SE 4485: Software Engineering Projects

Spring 2024

Final Report

| Group Number | **8** |
| --- | --- |
| Project Title | **Crop Yield Prediction Model** |
| Sponsoring Company |  |
| Sponsor(s) | **Marc J. Perna (*Main Sponsor*)** |
| Students | **1. Ryan Haven (*Team Leader*)**  **2. Cameron Sutton**  **3. Melvin Sajeev**  **4. Ibrahim Barney**  **5. Nisai Sun** |

# Executive Summary

**Overview:** Our crop yield prediction application harnesses the power of historical crop yield data, coupled with state and county-specific information, to provide accurate forecasts for agricultural production. By analyzing past yield trends and regional characteristics, our application offers valuable insights to farmers and stakeholders, aiding in decision-making processes and enhancing overall farming efficiency.

**Key Features:**

1. **Historical Data Analysis:** Utilizes historical crop yield data from previous years to identify patterns and trends, enabling accurate predictions for future crop production.
2. **Regional Specificity:** Incorporates state and county-level data such as soil types, precipitation levels, temperature variations, and crop rotation practices to tailor predictions to specific geographic regions.
3. **Statistical Modeling:** Employs statistical techniques and predictive analytics to analyze historical data and forecast crop yields, providing reliable insights for farmers and agribusinesses.
4. **User-Friendly Interface:** Offers an intuitive interface for easy data input and interpretation, allowing farmers to access predictions seamlessly and make informed decisions.
5. **Risk Assessment:** Assesses potential risks and uncertainties associated with crop production, such as weather fluctuations and pest outbreaks, enabling proactive risk management strategies.
6. **Decision Support Tools:** Provides decision support tools including yield maps, crop rotation suggestions, and planting recommendations to optimize farming practices and maximize yields.
7. **Accessibility:** Accessible via web platforms, ensuring convenience and flexibility for farmers to access predictions and insights anytime, anywhere.

**Benefits:**

1. **Precision Agriculture:** Facilitates precision agriculture practices by providing accurate predictions tailored to specific regions, optimizing resource allocation and improving crop management strategies.
2. **Risk Mitigation:** Identifies and mitigates potential risks associated with crop production, enabling farmers to implement proactive measures and minimize losses due to adverse conditions.
3. **Data-Driven Decisions:** Empowers farmers with data-driven insights derived from historical analysis, facilitating informed decision-making and enhancing overall farm productivity.
4. **Resource Optimization:** Optimizes resource usage including water, fertilizers, and pesticides, leading to improved efficiency, cost savings, and environmental sustainability.
5. **Market Competitiveness:** Enhances competitiveness by enabling farmers to produce high-quality crops efficiently, meet market demands, and adapt to changing environmental and economic conditions.

**Market Potential:** With the increasing demand for food security and sustainable agricultural practices, there is a growing need for innovative technologies to improve crop production and mitigate risks. Our crop yield prediction application addresses this demand by offering a reliable and cost-effective solution tailored to specific geographic regions. As a result, it presents significant opportunities for adoption by farmers, agribusinesses, and policymakers seeking to enhance agricultural productivity and sustainability.

**Conclusion:** Our crop yield prediction application leverages historical crop yield data and regional specificity to provide accurate forecasts for agricultural production. By empowering farmers with actionable insights and decision support tools, we enable them to optimize crop management strategies, mitigate risks, and enhance overall farming efficiency. With its user-friendly interface, tailored predictions, and tangible benefits, our application is poised to revolutionize crop production practices and contribute to a more sustainable and resilient food system.

# Table of Contents

[**Executive Summary**](#_wc0o814bk3so) **2**

[**Table of Contents**](#_j80mhxgpjeq0) **5**

[**List Of Figures**](#_4klb6mispism) **7**

[**List Of Tables**](#_ig3k1tv3tet4) **8**

[**Introduction**](#_sglpdf3bsm9) **9**

[**Project Management Plan**](#_v0hct38l636p) **10**

[Project Organization](#_oi1z6rmnhocd) 10

[Process Risks](#_58psvqvqtkxz) 10

[**Lifecycle Model**](#_7ztjkser357f) **11**

[**Risk Analysis**](#_93l5dbim8qh7) **12**

[**Possible Opportunities**](#_y6vvux6v70wn) **14**

[**Software and Hardware Resource Requirements**](#_9fi9b6v0l51o) **15**

[**Deliverable Schedule**](#_at48j4c5dpjo) **16**

[**Monitor / Control Mechanisms**](#_vay9z7icbnwl) **19**

[**Professional Standards**](#_6pi7pdv1e615) **20**

[**Impact of the project on individuals and organizations**](#_77tlkmjlutjp) **21**

[**Requirement Specifications**](#_93cx7qn68pj) **22**

[Stakeholders](#_m7srq6313qln) 22

[Use Cases](#_xkrm05w9vsrm) 22

[Use Case: Data PreProcessing](#_ads4frl65uqt) 22

[Use Case: Data Integration\*](#_bp7nbytn0hgn) 24

[Use Case: Machine Learning Model](#_daehe2wuu3v2) 25

[Use Case: Output Formatting](#_kfbcnf32dpn2) 26

[Use Case: Accuracy and Performance](#_w6m02q3lkvha) 27

[Use Case: User Interface](#_fzttlj4dea74) 29

[**Rationale For Use Case Model**](#_i3oqtoy02jv8) **31**

[**Non-Functional Requirements**](#_hh0u5qkjb81c) **32**

[**Future Requirements**](#_i1piffga6wf) **33**

[**Functional Requirements**](#_x26csbh8qpea) **34**

[**Traceability Table**](#_2ykaz0krmmsa) **36**

[**Client-Server Architecture**](#_3gr2542kne74) **37**

[**Architecture Model**](#_uoi3hk1aa1yj) **38**

[**Technology, Software, Hardware Used**](#_24oniud6mzc3) **39**

[**Rationale for your architectural style and model**](#_vhhm4t9p0z9o) **41**

[**Traceability From Requirements To Architecture**](#_sw37a0aqveyp) **42**

[**Design**](#) **43**

[GUI Design](#_67ypzdfrmz62) 43

[GUI Description](#_s844cgl9w2bb) 43

[Class Diagram](#_xj4rsuyfwasu) 44

[Class Diagram Description](#_mjmu0qjw07fs) 44

[Deployment Diagram](#_wo15yd9mf2rm) 45

[Deployment Diagram Description](#_jd3yuv9wjdzu) 45

[**Dynamic Models**](#_s7yelosevh5y) **47**

[Sequence Diagram](#_87dne5f6tseq) 47

[Sequence Diagram Description](#_dcd9crxo158s) 47

[Sequence Diagram predict() Diagram](#_ahbdzngdifwb) 48

[Sequence Diagram predict() Description](#_e57mlbk3a5tu) 48

[**Rationale for Detailed Design Model**](#_y850y8tp7vtt) **49**

[**Traceability From Requirements To Design Model**](#_kovbj2m7avvc) **51**

[**Requirements/Specifications-Based System Level Test Cases**](#_ewjlwcqvcach) **52**

[**Techniques for Test Generation**](#_z6du08dk1uzk) **55**

[**Traceability of Test Cases to Use Cases**](#_kglqgim8fwy6) **57**

[**Evidence of Report Under Configuration Management**](#_sk82karohqx8) **58**

[**Configuration Management Table**](#_ny884fwovkl6) **59**

[**Engineering Standards and Multiple Constraints**](#) **60**

[**Additional References**](#) **62**

[**Acknowledgments**](#) **65**

[Raytheon](#_vwn5broq679t) 65

[Marc Perna / Daryll Nelson (Raytheon)](#_p4m53o9j0h0v) 65

[Dr. Eric Wong](#_g64br7pgb0ru) 65

[Brooke Havens (Ryan Havens Spouse)](#_jfkwisq16jue) 65

# 

# List Of Figures

* [Data Preprocessing Use Case](#_xkrm05w9vsrm)
* [Data Integration Use Case](#_bp7nbytn0hgn)
* [Machine Learning Model Use Case](#_daehe2wuu3v2)
* [Output Formatting Use Case](#_kfbcnf32dpn2)
* [Accuracy and Performance Use Case](#_w6m02q3lkvha)
* [User Interface Use Case](#_fzttlj4dea74)
* [Architecture Model](#_uoi3hk1aa1yj)
* [GUI Design Images](#_67ypzdfrmz62)
* [Deployment Diagram](#_wo15yd9mf2rm)
* [Sequence Diagram](#_87dne5f6tseq)
* [predict() Sequence Diagram](#_ahbdzngdifwb)

# List Of Tables

* [Deliverable Schedule Table](#_at48j4c5dpjo)
* [Non Functional Requirements Table](#_hh0u5qkjb81c)
* [Future Requirements Table](#_i1piffga6wf)
* [Functional Requirements Table](#_x26csbh8qpea)
* [Traceability Table](#_2ykaz0krmmsa)
* [Traceability From Requirements To Architecture Table](#_sw37a0aqveyp)
* [Traceability From Requirements To Design Model Table](#_kovbj2m7avvc)
* [Techniques For Test Generation Table](#_z6du08dk1uzk)
* [Traceability Of Test Cases To Use Cases](#_kglqgim8fwy6)
* [Configuration Management Table](#_ny884fwovkl6)

# Introduction

In today's rapidly changing world where climate change poses a looming threat to our agricultural systems and food security, our project aims to develop a robust crop yield prediction model. We will be using modern cloud and data science tools to leverage datasets from data.gov that encompass historical crop yield data. We endeavor to address the pressing need for understanding the trends in crop yield. The prediction model will help in informing crop rotation, watering, and other useful information to optimize crop return. This document goes over our project, we will get into the initial plan for our project followed by the requirements that outline the project and then both the architecture and design of the project, lastly we will prove that we have fulfilled our requirements with a test plan.

