WeatherNow

Software Requirements & Design Document

Nexel Software

Logo made with OpenAI ChatGPT

Nexel Software

Table of contents

List of Figures

List of Tables

1. Introduction

We are making a Weather system that allows users to see past, present and future weather data, by collecting and showing data from our own sensors and YR's API.

1.1 Purpose

This document is a combination of a System Requirements Specification (SRS) and a Software Design Document (SDD). The purpose of this document is to serve as a guide detailing the functional and non-functional requirements of the system. It is intended for project managers, developers and possibly some users, to make sure there is a clear understanding of the system architecture and design.

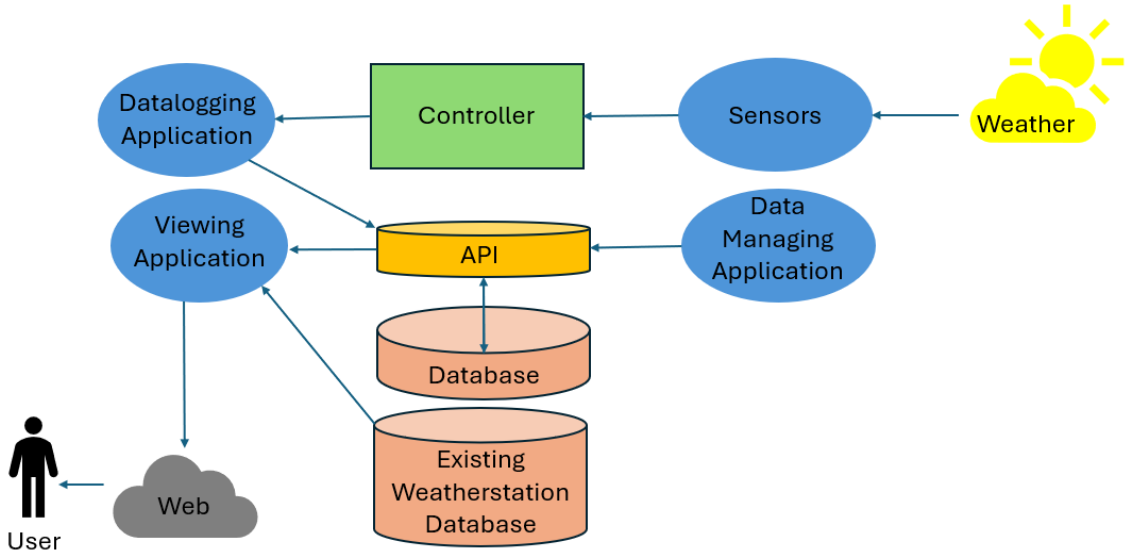

Figure 1.1 System overview - Sketch showing what each module in the project communicates with. Made with PowerPoint.

1.2 System Overview

The system consists of three main parts:

1. Sensor data logging: the module will gather the data captured by the sensors and store it in the database according to the type of data being gathered by the sensors.
2. Data storage: the data gathered from the sensors and the data gathered from the public API from "YR.no" will be processed, organized, cleaned and stored in a relational database.
3. Viewing module consists of two parts, one publicly available and the other only visible to administrators.
    a. Data management: the viewing module will have a login available for administrators to access the data stored in the database providing an easier way to manage the database and the information within.

b. Viewing data: this module will provide the end user with the means necessary to select what data they want to see at any given time and comfortably read the selected data.

2. Project description

2.1 Product Perspective

The weather logging application will directly interact with the sensors so that it can collect data in real-time. These sensors will be connected through a DAQ device and directly into the computer that's running the data logging application.

The System will also pull weather forecast data from the public data provided by “yr.no”. This data will be integrated with our own data which will enhance the users experience.

2.2 System Interfaces

Data logging application Interfaces:

- **Login Screen:** This interface will display fields for username and password, and a button for logging in after the fields have been filled in. When the user successfully manages to log in they will be redirected to the next page.
- **Data Logging Screen:** This screen is the main part of the data logging application. You will be able to select the chosen interval for how often you will log the data and what type of data you are collecting. You will then be able to start the data logging, and then stop it when required.

Viewing Application Interfaces:

- **Login Screen:** This Interface is an optional screen that appears if you chose to login while using the application. To login you will need an admin user since the only things that will change are things designed for the administrator.
- **User Data Viewing Screen:** This screen will be what appears first when opening the application. The user will be able to select what type of data they will want to display and then it will appear on the screen. There will also be an option to open the login screen for administrators.

