Software Engineering
Software Requirements Specification (SRS) Document

Factory Accountability Management

3/29/2023

Version 1

By:
William Harper
Darian Morales
Yeonseo Choi

Our words and actions reflect academic integrity.
have followed the UNCG Academic Integrity policy.

Table of Contents

1. Introduction

1.1. Purpose

The goal of our application is to allow factories to update and maintain a database of employees, supervisors, and administrators that will record the results of their work and make the facilitation of their duties easier.

1.2. Document Conventions

The purpose of this Software Requirements Document (SRD) is to describe the client-view and developer-view requirements for the Factory Accountability Management (FAM). Client-oriented requirements will describe the system from the client's perspective. These requirements include a description of the different types of users served by the system and the different functions provided by the application to the client. Developer-oriented requirements describe the system from a software developer's perspective. These requirements include a detailed description of functional, data, performance, and other important requirements.

1.3. Definitions, Acronyms, and Abbreviations

Java	A programming language originally developed by James Gosling at Sun Microsystems. We will be using this language to build the application.
MySQL	Open-source relational database management system.
.HTML	Hypertext Markup Language. This is the code that will be used to structure and design the web application and its content.
SpringBoot	An open-source Java-based framework used to create a micro Service. This will be used to create and run our application.
MVC	Model-View-Controller. This is the architectural pattern that will be used to implement our system.
Spring Web	Will be used to build our web application by using Spring MVC. This is one of the dependencies of our system.
Thymeleaf	A modern server-side Java template engine for our web environment. This is one of the dependencies of our system.
NetBeans	An integrated development environment (IDE) for Java. This is where our system will be created.
API	Application Programming Interface. This will be used to implement a function within the software where the current date and time is displayed on the homepage.

1.4. Intended Audience

Section 1 of this document is of interest to all stakeholders in the project. Section 2 is of interest to clients, project managers, and users that want to learn more about the product. Section 3, 4, and 5 describe the functional requirements of the project, and are mainly of use to developers and project managers to establish expectations for the project in development.

1.5. Project Scope

The goal of this software is to allow workers in a factory setting to make use of an easy to understand and easy to use interface to view employee data in a setting where employees may make mistakes that would affect their productivity. With the mistakes correctly recorded and assigned to the correct employee in the database, their supervisors will be able to more easily determine which employees are performing well in their position in order to reward their productivity, or to undertake punitive or corrective action in order to handle underperforming employees. This aligns with the business goals of the client because they are always seeking to optimize their manufacturing processes and supply chain efficiency.

The benefits of this project to the business include:

- Allowing the management team to identify and solve problems with employees
- Provide positive reinforcement to employees to encourage excellence in their job performance. In many hectic jobs it can be difficult to muster the enthusiasm to do a great job. When facing recognition for your hard work, it becomes much easier.

1.6. Technology Challenges

[This section left blank for now]

1.7. References

[This section left blank for now]

2. General Description

2.1. Product Perspective

This product was inspired by a previous job of one of its programmers. When they worked at a factory job, they noticed that the hectic work environment, with excessive heat, noise, and general chaos, could easily lead to mistakes that went unnoticed, and that it was difficult to trace the sources of these mistakes. This program is intended to solve this problem.

2.2. Product Features

This product divides factory employees into 3 separate user groups, with increasing levels of permissions for accessing program data, with all users being able to log in and out using their employee ID. The class with the least permissions, the employee, is able to participate in an abstraction of their work flow that is "gamified" by the prototype of the program in order to demonstrate its functions. The employee is also able to report repeated production line errors to their supervisor, which leads us to the functions available to supervisors. Supervisors have the authority to shut down production lines that aren't properly working using this program. They will also be able to view a database that contains the accuracy data of each employee's product inspections, as well as receive alerts about employees that are consistently underperforming, schedule meetings with employees to discuss issues with the quality of their work, or alert administrators about particularly problematic employees. The highest rank, Administrator, can receive reports from any supervisor in the factory, and they have the ability to read and write to the employee database, effectively allowing them to account for the hiring and firing of employees. Every rank is able to access all functions of their subordinate ranks, to account for situations in low man-power where a supervisor is needed to fill in for an inspector, or an administrator may need to fill in for a supervisor.

2.3. User Class and Characteristics

Our website application does not expect our users to have any prior knowledge of a computer, apart from using a web browser, or knowledge of astronomy. Our website application has removed the need for them to have astronomy, math, or science knowledge and allows the user to focus on exploring the night sky.

Our application is designed for ease of use for any employees and supervisors, with functions available for collating data for supervisors and administrators in an easy to understand way with little prerequisite computer or mathematical knowledge.

2.4. Operating Environment

The application is designed to operate on the web across many different devices.

This application is intended to be run in two different environments: the first environment is at a terminal located at each production line, where employees will "log in" with their ID when assuming their work stations, and the results of their work will populate the database. The second environment will be on any web page, accessible from many different devices.

2.5. Constraints

[May be left blank for now]

2.6. Assumptions and Dependencies

The software will be dependent on Spring Web and Thymeleaf in order to create and execute the MVC architecture that will be developed within NetBeans. The application will also use the World Time API (http://worldtimeapi.org/) to add timestamps to each database interaction.

3. Functional Requirements

3.1. Primary

- FR0: The system will populate and maintain a database that stores the result of an employee's work.
- FR1: The system will allow certain users to look up information about the quality of an employee's work at a glance.
- FR2: The system will prominently display alerts to users sent by other employees.

3.2. Secondary

- FR3: Password protection for information only accessible to supervisors and administrators.

4. Technical Requirements

4.1. Operating System and Compatibility

The application will be compatible with any operating system that is able to view and to interact with traditional web pages.

