**Smart Evaluator of MRO Supplies**

# Design Document

# Matthew Periut, Cohen Miller, Carter Ray, Justin Hughes, and Dawniqueca Steele

February 14th, 2024

Department of Software Engineering and Game Development

SWE 4724

Dr. Yan Huang

**TABLE OF CONTENTS**

1. INTRODUCTION

1.1 Purpose

1.2 Scope

1.3 Overview

1.4 Reference Material

1.5 Definitions and Acronyms

2. SYSTEM OVERVIEW

3. SYSTEM ARCHITECTURE

3.1 Architectural Design

3.2 Technical Design

3.3 Design Rationale

4. DETAILED DESIGN

5. DATABASE (DATA) DESIGN

6. HUMAN INTERFACE DESIGN

6.1 UI design

6.2 UX design

7. REQUIREMENTS MATRIX

8. APPENDICES

### INTRODUCTION

## Purpose

This design document describes the overall architecture and system design of the Smart Evaluator of MRO Supplies. It also serves as the first milestone in the project development timeline. This documentation aims to aid in the development of the software by providing the architectural, technical, and database designs of the system. It will be primarily used by both the client and development teams. The client will review and evaluate this documentation, and the development team will reference and amend when necessary.

## Scope

The Smart Evaluator of MRO Supplies is a web-based application that serves as the 2024 KSU SWE capstone project in collaboration with CribMaster of Stanley Black and Decker. The platform will feature a Node.js backend and a React and Vite frontend, with a CSS framework like Tailwind. A MongoDB cloud database will be utilized, and infrastructure-as-code may be implemented to enhance deployment flexibility. The Smart Evaluator’s main objective is to provide CribMaster’s sales representatives and other employees with a robust platform that assists them in the research of inventory for their clients. The system will achieve this by allowing users to import spreadsheets of their client’s inventory, and then perform both data collection and vendibility analysis. The process of data collection includes utilizing web-scraping tools and OpenAI API, which will gather key information such as item manufacturer details, SKUs, item cost, and physical details that can impact the vendibility of said item. Next, the system will analyze the vendibility of each item based on a selection of CribMaster’s hardware solutions (i.e. CribMaster ProStock, ProLock, and ToolBox). The system assesses this vendibility by utilizing the data collected to calculate each item’s suitability for each of the storage methods. Once these tasks are performed, the results are stored in the system’s database for future reference to reduce computation time should a particular item already exist.

## Overview

This document’s content contains the architectural and technical aspects of the Smart Evaluator of MRO Supplies. The content is expressed through the use of architectural diagrams, database diagrams, sequential diagrams, and UI prototypes. Overall, the document is designed to provide a robust and thorough understanding of the system’s structure and frameworks.

## Reference Material

1. **Project Plan for the Smart Evaluators of MRO Supplies:** Provides the necessary information about the project, its scope, and its developmental cycle.
2. **Guidelines from the KSU SWE Capstone Class:** Sets the educational standards and expectations for the capstone project.
3. **Documentation on CribMaster Products:** Offers detailed insights into CribMaster’s hardware and software offerings.
4. **OpenAI’s ChatGPT API User Guide:** Critical for understanding how the API functions and what its limitations are.
5. **Software Requirements (Developer Best Practices) by Karl Wiegers and Joy Beatty:** Reference material for requirement engineering and software diagrams.

## Definitions and Acronyms

* **API** – Application Programming Interface
* **MRO** – Maintenance, Repair and Operations
* **SKU –** Stock Keeping Unit

### SYSTEM OVERVIEW

The Smart Evaluator of MRO Supplies is a web-based application that will be utilized by CribMaster sales representatives and employees in their task of researching inventory vendibility for their clients. The system is being developed concurrently with CribMaster’s Solution Generator, whose main objective is to calculate the best hardware solutions based on an organization’s priorities (e.g., high-security, medium-capacity, price range). Thus, vendibility analysis algorithms and data from the Smart Evaluator may be implemented into the Solution Generator at the client’s discretion.

