Software Requirements Specification (SRS)
Project 1

Authors: Nathaniel Finley, Alex Lepird, Jack Mansueti, Caleb Sherman, Hansheng Zhao

Customer: General Motors

Instructor: James Daly

1 Introduction

The Software Requirements Specification (SRS) document is a detailed description of the requirements for General Motors' Report Generation system. The system automatically generates reports; highlighting the paint defects on cars that come from the assembly line. The SRS contains these subsections:

- Introduction
- Overall Description
- Specific Requirements
- Modeling Requirements
- Prototype Overview
- References

The subsections thoroughly describe the Report Generation system. Introduction and Overall Description give an overview of the SRS document as well as an overview of the system. A glossary of terms, context of the current product, a description of the functions the system will perform, and constraints/problems associated with the system are contained in the Introduction and Overall Description. Specific and Modeling Requirements sections specify requirements and use cases, as well as represent scenarios and diagrams related to individual parts and the overall system. Prototype Overview describes what the Prototype will show and what is needed to run the prototype. This section will also provide the URL for the prototype as well as sample scenarios for the system. References will provide the reader with a list of documents referenced in the SRS; each document will contain title, report number, date, publisher, sources from which the reference can be obtained, and an entry for the project website.

1.1 Purpose

The purpose of the SRS document is to provide a detailed description of a software system being created for General Motors. The system will help automate General Motors' process of documenting paint defects in cars coming off the assembly line. The intended audience of the SRS is the authors of the document as well as the customer of the system (General Motors). The SRS document provides context for the system as well as the architecture of the actual system itself.

1.2 Scope

The software system being created is a automated paint defect report generator. The application will adopt the following workflow: GM client analyst enters paint defect data (see section 4: Modeling Requirements for detailed defect data) into the software system. The defect data is placed into a database alongside other recorded defects. GM client analyst attempts to retrieve a defect data report based on a specific time window. The system creates a report based on the requested criteria and returns it to the GM client analyst. The new system implements a method that replaces the previous method which required the analyst to manually document the location, type, and severity of each defect, compile the defects following a certain time window, and generate paper reports.

The system will be a web-based application; this allows for access anytime and anywhere. The GM client analyst will be required to provide their GM employee credentials before using any feature on the system.

Note that the system will not automate the information gathering process. The original project specification (9) from the customer states "The client would like to automate the system to eliminate the paper diagrams and automatically generate the desired reports from the entered data." The system will do this, but will not contain additional features.

1.3 Definitions, acronyms, and abbreviations

Due to an extensive number of definitions, refer to the Appendix in section 7.

1.4 Organization

The remaining sections of the SRS contain a detailed description of the system, specific and modeling requirements, and class, state, data dictionary, and use case diagrams.

The remaining organizational structure of the SRS contains these subsections:

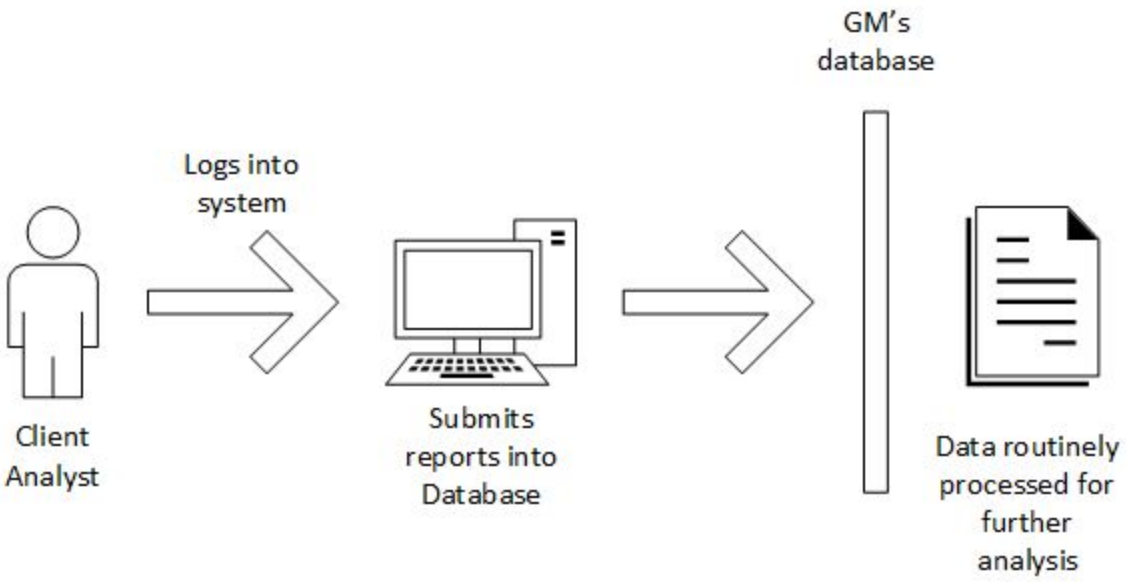
- Overall Description
- Specific Requirements
- Modeling Requirements
- Prototype Overview
- References
- Appendix
- Point of Contact

