SE 4485: Software Engineering Projects

Fall 2025

Requirement Documentation

|  |  |
| --- | --- |
| Group Number | Team 1 |
| Project Title | City Level Air Quality Prediction Application |
| Sponsoring Company | Raytheon (Team A) |
| Sponsor(s) | Ryan Havens <Ryan.Havens@rtx.com>,  Marc Perna <marc.perna@rtx.com>,  Trey Williams <trey.williams@rtx.com>,  Trevor Lang <trevor.a.lang@rtx.com> |
| Students | 1. Jay Chung <cwc130330@utdallas.edu>  2. Amelia Quinn <qcb220000>  3. Kevin Melo <ksm220005>  4. AJ Kimbrough <ank210005>  5. David Santos <des210001>  6. Andrew Enright <ame210008> |

**TABLE OF CONTENTS**

1. INTRODUCTION 3
2. USE CASE MODEL FOR FUNTIONAL REQUIREMENTS 3
3. RATIONALE FOR YOUR USE CASE MODEL 5
4. NON-FUNCTIONAL REQUIREMENTS 5
5. EVIDENCE OF CONFIGURATION MANAGEMENT 6
6. ENGINEERING STANDARDS AND MULTIPLE CONSTRAINTS 7
7. ADDITIONAL REFERENCES 7

**LIST OF FIGURES**

1. Figure 1 – Use Case Model 3

**LIST OF TABLES**

1. Table 1.1 – Evidence of Configuration Management 6
2. Table 1.2 – Evidence of Configuration Management 6

**INTRODUCTION**

This document presents the Requirements Specification for the City Level Air Quality Prediction (CLAP) web application. Its primary purpose is to define the system’s functional and non-functional requirements, ensuring that the application developed is in alignment with objectives set by the team and sponsor. The scope of this document includes the identification of system capabilities, constraints, and use cases that will guide the design, implementation, and validation of the CLAP system.

The CLAP web application is a predictive analytics tool designed to forecast next-day Air Quality Index (AQI) categories for a selected U.S. city. By combining historical AQI data from the Environmental Protection Agency (EPA) with weather forecasts from the National Weather Service (NWS), the system leverages machine learning models to produce accurate and interpretable predictions. This approach not only serves as a proof of concept for city-level forecasting but also provides an educational framework for developing and testing predictive models in environmental domains.

The CLAP system will provide an interactive dashboard to visualize recent AQI trends and predicted categories. These features are intended to support users in understanding air quality risks and making informed decisions about outdoor activities. Futthermore, the project aims to demonstrate the feasibility of lightweight, portable data-driven forecast classification solutions that can run locally on student hardware with minimal setup.

The remainder of this document is organized as follows: functional requirements (FRs), use case model, non-functional requirements (NFRs), textual use case, rationale behind the use case, and configuration management details.

**USE CASE MODEL FOR FUNCTIONAL REQUIREMENTS**

Figure 1 – Use Case Model: To be added after after approval of textual use case.

The following functional requirements (FRs) define the system’s expected capabilities and core functionality:

* Data Ingestion: Pulls EPA daily AQI data and NWS weather Data.
* Data Processing: Stores in MySQL, clean, and create AQI lag features.
* Prediction: Runs logistic regression to forecast tomorrow’s AQI category.
* Output API: Provides JavaScript Object Notation (JSON) with predicted category and probabilities.
* Dashboard: Displays last 30 days’ AQI and tomorrow’s prediction with a *Refresh* button.

Textual Use Case:

1. Case Name: AQI Forecasting Dashboard Web Application
2. Participating Actors: Both technical and non-technical users, including project stakeholders, who interact with the system to access AQI predictions and related insights.
3. Entry Conditions:

* Users have access to the AQI dashboard.
* EPA and NWS data sources are available.

1. Normal Flow of Events:

* User clicks the *Refresh* button on the dashboard.
* System fetches latest EPA daily AQI data and NWS weather forecast.
* Data is cleaned, merged, and features (e.g. AQI lags and weather attributes) are updated in MySQL.
* Logistic Regression model runs and predicts tomorrow’s AQI category.
* Prediction results are stored in the database and exposed as JSON.
* Dashboard updates show the chart of last 30 days’ AQI, as well as tomorrow’s predicted AQI category and probabilities.