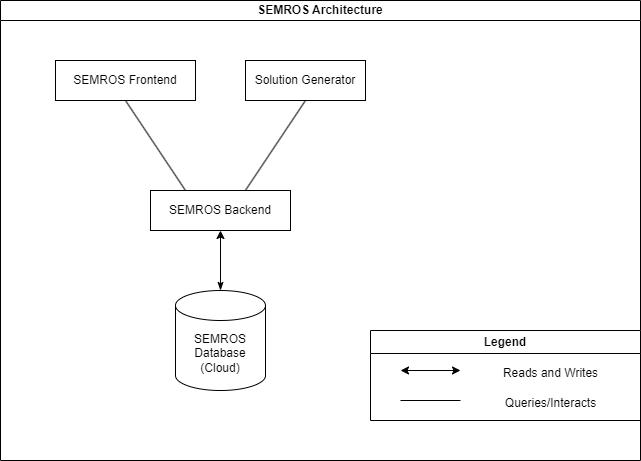
The system’s main functionality is data collection and vendibility analysis. The user will upload .csv spreadsheets files, and the system will read and perform a generative fill to find the missing data for each item on the spreadsheet by utilizing OpenAI API and web scraping tools. Once the data collection is complete, the system will then perform a vendibility analysis, in which it uses the gathered data to find the suitability of each item for any of the selected CribMaster hardware solutions.

### SYSTEM ARCHITECTURE

## Architectural Design

At the highest level of abstraction, the Smart Evaluator of MRO Supplies system only consists of four main components, one of which (Solution Generator) will not be developed directly by the authors of this plan. The architecture follows a service-oriented approach to delivering vendibility analysis, where the backend provides the application logic for calculating vendibility for individual items on a session-by-session basis. This system's components and their logical associations are listed in the diagram below:

**SEMROS Architecture**



**SEMROS Front-end:**

Description:Front end system created by the SWE Capstone team to provide a dedicated user interface to the SEMROS backend for testing, demonstration, and use by the CribMaster salespeople and clients.

Functionality:

* Accepts user input in spreadsheet format
* Uploads user input to backend system
* Queries backend system for item data.
* Displays backend output

**Solution Generator (Sol-Gen)**

Description: Full-Stack web application designed to assist CribMaster clients and salespeople in selecting the most effective CribMaster product based on clients’ needs. In the future, this component will also serve as an interface between users and the SEMROS backend.

Functionality:

* Uploads user input to backend system
* Queries backend system with item data
* Displays backend output

**SEMROS Backend**

Description:This component delivers the primary functionality of the SEMROS project. This component is responsible for each step in the vendibility analysis process: data collection, data analysis, and solution generation. This component receives and processes requests from external clients, namely the front-end and solution generator. This is also the only component that interacts with the cloud database.

Functionality:

* Accepts and loads input to backend system
* Accepts requests for vendibility analysis on an item-by-item basis.
* Potentially Calculates optimal CribMaster product based on the vendibility of all items analyzed.

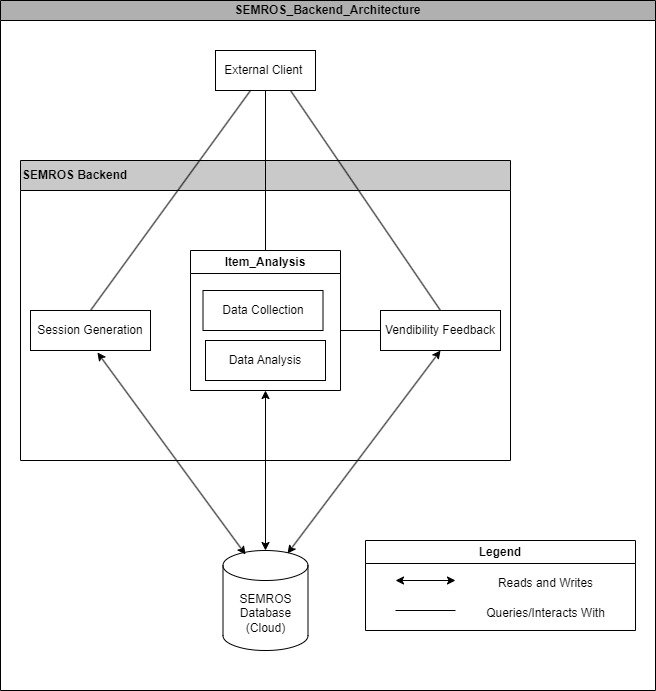
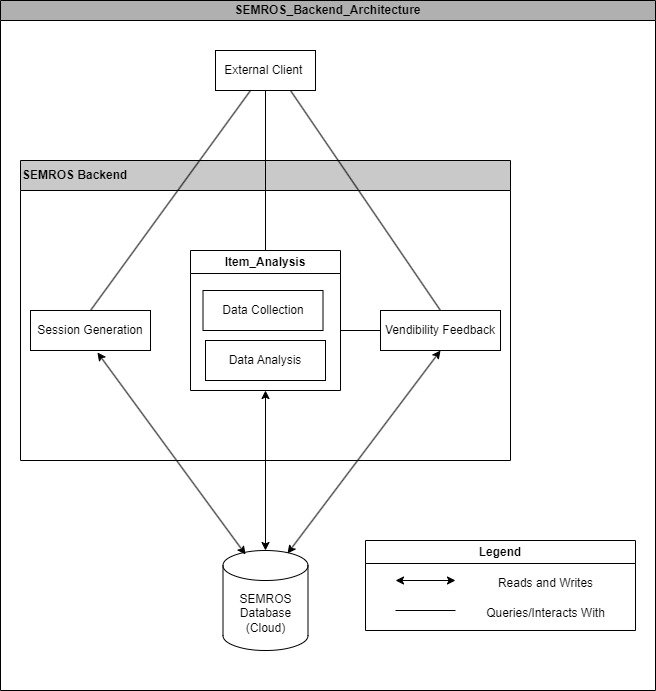
**SEMROS Database**

