Software Requirements Specification (SRS)
Automotive Paint Defect Analysis Project

Authors: CSE 435 Team 6:
Stephen Alfa, Logan Arent, Colin Coppersmith, Sean Joseph, James Murray
Customer: General Motors
Instructor: James Daly

1 Introduction

This Software Requirements Specification (SRS) document is to provide a detailed description of all software needed to complete the project. Each subsection in the document will be elaborated further below. The purpose of the Section 1 is to give an overview of the entire system, what purpose the project will serve, and lay the necessary groundwork for the rest of the SRS. Section 2 will go into greater detail about how the system will behave, what dependencies the system has, and what assumptions will have to be made to accomplish our goal. Section 3 is all about our specific requirements for how we would like the system to look, behave, and how we are meeting our goal. Section 4 is about modeling our system so when it is designed, it will make sense how all of our classes behave, and other models will explain how the system will look to the user. Section 5 is about the prototypes of the system, how to run them, and how they should behave. Section 6 includes all of our references needed throughout the SRS document (not yet applicable). Section 7 includes information about our point of contact for this project, Professor James Daly.

1.1 Purpose

The purpose of the SRS document is to provide the necessary requirements for the team of developers to complete their task, and outlines specifically how the team will meet their goal(s). The intended audience is all of those directly involved with the project, including developers, and project managers. Some models included in the document may be useful to the clients, but the intended use for the SRS document is to outline the system and implementation for the developer.

1.2 Scope

The scope of the project is to create a system to replace the current paper-based process of recording paint defects and make creating reports on the defects a trivial task. The system will eliminate the need for paper records and save analysts a lot of the time previously required to generate reports. The domain of our product is the web.The product will a web interface for analysts to record the defects of a vehicle, including type, location, and severity information. The product will also allow editing of data after it has been entered, and allow analysts to easily create daily, weekly, and monthly reports as well as reports including a custom set of vehicles over a custom time period. The product will support the analysis of paint defects over time and ensure the security of the system by requiring user verification. This product won't make any decisions about the information or give proposals to diminish paint defects on the cars. This product won't mechanize the transferring of old information. This product won't identify and record defects on cars.

1.3 Definitions, acronyms, and abbreviations

The terms and abbreviations are as follows:
Analyst: User for the system, interacts with the system by marking on the vehicle models where the paint defects are, the type of the defect, and the severity of these defects.

Data: Values to be input into the system that enables it to be able to generate a report, such as location, type, and severity of the defect, or number of cars, etc.

Checkpoint: A point in the vehicle production process where the vehicle is inspected.

Report: The diagram of the vehicle model displayed with defects that is outputted based on the data provided. Includes a defect type legend, the analyst’s name, date, and other important information, and may be generated based on time period. Can be at different checkpoints in the assembly line as well, such as Prime Review, Polish Deck, etc.

Vehicle Model: Diagram of the types of cars that will be involved in our system. The models will be different for each vehicle model (including GMC Acadia, Chevrolet Traverse, and Buick Enclave), and for each different vehicle, the vehicle model will include a right vertical view, left vertical view, and roof outline.

Automotive Paint Defect: When the exterior of a vehicle contains flaws or imperfections in the paint and finish.

Defects Per Unit: The number of defects on a specific area of the vehicle.

Abbreviations: Automotive Paint Defect (APD), Defects Per Unit (DPU)

1.4 Organization

The rest of the Software Requirements Specification will contain: the product perspective (2.1), product functions (2.2), user characteristics (2.3), constraints (2.4), assumptions and dependencies (2.5), approportioning of requirements (2.6), specific requirements (3), modeling requirements (4), how to run the prototype (5.1), sample scenarios of running prototype (5.2), references (6), and point of contact (7).

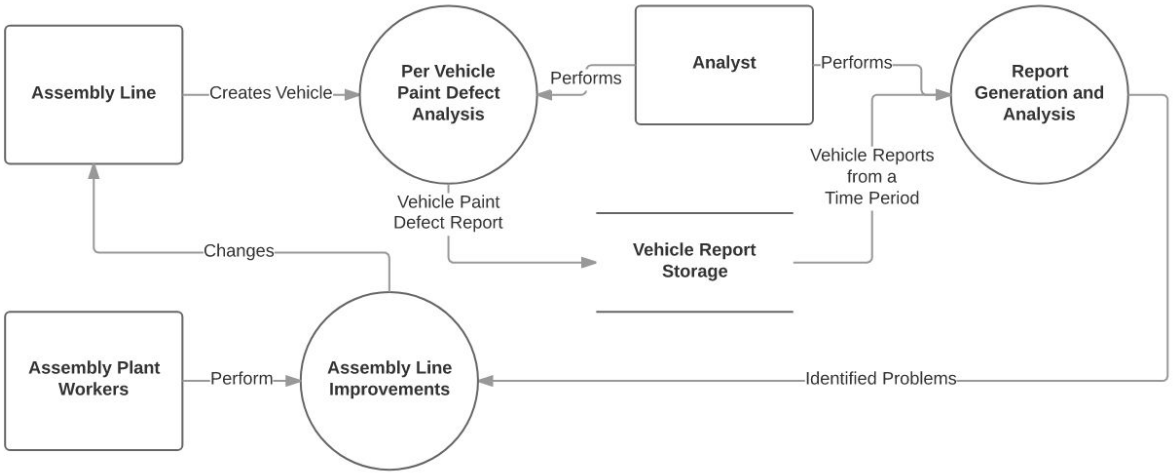
2 Overall Description

This section details the environment and functions of the product as well as who is expected to use it. Constraints, assumptions made for development, and systems that the product depends upon are addressed in this section. Finally, a division of requirements is proposed, dividing features between the initial product and later versions.

