FTL (Forecasting Time Series Library) - Team5 Software Requirements Specification

Julian Albert, Nathaniel Mason, Hannah Sagalyn, Carlos Villarreal-Elizondo

Github link: https://github.com/n-mason/CS422_Team5_Project1

Table of Contents

1. The Concept of Operations (ConOps)

1.1. Current System or Situation

The current system for stock prediction involves data analysts and traders using various techniques to forecast future stock prices. These techniques may include manual analysis of historical price charts, financial news, and other market indicators, as well as the use of statistical models and machine learning algorithms. However, these methods are often time-consuming and may not produce accurate predictions due to the complexity and volatility of the stock market.

Traders and investors may also rely on stock market experts and advisory services, but these can be costly and may not provide consistent or reliable advice. Additionally, there is a lack of transparency in the stock market, as insider information can affect stock prices, and there is no guarantee of success in investing.

Given these limitations, there is a need for a more efficient and accurate system for predicting stock prices. Our proposed system aims to address this need by providing a centralized platform for contributors to upload historical stock price data and other relevant information, and for data scientists/machine learning engineers to use this data to formulate and train forecasting models.

The proposed system will analyze time series data and generate insights that will help stakeholders make informed decisions, using various techniques, such as statistical models and machine learning algorithms, to analyze the time series data. The system will provide accurate and timely predictions to help users make informed investment decisions and will offer a user-friendly interface that is accessible to all levels of expertise.

1.2. Justification for a New System

Time series analysis is a widely used technique to predict future trends based on historical data. However, many organizations still rely on manual methods to perform this analysis, which can be time-consuming, error-prone, and subject to individual biases. This process involves collecting data from various sources, cleaning and transforming it into a usable format, and applying statistical techniques to identify trends and patterns.

The need for a new method may be beneficial for all customers. Not only may it be error-prone but the process is time-consuming, as it may take several days or weeks to collect and process the data, especially if it comes from multiple sources.

To address these issues, our project aims to develop a web-based application that automates the time series analysis process and provides users with real-time insights into the key drivers of revenue and profitability. The application will leverage machine learning algorithms to identify the most relevant variables and models that explain the historical data

1.3. Operational Features of the Proposed System

The proposed system aims to provide error analysis for time series data. Unlike existing systems focusing on forecasting, this system will analyze and evaluate the accuracy of existing forecasting models. The key operational features of the proposed system include the ability to import time series data, visualize the data, apply forecasting models, and evaluate the accuracy of the forecasts.

The system will also provide users with a variety of error metrics to choose from and allow them to compare the performance of different forecasting models. The system will be user-friendly and provide users with clear and concise error analysis reports. Additionally, the system will have the capability to store and retrieve past error analysis reports allowing for comparisons with previous analyses.

1.4. User Classes

The users for the project are the following:

Data Analyst: Data analysts will have a strong background in statistical analysis and data visualization. They will use the system to import time series data, perform error analysis, and generate reports on the results. They will also have the ability to customize the system's settings and parameters to fit their specific needs.

System Admin: system administrators may be responsible for creating and maintaining the infrastructure necessary for the software to run, and are also involved in the actual development of the software itself.

Software Engineer: Software engineers would be responsible for designing, developing, and maintaining the software system. This could include both front-end and back-end development.

> *Front-end:* Focus on designing and implementing the user interface, creating interactive features, and ensuring that the system is easy to use and visually appealing. This might involve using programming languages such as HTML, CSS, and JavaScript.
>
> *Back-end:* Focus on developing the logic of the system, including the data model, algorithms, and business logic. This could involve using programming languages such as Python and working with databases to store and retrieve data.

Program Manager: A program manager would typically oversee the entire project and be responsible for its success. Such as defining the project goals and objectives, creating and managing the project timeline, ensuring that all work is completed on time, and providing regular status updates to customers.

1.5. Modes of Operation

There will be two main user classes: MLEs and contributors.

The MLEs will be responsible for uploading and analyzing time series data, developing and testing machine learning models, and evaluating the accuracy of the models. They will have a high level of technical expertise and experience in machine learning, statistics, and programming.

The contributors, on the other hand, will be responsible for providing the time series data and associated metadata. They may have varying levels of technical expertise but should have a basic understanding of the domain and context of the time series data they are providing. Contributors will be able to view the results of the machine-learning models, but will not have the ability to upload or modify models.

1.6. Operational Scenarios (aka "Use Cases")

Use Case: Get an error analysis of TS

Brief Description: This use case outlines how a customer can generate an error analysis by utilizing their provided data sets.