# Project Management Plan

### Project Organization

Ryan will be the team leader for this project. The team will prioritize tasks and assign collaboratively. Ryan will make sure everything is allocated equitable and will break any ties.

### Process Risks

* Out Of Date Documentation:
  + **Description**: Server/main host documentation is out of date.
  + **Process**: All documentation will be updated the day of the application update implementation.
* Team does not have access to source code locally in offline situation:
  + **Description:** team member forgets to clone/update local repository, does not have access to most up to date version.
  + **Process:** Provide a backup of the most up to date source code.
* Team member needs to be caught up to speed:
  + **Description:** Team member either misses a meeting or important update.
  + **Process:** all updates will be posted /provided in discord for viewing.
* Problems with communicating with sponsors:
  + **Description:** Problem communicating with a sponsor in a specified week
  + **Process:** Professors notified. Continue with production and prepare questions / conversation topics for the following week.

# Lifecycle Model

Our project will be developed using an Iterative model. Our project can be split up into several well-defined steps (Collection, Preprocessing, Processing, Visualization, etc.) and an iterative lifecycle model will let us develop our project most efficiently.

Since we have a broad knowledge of each step but not much detailed information on the best way to implement each step, an iterative process will let us take on each step one at a time and iterate/improve on it until we have something that works smoothly and robustly. By doing this for each step of our project, we will be left with a well-polished product that should work very smoothly.

We first considered a Waterfall lifecycle but decided against it since it offers very little support for going back and changing something after it has been created. We would have been locked into our first design choice which isn’t ideal since we wouldn’t be able to effectively incorporate feedback from the sponsors.

We also considered an Agile system but realized the quick prototyping nature is not well suited for our project. A barebones prototype of our project would be a glorified “hello world” program that processes hardcoded data which didn’t seem ideal. This was our rationale that led us to the Iterative lifecycle model and we think it will be very effective for us when creating our project.

Communication between sponsors will be through email and zoom meetings. Team members will communicate with one another through discord / email / in person meetings. Team members will communicate with UTD professors via email.

# Risk Analysis

* Online data access being taken offline on the API provider’s end
  + **Action**: If data becomes unavailable by the provider, backups to the data will be created to allow access to the
  + **Mitigation**: Try to obtain and keep an offline copy of the live data as much as possible, within our budget, scope, and resources.
  + **Retirement:** Data fully backed up; ready to access and use.
* Team member is out sick
  + **Action:** If a team member is out of commission, other team members will be notified as soon as possible to allow leeway to make urgent and critical decisions
  + **Mitigation:** Fallback team members assigned to tasks, in the event that one of the team members are unavailable or unable to fulfill tasks, whether it is emergencies or events out of any of the members’ control.
* Github down for maintenance
  + **Action**: If Github goes down for maintenance team members will be notified to ensure that there is not a disruption in version control.
  + **Mitigation**: Multiple ways to reduce risk is to mirror the git repo to other platforms or a self-hosted git repository that is limited to use and viewing by team members, and possibly sponsors upon request and approval by the team.
* Lack of technical knowledge
  + **Action:** If there is a certain skill set that the team is lacking that is required for completion of the project. Then a task will be delegated out to a team member for research.
  + **Mitigation:** Planning ahead of time to foresee a possible area where there would be a lack of expertise will help with preventing a slowdown.
* Unable to display results
  + **Action:** If the team is unable to display results to sponsors in an appealing way, steps will be taken to ensure a presentation presenting the predictions concluded by the application
  + **Mitigation:** Ensure enough time to tackle the time intensive task of creating proper visualization
* Unable to create correlations
  + **Action:** If the application is unable to produce the appropriate results, examination of the data / algorithms being used will be applied
  + **Mitigation:** Proper prep work and question asking in terms of what the data can answer will be established to prevent possible issues

# 

# Possible Opportunities

The satisfaction of having learned about and created a tool that could help with decisions related pertaining to food insecurity and crop yields. Farmers having a more positive outlook on AI and machine learning technology to help plan and chart out their future. This could also help pave the way for planning new farmland dependent upon prediction.

### 

# Software and Hardware Resource Requirements

We will be using Python, It is widely used in data science and machine learning. Python has extensive libraries and frameworks such as NumPy, and TensorFlow that are crucial for data analysis and modeling. Version Control will be Git and it is essential for tracking changes in your codebase, collaborating with others, and maintaining a version history. Geospatial Analysis (Optional), If the analysis involves geospatial data, these libraries will be helpful for reading, writing, and processing spatial data.For the Hardware Requirements, it will be a Processor (CPU), A multi-core processor (Intel Core i5 or i7) for faster data processing. Random Access Memory At least 16 GB of RAM is recommended, especially for handling large datasets. Also, sufficient disk space to store datasets, code, and model files.

*Rationale*: Python: Chosen for its readability, extensive libraries, and popularity in the data science community. Database Management: PostGreSQL. Geospatial Analysis (Optional): Useful if the analysis involves geographical aspects of crop yield. Version Control (Git): Facilitates collaboration, code tracking, and easy rollback to previous versions. GPU (Optional): Accelerates model training, especially for deep learning, significantly reducing processing time. RAM and Storage: Essential for handling large datasets and ensuring smooth execution of resource-intensive tasks.

The languages the team learned to follow through with the project involved Javascript, CSS, HTML, React, and PostgreSQL. Also learning how to utilize the libraries and have the, communicate with one another.

# 

# Deliverable Schedule

| **Date/Time** | **Participants** | **Activity** | **Rationale** | **Dependencies** |
| --- | --- | --- | --- | --- |
| February 3rd, 2024 | *Group Team 8* | Git Repository Overview, Setup, and Familiarization | Make sure the whole team is on the same page on how code revisions will be processed and published | Git, GitHub, in-person meeting, computers |
| February 10th, 2024 | *Group Team 8* | Select data sets | The data sets are what make this project work, so selecting appropriate data is crucial | Related data repositories, backup servers, |
| February 17th, 2024 | *Group Team 8* | *First iterative stage of the development process* | This is the first iteration of the project, so starting from scratch with all the material and software tools at our disposal. | In-person meeting, sponsors feedback noted, access to data and code, computers |
| February 24th, 2024 | *Group Team 8* | *Second iterative stage of the development process* | After getting feedback from the first iteration demonstrated to the sponsors, changes (additions, removals, etc) will be made. | In-person meeting, sponsors feedback noted, access to data and code, computers |
| March 2nd, 2024 | *Group Team 8* | *Third iterative stage of the development process* | Again, further development made based on what was accomplish, what needs to be added, and what changes must be made based on feedback on the previous meeting with sponsors | In-person meeting, sponsors feedback noted, access to data and code, computers |
| March 8th, 2024 | *Group Team 8* | *Fourth iterative stage of the development process* | Continue with the iterative development process and prepare for the in person meeting based on criteria set by sponsors the coming week | In-person meeting, sponsors feedback noted, access to data and code, computers |
| March 22nd, 2024 | *Sponsors + Group Team 8* | In-person meeting at the Richardson campus of Raytheon | Make sure to bring everything agreed upon by sponsors and make notes and use opportunities to meet face-to-face with deep-in-the-thick-of-things developers | In-person meetup, notepads, transportation, sponsors being present (for security reasons) |
| March 29th, 2024 | *Group Team 8* | *Fifth iterative stage of the development process* | Continuing the iterative process | In-person meeting, sponsors feedback noted, access to data and code, computers |
| April 5th, 2024 | *Group Team 8* | *Sixth iterative stage of the development process* | Continuing the iterative process | In-person meeting, sponsors feedback noted, access to data and code, computers |
| April 12th, 2024 | *Group Team 8* | *Seventh iterative stage of the development process* | Continuing the iterative process | In-person meeting, sponsors feedback noted, access to data and code, computers |
| April 19th, 2024 | *Group Team 8* | *Eighth iterative stage of the development process* | Continuing the iterative process | In-person meeting, sponsors feedback noted, access to data and code, computers |
| April 26th, 2024 | *Group Team 8* | *Ninth stage of the development process* | Continuing the iterative process | In-person meeting, sponsors feedback noted, access to data and code, computers |
| May 3rd, 2024 | *Group Team 8* | *Final stage of the development process* | Likely polishing, in terms of minor bug fixes, polishing the UI/UX, and addressing any final feedback given by the sponsors of the project | In-person meeting, sponsors feedback noted, access to data and code, computers |

# 

# Monitor / Control Mechanisms

**Monitoring Mechanisms**

* Weekly Progress Report
  + *Description*: Weekly report that will consist of all work completed in the current week. The report will also include what the plan for the following week will be.
  + *Rationale*: This will give a clear picture as to what work has been completed, and give an established goal as to what needs to be accomplished in the coming week.
* Attendance Report
  + *Description*: Weekly Attendance Report to be sent to instructor and TA’s. This will keep track of each team member's participation with our sponsors and within our group.
  + *Rationale*: Required reporting for senior design class. This will help keep in compliance with the requirements needed by the class.