2.1 Product Perspective

The product will be used in vehicle manufacturing plants. An analyst will use the system to record paint defects while examining cars and later generate reports on these defects. These reports are used to track daily rates of paint defects as well as trends in defects on the different surfaces of the vehicles being manufactured. Examination of these trends can lead to improvements in the manufacturing process. This system is diagrammed in Figure 1.

The product will have two main sides to it, one for recording paint defects and another for generating reports on the defects. These sides will be used in different environments so they have different constraints. When an analyst is recording paint defects, they will be on the assembly line, moving around the car. This means they will be using a portable device, most likely a tablet computer to record the defects. This will be a necessary change from the paper and pencil system that is in place. Due to the nature of assembly lines, the analyst must be able to record defects rapidly so the product is does not decrease the rate of production of the assembly line. For the report creation side of the product, the user is expected to use a desktop computer. The report creation process is designed to save the analyst time, so it needs to be easy to use and efficient.



2.2 Product Functions

Our product will save the analysts time by streamlining their workflow and allowing for easy creation of reports. A list of functions taken from the Requirements Definition [1] follows:

1. Implement a way for analysts to record the defects of a vehicle, including type, location, and severity information.
2. Allow editing of data after it has been entered.
3. Allow analysts to easily create daily, weekly, and monthly reports as well as reports including a custom set of vehicles over a custom time period.
4. Support the analysis of paint defects over time.
5. Ensure the security of the system by requiring user verification.

2.3 User Characteristics

The users of our product will be paint defect analysts already employed at manufacturing plants. The analysts will be skilled in identifying paint defects and their severity and familiar with creating reports on these defects. The users will be comfortable using a tablet computer with a touch-screen interface as well as a desktop computer which they will have use to do tasks such as downloading files and printing reports.

2.4 Constraints

This system does not have any safety critical components as it is merely keeping track of reported defects and generating reports based on the reports. It might be detrimental to lose data due to a system crash or a bug, but no one will be in mortal danger if a few days worth of data is lost..

If some data is not saved properly or data entries get deleted, the system will not be able to generate the reports. These mistakes would only occur at the hands of a user with a fundamental misunderstanding of the way the system functions. Thorough training is recommended in the reporting software to prevent misuse of the forms.

2.5 Assumptions and Dependencies

This software will require a some type of computer to be present on the line when workers are inspecting and recording defects. It will also require that the computer have a basic internet browser installed on it as this software will exist as a web application. User interaction will be primarily through forms. Form types will be data input, or requests for different reports from the logged data.

2.6 Approportioning of Requirements

This software will extend functionality to any arbitrary car model. The primary focus should be on getting the report functionality working before implementing an interface for adding new vehicles to the reporting system.

3 Specific Requirements

1 External Interface Requirements

- 1.1 User Interfaces
  - 1.1.1 The system should work through website navigation.
  - 1.1.2 The system interface should be easy to navigate.
  - 1.1.3 The system must include clear ways to get to every page.
- 1.2 Communications Interfaces
  - 1.2.1 Analyst can leave notes about any report as necessary for others to read.

2 Functional Requirements

- 2.1 Create Session
  - 2.1.1 Analyst should be able to log in and create a session.
  - 2.1.2 Analyst should be able to edit created sessions.
- 2.2 Add Car
  - 2.2.1 Analyst should be able to add cars to the current session.
- 2.3 Add Defect
  - 2.3.1 Analyst can add multiple defects for any car in the session.
  - 2.3.2 Analyst can enter location, type, and severity of each defect.
- 2.4 Add Report Time Interval
  - 2.4.1 Analyst can specify time interval over which report is generated.
- 2.5 Generate Report
  - 2.5.1 The system is able to take the data provided and display a report.
  - 2.5.2 Analyst can add as many reports to the system as desired.
  - 2.5.3 Report displays defects and their type and severity.
  - 2.5.4 Defect locations are displayed on car visuals.
- 2.6 Update Report
  - 2.6.1 Analyst can edit any of the system's reports at any time.
- 2.7 Remove Report
  - 2.7.1 Analyst may delete any report at any time.

