Software Requirements Specification

for

Hump Yard Scheduling

Version 1.0 approved

Prepared by Adam Halley

Group: Thomas Anthone, Anna Roo, Chris Recinos, Adam Halley

November 7th, 2022

Table of Contents

Revision History

Name	Date	Reason For Changes	Version

1. Introduction

1.1 Purpose

The Yard Schedule Application v1_0_0 is designed to help yard masters working for freight railroads better track yard occupancy. The specific use case of version 1.0.0 is hump yard operations. Ideally, with this tool's help, the individuals will be able to better track the location of cars for building trains for departure routes. Moving forward we will want to have later versions cover all yard operations outside of only humps.

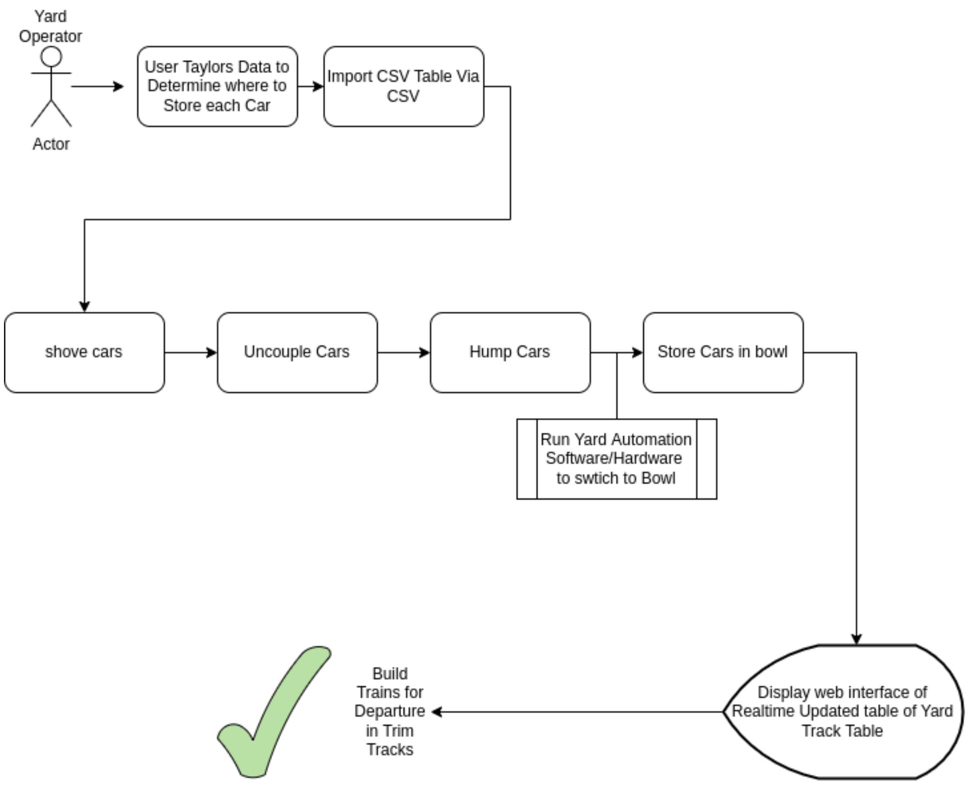
Yard Operator
Actor
User Taylors Data to Determine where to Store each Car
Import CSV Table Via CSV
shove cars
Uncouple Cars
Hump Cars
Store Cars in bowl
Run Yard Automation Software/Hardware to swtich to Bowl
Display web interface of Realtime Updated table of Yard Track Table
Build Trains for Departure in Trim Tracks

Figure 1User workflow diagram for Yard Scheduler application

1.2 Document Conventions

Arial italic font was used throughout the document. Sections are separated by bold face and signify each section of the document. Any references from outside the scope of the author will be hyperlinked and referenced in project documentation.

1.3 Intended Audience and Reading Suggestions

This document is intended for developers who will provide continual improvement for this product. Project managers who will be working closely with individual rail roads for integration and enhancements depending on customer needs. The marketing teams will be pitching the product to all potential customers. And for the User and Testers as a guide to functionality that should meet the minimum compliance outlined in the following text.