Description: This component serves as the primary data repository for the system’s backend. It is only accessible by the backend component.

Functionality:

* Store a list of items associated with each session.
* Store a list of previously analyzed items.
* Store calculation and feedback-related information.
* Store information regarding the various crib master vending machines: type, internal dimensions, and vendibility methods.

**SEMROS Backend Architecture**



**Session Generation:** accepts spreadsheet files from the front end and generates a new session and session id. This session id is stored in the database to associate users with their lists. This component also loads spreadsheet data into the database.

**Data Collection Module:** accepts requests from the client on an item-by-item basis, and collects various types of data used for vendibility analysis. This includes manufacturer-related data, physical properties of each item, and any other information pertinent to vendibility. This data is stored in the database for use during the data processing phase.

**Data Analysis Module:** conducts vendibility analysis for each individual tool by analyzing the data stored in the data collection phase. This module analyzes each individual tools’ vendibility.

**Vendibility Feedback Module:** accepts vendibility feedback from the client on an item-by-item basis. Data collected by the data collection and analysis modules may be verified or rejected. This module directly interfaces with the data collection and analysis modules to fine-tune the output for future collection processes.

## Technical design

The technical design is split into two sections, which is the front-end and back-end.

The front-end is to test the API, by being able to provide a way to give the API data to fill in the data and return other information. This fundamentally makes the front-end a test bed for the project.

The back end is the core component of the project, and what is being most intensively worked on. The back end will be split into three modules:

1. Create a database entry called session with a unique identifier for an input document (excel document or compatible relational database information) that contains all the items that the user wants to put inside of vending machines as well as its correlated information. This process will propagate the database session with information relating to the items that are provided.
2. A generative fill using AI (Artificial Intelligence) and Web Scraping will fill empty cells with information using both the row and column identifiers. This will utilize many algorithms to find multiple sources and then cross match them to get the most accurate results.
3. Using AI, there will then be a consideration for each item, in which vending machine the item can go in and which positions are appropriate for the item. Furthermore, after each item’s possible locations have been calculated with AI, there is a regular algorithm that will consider all the possible positions of each item and properly give possible combinations of machines to accommodate for the items.

The Technologies being used to create the project:

1. NodeJS is being used to host both the front end and back end, this serves the purpose of being a server program to host to website with, and having many packages that allows any sort of data processing with the use of NPM packages.
2. Express is a backend framework that allows for the hosting of websites, but in our case most importantly allows us to process a post request and give a response, which is what we are using in the backend of our project.
3. ExcelJS is a framework that has reverse engineered the xlsx format, which is a proprietary format to allow us to use an open-source way of reading excel documents to use when propagating the database in module 1 of the backend. It is also used to load the contents of the excel document in the front end to display the contents before generative fill.
4. OpenAI’s GPT is being used to generate information based on other information of the item. This is used heavily in module 2. In addition is being used to query the relations between item and vending machine, and their slots in module 3.
5. React is a front-end framework that allows for easier integration of components on the webpage with functionality. This is being used in our front-end to quickly and easily prototype changes to test our back-end API.
6. Axios is a front-end framework that allows communication between the front-end and back-end. It is required because express is a back-end framework that accepts post requests, and to test our backend framework, we must make post requests. Axios gives us that ability.

