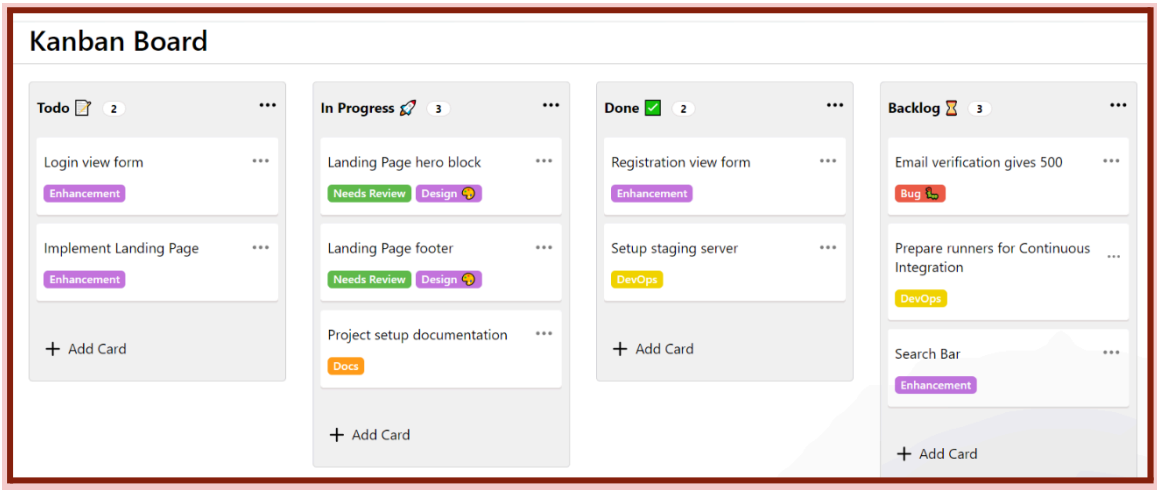
Communication Type	**Audience**	**Frequency**	**Method**	**Responsibility**
Sprint Standups	Project Team	Weekly	Instagram / Zoom Meet	Project Manager

Weekly Status Updates	Tutor	Weekly	Email / Instagram submission	Entire Team
Progress Review + Feedback	Stakeholders (Tutor)	Fortnightly	In-class demonstration	Entire Team
Usability Feedback	Test Users	Sprint 3–4	Online Form / Demo session	UX Lead and Entire Team

We will also make use of Github for task tracking and project updates.

3.4.3 Monitoring and Controlling Mechanisms

To make sure our project stays on track and meets both academic goals and user needs, we've set up several methods to monitor progress, manage risks, and control any changes. These strategies focus on team communication, technical tracking, and regular feedback.



1. **Weekly Sprint Progress Review**
   Every Weekend, our team will meet on Zoom for a 30-minute sprint check-in. We'll go through tasks on our GitHub board, which are labelled as "To Do," "In Progress," "Blocked," or "Completed." This helps us keep track of what's done, what's stuck, and what needs attention. We'll share a short meeting summary in our Instagram group chat and make task changes if necessary.

2. **GitHub Issue Tracking and Version Control**
   All tasks—whether a new feature, bug fix, or improvement—will be added to GitHub as issues. Each issue links to a branch and is assigned to a team member with a deadline. Pull requests must be reviewed by another team member before merging to the main branch, which helps keep our code clean and well-tested.

3. **Feature Checklists and Acceptance Tests**
   To make sure each feature works properly, we'll include a checklist on each GitHub issue. This will include items like:

- No errors on screen

- Works in Chrome and Firefox

- Updates the map correctly

- Meets accessibility standards

  One team member must go through the checklist and show the feature working during a team review.

4. **Mid-Sprint Peer Reviews**

   From Week 3 to Week 10, we'll have a mid-week check-in every Weekday. Team members will share their screen and show what they're working on (e.g., the filter tool or UI elements). This gives everyone a chance to spot bugs early, ask questions, and give helpful suggestions.

5. **Risk Register**

   Our Project Manager (Aryan) will keep a risk log updated every week. For each risk, we'll list how likely it is, how serious it could be, and how we plan to manage it.
   Example:

- Risk: Too many API requests

- Likelihood: Medium

- Impact: High

- Solution: Add a cache system (e.g., Redis) and limit user requests

6. **Tutor Feedback**

   We'll use the feedback given by our tutor during class check-ins. Each piece of feedback will be written in a Google Sheet and assigned to the right person. We'll also set deadlines and add related tasks to our GitHub board.

7. **Usability Testing – Week 12**

   In Week 12, we'll ask 4–5 users with no technical background to test the platform and fill out a feedback form. Our UX lead (Xin) will review their responses and turn important points into GitHub issues to fix in the final sprint.

8. **Final Quality Checks Before Submission**

   At the end of Week 12, we'll run a final test to make sure everything works. We'll check:
   - All features work as expected
   - Map loads in under 2 seconds
   - Meets basic accessibility standards
   - Works on multiple browsers
   We'll do a full demo as a group to check everything together before the final

submission.

9. **Scope and Change Control**
   Our scope (defined in Section 3.2) will be tracked by the Project Manager. If someone suggests a new feature (like a new animation), it must be discussed during a team meeting and written down as a change request. We'll review how it affects time and resources before deciding whether to add it.