1.4 Product Scope

The Yard Scheduling application v1_0_0 will provide means to track car movement more easily in yard from the yard master's perspective. As the rail industry strives to improve overall efficiency and combat supply chain planning issues in large intermodal and yard exchange areas automation to these challenges are sought after. Helping the yards master's decision making and visualization of yard overflow allows for more dynamic trim work as scheduled trains depart the yard for routing.

2. Overall Description

2.1 Product Perspective

Yard Scheduler v1_0_0 is a member of the yard automation family at major freight railroads. While most tools in those, such as Union Pacific's yard automation tool, focus on hardware automation within the yards, the scheduler is designed to help managers plan for rolling stock in yards and visualize capacity.

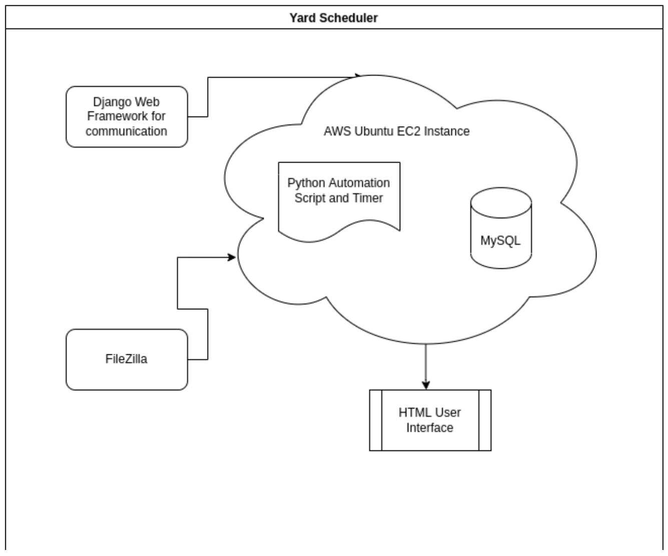
Yard Scheduler
Django Web Framework for communication
AWS Ubuntu EC2 Instance
Python Automation Script and Timer
MySQL
FileZilla
HTML User Interface

Figure 2 High level overview of product workflow for Yard Scheduler application

2.2 Product Functions

Product Functions of the Yard Scheduler Include (See Appendix Analysis Models):

- *Requesting data from user*
- *Importing data into table style database*
- *processing the information to determine which car is going to each track*
- *start a time for the actual hump operation*
- *log track capacity*
- *ensure current tack is not full before switching to a new track*

2.3 User Classes and Characteristics

- *Developer:* *Testing / modification of software features and services*
- *IT / Administrative:* *Granting user privileges, overriding / stopping automation, data modification / updates*
- *Operational:* *Uploading car schedule, emergency / service halts (Most Important user)*
- *Analyst / Executive:* *Data Analysis / Reporting / Tracking*

2.4 Operating Environment

The Yard Scheduler will be running on an Ubuntu AWS (Amazon Web Services) EC 2 instance. The framework platform being used will be Python Django which will be communicating with Apache Tomcat. All the data will be stored on the cloud (AWS EC2 Instance) via a MySQL database.

2.5 Design and Implementation Constraints

Developers may experience design and implementation constraints as follows:

- *Network*
  - *Yards can be in remote locations with less-than-ideal connection speeds*
  - *Some Rail companies have servers that are centralized so that the software will be running in Livonia Louisiana but be communicating with a server in Portland Oregon*
- *Hardware/infrastructure*
  - *For some company's hardware may be limited to what is available*
  - *Smaller rail companies may not have cloud infrastructure readily available so operation will need to localize*
- *Federal regulations*
  - *All moving equipment is regulated by the United States government (Federal Railroad Administration - FRA, Department of Transportation - DOT). Rules and standards must be followed in all design and implementation decisions as to not invoke penalties from these regulatory departments.*

2.6 User Documentation

The user manual will be given digital and hard copies to the Yard Scheduler application customers. A help document will also be available on the cloud server and made available through the HTML portal along with this SRS document.