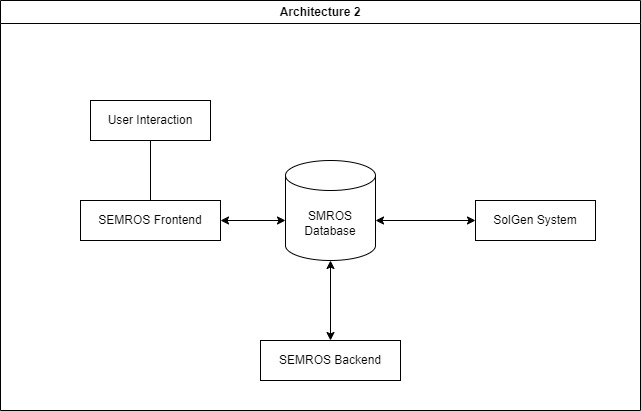
## Design Rationale

Discuss the rationale for selecting the architecture (and its underlying techniques or technologies) described in 3.1 and 3.2 including critical issues and trade/offs that were considered. You may discuss other architectures that were considered, provided that you explain why you didn’t choose them.

**3.3.1 Alternate Architectures for Database Placement**

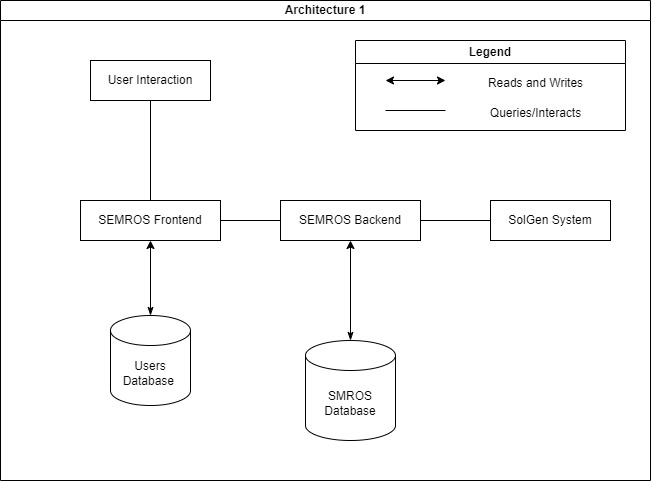
Two alternative architectures for database placement were considered. The first architecture seeks to reduce the coupling between the front-end and back-end components. Notice that in this model, all client systems must interact with the database for data collection.

**Arch. 1: More Database Interaction**



The model below was generated as an alternative to the previous model to reduce coupling between components and reduce the need for database access.

**Arch. 2: Minimal Database Interaction**



### DETAILED DESIGN

The front and backend components of the SEMROS system communicate using a request-response model. There are three types of requests that the external client may use:

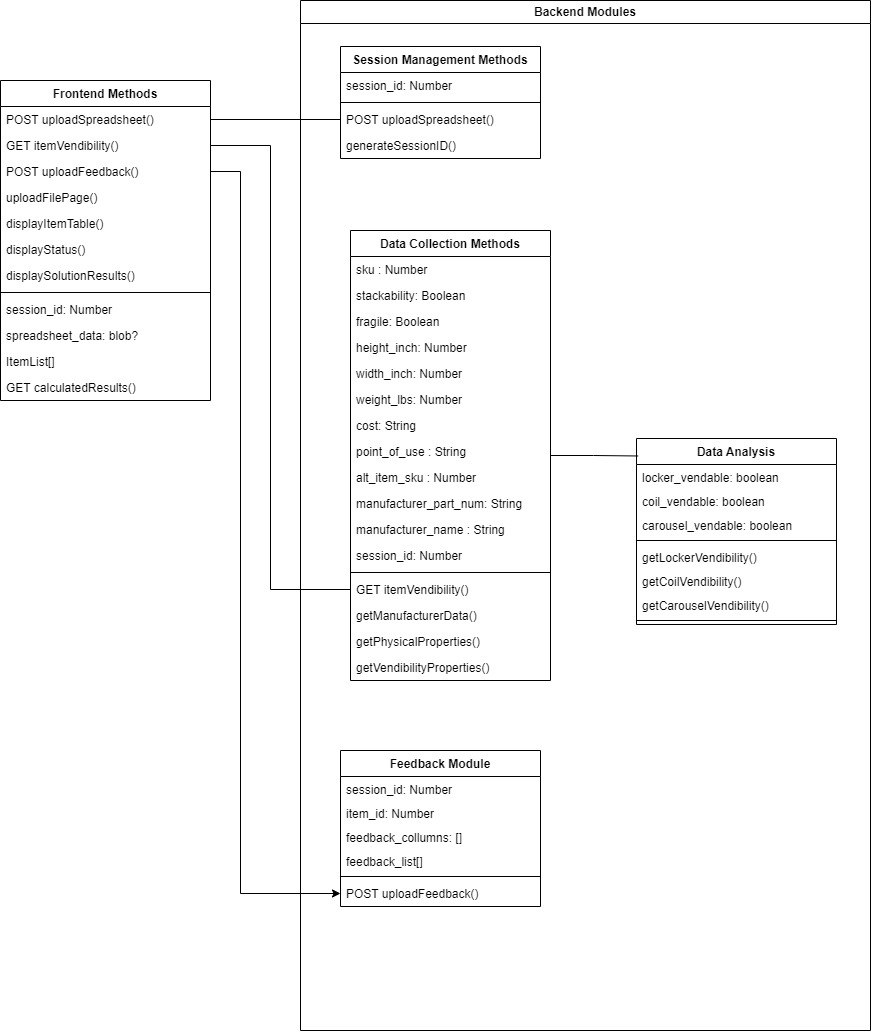
**POST uploadSpreadsheet():** uploads the spreadsheet data to the backend and returns a session ID to be used in subsequent requests.

**GET itemVendibility():** allows the client to request data collection and analysis for a single item. The request must contain the item id and session id, and the response will return any missing fields in the item’s database entry.

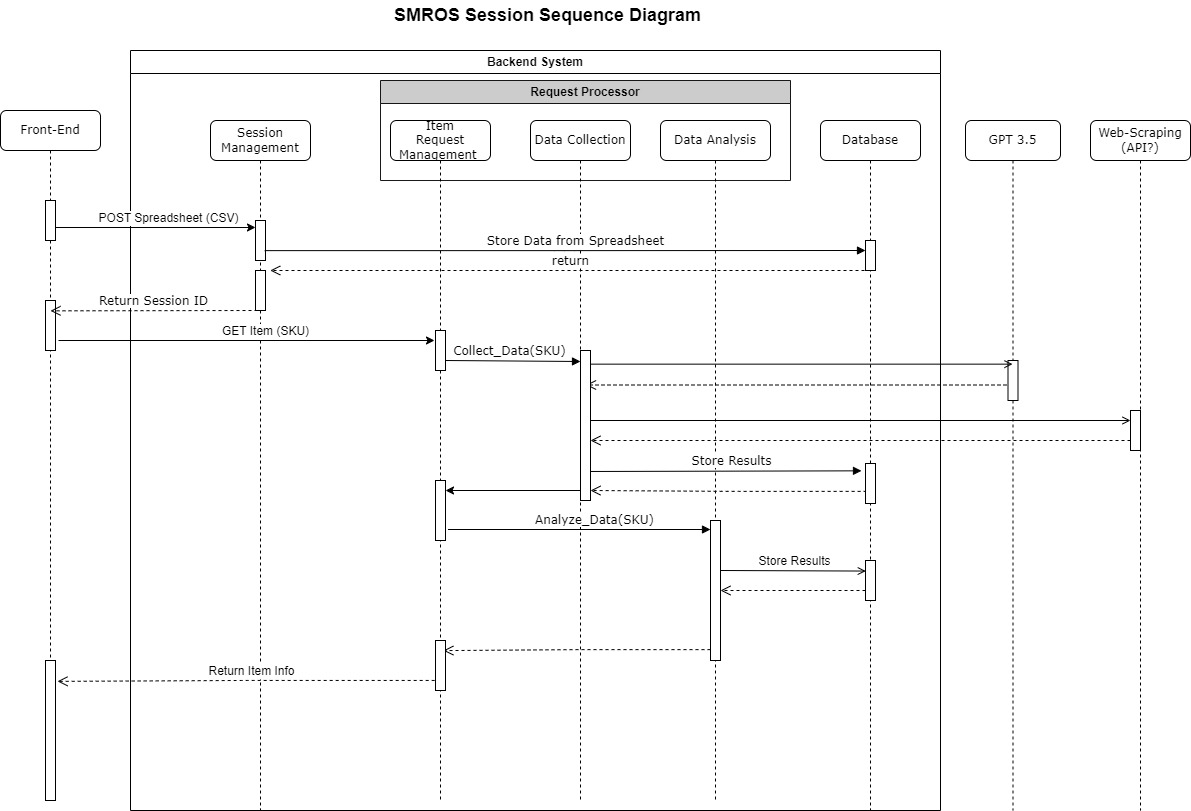
**POST uploadFeedback():** allows the client to provide feedback for one or more data points associated with a single item. Feedback is stored in the database and incorporated into future processing.

