SE 4485: Software Engineering Projects

Spring 2025

Final Report

|  |  |
| --- | --- |
| Group Number | 3 |
| Project Title | Predictive Analytics Application |
| Sponsoring Company | Raytheon |
| Sponsor(s) | Marc Perna, Trey Williams, Trevor A. Lang, Ryan Havens |
| Students | 1. Brie Haynes  2. Srinesh Ganesh  3. Gregory Rodriguez  4. Victor Sim  5. Josh Pahman  6. Yousuf AlMakhamreh |

Emission Analyzer

Group 3

Executive Summary

Emission Analyzer is a web application that predicts the carbon monoxide and nitrous oxide emissions of commercial jet engines based on three operational parameters: rated thrust, bypass ratio, and pressure ratio. This report provides insight into the application’s development, ranging from its project management plan to its test plan.

Table of Contents

[1. Introduction 2](#_Toc419426071)

[Purpose and Scope 2](#_Toc754020786)

[Product Overview 2](#_Toc935683739)

[Structure of the Document 2](#_Toc1969305521)

[Terms, Acronyms, and Abbreviations 2](#_Toc2572350)

[2. Project Management Plan 3](#_Toc2019168694)

[Project Organization 3](#_Toc1868664413)

[Lifecycle Model Used 3](#_Toc1501411129)

[Risk Analysis 3](#_Toc1134508334)

[Software and Hardware Resource Requirements 4](#_Toc1726295747)

[Deliverables and Schedule 5](#_Toc813272803)

[Monitoring, Reporting, and Controlling Mechanisms 6](#_Toc62097326)

[Professional Standards 7](#_Toc2117506918)

[Evidence the Document has Been Placed Under Configuration Management 7](#_Toc264167174)

[Impact of the Project on Individuals and Organizations 8](#_Toc1797807907)

[3. Requirement Specifications 8](#_Toc1930632920)

[FUNCTIONAL REQUIREMENTS 15](#_Toc1387766673)

[Create Account 15](#_Toc222612403)

[Add Engine 15](#_Toc1411861870)

[Edit Engine 15](#_Toc754063491)

[Delete Engine 16](#_Toc1917351605)

[Perform Emission Prediction 16](#_Toc237690416)

[Generate Visualizations 16](#_Toc676968650)

[Download Prediction Data 17](#_Toc1717494711)

[NON-FUNCTIONAL REQUIREMENTS 17](#_Toc1596956779)

[Availability 17](#_Toc1026272257)

[Error Handling 17](#_Toc1419799944)

[Usability 17](#_Toc845020114)

[Simple Input Fields 18](#_Toc1163789962)

[Ease of Navigation 18](#_Toc212762776)

[Maintainability 18](#_Toc985596810)

[Documentation 18](#_Toc1034965627)

[Interoperability 18](#_Toc1836828162)

[Browser Support 18](#_Toc1277268617)

[4. Architecture 18](#_Toc434501775)

[Architectural Style 18](#_Toc519052053)

[Feature Support 18](#_Toc133379304)

[Create Account 18](#_Toc2094195730)

[Add Engines to Account 19](#_Toc2020961094)

[Edit Engines 19](#_Toc1255795643)

[Delete Engines 20](#_Toc1126117967)

[Predict Emissions 20](#_Toc1839733549)

[Visualize Results 20](#_Toc502577124)

[Download Prediction Data 21](#_Toc2093768763)

[Technology 23](#_Toc573105542)

[Programming Languages 23](#_Toc501304472)

[Frameworks and Libraries 23](#_Toc893899756)

[Database Management System 23](#_Toc498729652)

[Software 23](#_Toc92305074)

[Containerization 23](#_Toc587480919)

[Developer Tools 23](#_Toc223578776)

[Prediction Model 23](#_Toc1400690561)

[Hardware 23](#_Toc1791600144)

[Minimum Requirements 23](#_Toc2083837862)

[Recommended Requirements 23](#_Toc734641543)

[End User Hardware 23](#_Toc1774154583)

[Communication Between Application and Servers 23](#_Toc1642598955)

[5. Design 23](#_Toc1509617854)

[Entity-Relationship 23](#_Toc1861854126)

[Add Engine 23](#_Toc1685056445)

[GUI Rationale 23](#_Toc1660576753)

[Landing Page 23](#_Toc265686911)

[Home Page 23](#_Toc931635559)

[Add Engine Page 23](#_Toc988311407)

[Engine Page 23](#_Toc1465093025)

[Edit Engine Page 23](#_Toc608024394)

[Class Diagrams Rationale 23](#_Toc1702220316)

[Sequence Diagrams Rationale 23](#_Toc1439310166)

[6. Test Plan 23](#_Toc1150568935)

[Create User 23](#_Toc1238627129)

[Add Engine 23](#_Toc956395906)

[Edit Engine 23](#_Toc1852700434)

[Delete Engine 23](#_Toc817688518)

[Predict Emissions 23](#_Toc1177131732)

[Generate Visualizations 23](#_Toc1297823165)

[Download Prediction Data 23](#_Toc780847337)

[Techniques 23](#_Toc327084558)

[Criteria 23](#_Toc870559169)

[7. Evidence the Document Has Been Placed under Configuration Management 23](#_Toc889054443)

[8. Engineering Standards and Multiple Constraints 23](#_Toc2062390547)

[9. Additional References 23](#_Toc460881777)

[Acknowledgment 23](#_Toc678444170)

List of Figures

[Use Case 1: Create Account 13](#_Toc197716784)

[Use Case 2: Add Engine 14](#_Toc197716785)

[Use Case 3: Edit Engine 14](#_Toc197716786)

[Use Case 4: Delete Engine 14](#_Toc197716787)

[Use Case 5: Predict Emissions 15](#_Toc197716788)

[Use Case 6: Generate Visualizations 16](#_Toc197716789)

[Use Case 7: Download Prediction Data 17](#_Toc197716790)

[Figure 1: Architectural Model 26](#_Toc197718333)

[Figure 2: Requirements Traceability Matrix 29](#_Toc197718334)

[Figure 3: Landing Page 30](#_Toc197718335)

[Figure 4: Home Page 31](#_Toc197718336)

[Figure 5: Add Engine Page 31](#_Toc197718337)

[Figure 6: Engine Page 33](#_Toc197718338)

[Figure 7: Edit Engine Page 33](#_Toc197718339)

[Figure 8: Entity-Relationship Class Diagram 34](#_Toc197718340)

[Figure 9: Software Components Class Diagram 35](#_Toc197718341)

[Figure 10: Component Traceability 43](#_Toc197718342)

[Sequence Diagram 1: Add Engine 35](#_Toc197718389)

[Sequence Diagram 2: Edit Engine 36](#_Toc197718390)

[Sequence Diagram 3: Delete Engine 37](#_Toc197718391)

[Sequence Diagram 4: Predict Emissions 38](#_Toc197718392)

[Sequence Diagram 5: Generate Visualizations 39](#_Toc197718393)

[Sequence Diagram 6: Download Prediction Data 40](#_Toc197718394)

[Request 1: Create User 43](#_Toc197718376)

[Request 2: Add Engine 44](#_Toc197718377)

[Request 3: Edit Engine 45](#_Toc197718378)

[Request 4: Delete Engine 46](#_Toc197718379)

[Request 5: Predict Emissions 47](#_Toc197718380)

[Request 6: Generate Visualizations 48](#_Toc197718381)

[Request 7: Download Prediction Data 49](#_Toc197718382)

List of Tables

[Textual Description 1: Create Account 17](#_Toc197722962)

[Textual Description 2: Add Engine 18](#_Toc197722963)

[Textual Description 3: Edit Engine 19](#_Toc197722964)

[Textual Description 4: Delete Engine 19](#_Toc197722965)

[Textual Description 5: Predict Emissions 20](#_Toc197722966)

[Textual Description 6: Generate Visualizations 21](#_Toc197722967)

[Textual Description 7: Download Prediction Data 22](#_Toc197722968)

[Feature Support 1: Add Account 22](#_Toc197718981)

[Feature Support 2: Add Engines 23](#_Toc197718982)

[Feature Support 3: Edit Engines 23](#_Toc197718983)

[Feature Support 4: Delete Engines 23](#_Toc197718984)

[Feature Support 5: Predict Emissions 24](#_Toc197718985)

[Feature Support 6: Visualize Results 24](#_Toc197718986)

[Feature Support 7: Download Prediction Data 25](#_Toc197718987)

[Test Case 1: Create User 44](#_Toc197719147)

[Test Case 2: Add Engine 45](#_Toc197719148)

[Test Case 3: Edit Engine 46](#_Toc197719149)

[Test Case 4: Delete Engine 47](#_Toc197719150)

[Test Case 5: Predict Emissions 48](#_Toc197719151)

[Test Case 6: Generate Visualizations 50](#_Toc197719152)

[Test Case 7: Download Prediction Data 51](#_Toc197719153)

# Introduction

## Purpose and Scope

The purpose of this document is to consolidate information from the various reports our team has produced thus far into a single, cohesive final report containing all necessary information relevant to the project. This includes information from our Project Management Plan, Requirements Documentation, Architecture Documentation, Detailed Design Documentation, and Test Plan. As such, the scope of this document is the sum of the scopes of those documents; that is, every aspect of the project.

## Product Overview

The emission analyzer will evaluate the impact of operational parameters on NOx and CO emissions from commercial jets. Using predictive models and interactive visualizations, it will help airlines, manufacturers, and regulators balance emissions while maintaining efficiency. The model will be able to explore operational scenarios and generate downloadable reports. By leveraging machine learning and data analysis, the tool provides actionable insights for optimizing engine performance and reducing environmental impact. This project supports sustainable aviation by enabling data driven decision making to minimize emissions without compromising operational effectiveness.

## Structure of the Document

Please see the table of contents above for a general overview of the document’s structure.

## Terms, Acronyms, and Abbreviations

**Terms:**

**B/P Ratio:** Bypass Ratio; the amount of air that bypasses the compressor portion of a jet engine.

**Rated Thrust:** The maximum thrust potential of a jet engine, given in units of Newtons.

**Pressure Ratio:** The amount of compression a jet engine applies to the air that does not bypass the compressor.

**Acronyms:**

**NOx:** Nitrous Oxide

**CO:** Carbon Monoxide

**API:** Application Programming Interface

**UI:** User Interface

**UX:** User Experience

**GPU:** Graphics Processing Unit

**CPU:** Central Processing Unit

**RAM:** Random Access Memory

**FR:** Functional Requirement

**NFR:** Non-Functional Requirement

# Project Management Plan

## Project Organization

Our team will be divided into two groups, with three members assigned to each one:

**Data and Analytics Team**

* Clean, preprocess, and organize the data for analysis
* Set up and manage the data pipeline (ingest, storage, and access)
* Develop predictive models using the data
* Collaborate with the Application Team to integrate the models into the app

**Application Team**

* Design and build the user interface
* Develop back-end logic and APIs to connect the UI with the data and models
* Implement data visualizations dashboards

*Table 1*

|  |  |
| --- | --- |
| **Teams** | **Members** |
| Data and Analytics | Josh Pahman |
|  | Victor Sim |
|  | Gregory Rodriguez |
| Application | Brie Haynes |
|  | Srinesh Ganesh |
|  | Yousuf Al Makhamreh |

## Lifecycle Model Used

Our team will adopt the Agile methodology, specifically the Scrum framework, to guide our development process. This approach will foster a highly communicative and collaborative environment, ensuring that all team members remain aligned on project goals and progress. Additionally, this framework will enable us to deliver incremental value while maintaining steady progress, mitigating risks, and improving the overall quality of our work.

## Risk Analysis

**Data-Related Risks**

* Data Quality and Bias: The emissions dataset may contain biases, missing values, or incorrect information, impacting model performance.
  + Mitigation: Implement robust data preprocessing, augmentation, and bias detection techniques to improve model fairness and accuracy.
* Data Availability and Consistency: Inconsistent or insufficient data may hinder model training and validation.
  + Mitigation: Establish standardized data collection protocols and use synthetic data generation techniques when necessary.

**Prediction Model Risks**

* Model Accuracy and Generalization: Machine learning models may not generalize well to unseen operational conditions, leading to inaccurate predictions.
  + Mitigation: Use high-quality, diverse training datasets, perform rigorous validation, and continuously refine models with real-world data
* Overfitting and Underfitting: The model may be overfit to training data or fail to capture complex relationships in emissions data
  + Mitigation: Use regularization techniques, cross-validation, and hyperparameter tuning to ensure a balanced model.
* Computational Efficiency: Training and inference of complex models may require significant computational power, leading to performance issues.
  + Mitigation: Optimize model architectures, use efficient algorithms, and employ local computing resources effectively.