2.7 Assumptions and Dependencies

Assumption that all yard operations and proximity switches are automated and working properly any hardware issues that occur during the implementation will be the responsibility of the Yard Scheduling software. The Yard Scheduling application also assumes that all the entry cars are in order such an order that will not disrupt distribution between yard tracks, i.e., the train is built in such a way that will allow variability that a car can be lost between trains. All bugs and defects will be kept confidential and will only be subject to review by Anna, Thomas, Chris, and Adam Inc.

3. External Interface Requirements

3.1 User Interfaces

All user interfaces for the Yard Scheduler application will be html web pages using standard buttons and tables. Logo from Union Pacific will be supplied but the icons can be switched to any new customers that may buy the software. The color pallet will match that of PS Technology which will be Orange(#FF4D12) on Black (#222) with white text and grey (#333) accents for the navigation bar. Some accents may change based on choice as noted in the list below.

3.2 Hardware Interfaces

This application will be able to run on any Mac OS, Linux, and Windows desktop device. Storage and memory can be upgraded from the smallest requirements in both the local and the cloud for better overall performance and ease of use. Depending on how many yards are planned to be stored, it will decide how often storage issues may arise or need to be stored locally.

Network connectivity is needed to run the Yard Scheduler application. It is recommended to run a 1Gib/second wired connection during runtime.

Minimum Cloud Requirements
CPU – Intel(R) Xeon(R) CPU E5-2676 v3 @ 2.4ghz
System memory- 1Gib System memory
Physical memory - 1Gib DIMM RAM
Storage – 8 Gib of physical storage

Minimum Local Requirements
CPU – Any gen Intel 5 @ 2.4ghz or better
System memory- 8Gib System memory

Physical Memory - 8Gib DIMM RAM
Storage – 500 Gib of physical storage (or the ability to store this information on a client server or more space on the cloud)

3.3 Software Interfaces

Software interfaces for the Yard Scheduler include python, Django, html, CSS, JavaScript, Ubuntu, MySQL, AWS, and Apache.

3.4 Communications Interfaces

The Yard Scheduler application is built in the Django framework, so a network connection is required to connect to the AWS EC 2instance. An FTP connection can be made to the EC2 instance to access all the input data, database (.db) files, and python scripts used for the hump yard operation.

4. System Features

4.1.1 Description and Priority

Yard Scheduler Django Web Project – Critical – Framework to be used for the Yard Scheduler project.

Html Portal – Critical – User web interface that will be the front end of the Yard Scheduler project.

Hump Timer – Critical – Script to run all the timers that will run the yard automation software and ensure there are no/collisions or overlaps in car movement.

Humper Operation Script – Critical – Script that will read user input, read to the MySQL database, run the hump timer, and update the MySQL database.

AWS Instance – Critical – EC 2 instance that will be used to maintain data workflow and run the Yard Scheduler's server.

MySQL – Moderate – Maintain MySQL database with current track records as per customer needs

Import csv to table – Moderate – Take user input data and import it into trains table for cars and their respective tracks.

Apache Tomcat– Moderate – set up Apache web server for networking between Django deployment and AWS EC2 Instance

4.1.2 Stimulus/Response Sequences

1. *User determines which cars should go to which track and modifies the csv to import*
2. *The user can then see the updated track capacity which will be 100% initial import*
3. *Once the user begins shoving cars over the hump the appropriate capacities will be updated*
4. *Of the cars have been humped A list of out sorted cars will be provided*
5. *The Yard expert can then start building the trains on the trim for departure as needed.*