10. **Final Reflection – Week 13**
    In the last week, we'll reflect on what went well and what we struggled with. Each team member will write a short review about communication, tools, time management, and technical work. This will be combined into a final "Post-Mortem Document" and submitted with the report.

3.5 Schedule and Resource Requirements

3.5.1 Schedule**3.5 Schedule and Resource Requirements**

3.5.1 Schedule

The table below outlines the project schedule broken into sprints. Each sprint includes a specific theme, key tasks, and expected deliverables. This detailed breakdown supports agile development and aligns with the Gantt chart provided in Appendix A.

Weeks	Sprint Theme	Key Tasks	Expected Deliverables
1–2	Planning & Design	Define project scope, finalise team roles, conduct literature review, create UI wireframes	Scope doc, Figma wireframes, team charter
3–5	Core Data & Mapping	Integrate BOM/OpenWeather API, render map layers, basic data processing	Working map, API demo, basic JSON data visualisation
6–8	UI Interactions & Uploads	Add filters, sliders, CSV upload, handle missing data, user feedback mechanism	Interactive UI prototype, upload validation

9–10	Testing & Refinement	Conduct usability testing, cross-browser checks, performance tuning	Usability report, bug-fix log, updated prototype
11–12	Final QA & Documentation	Polish UI, finalise accessibility (WCAG), write user manual and technical doc	Final documentation pack, deployment-ready version
13	Submission & Presentation	Submit project ZIP and report, prepare for class demonstration	Final GitHub ZIP, report, presentation slides

Note: A full Gantt chart visualising the above schedule is provided in Appendix A. Tasks are also managed and updated weekly using a GitHub Kanban board.

3.5.2 Resource Requirements

All tools, platforms, and APIs used in the project are free, available, and currently in use. No commercial licenses or paid tools are required.

Human Resources

Role	Member	Contribution Summary
Project Manager	Aryan	Sprint scheduling, communication, quality control
Frontend Developer	Xin	UI design, Leaflet maps, slider, filters
Backend Developer	Harsh	Data processing, upload module, API handling
Data Engineer	Sam	Weather API integration, data cleaning, source formatting

All members contribute to testing, documentation, and feedback analysis.

Tools & Technology

Tool/Platform	Purpose
GitHub	Version control, task management
Figma	Interface design
Leaflet.js	Web map rendering
Python + Flask	Backend logic, CSV parsing
Postman	API testing
Google Docs/Sheets	Collaborative planning and documentation
GitHub Pages	Hosting of final project prototype

Data & API Sources

Source	Use Case
Bureau of Meteorology	Real-time Australian weather data
OpenWeatherMap	Global weather metrics
User CSV Upload	Localised data visualisation support

All APIs are accessible via free academic tiers or open public access, ensuring no cost barrier.

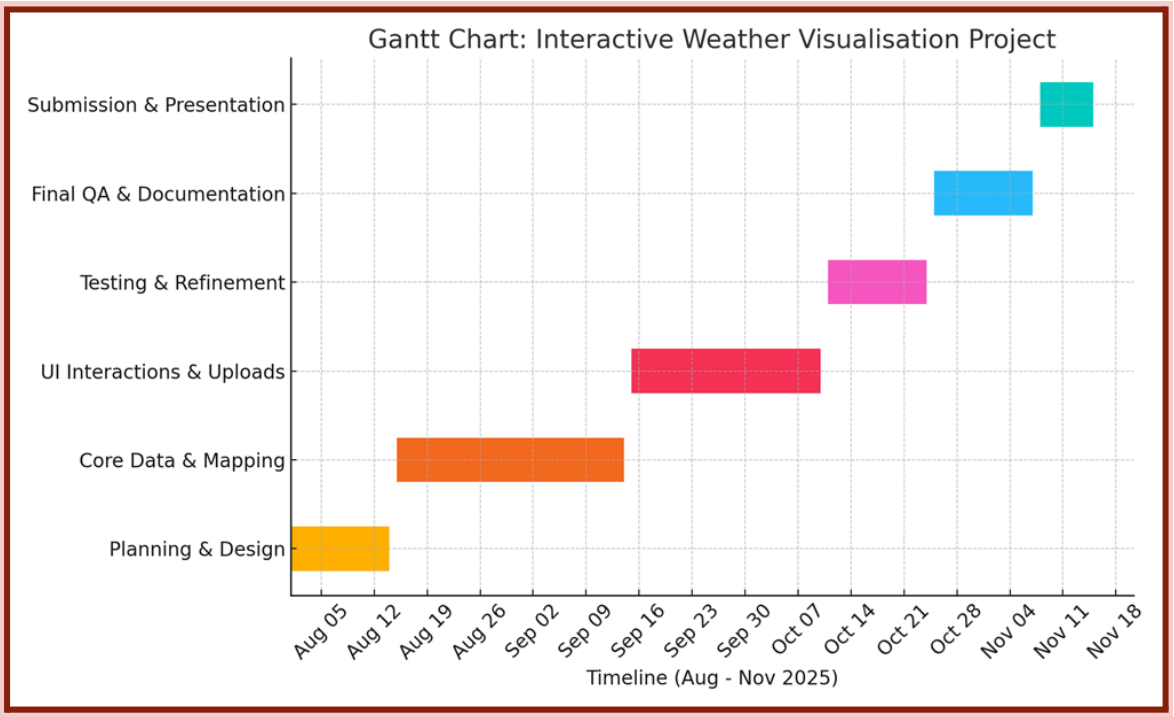
Summary

This project schedule is built around short development cycles and regular feedback, which helps us stay flexible and focused on what users need. We've planned our resources carefully so that everything we use—software, data, and tools—is free or open-source, which keeps costs at zero. Our weekly sprints help us track progress, fix issues early, and make sure we deliver the key features on time.