2 Overall Description

- The following sections will cover the internal and external factors that play an important role in the operation of the product. The sections will also give a better understanding of what it is the product is meant to do and not do.

2.1 Product Perspective

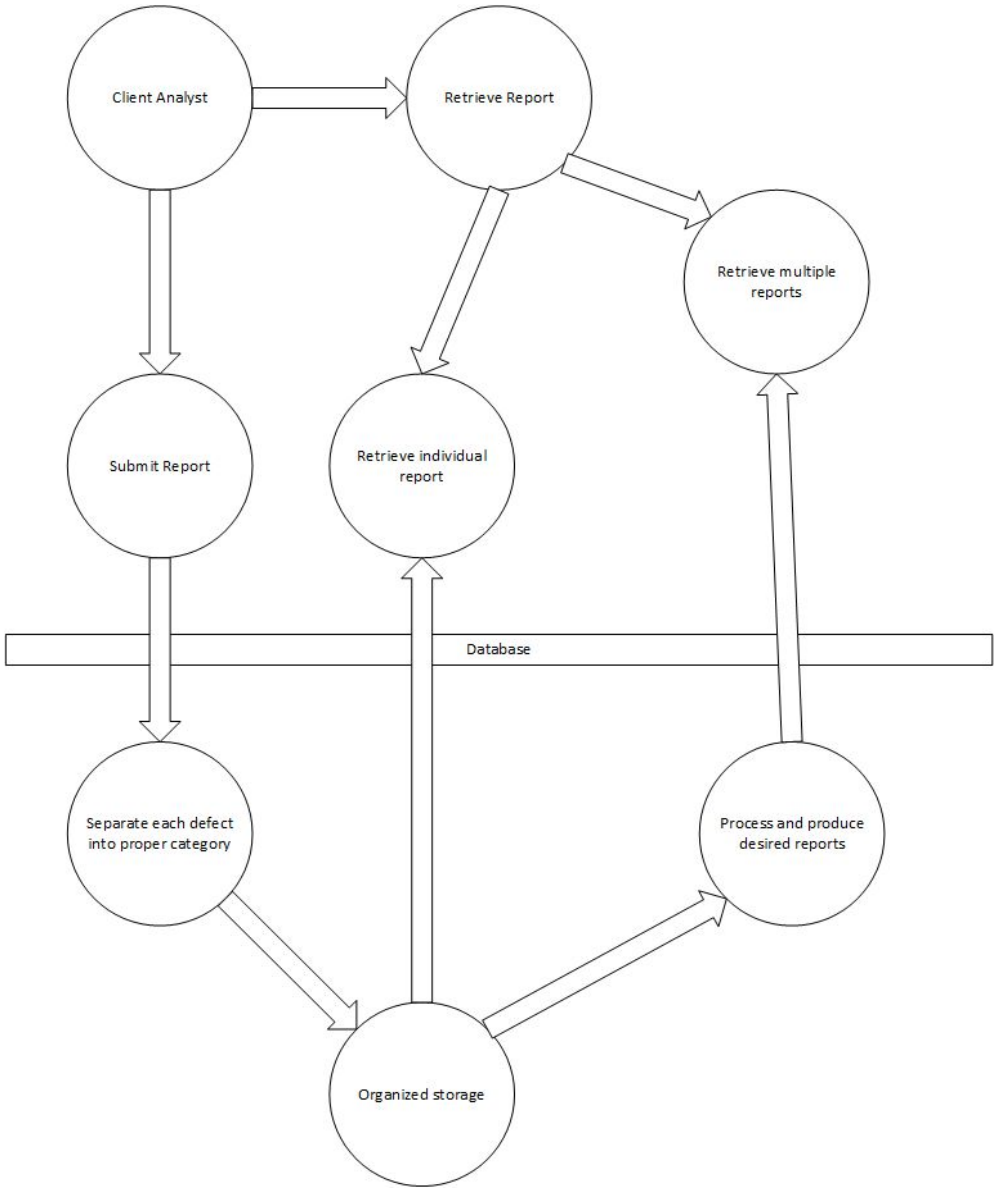
- Currently in automotive manufacturing, a client analyst will inspect and review certain vehicles that come down an assembly line based on certain checkpoints. During these inspections, the analyst will mark down on a sheet of paper the type of defect that was detected using a certain color coding system. He'll indicate where the defect was located and how large it was. This product is geared to reduce the usage of paper as well as provide an automated way to generate the desired reports from any entered from the three GM plants: Lansing Delta Township, Lansing Grand River, and Lake Orion Assembly.
- This product will access a database that will automatically process and collate requested reports defined by the user. This database will be shared among the three different GM plants to provide better reports.