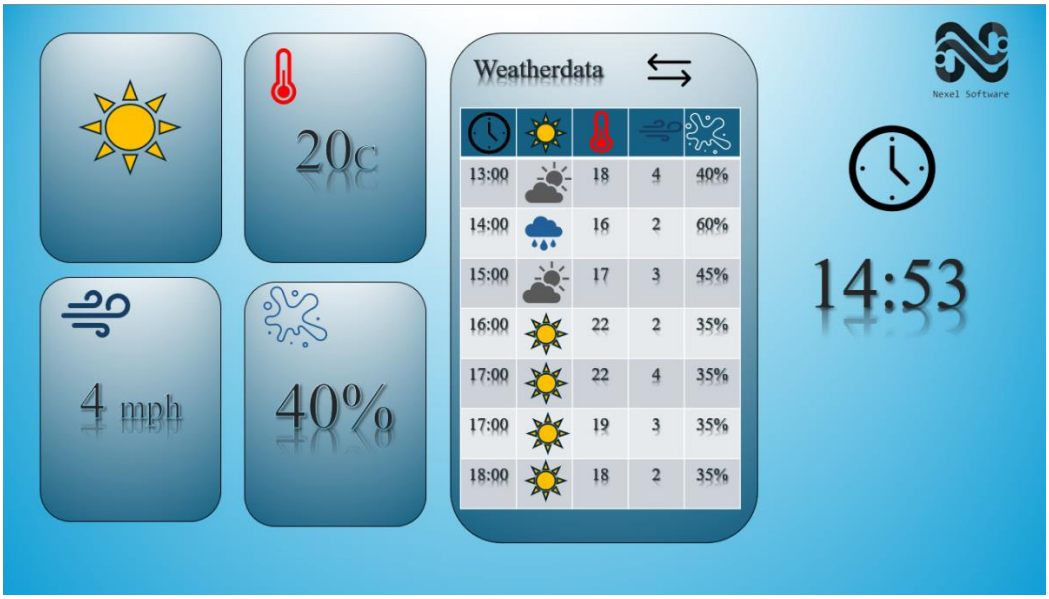

Figure 2.1 User Data Viewing Screen. This is the screen normal users will be using in the viewing application. Made with PowerPoint.

- **Admin Data Viewing Screen:** This screen will be what appears if you successfully login with an admin user in the User Data Viewing Screen. This screen will have all the features of the previous screen but a few key differences.
    - The Admin will be able to select some data that they would like to edit and then choose a new value for it and then change it or simply delete it.
    - They will be able to add new data in the database but in that case they will need to select what type of data it is, the value of the data and a timestamp for it.
    - The last feature would be to create or remove Admin users. This will be useful if somebody joins or leaves the team.

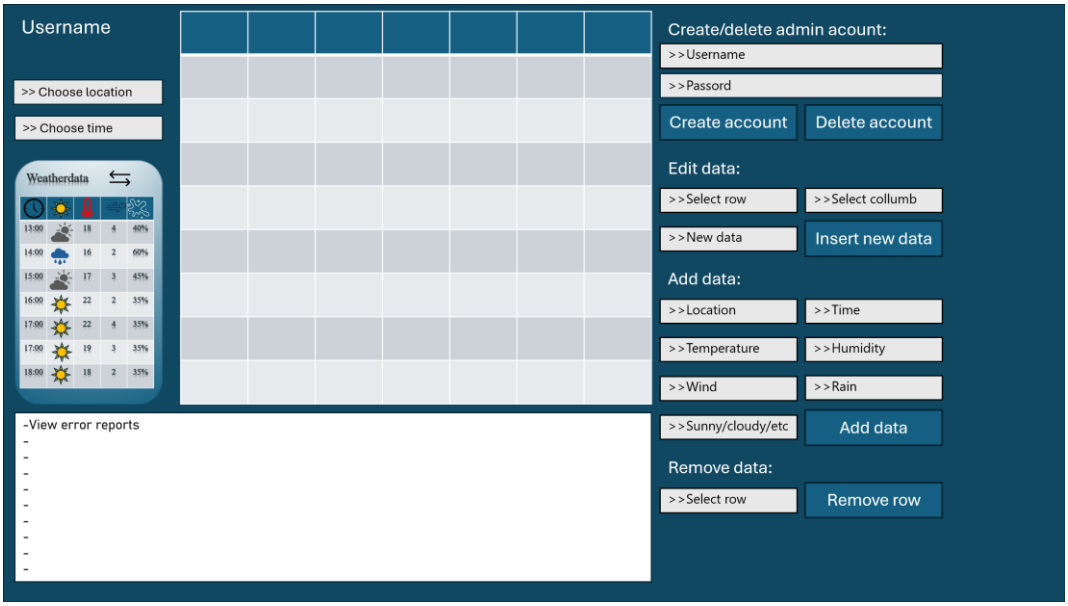

Figure 2.2 Sketch of the administrator screen where administrators can view and manage data on the database. Made with PowerPoint.

