CS CAPSTONE PROJECT ARCHIVE DOCUMENTATION

MAY 29, 2020

AUTOMATE THE SETTINGS THAT CONTROL A MILLION-DOLLAR PRINTING PRESS

PREPARED FOR

HP, INC

PIETER VAN ZEE

PREPARED BY

GROUP62

PROPRIETORS OF THE PRESS

KUAN-YU LAI
COLE JONES

Abstract

During the 2019-2020 school year, our Capstone group created a web application for HP that uses a rule-based decision engine to provide users a starting point for determining the optimal settings for an industrial printing press for a given print job. This document is a compilation of all of the documents that were created during the year, as well as blog posts and conclusions from team members, and feedback from peers from the design and code reviews. The web application we created is planned to be deployed on an HP site so that users around the world may use it.

CONTENTS

LIST OF FIGURES

1 Forward

Release 1.0 of this project is functionally complete. The only thing that still needs to be implemented is the logging feature in both of the back-end servers. The J-easy Rules Engine already has a logging feature, but it still needs to be adjusted so it can be output to a file. As for the SJA Engine, the logging feature is almost done, except it needs to be saved into a log file. A good place to start would be to run some unit tests, as we were unable to do so in the allotted time. And for the website, adding functionality shouldn't be too difficult as each page of the site is a separate component.

2 Introduction to Project

2.1 Who Requested It?

This project was requested by our clients at HP's Corvallis campus, Pieter van Zee and Ronald Tippets.

2.2 Why Was It Requested?

Our clients were trying to build a system that can eventually allow any kind of people to operate their industrial printing presses. The ultimate goal is to build a fully automatic printing press that the user only needs to press a start button and the printer will handle the rest. In order to achieve the goal, our client will have to generate the optimal settings for each printing job first. They tried to build this system with machine learning last year but it wasn't working as they expected. As a result, they wanted to try building the system again without using machine learning this year.

2.3 What Is Its Importance?

None of the companies in the industry have successfully built a system that can automate (or even semi-automate) the printing presses. Therefore, this system will be the first automated system in the industry. Not only can HP keep its lead beyond the competitors but also save a lot of money for their clients. Since operating the printing presses currently requires an expert that takes about 7 months to train, using a computer to operate will reduce the cost significantly. Also, HP continuously researches the settings so the optimal settings might change over time. It's easier to update the computer program instead of training people with new research results. The computer can avoid human errors as well.

2.4 Who Was/Were Your Client(s)?

Pieter van Zee and Ronald Tippets of HP's Corvallis campus.

2.5 Who Are the Members of Your Team?

Cole Jones and Kuan-Yu Lai.

2.6 What Were Their Roles?

Cole worked almost exclusively on the front-end website and Kuan-Yu Lai worked almost exclusively on the back-end Smart Job Advisor engine and Rules engine. There was a fair amount of crossover, and we each designed our components with the other in mind. We worked together often to ensure that the output from the front-end matched the expected input on the back-end and vice versa.

2.7 What Was the Role of the Clients?

Our clients worked on compiling rules and rulesets by communicating with experts at HP. They also helped us find the appropriate technologies to use when we were having trouble. If we had problem during the development, they would provide suggestions or find other experts in HP to help us.

2.8 How Did the Changes in Spring Term Affect Your Deliverables?

Since all of our work was already being done remotely (with Kuan-Yu Lai in Taiwan for the majority of Spring term) including meetings with our clients, the changes in Spring term had no effect on our ability to work on the project.

2.9 How Do You Recommend the Next Team Use This Final Documentation to Pick Up Where You Left Off?

Use our setup guides to get all of the front-end and back-end components deployed and to familiarize yourself with the code base. Once the project is set up and working, new code may be pushed to github and deployed on the hosted site simply by pulling the updated repo, without having to go through all the trouble of rehosting.

CS CAPSTONE REQUIREMENTS DOCUMENT
OCTOBER 25, 2019

AUTOMATE THE SETTINGS THAT CONTROL A MILLION-DOLLAR PRINTING PRESS

PREPARED FOR

HP, INC

PIETER VAN ZEE

PREPARED BY

GROUP62

PROPRIETORS OF THE PRESS

KUAN-YU LAI
COLE JONES

Abstract

This document describes the details of the automated setting select project that will control the million-dollar presses from HP. It contains subjects like timeline throughout the term, use cases, etc. . . The main purpose of this document is for people to know what happened during the development process and give the general idea of the final product.

3.1 Change Table

Section	Original	Change
3.2 - Overview	The engine will run alongside pre-existing internal tools in a queue.	The engine is mostly independent, doing its own PDF analysis and holding the job results, not appending them to anything else
3.4.2 - Use of Decision-Making Engine in Pre-Pass Check	Use of Decision-Making Engine in Pre-Press Check	Use of Decision-Making Engine to generate the recommended setting
3.4.4 - Use of Decision-Making Engine Current Operating Task and Progress	Use of Decision-Displaying Current Operating Task and Progress	Operation schedule feature no longer belongs to our application
3.4.5 - Use of GUI to Queue Jobs	Use of GUI to Queue Jobs	Queuing jobs no longer supported by our application
3.4.6 - Use of API Calls to Queue Jobs	Use of API Calls to Queue Jobs	Queuing jobs no longer supported by our application
3.5.1 - Existing Internal Tool	Existing Internal Tool	We ended up using a different PDF evaluation program instead of the one from HP
3.5.5 - Database	Database	Changed to using a filesystem instead of a database

3.2 Overview

The objective of this project is to create a decision-making engine that takes an input of information about the PDF file to be printed and a data file with additional information about the job (like paper type, color profile, etc.), and outputs a selection of settings used to control the printing press. The engine will run alongside pre-existing internal tools to analyze print jobs within a queue and generate a settings profile (or choose a pre-existing settings profile) for each job on its way out of the queue into the printing press. Once the printing job has reached the printing press, it will display to the operator the settings it has generated, along with a justification as to why that setting was picked and if it aligned with the settings that were chosen or imported. The overall goal is to streamline the processing of picking the correct settings for a print job, reducing the amount of interaction between the operator and the printing press.

3.3 Glossary of Terms

Term	Description
Ruleset	A set of rules which the decision-making engine uses to generate optimal job settings. Includes information about best practices, optimal ink coverage
Decision-Making Engine	An engine that allows developers build and run machine reasoning models
GUI	Graphical User Interface. The web application used to interact with the engine
API	A protocol that allows client communicate with the server easily

3.4 Use Cases

3.4.1 Implementation of Decision-Making Engine

Scenario: A decision-making engine will be implemented. This engine will take a set of inputs and produce a set of press settings to append to a print job.

How: A collection of candidate engines will be selected through interviews with machine-reasoning experts at OSU in addition to independent research. These decision-making engines will have multiple representative rulesets implemented for testing. Once the engine most suited to the task is found, it will be used to create the final implementation.

Success: The final implementation of the engine will be chosen. It will have, in addition to a GUI, an API framework that can be used to communicate with the engine outside of the application.

3.4.2 Use of Decision-Making Engine in Pre-Press Check

Scenario: The decision-making engine will be used to check if the settings chosen for a job are appropriate.

How: This use of the engine will be part of the pre-press check, a check that occurs before a job is queued to make sure it is fit for printing. The engine will analyze the info about the PDF and applied settings and determine whether or not the job should be printed, and if the chosen settings are appropriate or not.

Success: The job is checked and a justification for why the settings are appropriate or not is produced. The job will not be sent to a queue.

3.4.3 Use of Decision-Making Engine to Generate Job Settings for Job Ticket

Scenario: The decision-making engine is used to generate optimal job settings from the provided job information. A list of all processed jobs and their outputs is available to the user.

How: The engine is used to process multiple jobs and provide an output that consists of a collection of jobs settings stored in a job ticket in addition to justifications as to why those settings were chosen. The engine is not hooked up to a queue. Instead, the user is able to view a list of completed jobs and their outputs, and can take the generated job settings and save them for later use.

Success: The user will be able to save created optimal job settings in a database, then use them in later jobs that have similar inputs.

3.4.4 Use of Decision-Displaying Current Operating Task and Progress

Scenario: When the user decides to process their task, they will submit the job ticket so the system can schedule the operation.

How: User submits their job ticket, the ticket will be put in the task queue and wait for the press to process the task.

Success: The user will be able to see all the tasks, job ticket, in task queue and view the status of each tasks.

3.4.5 Use of GUI to Queue Jobs

Scenario: User has access to a responsive web-based application and make the printing request for the press.

How: The web-based application will have 3 functions: job acquisition, job processing, and job results. Job acquisition only gets the input from the user, the PDF info and job ticket data. The job processing function allows the user to choose some of the printing settings or a settings profile. The settings are different from those that the operator sees, it will be slightly more complicated, with additional information to select rules that the decision-making engine will use. The

job results present the current working process of each task and the failure reason if the task failed. Also, it provides a justification for the setting, explaining if the chosen setting matches the setting generated by the decision-making engine or not.