- Due to the system being web-based, recording the defects must being done in an office setting as opposed to on-site. The application also is not designed for mobile use, making a desktop computer or laptop the most efficient way to record defects.
- Some other constraints would be that the system is not customizable. The user must use the system as designed. There also may be memory constraints, as the system is only meant to permanently hold all of the data. However, the data can be stored locally as a text document.

2.2 Product Functions

- This product consists of four major functions: an interface that allows the user to enter and submit reports of defects, a function to retrieve individual reports, a function to retrieve a specified range of reports that will automatically produce desired reports, and lastly a function that will parse through the individual reports to provide the user with their requested information.



2.3 User Characteristics

- The expectations that the user should have in order to work on this product includes a very deep understanding of the different types of defects that occur on vehicles. The user will also need to understand the information that is produced from the product. Aside from needing certain credentials to log into the system, the product will be user-friendly enough to simply need a very basic understanding of computers.

2.4 Constraints

- One constraint with this product is ensuring that there's always enough space within the database that contains the reports. This will require monitoring the system which will notify developers when space is almost full.
- Another constraint is not being able to verify if the user enters the intended information into the system. It's possible to provide a pop-up that warns the user to double-check what they entered, but again it's very likely that they could just skip this warning without checking.

2.5 Assumptions and Dependencies

- Due to the fact that the product is a web-based system, we assume that users will have a hardware system that is capable of accessing the internet over a secured network. We also assume that the user's system is always updated with the newest version of the browsers that they use to operate on the product. For security, the user will only access the product at any one of the three GM plants. We assume that GM will handle this by requiring an extra layer of security to prevent access to the product outside of the plants.

2.6 Apportioning of Requirements

- Currently, the analyst is responsible for analyzing the data provided by the product and determining when a specific defect is occurring too much and other similar issues. Future versions/releases of this product will provide assistance with notifying the user when it's noticed a pattern or a common occurrence of a certain defect.