Actors: A customer/user.

Preconditions:

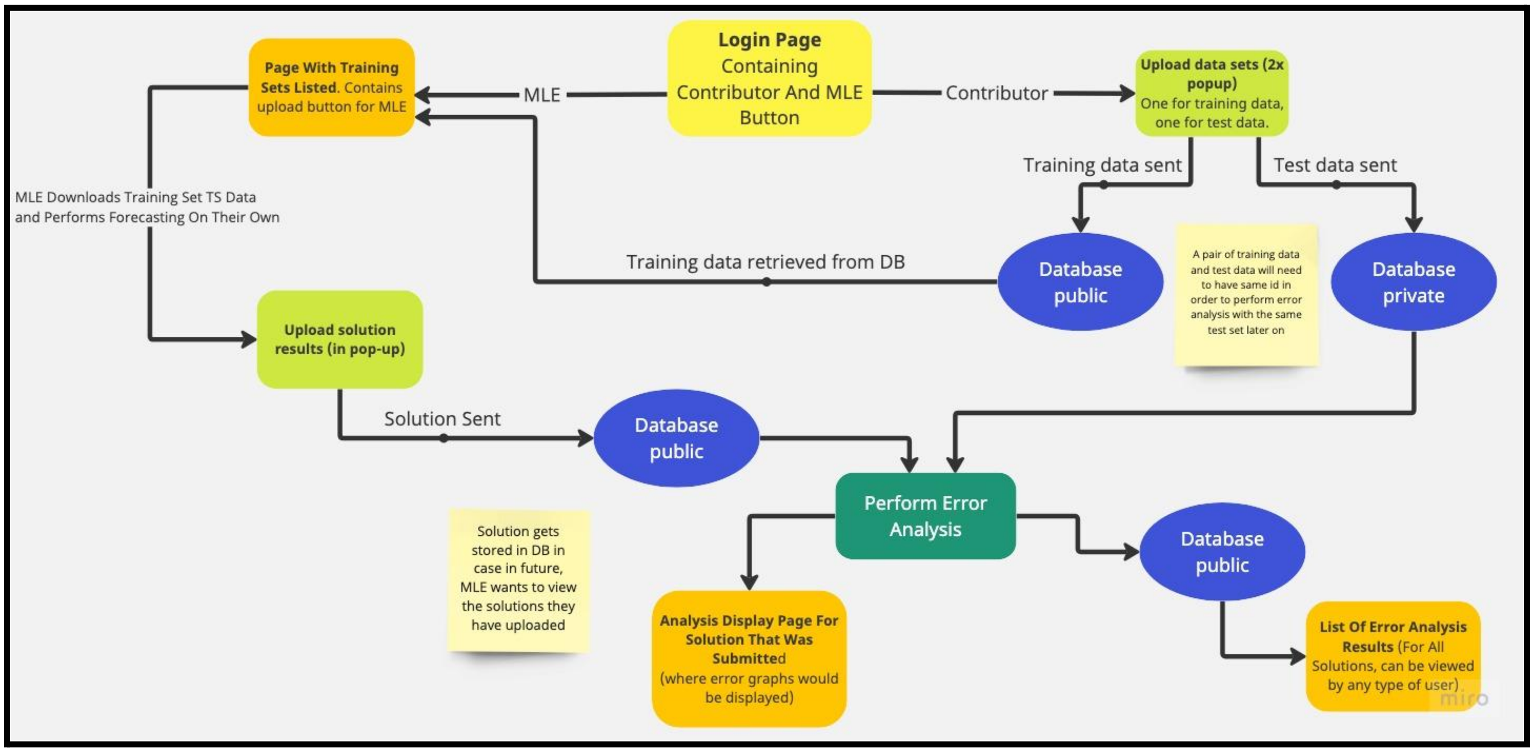
1. The customer has provided the necessary data sets to the system.
2. The system has completed the training and testing phase for the model using the provided data sets.
3. The customer has the appropriate permissions and access to run an error analysis on the model.
4. The system has the necessary software and hardware resources to perform the error analysis.
5. The customer has a basic understanding of the error analysis process and its outputs.

Steps to Complete the Task:

1. Gather necessary data sets that will be used for the error analysis
2. Load the data sets into the system and ensure they are properly formatted for analysis.
3. Select the appropriate training and test cases for the analysis, based on the type of TS and the desired level of accuracy.
4. Configure the system to run the error analysis on the selected training and test cases.
5. Run the error analysis and review the results to identify any areas of concern or improvement.
6. Repeat the analysis as needed to ensure ongoing accuracy and effectiveness.