4.2. Interface Requirements

4.2.1. User Interfaces

The program starts with a login screen, at which the user is prompted to enter their employee ID and password, and if they are at a production line terminal, they will then designate whether they are on the left or right side of the production line, or in the front (product inspector) or back (product packer).

Employees that are in a product inspector station will be able to participate in an abstraction of the employee's work flow that is designed to show the capabilities of the program in prototype form. A product object is generated with a quality attribute that is normally distributed around a specific number that is determined when the user starts. The inspector is prompted to inspect all of these products quickly, and dispose of products that don't meet the threshold of quality. After the production line is observed to be creating a disproportionate amount of low quality products, the employee can send an alert to a supervisor, telling them of a mechanical issue with the production line. After every function, there is a check to the alerts system, which will be displayed on one part of the screen.

The supervisor, when logging in at their office, will be prompted to designate up to 12 of the currently active production lines as being under their supervision. All alerts sent from employees will be sent to the supervisor currently overseeing their line, and these alerts will be visible on one corner of the screen at all times until dismissed. Supervisors will be able to access a function from the home page that allows them to "restart" active production lines, which is abstracted by our prototype by "re-seeding" the mean of the production lines with a new number to be normally distributed around. Supervisors will also be able to navigate to a page which allows viewing database elements. From this page, the supervisor will be able to view the entire database (usually not useful due to the volume of data but still important to include) as well as requesting information about specific employees in either a raw or condensed form that communicates the quality of their work.

The administrator, when logging in at their office, will not be prompted to choose which production lines are under their control, because they are in control of all production lines by default. From their home page, they will be able to access any functions that a supervisor can see, as well as an alerts area, but they will have permission to edit the data in the database to update employee information as well as add and remove employee data.

4.2.2. Hardware Interfaces

The web application will run on any hardware device that has access to the internet, the ability to display webpages, and the ability to interact with web pages. This includes, but is not limited to, smartphones, tablets, desktop computers, and laptops.

4.2.3. Communications Interfaces

It must be able to connect to the internet as well as the local database on phpMyAdmin. The communication protocol, HTTP, must be able to connect to the World Time API and return the current date and time.

4.2.4. Software Interfaces

We will use React and Spring Boot ThymeLeaf to help build the frontend, as well as JPA for the backend database functionality. We will also use Spring Boot with Java to connect the frontend to the backend.

5. Non-Functional Requirements

5.1. Performance Requirements

- NFR0(R): The local copy of the employee database will consume less than 20 MB of memory
- NFR1(R): The system (including the local copy of the employee database) will consume less than 50MB of memory

5.2. Safety Requirements

- NFR2(R): The system will provide confirmation pop-ups to prevent accidental deletions of data or sending of messages.

5.3. Security Requirements

- NFR3(R): The system will only be usable by authorized users.

5.4. Software Quality Attributes

5.4.1. Availability

- Program will be available to clients wishing to purchase it on request. It will not be available over the counter because, in many cases, it will need to be modified to suit the needs of the client before use.

5.4.2. Correctness

- All information stored in the database will be accurate and correct, to give an accurate representation of facts to the client.

5.4.3. Maintainability

- The program will be written in a way that makes it easy for the client to modify to suit their changing needs. Functions will be modular and easy to expand upon for future use.

5.4.4. Reusability

- Because the program will be written in a modular fashion, pieces can be discarded or expanded upon for later use in different contexts if desired.

5.4.5. Portability

- This program will be usable on any program capable of opening a web browser.

5.5. Process Requirements

5.5.1. Development Process Used

- We intend to use a modified waterfall model for this project due to the simplicity of the model and the small size of our team. Numerous checks will be made among the group to ensure that all progress is continuing at a good pace.

5.5.2. Time Constraints

- A prototype must be deliverable and presentable by Mar 21, 2023.