**Deployment Risks**

* System Integration Challenges: The application may face compatibility issues with existing aviation systems and workflows.
  + Mitigation: Design modular APIs, follow industry standards, and conduct thorough integration testing.
* Scalability Limitations: The system may struggle to handle large datasets or multiple users efficiently.
  + Mitigation: Implement efficient data structures, caching mechanisms, and conduct load testing to ensure scalability.

## Software and Hardware Resource Requirements

**Hardware Requirements**

* High-performance CPU (Multi-core i7/i9, Ryzen 7/9, or Apple M-series chip) – Required for data preprocessing and computational tasks.
* At least 16GB RAM (32GB recommended) – Ensures smooth handling of large datasets and real-time computations.
* GPU (NVIDIA RTX 3060 or higher, or Apple M-series GPU) – If using deep learning models or large-scale simulations for aircraft emissions.
* 500GB+ SSD Storage (1TB preferred) – For storing datasets, models, and logs efficiently.

**Software Requirements**

* **Programming Languages**
  + Frontend: React.js + D3.js for interactive UI + trade-off curves and heatmaps for NOx vs CO emissions visualization.
  + Backend: Python + Django for backend API development to process and serve emission predictions.
* **Data Processing, Analysis & Visualization**
  + PostgreSQL/MySQL for storing engine operational data, emissions levels, and user-uploaded datasets.
  + Jupyter Notebook for data preprocessing, exploratory data analysis (EDA), and model testing.
  + Matplotlib/Plotly to generate static and dynamic emission reports
* **Machine Learning & Simulation Tools**
  + Machine learning: Scikit-Learn + XGBoost for predictive modeling.
  + Emissions modeling: Cantera/NASA CEA for physics-based simulations.
* **Version Control**: GitHub/GitLab for code collaboration and tracking changes.
* **Project Management Tools**: GitHub Projects.

## Deliverables and Schedule

**Deliverables**

* **Data Collection and Preprocessing**
  + Gather historical flight operation-related data, engine parameters, and combustion/emissions data.
  + Format data for AI model training and visualization.
* **AI Model Development**
  + Train an AI model to predict NOx and CO emissions.
* **Visualization and UI Development**
  + Build a graph visualization of trade-off curves for NOx vs. CO emissions and other related data using tools. like D3.js, Recharts, etc....
  + Integrate a way for users to insert their own data into the model to show their own graphs.
* **Model Evaluation and Testing**
  + Validate predictions using historical emissions-related data
  + Refine model accuracy and optimize performance.
* **Final Report and Deployment**
  + Document findings and prepare a presentation.
  + Deploy a functional prototype for user feedback.

**Schedule** **(11 Weeks)**

We intend to start development in the week of February 24th.

*Table 2*

|  |  |  |
| --- | --- | --- |
| **Week** | **Data and Analytics Team** | **Application Team** |
| 1-2 | Review, collect, and preprocess data from sources. | Design wireframes and define UI requirements. |
| 3-4 | Perform exploratory data analysis and identify key input variables. | Develop back-end API for data upload and model integration. Use mock data if needed. |
| 5-6 | Develop and train initial model to predict NOx and CO emissions. | Develop front-end interface for data upload and results display. |
| 7-8 | Refine models and analyze NOx-CO trade-offs. | Design and implement visualizations (e.g. trade-off curves, heat maps). Integrate model. |
| 9-10 | Test and optimize models and analytics. | Test and deploy the web application. |

**Dependencies**

*Team Dependencies*

Many of the project’s dependencies exist between the Data Analytics Team and the Application Team. To develop a fully functional user interface, the Application Team requires the API built by the Data Analytics Team. However, this dependency can be mitigated by using mock data until the official API and data are ready. Beyond the API and data, another dependency that exists between the Data Analytics Team and the Application Team is the AI Model. To develop heatmap visualizations and trade-off curves, the Application Team will need the AI-generated emissions predictions.

*Estimation Logic*

The estimation logic has a dependency on the AI model, as it analyzes operational parameters to estimate NOx and CO emissions. To have this functionality, the AI model needs to be fully implemented, with the ability to predict emission levels based on engine-specific data and format the results into an output that can be used for visualization and optimization.

*Emissions Prediction Model*

The emissions prediction model has a dependency on high-quality historical and real-time engine operational data, including thrust settings, fuel flow rates, and combustion temperatures. The model must be fully trained and optimized to accurately predict NOx and CO emissions, ensuring that downstream applications, such as visualizations and operational recommendations, can function effectively.

## Monitoring, Reporting, and Controlling Mechanisms

To ensure our project is successful, the following reports will be produced regularly, providing insight into our team’s progress:

* Progress Reports: A detailed progress report will be generated at the end of each sprint to track completed tasks, ongoing activities, and any challenges encountered.
* Quality Assurance Reports: These will be generated after each phase of testing. They will document test results, any identified faults, failures, or errors, and the steps taken to resolve them.

The control mechanisms that we will use include:

* Kanban Boards: A Kanban Board will be used to track our tasks. It will provide different statuses for each one such as “To Do”, “In Progress”, and “Done”. It will also provide a level of importance, such as “Low”, “Medium”, and “High”, for prioritization.
* Weekly Updates: Team members will do a weekly status update in team meetings.

## Professional Standards

* IEEE Std 1058-1998: Software Project Management Plans [[pdf](https://course.techconf.org/se4485/IEEE/IEEE-Std-1058-1998-Software-Project-Management-Plans.pdf)]
* PMBOK® Guide: Project Management Body of Knowledge [[pdf](https://course.techconf.org/se4485/IEEE/PMBOKR.pdf)]
* IEEE Std 12207: Software Life Cycle Processes [[pdf](https://course.techconf.org/se4485/IEEE/IEEE%2012207%20(2017)%20-%20Software%20Life%20Cycle%20Processes.pdf)]
* IEEE Std 15939: Measurement Process [[pdf](https://course.techconf.org/se4485/IEEE/IEEE%2015939%20(2017)%20-%20Measurement%20Process.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)]

There are no additional engineering standards or constraints that our sponsors would like us to apply to this document.

## Evidence the Document has Been Placed Under Configuration Management

We will be using Microsoft Word for our Configuration Management.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Version | Checked Out By | Checked In By | Changes Made | Reviewers | Ship-It Status |
| v1.0 | Brie | Brie | Added abstract, introduction, lifecycle model, and project organization. | Josh, Greg | Ship-It ✅ |
| v1.1 | Srinesh | Srinesh | Added software and hardware requirements | Brie, Greg | Ship-It ✅ |
| v1.2 | Josh | Josh | Added prediction model risks and deployment risks | Brie, Victor | Ship-It ✅ |
| v1.3 | Brie | Brie | Added deliverables, schedule, and dependencies | Josh, Victor | Ship-It ✅ |
| v2.0 | Brie | Brie | Edited schedule | Josh, Victor | Ship-It ✅ |
| v2.1 | Greg | Greg | Updated the project description | Josh, Victor | Ship-It ✅ |
| V2.2 | Josh | Josh | Updated risk analysis | Victor, Yousuf | Ship-It ✅ |
| V2.3 | Victor | Victor | Updated the Deliverables section, got rid of something in hardware requirements. | Josh, Srinesh | Ship-It ✅ |
| V2.4 | Brie | Brie | Added references | Josh, Srinesh | Ship-It ✅ |
| V2.5 | Yousuf | Yousuf | Updated the dependencies | Srinesh, Josh | Ship-It ✅ |
| V2.6 | Srinesh | Srinesh | Updated the software + hardware requirements | Josh, Brie | Ship-It ✅ |

## Impact of the Project on Individuals and Organizations

This project is focused on predicting NOx and CO emissions from commercial jet engines using machine learning. The goal is to help airlines and manufacturers make better decisions about engine performance and emissions. The tool will give users visual reports and predictions based on real-world engine data.

For individuals, especially those concerned with environmental issues, this project could help reduce the pollution caused by flying. That matters because NOx and CO emissions are harmful to air quality and can affect public health. By helping airlines lower emissions, the project can contribute to cleaner air and a healthier environment.

On a bigger scale, it supports public welfare by encouraging more sustainable aviation. It can also help companies meet environmental regulations and avoid fines. Economic factors are considered too, since improving efficiency while lowering emissions could reduce fuel costs.

The project doesn't directly deal with cultural or social factors, but the tech could be applied globally, since air travel is used worldwide. Overall, this project aims to balance performance and environmental impact, which is useful for both people and organizations.

# Requirement Specifications

* 1. Stakeholders for the system

Company

Raytheon (Sponsoring Company)

Marc Perna, Trey Williams, Trevor A. Lang, Ryan Havens (Sponsors)

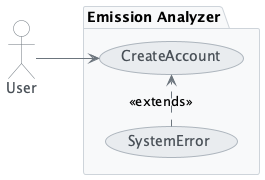
Users

Engine manufacturers and designers

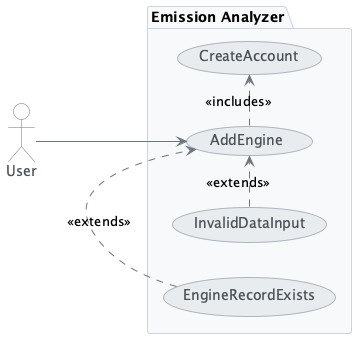
Environmental analysts

* 1. Graphic Use case model for functional requirements

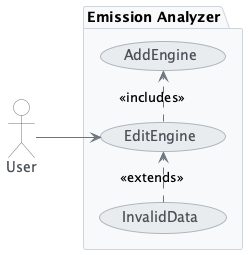
Use Case 1: Create Account



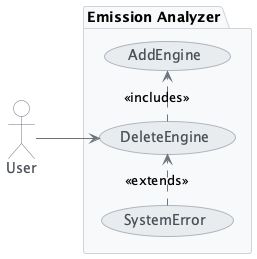
Use Case 2: Add Engine



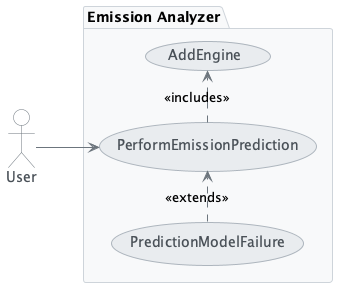
Use Case 3: Edit Engine



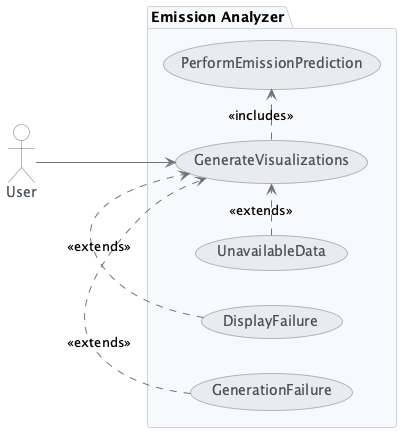
Use Case 4: Delete Engine



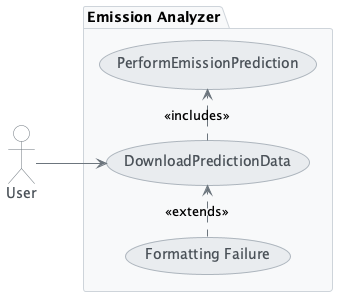
Use Case 5: Predict Emissions



Use Case 6: Generate Visualizations



Use Case 7: Download Prediction Data



* 1. Rationale for each use case model

**FR1.** The system shall allow users to create an account. This allows for engines to be associated with a particular user.

**FR2.** The system shall allow users to add an engine to their account. This establishes a structured framework for tracking and managing engine-specific data, enabling users to organize all relevant parameters and predictions associated with a particular engine.

**FR3.** The system shall allow users to edit engines associated with their account, enabling them to update engine data or correct any errors.

**FR4.** The system shall allow users to delete engines associated with their account, enabling them to remove any engines they no longer wish to keep.

**FR5.** The system shall predict NOx and CO emission levels based on the engine parameter data associated with an engine. This allows users to analyze the environmental impact of different operational settings, enabling informed decisions to optimize engine performance while minimizing emissions.

**FR6.** The system shall generate and display visualizations showing the predicted NOx and CO emission levels. This enables users to interpret the results of the emission predictions in a visual manner, facilitating better understanding of the relationship between operational parameters and emissions.