Appendix A: Gantt Chart

Figure A.1 – Gantt Chart: Interactive Weather Visualisation Project

This updated Gantt chart aligns with the actual academic calendar, beginning in August and concluding mid-November. It highlights the sprint-based structure, key milestones, and deliverable phases of the project.



Appendix B: Responsibility Matrix (RACI)

Task	Aryan (PM)	Xin (Frontend)	Harsh (Backend)	Sam (Data Eng)
Define project scope	R	A	C	C
Assign roles and team setup	R	C	C	C
UI wireframe (Figma)	C	R	A	A
Map rendering (Leaflet)	A	R	C	C
Weather data API integration	A	C	C	R
Data cleaning & preprocessing	C	C	A	R

Filter/slider design & testing	C	R	A	A
Backend logic (Flask)	A	A	R	C
CSV upload feature	C	A	R	A
Usability testing & bug reporting	A	R	R	R
Documentation & final report	R	A	A	A

Legend:

- **R** = Responsible
- **A** = Accountable
- **C** = Consulted

Appendix C: Work Breakdown Structure (WBS)

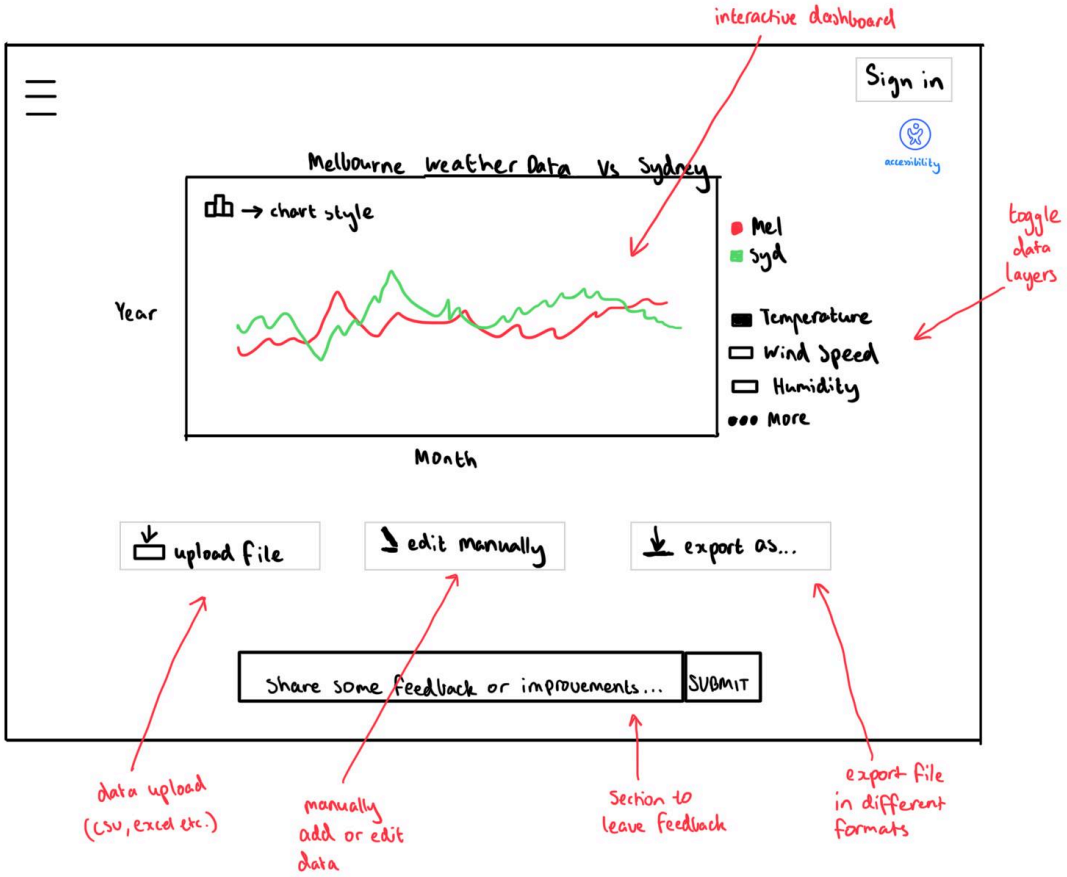
WBS ID	Task Name	Assigned To
1	Project Planning	All
1.1	Define project scope and goals	Aryan
1.2	Team role assignment	Aryan
1.3	Literature review and tool selection	All
2	System Development	Xin, Sam, Harsh
2.1	Map rendering with Leaflet	Xin
2.2	Weather data integration	Sam
2.3	UI/UX design and interactivity	Xin
3	Testing and QA	All
3.1	Usability testing	Xin
3.2	Performance and bug testing	Harsh
4	Deployment & Documentation	All
4.1	Final submission to LMS	Aryan
4.2	Prepare report and user guide	Sam

4. External Design

4.1: Overview

Our project aims to develop an interactive dashboard for weather data visualisation, that enables users to explore and compare weather data across spatial and temporal dimensions. The platform should enable users to analyze the different locations based on the visualisation and insights that we generated. Insights including temperature, wind speed and humidity patterns in different locations. All of these insights will be supported and shown on our dashboard with interactivity.
The system will support both real time and historical datasets.All of these datasets will get from the Bureau of Meteorology and Meteostat API or via users by uploading it to our database with common spreadsheet file extension. This platform allows both public and professional users to get access to more complex meteorological information by simplifying them.