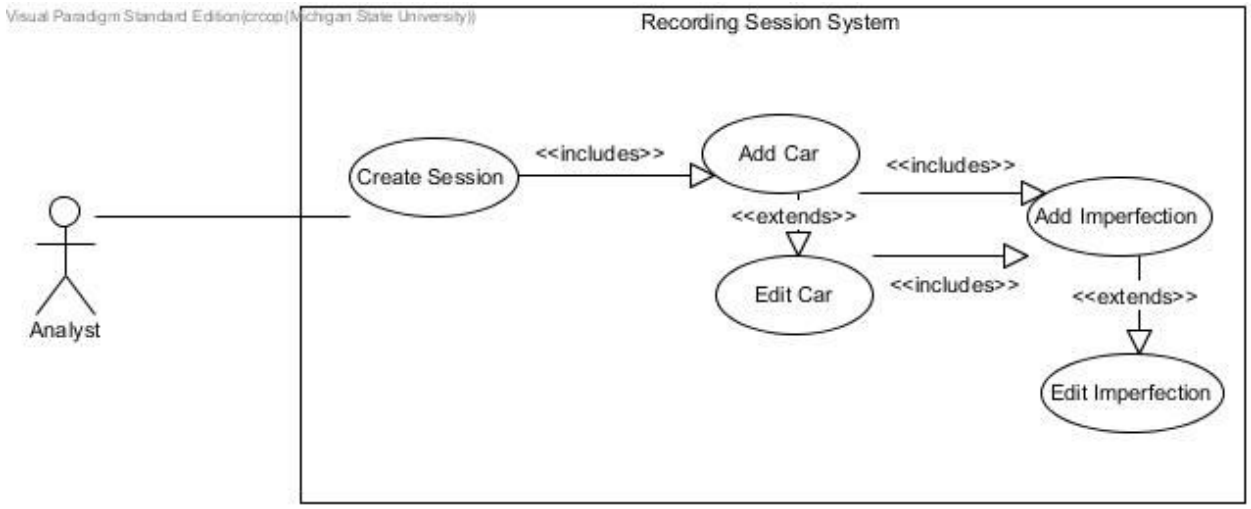
3 Performance Requirements

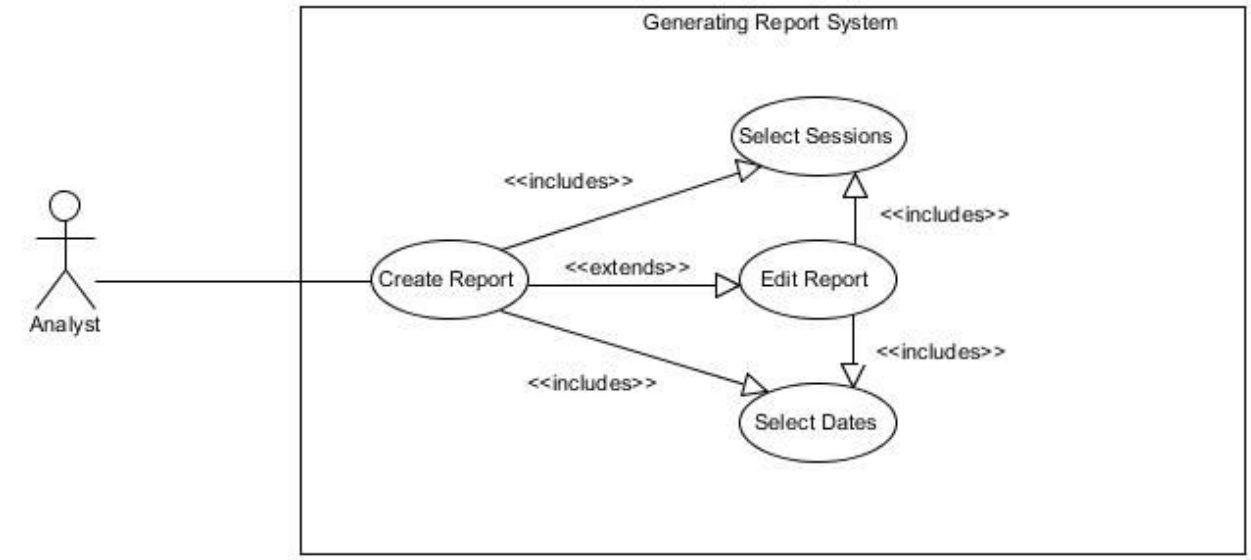
- 3.1 Reports are generated very quickly.
- 3.2 Website navigation should be smooth and always work.

4 Quality Requirements

- 4.1 The system must be available for use at all times.
- 4.2 Defect analysis is very clear and displayed correctly.

4 Modeling Requirements





Use Case: Create Session
Actors: Analyst
Type: Primary and Essential

Description: The Analyst clicks a button on the screen to create a session. This is where the analyst will input all imperfections for as many vehicles as they want. The analyst must select a location here. Many sessions are used to generate a report, many cars are in a session, and many imperfections are in a car.
Cross Ref.: Requirement 1

Use Case: Add Car
Actors: Analyst
Type: Secondary
Description: The Analyst clicks a button within the session screen to add a car. The Analyst will select a type of car. Here, the user will add imperfections.
Cross Ref.: Requirement 2
Use-Cases: Analyst must have completed the *Create Session* use case.

Use Case: Edit Car
Actors: Analyst
Type: Secondary
Description: The Analyst can edit a car from the main session screen. Here, the Analyst can make any necessary changes to a car, like add or edit imperfections as well. Special case of the *Add Car* use case.
Cross Ref.: Requirements 2, 3
Use-Cases: Analyst must have completed the *Add Car* use case.

Use Case: Add Imperfection
Actors: Analyst
Type: Secondary
Description: The Analyst a button on the screen to add as many imperfections as are applicable to this car. The Analyst will select a location and severity of imperfection.
Cross Ref.: Requirement 3
Use-Cases: Analyst must have completed the *Add Car* or *Edit Car* use cases.

Use Case: Edit Imperfection
Actors: Analyst
Type: Secondary
Description: The Analyst can edit an imperfection from the add car or edit car screen. Here, the Analyst can make any necessary changes to an imperfection, like change location or severity. Special case of the *Add Imperfection* use case.
Cross Ref.: Requirement 3
Use-Cases: Analyst must have completed the *Add Imperfection* use case.