**FR7.** The system shall allow users to download prediction data for all engines, providing access to the raw data underlying the visualizations.

* 1. Textual Description for each use case

Textual Description 1: Create Account

|  |  |
| --- | --- |
| **Create Account** | |
| **Participating Actors** | User |
| **Entry Condition(s)** | The user must select the option to create an account |
| **Normal Flow of Events** | The user selects the option to create an account  The system generates the necessary data  The system stores the data  The system navigates the user into the application |
| **Exit Condition(s)** | **Successful Exit**  The system has successfully stored the data  The user is able to access the application  **Unsuccessful Exit**  The system is unable to generate the necessary data to create an account  The system notifies the user and suggests to try account creation again |
| **Exceptions** | **System Error:** If the system is unable to create the user account (e.g. the API is down), the system will notify the user and prompt them to try account creation again. |
| **Special Requirements** | No special requirements. |

Textual Description 2: Add Engine

|  |  |
| --- | --- |
| **Add Engine** | |
| **Participating Actors** | User |
| **Entry Condition(s)** | The user must have an account  The user must have valid input data |
| **Normal Flow of Events** | The user selects the option to add an engine  The system prompts the user to enter the engine data  The user enters the data  The system validates the data  The system stores the data  The system notifies the user that the data has been successfully entered |
| **Exit Condition(s)** | **Successful Exit**  The system has successfully stored the data  The user receives confirmation from the system that the data has been successfully entered  **Unsuccessful Exit**  The data provided by the user is invalid or incomplete  The system notifies the user of the error and prompts them to correct their input |
| **Exceptions** | **Invalid data:** If the data is invalid, the system will prompt the user to re-enter their input. This exception also includes the case where there is missing *required* data such as the engine identification and engine type.  **Already exists:** If the engine record already exists, the system will notify the user and prompt them to navigate to its web page. |
| **Special Requirements** | Each engine record ID must be unique within the system (e.g. two engine records cannot have “HTF7000” as their engine identification) |

Textual Description 3: Edit Engine

|  |  |
| --- | --- |
| **EditEngine** | |
| **Participating Actors** | User |
| **Entry Condition(s)** | The user must have at least one engine associated with their account |
| **Normal Flow of Events** | The user selects the option to edit the engine  The system prompts the user to enter the new engine data  The user enters the data  The system validates the data  The system stores the data  The system notifies the user that the engine’s data has been successfully updated |
| **Exit Condition(s)** | **Successful Exit**  The system has successfully stored the data  The user receives confirmation from the system that the data has been successfully entered  **Unsuccessful Exit**  The data provided by the user is invalid or incomplete  The system notifies the user of the error and prompts them to correct their input |
| **Exceptions** | **Invalid data:** If the data is invalid, the system will prompt the user to re-enter their input. This exception also includes the case where there is missing *required* data such as the engine identification and engine type. |

Textual Description 4: Delete Engine

|  |  |
| --- | --- |
| **DeleteEngine** | |
| **Participating Actors** | User |
| **Entry Condition(s)** | The user must have at least one engine associated with their account |
| **Normal Flow of Events** | The user selects the option to delete an engine  The system prompts the user to verify their selection  The user verifies their selection  The system deletes the engine  The system notifies the user that the engine has been successfully deleted |
| **Exit Condition(s)** | **Successful Exit**  The system has successfully deleted the engine  The user receives confirmation from the system that the engine has been successfully deleted  **Unsuccessful Exit**  The system is unable to delete the engine  The system notifies the user of the error and prompts them to attempt deletion again |
| **Exceptions** | **System Error:** The system is unable to fulfill the request, either due to the application, API, or database. |

Textual Description 5: Predict Emissions

|  |  |
| --- | --- |
| **PerformEmissionPrediction** | |
| **Participating Actors** | User |
| **Entry Condition(s)** | The user must have at least one engine associated with their account |
| **Normal Flow of Events** | The user selects the option to perform emission prediction for a specific engine  The system performs the emission prediction  The system generates the prediction results  The system stores the prediction results in the database and sends them to the front-end  The system presents the prediction results to the user |
| **Exit Condition(s)** | **Successful Exit**  The system successfully performed the emission prediction  The system successfully generates the prediction results  The system successfully displays the prediction results  **Unsuccessful Exit**  The system is unable to:  Perform emission prediction  Generate the prediction results  The system notifies the user and suggests attempting emission prediction again. |
| **Exceptions** | **Prediction Model Failure:** If the prediction model fails, the system will notify the user and suggest performing the emission prediction again. |
| **Special Requirements** | There are no special requirements for this use case. |

Textual Description 6: Generate Visualizations

|  |  |
| --- | --- |
| **GenerateVisualizations** | |
| **Participating Actors** | User |
| **Entry Condition(s)** | The user must have at least one engine associated with their account  The user must have performed emission prediction on a specific engine |
| **Normal Flow of Events** | The user selects the option to predict the emissions of a particular engine  The system generates the prediction results  The system generates the visualizations based on the prediction results  The system displays the visualizations to the user |
| **Exit Condition(s)** | **Successful Exit**  The system successfully generated and displayed the visualizations  **Unsuccessful Exit**  The system is unable to generate and display the visualization  The system prompts the user to attempt predicting the emissions again |
| **Exceptions** | **Generation Failure:** The system is unable to generate the visualizations  **Display Failure:** The system is unable to display the visualizations  **Data Unavailable:** The data for visualization is missing, corrupt, or improperly formatted |
| **Special Requirements** | There are no special requirements for this use case. |

Textual Description 7: Download Prediction Data

|  |  |
| --- | --- |
| **DownloadPredictionData** | |
| **Participating Actors** | User |
| **Entry Condition(s)** | The user must have at least one engine associated with their account  The user must have performed emission prediction on a specific engine |
| **Normal Flow of Events** | The user selects the option to download their prediction data  The system retrieves the prediction results  The system formats the prediction result to JSON  The system downloads the prediction results onto the user’s computer |
| **Exit Condition(s)** | **Successful Exit**  The system successfully retrieved and downloaded the prediction results in the desired format  **Unsuccessful Exit**  The system is unable to retrieve the prediction results  The system prompts the user to attempt downloading the prediction data again |
| **Exceptions** | **Formatting Failure:** If the system is unable to format the prediction model response into the desired format, the system will notify the user and suggest performing the emission prediction again. |
| **Special Requirements** | There are no special requirements for this use case. |

3.5. Non-functional requirements

## Availability

### *Error Handling*

**NFR1.** The application must display an error message within 2 seconds of error detection that includes the error code and a description of the error.

**NFR2.** The system must not crash or lose data during errors.

## Usability

### *Simple Input Fields*

**NFR3.** The user interface must include labeled input fields for the supported operational parameters (e.g. rated thrust, B/P ratio, pressure ratio).

**NFR4.** Each input field should validate for numeric input only and display a warning message if the input format is incorrect.

**NFR5.** The system must reject inputs that exceed **valid ranges.**

Turbofan

Rated Thrust: 22 kN to 489 kN

Bypass Ratio: 1 to 15:1

Pressure Ratio: 9.5:1 to 50:1

Mixed Turbofan

Rated Thrust: 22 kN to 356 kN

Bypass Ratio: 0.5 to 12:1

Pressure Ratio: 20:1 to 50:1

### *Ease of Navigation*

**NFR6.** The application must allow users to input data, view results, and navigate through the application with no more than 3 clicks from one task to another.

## Maintainability

### *Documentation*

**NFR7.** Documentation that explains how to use the system’s main functions must be provided.

## Interoperability

### *Browser Support*

**NFR8.** The application must work correctly on the latest two versions of Firefox and Chrome.

# Architecture

* 1. Architectural style(s) used

## Architectural Style

The architectural style used for this application is the layered architecture pattern. There are five layers: Presentation, Application, Service, Data Access, and Database. This style was chosen due to its focus on separation of concerns, modularity, and maintainability.

## Feature Support

### *Create Account*

Feature Support 1: Add Account

|  |  |
| --- | --- |
| **Layers** | **Actions** |
| Presentation | * Provides option for creating user account * Displays success/error message based on API response |
| Application | * Implements endpoint for creating a user account * Forwards engine data to the Service Layer |
| Service | * Validates engine data against business rules (i.e. no duplicate IDs) * Forwards validated engine data to the Data Access Layer |
| Data Access | * Maps engine data to the “User” table * Builds database query logic * Executes query logic |
| Database | * Stores user data in the “User” table |

### *Add Engines to Account*

Feature Support 2: Add Engines

|  |  |
| --- | --- |
| **Layers** | **Actions** |
| Presentation | * Provides form for engine details * Validates input and sends data to API endpoint * Displays success/error message based on API response |
| Application | * Implements endpoint for adding engines to account * Forwards engine data to the Service Layer |
| Service | * Validates engine data against business rules (i.e. no duplicate IDs) * Forwards validated engine data to the Data Access Layer |
| Data Access | * Maps engine data to the “Engine” table * Builds database query logic * Executes query logic |
| Database | * Stores engine data in the “Engine” table |

### *Edit Engines*

Feature Support 3: Edit Engines

|  |  |
| --- | --- |
| **Layers** | **Actions** |
| Presentation | * Provides form for editing engine details * Validates input and sends data to API endpoint * Displays success/error message based on API response |
| Application | * Implements endpoint for editing engines associated with account * Forwards engine data to the Service Layer |
| Service | * Validates engine data against business rules * Forwards validated engine data to the Data Access Layer |
| Data Access | * Maps engine data to the “Engine” table * Builds database query logic * Executes query logic |
| Database | * Stores engine data in the “Engine” table |

### *Delete Engines*

Feature Support 4: Delete Engines

|  |  |
| --- | --- |
| **Layers** | **Actions** |
| Presentation | * Provides “Delete Engine” button * Displays success/error message based on API response |
| Application | * Implements endpoint for deleting engines associated with account * Forwards engine data to the Service Layer |
| Service | * Forwards validated engine data to the Data Access Layer |
| Data Access | * Builds database query logic * Executes query logic |
| Database | * Deletes selected engine |

### *Predict Emissions*

Feature Support 5: Predict Emissions

|  |  |
| --- | --- |
| **Layers** | **Actions** |
| Presentation | * Sends data to designated API endpoint |
| Application | * Implements endpoint for performing emission prediction * Forwards engine parameter data to the Service Layer * Packages the prediction results into a JSON response and sends it to the front-end |
| Service | * Performs the emission prediction and returns the results to the Application Layer * Forwards the prediction results to the Data Access Layer |
| Data Access | * Maps prediction data to “Prediction” table * Builds database query logic * Inserts data |
| Database | * Stores prediction results in the “Prediction” table |

### *Visualize Results*

Feature Support 6: Visualize Results

|  |  |
| --- | --- |
| **Layers** | **Support** |
| Presentation | * Requests prediction data from the designated API endpoint * Renders the data visualization upon success |
| Application | * Provides API endpoint for retrieving existing prediction data * Forwards request to the Service Layer |
| Service | * Implements business logic for validating the data * Forwards validated data to the Data Access Layer |
| Data Access | * Builds database query logic |
| Database | * Retrieves necessary data from the “Prediction” table in the database |

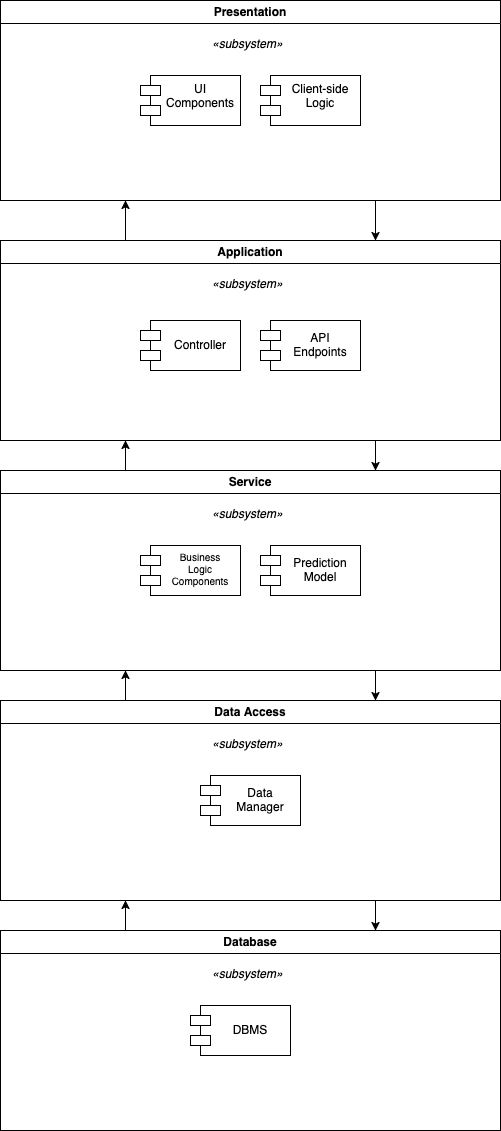
### *Download Prediction Data*

Feature Support 7: Download Prediction Data

|  |  |
| --- | --- |
| **Layers** | **Support** |
| Presentation | * Provides “Download Prediction Data” button * Checks the application store for prediction data. Sends data to the designated API endpoint if needed |
| Application | * Provides API endpoint for retrieving existing prediction data * Forwards request to the Service Layer |
| Service | * Implements business logic for validating the data * Forwards validated data to the Data Access Layer |
| Data Access | * Builds database query logic * Retrieves necessary data from the “Prediction” table in the database |
| Database | * Retrieves prediction data from the Prediction table |