4.2 User Interface Design



The main user interface is a responsive dashboard implemented in React, which is optimized for both desktop and mobile devices. An interactive map is rendered using Leaflet.js which enables spatial exploration instead of just focusing on few locations and for other generic visualisation graphs such as bar chart, line chart and others are plotted using

Chart.js and D3.js. Users can toggle between geospatial weather layers from the dashboard. This includes that the user can:

- Toggle between different weather variables (eg. wind speed, temperature)
- Adjust between different time frame such as predefined seasonal data or custom input from the users
- Able to upload personal datasets for further visualisation and deeper analysis such as comparison with other locations via CSV file or excel file extension
- Exports visualisation in formats such as PNG, JPG or others supported images formats for sharing
- Submit feedback or report bugs to the team via an embedded form.
- Access the dashboard without login but an account is necessary for datasets uploading.

The overall design of the dashboard complies with the WCAG 2.1 standard where it will support colour contrast for visual impairment and more.

4.3 Data Model

The backend uses PostgreSQL and PostGIS to store and query spatial weather data. The database will contain two core tables(entities) for storing data:

- stations : station metadata (station_id, name, lat, lon, altitude)
- observation: weather data metric (obs_id, station_id, datetime, temperature, wind_speed, humidity)

Indexes will be implemented on frequently queried fields for fast filtering and by using composite indexes of datetime and station_id to enhance the query performance.

PostGIS can spatially indexed geometry/geography types to enable fast map filtering, distance queries, and clustering operations.

4.4 Data Pipeline

The system pipeline is fully automated to complete the following tasks:

1. **Ingestion:** Weather data is fetched regularly from Bureau of Meteorology(BOM) and Meteostat API, or data uploaded by user
2. **Validation & Data Cleaning:** Data fetched will be checked for error and will be preprocessed the data by removing null value and outlier and ensuring that all data points are within an expected physical ranges (temperature withing -50 to 50 degree Celsius)
3. **Transformation:** All data is normalized and unit is standardized such as wind speed unit will be km/h. Geographic coordinates will be converted into spatial coordinates so that it can be used by PostGIS for mapping.
4. **Database:** Cleaned data will be inserted into the database and will be used for visualisation.
5. **API Access:** FastAPI is used to provide JSON/GeoJSON endpoint and make use of Redis Cache to enhance performance.
6. **Visualisation:** Chart.js and D3.js will be used to render data graph and spatial layer by getting the data from the database.

4.5 Technology Stack

1. **Frontend**: React, Leaflet.js
2. **Backend**: FastAPI (Python)
3. **Database**: PostgreSQL + PostGIS
4. **Data Analysis**: Jupyter, Pandas, GeoPandas
5. **Deployment**: Docker, AWS ECS, GitHub Actions
6. **Visualisation**: Chart.js, D3.js

This stack is chosen for its scalability, low difficulty in developing and performance wise in handling large amounts of data.

4.6 Data Analysis Summary

Before developing the system or ingesting data into the system, an exploratory data analysis was conducted to have a better understanding of the datasets' structures, quality and statistical properties. Two station datasets from the Essendon Airport and Viewbank were collected to have an analysis on the Melbourne city weather in the year of 2024.

4.6.1 Data Preprocessing and Cleaning

- Removed all high null columns and filtered out invalid data (eg. outlier) and remove row with missing data
- Verified that all the data point is within an expected ranges

After these steps above, there are still 354 records of data retained which is enough for further analysis and plotting visualisation

4.6.2 Insights

- Temperature Range: -2.1 to 40.2°C
- Heavy Rain (> 25mm): 2.26% for the whole year
- Wind Patterns: North and South West direction
- Seasonal Trend:
    - Winter: June and July
    - Summer: November and December

From the summary data above we can see that the typical ranges for temperature in Melbourne City. We can tell that there is quite less heavy rain as there is only 2.26% of heavy rain happening in Melbourne. The seasonal trend followed current trends where winter is around June and July and Summer is around November and December.

This analysis allowed us to make a better design for our system such as allowing users to view the data based on predefined seasonal selection and enhance the preprocessing mechanism to provide useful insight and correct visualisation for the users. But due to limitations to this EDA, the system should include more location for analysis and allow users to make comparisons between them.

4.7 Summary (Can add more for better understanding) :)

The overall design of the system provides a user-friendly and clear interface for users but at the same time able to process large amounts of weather datasets. All features designed satisfied all the requirements(functional and non-functional) in the RTB.