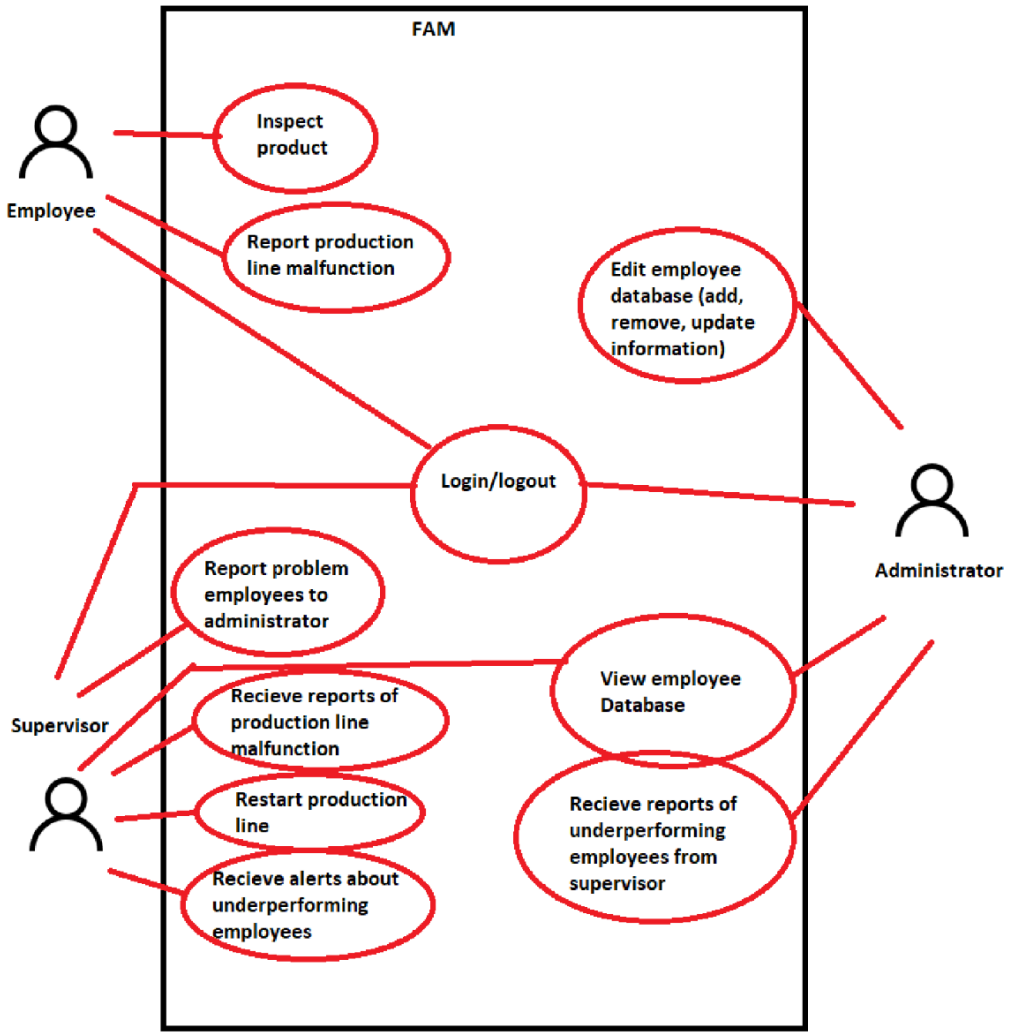
5.5.3. Cost and Delivery Date

- The development of the project is not anticipated to cost any money.
- The delivery date for the final product will be Apr 28, 2023.

5.6. Other Requirements

N/A

5.7. Use-Case Model Diagram



5.8. Use-Case Model Descriptions

5.8.1. Actor: Employee (William Harper)

- **Inspection**: After logging into an inspection station, the employee will experience an abstraction of the product inspection process, approving high quality products and disposing of low quality products.
- **Report Production Fault**: After a period of time when the production line seems to be producing faulty products, the inspector will be able to send an alert to their line supervisor to let them know they may want to consider restarting the production line.
- **Receive Alerts:** The employee can see their current alerts sent to them from supervisors or employees at all times.

5.8.2. Actor: Supervisor (Darian Morales)

- **Restart Production Line**: At any time, a supervisor can issue a command to one of their lines to restart the production lines, or shut them down.
- **View Database**: The supervisor can view any element of the employee database, and there are functions they can call that will present employee data in a way that gives a good idea of the employee's competence and diligence.
- **Send and View Alerts:** Alerts can be viewed at all times, and alerts can be sent to individual problem employees to issue warnings or tell them to meet with their supervisor at the end of their shift to discuss problems with their performance. Alerts can be sent to administrators when all other options have been exhausted to improve an employee's performance and they suggest the employee be fired.

5.8.3. Actor: Administrator (Yeonseo Choi)

- **Edit Database** : The administrator is able to edit information in the database, including adding and removing employees, and editing employee information such as a change in address, name, or contact information.
- **View Alerts**: Alerts from a supervisor about problem employees are displayed.

5.9. Use-Case Model Scenarios

5.9.1. Actor: Employee (William Harper)

- **Inspection**:
  - **Initial Assumption**: Employee is in the database and an employee level or higher.
  - **Normal**: Employee logs in and chooses from the four different stations on a line, Inspector Left/Right, and Packer Left/Right. If they are an inspector, they will experience an abstraction of the inspection process.
  - **What Can Go Wrong:** The employee has lost their employee ID which contains their ID number, and/or has forgotten their ID number and password. They are directed to alert the front office about the issue.
  - **Other Activities**: N/A
  - **System State on Completion**: If the inspection function is not in use then the employee is logged out and the program waits for a new user to log in.
- **Send Alert of Production Fault**:
  - **Initial Assumption**: Employee is currently engaged in the inspection process and 15 minutes have passed in which more than 30% of the products produced are unusable
  - **Normal**: Inspector is prompted to send an alert to their supervisor.

        - **What Can Go Wrong**: The inspector could be disposing of products for no reason, making it look like there is a problem with the production line when it really lies with the inspector
        - **Other Activities**: N/A
        - **System State on Completion**: After the alert is sent the inspector continues with the inspection process.
    - **View messages**:
        - **Initial Assumption**: Employee is in the database and an employee level or higher.
        - **Normal**: Most of the time an employee is logged in and they enter a new page, a section of alerts will be displayed on the page. They can be clicked to view the message, and from the body of the message, it can be marked as read to be hidden from view in the main display.
        - **What Can Go Wrong**: An employee on the floor shouldn't be distracted by messages while working, they will only be alerted to new messages after they have exited the inspection page.
        - **Other Activities**: N/A
        - **System State on Completion**: This is a background function that is always on, it is disabled when the program is no longer running.

5.9.2. Actor: Supervisor (Darian Morales)

- **View Database**:
    - **Initial Assumption**: User has supervisor or administrator permissions in the employee database and has logged in.
    - **Normal**: Supervisor clicks a button to view the database containing all data regarding a worker's efficiency history.
    - **What Can Go Wrong**: If the user is a production employee, they will not be able to see the button that lets them view the database. The data can be overwhelming if the employee is not trained to use the database to figure out how to glean meaningful data from it with the side functions.
    - **Other Activities**: Functions on the database page can restrict the data shown to only data from a specific person, or rank each employee by efficiency on ascending or descending order based on ratio of mistakes to successful inspections.
    - **System State on Completion**: When the supervisor is done viewing the database, he can return to the main page.
- **Send message**:
    - **Initial Assumption**: User has supervisor or administrator permissions in the employee database and has logged in.
    - **Normal**: Supervisor clicks a button that allows them to compose a message to an employee or administrator. Such alerts are useful to schedule meetings with employees to iron out problems, or drawing consistent underperformers to the attention of administrators that have the final say on whether they may be fired.
    - **What Can Go Wrong**: User could accidentally click send when the message is not finished. A prompt will be implemented to ensure that the user is sure that they are done writing the message.
    - **Other Activities**: Templates will be available for routine messages.