* 1. Architectural model

Figure 1: Architectural Model



* 1. Technology, software, and hardware used

## Technology

### *Programming Languages*

* JavaScript (Front-end)
* Python (Back-end)
* C/C++ (Prediction model)

### *Frameworks and Libraries*

* React + Vite (Front-end)
* Django (Back-end)
* Plotly.js (Data visualizations)

### *Database Management System*

* PostgreSQL

## Software

### *Containerization*

* 1. Docker
  2. Used for containerizing the various components of the application including the front-end, back-end, database, and prediction model.
* Docker Compose
* Used to manage the Docker containers

### *Developer Tools*

* Command Line Interface (CLI)
* Integrated Development Environment (IDE)

### *Prediction Model*

* 1. MATLAB
  2. Used to create the prediction model

## Hardware

### *Minimum Requirements*

* CPU: 2 cores
* RAM: 8 GB
* Storage: 100 GB

### *Recommended Requirements*

* CPU: 4 cores
* RAM: 16 GB
* Storage: 256 GB

### *End User Hardware*

The user’s hardware must support the ability to render HTML, CSS, and JavaScript. Most modern computers with a sufficient browser will be able to run this application.

### *Communication Between Application and Servers*

The application and server will communicate through the application programming interface (API), which acts as an intermediary component between them. The API will forward all requests from the client to the server, which will then carry out the requested response. The server will then send the response to the API, which will be routed back to the client.

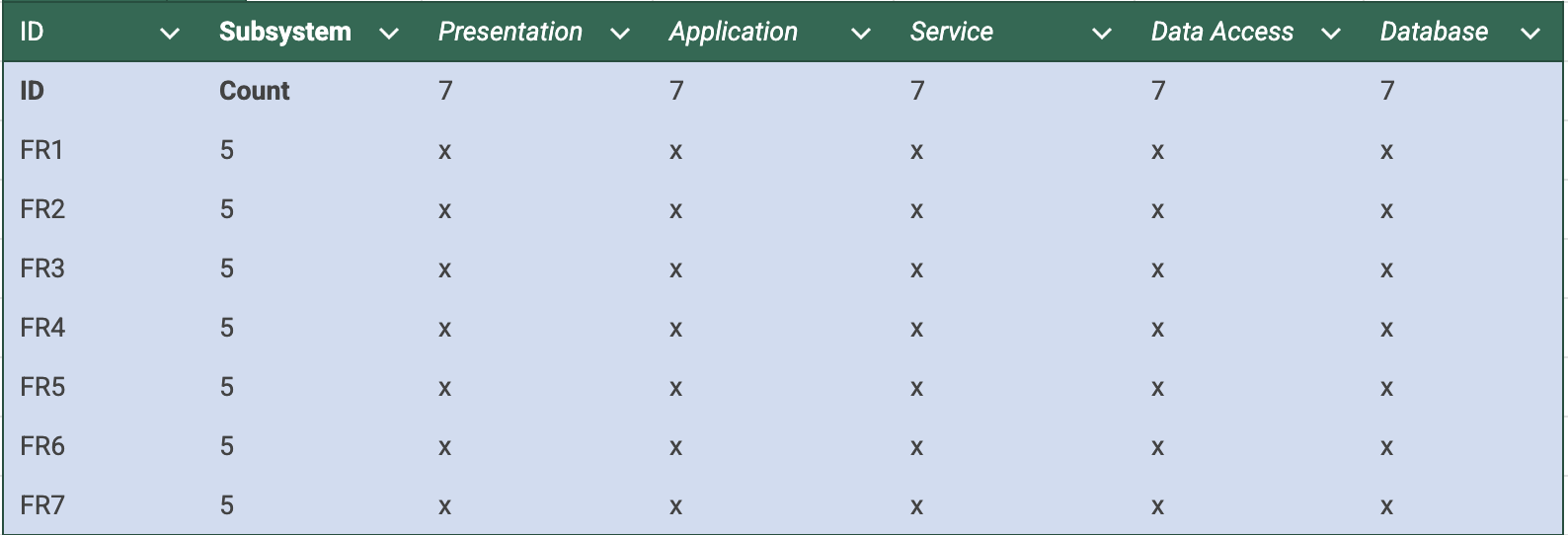
* 1. Rationale for your architectural style and model

Emission Analyzer uses a layered architecture approach, allowing for each subsystem to have a dedicated layer to implement its distinct functionality. This approach promotes modularity, maintainability, and scalability, as each layer has a specific role, changes in one layer do not intrinsically impact others, and each layer can be scaled independently of others.

The application’s subsystems include the Presentation Layer, Application Layer, Service Layer, Data Access Layer, and Database Layer. The Presentation Layer handles user interaction and the visual display of the data, the Application Layer manages the API endpoints and communication with the Service Layer, the Service Layer implements business logic and data validation, the Data Access Layer manages database interactions and implements object-relational mapping, and the Database Layer stores the data.

* 1. Traceability from requirements to architecture

Figure 2: Requirements Traceability Matrix



These are the following functional requirements:

* FR1: The system shall allow users to create an account.
* FR2: The system shall allow users to add an engine to their account.
* FR3: The system shall allow users to edit an engine associated with their account.
* FR4: The system shall allow users to delete an engine associated with their account.
* FR5: The system shall predict NOx and CO emission levels based on the parameter data associated with an engine.
* FR6: The system shall generate and display visualizations, showing the predicted NOx and CO emission levels.
* FR7: The system shall allow users to download the prediction data of all engines associated with their account.

