WeatherNow

Software Development Plan

Nexel Software

WeatherNow 1

Software Development Plan 1

1. Introduction 4

1.1 Project overview 4

1.2 Objectives 4

1.3 Scope 4

2. System Requirements 5

2.1 Functional Requirements 5

2.2 Non-Functional Requirements 5

2.3 Technical Requirements 6

3. Software Architecture 8

3.1 Components 8

3.2 Data Flow 9

4. Development Plan 10

4.1 Development Methodology 10

4.2 Project Phases 10

4.3 Milestones and Deliverables 11

4.4 Schedule 12

4.5 Next Steps 13

5. Team Organization 14

5.1 Team Structure 14

5.2 Responsibilities 14

6. Risk Management 15

6.1 Risk Identification ..... **Feil! Bokmerke er ikke definert.**

6.2 Risk Reduction ..... **Feil! Bokmerke er ikke definert.**

7. Quality Assurance 16

7.1 Testing Strategy 16

7.2 Testing Tools 16

7.3 Bug Tracking 16

8. Deployment Plan 17

8.1 Deployment Strategy 17

8.2 Maintenance 17

9. Documentation 18

9.1 User Documentation 18

9.2 Technical Documentation 18

1. Introduction

1.1 Project overview

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.

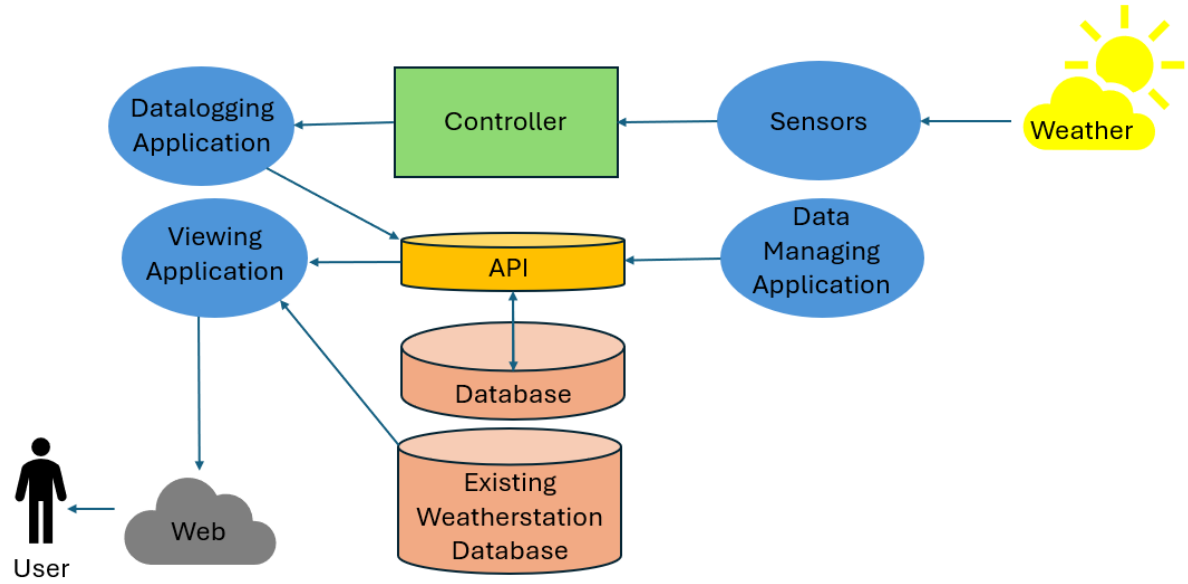

Figure 1.1 System Sketch – Sketch showing how each module communicates, and what they communicate with.

1.2 Objectives

We aim to make an application that shows the weather that has occurred in a systematic and informative way. This would help in mapping the change in the environment and predict the weather in the future. There will also be two other applications for storing and editing the data we collect.

1.3 Scope

The project will save a diverse amount of data such as temperature, humidity, wind speed and light amount. It will display the data in an informative way for any application you could need it for, and we will integrate data from "YR.no" to display some data about future conditions.

2. System Requirements

2.1 Functional Requirements

The logging application should 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.

2.2 Non-Functional Requirements

The logging application should 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.

2.3 Technical Requirements

For the Logging application we will need to use a module to connect the sensors which will most likely be a USB-6008 device. We don't have a sensor that measures all the data we need so we will need separate sensors for each type of measurement, except for temperature and humidity which we have a combined sensor for.

The project will consist of a logging application, data editing application and a viewing application, and they will be connected to a database through an API.