**Control Mechanisms**

* Version Control

*Description*: The project will be version controlled through the use of Git.

*Rationale*: To ensure consistency and protection of the code base. As the project continues to grow, this will help with ensuring protection of the overall production cycle of the project, reducing the risk of needing to start over.

## 

# Professional Standards

Ensure compliance with data protection laws and regulations. Implement secure storage and processing practices to safeguard sensitive information. Collect and use data ethically, respecting the rights and privacy of individuals. Communicate the purpose of data collection and obtain appropriate consent. Document your methods, assumptions, and limitations. Transparency is essential for others to understand and trust your analysis. Structure your code and analysis to allow others to reproduce your results. This promotes transparency and helps in the validation of findings. Be aware of potential biases in your data and algorithms. Strive to address and mitigate biases to ensure fair and equitable predictions. Maintain comprehensive documentation for your project, including data sources, preprocessing steps, model architecture, and evaluation results. Encourage peer review of your work. Collaborate with domain experts and stakeholders to ensure your analysis aligns with domain knowledge and addresses relevant issues. Stay updated on the latest data science, machine learning, and agricultural advancements. Continual learning ensures that your analysis incorporates the most recent techniques and knowledge. Consider the environmental impact of your analysis. Optimize code and algorithms to minimize resource consumption, especially when deploying models in resource-constrained environments.

## 

# Impact of the project on individuals and organizations

In today's rapidly changing world where climate change poses a looming threat to our agricultural systems and food security, our project aims to develop a robust crop yield prediction model. This could also personally affect farmers with the personal crops or big corporations with financial preparations as they prepare for the coming year based upon predicted yield.

## 

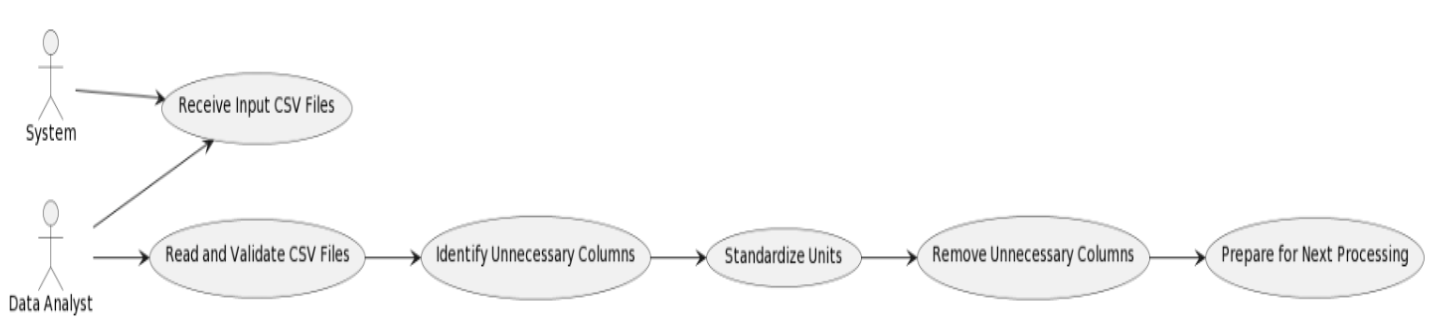
# Requirement Specifications

### Stakeholders

Anyone doing the farming, Researchers, Government agencies, Environmentalists, The Developers

### Use Cases

## Use Case: Data PreProcessing



Participating Actors: System, Data Analyst

Entry Condition(s):

* The system has received valid input data in the form of CSV files.
* Necessary libraries and dependencies for data preprocessing are available.
* Normal Flow of Events:
* The system reads and validates the input CSV files to ensure they meet the required format and structure.
* Unnecessary columns in the input data are identified and marked for removal.
* Units of the data are standardized to a common format for consistent analysis.
* The system removes the marked unnecessary columns from the dataset.
* The preprocessed data is prepared for the next stage of processing.

Exit Condition(s):

A preprocessed dataset with standardized units and removed unnecessary columns is available for further processing.

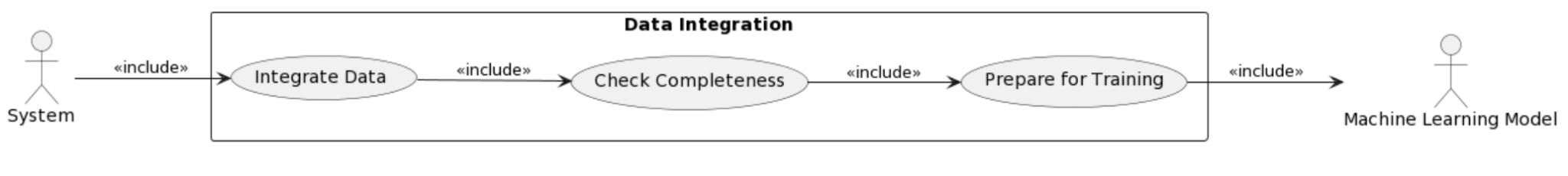
Exceptions (Alternate Flow of Events):

* If the input data is not in the expected format, an error message is logged, and the process is halted.
* If the data contains missing values that cannot be handled, the system logs an error and halts the process.

Special Requirements:

The data preprocessing steps should be performed efficiently to minimize processing time.

## Use Case: Data Integration\*



Participating Actors: System, Machine Learning Model

Entry Condition(s):

Preprocessed historical corn crop yield data and pesticide usage data are available.

Normal Flow of Events:

* The system integrates historical corn crop yield data with pesticide usage data based on common identifiers (e.g., county, year).
* The integrated dataset is checked for completeness, and missing values are handled appropriately.
* The integrated dataset is prepared for training the machine learning model.

Exit Condition(s):

A complete and integrated dataset is ready for training the machine learning model.

Exceptions (Alternate Flow of Events):

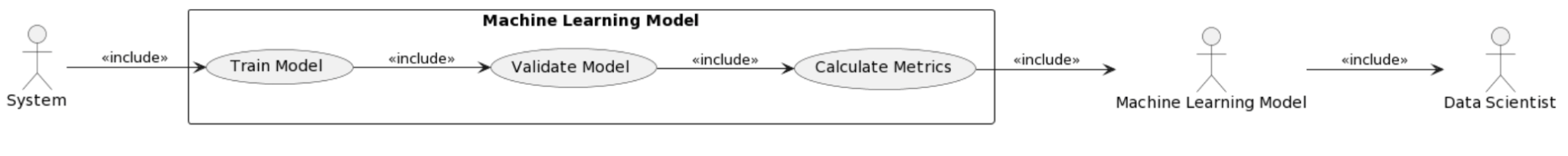
If there are inconsistencies or missing values in the integrated dataset, the system logs an error and halts the process.

Special Requirements:

The integrated dataset should maintain data integrity and consistency.

**\*This has been removed as we are no longer using pesticide data. We determined there wasnt enough pesticide data to be able to properly assume what its future impact would be. This was approved by the sponsor.**

## Use Case: Machine Learning Model



Participating Actors: System, Machine Learning Model, Data Scientist

Entry Condition(s):

A complete and integrated dataset is available for training and validation.

Normal Flow of Events:

* The machine learning model is trained on the historical data, considering both crop yield and pesticide usage.
* The model is validated on a separate dataset to ensure accuracy.
* Performance metrics are calculated to assess the model's effectiveness in predicting corn crop yield.

Exit Condition(s):

A trained and validated machine learning model is ready for prediction.

Exceptions (Alternate Flow of Events):

* If the model fails to converge during training, an error message is logged, and the process is halted.
* If the validation accuracy falls below a specified threshold, the model may need to be retrained.

Special Requirements:

The machine learning model should be optimized for performance and accuracy.

## Use Case: Output Formatting

Participating Actors: System, Output Formatter

Entry Condition(s):

The machine learning model has successfully predicted corn crop yields.

Normal Flow of Events:

* The predicted crop yield data is formatted according to the specified output requirements.
* Units are converted if necessary.
* The formatted output data is prepared for export.

Exit Condition(s):

Formatted output data ready for export is available.