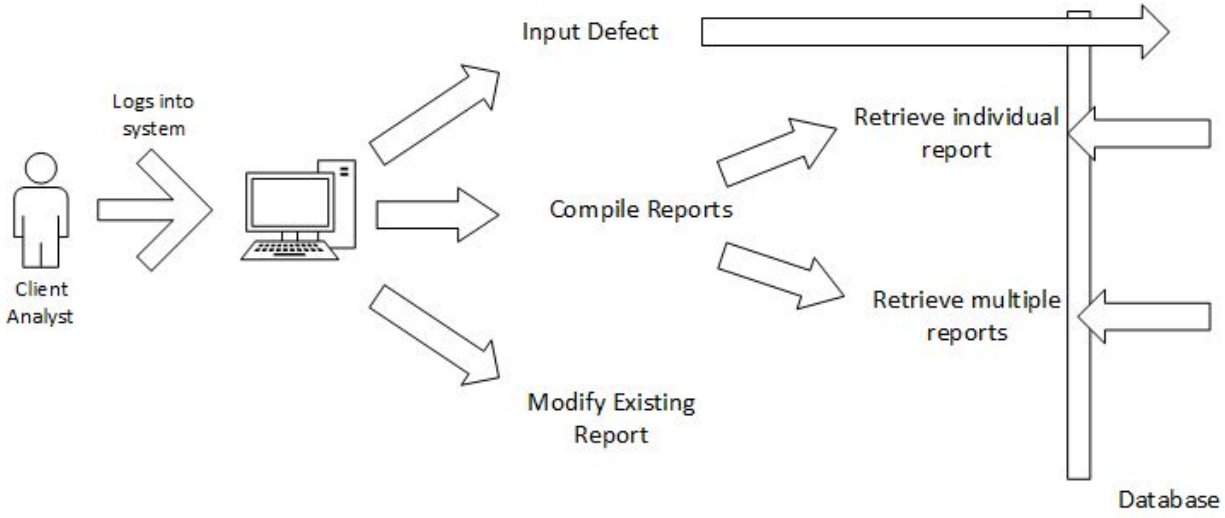
3 Specific Requirements

1. User Session
    - 1.1. Login information
        - 1.1.1. Name
        - 1.1.2. Checkpoint
        - 1.1.3. Shift start and end time
        - 1.1.4. Plant
    - 1.2. Session recovery
2. Defect Analysis & Data Entry
    - 2.1. Interface

2.1.1. Car model selection
2.1.2. Defect location
2.1.2.1. Top side
2.1.2.2. Left side
2.1.2.3. Right Side
2.1.3. Severity of Defect
2.1.3.1. Three severity levels
2.1.3.1.1. Sev 1
2.1.3.1.2. Sev 5
2.1.3.1.3. Sev 10
2.1.4. New vehicle model addition

3. Report Generation
3.1. Defect reports are generated from three checkpoints
3.1.1. Polish Deck
3.1.2. Prime Review
3.1.3. E-Coat
3.2. Daily Reports
3.2.1. Focused on defects per unit
3.3. Weekly Reports
3.3.1. Focused on defects per surface
3.4. Monthly Reports
3.4.1. Focused on trends
3.5. Reports stored on local machine and database
4. System Application
4.1. Software used at multiple GM plants

4 Modeling Requirements



Use Case Name:	Input Defect
Actors:	Client Analyst
Description:	Client will submit a report consisting of 1 or more detected defects. The report will be separated individually into the database to allow easier collation when requested by the client analyst
Type:	function

Use Case Name:	Compile Reports
Actors:	Client Analyst
Description:	When prompted by the client analyst, the system will retrieve the requested amount of reports. If the analyst chooses only one report, he will be given all the necessary information for that specific report. If the analyst chooses to receive multiple reports, the product will gather the requested range of reports and compile it into the desired reports
Type:	function

Use Case Name:	Modify Existing Report
Actors:	Client Analyst
Description:	Using unique information about the defect report, the client analyst can retrieve a specific report to either modify or delete. If choosing to delete, the analyst must explain why before finishing. Afterwards, the report will stay within the system for 3 months before being completely removed, but will not be included in any future reports.
Type:	function

Element Name		**Description**
Factory		A specific GM plant
Attributes		
	Name: string	Name of the plant
	Location: string	Location of the plant

Element Name		**Description**
Defect		The defect for a car
Attributes		
	Type	Type of defect
	Location	Location of defect on car
	Severity	Severity of the defect
	Size	The defect size
	Time	Date and time of defect recorded
	User	User id who submitted report

Element Name		**Description**
Report		Type of report
Attributes		

	Date	Date of the report
	Shift	Shift of the analyst

Element Name		**Description**
DatabaseModifier		Class to interact with the database
Attributes		
	createReportEndpointURL	Endpoint url for creating a defect report
	retrieveReportsURL	Endpoint url for retrieving a group of reports
	retrieveReportURL	Endpoint url for retrieving a single report
Methods		
	createReport	Create a report
	retrieveReportById	Retrieve a single report
	retrieveReports	Retrieve a group of reports

5 Prototype

The prototype will be provided as a web interface, there is no external dependencies for it to run.