Table 2.1 Technical requirements overview

Application	Requirement	Functional	Non-functional	Technical
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.			
Viewing application	Display weather data.			
	Support for most major browsers and devices			
	Minimal data loading.			
	Simple and intuitive design.			
	Secure communications.			
	Clean code.			
API	Simple database communication.			

3. Software Architecture

The software architecture 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.

3.1 Components

The main components of the system are the following:

- Logging Application
- Viewing Application
- Data Management Application
- API
- Database

The Logging Application will be able to save the data were monitoring with the DAQ device in a chosen interval. All the data will be transferred over to the database with the help of the API.

The Data Management Application will be able to edit the data that has been stored in the database.

The Viewing Application will be able to receive the chosen data from the database and monitoring it. You won’t be able to do anything else than viewing through this application.

The API will consist of functions in C# and views/stored procedures in the database. These will make it significantly easier to communicate between the different modules.

The database will just be saving the data from the logging application and then display the data in the viewing application. It will also be able to edit the data through the logging application in case that's needed.

3.2 Data Flow

The Logging application collects data from the DAQ module, sends it to the database in a chosen interval and then it gets stored there. When somebody selects some data in the viewing application it gets requested from the database and displayed in the viewing application. If an authorized administrator wants to edit some of the stored data, they open the editing application, select the data they want to edit and choose a new value to update the existing one in the database. External data from "YR.no" is fetched via the API and merged with the system's predictions.

4. Development Plan

4.1 Development Methodology

We've chosen to implement the Waterfall development methodology for this project because it aligns well with our working style and offers key benefits. Waterfall's structured approach allows us to break the project into distinct phases, making it easier to track progress and ensure each stage is completed with attention to detail. Waterfall helps us minimize risks and ensure a high-quality final product.

4.2 Project Phases

Table 4.1, Table of phases

Phases
Planning
Programming phase
Implementation
Testing
Deployment and Maintenance

4.3 Milestones and Deliverables

Identify key milestones and deliverables for each phase. What are the critical deadlines?

Table 4.2, table of project phases and milestones

Planning phase	Choose a project
	Brainstorm
	Make software development plan
Programming phase	Make database
	Make API
	Make logging application
	Make data collection from YR API program
	Make data collection from sensor program
	Make website
	Make GUI
	Make monitoring application
	Make data management application
Implementation phase	Working alpha version
Testing phase	Identify bugs and security risks
	Fixed all identified bugs and security risks
	Ready for deployment
Deployment and Maintenance phase	Successful deployment

4.4 Schedule

Table 4.3, table of schedule

Phase	Start date	End date
Planning phase	15/08	26/08
Programming phase	26/08	03/10
Implementation phase	03/10	07/10
Testing phase	07/10	18/11
Deployment and Maintenance phase	18/11	24/11

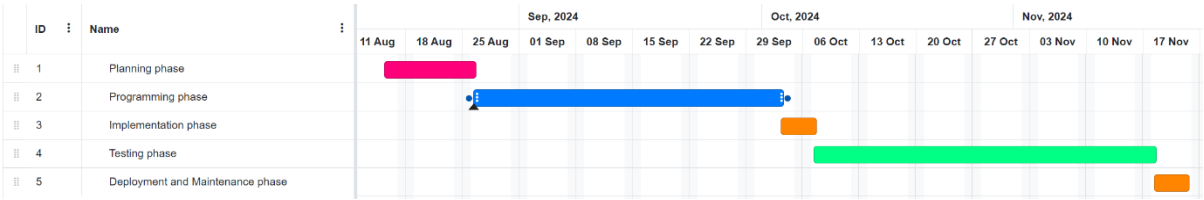

Figure 4.1 Gantt diagram – Overview of the allocated time for each step in the project.

4.5 Next Steps

The next steps involve starting to actively develop the project. The team will review the requirements for the project and ensure that we understand what we are supposed to create. If any hardware requirement hasn’t yet been met in this phase of the project, we will acquire the necessary hardware to continue with the next steps, otherwise the project will proceed to the creation of the database.

For the creation of the database, we will follow the design scheme of the relational databases to ensure it meets the requirements, once the database design is finished, we will be able to create the other applications and the API.

During and after the coding there will be several tests to ensure that they work as intended, and control that it is what we envisioned.

The documentation will be made during the design and coding phase to ensure we can include the decision-making process, issues that could arise and the solutions implemented or changes in the technology us When we’re happy with the project we will start the documentation and deploy the project to an online system so that others can use it. After deployment there will probably appear some bugs that haven't been dealt with so the team will continue working with the system until it works as intended.