- **System State on Completion**: When the supervisor is done sending messages, he can return to the main page.

5.9.3. Actor: Administrator (Yeonseo Choi)

- **Edit Database**:
  - **Initial Assumption**: The user has an administrator rank, and has logged into the program.
  - **Normal**: When accessing the database function, the administrator will have more options available to them. They have permissions to edit employee data, add new employees, and delete employee records.
  - **What Can Go Wrong**: A user could accidentally delete an entry. A prompt will warn the user about what they are about to do, and that it is not reversible, if they want to remove data. A user of this functionality could make spelling mistakes, but they can always go back and correct the data by using this function again.
  - **Other Activities**: N/A
  - **System State on Completion**: Upon finishing making edits to the database, the user will be on the database screen to review their edits in the context of the database.
- **Message system**:
  - **Initial Assumption**: The user is an administrator and is logged in.
  - **Normal**: There will be a section of the screen where recent messages are displayed to the administrator. The administrator will mainly receive messages from supervisors about problematic employees that have demonstrated a pattern of poor performance and attempts to intervene and improve have failed. It is the executive's decision to evaluate the data.
  - **What Can Go Wrong**: Messages deleted can no longer be read. For this reason messages marked as deleted are moved into a trash folder instead and are deleted after a certain period of time, or when the user runs out of designated space to store messages, before which the message can be retrieved.
  - **Other Activities**: N/A
  - **System State on Completion**: This is a background function that is always on, it is disabled when the program is no longer running.

6. Design Documents

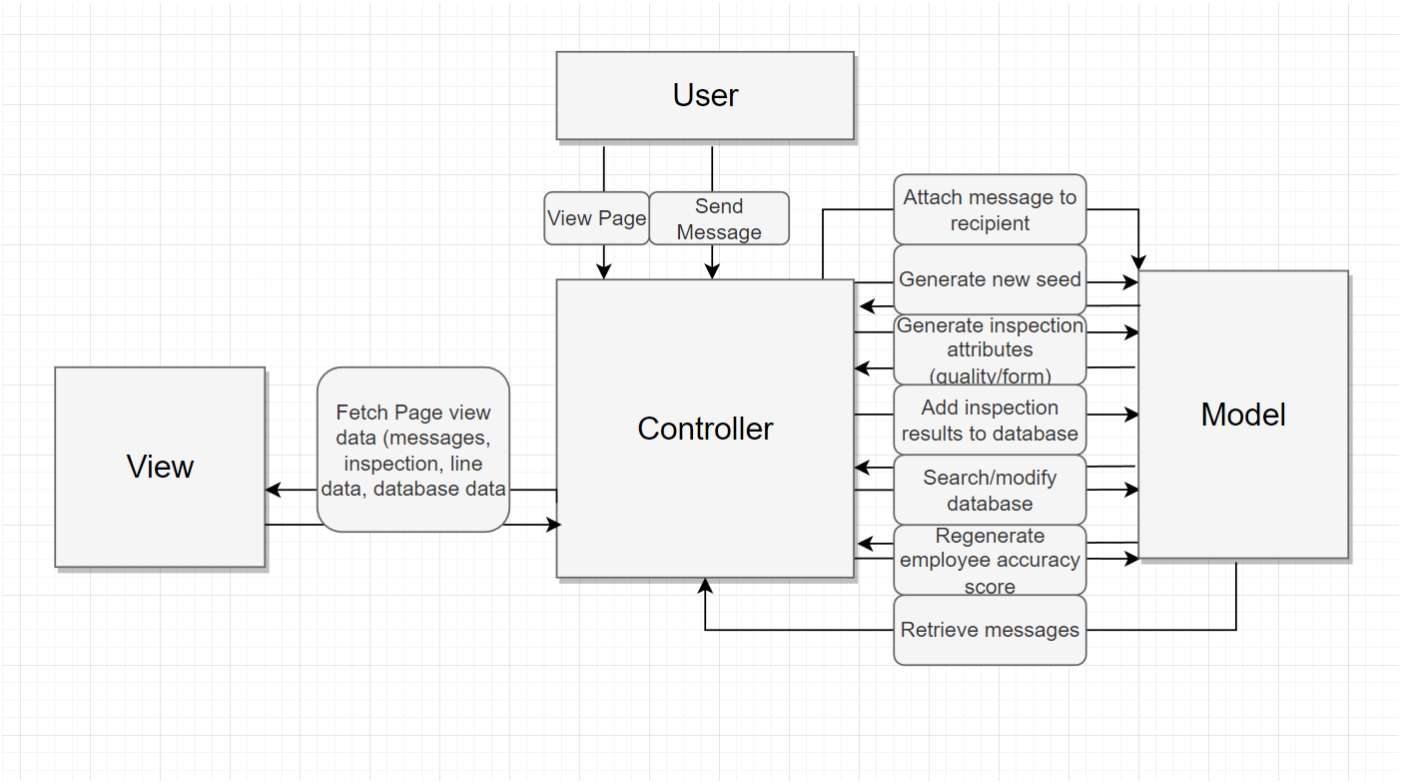
6.1. Software Architecture

6.1.1. We plan to use a MVC architecture with this program.

6.1.1.1. There will be an inspection program inspection() that begins the inspection process, and returns an int value that will contain a 0 in the case that a bad product had passed inspection, a 1 in the case that a good product had failed the inspection, or a 2 in the case that a good product passed inspection, or a bad product failed inspection.