Postconditions:

The user has a generated error analysis report and is available for review in readable formats. The report includes a detailed breakdown of the errors. The report is saved in a secure location and is accessible only to authorized users that may want to review it again.


Login Page Containing Contributor And MLE Button
MLE
Contributor
Page With Training Sets Listed. Contains upload button for MLE
Upload data sets (2x popup) One for training data, one for test data.
Training data sent
Test data sent
MLE Downloads Training Set TS Data and Performs Forecasting On Their Own
Training data retrieved from DB
Database public
A pair of training data and test data will need to have same id in order to perform error analysis with the same test set later on
Database private
Upload solution results (in pop-up)
Solution Sent
Database public
Perform Error Analysis
Database public
Solution gets stored in DB in case in future, MLE wants to view the solutions they have uploaded
Analysis Display Page For Solution That Was Submitted (where error graphs would be displayed)
List Of Error Analysis Results (For All Solutions, can be viewed by any type of user)
miro

Dynamic Block Diagram representing our system

2. Specific Requirements

1. Introduction
    1.1. The purpose of this system is to provide users with the ability to analyze and interpret time-series data
2. Behavioral Requirements
    2.1. Input requirements
        2.1.1. The system accepts formatted data
    2.2. Processing Requirements
        2.2.1. The system shall perform time-series analysis

2.3. Output Requirements
  2.3.1. The system shall generate tables of MLE for each time-series training set
  2.3.2. The system shall generate visualizations of time-series data and MLE solutions.
2.4. System Performance Requirements
  2.4.1. the system performs analysis on datasets in a fast manner
  2.4.2. The system responds to all user inputs

3. Environmental Requirements
  3.1. Operating System: Is compatible with Mac, Windows, Linux, or other OS.
4. Quality Attributes
  4.1. Usability: System is intuitive and easy to use
  4.2. Error handling: Handles errors gracefully

2.1. External Interfaces (Inputs and Outputs)

This section should describe inputs into and outputs from the software system. (ISO/IEC/IEEE 29148:2011)

Inputs

- Forecast sets computed by MLE
    - Input from MLE
    - Used to calculate time series
    - any kind of file that contains the information for the set (Json, Csv, etc)
    - Units of measure depend on data
    - Information and parameters for aforementioned sets
- Test set passed in by contributor
    - Input from contributor
    - Used to compare to forecast sets
    - any kind of file that contains the information for the set (Json, Csv, etc)
    - Units of measure depend on data
    - Information and parameters for aforementioned sets

Outputs

- Raw difference graph
    - Output to MLE page
    - Used to show difference between forecast and test case
    - Single png image
    - Units of measure depend on parameter, always over time
- Sorted table of error measurements
    - Output to MLE page
    - Used to calculate errors for a single test case
    - Sorted table
    - May be percentages or raw values
- Forecast test file
    - Output to user
    - Used to get a forecast file another user has contributed

- csv file
- units depend on the units in the file itself

2.2. Functions

Validity checks on the inputs:
The software system should check the validity of input data to ensure that it meets the necessary criteria. For example, the system should check that the data is in the correct format, that there are no missing values, and that the data is within a reasonable range of values.

Sequence of operations in processing inputs:
Once the input data has been validated, the software system should perform a sequence of operations to analyze the data. This may include pre-processing the data to remove noise, trend, or seasonality, selecting an appropriate statistical model, estimating the model parameters, and evaluating the model performance. The sequence of operations should be clear and well-documented to enable users to understand the analysis process.

Responses to abnormal situations, including error handling and recovery:
The software system should be able to handle abnormal situations, such as missing data or unexpected input formats. The system should provide informative error messages to users when errors occur and provide suggestions for how to recover from errors. In addition, the system should have a mechanism to save user progress in case of unexpected system failures.

Relationship of outputs to inputs, including:
(a) input/output sequences:
The software system should clearly document the relationship between input data and output results, including the sequence of steps taken to produce the results. This will help users to understand the output results and their relationship to the input data.

(b) formulas for input-to-output conversion:
The system should document the formulas and methods used to convert input data into output results. This will help users understand them if wanting to reproduce the results.