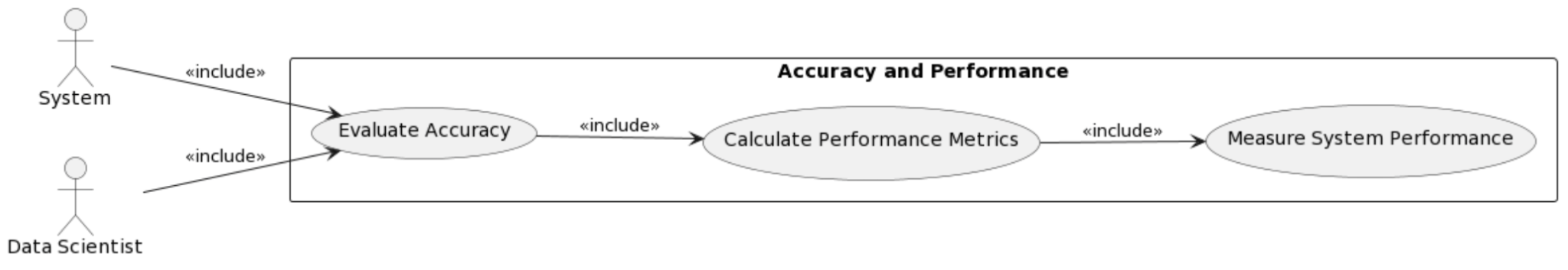
Exceptions (Alternate Flow of Events):

If there are issues with the formatting process, an error message is logged, and the process is halted.

Special Requirements:

The output formatting should be consistent and accurate.

## Use Case: Accuracy and Performance



Participating Actors: System, Data Scientist

Entry Condition(s):

The machine learning model has been trained on historical data, and the system is prepared to assess its accuracy and performance.

Normal Flow of Events:

* The system evaluates the accuracy of the trained machine learning model using a validation dataset.
* Performance metrics, such as precision, recall, and F1 score, are calculated to assess the model's effectiveness in predicting corn crop yield.
* The computational efficiency and response time of the system are measured to ensure optimal performance.

Exit Condition(s):

Accuracy and performance metrics are obtained and available for analysis.

Exceptions (Alternate Flow of Events):

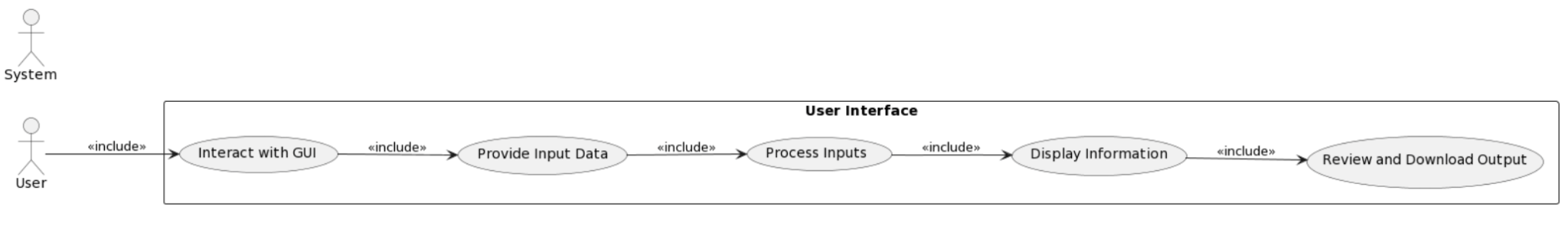
* If the model's accuracy falls below a specified threshold, the system may trigger a retraining process and log a warning message.
* If the performance metrics do not meet the defined criteria, the system may prompt further optimization.

Special Requirements:

* The system should provide detailed accuracy metrics to assess the reliability of the predictions.
* Performance measurements should be collected under various scenarios to ensure consistency and reliability.
* Continuous monitoring and periodic evaluation of the model's accuracy and performance should be implemented to address potential drift over time.

## 

## Use Case: User Interface



Participating Actors: User, System

Entry Condition(s):

* The system is initialized and ready to receive user inputs.
* The necessary libraries and dependencies for the user interface are available.

Normal Flow of Events:

* The user interacts with the system through the graphical user interface.
* The user provides input data, such as selecting input files or specifying parameters for the prediction model.
* The system processes user inputs and executes the necessary functionalities, such as data preprocessing, integration, and model training.
* The system displays relevant information to the user, such as progress updates or the generated predictions.

Exit Condition(s):

The user has successfully interacted with the system, and the desired functionalities have been executed.

Exceptions (Alternate Flow of Events):

* If the user provides invalid input, the system displays an error message and prompts the user to correct the input.
* If an unexpected error occurs during processing, an error message is displayed, and the user is informed about the issue.

Special Requirements:

The user interface should be intuitive and user-friendly.

User inputs should be validated to ensure they meet the required format and criteria.

The system should provide clear instructions and feedback to the user during the interaction.

# 

# Rationale For Use Case Model

* Data PreProcessing
  + Standardizing units and removing unnecessary columns contribute to a more consistent and streamlined dataset, simplifying subsequent analysis and modeling.
* Data Integration
  + The use case addresses the need for a unified dataset that incorporates relevant information from different sources, providing a holistic view for the machine learning model.
* Machine Learning Model
  + The primary objective of the system is to predict corn crop yield accurately. This use case involves training a machine learning model on historical data and validating its performance.
* Output Formatting
  + Once predictions are made, presenting the results in a consistent and understandable format is essential for practical use. This use case is relevant when it comes to the end user experience. This is vital and necessary.
* Accuracy and Performance
  + Being able to make sure the machine learning model is accurate is necessary in order to make sure it is reliable
* User Interface
  + Providing a user-friendly interface enables stakeholders, who may not be familiar with technical details, to input parameters, monitor progress, and access results without delving into the intricacies of the underlying processes.

# Non-Functional Requirements

| **Non-Functional Requirements No.** | **Category** | **Attributes** | **Non-Functional Requirements Description** | **Priority** |
| --- | --- | --- | --- | --- |
| NFR 1 | Interoperability | Data Preprocessing, Data Integration, Machine Learning | The system shall make use of data between different modules in the software system is critical to make use of all the information being inputted and outputted. | **Critical** |
| NFR 2 | Usability | User Interface, Output Formatting, Accuracy and Performance` | The software shall make sure the project is easily used by both non-technical and technical users, with clear visualization of data. | **High** |
| NFR 3 | Scalability | Accuracy and Performance | The system will account for demand for this model and its usage will demand scalability, when more users are using the software system. | Low-Moderate` |

# 

# 

# Future Requirements

| **Functional or Non-Functional** | **Category *(NFR only)*** | **Description** | **Priority** |
| --- | --- | --- | --- |
| FR 1 | N/A | The system shall convert the raw data set into a csv file. | Moderate |
| FR 2 | N/A | The system shall import the preprocessed csv file into the DBMS. | Moderate |
| FR 3 | N/A | The system shall remove any and all irrelevant database columns to the function of the app. | Moderate |
| FR 4 | N/A | The system will expand to utilizing a neural network to make more accurate / deep predictions. | Moderate |
| FR 5 | N/A | The system will combine both datasets from corn and pesticide use into one file. | Low |

*N/A = Not Applicable*

# 

# Functional Requirements

| **Functional Requirements No.** | **Functional Requirements Description** |
| --- | --- |
| FR1 | The system shall remove any data that has an non-numeric entry in the “VALUE” column |
| FR 2 | The system shall remove all non-county data |
| FR 3 | The system shall remove any data that is not reported in bushels |
| FR 4 | The system shall only consider data after the year 2004 |
| FR 5 | The system shall filter data to be only data about corn |
| FR 6 | The system shall export the cleaned data from the DBMS into a csv file |
| (Removed)FR 7\* | The system shall remove any pesticides not used for corn farming from the pesticide dataset |
| FR 8 | The system shall convert exported data from DBMS for the machine learning algorithm to use |
| (corrected)FR 9\* | The system shall have a GUI that accepts a US county (this portion removed)and pesticide usage in KGs |
| (corrected)FR 10\* | The system shall take user input for US county (this portion removedand pesticide usage in kgs |
| FR 11 | The system shall input the user input into the model for prediction |
| FR 12 | The system shall receive the output from the model based on the user input |
| FR 13 | The system’s GUI shall display the output of the model in bushels of the corn expected to be produced |
| FR 14 | The system shall use machine learning algorithms to make predictions based on the data |
| (corrected)FR 15\* | The system shall utilize cleaned datasets from the corn data to use in the model |
| FR 16 | The system shall utilize the Python libraries to provide accuracy and precision results |

*\*We no longer have pesticide data so we corrected/removed any FRs that had the pesticide data in them. We determined there wasn't enough pesticide data to be able to properly assume what its future impact would be. This was approved by the sponsor.*