6.1.1.2. This data will be added to the inspection record of the employee that inspected it, which is stored in the database.

6.1.1.3. All other interactions will occur with elements of the database. For example, messages sent between users will be stored as strings in a database that is specific to each user.



6.2. High-Level Database Schema

6.2.1. Data is stored using a series of databases. The first database contains:

6.2.1.1. **employeeID**, which can be referenced to access other tables specific to that employee.

6.2.1.2. **nameFirst,** which stores the first name of the employee

6.2.1.3. **nameLast,** which stores the last name.

6.2.1.4. **clearance,** which is an int that determines what functions the employee has access to. An inspector has 0 clearance, a supervisor has 1 clearance, and an administrator has 2.

6.2.1.5. **phoneNum,** which stores the phone number of the employee as a string.

6.2.1.6. **accuracyScore,** which is a double that can be regenerated by the administrator with their most recent inspection records, to display how well they perform on their inspection duties.

6.2.1.7. **performanceFlag,** a boolean value that flags the employee for poor performance, can be set by administrators or supervisors.

6.2.1.8. **messageDB_ID,** a String value that is the primary key of another table, messageDB

6.2.1.9. **inspectionDB_ID,** a String value that is the primary key of another table, inspectionDB

6.2.2. The second database, messageDB, stores the messages that have been sent to that employee.

6.2.2.1. **ID** is the primary key, and stores the ID of that table.

6.2.2.2. **timestamp** is a string value from the world time API that is applied to the message when it is sent.

6.2.2.3. **messageSender** is a string value that contains the ID of the employee that sent the message.

6.2.2.4. **readOrUnread** is a boolean value that is initially set to false, but can be set to true by the employee that views it to mark it as read.

6.2.2.5. **messageBody** is a string that contains the body of the message.

6.2.3. The third database, inspectionDB, stores the results of all recent inspections.

6.2.3.1. **ID** is the primary key, and stores the ID of this table.

6.2.3.2. **lineNumber** stores the number of the line that was being worked at with the specific inspection record.

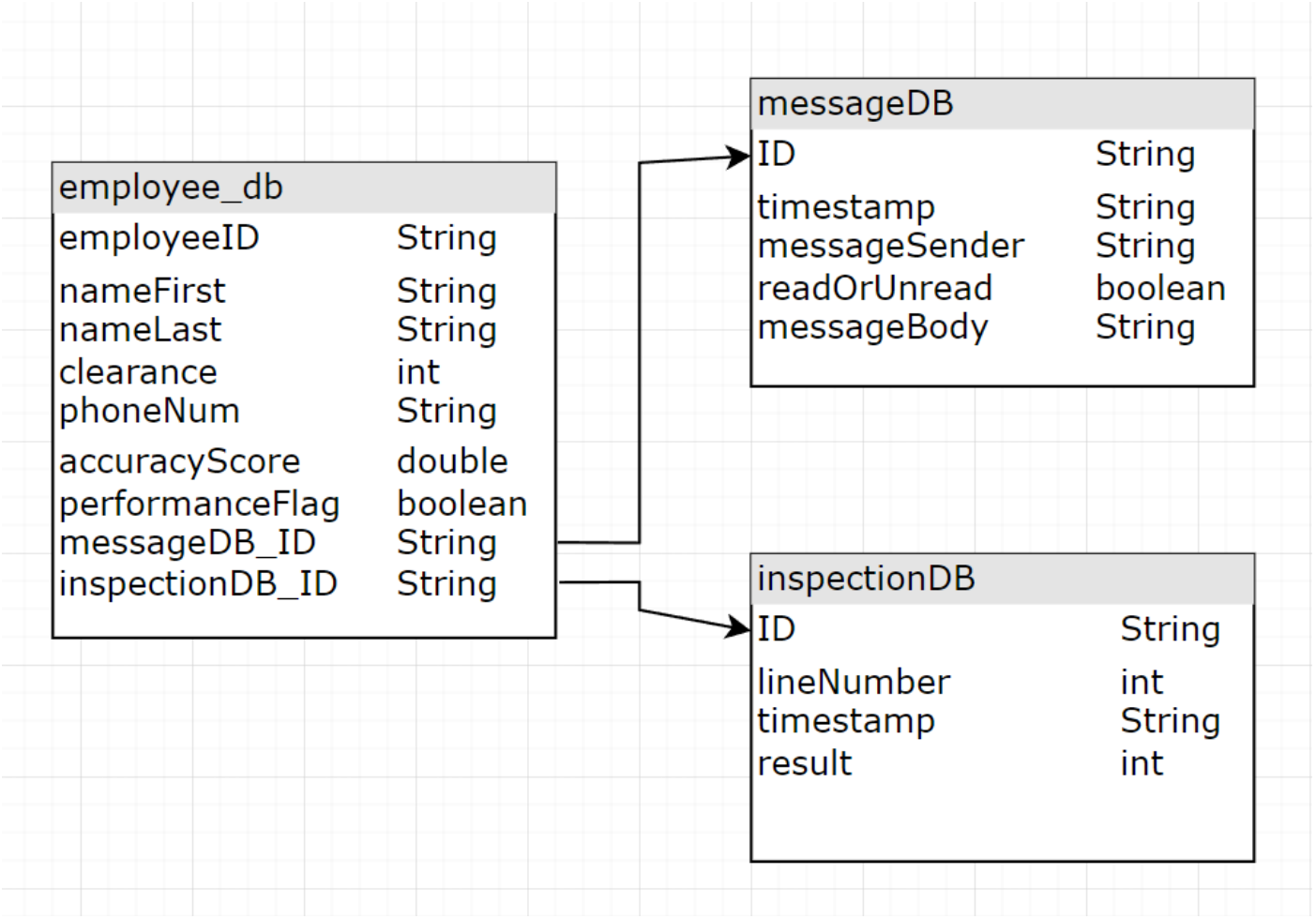
6.2.3.3. **timestamp** is a string value from the world time API that is recorded so there is a record of exactly when the inspection was performed.