Success: Different users will be able to access the same web-based application to queue up jobs or view the queue. The user will not require press operation expertise due to the user-friendly UI.

3.4.6 Use of API Calls to Queue Jobs

Scenario: User can directly call the decision-making engine's API to post PDF info and job ticket data to create a print job.

How: A user will use the decision-making engine's built-in API to make a POST call, providing documents relevant to the print job. Similarly to how the hotfolder works, the engine then takes the provided files as input, generate the appropriate settings, and move the job into the print job queue.

Success: The files have successfully been processed by the engine and placed into the print job queue with relevant settings information attached.

3.4.7 Selection of Rules for Engine

Scenario: A user will be able to specify which ruleset the decision-making engine uses for a particular print job.

How: During the use of the GUI to select appropriate settings for the print job, the user may specify what ruleset the decision-making engine will use. Each ruleset is like a profile that's adapted to fit certain constraints better, such as the type of paper or the printing press being used.

Success: The settings created for a print job will be better-suited for the type of paper and press being used depending on the ruleset used.

3.5 Tools & Applications

3.5.1 Existing Internal Tool

There exists an internal tool within HP that allows users to import a PDF of what they want to be printed and XML file that contains some rules about the print job (such as width, paper type, etc.). The user can then select a myriad of settings that the printing press will use to print the job. We will be piggybacking on this tool, inserting our decision-making engine into the pipeline between this internal tool and the print job queue.

3.5.2 Decision-Making Engine

Our decision-making engine will take the settings chosen from the internal tool (or from an XML file) in addition to the PDF of what to print and generate what it determines to be optimal settings for the job. The settings it chooses are based on its analysis of the PDF info and job ticket data. It analyses the size of the document, the density of ink per page, the use of gradients, color fills, and graphics, and determines the optimal printing speed, press tension, and drying heat/time. It will attach the generated settings to the print job when it adds it to the queue, in addition to supplying a justification for each setting choice and a log file to analysis.

3.5.3 GUI

The GUI will be created for user-friendly interaction with the decision-making engine. It will take a PDF info and job ticket data as input. The site will allow the user to drag and drop files into a web interface. It will also allow users to select which ruleset they want the engine to use.

3.5.4 API

An API will be created for the engine to permit its use without having to interact with the GUI. It will be necessary for other applications to provide input without human interference, permitting the engine to run alongside existing programs in the job creation pipeline.

3.5.5 Database

An external database will exist to hold all of the generated rulesets that the engine will use. It will also hold engine output that is deemed reusable for jobs with similar inputs.

3.6 Gantt Chart

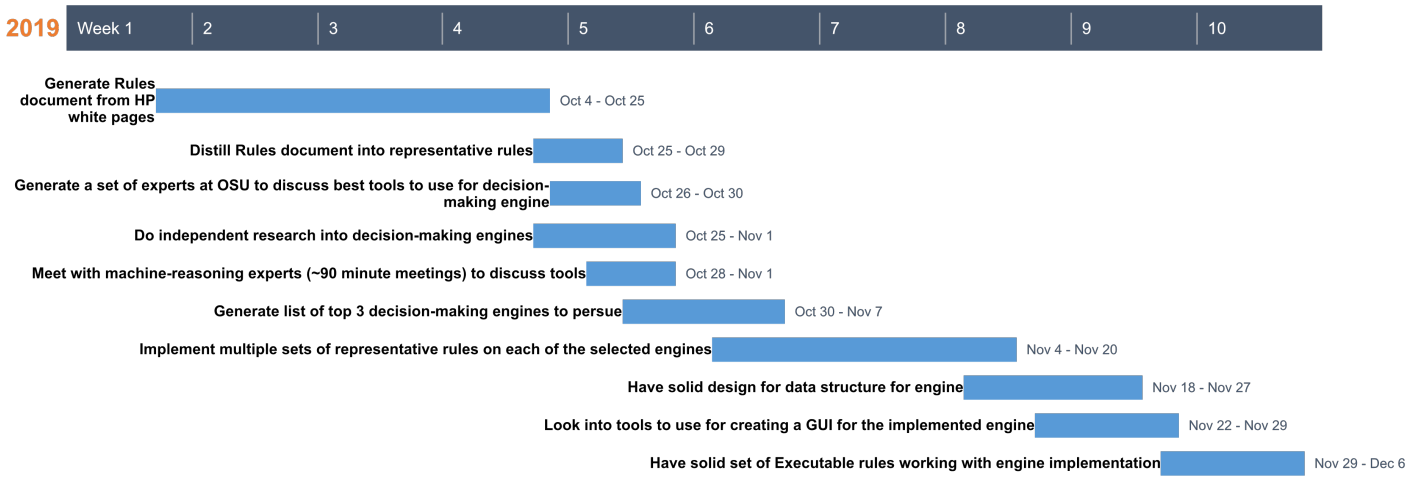


Fig. 1: Requirements Document Gantt Chart

3.6.1 Gantt Chart Analysis

The following paragraph details our execution of the tasks outlined in the gantt chart above, explaining whether or not we were able to complete the tasks in the allotted time.

It took us a bit longer to generate the rules from the white pages that HP gave us than we had originally anticipated. We ended up having to extend that by about half a week. The chunk of tasks between weeks 4 and 5 were a bit too compact, and it ended up taking us quite a bit longer to decided on a rules engine that we wanted to implement. That being said, since it took us to long to find the engine, we were not able to implement multiple rulesets until later in Spring term. The task for having a solid design of the engine's data structure was completed a bit ahead of time since it wasn't as difficult as originally anticipated. We completed the task for finding a GUI tool because we already had experience with ReactJS and Ant Design, making that the obvious choice. We were able to compile the set of executable rules by the end of the term.

CS CAPSTONE DESIGN DOCUMENT
NOVEMBER 23, 2019

AUTOMATE THE SETTINGS THAT CONTROL A MILLION-DOLLAR PRINTING PRESS

PREPARED FOR

HP, INC

PIETER VAN ZEE

PREPARED BY

GROUP62

PROPRIETORS OF THE PRESS

KUAN-YU LAI
COLE JONES

Abstract

This document describes a year plan of the project that builds the assisting feature for the million-dollar printing press from HP. The goal of the project is to create a feature that can recommend the press settings of the giant press based on job inputs uploaded from the user. There are several components involved in building the feature: the rules engine, database, and website. This document describes the design viewpoint from different aspects for those components use in the project, such as what is the purpose of having this component and how does it interact with others components. It also contains the different technologies uses and the design schedule throughout the year.

4.1 Change Table

Section	Original	Change
4.5.2.4 - Interaction	This API will be called by the website to provide user input, so the user does not have to directly interact with the engine	We now have a middle server that facilitates interaction between the rules engine and the website
4.6 - Database	Database	No longer using a database, decided to use a filesystem instead
4.7.2.4 - Interaction	The website will provide three different modes for the user	The website only has one mode which is generating the recommended settings
N/A	N/A	Added the Smart Job Advisor server that is responsible for the interaction between the rules engine and the website. Also, it acts as a file system as well

4.2 Overview

4.2.1 Scope

The objective of this project is to create a rules-based engine that takes an input object containing information about a print job and outputs a selection of settings used to control the printing press. The engine will run alongside existing internal tools, taking input from upstream pre-flighting and handing output downstream into the print job queue. The main components of the project will be the rules engine itself, a database to hold variables and constraints for the engine in addition to its outputs, and a web-based front-end with which the user will interact to provide job inputs.

4.2.2 Purpose

The purpose of this project is to simplify the process of selecting the optimal printing press settings for any given print job on any given press. The purpose of this document is to provide a detailed description of the design of each component of the project in addition to providing a clear path of development.

4.2.3 Description of stakeholders

Our clients are Mr. Pieter Van Zee and Mr. Ronald Tippetts from HP. They are both senior engineers with excellent experience with the printing presses. Their ultimate goal is for us to make a system for the printing press that only require a user to press a single button and then the machine will cover the rest. As the leader and user of the project, they provide feedback and suggestion to us based on our weekly research. At the same time, they are also researching on this topic as well, sharing their knowledge with us during the weekly meetings.

4.3 Glossary of Terms

Term	Description
Ruleset	A set of rules which the decision-making engine uses to generate optimal job settings. Includes information about best practices, optimal ink coverage
Rules Engine	A software system that executes one or more rules in a runtime production environment. Rules take the form of true/false assertions, much like an if statement
Black Box	A system whose internal mechanism is usually hidden from or mysterious to the user.
API	A protocol that allows client communicate with the server easily
Redis	An open-source database that use in-memory data structure to store all of the data
Drools	A rules engine developed by Red Hat. Part of jBPM Workbench.