5. Methodology

5.1 Tools Used

Our project solution can be achieved using the following tools where each tool has their responsibilities on each small part which finally builds the three main components of the system which are the backend fetching and processing, map visualisation and graph plot.
The following tools that being used are:

- Frontend: React, Leaflet.js, Chart.js and D3.js
- Backend: Python 3.8+, FastAPI
- Database Management: PostgreSQL + PostGIS
- Data Processing: Jupyter Notebook, Pandas, GeoPandas
- Caching: Redis
- Deployment: Docker, AWS ECS
- CI/CD pipeline: GitHub

5.2 System Architecture

This section will contain both frontend and backend implementation. The system is built based on the client-server architecture where the user(client) sends requests to the servers to obtain data from the server (Jain, 2025), ability to process large amounts of datasets including the spatial data and generic data type(eg. weather metric). The core components that made up the systems are:

- **Frontend:** Web based dashboard that is built with React that handles user interactions which include data filtering for different layers of spatial data. The dashboard will display different types of visualisations which include the generic charts visualisation and an interactive map that allows users to interact with and make comparisons between the different locations. Leaflet.js is used to render spatial data into an interactive and use Chart.js and D3.js to generate generic charts for users to view, export and analyse. Additionally, users will be able to submit feedback or report any bugs found in the system via an input box. This is to allow users to communicate with the developer teams about their experience and use these feedback for further updates. These data will be sent to the backend via FastAPI.
- **Backend:** There is two main cores features that need to be achieved here:
    a. **Handle user interaction and Post Data:** By using FastAPI, the backend is able to handle any response coming from the user via the frontend which is the interactive dashboard. This includes processing temporal data filtering, handles user uploaded datasets and processes(include extraction) the user uploaded spatial data. All the process data will be stored in the database and posted to the frontend to update the dashboard in GeoJSON(/JSON formats.

GeoJSON format is for Leaftlet.js and JSON is for Chart.js and D3.js .The backend will need to handle feedback mechanisms and export requests from the users.

   b. **Handle fetched data:** By integrating different APIs (eg. Meteostat API) into the system to fetch data from these reliable sources. The backend will need to periodically fetch data from these sources and process them regularly to ensure that the amount of data will not slow the system performance or even crash the system due to data overload. All the processed data will be saved in the database and update the dashboard with the latest data.
   c. **Handle feedback logic:** Users are allowed to submit any feedback for improvement or make a bug report via feedback. The system accepts these data from frontend then saves these to the database for further analysis. All these feedbacks will be anonymous to protect the user's privacy.

- **Database:** By using PostgreSQL and PostGIS (spatial extension of PostgreSQL) to do data manipulation and data retrieving. All processed and cleaned data will be stored into the database via SQL queries. PostgreSQL is used to execute all the queries in manipulating the database. Using PostgreSQL over MySQL(generic version) provides a more structured format for storing the data and it can provide more complex queries. PostgreSQL is more new over MySQL which complies with the latest industry standards. PostgreSQL allows complex indexing and creates spatial queries with the help of PostGIS which allow fast data retrieval.
- **Caches:** Redis is used for handling all the cache saved and post it to the frontend if needed. All the frequently requested queries with their respective result will be saved to cache. An API is provided by Redis to efficiently manage cache. For example, when a user sends a query request to the frontend, Redis API will first check the cache before proceeding the request to the backend which directly gets the outcome from the database. By implementing a cache system will lower down the workload for the database and this can increase the efficiency and improve response time.
- **API Integration:** As mentioned above, there will be two types of APIs being used, one of them is for the system to process data and another type will be the API to fetch external data and ingest into the database. FastAPI and the Redis API will be used in the system to achieve some of the features such as cache management system. For the fetching API, it will be the API key provided by the sources we collect data from.
- **Security:** All data transition between the user and frontend will be encrypted using HTTPS. Based on (Sharma, 2025), HTTPS makes use of TLS security protocol for communication between browser and the website where TLS provides stronger and better security. TLS Handshake is introduced to ensure secureness and efficiency. For the user account registration and login, password will be stored into the database with hashing with salt. Two factor authentication will be advised to implement for better security.

5.3 Data Pipeline

This data pipeline helps the system to do data ingestion, cleaning, transformation and storage of both external data fetched or user given datasets. It supports real-time integration of weather data while maintaining the data quality, consistency and system performance.

1. **Ingestion:** Majority of the external data will come from two source which is:

- Bureau of Meteorology (BoM)
- Meteostat

A scheduled program will be used such as Airflow to fetch data from these sources via APIs key provided by them. The program makes use of HTTPS requests to fetch the data. The response from these sources will be in JSON or CSV format. A typical way to request from Meteostat using its API key will be something like this:

```sh
curl --request GET \
    --url 'https://meteostat.p.rapidapi.com/stations/meta?id=10637' \
    --header 'x-rapidapi-host: meteostat.p.rapidapi.com' \
    --header 'x-rapidapi-key: {key}'
```

The response will be in JSON format:

```json
{
    "meta": {
        "exec_time": 0.003,
        "generated": "2021-06-21 18:23:58"
    },
    "data": {
        "id": "10637",
        "name": {
            "de": "Frankfurt Flughafen",
            "es": "Aeropuerto de Fráncfort",
            "en": "Frankfurt Airport"
        },
        "country": "DE",
        "region": "HE",
        "identifier": {
            "national": "01420",
            "wmo": "10637",
            "icao": "EDDF"
        },
        "location": {
            "latitude": 50.05,
            "longitude": 8.6,
            "elevation": 111
        },
        "timezone": "Europe/Berlin",
        "inventory": {
            "model": {
                "start": "2018-01-28",
                "end": "2021-06-29"
            },
            "hourly": {
                "start": "1926-01-01",
                "end": "2021-06-20"
            },
            "daily": {
                "start": "1934-05-01",
                "end": "2021-06-18"
            },
            "monthly": {
                "start": 1934,
                "end": 2021
            },
            "normals": {
                "start": 1961,
                "end": 2020
            }
        }
    }
}
```


The above example is from the official website from Meteostat.