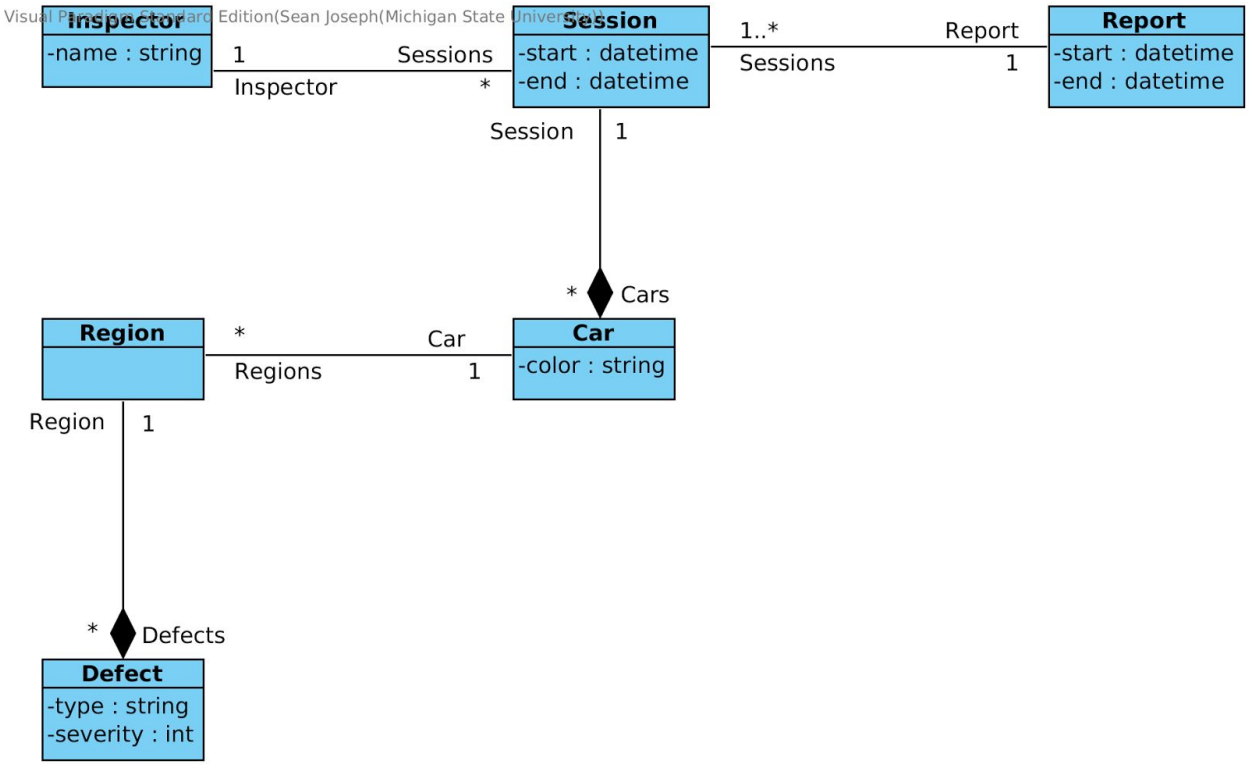
Use Case: Create Report
Actors: Analyst
Type: Primary and Essential

Description: The Analyst clicks a button on the main screen to generate a report. From here, the Analyst will select a type of report that they want to generate, and the Analyst must select a location of where the report will be generated from. From here, the Analyst will go on to select the reports or dates that they want to generate a report from.
Cross Ref.: Requirement 5

Use Case: Select Sessions
Actors: Analyst
Type: Secondary
Description: The Analyst can select sessions, which will be given a listed format, from the pre-selected location. An Analyst must select at least 10 sessions in order to successfully create a report. The Analyst can also edit their changes at this point, selecting and deselecting sessions as they like.
Use-Cases: Analyst must have completed the *Create Report* or *Edit Report* use case.

Use Case: Select Dates
Actors: Analyst
Type: Secondary
Description: The Analyst can select a start date and an end date for which they would like to generate a report of for the pre-selected location. The length of time selected must be at least a week, as most reports generated are either weekly or monthly reports. The Analyst can also edit their changes at this point, changing start and end dates as they like.
Cross Ref.: Requirement 4
Use-Cases: Analyst must have completed the *Create Report* or *Edit Report* use case.

Use Case: Edit Report
Actors: Analyst
Type: Primary
Description: The Analyst can edit a report, which will allow them to delete the report, change the report location, change the report type or access the *Select Sessions* and *Select Dates* use cases.
Cross Ref.: Requirements 5, 6, 7
Use-Cases: Analyst must have completed the *Create Report* use case.