4.4 Timeline

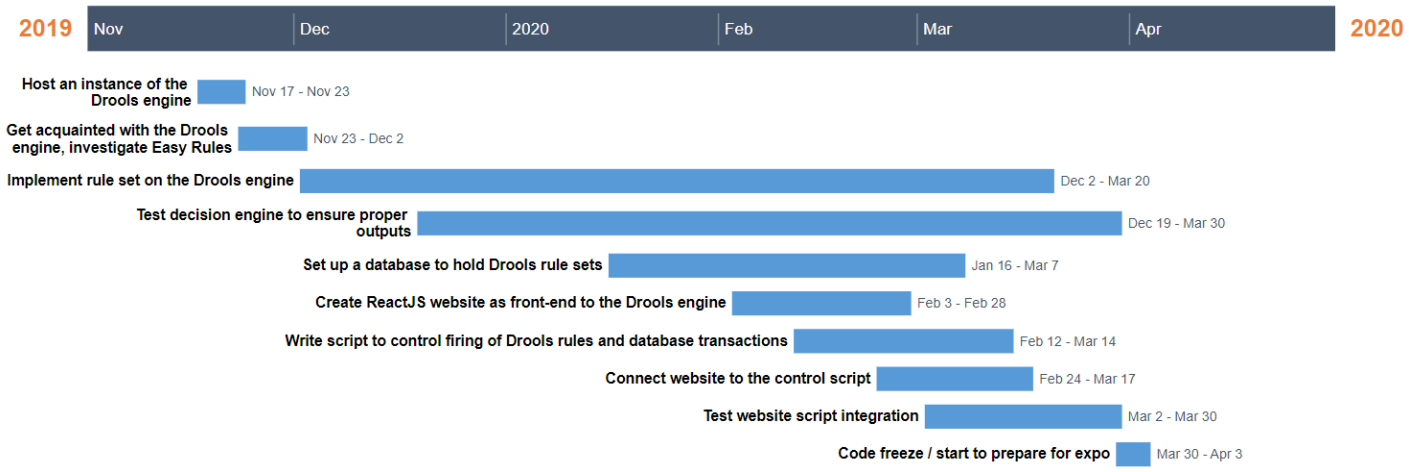


Fig. 2: Design Document Gantt Chart

4.5 Rules Engine

The rules engine will be handled by both Kuan-Yu and Cole equally, as it's the most integral part of the project and therefore requires the most attention.

4.5.1 Description of Design

The rules engine is the core of this project since it generates the ideal output for the user. One engine that has been considered is Drools, a Java-based business rules management system (BRMS) with both forward and backward chaining [1]. It allows the user to generate their own ruleset, then fire as many rules as they deem necessary in any order. The engine uses an algorithm called the Phreak rule algorithm to evaluate the rules. Phreak is a faster and more efficient version of the Rete algorithm (pattern matching algorithm invented in 1979) and supports more evaluation methods [2].

Easy Rules has also been considered for use as it implements a similar rules-based engine, albeit in a much simpler way, foregoing the additional functionality of an engine like Drools for simplicity and portability [3].

4.5.2 Design Viewpoints

4.5.2.1 Context

The rules engine will function as a sort of "black box." The user will provide input through either an API call or, more likely, the front-end website user interface. A series of scripts will manipulate the users inputs and call the engine's rules in a specific order to generate the output. The output will then be saved to the database and presented to the user.

4.5.2.2 Composition

The rules engine's functionality is defined in the following steps:

1) Take user input passed from website/API call
2) Use script to fire rules related to determining if the selected press is optimal
3) Return to script and gather data generated from fired rule

4) Continue firing rules to determine optimal settings
5) When done firing rules, package output data and save to database
6) Also return output data to front-end to present to the user

4.5.2.3 Dependency

The rules engine depends on the existence of the database so that different rulesets can be fetched before runtime and engine outputs can be saved. Although it technically doesn't rely on the existence of the website, the site is the optimal method for interacting with the engine. The engine also depends on having an instance of the jBPM Workbench up and running, as the Drools engine was merged into the tool, and an instance of the KIE Execution Server to be able to use its built-in API for making calls to the engine.

4.5.2.4 Interaction

The rules engine will expose a REST API to the user so that calls may be made to fire its rules. This API will be called by the website to provide user input, so the user does not have to directly interact with the engine. Apart from passing rules and receiving output, the engine is a "black box." The user does not know what it going on inside of the engine. Outputs are also posted to the database by the control script.

4.5.3 Design Rationale

The reason why Drools/Easy Rules was chosen for the rules engine is because some technology akin to a decision tree was required, as opposed to the initial idea of using some type of machine learning. It was determined that the task was not suited to machine learning, and rule-based reasoning was settled on. Drools was chosen because it is a powerful open-source program with a large user base, and is currently in use by some groups at HP.

4.5.4 Design Implementation

The rules engine will be composed of a series of scripts, most likely written in a language like Python, that compiles user data into the correct format and sends it to the Drools engine by calling its built-in REST API. The script will be called from the website when the user submits their input data. It will most likely call the Drools engine multiple times, firing different rules to categorize the job data for future calls. After all of the rules have been called and the output is finalized, the script will make a call to post the data to the database, then will return the output to the website so it can be displayed to the user.

4.6 Database

Kuan-Yu will be responsible for this piece of the project, but others will help during the implementation as well.

4.6.1 Description of Design

The purpose of the database is to store all of the data that is used in both the website and the rules engine. The database will categorize the data and provide quick access to the data while needed.

4.6.2 Design Viewpoints

4.6.2.1 Context

The database provides important functionality to both the website and the rules engine. It will provide stable data access whenever needed by the rules engine or website. As a result, it is considered one of the most important pieces of this project. The data for the rules engine will be pre-stored in the database, but the data for the website will be based on the input of the user.

4.6.2.2 Composition

The database functionality is defined in the following steps:
For interacting with the rules engine:

1) Provide rules engine the rule set used to generate the result
2) Provide rules engine the file uploaded from the user as the input
3) Store the result generated from the rules engine

For interacting with the website:

1) Store the file provided from the user
2) Store the setting set from the user
3) Provide the recommended setting data generate from the rules engine
4) Provide the data of existing tasks on the system

4.6.2.3 Dependency

The database is dependent on both the website and the rules engine. Whenever they need a data-related service, it will be handled by the database. Therefore the input and output for the rules engine will be the recommended settings generated by the engine and the file uploaded from the user. For the website, the input and output will be the file uploaded from the user and the recommended settings as well as existing tasks.

4.6.2.4 Interaction

The database does not directly interact with any other piece, rather the other pieces interact with the database. The control script running the rules engine will make a call to the database to fetch a ruleset or post its outputs. The website will make a call to the database to fetch all of the tables and update them if required, or to upload the user's input file.

4.6.3 Design Rationale

The reason why the database is a good choice to store the data is the categorize feature. It allows the user to categorize different kinds of data and find the relation of it. Therefore, it will increase the speed of searching and access to the data. In addition, there are different kinds of databases provided in the community and the majority of them are open-source, as a result, finding the right database can maximize the data access speed.

4.6.4 Design Implementation

The database will be implemented by a software called Redis [4]. The wide range of data types supported is the best fit for all of the data in this project. Furthermore, for the data which doesn't have any relation, Redis can access the specific data directly. The database will be held on the server and wait for the request all the time. Whenever the database receives an output request, it will search the data based on the request constrain and send back the data. If it receives an input request, it will store the data in the categorize space and assign a corresponding key to it.

4.7 Website

While anyone in the group is free to contribute to the website, the bulk of its creation will be handled by Cole.

4.7.1 Description of Design

The purpose of the website is to provide a front-end through which the user can interact with the tool. Both the rules engine and the database will be exposed to the website.

4.7.2 Design Viewpoints

4.7.2.1 Context

The website will provide a sort of ”black box” view of the rules engine, allowing users to provide inputs about a print job and returning the outputs of its computation. By using a front-end, a minimal amount of the rules engine can be exposed to the user so as to minimize complexity. Users will also be able to view and manipulate the databases that store the variables that are used by the rules engine and the engine's previous outputs.

4.7.2.2 Composition

The website's functionality is defined in the following steps:
For interacting with the rules engine:

1) Provide user a portal for entering print job inputs
2) Make API call to rules engine, sending user input as body
3) Receive output from rules engine
4) Save output to database, also present to user

For interacting with the database:

1) Make API call to fetch the tables from the database
2) Present the table to the user, allow them to edit entries
3) Take the user's input and update the table in the database

4.7.2.3 Dependency

The website depends on the existence of the rules engine, as well as its ability to take inputs and return outputs. It is also dependent on the existence on the database. The input will be information about the print job (collected via a form) like coverage and number of pages, and the output will be the result generated from the rules engine based on the input.