Minority of the data will be uploaded by the users. Users are allowed to upload general spreadsheets including file extensions of .csv and .xlsl. These files will then be parsed to the backend via Pandas library with the use of the built-in function such as read_csv() for reading csv file and read_excel() for excel file. This will change the structure of the data into a data frame. This step is compulsory before heading to the next step since the other steps are designed to take in data frame type of data. Hence, file verification can be used here to ensure that only permitted file types are uploaded.

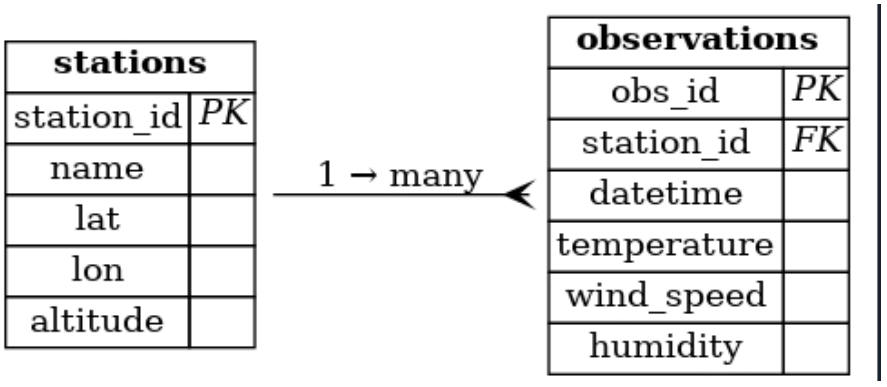
2. **Data Validation and Cleaning:** After ingestion is done, there will be a function that checks if all the data imputed have expected attributes and all the data points are within the expected range(such as temperature within -50 to 50°C) . The data will go through several steps to ensure that they are usable when generating the visualisation.
    a. Null value will all be removed using dropna() function to ensure that incomplete rows will be excluded from the datasets.
    b. Each of the attribute data types will be cast into a standard such as datetime64[ns] and others.
    c. Outlier removal: According to (Jain, 2024), Z-score will be used to detect outliers by setting a threshold value, any value higher than the threshold will be excluded after applying the Z-score formula to them.
3. **Data Transformation**
   This step is to standardize all the attributes and convert them based on a specific standard. The following will be the tasks to be done:
    - Convert all the current attributes' units into a standard unit such as all temperature related attributes will be degree Celsius(°C) and wind speed should be in km/h.
    - Convert all timestamp to UTC standard
    - Convert all geospatial data into structured data that the system can process using PostGIS

   Code example for geospatial data conversion from here:

```
-- Create a point in the WGS 84 geodetic CRS
SELECT ST_SetSRID(ST_MakePoint(-71.1043443253471, 42.3150676015829),4326);
```

4. **Database Insertion**
   After steps above, the data is ready to be inserted into the database. There are two primary tables which are:
    - stations : station metadata (station_id, name, lat, lon, altitude)
    - observation: weather data metric (obs_id, station_id, datetime, temperature, wind_speed, humidity)

   Below is a brief ERD diagram for these two tables which shows the relationships.



5. **Data Posting**
   This will be the final step in the pipeline where frontend accessing data via FastAPI and reflect it on the dashboard. The process data will be queried, aggregated if needed and posted to frontend in the form of JSON/GeoJSON. Commonly requested data will save to cache using Redis.

6. **Error Handling and Log Audition**
   This is a mechanism that is implemented throughout the pipeline which includes checking for errors while processing the data such as invalid file format uploaded or API failure during the process. All these error messages will be saved as system logs for later investigation. Auditing these logs can help in debugging and each process system logs will be saved for ease of debugging as developers can easily run through the log files and target which process is failing.

5.4 Deployment

The whole system is containerised using Docker. There are few reasons for using docker such as:

- Environment consistency: The environment of the internal system will always remain the same no matter the host environment (eg. Windows, MacOS or Linux) which ensure that the system will not re-setup the environment everytime running the program on different platforms.
- Dependencies Isolations: The libraries and modules are isolated within the container, meaning that this can reduce conflict if there is any file or folder with the same name as the dependencies.

A docker file will be used during the development and the container is built from Dockerfiles and deployed independently.

5.4.1 Deployment at AWS ECS

By using cloud infrastructure provided by AWS, the developer teams will be using ECS (Elastic Container Service) to deploy the container file (which is the system file) and this service allows the developer team to easily deploy and manage the application (*What Is Amazon Elastic Container Service? - Amazon Elastic Container Service*, n.d.). Since this is just a proposal, the setup for ECS will not be elaborate more but will be following this website for the deployment.