6.2.3.4. **result** is an int value that records the result of the inspection.

6.2.3.4.1. A value of 0 represents the outcome of a bad product passing inspection,

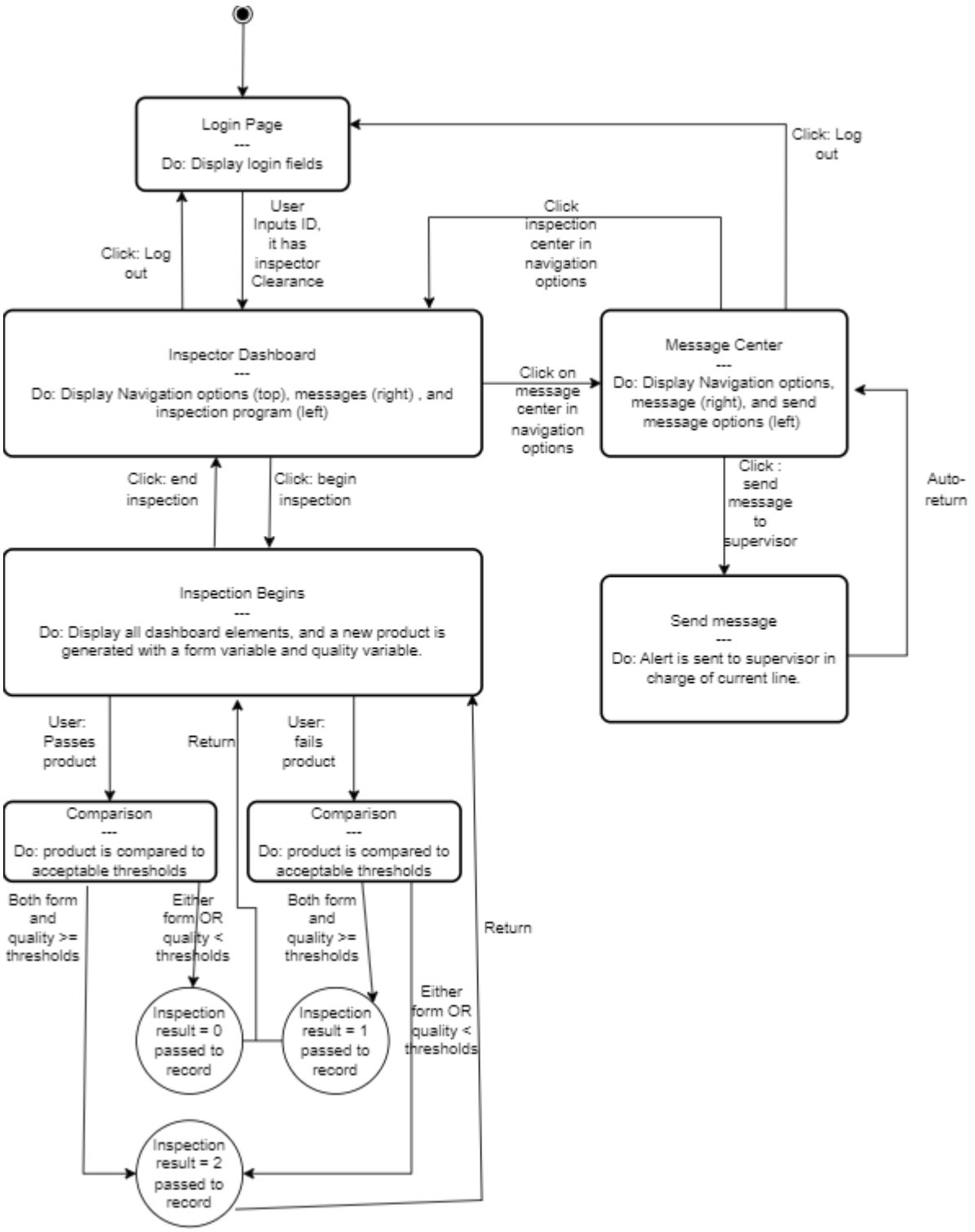
6.2.3.4.2. a value of 1 represents the outcome of a good product failing inspection,

6.2.3.4.3. and a value of 2 represents the outcome of a good product passing, OR a bad product failing inspection.