4.7.2.4 Interaction

The website will provide three different modes for the user. First is the submission mode: in this mode the user will be able to submit the files they want to print. Second is the setting editing mode: this is for people to configure the setting of each individual printing task. Third is the submit mode: the user (most likely a manager or admin) can make sure the task is ready to submit.

4.7.3 Design Rationale

The reason why a website was chosen to create a front-end for used interaction, as opposed to, say, exposing only an API to the user, was to reduce the complexity of interacting with the rules engine and database. This front-end allows users to upload their job details via a form and to view the tables in the database without having to make database calls or API calls with curl. While the API is still available for users to call on their own, and can be used as an alternate method for invoking the rules engine, having a web application produces a sort of ”black box” around the engine. This way, less responsibility is placed on the user.

4.7.4 Design Implementation

The website will be created using ReactJS as the main framework for structure, Bootstrap as the CSS framework for design, and React-Redux for managing application state. ReactJS is an open-source JavaScript library for creating user interfaces. It was launched by Facebook in early 2013 [5]. It is optimized for applications where data on the page is changing rapidly. Bootstrap is the number one most popular open-source CSS framework that uses jQuery and JavaScript design templates to create user interfaces. It was created by Twitter in August 2011, and has since undergone three major rewrites to bring it up to version 4.0 [6]. React-Redux is a JavaScript library for managing application state, based paritally on Facebook’s Flux [7].

The design of the website is going to prioritize efficiency and usability. The user will be presented a simple and clean Bootstrap user interface that is easy to interact with. All of the API calls will be made using ReactJS to ensure speed when fetching database tables or making API calls. The website will be easy to use and efficient in its handling of data.

4.8 Conclusion

In conclusion, this project consists of three major sections: the rules engine for evaluating inputs and producing optimal press settings as its output; a database to store all of the rulesets for the rules engine as well as all of its past outputs; a website to present a UI to the user and allow them to invoke the rules engine or view the contents of the database. The rules engine will be controlled by a script that determines the order in which to fire the rules, and the script will be invoked by the website when the user submits their inputs.

The rules engine will be either Drools, an open-source business rules management system created by Red Hat, or Easy-Rules, a lightweight rules engine written in Java. The database will be constructed using Redis, an in-memory data structure project created by Redis Labs. The website will be a combination of ReactJS, a java framework created by Facebook, and Bootstrap, a CSS framework created by Twitter.

4.9 References

[1] “Drools - Business Rules Management System (Java™, Open Source),” Drools. [Online]. Available: https://drools.org/. [Accessed: 23-Nov-2019].

[2] “Chapter 5. Rule Algorithms Red Hat JBoss BPM Suite 6.2,” Red Hat Customer Portal. [Online]. Available: access.redhat.com/documentation/en-us/red_hat_jboss_bpm_suite/6.2/html/development_guide/chap-rule_algorithms. [Accessed: 23-Nov-2019].

[3] M. B. Hassine, “j-easy/easy-rules,” GitHub, 18-Nov-2019. [Online]. Available: https://github.com/j-easy/easy-rules. [Accessed: 23-Nov-2019].

[4] Redis.io. (2019). Redis. [online] Available: https://redis.io/documentation [Accessed 23-Nov-2019].

[5] A. Papp, “The History of React.js on a Timeline: @RisingStack,” RisingStack, 04-Apr-2018. [Online]. Available: https://blog.risingstack.com/the-history-of-react-js-on-a-timeline/. [Accessed: 23-Nov-2019].

[6] M. Otto and J. Thornton, “About Bootstrap,” Bootstrap. [Online]. Available: https://getbootstrap.com/docs/4.3/about/overview/. [Accessed: 23-Nov-2019].

[7] “Redux · A Predictable State Container for JS Apps,” Redux. [Online]. Available: https://redux.js.org/. [Accessed: 07-Dec-2019].

CS CAPSTONE TECH REVIEW DOCUMENT
NOVEMBER 8, 2019

AUTOMATE THE SETTINGS THAT CONTROL A MILLION-DOLLAR PRINTING PRESS

PREPARED FOR
HP, INC
PIETER VAN ZEE

PREPARED BY
GROUP62
PROPRIETORS OF THE PRESS
KUAN-YU LAI

Abstract

The document includes three pieces from the term project “Analysis of input data for generation of outputs”, “Hosting the feature on the web”, and “Storage of outputs and profiles”. In each piece, it contains three different technology that is related to the pieces. The technology is considered to be a temporary option in the project.

5.1 Introduction

Our team project is to design an assisting feature for the user of the HP PageWide printing presses. This assisting feature will help the user to find the best set of each individual printing tasks they submit into the system. The decision of the setting will be based on factors such as page type, printing quality, or ink usage.
The core of this feature is an engine that can consider every input and generate the most optimal set of settings. The feature can be broken down into several pieces which are “Analysis of input data for generation of outputs”, “Hosting the feature on the web”, and “Storage of outputs and profiles”.

5.2 Piece1: Analysis of input data for generation of outputs

The analysis of input data is the main piece of the project, it determines the reliability and the meaning of the assisting feature. It is an engine that will take all of the input into consideration based on the ruleset we decide. Eventually, generate the optimal set of setting for the users.

5.2.1 Drools

Drools, also known as knowledge is everything(KIE), is a business rules management system(BSMS) that allows the user to generate the result based on the ruleset of their own. It is also a Java rules engine with forwarding and backward chaining. The engine uses an algorithm called the Phreak rule algorithm to evaluate the rules. The algorithm is an enhanced version of the Rete algorithm, a pattern matching algorithm invented in 1979. It’s better at speed and supports more evaluation methods. A benefit of using this technology is that it the same concept with our assisting engine. Therefore the implementation will be easier and possibly more time-efficient.[1]

5.2.2 Google Cloud Platform

Google cloud platform is a platform announced by Google in 2008. It’s a cloud-computing platform similar to Amazon Web Services(AWS). It allows you to use the well-trained machine learning engine from Google. The user will be able to input data into the engine and gets the analytical data from the engine. It supports Rest API so it’s very friendly for the user who doesn’t want to train their own engine. However, Google does support customize engine. Users can either use the built-in algorithm or create their own training application and run it on the cloud. A benefit of using this technology will be the advantage of the pre-trained engine. The engine already has high accuracy on the picture analysis which might be very helpful for our input analysis.[2]

5.2.3 TensorFlow

TensorFlow is an open-source platform specifically for machine learning from Google. It allows the user to build the machine learning application with a variety kind of computer languages. The high-level API makes the building and training steps easier and the user can deploy the application on to their product eventually. A benefit of using this technology will be a friendly development environment for the beginner. At the same time, it is also very powerful enough to generate a decent machine learning engine.[3]

5.3 Piece2: Hosting the feature on the web

To provide a convenient service to the client of the presses. Our project will have a cloud application that allows users to submit their printing tasks with the setting they want on the cloud. The cloud application will then sent the input data from the user to the engine and generate the result for them. The cloud application and the engine will be communicated through API. As a result, we have to find a reliable and customizable web server.

5.3.1 Oregon State University engineering server

Oregon State University allows students to publish their own applications using their servers. Each student will be assigned a cloud space with limited storage. Students then can publish the application they made by simply move the data into the publish folder. After that, anyone can access the application with the link http://web.engr.oregonstate.edu/~your_engineering_username. However, the service isn't very customizable but they do provide the other paid server which costs 88 dollars monthly. A benefit for using this server will be the server is very stable and is maintained by the university. One disadvantage of using this server is the free version might support the engine we built. Also, we will have to pay monthly if we want to have an advanced server.[4][5]

5.3.2 Heroku

Heroku is a cloud application platform that allows you to put your application on the cloud and publish it to the world. Its popularity is growing for the past few years due to its simplicity and highly customizable property. The reason why it's simple to use is that the developer will not have to worry about the infrastructure of the server. Heroku handles both hardware and server for you so you can focus on perfect your own application. This is very friendly to people who don't have lots of experience in setting up and maintaining the server. As a result, it's being used by large non-tech related companies such as Toyota. The benefits of using Heroku will be, eliminate all the issues from the server and hardware since this is handled by Heroku. Some disadvantages will be the collaboration issue and the active time of the application. The free version of Heroku only allows one person to use and will put the application into sleep mode if it doesn't receive any traffic within one hour. Although the manager will wake the server up when it receives a request. It will still cause a significant delay for the application to restart.[6]

5.3.3 Amazon Web Services

Amazon Web Services(AWS) is a well-known cloud platform from Amazon since 2006. It's being used by many large companies such as General Electric. The platform provides thousands of features which also include hosting the application. The server provided by AWS is extremely customizable and stable. It also supports some other popular platform like docker, a set of platforms that allows you to build your application from scratch rapidly. It also supports machine learning computation similar to the Google Cloud Platform. A benefit of using AWS will be highly customizable and supported properties. We won't have to worry about almost any cloud-related technology that doesn't support our server. A disadvantage for this is the services isn't free but they offer a free trial for 12 months after the first sign-up.[7]

5.4 Piece3: Storage of outputs and profiles

To build an application, it's essential to have a storage space that can keep all the user data and the ruleset we made. This is the fundamental piece of the project since all of the features is based on the data stored in the database. Thus, it's important to choose a reliable database that is suitable for our data.