# Traceability Table

| **Use Case** | **Functional Requirements** | **Non-Functional Requirements** | **Description** |
| --- | --- | --- | --- |
| Data Preprocessing | FR 1 - FR 7 | NFR 1 | Processing the data set so it’s ready to be inputted into the model. |
| Data Integration | FR 15 | NFR 1 | The two data sets, corn crop yield and pesticide, will be integrated to make one file for the model to be trained on |
| Machine Learning | FR 14 | NFR 1 | The model is trained on the data we have |
| Output Formatting | FR 12 / 13 | NFR 2 | The data is taken from the model and is formatted for additional use |
| Accuracy and Performance | FR 16 | NFR 2, NFR 3 | Make sure the model is able to accurately predict the data |
| User Interface | FR 9 - FR 13 | NFR 2 | The user will get the output based on the user input |

# 

# Client-Server Architecture

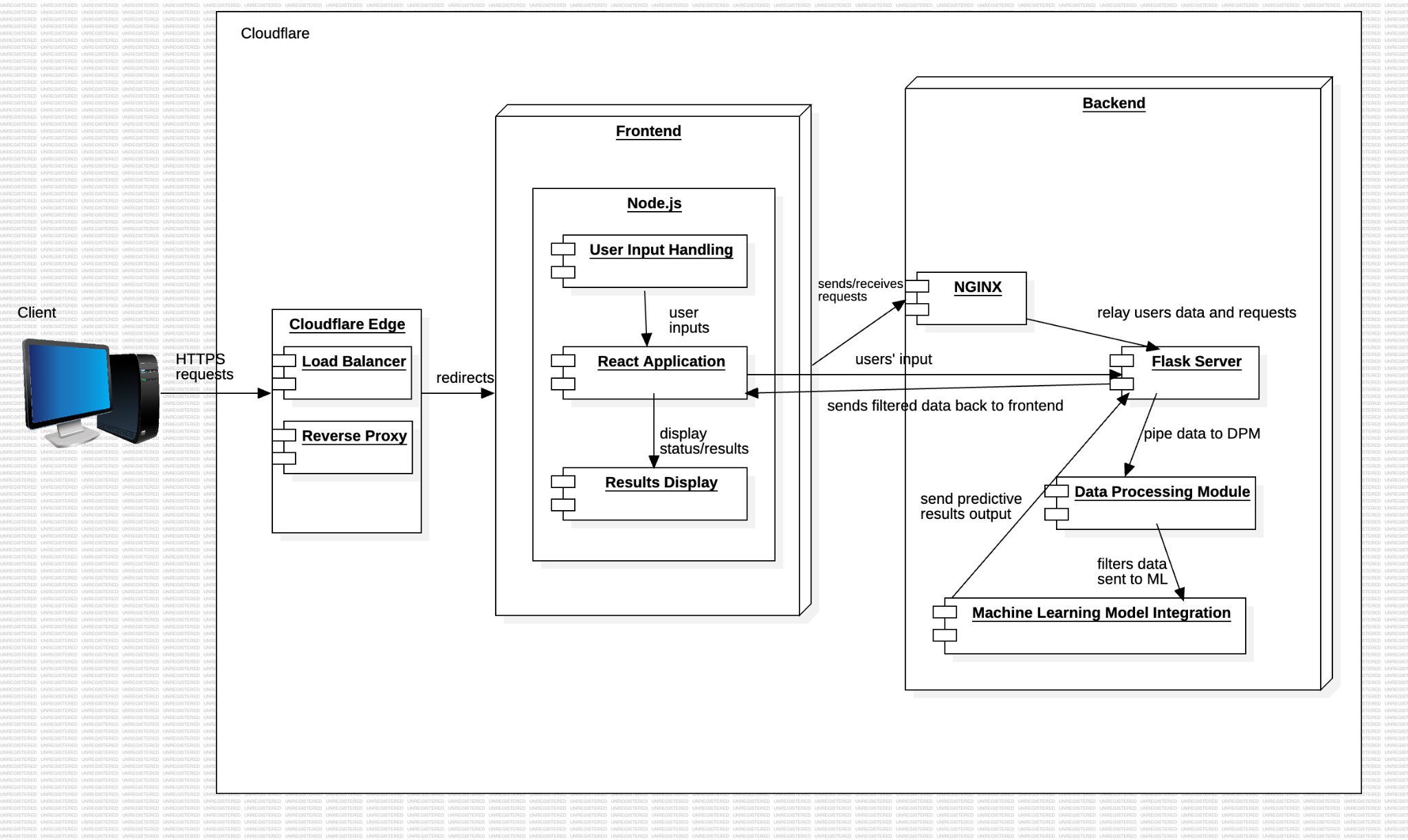
The client-server architecture style works with our model since this will allow us to divide the crop yield data model into various parts and we can manage the different parts with different members of the team.

On the backend we are utilizing our primary server.py file that handles ingesting the data that is received from the front end. It then calls upon another datafilterer.py file on the backend to slim down the overall dataset to solely that specific selected state/county. Once the data is slimmed it is passed back to our server.py file to run a linear regression machine learning algorithm to create the future crop yield prediction.

On the front end, we have a React static web page that takes in a state/county inputted by the user. Once the data has been processed by the backend and sent back to our front end user facing component, it utilizes the chart.js library to generate plotting for displaying data. It also provides a simple textual description of the value. The React page is also customized to allow regeneration of different inputs with the need to refresh the web page through tricking the browser cache.

# 

# Architecture Model



# 

# Technology, Software, Hardware Used

The software used for this project includes use of Python and Javascript, in conjunction with various frameworks and libraries. Flask and React.js are chosen for the front-end GUI portion of the software, in order to make it easy to use for users. Python was used to also help pre-process the data used for the crop yields, and to keep the project focused and less bloated. Database management system of choice is PostGreSQL, due to its community support, high performance under even stressful and high loads, and wide industry adoption; however, of note, the database management system (DBMS) for this project, PostgreSQL, will be used as a repository for data records to train the model that will interface with the client.

For the hardware, using a decommissioned workstation as a compute server, this is used to help train the neural network models. The compute GPU, an NVIDIA Tesla P40, is for training the model that we have setup, to predict the future crop yield based on a wide variety of factors and previous datasets that the model is fed and trained under. ECC RAM is used to make sure that any data that goes through the RAM will not be susceptible to data corruption or errors, in critical situations. As a risk mitigation, a CyberPower 1500VA/900W UPS battery backup will be used in the event of a brownout, or worst, a blackout to make sure critical information going in-and-out of the server will not be corrupted, so the system will be able to safely shut down in a critical situation, mentioned previously.

**Hardware**

* + Lenovo ThinkStation P500 Workstation
    - GPUs:
      * NVIDIA Tesla P40 (24GB GDDR5 VRAM, *Compute*)
      * AMD Radeon GPU R9 255 OEM (*Display/Primary*)
    - RAM: 64GB (RDIMM, ECC, DDR4)
    - CPU: Intel Xeon E5-2699 v3 (18-core, 36-threads, 2.3 GHz)
  + CyberPower UPS 1500VA/900W Battery Backup

**Software**

* + Operating System
    - Ubuntu Linux 64-bit, 22.04
  + Programming Languages
    - Python 3
    - JavaScript
  + Programming Libraries / Frameworks
    - React.js (JavaScript library)
    - Flask (Python framework)
  + Database Management System
    - PostgresSQL 16.x
  + Programs
    - JetBrains’ Data Grip
    - Git
    - VSCode

**Communication between the Application Server and the Database Server**

In this current server infrastructure, the application server communicates with the dataserver by communicating to the local IP address at a different port (ex. 127.0.0.1:8006, localhost IP at a different listening port) to retrieve specific data found within the database, matching certain conditions of the user’s input. For purposes of recordkeeping many datasets, in order for the model training to be set, the PostgreSQL database will only for data record keeping to feed, store, and train the crop yield prediction model. Python libraries and programs will also be utilizing the *virtual environment* and *psycop2* library (C language-based PostgreSQL database adapter for the Python programming language) to communicate with the DBMS (PostgreSQL), sending various PostgreSQL commands and interfacing with the database(s), when necessary.

# 

# Rationale for your architectural style and model

The client-server architecture style allows us to compartmentalize the model, since each part needs to work well for the entire project to work, this style works well with our project, as it allows us to give each component the attention it needs. This style allows the project to be divided into easy-to-work-with parts, allowing our team members to divide the work and manage the whole project easily. Distinguishing between the back end and the front end will allow us to focus on a user-friendly interface. This style works well with the plan we had for our project.

# 

# Traceability From Requirements To Architecture

| **Layer** | **FR** | **Use Cases** | **Description** |
| --- | --- | --- | --- |
| Data Processing | FR1 - FR7 | Data Pre-Processing | Architecture must include components for importing, validating, cleansing, and standardizing data, as well as exporting the cleaned data for further use. |
| \*(removed) Data Integration | FR15 | Data Integration | The architecture should facilitate the merging of different data sources (crop yield and pesticide usage) into a unified dataset for analysis. |
| Model Training | FR14, FR16 | Machine Learning Model | Components for training machine learning models on integrated data and evaluating their performance are required. |
| Presentation/User Interface | FR9 - FR13 | User Interface | The architecture needs to include a user interface that allows input of parameters and displays predictions in an understandable format. |
| Data Output and Analysis | FR12, FR13, FR16 | Output Formatting, Accuracy and Performance | Ensures the system can format the output correctly and measures performance accuracy and precision |

**\*This has been removed as we are no longer using pesticide data. We determined there wasnt enough pesticide data to be able to properly assume what its future impact would be. This was approved by the sponsor.**

# Design

### GUI Design

# 

### GUI Description

**State/County Code:** The code submissions are set up with a simple React form submission. The reasoning for this GUI design is to have an easy and simplistic approach to make for an easy user experience. (Subject to be modified to state name and county name)

**Output: Numerical/Graphical Display:** Prediction results are displayed to the user in a simple statement of the numerical value of the yield. The user is also displayed all previous years crop yield data for that county.