4.1.3 Functional Requirements

REQ-1: Establish connection to the cloud via Django
REQ-2: Take User Input to build yard table for movement scheduling. Currently only allow a .csv file to be added. Otherwise, request the user transform the data to a usable format.
REQ-3: Establish connection to MySQL database, if cannot establish contact database administrator (DBA)
REQ-4: Store yard table meaningfully that can be accessed via the cloud or SFTP, if connection cannot be made contact the DBA or the information technology call center.
REQ-5: Read yard table to hump operation python script and parse data by column. If any of the data in the table is Mal formatted inform the user to reinput that data.
REQ-6: Begin hump timer to move cars up the up and down to the bowl tracks for staging. Staging order is based on user input.
REQ-7: Determine if destination yard track is full, if so, move to next empty track if all tracks have at least one care stack cars in yard next available yard track.
REQ-8: Run Yard Automation software to move cars through tracks and reset switches once car is past last switch.
REQ-9: Update Track Compacity in MySQL yard table.
REQ-10: Hump Next car and repeat REQ 6-9

5. Other Nonfunctional Requirements

5.1 Performance Requirements

The Yard Scheduling tool is intended for everyday use. Use during network or operational outages will not be tested and the likeliness of success in any other circumstance is unknown. Using this application in such a situation should be under the direct supervision of the telecommunications officer and a director of information technology. The developers of this tool will not be responsible for any modifications to or changes in network communications that would affect the performance of the Yard Scheduler.

5.2 Safety Requirements

Length of track must be maintained, and occupancy must be tested thoroughly before input data is entered into the Yard Scheduler. Over filling a bowl track can lead to devastating collision that may result in derailments or even fatal injury. The data entry at this point should be double checked by a manager until tests can be added to make sure that the length of track never exceeds the previous entry and as per scheduled maintenance work. All operations related to or included the Yard Scheduler must be within the guidelines of the Federal Railroad Administration, any deterrent is punishable by law and may result in a financial penalty

5.3 Security Requirements

All data contained within the Yard Scheduler's database is sensitive and proprietary to its customers. This data should not be released to the public unless so specified in the governing contract. Amazon AWS security should be provided in such a way that only those with primary keys can access the databases and extract any data. All interpolated points, car sizes, and track information should not be used outside of the Yard Scheduler Application.

5.4 Software Quality Attributes

Access will be granted to customers for maintenance and quality assurance testing, or this can be added as a maintenance subscription to the development team. The Database is completely expandable within the hardware specifications of the Amazon AWS instance. The data provided in the MySQL tables can be accessed, saved off locally, and used for analytics predictive modeling, and usage descriptions for all yard inventoried.

5.5 Business Rules

Yard Operators and all who they directly report to will have access to the components and the ability to access the software. All operation management will also have access to the application in case of a leave of absence by the yard operator, or some externality that is not foreseen the following onsite personnel will have access:

- *Superintendent*
- *Director of Train Operations*
- *Director of Yard Operations*
- *Manager of Terminal Operations*
- *Manger of Yard Operations*
- *Yard Operators*

6. Other Requirements

Appendix A: Glossary

Hump – a yard-controlled track with a grade greater than 2 degrees.

Lap switch – Three ways switch used to control the movement of car into the bowl track

Bowl track – series of tracks controlled by lap switch that are used to store cars before departure served by hump.

Hump Yard – A specialty yard used to hump cars into bowl tracks.

FRA – Federal Railroad Administration

DOT – Department of Transportation Regulatory Commitee

EC2 – Elastic Compute Cloud

AWS – Amazon Web Services (cloud computing services)

MySQL – Database administration system for housing data

Django – Python based web framework

Appendix B: Analysis Models

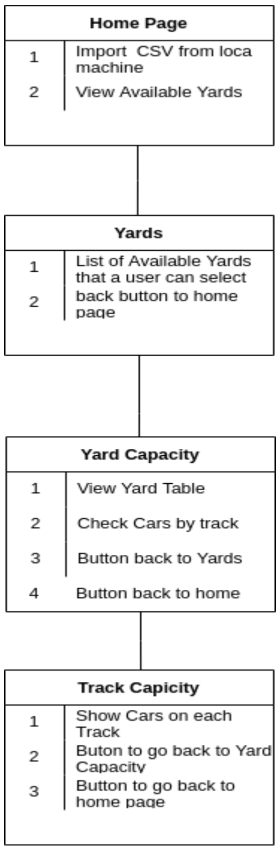
Home Page
1 Import CSV from loca machine
2 View Available Yards
Yards
1 List of Available Yards that a user can select
2 back button to home page
Yard Capacity
1 View Yard Table
2 Check Cars by track
3 Button back to Yards
4 Button back to home
Track Capicity
1 Show Cars on each Track
2 Buton to go back to Yard Capacity
3 Button to go back to home page