The Viewing Application will be the only application with screens that change depending on if the person using it is an administrator or not. If the User chooses to login and successfully does that then the screen will change to a pretty similar screen but with additional administrative features, please check chapter 2.1.1 for more details.

2.3 Constraints, Assumptions, and Dependencies

- List any constraints (technical, business, etc.), assumptions made during development, and dependencies on other systems or components.

Constraints:

- Weather prediction is quite a complicated process which involves numerous professionals that has studied that field to be able to predict. This is why we will integrate predictions from “yr.no” in our application. This means that our application by itself won’t do much in the prediction aspects, but it will display it nonetheless.
- Our budget isn’t that large, so our material limitations are mainly restricted to stuff in the classroom and stuff we have at home. We are able to but something if we really need to but I don’t think this will actually be a problem, since we already have most of the things we will need.

Assumptions:

- We will assume that it's no problem using the public data displayed in the API for "yr.no". The information is public so this should be no problem, and we will be showing sources for the information shown if it's not our own.

Dependencies:

- The system relies on the YR.no api for weather predictions. If the API changes or becomes unavailable for some reason, the system will no longer be able to display future weather data. The system will store some weather information that has been imported so it will work if the system goes down temporarily but not for longer periods. The system will notify the team in case there has been an error for a long period of time.

3. System Requirements

3.1 External Interface Requirements

- Detail the specific requirements for external interfaces, including formats, protocols, and communication standards.

The project contains four major areas where interface communication is crucial to maintain a correct data transmission.

Table 3.1. Sensor and DAQ to Raspberry PI

Communication protocol	The best communication would be through Universal Serial Bus due to it being a reliable and well-known communication medium between devices, this way we could avoid dropping information packets while using Wi-Fi or Bluetooth.
Data format	JSON will be the chosen format due to the ease of readability and parsing.
Error handling	Whenever possible we will try to include CRC checks and ACK packets to ensure the integrity of the data being transmitted.

Table 3.2. Raspberry Pi to Database

Communication protocol	The communication protocol will be through an API.
Data format	Data format for the data transmission will be JSON which will be processed by the API to send to the database.
Error handling	We will use exception handling and detailed logging to help us recover from any unforeseen data transmission interruption.

Table 3.3. Database to web interface

Communication protocol	For the web interface we'll be using GET/POST requests in HTTP/HTTPS to receive the data in a fast way.
Data format	JSON
Error handling	Most of the error will be handled Server-side to avoid load on the client browser and focus the processing power from the client browser only on displaying information.

3.2 Functional Requirements

The logging application must be able to save the data we're measuring in a chosen interval and send it to the API to store it in the database.

The logging application should validate the new data being stored and data editing, it also needs a high level of security since you could alter stored data and negatively influence the predictions, the shown data and modify administration permissions.

The data management application should allow the administrators to edit information regarding weather data and manage administrative permissions, all data editing will be done through the data management application but will be handled by the API to ensure consistency in the database.

The viewing application should be able to display the chosen data in a systematic and organized manner, querying the API for the necessary data.

The web interface should be available on most major web browsers.

The API will handle all communications from the logging, viewing and data management applications with the database ensuring a consistent and centralized data access/storage, allowing us to create a secure single point of communication instead of three separate security checks from each application to the database.

3.3 Non-Functional Requirements

The logging application must be capable of working uninterruptedly despite possible data transmission losses, power outages or connection drops, picking up the tasks from the last point of validated entries.

The data management application should allow administrators to log into the application and allow several levels of data management depending on their clearance level.

The viewing application should only load what it currently has to display or is using in the background to have minimal loading time and processing power. The user interface should be simple and intuitive to use, providing the user with a comfortable experience while navigating the application while also being able to know what they're looking at without needing to be taught about the application.

The viewing application should use some form of secure communication like HTTPS and have a clean code to ensure an adequate maintenance can be maintained.