The system will be designed as a two-part application: the web interface, and the Python coded backend. The web interface will utilize PHP, JavaScript, HTML and CSS as its building block, where the backend will use Python and a database engine (possibly SQLite/GDBM). The interface will send the requests to the backend, and the backend shall process the input, serialize, persist and generate reports based on persisted dataset.

There will be no plugins except for the required libraries at each end. For the frontend, we will be needing Axios javascript library for initiating HTTP requests; for the backend, we will be needing database drivers for persisting and querying datasets, Falcon (2) and Gunicorn (3) for REST interface.

The frontend interface shall provide the clients with a form to file, and the data collected shall be send to the backend server, the backend server then can persist the data in a database. Depends on the request, the backend server shall retrieve stored data and generate report either periodically or on-demand, and send the compiled data to the web interface for presentation.

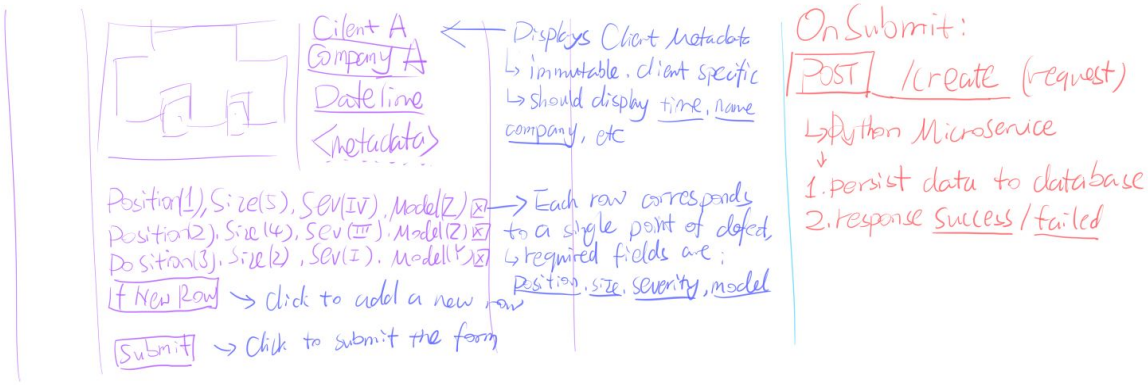
5.1 How to Run Prototype

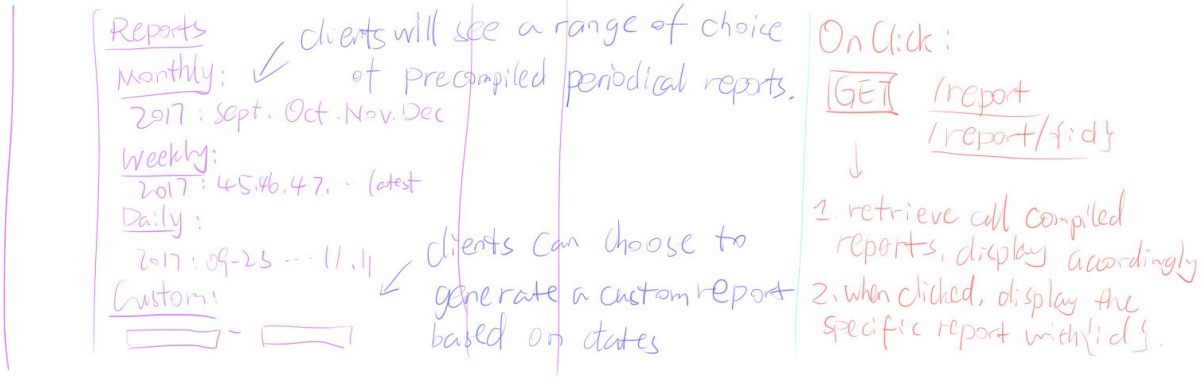
In order to run the prototype, the service provider will need two sets of programs. For the frontend, which was build on-top of PHP (Laravel), SQLite, VueJS, Axios, etc., the service provider needs a host system that at least has PHP interpreter, HTTP server, NodeJS. For this task, the work group has chosen to run the application on a FreeBSD (4) host with PHP version 7.1.11, Nginx (5) version 1.12.2 and NodeJS (6) version 8.9.1. To make the interface more secure, the work group has chosen Let's Encrypt (7) as its SSL certificate vendor for HTTPS support.