# Design

* 1. GUI (Graphical User Interface) design

Figure 3: Landing Page



Figure 4: Home Page

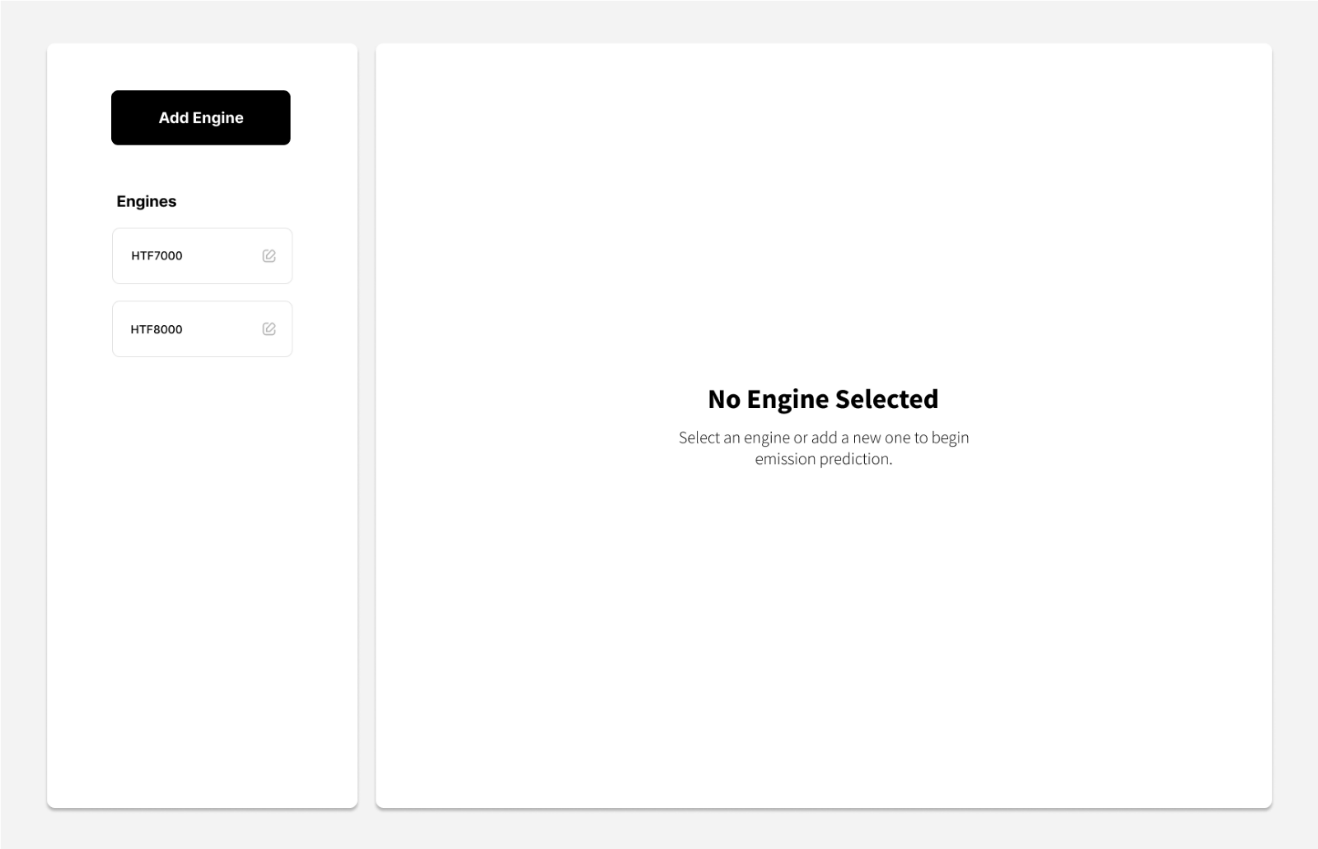


Figure 5: Add Engine Page

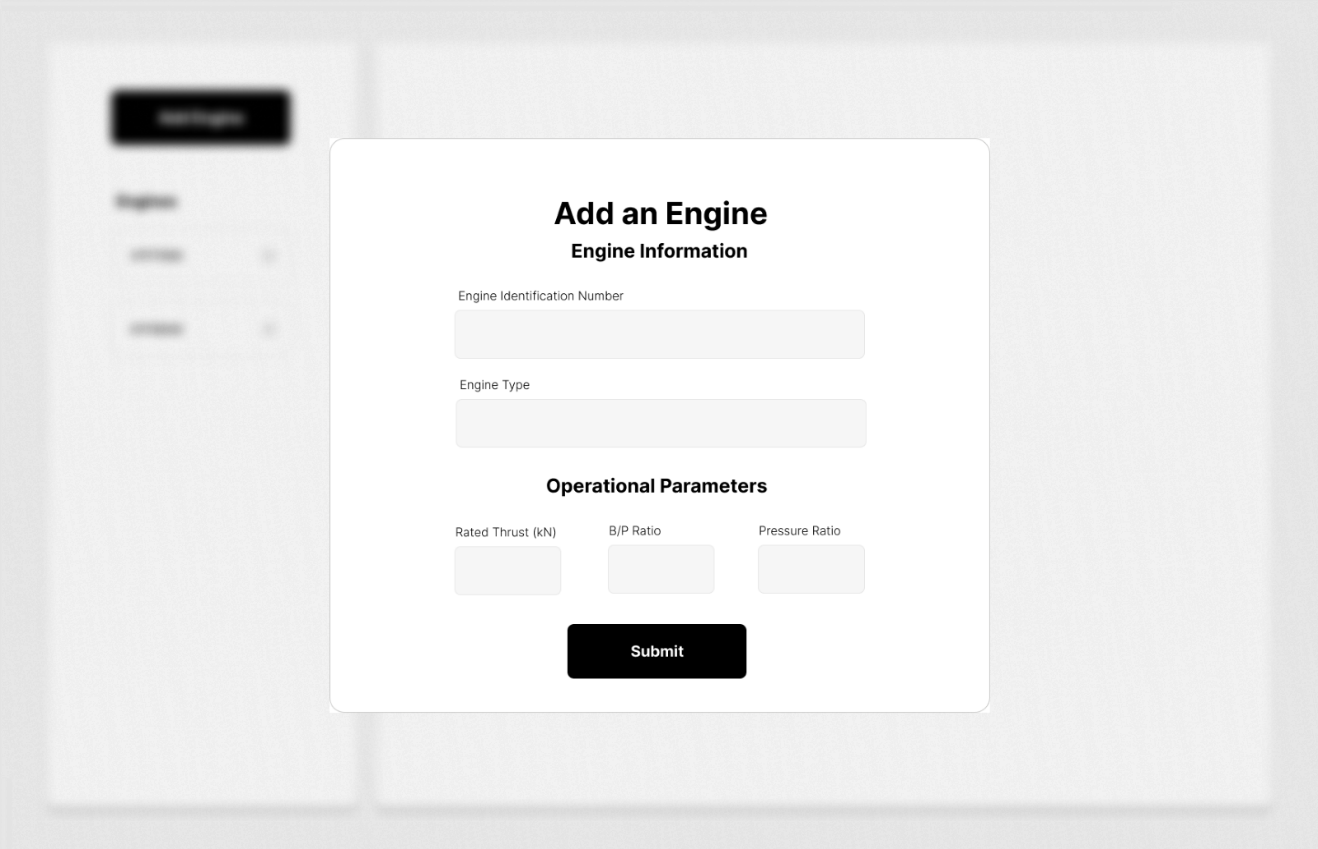


Figure 6: Engine Page

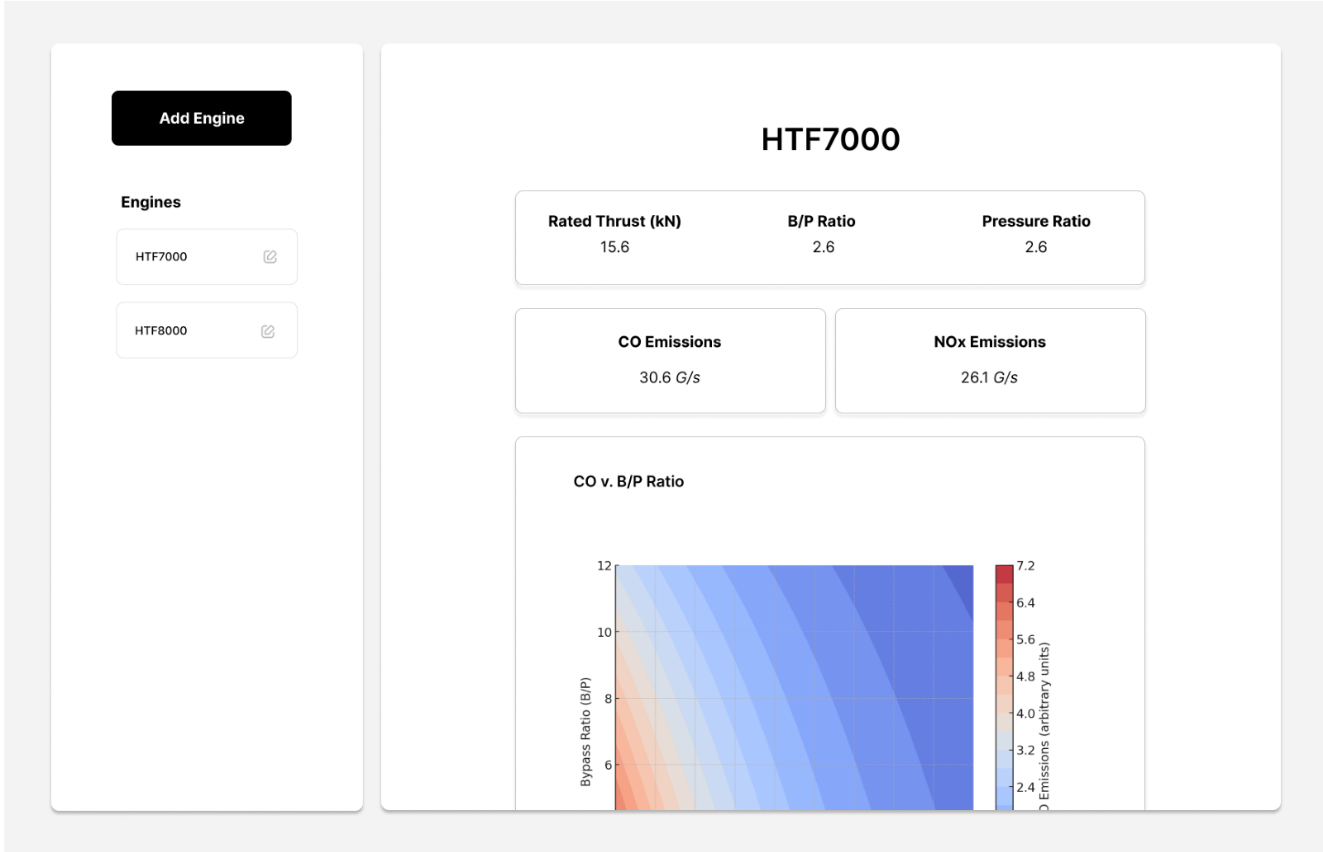
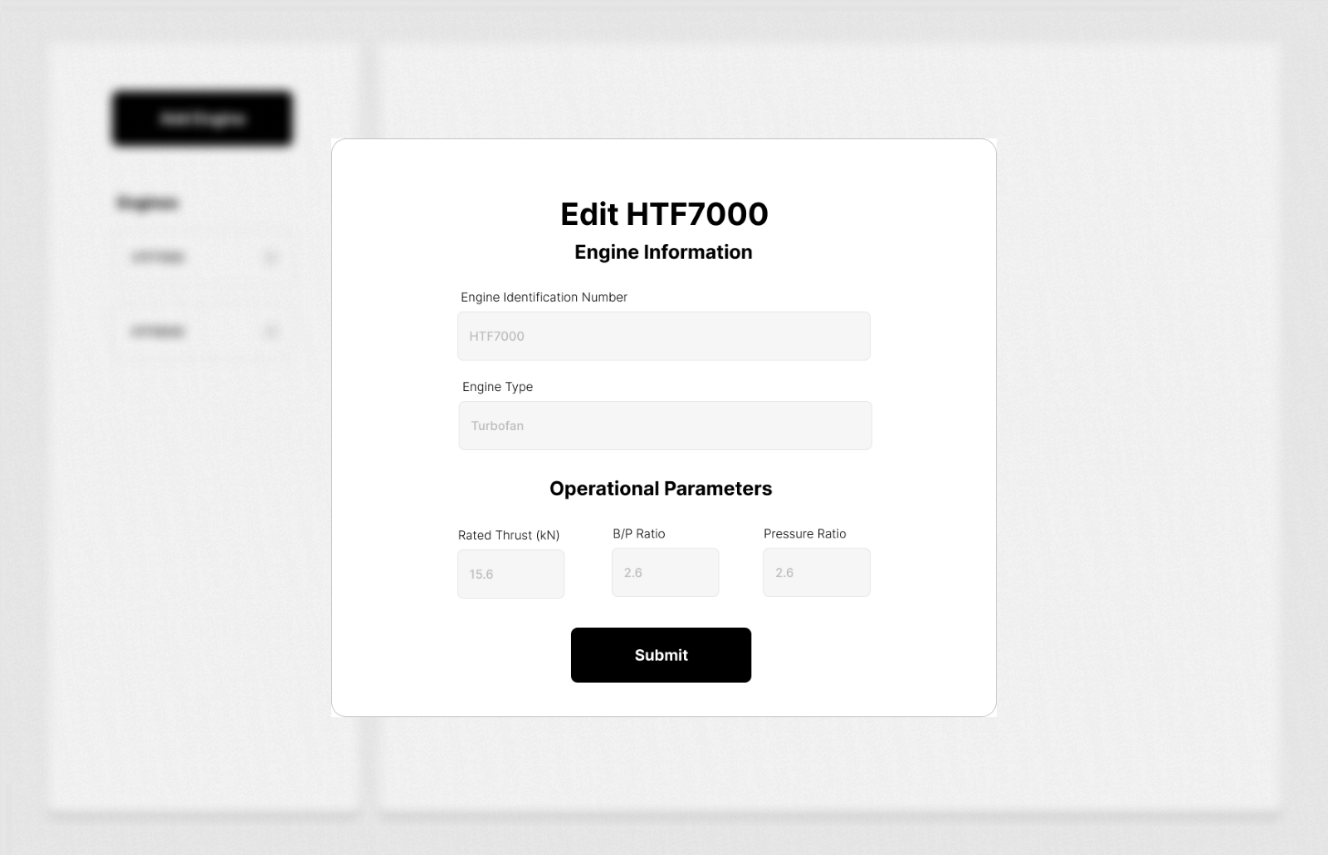


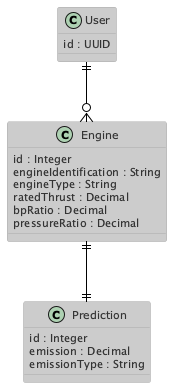
Figure 7: Edit Engine Page



* 1. Static model – class diagrams

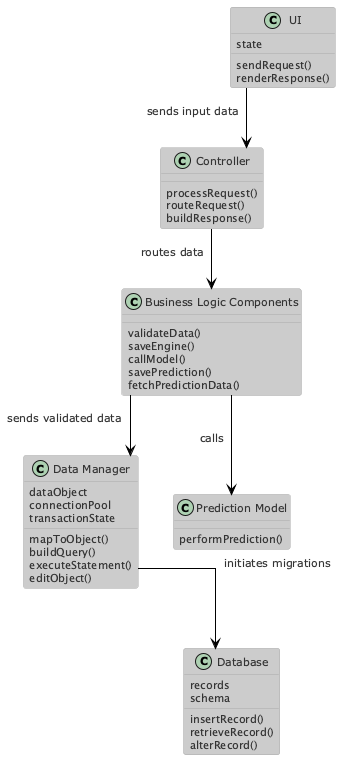
## Entity-Relationship

Figure 8: Entity-Relationship Class Diagram



Software Components

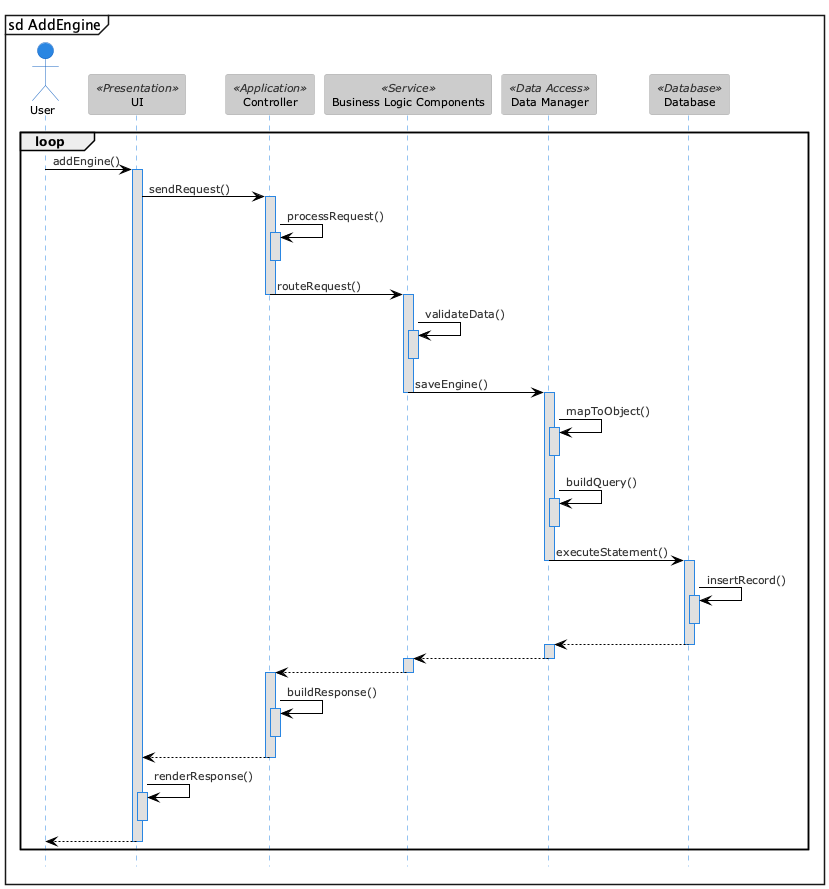
Figure 9: Software Components Class Diagram