Table 3.4, Organized table of functional and non-functional requirements

Application	Requirement	Functional	Non-functional	Technical
Hardware	Has temperature and humidity sensors			
	Good cable management			
Logging	Store data in database.			
	Data storage validation.			
	Data editing validation.			
	Administration permission validation.			
	Sensor data validation.			
	API communication.			
	Robust system.			
	Efficient resource use.			
	Sensor interconnection with USB-6008.			
	USB-6008 to database connection.			
Data management	Allow administrator login.			
	Edit weather data.			
	Edit administrator permissions.			
	Raspberry Pi data storage controller.			
	API communication.			
	Batch/single data handling.			
	Sensible data protection.			
	Secure communications.			
	Logging and editing history.			
	Clean code.			
	Display weather data.			

Viewing application	Support for most major browsers and devices			
	Minimal data loading.			
	Simple and intuitive design.			
	Secure communications.			
	Clean code.			
API	Simple database communication.			
Database	Well organized			

3.4 Use Cases

Table 3.5, Table presenting the "Viewing current weather" use case.

Viewing current weather		
Description	The user wishes to view current weather conditions in a specified place.	
Pre-conditions	Do not need to log in	
Post-conditions	Nothing	
Main flow	User enters website	
	Chooses a location	
	Views the data	
Alternative flows	Data won't load	User enters website
		Chooses a place
		Error message appears

Table 3.6, Table presenting the "Viewing past weather" use case.

Viewing past weather		
Description	The user wishes to view past weather conditions in a specified place and time.	
Pre-conditions	Do not need to log in	
Post-conditions	Nothing	
Main flow	User enters website	
	Chooses a location	
	Chooses a time	
	Views the data	
Alternative flows	Data won't load	User enters website
		Chooses a location
		Chooses a time
		Error message appears

Table 3.7, Table presenting the "Viewing future weather" use case.

Viewing future weather		
Description	The user wishes to view a forecast of future weather conditions in a specified place and time	
Pre-conditions	Do not need to log in	
Post-conditions	Nothing	
Main flow	User enters website	
	Chooses a location	
	Chooses a time	
	Views the data	
Alternative flows	Data won't load	User enters website
		Chooses a location
		Chooses a time
		Error message appears

Table 3.8, Table presenting the "Logging in" use case.

Logging in		
Description	The user wishes to view a forecast of future weather conditions in a specified place and time	
Pre-conditions	User is an administrator	
	login page is accessible	
Post-conditions	User is redirected to administrator screen	
Main flow	User enters the login page	
	Enters their username and password	
	Clicks the "Login" button	
	System verifies the entered credentials	
	System redirects them to the administrator site	
Alternative flows	Invalid Password	User enters the login page
		Enters their username and password
		Clicks the "Login" button
		System detects that the entered password is incorrect
		System displays an error message
		The user has the option to re-enter their credentials or click "Forgot Password"
	Forgotten Password	User enters the login page
		Clicks on the "Forgot Password" button
		When pressed, it will ask for the login information, validate the input and notify the user when the input has been admitted.

Table 3.9, Table presenting the "Making administrator account" use case.

Making administrator account			
Description		The user wishes to make an administrator account	
Pre-conditions		User is an administrator with high enough clearance	
		New username us unique	
Post-conditions		New account is made and registered	
Main flow		User logs in	
		Writes the necessary data in their respective text boxes (see figure 2.2)	
		Clicks the "Create account" button	
Alternative flows	User input cannot be converted to correct format/data type	User logs in	
			Writes the necessary data in their respective text boxes (see figure 2.2)
			Clicks the "Create account" button
			The system fails to convert the data
			Error message appears, explaining the problem and asks the user to try again

Table 3.10, Table presenting the "Deleting administrator account" use case.

Deleting administrator account		
Description	The user wishes to delete an administrator account	
Pre-conditions	User is an administrator with high enough clearance	
Post-conditions	Chosen account is deleted	
Main flow	User logs in	
	Writes the necessary data in their respective text boxes (see figure 2.2)	
	Clicks the "Delete account" button	
Alternative flows	User input cannot be converted to correct format/data type	User logs in
		Writes the necessary data in their respective text boxes (see figure 2.2)
		Clicks the "Delete account" button
		The system fails to convert the data
		Error message appears, explaining the problem and asks the user to try again
		User logs in

	Input username and/or password is incorrect	Writes the necessary data in their respective text boxes (see figure 2.2)
		Clicks the “Delete account” button
		The system cannot find the user in the database
		Error message appears, explaining the problem and asks the user to try again

Table 3.11, Table presenting the “Editing stored data” use case.