5.4.1 MYSQL

MYSQL is a traditional open-source database developed by a tech company called Oracle. It's a database that uses structured query language(SQL) and stores the data as tables. MYSQL is widely used in the industry, it's being used by companies like Walmart or Spotify. Some benefits of using MYSQL will be it's widely support property and powerful query feature. A disadvantage of this is that our rules don't have a significant relationship. This might make the query a little bit complicated when pulling the rules.[8]

5.4.2 MongoDB

MongoDB is a modern database that is considered to be the most popular database nowadays. Unlike the traditional row and column-based structure, it uses the document-based structure and is built for the cloud era. The data will be formed as a JSON file, so developers can modify the JSON document to edit the data in the database. The powerful query feature is another reason why it's extremely popular. Furthermore, it provides a statistical feature that allows the developer to visualize the data stored in the database. Due to its outstanding ability in this era, it is being used by giant tech companies like Google and Facebook. A benefit of using MongoDB will be the usage of powerful query feature. Since our ruleset doesn't have a significant relationship with each other. The powerful query feature will simplify the query compared to the traditional SQL database.[9]

5.4.3 Redis

Redis is also an open-source database. Unlike the traditional row and column-based structure and document-based structure. Redis uses key-value based on its structure. That is, developers can pull out the specific data by query the key of that data. Also, Redis stores the data in the memory instead of disk, so the speed is extremely fast compared to the database like MongoDB or MYSQL. A benefit of using Redis is that we will be the property of easily pull out specific data. The data has a corresponding key so the relationship between the data won't affect the query complexity significantly. A disadvantage of using Redis will be the in-memory storing structure because it's not stored permanently. Thus, it will have to rebuild the database whenever the server restart.[10]

5.5 Conclusion

In this modern world, there are thousands of technology that exist and more and more coming out in the future. In addition, finding and choosing the appropriate technology will increase the performance and the development time of the application. Therefore, the technology present in each piece will not be the only option to consider.

5.6 References

[1] "Drools Documentation. [online] Docs.jboss.org. Available at: https://docs.jboss.org/drools/release/7.29.0.Final/drools-docs/html_single/index.html [Accessed 9 Nov. 2019].

[2] Google Cloud. (2019) Training Overview AI Platform Google Cloud. [online] Available at: https://cloud.google.com/ml-engine/docs/training-overview [Accessed 9 Nov. 2019].

[3] TensorFlow. (2019). Introduction to TensorFlow — TensorFlow. [online] Available at:https://www.tensorflow.org/learn [Accessed 9 Nov. 2019].

[4] Information Services. (2019). Server Management — Applications Access and Deployment, Shared Infrastructure Group — Information Services — Oregon State University. [online] Available at: https://is.oregonstate.edu/service/server-management [Accessed 9 Nov. 2019].

[5] It.engineering.oregonstate.edu. (2019). Where do I put my personal webpages? — Information Technology and Computing Support — Oregon State University. [online] Available at:
https://it.engineering.oregonstate.edu/where-do-i-put-my-personal-webpages [Accessed 9 Nov. 2019].

[6] Devcenter.heroku.com. (2019). How Heroku Works — Heroku Dev Center. [online] Available at: https://devcenter.heroku.com/articles/how-heroku-works [Accessed 9 Nov. 2019].

[7] Docs.aws.amazon.com. (2019). What Is Amazon EC2? - Amazon Elastic Compute Cloud. [online] Available at: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/concepts.html [Accessed 9 Nov. 2019].

[8] Dev.mysql.com. (2019). MySQL :: MySQL 8.0 Reference Manual :: 1 General Information. [online] Available at: https://dev.mysql.com/doc/refman/8.0/en/introduction.html [Accessed 9 Nov. 2019].

[9] MongoDB. (2019). What Is MongoDB?. [online] Available at: https://www.mongodb.com/what-is-mongodb [Accessed 9 Nov. 2019].

[10] Redis.io. (2019). Redis. [online] Available at: https://redis.io/documentation [Accessed 9 Nov. 2019].

CS CAPSTONE TECH REVIEW DOCUMENT
NOVEMBER 8, 2019

AUTOMATE THE SETTINGS THAT CONTROL A MILLION-DOLLAR PRINTING PRESS

PREPARED FOR

HP, INC

PIETER VAN ZEE

PREPARED BY

GROUP62

PROPRIETORS OF THE PRESS

COLE JONES

Abstract

This document presents three pieces of the overall project: the capturing of input data from the user; the API to be used for interacting with the tool; and the presentation of the tool to the user via some web application. Each piece listed has three technologies taken into consideration, weighing the pros and cons to determine whether or not the technology is worth using. Although a conclusion is provided that states which technologies we are likely to use going forward, those choices are subject to change in the future depending on the desires of the project's clients.

6.1 Introduction

Our team is attempting to create a rules-based decision-making engine to help provide recommendations for settings of different HP PageWide printing presses. These settings are based on a number of factors, namely the weight and density of the paper selected, the size of the material to be printed, and the maximum ink coverage. These inputs, among others, will be used by the decision-making engine to determine the optimal output settings that control the speed of the press, the temperature of the press' dryer, the tension of the press' rollers, and whether or not to use bonding agents, primers, and moisturizers.

The decision-making engine will require some method of capturing user inputs, presenting a front-end to the user, analyzing user inputs to determine optimal press settings, storing previous outputs, and providing an API so the engine can fit inside an existing job-creation pipeline.

6.2 Piece 1: Capture of Input Data from User

The handling of input data is the first step in the project's pipeline. At this point it is assumed that all pertinent data has been scraped from the customer's PDF in the PDF pre-flighting, so no analysis or risk-assessment is done on our part. There are multiple ways of collecting data about a print job, which will be discussed in the following subsections.

6.2.1 User Interface

The first logical step is to present the user with an interface. This interface will act as the front-end of the tool, providing a place for users to submit a PDF file and an XML file for their print job. This may not be the easiest to implement, as it would involve creating and hosting a web application, but it would most likely be the most user-friendly.

A benefit of using a user interface is that the user would be able to see the progress of the decision engine, and would immediately be able to whether or not the document was accepted, and if so, view its outputs and recommendations. Users would be able to manually determine the next step of the print job: they continue with recommended settings produced from the engine; they see that the job was rejected or that the recommended settings do not align with what they want, and they pull the job ticket; they continue along the pipeline, sending the job and its ticket into the printing queue.

6.2.2 Email

Another possibility for capturing user input would be to allow the user to simply email their input PDF and XML files. From there either an automated system is set up (this would most likely be something that HP does themselves that we are not a part of creating) to process the email and send its attachments to the decision engine, or this would have to be done manually by an HP employee.

A benefit of using an email-based system is that it's fire-and-forget. A user sends an email with the input files as attachments and later receives a reply with the verdict of the decision engine. The problem with this system however, is that there is no way for the user to see the status of their submitted job - they only get a response when it's finished. They also have no way to decide whether or not they want to pull the job from the queue, as the decision engine makes the choice to accept or reject the submission.

6.2.3 Hot Folder

Also known as a "watched folder" a hot folder is a network folder that is associated with the decision engine [1]. It allows users with access to drop files inside, which are then processed by a script that checks for new files. The script takes the file directly from the hot folder and sends it to the decision engine for processing.

A benefit of using a hot folder it that it's even more hands-off than emailing the print job. Once the files are dropped into the hot folder, they are picked up into the queue for the decision engine. A major drawback to hot folders is that they separate jobs by file. That is, they are incapable of selecting multiple files for the same job [1]. This drawback makes them an unlikely candidate for user input capture.

6.3 Piece 2: Tool Interaction Using API

In order to fit alongside the existing job-creation pipeline, the decision-making engine must be able to be interacted with via scripts that make API calls. That way the first stage of job creation, pre-flighting, will be able to pass PDF and XML files directly to the engine without human intervention. After the engine has processed the files and made its settings recommendations, it will be able to pass those outputs to the job queue and a database for storage. There are many options out there for creating an API, but I will only be looking at three: REST, SOAP, and JSON-RPC.