* 1. Dynamic model – sequence diagrams

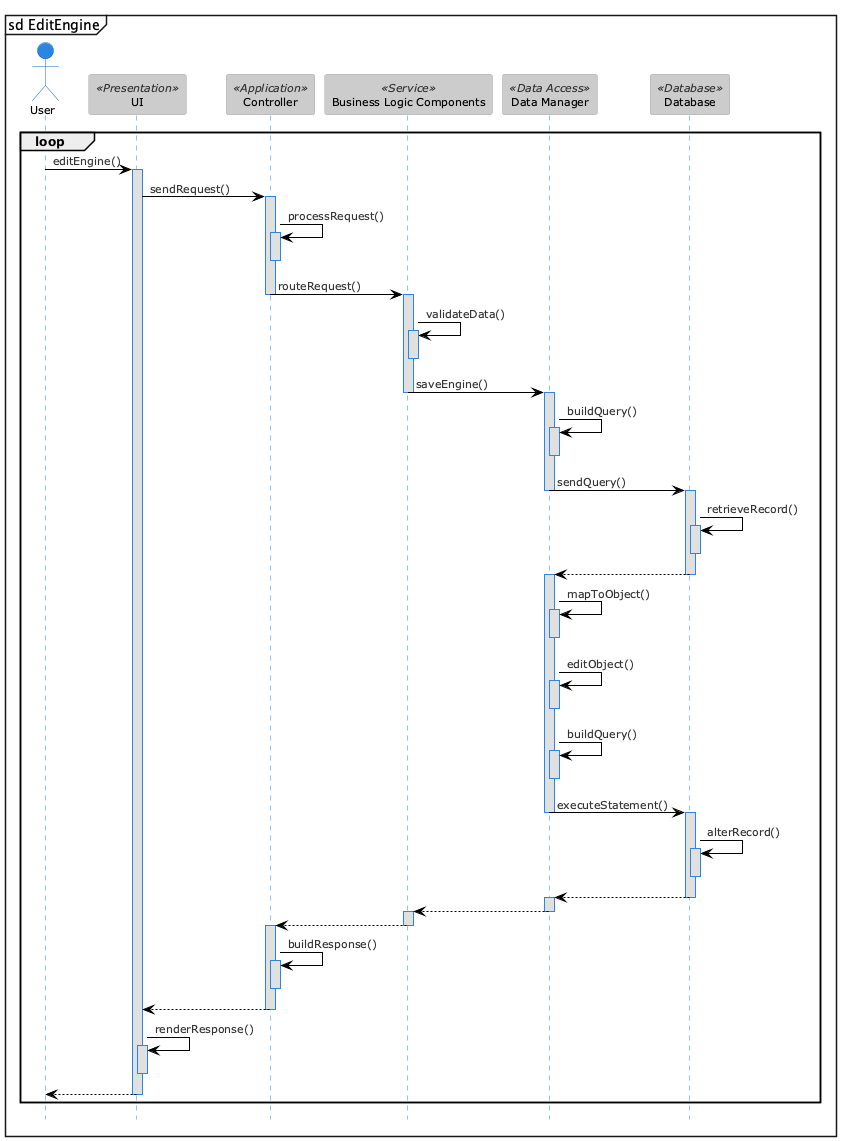
## Add Engine

Sequence Diagram 1: Add Engine



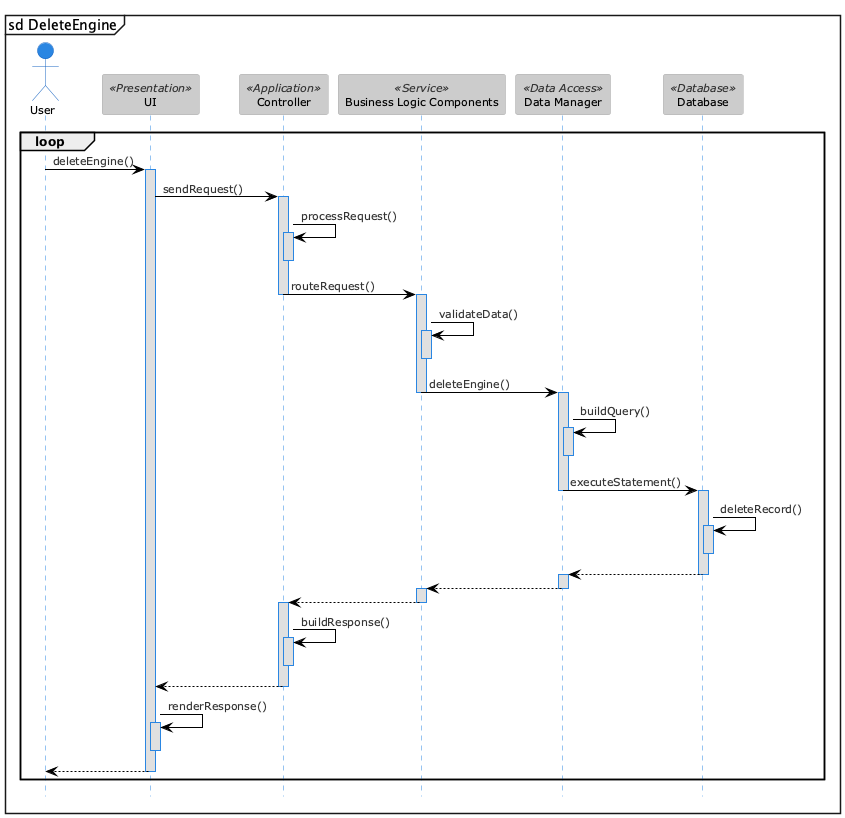
Edit Engine

Sequence Diagram 2: Edit Engine



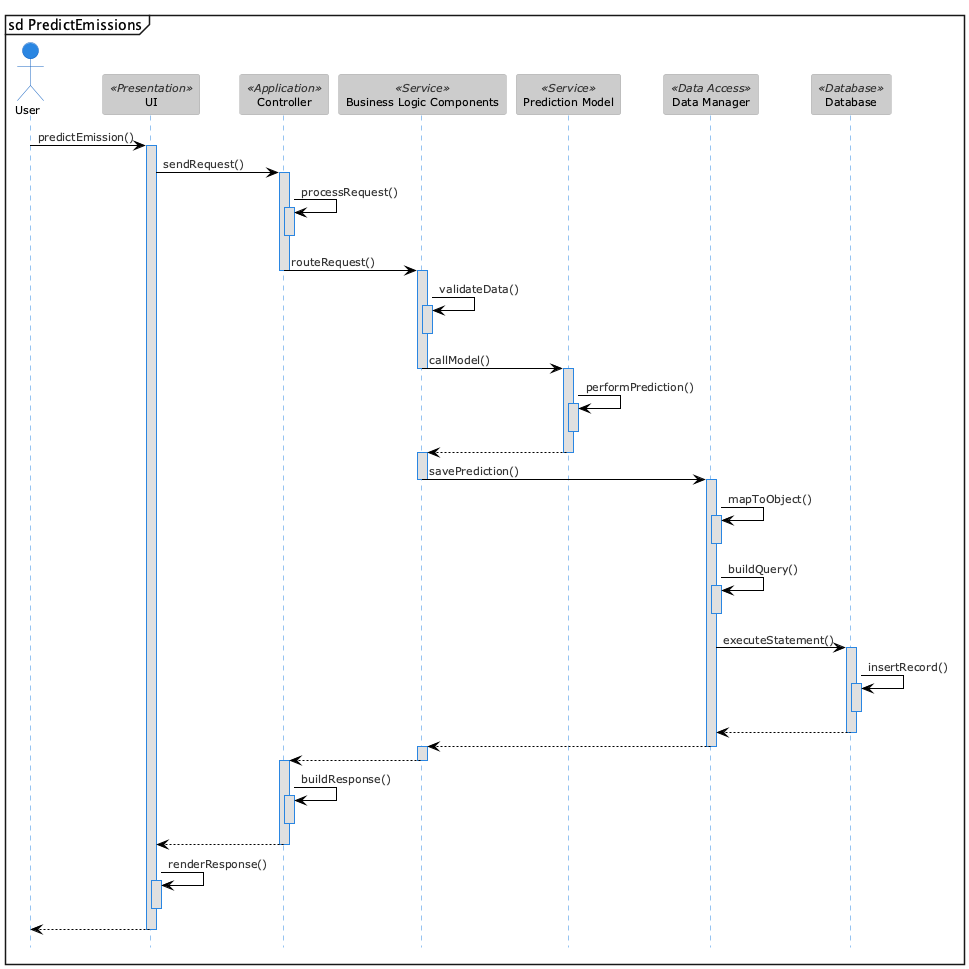
Delete Engine

Sequence Diagram 3: Delete Engine



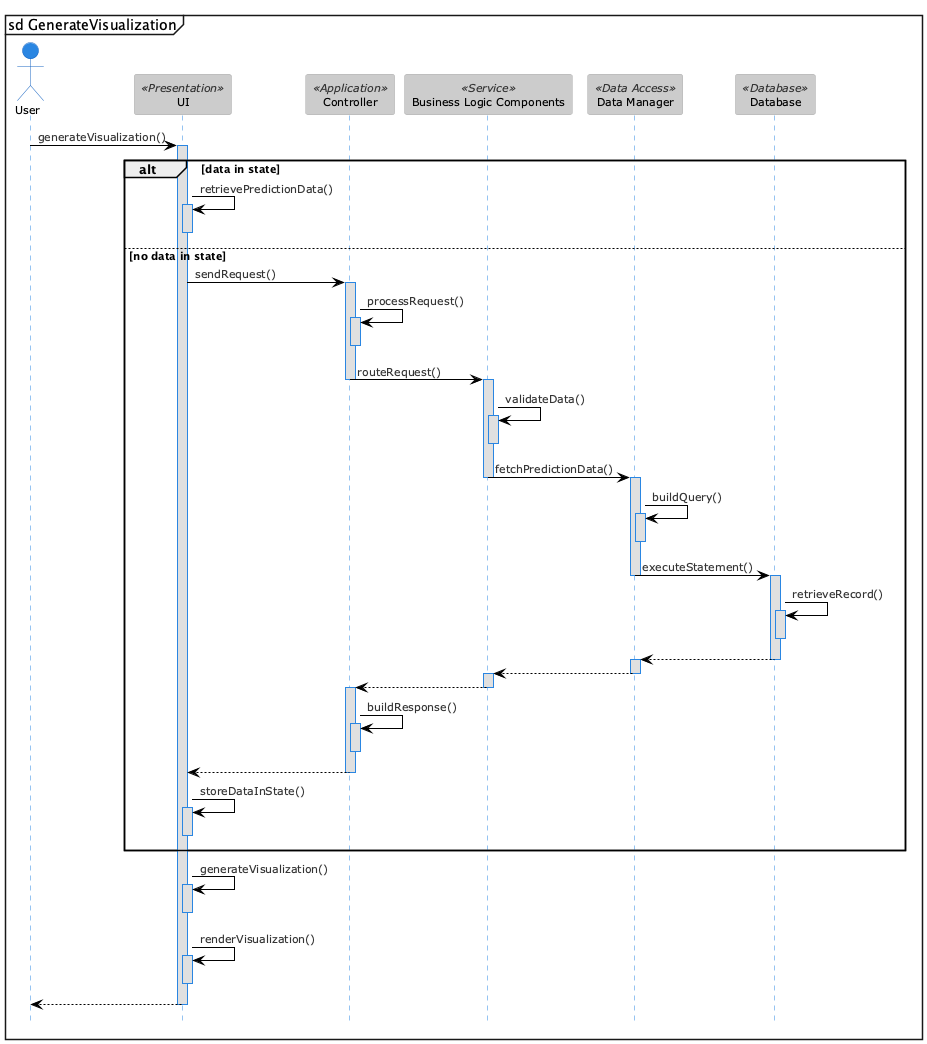
Predict Emissions

Sequence Diagram 4: Predict Emissions



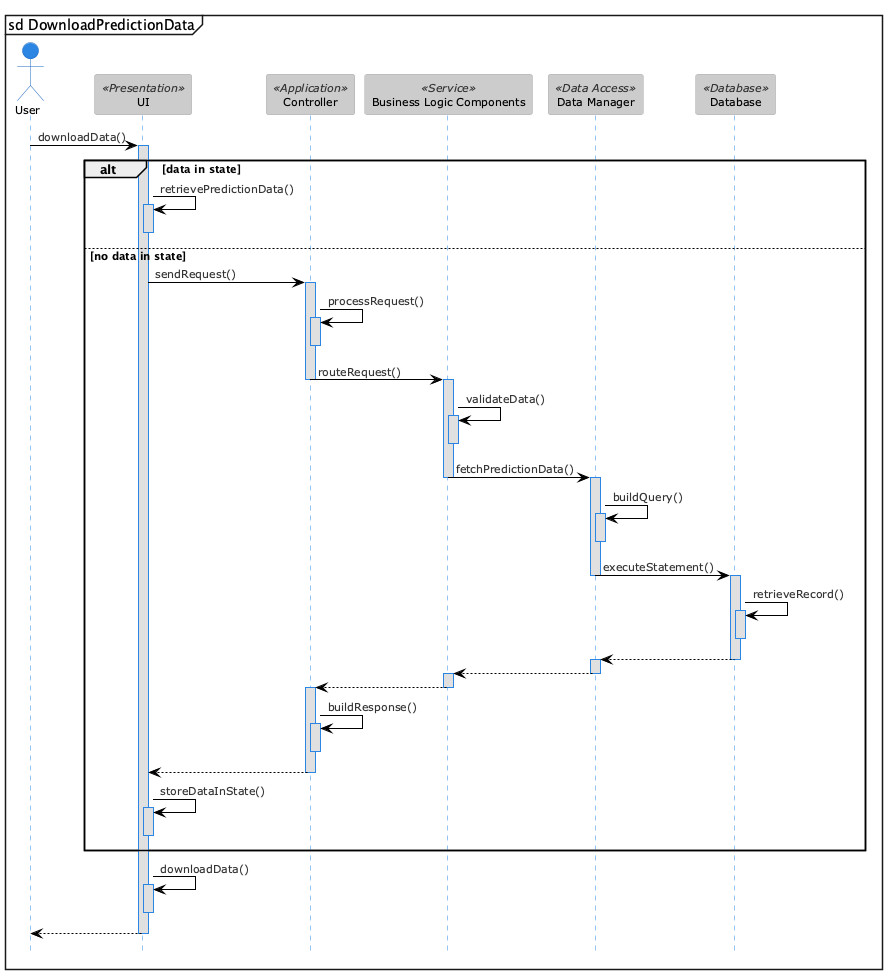
Generate Visualizations

Sequence Diagram 5: Generate Visualizations



Download Prediction Data

Sequence Diagram 6: Download Prediction Data



* 1. Rationale for your detailed design model

## GUI Rationale

### *Landing Page*

The landing page introduces the application’s purpose and objectives effectively, ensuring users immediately understand its value. Emission Analyzer’s landing page features a tagline summarizing its core functionality, helping users quickly grasp its main use case. It also highlights key features to attract user interest. Additionally, the landing page serves as an entry point, with a prominent “Get Started” button designed to streamline navigation. Clicking this button will provide users with immediate access to the application.

### *Home Page*

The home page is the first page users will see upon accessing the application. It consists of two sections: a side bar for navigation and a main display area that updates based on the selected option. The sidebar includes an “Add Engine” button and a dropdown menu listing the engines associated with a user’s account. If the user clicks the “Add Engine” button, the main display will render the form for entering and submitting the engine data. If the user selects a particular engine from the dropdown menu, the main display will render the data and visualizations associated with the engine. This page structure ensures an intuitive user experience by keeping navigation simple and concise.

### *Add Engine Page*

The Add Engine page is accessed when the user clicks the “Add Engine” button in the side bar. The main display will render a form that allows the user to add an engine to their account, with fields for the engine identification number, engine type, rated thrust, B/P ratio, and pressure ratio. The application will include error checking to ensure the data is valid. For example, the rated thrust, B/P ratio, and pressure ratio must be numerical values (either integers or decimals), while the engine identification number and engine type must be strings. When the user clicks “Submit,” the application will send a GET request to the back end to store the data. This page offers an intuitive way to add an engine to a user's account.

### *Engine Page*

The Engine page is accessed when the user selects an engine from the dropdown menu labeled “Engines.” The main display will show the engine’s information, including its engine identification number, engine type, and data parameters. Additionally, the page will provide options to perform emission predictions and generate visualizations, along with an option to edit the selected engine. This page serves as a visual representation of the data entered by the user, while also offering access to the core functionalities of the application.

### *Edit Engine Page*

The Edit Engine page is accessed by selecting the edit icon in the top-right corner of the Engine page. This is the only way to access this page, as an engine must first be selected for editing to be available. When the user clicks the edit icon, the main display will render a form that allows the user to modify the selected engine, with the same fields as those in the Add Engine form. However, the fields will be pre-filled with the data of the selected engine. This page provides an intuitive way for users to edit an engine associated with their account.

## Class Diagrams Rationale

The Entity-Relationship class diagram consists of three classes: User, Engine, and Prediction. The User class serves to establish the association between engine data and prediction data. The Engine class is responsible for storing information related to an engine, while the Prediction class stores data related to a prediction. The Engine class also allows a Prediction to be linked to it, and by extension, to a User. Since each class represents an entity to be stored in the database, no methods are associated with them.

The Software Components class diagram consists of six components: UI, Controller, Business Logic, Prediction Model, Data Manager, and Database. These components represent different layers of the application’s architecture. The UI displays information to the user, while the Controller routes requests to the appropriate Business Logic. The Business Logic validates data and forwards it either to the Prediction Model or the Data Manager, depending on the request. The Data Manager creates data objects and builds queries, and the Database stores the data.

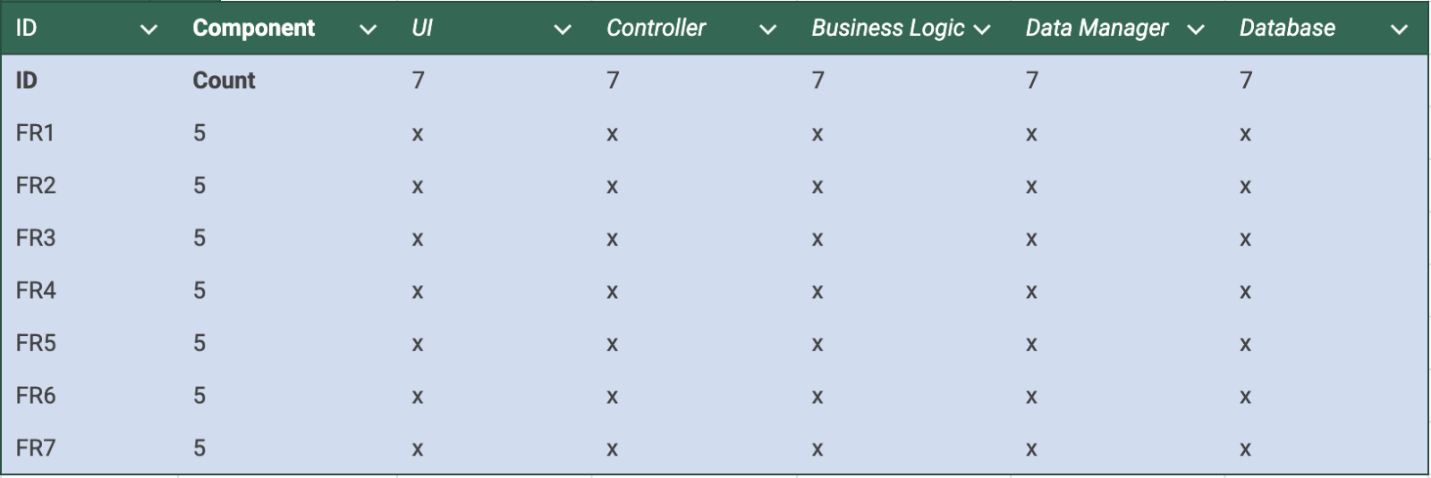
## Sequence Diagrams Rationale

For each functional requirement, all layers within the architecture work together to facilitate it. The Presentation Layer displays the user interface through the UI component and sends requests to the Application Layer. The Application Layer uses the Controller component to process and route these requests. The Service Layer implements the business logic, performing tasks such as emission prediction through the Prediction Model component. It also interacts directly with the Data Access Layer, which includes the Data Manager component. The Data Manager handles the object-relational mapping logic and communicates with the Database Layer to persist user-entered data or application-generated data (e.g., from the Prediction Model).

In each sequence diagram, the Presentation Layer remains inactive only when the Controller has responded to its initial request. This is because the client is asynchronously waiting for the response, ensuring the data is rendered to the user once received.