The developer team will be able to set up the cloud through initial configuration and manage the internal system including access and key rotation of the internal system via the portal provided by AWS.

5.5 Justification for Methodology

The methodology was chosen for its scalability and clarity. The architecture of the system allows each system component to work together and develop a solution to achieve the goal. FastAPI and Redis API are both used in the system for communication and cache management for both frontend and backend. PostgreSQL and PostGIS are used for setting up the database and ensuring that spatial data is capable of being saved and used. These components worked together to ensure efficiency and high performance. The data pipeline ensures clean, standardised and normalised data from the user input or external datasets. While the Redis API ensures that the cache system can be used for saving frequent query and data to improve response time and lower the workload for the backend. Deployment via AWS ECS ensures scalability and maintainability. By having this division, it will enable an efficient, reliable and user-centric weather visualisation system that is suitable for current implementation and future extension.

6. Test Planning

6.1 Testing Objectives

The goal of our testing plan is to ensure that all core functionalities of the weather visualisation platform operate as expected, are user-friendly, and remain stable under typical usage. Testing will focus on data accuracy, UI responsiveness, feature completeness, and user interaction flow.

6.2 Components to be Tested

Component	Test Focus
Interactive Map	Correct rendering of weather layers, zoom/pan behaviour, layer toggling
Time Slider	Smooth movement, accurate data change per step, correct time granularity
CSV Upload Module	File validation, feedback on incorrect structure, data display correctness
Filter Sidebar	Filter logic correctness, real-time update of map visuals
Cross-Browser Compatibility	Visual consistency and functionality on Chrome, Firefox, Edge
Responsiveness	Layout adapts on tablets and various screen sizes
API Integration	Proper handling of weather API responses, fallback mechanisms for failures

6.3 Testing Methods

- **Manual Testing**:
  Core feature walkthroughs during each sprint (map rendering, file upload, UI feedback)

- **Functional Testing**:
  Test weather layer toggles, time slider behaviour, upload and visualisation workflows

- **User Testing**:
  Invite 4–5 users (non-technical) to try the platform and provide feedback

- **Edge Case Testing**:
  Test with malformed CSV files, missing fields, large datasets, and empty API responses

- **Cross-Browser Testing**:
  Confirm consistent layout and behaviour across modern browsers

6.4 Tools and Resources

Tool	Use
Browser DevTools	Network/API monitoring, performance checks
CSV Validator (custom Python script)	For automated file testing
GitHub Issues + Labels	Track bugs and assign responsibilities
Figma Feedback	UI/UX improvement tracking from testers

6.5 Schedule and Integration

Week	Testing Activity
Week 9	Unit testing on map and time slider components
Week 10	Integration testing (map + filters + API)
Week 11	CSV upload testing with sample datasets
Week 12	Internal user testing + feedback round
Week 13	Bug fixing + final QA round before submission

6.6 Summary

Our test plan ensures that all major components of the platform are validated through a combination of manual, functional, and user-focused testing. By planning iterative test rounds throughout development, we aim to detect issues early and ensure the system remains usable, responsive, and reliable by the final demo.

Appendix A: Test Responsibilities Table

Team Member	Module Responsible	Testing Tasks	Tools / Methods Used
Aryan Chugh	Time slider, map logic	Functionality & edge-case testing	Manual tests, browser console
Xin Liu	UI, responsive design	Cross-browser testing, mobile responsiveness	Chrome DevTools, Figma mockups
Harsh Sharma	Backend (CSV & API)	File validation, API response handling	Python scripts, Postman
Sam Chan	Data integration & layers	Weather layer rendering, performance validation	Leaflet test setup, network monitoring
All members	Full system integration	End-to-end walkthroughs, bug tracking, regression	GitHub Issues, sprint testing

Appendix B: User Feedback Form (Week 12)

> To be completed by non-technical test users during our usability testing sessions.

Basic Info (Optional):

- Name/Nickname:
- Device/Browser Used:
- Technical Background: ☐ None ☐ Some ☐ Advanced

1. How easy was it to use the following features?

Feature	Very Difficult	Difficult	Neutral	Easy	Very Easy
Map zoom & pan	☐	☐	☐	☐	☐
Weather layer toggles	☐	☐	☐	☐	☐
Time slider navigation	☐	☐	☐	☐	☐
CSV upload	☐	☐	☐	☐	☐
Variable filtering	☐	☐	☐	☐	☐

---

2. Did you encounter any problems or confusion while using the system?
(Examples: Slow map loading, upload failure, unclear controls)

---

3. Which feature did you find the most useful, and why?

---

4. What do you think could be improved?

---

5. Overall rating (0–10):
☐ 0 ☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐ 6 ☐ 7 ☐ 8 ☐ 9 ☐ 10

7. Conclusion

Our project, *Interactive Visualisation of Spatial Data with Weather Data*, aims to solve the main problems found in existing weather platforms by offering a more interactive and user-friendly experience. Unlike some tools that are hard to use or limited in what they show, our platform makes it easier for people to explore both real-time and historical weather data through maps, filters, and simple controls. By combining open-source APIs, spatial mapping tools, and clean UI design, we created a prototype that makes complex weather information easier to understand for all kinds of users.