6.3.1 REST

REST is an acronym for **RE**presentational **S**tate **T**ransfer. It was created in 2000 by Roy Fielding for use in distributed hypermedia systems. REST is a method for creating APIs that is guided by a number of principles. Namely, that it's client-server based (UI concerns are separated from data storage concerns), stateless (each request contains all the information to understand the request), and cacheable [2]. It uses the same HTTP protocol that's used to view regular web pages, making implementation faster and easier. REST APIs allow making available through an API information that is already present on the web page [3].

The benefits of using a REST API are mainly due to the separation between client and server. Because the client does not need to know how the server interacts with the system, REST APIs tend to scale better than other methods, tend to be more flexible and portable, user less resources, and are typically much simpler [4]. However, there are some drawbacks. First, REST APIs have no way of addressing unsuccessful requests other than simply attempting the request again. Second, REST APIs are not as secure as other methods, such as SOAP [5].

6.3.2 SOAP

SOAP is an acronym for **S**imple **O**bject **A**ccess **P**rotocol. It was created in 1998 for Microsoft, and quickly became popular for creating APIs. SOAP relies on XML, with each operation being explicitly defined. Unlike REST, which is required to use HTTP to send its requests, SOAP requests can be sent over almost any protocol, including but not limited to HTTP, SMTP, TCP, and JMS [6]. SOAP messages are defined by the WSDL (Web Service Definition Language), an XML document that defines the functions that are implemented and presented to the user.

SOAP APIs have a number of benefits over REST APIs. In addition to being able to be sent over almost any protocol, SOAP APIs have built-in error handling, it's standardized, it's language and platform independent, and works well in distributed environments (whereas REST assumes point-to-point communication) [6].

There are some critiques of SOAP, however. One big drawback is that if you want to make a change to your API implementation, then the WSDL must change, which means that the client will have to recompile their application. Another is that although SOAP APIs can use almost any transfer protocol, this is not often taken advantage of, as the industry-standard is to use HTTP. Also, SOAP is not nearly as scalable, flexible, and lightweight as REST [6].

6.3.3 JSON-RPC

JSON-RPC is a stateless remote procedure call (RPC) protocol. It is used to request a service from a program on the same network, and utilizes the JSON notation to define the properties of the request. It is much simpler than SOAP or REST APIs, and therefore uses significantly less bandwidth [7].

Some advantages it has over REST and SOAP APIs are as follows: like SOAP APIs, JSON-RPCs are agnostic to the protocol, meaning that it can use TCP or SMTP instead of HTTP, reducing some overhead; REST verbs are limited, and JSON-RPC can define its own methods; passing parameters is much easier, as they are defined in the body of the JSON object [7].

6.4 Piece 3: Presentation of Tool to User

In order for the user to be able to directly interact with the decision-making engine, a user interface must be created and hosted. From this UI, the user will be able to submit print jobs, view the status of jobs currently in the engine's queue, and view the output of both current and previous jobs. All of the functionality of this web tool will be tied to the existing print job pipeline via an API, discussed in the previous section. There are a number of ways of going about this, but the most appropriate method is to use a lightweight web framework. We have chosen three potential technologies to use: ReactJS, Bootstrap, and Angular.

6.4.1 ReactJS

ReactJS is an open-source JavaScript library for creating user interfaces. It was launched by Facebook in early 2013 [8]. It is optimized for applications where data on the page is changing rapidly. It achieves this optimization by using a virtual DOM (document object model) to update only the components on the page that changed, instead of rerendering the whole page. This makes it very useful for fetching large amounts of data from a database that is constantly changing.

There are many advantages to using ReactJS, including but not limited to: the ability to reuse the same component to render different objects on a page; downward data binding, so that changes to a child component does not affect the parent; a wide set of UI libraries to use, created by a large open-source community of developers [9].

In addition to its many advantages, there are some downsides. For one, ReactJS has a high pace of development. Things are constantly changing and adapting, and it can be difficult to keep up. A side effect of this rapid development is that there is poor documentation. Since changes are happening so frequently that there's no time to write proper documentation [9].

6.4.2 Bootstrap

Bootstrap the number one most popular open-source CSS framework that uses jQuery and JavaScript design templates to create user interfaces. It was created by Twitter in August 2011, and has since undergone three major rewrites to bring it up to version 4.0 [10]. Its two main appeals are that it lets users create clean-looking responsive websites without having to write thousands of lines of CSS, and that it has a built-in responsive grid system that allows content to be segmented easily.

Some of its advantages include the built-in grid system mentioned above, responsive images that resize automatically based on current screen size, and a whole host of control components like navigation bars and dropdown menus. Also, Bootstrap has very well-defined documentation, much better than the aforementioned ReactJS [11].

There are, however, some downsides. One of the major complaints with Bootstrap is that its syntax can be confusing, making the library difficult to lean. Also, Bootstrap files are notoriously large, which can lead to an increase in load time for websites built using the framework [11].

6.4.3 Angular

Angular is a JavaScript framework that was created by Google in 2012. Over the years it has been updated and majorly rewritten a few times, sitting at version 9.0 today. Similarly to ReactJS, Angular works by controlling elements of a page individually, updating a specific element when its data is changed [12]. It does so using the Model-View-Controller (MVC) architecture. Essentially, in binds JavaScript and HTML, accepts user input using JavaScript, and uses said input to modify the HTML [13]. Although it's possibly to write Angular code in JavaScript, it is recommended to use TypeScript.

Some of Angular's benefits include data binding, allowing the web application to quickly respond to user input, modularity, allowing components to be broken up into modules which can then be puzzle-pieced together with other modules to form a responsive page, and custom directives, which allows for the manipulation of HTML functionality [14].

Some of the downsides are similar to those listed for the other UI framework options. Although Angular can be easy to learn at first, it has a fairly steep learning curve, taking a lot of time to figure out how to use its more complex features. Like React, the documentation is fairly good, but in need of improvement. It's also fairly difficult to debug because it can be difficult to determine what scope is being used [14].

6.5 Conclusion

In conclusion, there are a significant number of free open-source technologies out there for use in this project. The ones included in this document were considered to be the strongest, be it through its user base or recency. All things considered, the technologies we are most likely to use going forward in this project are as follows: user interface for capturing input data (although as a stretch goal, all of the technologies listed will be implemented), REST for the API (as it seems like this is the industry standard, and I have some experience with it), and as for UI, we're not entirely sure yet. React seems to be the new industry standard, and I do have a lot of experience with it, but all of the technologies

listed are valid for use in creating a web application. Most likely, the web app will be a combination of all three, using React + Angular for structure and data handling and Bootstrap for styling. However, this is all up for depending on what our client deems is appropriate. As such, all of these choices are subject to change in the future.

6.6 References

[1] "Hot folder," DAM Glossary, 17-Apr-2014. [Online]. Available: https://damglossary.org/hot-folder. [Accessed: 08-Nov-2019].

[2] "REST API Tutorial," What is REST – Learn to create timeless REST APIs. [Online]. Available: https://restfulapi.net/. [Accessed: 08-Nov-2019].

[3] G. Levin, "RESTful APIs Technologies Overview," RestCase, 18-Nov-2017. [Online].
Available: https://blog.restcase.com/restful-apis-technologies-overview/. [Accessed: 08-Nov-2019].

[4] "What are the advantages of a REST API?," Chakray, 05-Jun-2019. [Online].
Available: https://www.chakray.com/advantages-of-rest-api/. [Accessed: 08-Nov-2019].

[5] D. Breaker, "SOAP vs REST - Which API Architecture Reigns Supreme?," DreamFactory Blog, 18-Sep-2018. [Online].
Available: https://blog.dreamfactory.com/soap-vs-rest-apis-understand-the-key-differences/. [Accessed: 08-Nov-2019].

[6] "SOAP vs REST 101: Understand The Differences," SoapUI. [Online].
Available: https://www.soapui.org/learn/api/soap-vs-rest-api.html. [Accessed: 08-Nov-2019].

[7] "JSON-RPC vs REST for distributed platform APIs," Radix DLT - Decentralized Ledger Technology, 13-Apr-2018. [Online]. Available: https://www.radixdlt.com/post/json-rpc-vs-rest/. [Accessed: 08-Nov-2019].

[8] A. Papp, "The History of React.js on a Timeline: @RisingStack," RisingStack, 04-Apr-2018. [Online].
Available: https://blog.risingstack.com/the-history-of-react-js-on-a-timeline/. [Accessed: 08-Nov-2019].

[9] "The Good and the Bad of ReactJS and React Native," AltexSoft, 10-Sep-2018. [Online].
Available: https://www.altexsoft.com/blog/engineering/the-good-and-the-bad-of-reactjs-and-react-native
[Accessed: 08-Nov-2019].

[10] M. Otto and J. Thornton, "About Bootstrap," Bootstrap. [Online]. Available: https://getbootstrap.com/docs/4.3/about/
[Accessed: 08-Nov-2019].