## 

## *Class Diagram*

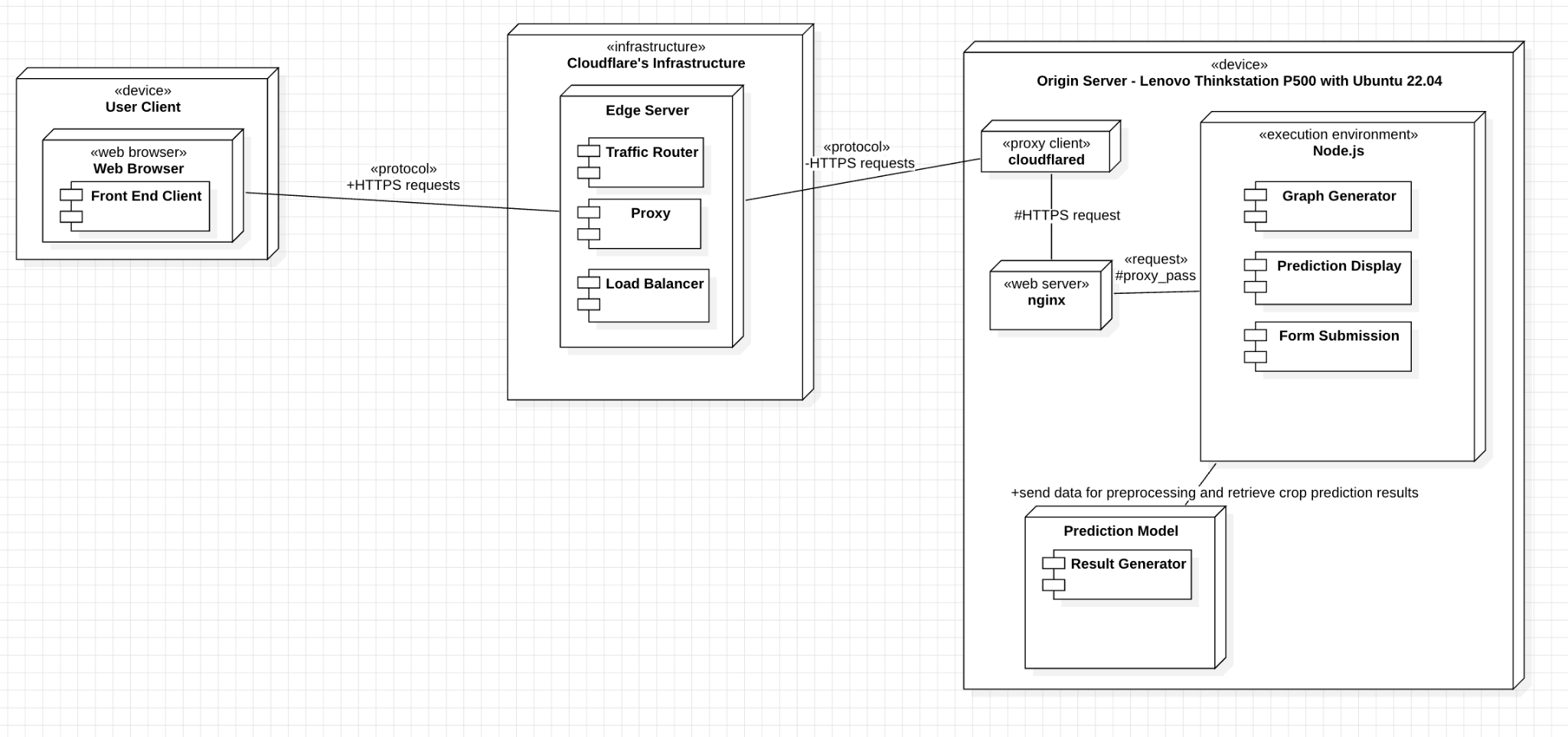
### *Class Diagram Description*

**Client**: Client side contains a form submission. The submitted data consists of two values: a county code and state county, which are passed back as integer values. It also displays the prediction for the data.

**Server**: consists of handling incoming state and county data. Once data is received it holds the predict function that runs the main task of slimming the data, running ML algorithm, and passing back data

**DataFilter**: Class used on the backend that takes in the state and county code from the front end and slims the main dataset down to just that state and county. Once slimmed it passes the cleaned dataset back to the Server class to run the prediction.

## *Deployment Diagram*



### *Deployment Diagram Description*

**User Client:** For the deployment diagram, the user client(s) will display and access the web application through their ***web browser***. From there, the user client using a web browser will communicate to Cloudflare’s web infrastructure. It will communicate on both ends with HTTPS requests protocol; since many modern internet browsers default to HTTPS traffic as a security precaution, by default.

**Cloudflare’s Infrastructure:** Then, the browser sends out HTTPS requests in order to connect to their edge server in an attempt to communicate with the origin server through a proxy service, which helps to improve security and obfuscate the origin server’s IP address in the event of attempting to find where the origin server is located and mitigate DDoS (*distributed* denial of service) attempts. It will communicate between the***edge server*** and the origin server’s ***cloudflared*** proxy client, by the HTTPS requests protocol.

**Origin Server:** After connecting to the origin server and communicating via the default HTTP/HTTPS ports, the proxy client ***cloudflared*** will communicate with ***nginx*** to proxy traffic into the execution environment using ***Node.js***. It will listen for the customer to generate crop yield predictions based on information entered by the user (if incorrect data is inputted, it will take care of that), and then communicate with the prediction model stored on the origin server. Many of the tasks will be processed in and out of the server to be piped through Cloudflare’s infrastructure, back to the user’s internet browser, using the HTTPS protocol, primarily.

# 

# Dynamic Models

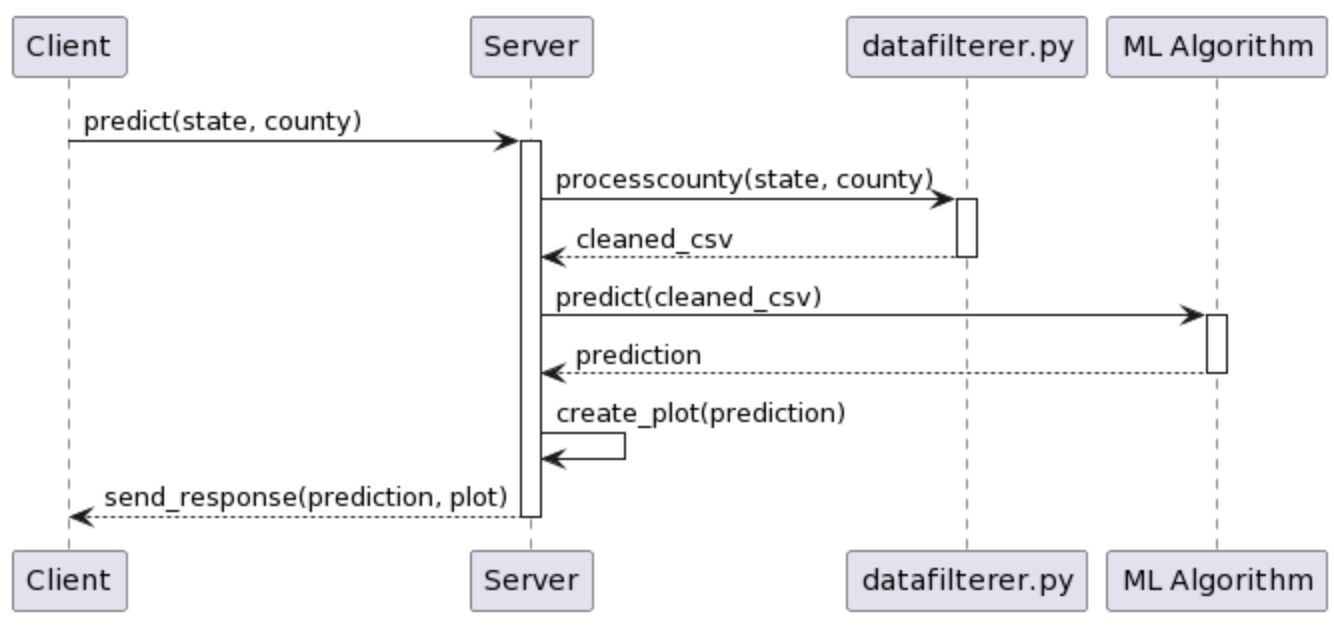
### *Sequence Diagram*

### *Sequence Diagram Description*

**User State/County Entry Sequence**: User submits the state and county information into the form on the front end react component. The data is then sent to the backend to be predicted within the predict function. Once the prediction results are created the results are then returned and received by the front end. The front end then displays the results to the user on the front web page.