Each of these requests are handled by a separate component within the SEMROS backend. These requests and their logical associations are listed in the class diagram below. The session diagram outlines a basic flow of actions during a typical vendibility analysis session.

**SEMROS Class Diagram**

S

**SEMROS Session Sequence Diagram**



### DATA (DATABASE) DESIGN

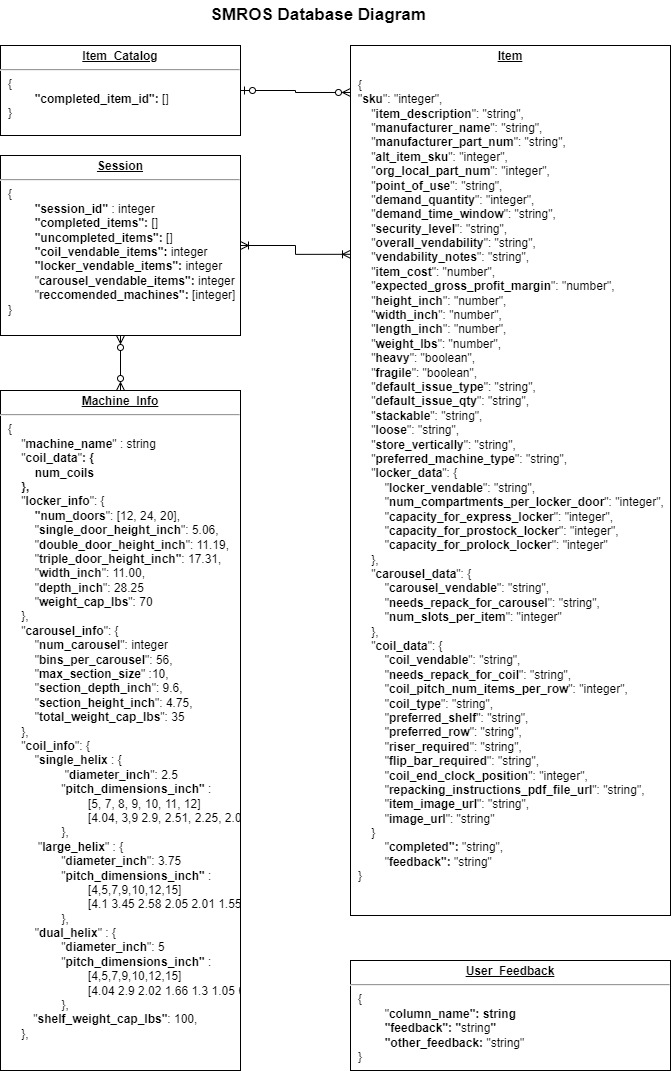
**Item Catalog:** Stores the ID numbers of each item that has been analyzed by the SEMROS system. This does not include items that have uncompleted data. This document will be queried when an item is requested by the external client. If a requested item matches an item stored in this list the system shall return the information associated with the previously calculated item.

**Item:** Stores the item id and its associated information. This includes data collected during both major phases of the vendibility analysis process.

**Session:** Stores all information associated with a particular vendibility analysis session. This includes the session ID, a list of completed item ids, a list of uncompleted items

**Machine Info:** Stores all information associated with a particular CribMaster machine. This information will be used to calculate vendibility during the data analysis phase.