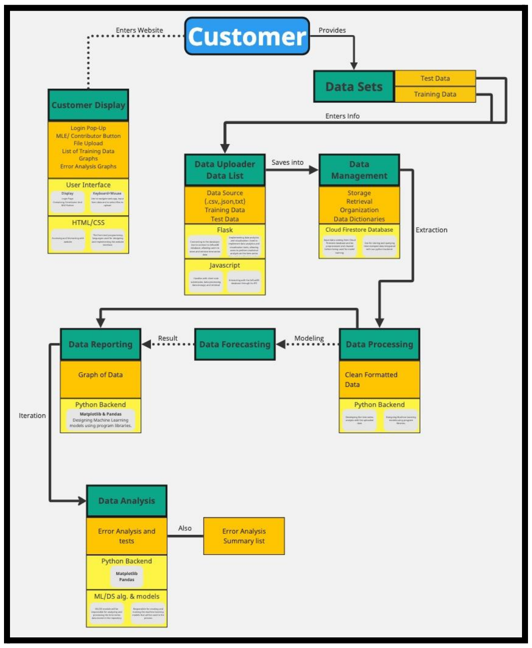
2.3. Usability Requirements

1. **<u>Effectiveness</u>:** The software system is effective in helping users to analyze and interpret time-series data. This can be measured by the accuracy and completeness of the analysis results to achieve their analysis goals. The system provides complete analysis results, covering all relevant aspects of the time-series data being analyzed. This can be measured by comparing the completeness of the results generated by the software with the expected output.

2. **Efficiency:** In order to be efficient, our system is designed with the user's time and effort in mind. The system is easy to navigate, and users should be able to perform tasks quickly and accurately. Users should be able to load and process large datasets quickly and without delay. Another factor that contributes to the efficiency of the system is the ease of navigation and use. The system has a user-friendly interface that is easy to understand and use. Users should be able to perform tasks without having to go through multiple steps or navigate complex menus.

3. **Satisfaction:** The system was designed to meet the needs and preferences of the client and project criteria. Allowing the user to receive the answer and satisfaction they have come for.

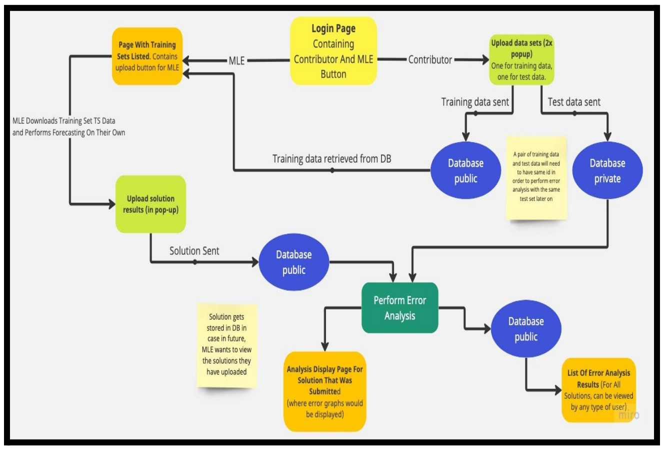
2.4. Performance Requirements



Static Diagram:

1. Customer enters the website with two data sets
    - Training Set & Test Set
2. Front-end to Back-end integration to allow file upload
3. Data is uploaded to the database in the available format
4. Parsing files to a proper format for the next step
5. Parsed data gets implemented into graph form
6. Use algorithms to audit the error possibilities

Dynamic Diagram


Page With Training Sets Listed. Contains upload button for MLE
MLE
Login Page
Containing Contributor And MLE Button
Contributor
Upload data sets (2x popup)
One for training data, one for test data.
Training data sent
Test data sent
MLE Downloads Training Set TS Data and Performs Forecasting On Their Own
Training data retrieved from DB
Database public
A pair of training data and test data will need to have same id in order to perform error analysis with the same test set later on
Database private
Upload solution results (in pop-up)
Solution Sent
Database public
Perform Error Analysis
Database public
Solution gets stored in DB in case in future, MLE wants to view the solutions they have uploaded
Analysis Display Page For Solution That Was Submitted
(where error graphs would be displayed)
List Of Error Analysis Results (For All Solutions, can be viewed by any type of user).
miro

1. The customer opens our website and sees the home screen
   - chooses green button, Contributor, or blue button, MLE
2. Option: **MLE:**
   - List of Metadata added by contributors
   i. Upload solution/test for metadata
   ii. Go to step 3

   Option: **Contributor**:
   a. Form page for data specifications
      i. Both Forms go to two databases; public and private
         2. Go to step 3

3. The site shows all the performance algorithms on error analysis
    a. If wanted, the user may look at all the performance analyses stored on the next site.