* 1. Traceability from requirements to detailed design model

Figure 10: Component Traceability

  
These are the following functional requirements:

* FR1: The system shall users to create an account.
* FR2: The system shall allow users to add an engine to their account.
* FR3: The system shall allow users to edit an engine associated with their account.
* FR4: The system shall allow users to delete an engine associated with their account.
* FR5: The system shall predict NOx and CO emission levels based on the parameter data associated with an engine.
* FR6: The system shall generate and display visualizations, showing the predicted NOx and CO emission levels.
* FR7: The system shall allow users to download the prediction data of all engines associated with their account.

# Test Plan

* 1. Requirements/specifications-based system level test cases

## Create User

Request 1: Create User



Test Case 1: Create User

|  |  |  |  |
| --- | --- | --- | --- |
| **Create User** | | | |
| Test Case | Prefix | Postfix | Expected Result |
| TC-1.1: Using an invalid key in the request body | Invalid key in the request body (e.g. first\_name) | Send request to the server | 400 Bad Request |
| TC-2.1: Providing valid key with the incorrect data type | Valid key, wrong data type (user\_id set to a numerical value) |
| TC-3: Submitting improperly formatted JSON | Malformed JSON (missing a curly brace or a quotation) |
| TC-4.1: Omitting required key from request body | Request body without user\_id |
| TC-5: Submitting duplicate key | Request body with valid, existing user\_id |
| TC-6: Valid user\_id key | Request body with valid user\_id | 200 OK |

## Add Engine

Request 2: Add Engine



Test Case 2: Add Engine

|  |  |  |  |
| --- | --- | --- | --- |
| **Add Engine** | | | |
| Test Case | Prefix | Postfix | Expected Result |
| TC-1.2: Using an invalid key in request body | Invalid key in the request body (e.g. combustor\_description) | Send request to server | 400 Bad Request |
| TC-2.2: Providing valid key with incorrect data type | Valid key, wrong data type (e.g. rated\_thrust: “4.5”) |
| TC-3: Submitting improperly formatted JSON | Invalid JSON syntax |
| TC-4.2: Omitting required key from request body | Request body without any required key |
| TC-7: Adding an engine that is already associated with a user | A valid request body that contains an engine identification number already associated with the user |
| TC-8: Providing an invalid engine type | A request body that contains an invalid engine type (e.g. engine\_type: “Diesel”) | 404 Not Found |
| TC-10: Using a nonexistent user\_id | Request body with a nonexistent engine\_id |
| TC-12: Using a valid user\_id and valid engine data | Request body with valid user\_id, valid engine data | 201 OK |

## Edit Engine

Request 3: Edit Engine



Test Case 3: Edit Engine

|  |  |  |  |
| --- | --- | --- | --- |
| **Edit Engine** | | | |
| Test Case | Prefix | Postfix | Expected Result |
| TC-1.2: Using an invalid key in request body | Invalid key in the request body | Send request to server | 400 Bad Request |
| TC-2.2: Providing valid key with incorrect data type | Valid key, wrong data type |
| TC-3: Submitting improperly formatted JSON | Invalid JSON syntax |
| TC-4.2: Omitting required key from request body | Request body without user\_id or engine |
| TC-8: Using an invalid engine type | Nonexistent engine type |
| TC-9: Using a nonexistent engine\_id | Request body with a nonexistent engine\_id | 404 Not Found |
| TC-10: Using a nonexistent user\_id | Request body with a nonexistent user\_id |
| TC-12: Using a valid user\_id and valid engine data | Request body with valid user\_id, valid engine data | 204 OK |

## Delete Engine

Request 4: Delete Engine



Test Case 4: Delete Engine

|  |  |  |  |
| --- | --- | --- | --- |
| **Delete Engine** | | | |
| Test Case | Prefix | Postfix | Expected Result |
| TC-1.3: Using an invalid key in request body | Invalid key in the request body | Send request to server | 400 Bad Request |
| TC-2.3: Providing valid key with incorrect data type | Valid key, wrong data type |
| TC-3: Submitting improperly formatted JSON | Invalid JSON syntax |
| TC-4.3: Omitting required key from request body | Request body without user\_id or engine |
| TC-9: Using a nonexistent engine\_id | Request body with a nonexistent engine\_id | 404 Not Found |
| TC-10: Using a nonexistent user\_id | Request boy with a nonexistent user\_id |
| TC-13: Valid user\_id, valid engine\_identification | Request body with a valid user\_id and engine\_identification |  | 200 OK |

## Predict Emissions

Request 5: Predict Emissions



Test Case 5: Predict Emissions

|  |  |  |  |
| --- | --- | --- | --- |
| **Predict Emissions** | | | |
| Test Case | Prefix | Postfix | Expected Result |
| TC-1.3: Using an invalid key in the request body | Invalid key in the request body | Send request to the server | 400 Bad Request |
| TC-2.3: Providing valid key with the incorrect data type | Valid key, wrong data type (e.g. setting rated\_thrust as a string) |
| TC-3: Submitting improperly formatted JSON | Malformed JSON (missing a curly brace or a quotation) |
| TC-4.3: Omitting required key from request body | Request body that is missing user\_id, rated\_thrust, bp\_ratio, pressure\_ratio, or all |
| TC-11: Prediction Model is unavailable | Request body with valid keys and their correct data types | 503 Service Unavailable |
| TC-9: Using a nonexistent engine\_id | Request body with a nonexistent engine\_id | 404 Not Found |
| TC-8: Using a nonexistent user\_id | Request boy with a nonexistent user\_id |
| TC-13: Valid user\_id, valid engine\_identification | Request body with a valid user\_id and engine\_identification | 200 OK |

## Generate Visualizations

Request 6: Generate Visualizations



If the prediction data for the engine is unavailable, the application will send a request to the server to retrieve it. It has the same test cases as Predict Emissions as it uses the same endpoints and data models.