**User Feedback:** Stores various feedback responses generated by the use of the POST uploadFeedback() function. This information is used during data collection and analysis phase to refine those processes.



### HUMAN INTERFACE DESIGN

## UI design

A medium-fidelity prototype was created using Figma. The following screens depict a general sense of workspace and flow for user interaction. To view the prototype on Figma, see the following link: <https://www.figma.com/proto/BcAQgcLax9AbEaDxOoBKuI/Prototype-V1?page-id=0%3A1&type=design&node-id=87-694&viewport=-87%2C349%2C0.18&t=roz6jWULGP5wKxmm-1&scaling=contain&starting-point-node-id=21%3A5&mode=design>.

A screenshot of a chat

Description automatically generated

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a video chat

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

## UX design

The main goal of the UI/UX design is to prioritize user familiarity and effective feedback. The overall user experience is supposed to mimic that of CribMaster’s Solution Generator, but with simple changes to provide our software with some differentiation between itself and the Solution Generator. Mainly, the work dashboard is most similar to the Solution Generator. Since the intended audience for the Smart Evaluator is the same as the Solution Generator, it is important to keep these two consistent, especially considering that the two software may be used in conjunction. Additionally, when the file is being uploaded and the software is calculating the items’ vendibility, there is a specific feedback screen that allows the user to know which step the program is on. The potential files that are going to be using this program may be large and require extensive amounts of data collection, which can result in a long wait time. In order to mitigate user frustration or confusion, providing which step the software is on can allow the user to be more informed and create reasonable inferences on when the calculations may be complete.

### REQUIREMENTS MATRIX

Provide a cross­-reference that traces components and data structures to the requirements in your project.

Use a tabular format to show which system components satisfy each of the functional requirements.

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Requirement** | **Component Name** | **Data Structure** |
| 1. | *The system must accept item lists in the form of a CSV file.* |  |  |
| 1.1 | *The system front-end must accept item lists in the form of a CSV file* | Frontend Component |  |
| 1.2 | *The system front-end must generate requests to the backend that contain input data* | POST uploadSpreadsheet() |  |
| 1.3 | *The system backend must accept requests containing input data* | POST uploadSpreadsheet() |  |
| 1.4 | *The system backend must load input data into the database for processing.* | POST uploadSpreadsheet() | Itemtable |
| 2. | *The system must collect item data that will be used for processing.* | GET itemVendibility() | Item table |
| 2.1 | *The system must collect manufacturer-related data for each item and store it in the database for processing* | DataCollection.getManufacturerData() |  |
| 2.2 | *The system must collect physical properties of each item and store it in the database for processing* | DataCollection.getPhysicalProperties() |  |
| 2.3 | *The system must collect other information pertinent to vendibility and store it in the database.* | DataCollection.getVendibilityProperties() |  |
| 3. | *The system must conduct vendibility analysis for each individual tool by analyzing the data collected during the data collection phase.* | Data Analysis Module |  |
| 3.1 | *The system must analyze data collected during the data collection phase to generate* ***locker-based*** *vendibility data.* | DataAnalysis.getLockerVendability() | MachineInfo.locker |
| 3.2 | *The system must analyze data collected during the data collection phase to generate* ***carousel-based*** *vendibility data* | DataAnalysis.getCoilVendibility() | MachineInfo.coil |
| 3.3 | *The system must analyze data collected during the data collection phase to generate* ***coil-based*** *vendibility data.* | DataAnalysis.getCarouselVendibility() | MachineInfo.carousel |
| 3.4 | *The system must determine the most appropriate vending method (locker, carousel, or coil-based) based on the generated data.* | Data Analysis Module |  |
| 4. | *The system must display the results of the vendibility analysis process* | Frontend Component |  |
| 4.1 | *The system shall display individual item information as it is calculated.* | Frontend Component |  |
| 4.2 | *The system shall display the current phase of the backend component* | Frontend Component |  |
| 5 | *The system must include a mechanism for users to provide feedback on the automated responses* | POST uploadFeedback() | User\_Feedback table |
| 6 | *The system must be designed in such a way that it can integrate with existing systems if necessary.* | POST uploadSpreadsheet() GET itemVendibility()  POST uploadFeedback() |  |