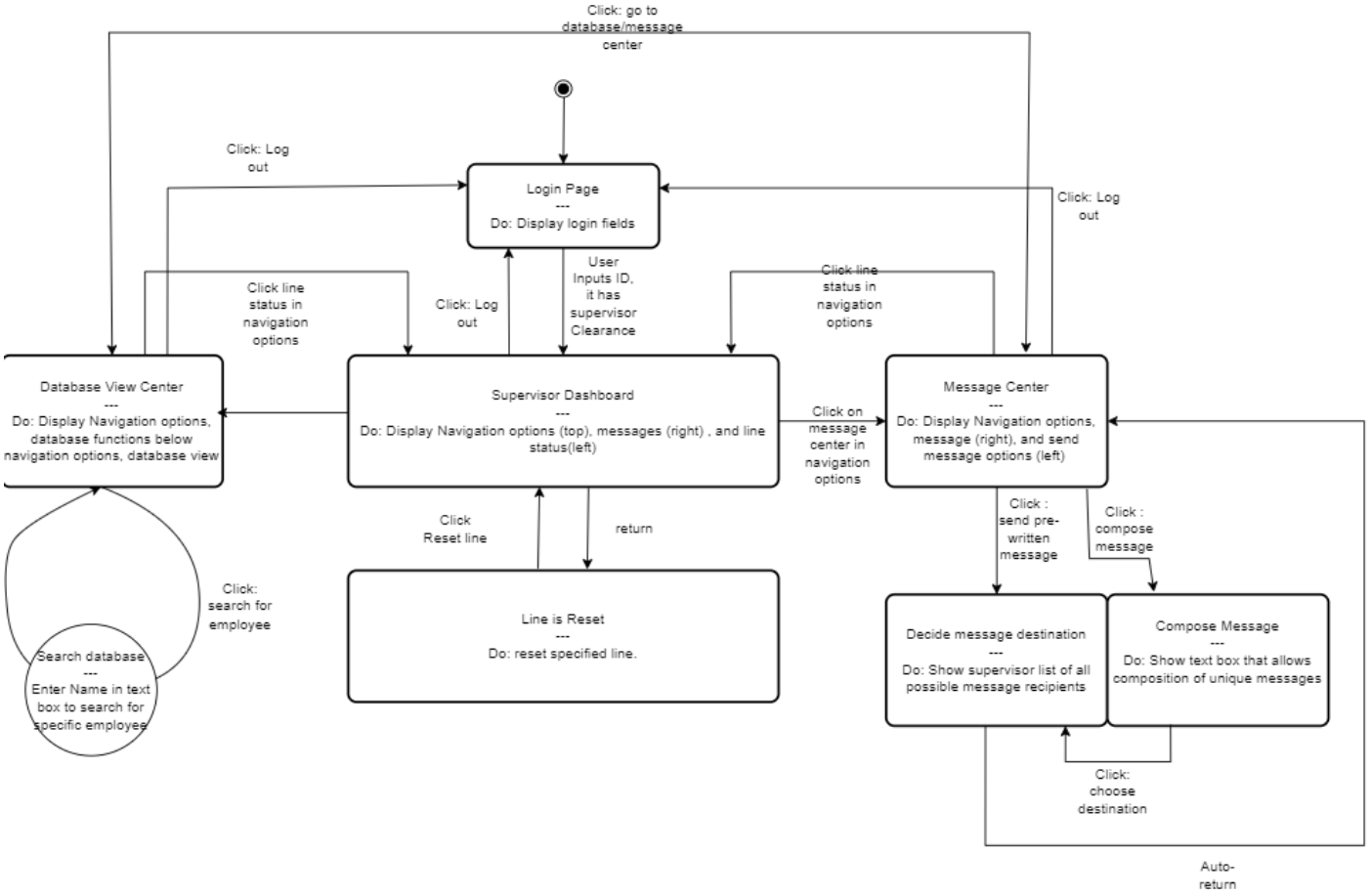

employee_db
employeeID String
nameFirst String
nameLast String
clearance int
phoneNum String
accuracyScore double
performanceFlag boolean
messageDB_ID String
inspectionDB_ID String
messageDB
ID String
timestamp String
messageSender String
readOrUnread boolean
messageBody String
inspectionDB
ID String
lineNumber int
timestamp String
result int

6.3. Software Design

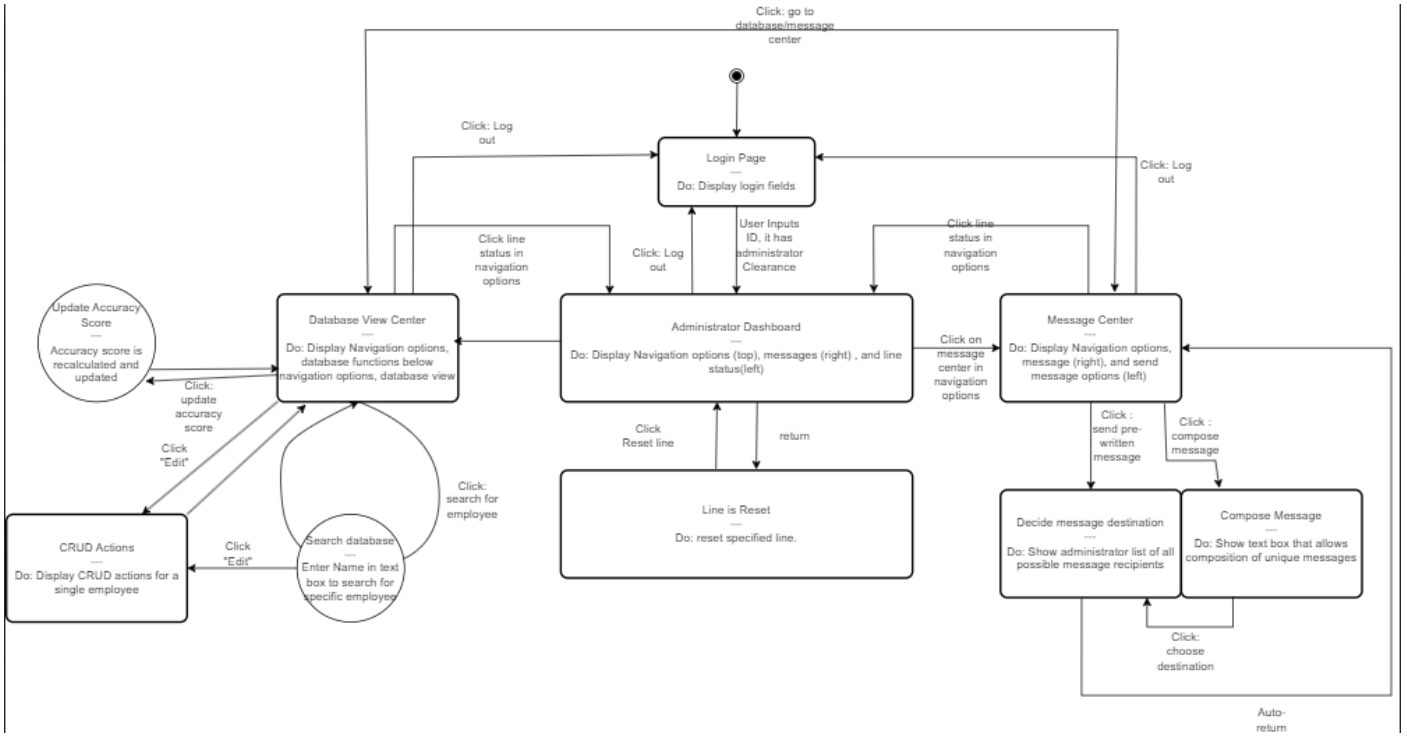
6.3.1. State Machine Diagram: Employee (William Harper)