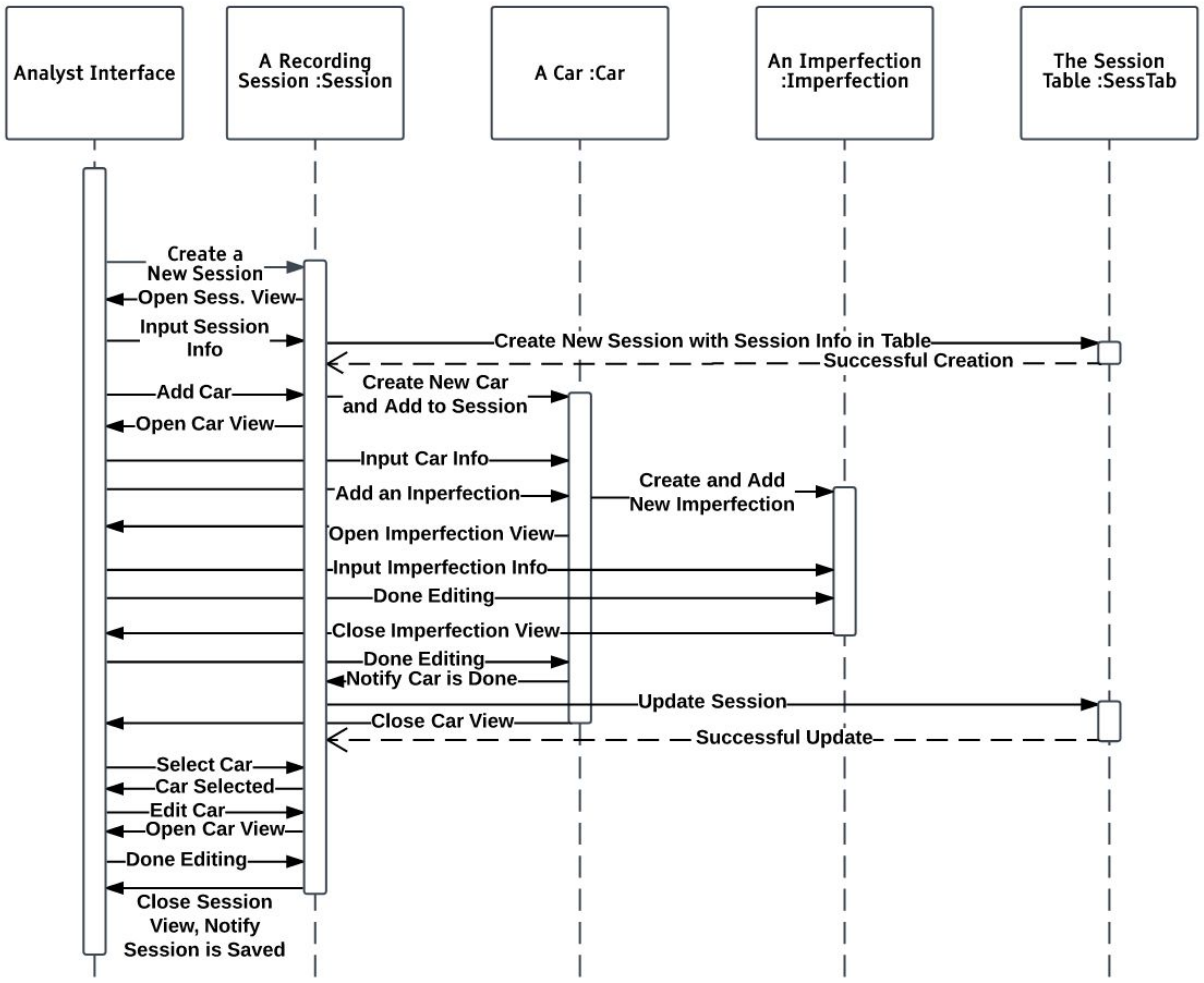


- Inspector – This models the inspector reporting
  - Attributes
    - name:string – This is the name of the inspector
  - Relationships
    - One Inspector can report for many sessions
- Session – This models some amount of time inspecting cars
  - Attributes
    - start:datetime – The start of the session
    - end:datetime - The end of the session
  - Operations
    - add_car(car:Car) - Adds a car to be inspected
    - remove_car(index:int) - Removes a car from inspection
  - Relationships
    - One Session can have many Cars
- Report - Metrics on some amount of sessions
  - Attributes
    - start:datetime – The start date for this report
    - end:datetime – The end date for this report
  - Relationships
    - A report has many sessions that correspond to the start and end date.
- Car – A car being inspected
  - Attributes
    - color:string – The color of the car paint

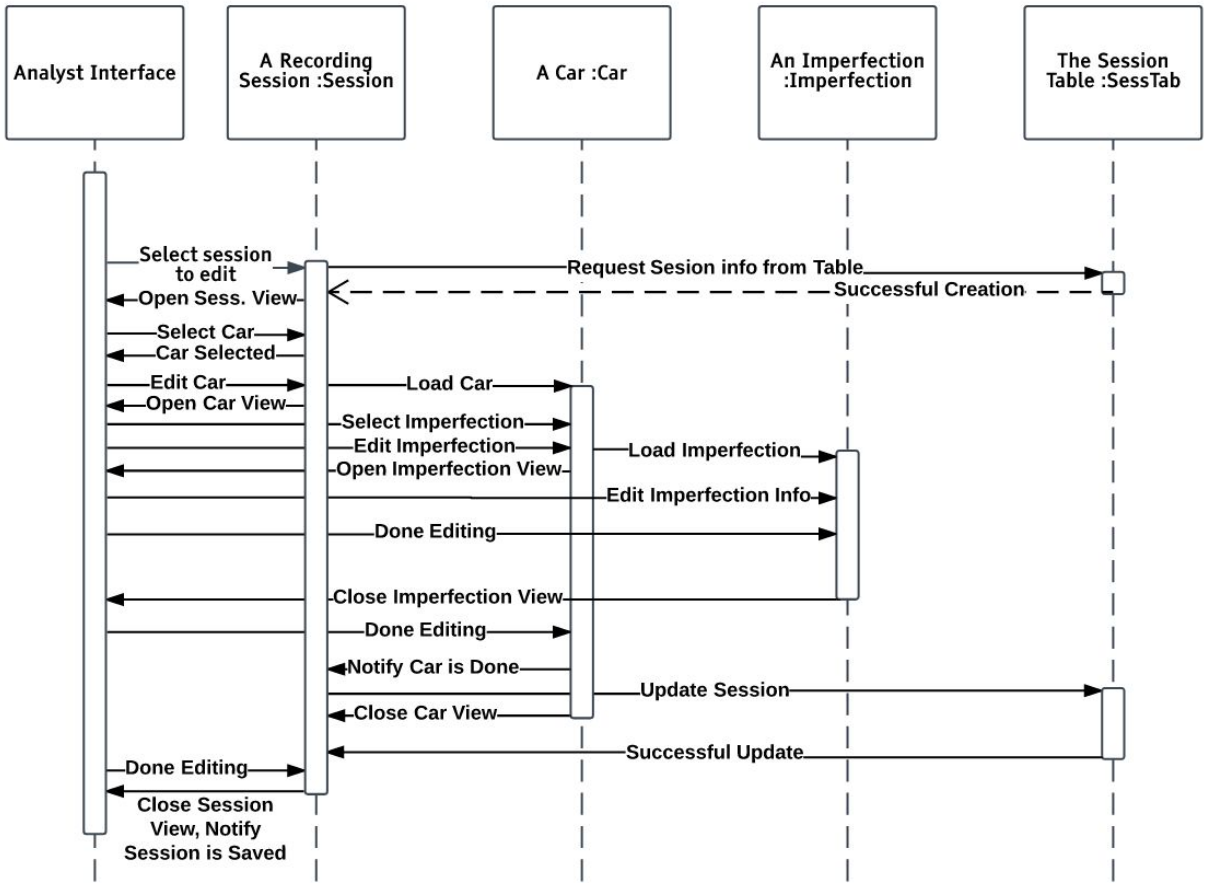
    - Operations
        - add_defect(defect:Defect) – Add a defect to the car
        - remove_defect(index:int) – Remove a defect from the car
    - Relationships
        - A car has multiple regions
        - A car has some number of defects
- Region – A location a car
    - Relationships
        - Many regions are on a car
- Defect – Some abnormality with the car paint
    - Attributes
        - type:string – The type of defect
        - severity:int – The severity rating for the defect
    - Relationships
        - Many defects can exist on a region of a car