## 

## *Sequence Diagram predict() Diagram*



### *Sequence Diagram predict() Description*

**predict() Sequence**: When the data enters the predict function it is first run through the data filterer file to be processed down into a cleaned csv file with only that state and county code. The cleaned data is then fed into the linear regression model to create a prediction for the future year’s yield. It finally also creates a graph for the prediction to also send back to the front end.

# 

# Rationale for Detailed Design Model

***Server-side Design***

On the ***server*** side, this design model prioritizes operational securities, uptime, and mitigating potential cyberattacks from bad actors.

Much of the online space is filled with DDoS attacks that have a wide variety of different motives, not exclusive to a few.

Bad actors can show up at a very inconvenient time and place, for both users and developers; it is best to have failsafe plans in case, in order to mitigate potential cyberattacks.

Hide the origin server’s IP address from being publicly exposed and potentially putting the host of the server in danger for bad actors, in regards to *operational security*.

Keep dependencies to a minimum, especially with the choice to use a pre-trained model to process and output data.

* + - * Thanks to this approach, there is no need to use or host a database management system (DBMS) on the server.
        + Thus, mitigating a major point of vulnerability for bad actors, given how precious and valuable data records are, as a commodity.

***Client-side Design***

On the ***client*** side, make it very easy and simple for a wide variety of users.

The target demographic of users potentially use devices that lack computational resources, so it is very important to offload the burden to the server, rather than to the user’s local device.

The GUI, on the client side, is designed the way it is in order to provide a straightforward and seamless user experience that prioritizes performance and simplicity.

# 

# Traceability From Requirements To Design Model

| **Client/Server** | **FR** | **Mapping** | **Description** |
| --- | --- | --- | --- |
| Server-side | FR1 - FR7 | Data Filter file | Architecture must include components for importing, validating, cleansing, and standardizing data, as well as exporting the cleaned data for further use. |
| Server-side | FR14, FR16 | Server Class, predict() function | Components for training machine learning models on integrated data and evaluating their performance are required. |
| Client-side | FR9 - FR13 | Client class, React component | The architecture needs to include a user interface that allows input of parameters and displays predictions in an understandable format. |
| Client-side | FR12, FR13, FR16 | Client class, React component | Ensures the system can format the output correctly and measures performance accuracy and precision |

# 

# Requirements/Specifications-Based System Level Test Cases

| **FRs** | **Test Case** | **How Test Case(s) cover the FR** |
| --- | --- | --- |
| FR 1 | TC 3  TC 5  TC 6 | This functional requirement is part of the pre-processing data portion for this we need the normal system coverage tests (5 and 6) as well as the pre-process functional test (3). |
| FR 2 | TC 3  TC 5  TC 6 | This functional requirement is part of the pre processing data portion for this we need the normal system coverage tests (5 and 6) as well as the pre process functional test (3) |
| FR 3 | TC 3  TC 5  TC 6 | This functional requirement is part of the pre processing data portion for this we need the normal system coverage tests (5 and 6) as well as the pre process functional test (3) |
| FR 4 | TC 3  TC 5  TC 6 | This functional requirement is part of the pre processing data portion for this we need the normal system coverage tests (5 and 6) as well as the pre process functional test (3) |
| FR 5 | TC 3  TC 5  TC 6 | This functional requirement is part of the pre processing data portion for this we need the normal system coverage tests (5 and 6) as well as the pre process functional test (3) |
| FR 6 | TC 3  TC 5  TC 6 | This functional requirement is part of the pre processing data portion for this we need the normal system coverage tests (5 and 6) as well as the pre process functional test (3) |
| FR 8 | TC 3  TC 5  TC 6 | This functional requirement is part of the pre processing data portion for this we need the normal system coverage tests (5 and 6) as well as the pre process functional test (3) |
| FR 9 | TC 1  TC 2  TC 4  TC 5  TC 6  TC 7 | This FR is part of the User interface portion,(as well as the output) so it needs tests for functional correctness (1 and 4) completeness(2) performance capacity(7) and then the system coverage tests (5 and 6) |
| FR 10 | TC 1  TC 2  TC 4  TC 5  TC 6  TC 7 | This FR is part of the User interface portion,(as well as the output) so it needs tests for functional correctness (1 and 4) completeness(2) performance capacity(7) and then the system coverage tests (5 and 6) |
| FR 11 | TC 1  TC 2  TC 4  TC 5  TC 6  TC 7 | This FR is part of the User interface portion,(as well as the output) so it needs tests for functional correctness (1 and 4) completeness(2) performance capacity(7) and then the system coverage tests (5 and 6) |
| FR 12 | TC 1  TC 2  TC 4  TC 5  TC 6  TC 7 | This FR is part of the User interface portion,(as well as the output) so it needs tests for functional correctness (1 and 4) completeness(2) performance capacity(7) and then the system coverage tests (5 and 6) |
| FR 13 | TC 1  TC 2  TC 4  TC 5  TC 6  TC 7 | This FR is part of the User interface portion,(as well as the output) so it needs tests for functional correctness (1 and 4) completeness(2) performance capacity(7) and then the system coverage tests (5 and 6) |
| FR 14 | TC 1  TC 2  TC 4  TC 5  TC 6 | This FR is about the model making predictions based on Machine Learning algorithms. For this we need to make sure the data is accurate (Functional correctness) (1 and 4) also requires functional completeness (2). and then the system coverage tests (5 and 6) |
| FR 15 | TC 1  TC 2  TC 4  TC 5  TC 6 | This FR is about the system using cleaned data sets.. For this we need to make sure the data is accurate (Functional correctness) (1 and 4) also requires functional completeness (2). and then the system coverage tests (5 and 6) |
| FR 16 | TC 1  TC 2  TC 4  TC 5  TC 6 | This FR is to make sure the data is accurate so we use functional correctness tests (1 and 4) functional completeness (2) and the system coverage tests (5 and 6) |

# 

# Techniques for Test Generation

| **Test Case** | **Technique** | **Description** | **Rationale** | **Quality Characteristic\***  **[1]** | **Black Box/White Box\*** |
| --- | --- | --- | --- | --- | --- |
| TC 1 | Functional Test | *Input Validation*: Provide system with invalid state and county names | Ensure system can handle wrong input gracefully | Functional Correctness | White Box |
| TC 2 | Functional Test | *Boundary Testing*: provide extreme values for input | Ensure system behaves correctly within limits | Functional completeness | White Box |
| TC 3 | Functional Test | *Input Validation*: Give a test csv file with sample data so we control the expected output | Verifies the data preprocessing produces the expected output. | Functional Correctness | White Box |
| TC 4 | Accuracy Test | Prediction Accuracy | Verify Accuracy of prediction against known data | Functional Correctness | White Box |
| TC 5 | Structural Test | *Code Coverage*: Exercise all branches and paths in backend code | Ensure comprehensive testing of backend logic | Functional completeness | Black Box and White Box  (Full System test but goes through each individual component) |
| TC 6 | System Integration Test | Tests end to end flow of the entire system | Goes through each component thoroughly to make sure the system performs as expected | Functional completeness | Black Box and White Box  (Full System test but goes through each individual component) |
| TC 7 | Performance Test | *Stress Testing*: Simulate high concurrent user requests | Assess system performance under heavy load | Capacity | White Box |

# 

\*The assessment of the goodness of the test is by the quality characteristic in the table.

# 

# Traceability of Test Cases to Use Cases

| **Test Case** | **FRs** | **Use Case** | **Use Case Description** |
| --- | --- | --- | --- |
| TC 3  TC 5  TC 6 | FR 1 - FR 7 | Data Preprocessing | Processing the data set so it’s ready to be inputted into the model. |
| TC 1  TC 2  TC 4  TC 5  TC 6 | FR 14 / 15 | Machine Learning | The model is trained on the data we have |
| TC 1  TC 4  TC 5  TC 6 | FR 12 / 13 | Output Formatting | The data is taken from the model and is formatted for additional use |
| TC 1  TC 2  TC 4  TC 5  TC 6 | FR 16 | Accuracy and Performance | Make sure the model is able to accurately predict the data |
| TC 1  TC 2  TC 4  TC 5  TC 6  TC 7 | FR 9 - FR 13 | User Interface | The user will get the output based on the user input |

# 

# Evidence of Report Under Configuration Management

1. **Name of the CM tool used by your team**: Google Docs

1. **Version number of each document after it is checked in**: Google Docs does not have version numbers, but has versions based on timestamps for each session. We can view these in the version history in *File > Version History*.

1. **Version number of each document before it is checked out**: Google Docs does not have check out, or check in, the team works on the latest time stamped version of the document.