[11] A. Ouellette, "What is Bootstrap: A Beginners Guide," What is Bootstrap? An Awesome Beginners Guide, 20-Sep-2017. [Online].
Available: https://careerfoundry.com/en/blog/web-development/what-is-bootstrap-a-beginners-guide/.
[Accessed: 08-Nov-2019].

[12] I. Bodrov-Krukowski, "Angular Introduction: What It Is, and Why You Should Use It," SitePoint, 22-Mar-2018. [Online]. Available: https://www.sitepoint.com/angular-introduction/. [Accessed: 09-Nov-2019].

[13] A. N, "What is Angular: All You Need to Know About the Popular JS Framework," Hostinger Tutorials, 10-Apr-2019. [Online]. Available: https://www.hostinger.com/tutorials/what-is-angular. [Accessed: 09-Nov-2019].

[14] "The Pros and Cons of choosing Angular for web app development," DDI Development, Jan-2019. [Online]. Available: http://ddi-dev.com/blog/programming/pros-and-cons-of-angular-web-app-development/. [Accessed: 09-Nov-2019].

7 Blog Posts

7.1 Fall Term

7.1.1 Cole Jones

Week 4	**Progress**: Our team has completed about half of the Rules document that our client wanted us to generate using a collection of HP white pages on best practices. We met with our client today to discuss our findings and lay out a roadmap for what the remainder of the term. They helped us with uncertainties about the Rules document and helped broaden our understanding of the project and what they want to see as a final product. We were also tasked with finding some help with machine-learning from experts in the field around OSU. **Problems**: We only had a minor issue with the Rules document. We were unsure if we were doing the assignment correctly, and had some concerns about whether we were looking hard enough in some of the documents or if there really was no information relevant to best practices. The client clarified their expectations about the assignment and we are not slated to finish the document over the weekend. **Plans**: Our plan for the immediate future is to finish gathering rules and best practices for the Rules document. After we are done with that, we are tasked with meeting up with experts in the field of machine-learning at OSU and discussing what kinds of tools would best fit the requirements of the project. We are also expected to distill the Rules document (when we are done writing it) to a smaller selection of rules that are deemed more important and encompassing of the main ideas.
Week 5	**Progress**: After reviewing the Rules document and seeing that it was mostly my work, our client decided that I should finish writing that document myself, and that my teammate should create his own version. I finished writing the document and we reviewed my findings during today's meeting. Our client also reviewed our problem statement and offered some suggestions on how to tweak the Gantt chart and some of the use cases. We ended up adding several use cases that more accurately represent the scope of the project for the remainder of the term. We also obtained a list of professors at OSU that are the in the machine-learning field. **Problems**: We didn't really have any problems. I was able to finish my Rules document on Friday morning, but York was not. He will finish the document within the next few days. **Plans**: We were tasked with combining both of our Rules documents into a single document, then narrowing it down into a set of representative rules. We are to take these representative rules and meet with a number of professors at OSU that have a background in machine-learning, gathering as much expert knowledge about machine-learning and the available tools as we can. Our client wants us to present these findings, along with additional independent research about available machine-learning engines, and present them during next Friday's meeting.
Week 6	**Progress**: Some research has been done into a couple of applications of computer vision. It's my thought that computer vision could be used to determine total area coverage of ink in the PDFs. We sat down with professor Tadepalli to ask him some questions about machine learning and how it can be used in our project. He said that we might be better off not using machine learning, and to instead use conditional logic or something simpler like a decision tree. He did, however, say that computer vision could be useful. I went to Scott Fairbank's office hours on Thursday to talk about machine learning and computer vision. He is going to sit in on our next meeting this Monday and help us get the information we need from our client to determine whether or not machine learning will be useful. **Problems**: We've found by talking to multiple people that our problem may not necessarily require machine learning, but would benefit more from computer vision and something like a decision tree. We will be meeting with our clients to discuss this. **Plans**: We will continue to do research about available technologies and how they may be useful in creating the engine for our clients. We will also meet with our clients on Monday, and Scott Fairbanks will join us to make sure we get the information we need, and to assist us with any machine learning-related questions.
Week 7	**Progress**: With the help of Mr. Fairbanks, we were able to talk to our clients about machine learning and how it relates to the project. He helped them understand that the scope of the project was too large, and clarified the appropriate amount of time per week we should be spending on it. He also helped us figure out what type of machine learning we should pursue.

	Our clients gave us a revised version of the project with clear goals, including a basic outline of what our engine's inputs and outputs should be and some research into a tool in use at HP that we could potentially utilize. We also revised our set of representative rules to filter out rules that are not applicable for the engine. **Problems**: After Mr. Fairbanks talked with our clients about the scope of the project, there have been no problems. **Plans**: We are going to look into the use of Drools as a rules-based engine to see if it's well suited to the goals of this project. We will also look into services available to host our tool, including Google Cloud and AWS.
Week 8	**Progress**: Our group managed to get an instance of the Drools Workbench up on both AWS and Microsoft Azure. My instance, the one on Azure, has data persistence, so that when the Docker containers are stopped and restarted, the application retains all data is was given. Our clients are pleased with our progress, and expect us to look further into Drools as our engine of choice. They still want us to look at alternative rules-based engines, but for the most part we are locked into Drools. **Problems**: The was a bit of difficulty getting the Drools image set up on Docker, but both my partner and I have been able to do so on our respective sites. I was having some trouble getting a non-empty response when calling the tool with its REST API, so I'm going to have to look into that. https://www.overleaf.com/project/5ed0673f4f22340001925472 **Plans**: We are going to continue working with the Drools engine and get to know how it works. We are also looking into some alternatives to Drools, but for the most part, we're sticking with Drools.
Week 9	**Progress**: I managed to get some sample rules set up on Drools. I was able to make an API call, passing an input object and receiving a modified object back. So now that I have some rules up and running and returning outputs, I just have to look deeper into the tool to find out how best to implement our rule set. My partner York looked into some different ways in Drools to create a rule set. **Problems**: There were not any problems this week. **Plans**: We are going to continue looking into Drools to find out the best method for implementing our rule set. Our clients also asked us to look into something called Easy Rules, a simpler rule creation engine that works similarly to Drools, albeit much simpler and therefore more lightweight. I'm going to focus in on exploring Easy Rules while my partner focuses in on Drools.

7.1.2 Kuan-Yu Lai

Week 4	**Progress**: Read half of the document provided by the client. Extract and record all of the rules and practices from the document into Google Doc file. Present the result to the client during the meeting on Friday. **Problems**: Not able to find content that the client wants in some document. **Plans**: Review all of the documents and produce evaluation ruleset.
Week 5	**Progress**: Finish document review as individual. Working on merging individual document to final document. Present individual result to the client and explain some difference between individual work. Clarify the term use and use cases expectation from client. **Problems**: Progress a bit behind, should've done the draft of interview question. **Plans**: 1. Finish merging individual document during weekend 2. Finish the interview question early next week 3. Schedule interview time with OSU machine learning professor 4. Start to research machine-learning (Decision-Making) engine online and record the research result in document
Week 6	**Progress**: 1. Merging all the document 2. Start contacting and interviewing professor 3. Start researching online about the technology we might use **Problems**: Need to verify the technology use from the client. After talking with the professor, decision-making tree might be a better option for us. **Plans**: 1. Meeting with client on Monday and decide which technology should we use, either machine learning or decision-making tree 2. Keep researching online after the meeting 3. Generate some experimental prototype during learning process
Week 7	**Progress**: 1. Research on the technology option for the project 2. Client narrow down the requirement of the project and clarify the goal again **Problems**: Have to make more progress in order to finish the project plan before the design document due two weeks later. **Plans**: 1. Research on Drools, the technology suggested by client 2. Generate experimental program with drools with the ruleset we made

Week 8	**Progress**: 1. Research on the server environment for future use 2. Set up environment on AWS and Microsoft Azure **Problems**: Since we are using free tier we need to estimate the budget as a free tier user. Also, AWS isn't being honest to the user. That means we might accidentally active paid service which charge hourly and will become a significant cost in the future. **Plans**: Start to implement the ruleset using drools.
Week 9	**Progress**: Research on drools tool and dockers on Microsoft Azure **Problems**: Drools is very complicated and hard to learn. This will cause the program hard to maintain in the future. As a result, maybe using the simpler engine will solve this problem since the rulesets of ours isn't large **Plans**: 1. Generating more stuff using drools 2. Take a look into the new tool called j-easy, a simple version of rules engine