Scenarios

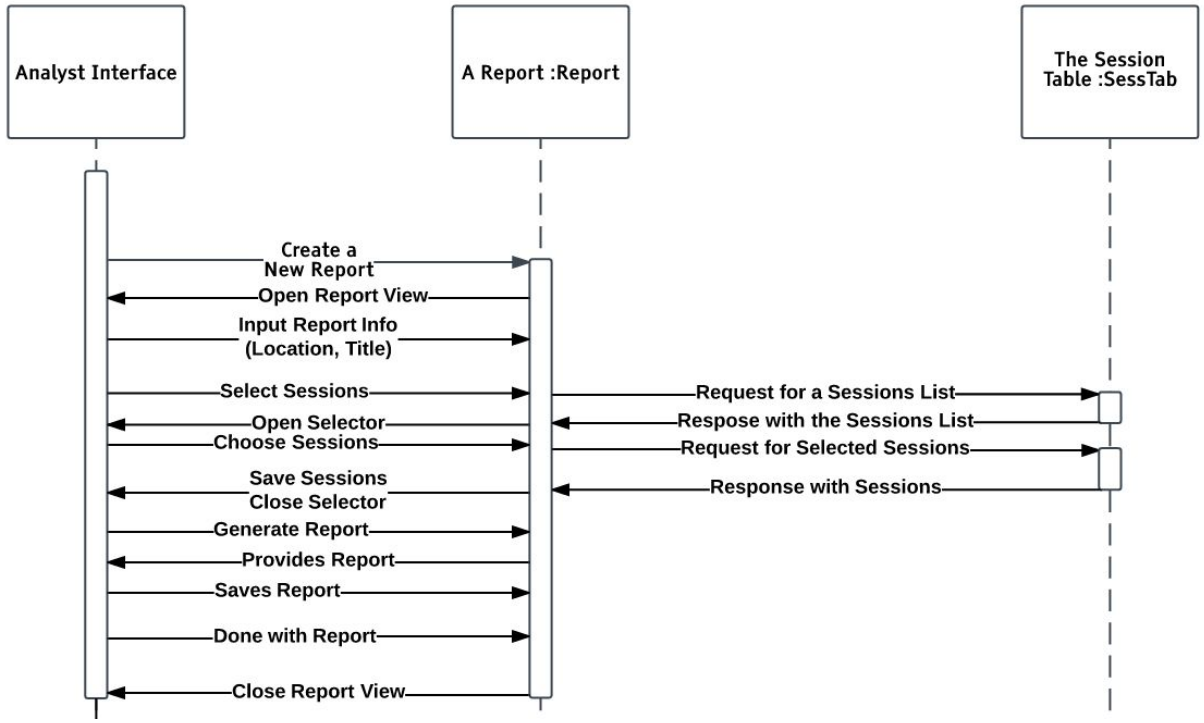
In the scenario depicted below, an Analyst accesses the interface for our product. He then creates a recording session (this is the Create Session use case) and inputs the necessary information. Within this session, he adds a car (the Add Car use case) and records its information and then adds an imperfection (the Add Imperfection use case). He records the imperfection details and closes out of the imperfection and car screens. He closes out of the recording session. Throughout the process, the system saves the session to the database, ensuring that no data will be lost even if the session view is not completed properly.