1. **Difference between two consecutive versions**: Users can view the revision history through Google docs to see the different versions of the document.

1. **Review of each change**:
   1. After the document is finalized, we have the team double check it to make sure all the input is correct.

1. **Other information that helps the understanding of each change**:
   1. Different team members are assigned different tasks and put this information in the table below.
   2. Every team member is added with their gmail accounts so we know who made each change.

# 

# Configuration Management Table

| **Section Updated** | **Update Description** | **Submitted by** | **Date Updated** |
| --- | --- | --- | --- |
| Document Creation | Creation and added members | Ibrahim | 4/26/2024 |
| Introduction | Completed | Ibrahim | 4/26/2024 |
| Project management plan | In progress | Ibrahim | 4/26/2024 |
| Requirements Document | Completed | Ibrahim | 4/26/2024 |
| Architecture Document | In progress | Ibrahim | 04/26/2024 |
| Detailed Design | Completed | Ibrahim | 04/26/2024 |
| Test Plan | Completed | Ibrahim | 04/26/2024 |
| Evidence of CM | Completed | Ibrahim | 04/26/2024 |
| Updated Formatting | Completed | Ryan | 04/26/2024 |
| Acknowledgements | Completed | Ryan | 04/26/2024 |
| Engineering Standards | Completed | Ryan | 04/26/2024 |
| References | Completed | Ryan | 04/26/2024 |
| Table of Contents | Completed | Ryan | 04/30/2024 |
| List of Figures | Completed | Ryan | 04/30/2024 |
| List of Tables | Completed | Ryan | 04/30/2024 |

# Engineering Standards and Multiple Constraints

* IEEE Std 1058-1998: Software Project Management Plans [[pdf](https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=25325)]
* PMBOK® Guide: Project Management Body of Knowledge [[pdf](https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1302773)]
* IEEE Std 12207: Software Life Cycle Processes [[pdf](https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8742773)]
* IEEE Std 15939: Measurement Process [[pdf](https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4775910)]
* IEEE 1490: Standard for Software Engineering Management Plans [[pdf](https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1302773)]
* IEEE 828: Standard for Software Configuration Management Plans [[pdf](https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6044675)]
* IEEE Std 830-1998: Software Requirements [[pdf](https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=720574)]
* IEEE Std 29148: Requirements Engineering [[pdf](https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6146379)]
* ISO/IEC/IEEE Std 29148-2018: Systems and Software Engineering
  + Life Cycle Processes [[pdf](https://jumpshare.com/s/wFpfesKlhstVatl5RnsU)]
  + Requirements Engineering [[pdf](https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6146379)]
* ISO/IEC/IEEE International Standard - Systems and software engineering - Requirements for managers of information for users of systems, software, and services [[pdf](https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8584518)]
* IEEE Std 1016-1998-(Revision-2009): Software Design [[pdf](https://cengproject.cankaya.edu.tr/wp-content/uploads/sites/10/2017/12/SDD-ieee-1016-2009.pdf)]
* ISO/IEC/IEEE Std 42030:2019: Software, systems and enterprise
  + Architecture evaluation framework [[pdf](https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8767001)]
* IEEE Std 1471-2000: Software Architecture [[pdf](https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=875998)]
* IEEE Std 829-1983: Software Testing [[pdf](https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=573169)]
* ISO/IEC/IEEE Std 29119-1-(Revision-2022): Part 1 - Software Testing General Concepts [[pdf](https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9698145)]
* ISO/IEC/IEEE Std 29119-2-(Revision-2021): Part 2 - Test Process [[pdf](https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9591508)]
* ISO/IEC/IEEE Std 29119-3-(Revision-2021): Part 3 - Test Documentation [[pdf](https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9591577)]
* ISO/IEC/IEEE Std 29119-4-(Revision-2021): Part 4 - Test Techniques [[pdf](https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9591574)]

# Additional References

* [1] "IEEE/ISO/IEC International Standard - Software and systems engineering--Software testing--Part 4: Test techniques," in ISO/IEC/IEEE 29119-4:2021(E) , vol., no., pp.1-148, 28 Oct. 2021, doi: 10.1109/IEEESTD.2021.9591574. keywords: {IEEE Standards;ISO Standards;IEC Standards;Software engineering;Software testing;System analysis and design}.
* [2] Jorgensen, P.C., 2013. Software Testing: A Craftsman's Approach. Auerbach Publications
* [3] Mathur, A.P., 2013. Foundations of Software Testing, 2/e. Pearson Education
* [4] *Software engineering: Project Monitoring and control - javatpoint*. www.javatpoint.com. (n.d.). <https://www.javatpoint.com/project-monitoring-and-control>
* [5] *Geospatial Data Analytics: What it is, benefits, and top use cases*. SafeGraph. (n.d.). <https://www.safegraph.com/guides/geospatial-data-analytics#:~:text=Geospatial%20data%20analysis%20involves%20collecting,in%20the%20relationships%20between%20places>
* [6] Larson, E. and Gray, C., 2014. *Project Management: The Managerial Process.* McGraw Hill
* [7] 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
* [8] M. T, K. Makkithaya and N. V. G, "A Federated Learning-Based Crop Yield Prediction for Agricultural Production Risk Management," *2022 IEEE Delhi Section Conference (DELCON)*, New Delhi, India, 2022, pp. 1-7, doi: 10.1109/DELCON54057.2022.9752836.
* [9] K. Styk, J. Liszcz and K. Drobek, "Basic Project Management Documentation Based on the Example of the Student Project AGH Lean Line," 2019 8th International Conference on Industrial Technology and Management (ICITM), Cambridge, UK, 2019, pp. 45-49, doi: 10.1109/ICITM.2019.8710717.
* [10] Lamsweerde, A.V., 2009. Requirements Engineering: From System Goals to UML Models to Software Specifications. John Wiley
* [11] NASA. (2023, July 26). *Appendix C: How to write A good requirement*. NASA. https://www.nasa.gov/reference/appendix-c-how-to-write-a-good-requirement/
* [12] Lane, G. Krüger and Charles. (n.d.). *HOW TO WRITE A software requirements specification (SRS document)*. Perforce Software. <https://www.perforce.com/blog/alm/how-write-software-requirements-specification-srs-document>
* *[13] Use-case diagrams*. in UML modeling. (n.d.). <https://www.ibm.com/docs/en/rational-soft-arch/9.6.1?topic=diagrams-use-case>
* [14] J. Horkoff, "Keynote - Requirements Engineering for Machine Learning: Non-functional Requirements as Core Functions," 2022 IEEE 30th International Requirements Engineering Conference Workshops (REW), Melbourne, Australia, 2022, pp. 141-141, doi: 10.1109/REW56159.2022.00034.
* [15] M. Nayebi, "Data Driven Requirements Engineering: Implications for the Community," 2018 IEEE 26th International Requirements Engineering Conference (RE), Banff, AB, Canada, 2018, pp. 439-441, doi: 10.1109/RE.2018.00058.
* [16] Lattanze, A.J., 2008. *Architecting Software Intensive Systems: A Practitioner’s Guide.* CRC Press
* [17] Bass, L., Clements, P. and Kazman, R., 2003. *Software Architecture in Practice.* Addison-Wesley
* [18] Vasav (2023) ‘Client Server Architecture: Types, Examples, & Benefits’, *Red Switches*, 20 October. Available at: https://www.redswitches.com/blog/client-server-architecture/ (Accessed: 18 March 2024).
* [19] Ashanin, N. (2019, June 4). *Documentation in software architecture*. Medium. <https://medium.com/@nvashanin/documentation-in-software-architecture-4f2e4159c4fc>
* [20] Cloudflare Inc., 2024. Cloudflare Docs: Cloudflare Tunnel
* [21] Larman, C., 2012. Applying UML and Patterns: An Introduction to Object Oriented Analysis and Design and Iterative Development. Pearson Education
* [22] Hyman, B., 1998. Fundamentals of Engineering Design. New Jersey: Prentice Hall
* [23] “Reference architecture: Evolving to a SASE architecture with Cloudflare · Cloudflare Reference Architecture Docs,” Cloudflare Docs, https://developers.cloudflare.com/reference-architecture/architectures/sase/ (accessed Apr. 5, 2024).
* [24] Simon, H.A., 2014. A Student's Introduction to Engineering Design: Pergamon Unified Engineering Series (Vol. 21). Elsevier

# Acknowledgments

### Raytheon

Our team would like to thank our sponsor Raytheon for providing this opportunity to work and learn from the project. The ability to visit the site and get hiring / interview experience was beneficial to our continuing careers.

### Marc Perna / Daryll Nelson (Raytheon)

Our team would like to thank Marc and Daryll for the weekly meetings and advice regarding the project and approaches to all levels of the design. As well as general career advice and direction.

### Dr. Eric Wong

Our team would like to thank Dr. Eric Wong for the instruction and guidance regarding proper guidelines in writing reports and communication through project development.

### Brooke Havens (Ryan Havens Spouse)

Ryan would like to acknowledge his wife Brooke for supporting him financially while he completed this coursework and degree.