Test Case 6: Generate Visualizations

|  |  |  |  |
| --- | --- | --- | --- |
| **Generate Visualizations** | | | |
| Test Case | Prefix | Postfix | Expected Result |
| TC-13: Valid user\_id, valid engine\_identification | Request body with a valid user\_id and engine\_identification | Send request to server | 200 OK |

## Download Prediction Data

Request 7: Download Prediction Data



If the prediction data for the engine is unavailable, the application will send a request to the server to retrieve it. It has the same test cases as Predict Emissions as it uses the same endpoints and data models.

Test Case 7: Download Prediction Data

|  |  |  |  |
| --- | --- | --- | --- |
| **Download Prediction Data** | | | |
| Test Case | Prefix | Postfix | Expected Result |
| TC-13: Valid user\_id, valid engine\_identification | Request body with a valid user\_id and engine\_identification | Send request to server | 200 OK |

* 1. Techniques used for test generation

## Techniques

To generate our tests, we will employ both scenario testing and unit testing. Unit tests for our database models will help ensure that data is being correctly inserted into the database, while scenario tests will verify that the system functions as expected from end to end.

Our testing strategy incorporates both black-box and white-box testing approaches. Black-box testing is used in our scenario tests to validate system functionality based on various user inputs, without knowledge of the internal code. In contrast, white-box testing is applied in our unit tests to verify the internal logic and structure of our code.

* 1. Assessment of the goodness of your test suite

## Criteria

To measure the quality of our tests, we will use criteria such as code coverage (aiming for 90 percent and above), inclusion of edge cases, repeatability of results, and clarity of test logic. We will also ensure that each test targets one behavior and runs independently.

* 1. Traceability of test cases to use cases

*Table 6: Traceability Matrix*

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Functional Req. Tested | FR1 | FR2 | FR3 | FR4 | FR5 | FR6 | FR7 |
| Test Cases | 47 | 6 | 7 | 7 | 6 | 7 | 7 | 7 |
| TC-1.1 | 1 | X |  |  |  |  |  |  |
| TC-1.2 | 2 |  | X | X |  |  |  |  |
| TC-1.3 | 4 |  |  |  | X | X | X | X |
| TC-2.1 | 1 | X |  |  |  |  |  |  |
| TC-2.2 | 2 |  | X | X |  |  |  |  |
| TC-2.3 | 4 |  |  |  | X | X | X | X |
| TC-3 | 7 | X | X | X | X | X | X | X |
| TC-4.1 | 1 | X |  |  |  |  |  |  |
| TC-4.2 | 2 |  | X | X |  |  |  |  |
| TC-4.3 | 4 |  |  |  | X | X | X | X |
| TC-5 | 1 | X |  |  |  |  |  |  |
| TC-6 | 1 | X |  |  |  |  |  |  |
| TC-7 | 1 |  | X |  |  |  |  |  |
| TC-8 | 6 |  | X | X | X | X | X | X |
| TC-9 | 5 |  |  | X | X | X | X | X |
| TC-10 | 2 |  | X | X |  |  |  |  |
| TC-11 | 3 |  |  |  |  | X | X | X |
| TC-12 | 2 |  | X | X |  |  |  |  |
| TC-13 | 1 |  |  |  | X |  |  |  |

*Test Description 1: Create User*

|  |  |
| --- | --- |
| **FR1: Create User** | |
| TC-1.1 | Restricts request box to only valid key (e.g. user\_id) |
| TC-2.1 | Enforces correct data type (e.g. user\_id set to numerical value) |
| TC-3 | Enforces proper JSON format |
| TC-4.1 | Restricts omission of keys. (e.g. missing user\_id) |
| TC-5 | Restricts duplication of user\_id |
| TC-6 | Tests validation of user\_id |

*Test Description 2: Add Engine*

|  |  |
| --- | --- |
| **FR2: Add Engine** | |
| TC-1.2 | Restricts request box to only valid keys (e.g. user\_id, engine\_type, rated\_thrust) |
| TC-2.2 | Enforces correct data type (e.g. user\_id set to string) |
| TC-3 | Enforces proper JSON format |
| TC-4.2 | Restricts omission of keys (e.g. user\_id, engine\_type, rated\_thrust) |
| TC-7 | Restricts a user having duplicate engines |
| TC-8 | Enforces the use of a valid engine type |
| TC-10 | Enforces that the user must already exist |
| TC-12 | Tests validation of user\_id and engine data |

*Test Description 3: Edit Engine*

|  |  |
| --- | --- |
| **FR3: Edit Engine** | |
| TC-1.2 | Restricts request box to only valid keys (e.g. user\_id, engine\_type, rated\_thrust) |
| TC-2.2 | Enforces correct data type (e.g user\_id set to numerical value, or rated\_thrust being a string) |
| TC-3 | Enforces proper JSON format |
| TC-4.2 | Restricts Omission of keys (e.g user\_id, engine\_type, rated\_thrust, etc.) |
| TC-8 | Enforces the use of a valid engine type |
| TC-9 | Enforces that the engine must already exist |
| TC-10 | Enforces that the user must already exist |
| TC-12 | Tests validation of user\_id and engine data |

*Test Description 4: Delete Engine*

|  |  |
| --- | --- |
| **FR4: Delete Engine** | |
| TC-1.3 | Restricts request box to only valid keys (e.g. user\_id, engine\_identification) |
| TC-2.3 | Enforces correct data type (e.g. user\_id being a string) |
| TC-3 | Enforces proper JSON format |
| TC-4.3 | Restricts omission of keys (e.g user\_id, engine\_identification) |
| TC-10 | Enforces that the user must already exist |
| TC-9 | Enforces that the engine must already exist |
| TC-13 | Tests validation of user\_id and engine\_identification |

*Test Description 5: Perform Emission Prediction, Generate Visualizations, Download Prediction Data*

|  |  |
| --- | --- |
| **FR5: Predict Emissions, FR6: Generate Visualizations,**  **FR7: Download Prediction Data** | |
| TC-1.3 | Restricts request box to only valid keys. (e.g. user\_id, engine\_identification) |
| TC-2.3 | Enforces correct data type |
| TC-3 | Enforces proper JSON format |
| TC-4.3 | Restricts omission of keys (e.g. user\_id, engine\_identification) |
| TC-10 | Enforces that the user must already exist |
| TC-9 | Enforces that the engine must already exist |
| TC-11 | Enforces that the Prediction Model must be available |

# Evidence the Document Has Been Placed under Configuration Management

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Version | Checked Out By | Checked In By | Changes Made | Reviewers | Ship-It Status |
| V1.1 | Greg | Greg | Added Test plan, Traceability of test cases, Req for Test Generation | Victor, Brie | ✅ |
| V1.2 | Srinesh | Srinesh | Added Design section | Brie, Josh | ✅ |
| V1.3 | Yousuf | Yousuf | Added Architecture section | Victor, Josh | ✅ |
| V1.4 | Victor | Victor | Added Requirements (Functional and Nonfunctional); Rational of Use Case, Use cases graphics, Textual Description of Use Case; Nonfunctional Reqs. | Brie, Josh | ✅ |
| V1.5 | Josh | Josh | Added Project Management section | Brie, Srinesh | ✅ |

# Engineering Standards and Multiple Constraints

* IEEE Std 1058-1998: Software Project Management Plans
* PMBOK® Guide: Project Management Body of Knowledge
* IEEE Std 12207: Software Life Cycle Processes
* IEEE Std 15939: Measurement Process
* ISO/IEC/IEEE Std 29148-2018: Systems and Software Engineering
* Requirements Engineering [[pdf](https://course.techconf.org/se4485/IEEE/ISO-IEC-IEEE-29148-2018.pdf)]
* 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
  1. Life Cycle Processes
* IEEE Recommended Practice for Architectural Description of Software-Intensive Systems
* IEEE Std 1471-2000: Software Architecture
* IEEE Std 1016-1998-(Revision-2009): Software Design
* IEEE Std 829-1983: Software Testing [[pdf](https://course.techconf.org/se4485/IEEE/IEEE%20Std%20829-1983-Software-Testing.pdf)]
* ISO/IEC/IEEE Std 29119-1-(Revision-2022): Part 1 - Software Testing General Concepts [[pdf](https://course.techconf.org/se4485/IEEE/IEEE-Std-29119-1-(Revision-2022)-Software-Testing-General-Concepts.pdf)]
* ISO/IEC/IEEE Std 29119-2-(Revision-2021): Part 2 - Test Process [[pdf](https://course.techconf.org/se4485/IEEE/IEEE-Std-29119-2-(Revision-2021)-Test-Process.pdf)]
* ISO/IEC/IEEE Std 29119-3-(Revision-2021): Part 3 - Test Documentation [[pdf](https://course.techconf.org/se4485/IEEE/IEEE-Std-29119-3-(Revision-2021)-Test-Documentation.pdf)]
* ISO/IEC/IEEE Std 29119-4-(Revision-2021): Part 4 - Test Techniques [[pdf](https://course.techconf.org/se4485/IEEE/IEEE%2029119.4%20(2021)%20-%20Test%20Techniques.pdf)]

# Additional References

* 1. Larson, E. and Gray, C., 2014. *Project Management: The Managerial Process.* McGraw Hill
  2. Humphrey, W.S. and Thomas, W.R., 2010. *Reflections on Management: How to Manage Your Software Projects, Your Teams, Your Boss, and Yourself.* Pearson Education
  3. Lamsweerde, A.V., 2009. *Requirements Engineering: From System Goals to UML Models to**Software Specifications.* John Wiley
* Lattanze, A.J., 2008. *Architecting Software Intensive Systems: A Practitioner’s Guide.* CRC Press
* Bass, L., Clements, P. and Kazman, R., 2003. *Software Architecture in Practice.* Addison-Wesley
* Larman, C., 2012. *Applying UML and Patterns: An Introduction to Object Oriented Analysis and Design and Iterative Development.* Pearson Education.
* Arlow, J., 2005. *Analysis and Design with UML 2 and UP*. Zuhike Engineering.
* Hyman, B., 1998. *Fundamentals of Engineering Design.* New Jersey: Prentice Hall
* Simon, H.A., 2014. *A Student's Introduction to Engineering Design: Pergamon Unified Engineering Series* (Vol. 21). Elsevier
* Jorgensen, P.C., 2013. *Software Testing: A Craftsman's Approach.* Auerbach Publications
* Mathur, A.P., 2013. *Foundations of Software Testing, 2/e.* Pearson Education
* Amman, Offutt, 2016, *Introduction to Software Testing*. Cambridge University Press

# Acknowledgment

We would like to thank our sponsors Trey Williams, Trevor A. Lang, Ryan Havens, and Marc Perna for providing us with their guidance throughout the development of Emission Analyzer.