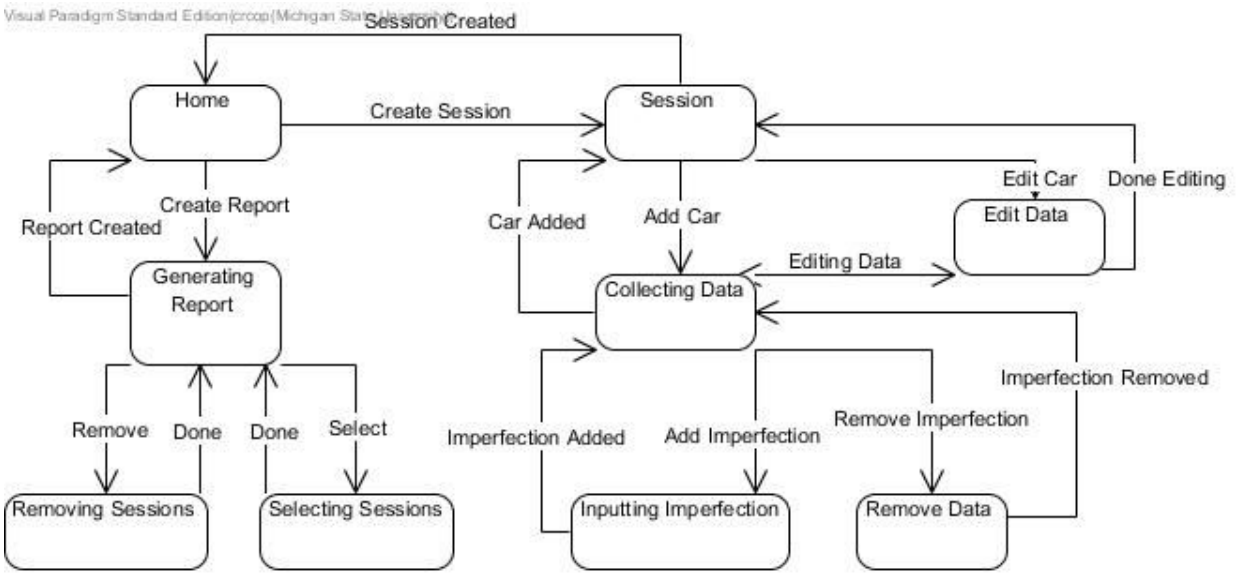


Then, in the second scenario, he decides to edit the imperfection he just made, so he selects the previous session for editing, selects the car he just added and chooses to edit it (the Edit Car use case) and does the same for the imperfection he created (the Edit Imperfection use case). While editing the imperfection, he changes the data he needed to change and closes out of the imperfection and car views.



In the third scenario, diagrammed below, an analyst creates a report using our system (the Generate Report use case). They tell the system they want to create a report, input the information for the report including which plant the report is for and a title of the report. Then they select the imperfection recording sessions that they want to create the report on and generate and save the report. The session table is accessed to get information about sessions for selection and to retrieve the selected sessions themselves.





The state diagram begins in the Home state, and from there you can either generate a report or create a session. In order to generate reports, you must have created multiple sessions to draw off of first. When generating a report you can add or remove as many sessions as you want to generate from. When in the Session state, you can either add or edit a car. Both adding and editing car then execute the same states, but a car must be created first before you can edit it.

5 Prototype

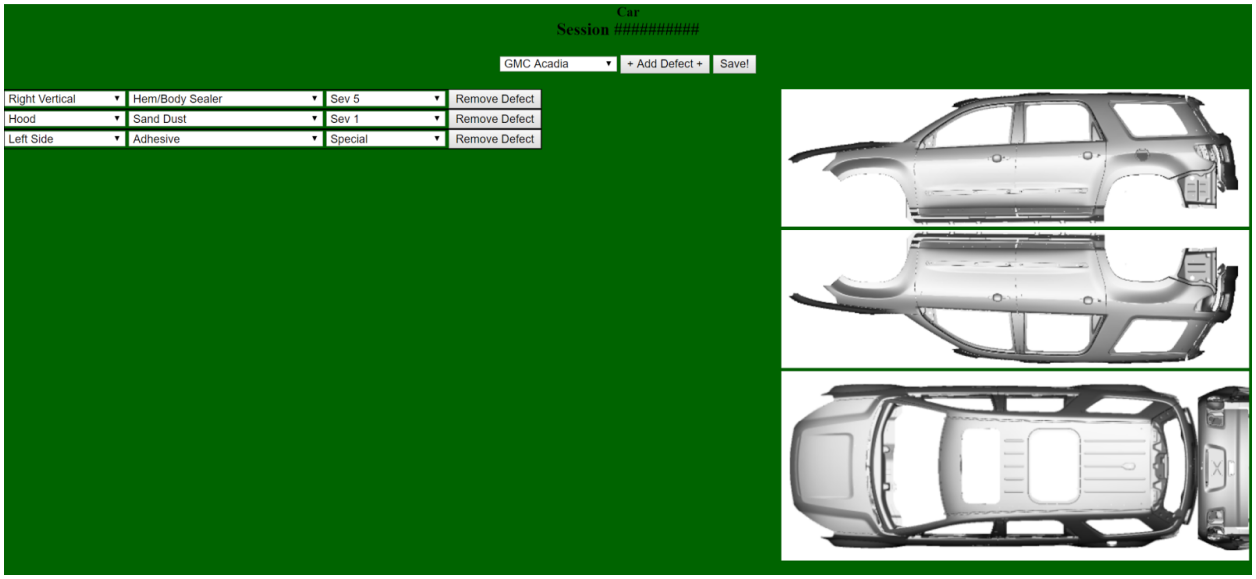
The initial prototype has the UI implemented and allows for navigation between different features of the system. This prototype is not functional and does not store data or generate real reports from data. It does contain example data and displays a basic example report. The second prototype will implement functionality. This means it will store data and allow for editing of data. It will create real reports.