For the backend, the service provider will need the latest Python 3 environment. Since the Python platform bundles with SQLite and GDBM support, the work group decided to use these libraries as the persistent database store for the reported data. In order to interface with the frontend, the backend is required to provide a REST API, and the work group has chosen Falcon library and Gunicorn library for the task.

The clients, however, only need the latest browser to use the prototype, simply direct the browser to https://tmp.ink/prototype, login with a valid account, and then it is ready to use. The web application is designed to fit for multiple platform, such flexibility allowed the clients to run the application on a cell phone, a tablet, a laptop or a desktop.

5.2 Sample Scenarios





Client A, directs his/her browser to the application, fills the form above. Each row shall contain these fields: position, size, severity and model, in this example, the user filled three rows:

position (1), size (5), severity (IV), model (Z) [X]

position (2), size (4), severity (III), model (Z) [X]

position (3), size (2), severity (I), model (Y) [X]

After filling the data, Client A then submits the form. The backend then handles the submitted form, serialize the data and saves them in the database. The saved metadata contains: user_id, company_id, created_at and report_id, then for each row, a record was created that contains the position, size, severity and model, then these data will be saved permanently in the database.

Then Client A goes to the report section, and sees that there are pre-compiled reports grouped by months, weeks, and days, if the user clicks any one of the reports, the browser shall show the compiled report in a new window. However, a client is given a choice to generate a custom report based on timeline, once the client selected desired time span, the backend server will generate a new report and directs the client to this report.

6 References

1. D. Thakore and S. Biswas, "Routing with Persistent Link Modeling in Intermittently Connected Wireless Networks," Proceedings of IEEE Military Communication, Atlantic City, October 2005.
2. Falcon Contributors. "The Falcon Web Framework" https://falcon.readthedocs.io/en/stable
3. Gunicorn Contributors. Gunicorn - Python WSGI HTTP Server for UNIX" http://gunicorn.org
4. The FreeBSD Foundation. "The FreeBSD Project" https://www.freebsd.org
5. NGINX Inc. "NGINX | High Performance Load Balancer, Web Server, & Reverse Proxy" https://www.nginx.org
6. Node.js Foundation. "Node.js" https://www.nodejs.org
7. Internet Security Research Group. "Let's Encrypt - Free Certificates" https://letsencrypt.org/
8. Study Group 4. "Study Group 4", https://tmp.ink
9. Software Engineering CSE435. "Automotive Paint Defect Analysis." 10 October 2017, www.cse.msu.edu/~cse435/Projects/F2017/Project-Description.pdf

7 Appendix

Term	Definition
Client Analyst	Employee at GMC that examines vehicles for flaws in the paint and finish and creates paper diagrams.
DB	Database
DPU	Defects per unit
GM	General Motors
GMC	General Motors Company
Sev	Severity of defect
Point Defect	Point defect is a paint defect on a specific location of the car (noted by a point on the car diagram)
Software Requirements Specification (SRS)	A document that completely describes all of the functions of a proposed system and the constraints under which it must operate.
SQL	Structured Query Language Used for database management.
Frontend	A web application's web page that the user

	can interact with.
Backend	The server process that handles user requests and generate response accordingly.
PHP	PHP: Hypertext Preprocessor, a popular language for building web applications.
SQLite	An embedded database engine that provides SQL language interface and database functionalities.
GDBM	GNU DBM implementation, a database engine for efficiently storing arbitrary key value pair.
JavaScript	A programming language that can run in the browser or on the server.
Python	A dynamic language that is suitable for building prototypes and production software.
SSL	Secure Socket Layer, a protocol for secure and encrypted browsing.
HTTPS	HTTP protocol that utilize SSL for security.
FreeBSD	A UNIX-like system that is both stable and secure.
REST	Representational state transfer, a method for communications between computer systems.
NodeJS	An efficient JavaScript interpreter for running JavaScript on servers.
API	Application Programming Interface, the interface that applications provide for developers to use to build other applications.
URL	Universal Resource Locator, a link that uniquely identifies a resource on the internet.

8 Point of Contact

For further information regarding this document and project, please contact **Prof. James Daly** at Michigan State University (daly 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.