Editing stored data			
Description		The user wishes to edit the data in our database	
Pre-conditions		User is an administrator with high enough clearance	
Post-conditions		The changes are saved	
Main flow		User logs in	
		Writes the necessary data in their respective text boxes (see figure 2.2)	
		Clicks the “Insert new data” button	
Alternative flows	User input cannot be converted to correct format/data type	User logs in	
			Writes the necessary data in their respective text boxes (see figure 2.2)
			Clicks the “Insert new data” button
			The system fails to convert the data
			Error message appears, explaining the problem and asks the user to try again

Table 3.12, Table presenting the “Adding data to the database” use case.

Adding data to the database		
Description	The user wishes to add data to our database	
Pre-conditions	User is an administrator with high enough clearance	
Post-conditions	The new data is saved	
Main flow	User logs in	
	Writes the necessary data in their respective text boxes (see figure 2.2)	
	Clicks the “Add Data” button	
		User logs in

Alternative flows	User input cannot be converted to correct format/data type	Writes the necessary data in their respective text boxes (see figure 2.2)
		Clicks the “Add Data” button
		The system fails to convert the data
		Error message appears, explaining the problem and asks the user to try again

Table 3.13, Table presenting the “Removing data from the database” use case.

Removing data from the database		
Description	The user wishes to remove data from our database	
Pre-conditions	User is an administrator with high enough clearance	
Post-conditions	The chosen data is removed	
Main flow	User logs in	
	Writes the necessary data in their respective text boxes (see figure 2.2)	
	Clicks the “Remove row” button	
Alternative flows	User input cannot be converted to correct format/data type	User logs in
		Writes the necessary data in their respective text boxes (see figure 2.2)
		Clicks the “Remove row” button
		The system fails to convert the data
		Error message appears, explaining the problem and asks the user to try again
	Input row number does not exist	User logs in
		Writes the necessary data in their respective text boxes (see figure 2.2)
		Clicks the “Remove row” button
		The system cannot find the given row
		Error message appears, explaining the problem and asks the user to try again

3.6 Security and GDPR

The following are some point we will be focusing on for security in the system. This will be to protect personal information and ensure that the system is working according to the General Data Protection Regulation(GDPR).

- The communication between the client and server, mainly the login credentials will be encrypted using SSL/TLS.
- User passwords will be hashed and salted to prevent anybody from accessing the password.
- Anybody can request for having their information stored in the system to be deleted, according to the GDPR.

4. System Sketches

4.1 UML Diagram

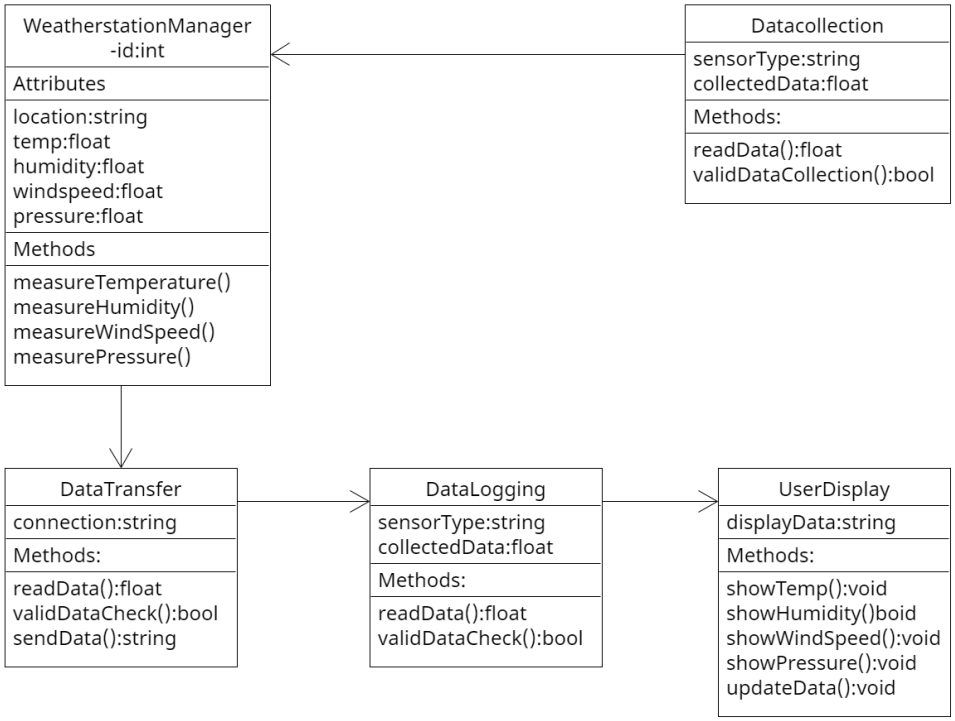

Figure 4.4 UML Diagram made in UMLetino