During the semester, we followed an agile development process with regular sprints and check-ins. This helped us manage our time, respond to feedback quickly, and stay organised as a team. From backend APIs to frontend visuals, every part of the system was built to show how data science can be applied in a real-world setting. We also made sure to include risk planning, testing steps, and acceptance checks to meet high standards for usability and performance.

Our platform is useful for many types of users—like students, teachers, planners, and researchers—who want to explore weather patterns across time and space. It lets people compare data, look for trends, and even upload their own files to see local changes. This makes it more flexible than most public weather tools.

Finally, because the system is modular and well documented, it can be improved in the future. We hope to add more features like mobile support, prediction tools, and data from more regions. Overall, the project not only meets its academic goals but also has the potential to make weather information more accessible and useful in everyday life.

References

A Step-by-Step Guide to Creating an AWS ECS Service. (2023, July 12). Medium. https://medium.com/@dghadge2002/a-step-by-step-guide-to-creating-an-aws-ecs-service-80f4527f38c

Australian weather and warnings information. (2025). Bom.gov.au; Bureau of Meteorology. http://www.bom.gov.au/vic/forecasts/map7day.shtml

Bishop, I. D., Pettit, C. J., Sheth, F., & Sharma, S. (2013). Evaluation of Data Visualisation Options for Land-Use Policy and Decision Making in Response to Climate Change. *Environment and Planning B: Planning and Design*, *40*(2), 213–233. https://doi.org/10.1068/b38159

Chaturvedi, M., & Shah, D. (2021). Leaflet vs. Mapbox: An interactive map framework comparison. GIS Technology Review, 9(3), 34–42. https://github.com/mapbox/mapbox-gl-leaflet

Daron, J. D., Lorenz, S., Wolski, P., Blamey, R. C., & Jack, C. (2015). Interpreting climate data visualisations to inform adaptation decisions. *Climate Risk Management*, *10*, 17–26. https://doi.org/10.1016/j.crm.2015.06.007

Goodwin, S., Dykes, J., & Wood, J. (2019). Geographical data visualisation research at the Monash IA Lab. Geographical Information Science, 20(2), 45–57. https://www.sciencedirect.com/science/article/pii/S2468502X2030067X

Haase, H., Bock, M., Hergenröther, E., Knöpfle, C., Koppert, H.-J., Schröder, F., Trembilski, A., & Weidenhausen, J. (2000). Meteorology meets computer graphics — a look at a wide range of weather visualisations for diverse audiences. *Computers & Graphics*, *24*(3), 391–397. https://doi.org/10.1016/s0097-8493(00)00035-2

Jain, S. (2024, June 17). *What is Outlier Detection?* GeeksforGeeks. Retrieved June 5, 2025, from https://www.geeksforgeeks.org/what-is-outlier-detection/

Jain, S. (2025, May 13). *Client-Server Model*. GeeksforGeeks. Retrieved June 4, 2025, from https://www.geeksforgeeks.org/client-server-model/

Joshi, Y. K., Chawla, U., & Shukla, S. (2020, January 1). *Rainfall Prediction Using Data Visualisation Techniques*. IEEE Xplore. https://doi.org/10.1109/Confluence47617.2020.9057928

Keim, D. A., Mansmann, F., Schneidewind, J., Thomas, J., & Ziegler, H. (2006). Visual analytics: Scope and challenges. In Information Visualization (pp. 76–90). Springer. https://doi.org/10.1007/978-3-540-70956-5_7

Rautenhaus, M., Bottinger, M., Siemen, S., Hoffman, R., Kirby, R. M., Mirzargar, M., Rober, N., & Westermann, R. (2018). Visualization in Meteorology—A Survey of Techniques and Tools for Data Analysis Tasks. *IEEE Transactions on Visualization and Computer Graphics*, *24*(12), 3268–3296. https://doi.org/10.1109/tvcg.2017.2779501

Sharma, S. (2025, May 20). *SSL, TLS, and HTTPS*. https://www.encryptionconsulting.com/ssl-vs-tls-vs-https/

Smuts, M., Scholtz, B., & Calitz, A. P. (2015). Usability guidelines for designing information visualisation tools for novice users. *Beyond development. Time for a new ICT4D paradigm*, 148-162.

The Global Leader in Weather Intelligence | Meteomatics. (n.d.). Www.meteomatics.com. https://www.meteomatics.com/

Weather Data Visualization – Visual Crossing Weather. (2024, March 11). Visualcrossing.com. https://www.visualcrossing.com/resources/blog/weather-data-visualization/

What is Amazon Elastic Container Service? - Amazon Elastic Container Service. (n.d.). AWS Documentation. Retrieved June 6, 2025, from https://docs.aws.amazon.com/AmazonECS/latest/developerguide/Welcome.html

Zhou, C., Xu, T., & Xie, Y. (2020). COVID-19: Challenges to GIS with big data. Geography and Sustainability, 1(1), 77–87. https://doi.org/10.1016/j.geosus.2020.03.005

Zhao, J., Nguyen, T., & Aris, S. (2022). An iterative approach toward development of ensemble visualization. Bulletin of the American Meteorological Society, 104(9), 1921–1935. https://journals.ametsoc.org/view/journals/bams/104/9/BAMS-D-22-0192.1.xml