Figure 3HTML workflow diagram for user interface

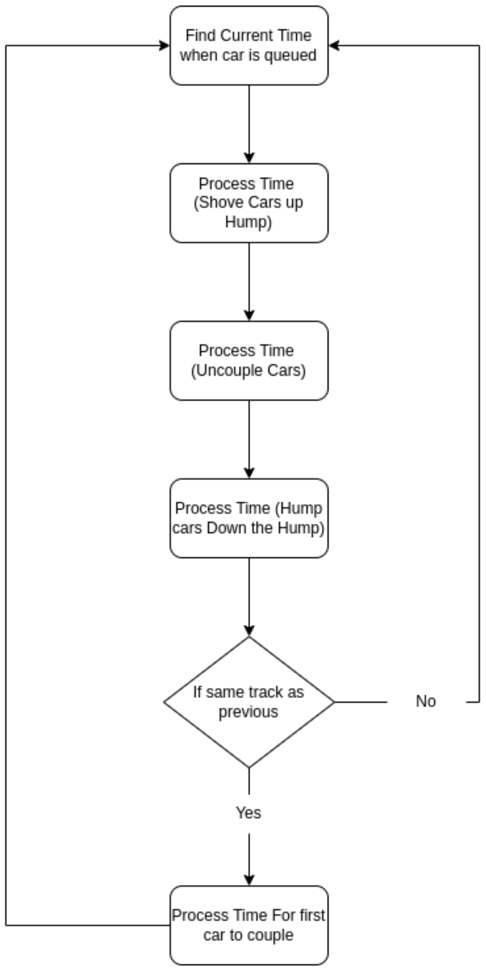
Find Current Time when car is queued
Process Time (Shove Cars up Hump)
Process Time (Uncouple Cars)
Process Time (Hump cars Down the Hump)
If same track as previous
No
Yes
Process Time For first car to couple

Figure 4 Workflow diagram for hump timer

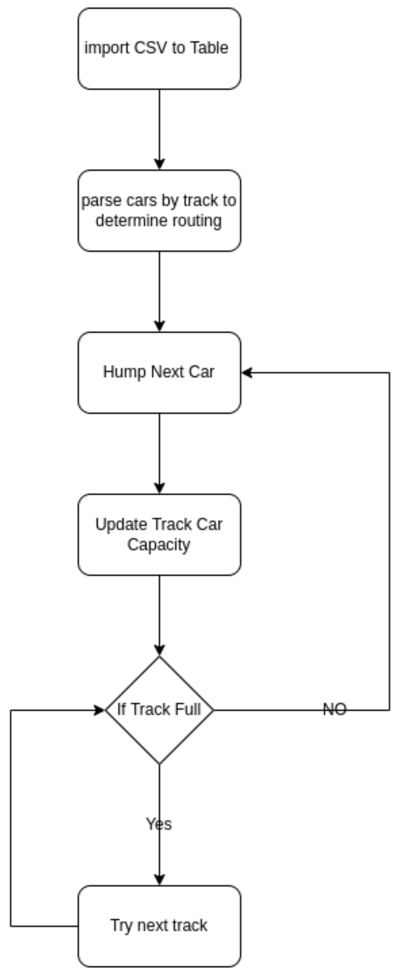
import CSV to Table
parse cars by track to determine routing
Hump Next Car
Update Track Car Capacity
If Track Full
NO
Yes
Try next track

Figure 5 Workflow diagram for hump yard operation and database management