5.1 How to Run Prototype

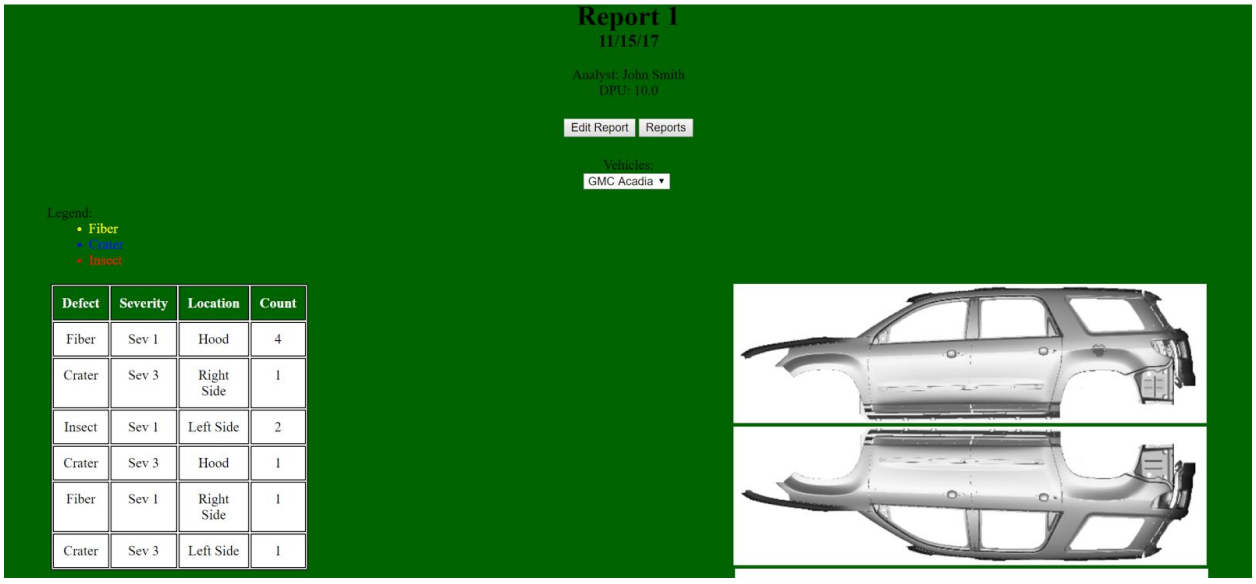
To run the prototype, a user can use any modern web browser such as Google Chrome, Mozilla Firefox, Opera, etc. The url that the user should go to is:
http://www.egr.msu.edu/~joseph62/cse435/
Once the user has navigated to the url, they will click on Prototype under Other Resources. The user will then click on the login button and will be able to use the prototype.

5.2 Sample Scenarios

Analyst logs in to the system.
Analyst then creates a session.
Analyst then adds a Sev 5 Hem/Body Sealer defect for a GMC Acadia (below).
Analyst then adds a Sev 1 Sand Dust defect for the GMC Acadia (below).
Analyst then adds a Special Adhesive defect for the GMC Acadia (below).

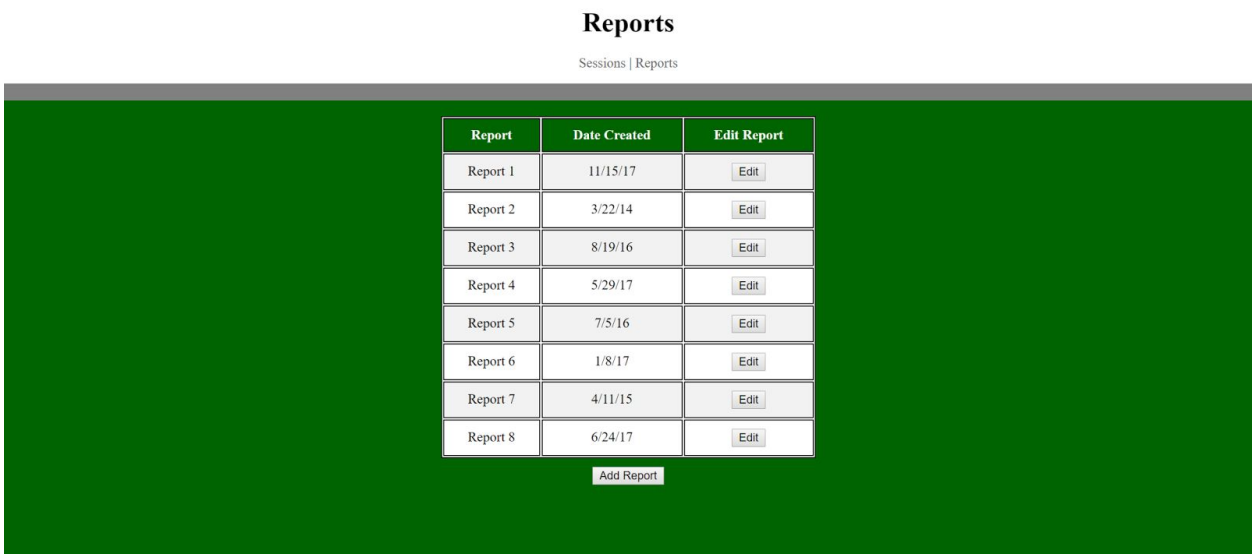


Analyst saves the defects for the GMC Acadia and generates the report.
The report displays information about the defects and visuals (below).



Defect	Severity	Location	Count
Fiber	Sev 1	Hood	4
Crater	Sev 3	Right Side	1
Insect	Sev 1	Left Side	2
Crater	Sev 3	Hood	1
Fiber	Sev 1	Right Side	1
Crater	Sev 3	Left Side	1

Analyst then accesses all prior created reports (below).



6 References

Documents can be obtained from the website cited as [2].

[1] S. Alfa, L. Arent, C. Coppersmith, S. Joseph, and J. Murray, “Requirements Definition,” November 2017.

[2] Joseph, Sean. CSE 435 Group 6. Oct 2017, www.egr.msu.edu/~joseph62/cse435

7 Point of Contact

For further information regarding this document and project, please contact **Prof. James Daly** at Michigan State University (dalyjame at cse.msu.edu). All materials in this document have been sanitized for proprietary data. The students and the instructor gratefully acknowledge the participation of our industrial collaborators.