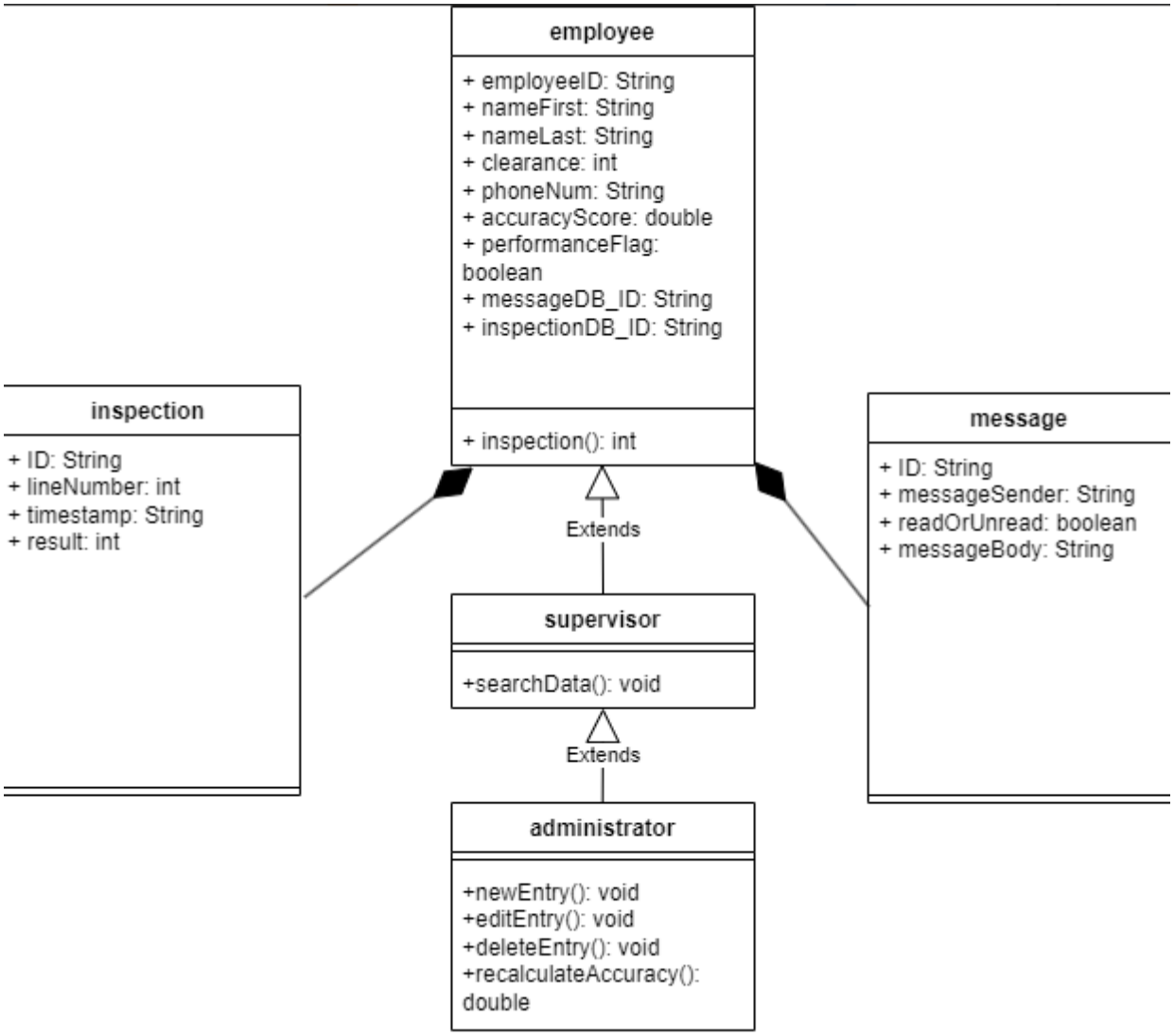
6.3.2. State Machine Diagram: Supervisor (Darian Morales)



6.3.3. State Machine Diagram: Administrator (Yeonseo Choi)



6.4. UML Class Diagram



7. Scenario

7.1. Brief Written Scenario with Screenshots