1. Success Condition:

* Dashboard displays updated historical AQI trends and tomorrow’s prediction.

1. Exit Condition:

* Latest forecast is stored in the database.
* The dashboard is visually updated without errors.

1. Exceptions:

* The EPA and/or NWS data is unavailable: System logs error and displays “Data not available” message.
* Prediction failure: The dashboard displays the most recent available prediction instead.

1. Special Requirements:

* The dashboard must use standard EPA AQI categories (e.g. Good, moderate, unhealthy, etc.).
* Predictions must be presented in a clear, user-friendly format (e.g. color, coding, labels).

**RATIONALE FOR YOUR USE CASE MODEL**

The use case model demonstrates how users and stakeholders interact with the system to obtain accurate and timely AQI predictions. It highlights the flow of data from external sources to the prediction model and visualization dashboard, ensuring that the system provides information in a simple and intuitive manner.

**NON-FUNCTIONAL REQUIREMENTS**

The non-functional requirements (NFRs) define the quality attributes and operational constraints of the system, describing how the system performs rather than what it does. Each requirement is ranked by priority to indicate its relative importance for successful implementation and stakeholder satisfaction.

1. Performance: The refresh workflow (e.g. data ingestion 🡪 prediction 🡪 update) must be completed within one minute for one city.
2. Usability: Dashboard must be responsive and easy to navigate for non-technical users.
3. Reliability: Daily ingestion and processing must succeed >95% of the time. Graceful error handling when API data is missing.
4. Reproducibility: The refresh workflow must be fully deterministic and version-controlled for experiments.
5. Portability: The application must run locally on a laptop with minimal setup (e.g. SQLite or MySQL backend).

**EVIDENCE THE DOCUMENT HAS BEEN PLACED UNDER CONFIGURATION MANAGEMENT**

The team has selected GitHub as the configuration tool for this project. The tables below provide evidence of configuration management by recording version history, authorship, and reviews of document changes. The *ID* column identifies each entry. The *date of change* column indicates when a modification was made to an existing file, and the v*ersion (before & after)* columns include the associated Git commit hash for distinction. The *author* column refers to the author of the new version. The *difference link* column provides a URL to the GitHub comparison view between two consecutive commits. The format of the difference link is as follows:

“https://github.com/cchung7/rtx\_team1/compare/<ver-before-hash>..<ver-after-hash>”.

Table 1.1 – Each entry (or row) tracks a single file revision.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **ID:** | **Date of Change:** | **Version Before:** | **Version After:** | **Author:** | **Review -Change Summary:** | **Reviewers:** |
| 1 | 9/10/2025 | v0.1 () | v0.2 () | AJ Kimbrough (ank210005) | Added refresh workflow pipeline. Updated NFRs and use case. | Jay Chung (cwc130330) |
| 2 | TBD | TBD | TBD | TBD | TBD | TBD |
| 3 | TBD | TBD | TBD | TBD | TBD | TBD |
| 4 | TBD | TBD | TBD | TBD | TBD | TBD |

Table 1.2 – Each entry (or row) lists a difference link.

|  |  |
| --- | --- |
| **ID:** | **Difference Link:** |
| 1 | https://github.com/cchung7/rtx\_team1/compare/.. |
| 2 | TBD |
| 3 | TBD |
| 4 | TBD |

**ENGINEERING STANDARDS AND MULTIPLE CONSTRAINTS**

* IEEE Std 830-1998: Software Requirements [[pdf](https://course.techconf.org/se4485/IEEE/IEEE%20Std%20830-1998-Software-Requirements.pdf)]
* IEEE Std 29148: Requirements Engineering [[pdf](https://course.techconf.org/se4485/IEEE/IEEE%2029148%20(2011)%20-%20Requirements%20Engineering.pdf)]
* ISO/IEC/IEEE Std 29148-2018: Systems and Software Engineering
  + Life Cycle Processes
  + Requirements Engineering [[pdf](https://course.techconf.org/se4485/IEEE/ISO-IEC-IEEE-29148-2018.pdf)]

**ADDITIONAL REFERENCES**

* + Lamsweerde, A.V., 2009. *Requirements Engineering: From System Goals to UML Models to Software Specifications.* John Wiley
  + OpenAI. (2025). *ChatGPT-5. (Aug. 7 version) [Large language model].* Retrieved from: <https://chat.openai.com/chat>