5. Team Organization

5.1 Team Structure

The team members working on the project are the following:

Table 5.1 Team members and their roles.

Team Member:	Role:
Aleksander Kolstad	Developer
Emiel L. Honningsvåg	Team Leader and Developer
Emil Christopher Uleberg	Developer
Manuel Contrereas	Developer

Emiel acts as the Team Leader and delegates the tasks to each team member thar week. This means that Emiel will work as Team Leader about 20% of the time and Developer 80% of the time.

5.2 Responsibilities

Table 5.2 Team Roles and their responsibilities.

Role:	Responsibility:
Team Leader	Makes sure that everyone knows what they’re supposed to do that day and delegates tasks.
Database Designer	Designs and manages the database.
Frontend Developer	Designs the GUI and makes it convenient and intuitive.
QA Engineer	Tests applications and ensures good security.
Lead Coder	Leads the coding process.

6. Risk Management

Here we identify the risks and their severity and find solutions that minimise their risk.

Table 6.1, Risk matrix

	Negligible	Minor	Moderate	Significant	Severe
Very Likely					
Likely					
Possible					
Unlikely					
Very Unlikely					

Table 6.2, Table of risks, their severity and solutions

Risk	Description	Solution
Very Low	The end times arrive	Pray
Low	Missing essential hardware	We have addressed our hardware limitations by using sensors we owned beforehand and collecting data from the YR API
	YR servers shut down	Pray
Medium	A team member gets sick for an extended period	Go to the doctor
High	USN servers shut down	Pray
	We lose access to The Database, codebase, or website	Keep backups of everything
Very High	Emiel doesn't assign us tasks	Reprimand him

7. Quality Assurance

7.1 Testing Strategy

Unit Testing: Each module of the software that we have will be tested separately, where we will test if they work as we intendent.

Integration Testing: After we have performed the unit test, we will test the modules combined to ensure that they work together as we intended.

System Testing: The project will be tested in a virtual environment so that we can control that it meets the requirements. This will ensure that the project doesn't only work on the computer it was tested on.

User Acceptance Testing: We will gather a small group of users that will test the project and gather their feedback. This will ensure that the project works at they intended and if there are some things that will need to be adjusted.

Regression testing: If some things had to be adjusted then we would need to ensure that no new bugs have been created, so if nothing has been changed then this step will be skipped.

7.2 Testing Tools

Virtual Test Environment: VMware Workstation Player will be used for the Virtual Test Environment.

SQL Server Testing Tools: SQL SERVER Express and SQL Server Data Tools will be used to test the stored procedures, views and queries. Database Script Generator will be used to update the database.

7.3 Bug Tracking

To track the code of our project, facilitate collaboration and bug handling we have decided to make use of Azure DevOps which can really streamline the abovementioned process. With everything in one place, from code repositories to bug reports, it’s easier to keep an eye on progress and catch issues early. The built-in tools for version control mean we can see exactly what changes were made and when, which helps a lot with debugging. Plus, having a clear view of our workflow, with automated tracking and reporting, keeps everyone on the same page and helps us stay on top of any problems before they turn into bigger issues.

8. Deployment Plan

8.1 Deployment Strategy

We will deploy everything on a single pc where the database will be hosted first where everything will be tested. When everything works as intended the project will be hosted on Microsoft Azure. This will ensure that everything works as intended before we deploy the project to Azure.

8.2 Maintenance

The project will be maintained and edited by the team until the program is in a state deemed sufficient for the task it serves.

There will be a minimal routine maintenance, along with several checks and security measures, to ensure the integrity of the sensors, database and viewing application is unchanged due to any unforeseen reasons like broken sensors, change of “YR.no” API URL or specifications, and database or viewing page breach.

If there is a case where an update is necessary for the project to maintain a decent usability, a database/server migration must occur, there is a need to change the sensors currently put in place or any other unforeseen event, the team will handle the necessary changes swiftly.

9. Documentation

9.1 User Documentation

The viewing application will provide a FAQ, help guide, page to help any doubts our final users could encounter during their use of our application.

For the administrators there will be a short user manual detailing concisely what types of data management are possible with their access privileges to the data administration page.

9.2 Technical Documentation

A system architecture, system creation and an API document will be created to help future developers understand the project in its whole and ensure their complete knowledge on every aspect of the project to certify they will be sufficiently qualified to develop and maintain key features of the project.