2.5. Software System Attributes

- **Reliability:** The software system shall be dependable and accurate in its operations
    - Use established methodologies
        - Scrum and Kanban.
    - Clearly define and document your software requirements and design
    - Implement robust error handling and testing mechanisms to catch and correct any errors or bugs in your code.
    - Conduct thorough testing and validation of your software using a variety of test cases and scenarios.
    - Monitor and maintain your software, including applying updates and patches as necessary to ensure continued reliability and performance.

- **Security:** Ensure that only authorized personnel can access and modify the data.
    - Identify the specific security requirements for your project. This includes specifying what data or assets need to be protected, and who should have access to them.

- **Maintainability:** The software system shall be easy to modify and maintain over its lifetime.
    - Write clean, modular, and well-structured code, and ensure adherence to coding standards and best practices.
    - Implement automated testing and continuous integration processes to catch and fix issues early in the development cycle.
    - Use version control systems to manage changes to the codebase, and ensure that branches and commits are properly labeled and documented.
        - Git, GitHub, etc.
    - Provide clear and comprehensive documentation, including code comments, user manuals, and technical documentation.

- **Usability:** The software system shall be user-friendly and easy to use. It shall have an intuitive interface and provide clear feedback to the user.
    - Create user classes to represent the different types of users that will be using your product.
    - Conduct usability testing to evaluate the effectiveness, efficiency, and satisfaction of the product in meeting the defined usability requirements.

    - Conduct ongoing usability testing throughout the development process to ensure that the product remains user-friendly and meets the defined usability goals.

- **Performance:** The software system shall be able to process a large volume of data quickly and efficiently. It shall be able to handle peak loads without compromising its performance.
    - Specify the performance requirements and objectives for the software system.
    - Optimize the code and algorithms used by the software system to improve performance
    - Ensure that the hardware and software used by the system are appropriate for the expected workload.

3. References

This section lists the sources cited in the creation of this template document. An SRS should reference all of the sources that it draws from. If sufficient citations are provided "in line" (at the point of reference) in the document, this section may not be necessary.

IEEE Std 1362-1998 (R2007). (2007). IEEE Guide for Information Technology–System Definition–Concept of Operations (ConOps) Document. https://ieeexplore.ieee.org/document/761853

IEEE Std 830-1998. (2007). IEEE Recommended Practice for Software Requirements Specifications. https://ieeexplore.ieee.org/document/720574

ISO/IEC/IEEE Intl Std 29148:2011. (2011). Systems and software engineering — Life cycle processes — Requirements engineering. https://ieeexplore.ieee.org/document/6146379

ISO/IEC/IEEE Intl Std 29148:2018. (2018). Systems and software engineering — Life cycle processes — Requirements engineering. https://ieeexplore.ieee.org/document/8559686

Faulk, Stuart. (2013). *Understanding Software Requirements*. https://projects.cecs.pdx.edu/attachments/download/904/Faulk_SoftwareRequirements_v4.pdf

Oracle. (2007). White Paper on "Getting Started With Use Case Modeling". Available at: https://www.oracle.com/technetwork/testcontent/gettingstartedwithusecasemodeling-133857.pdf

van Vliet, Hans. (2008). *Software Engineering: Principles and Practice*, 3nd edition, John Wiley & Sons.

Work Breakdown Structures. In *Wikipedia*, n.d. https://en.wikipedia.org/wiki/Work_breakdown_structure.

N2 Charts. In *Wikipedia*, n.d. https://en.wikipedia.org/wiki/N2_chart.

Functional Flow Block Diagrams. In *Wikipedia*, n.d. https://en.wikipedia.org/wiki/Functional_flow_block_diagram.

Structure Chart. In *Wikipedia*, n.d. https://en.wikipedia.org/wiki/Structure_chart.

Data-flow Diagram. In *Wikipedia*, n.d. https://en.wikipedia.org/wiki/Data-flow_diagram.

Object Diagram. In *Wikipedia*, n.d. https://en.wikipedia.org/wiki/Object_diagram.

System Context Diagram. In *Wikipedia*, n.d. https://en.wikipedia.org/wiki/System_context_diagram.

Storyboard. In *Wikipedia*, n.d. https://en.wikipedia.org/wiki/Storyboard.

Entity Relationship Model. In *Wikipedia*, n.d. https://en.wikipedia.org/wiki/Entity%E2%80%93relationship_model.

4. Acknowledgements

All sources used in the creation of the document and support you received from anyone not in your team should be listed here.