7.2 Winter term

7.2.1 Cole Jones

Week 1	**Progress**: Our group spoke with our clients to lay out the plan for this term. We made a decision on what rules engine we want to continue working with. We also talked about getting a front-end UI up to provide input to the engine and a back-end to process that input and package it in a format that work with the rules engine. **Problems**: My free credits for Microsoft Azure expired and I'm no longer able to access neither my virtual machine nor the files that were on it. I'm going to have to either pay some money to start the VM again or just start the rules engine from scratch. **Plans**: The overall plan for the following week is to draw a mock-up of what the website UI will look like, implement a very basic version of said UI with a simple back-end to communicate with the rules engine. We will also continue to work with the rules engine to add more functionality. Hopefully by the end of next week we will have a simple website (or at least the designs for one) and a simple working prototype rules engine.
Week 2	**Progress**: We had a meeting with our client where we went over the work that we completed over the past week. York demonstrated the scripts he wrote for interacting with the rules engine, and I did a short tour of the template website that I put together. Also, I was able to upgrade my subscription to Azure to get the files from the virtual machine that was locked to me when my free trial ended. **Problems**: No problems were encountered this week. **Plans**: The plan for the following week is to continue working on the website and the rules engine. We were told by our client that it would be idea if we were using Java for the rules engine instead of python (which the demo was written in) and JavaScript for the website's backend instead of C. We were also given direction on how they want the website to look, including how the results page should be laid out and stored in the database.
Week 3	**Progress**: I met with our client alone this week since my partner was out sick. We went over my progress on the website for the tool and discussed changes that we would like to see in the future. We discussed the possibility of using actual product names in the developer version of the tool, and we decided that we could use the actual data locally and add the file to .gitignore so it doesn't accidentally get uploaded to github. The clients are pleased with my work so far, and are working to compile more inputs for the tool. We also discussed the possibility of using the tool on a mobile device, so now I'm going to have to take into consideration that the UI might need to scale down to fit on a phone screen. **Problems**: York was sick for the meeting this week, but we've scheduled another meeting on Monday so that he can join and be caught up. **Plans**: The plan for the following week is to continue working on the website using additional data that our clients are in the process of collecting for us. Also, I'm going to split my attention between the website and Easy Rules, since I was exclusively focused on the website this week.

Week 4	**Progress**: York and I shared our progress with our clients during our regularly scheduled Friday meeting. The clients seemed pleased with my progress and like how the UI is coming together. They suggested a few things that I could tweak, and said that I should finish that up quickly and move on to the job results page. York showed off his progress with the rules engine, which unfortunately hit a stop because of an error he was unable to figure out. The clients suggested that he put the time in to break through the error and finish up base functionality. **Problems**: York hit a stopping point in the rules engine because of a bug that he is yet unable to identify. Other than that, there were no problems this week. **Plans**: I plan on finished up the job submission form, ironing about any bugs and refactoring messy code. After that, I'll move on to the job results page. I'm hoping that York will be able to get the server up and running so that I can use actual rules engine outputs for my data, but if he isn't able to do so quick enough then I'll just use mock data.
Week 5	**Progress:** We had our typical Friday meeting with our clients and demoed the progress we made since last week. I was asked to add a bit of functionality to the paper selection section of the job input form, and was able to do so fairly easily. Unfortunately, it's been a very busy week with midterms coming up, and I have not been able to make any more progress that that. **Problems**: Because midterms are coming up, little progress was made this week on both my end and my partner's end. We've decided that this weekend we're going to put in extra time. Our Friday meeting was pushed back to Monday because of a time conflict, so we should be able to get some more work done before meeting with our clients again. **Plans**: We plan to kick it into overdrive in the coming week to meet the deadline for alpha functionality. We have a pretty good framework right now; we just need to connect the front-end and backend.
Week 6	**Progress:** During our Friday meeting, I showed off the progress that I made on the website form. I converted all mentions of "weightclass" into "weightgsm" as the client requested, and added functionality for having a form element change when a checkbox is clicked. York showed off his progress with the API endpoints and how they accept POSTs from my site. **Problems**: There weren't any major problems, but our client did request that we create some diagrams about our project that we never got around to doing. Since we need those files for the design review next Thursday, we are going to complete them anyways. **Plans**: We are going to continue to push towards alpha functionality for the design review next Thursday. I am going to create a new page to display the job results from York's rules engine, and York is going to change his engine to accept the new format of input from my site. The results page will be fairly basic, but enough to show the output of the engine.
Week 7	**Progress:** During our Friday meeting, we discussed the design review, how it went, and what kinds of questions people were asking. We told our client that everything went well. Unfortunately, because of the rush to complete everything before the design review, we were not able to make much more progress on the website and backend (although progress was made). We also presented a few diagrams that we made for the design review. Our client told us what he wants us to do in the coming week, basically to continue implementing rules in the backend, and to add a spreadsheet to the job history view so that multiple jobs can be compared. **Problems**: The design review took up a lot of our focus this week, so we weren't able to get as much work done as we wanted. However, it's out of the way now, so it should be smooth sailing from here on out. **Plans**: We are going to continue to implement functionality in the rules engine, including starting to build a collection of different rules files. I will implement the job results page and continue to update the job history page so that jobs may be compared with one another.
Week 8	Forgot to do this one.
Week 9	**Progress:** During our Friday meeting, we discussed the changes we made to the output format of the rules engine, the addition of a Job Results screen that is automatically linked to upon New Job submission, and the changes made to the rules engine. Our clients seem very pleased with our progress so far, even going as far as calling it a "milestone release." After sharing our work, our clients outlined a list of features that they would like to see implemented in the front-end, and requested some code in the back-end be put into YML files instead of being coded in Java. Overall, everything went well. **Problems**: There were no problems this week, everything went well. **Plans**: I am going to add a column to the Job Results spreadsheet for the justifications (even though they don't exist yet) in addition to a checkbox that will allow users to toggle said column. I also want to add an export button to quickly dump the contents of the spreadsheet into a CSV file. York will work to move code out of Java and into the YML files, and will start to implement justifications for why the rules engine chose a specific output.

Week 10	**Progress:** Not a lot of progress was made this week. All group members were too busy finishing up projects due on dead week to make any meaningful progress. I was able to add an "export to CSV" button on the results page, but otherwise nothing else **Problems**: Not enough available time this week to work on the project. **Plans**: I plan on doing everything in the coming week what I said I was going to have done by today. After finals I'll have plenty of free time to make up this week.

7.2.2 Kuan-Yu Lai

Week 1	**Progress**: Meeting with our client and discuss about the future plan of this term. **Problems**: Haven't start the project so we don't have any problem this week. **Plans**: Generate GUI prototype before next meeting, this should also include the basic backend function.
Week 2	**Progress**: Both of us are nearly done with out type of the prototype, which mean 80% of the functionality of our parts are working. **Problems**: Our client will like to see the program coded in java related language because he says it good for long term usage. Therefore, we need to decide which java-related language to use. Probably javascript since we are both familiar with. **Plans**: Merge our prototype together, back-end and front-end, and finish the prototype.
Week 3	**Progress**: Our client specifies the input and output of the program so we adjust our own piece and trying to merge it together. **Problems**: I was very sick this week so I didn't make significant progress and can't participate the weekly meeting on Friday. My progress next week has to catch up the schedule. **Plans**: Merge our prototype together, back-end and front-end, and keep optimizing the rules engine.
Week 4	**Progress**: Almost done with the back-end server prototype. **Problems**: We planned to merge the website and the server together this week but due to the bugs in the back-end server, we weren't able to merge it together. **Plans**: Fix the back-end server bugs and merge our prototype together.
Week 5	**Progress**: Get both online servers working. **Problems**: Our client has time conflict with the weekly meeting so we have to discuss further detail on next Monday. **Plans**: Improving the accuracy of the rules engine.
Week 6	**Progress**: Finish the SJA Engine implemented by Node.js. **Problems**: There were a bit of communication mistake between me and Cole, so the front-end and back-end doesn't connect very well. **Plans**: Update the rules engine so that it can take the up-to-date input from the front-end.
Week 7	**Progress**: Update the input format of the rules engine and implement some more rules. **Problems**: Our rules engine is a bit buggy during design review but it ends up well. **Plans**: Implementing more rules, increase the accuracy of the rules engine.
Week 8	**Progress**: Implementing 80% of the rules given by the client. **Problems**: We don't have any problem this week. **Plans**: Implementing more rules send form the client.
Week 9	**Progress**: Implement all the rules from given from the client. **Problems**: We don't have any problem this week. **Plans**: Implement the new set of rules that the client will send us in the next couple days.
Week 10	**Progress**: Fixed the issue where the rules aren't implement in the right way. **Problems**: I accidentally missed the weekly meeting so I wasn't able to show my progress to the client. **Plans**: Implement as many rules as I can in the new ruleset that our client will send us over the weekend.

8 Final Poster

COLLEGE OF ENGINEERING | Electrical Engineering and Computer Science | CS 62

PROJECT INTRODUCTION

Having a chance to cooperate with a leading tech company in the industry is a valuable experience for us. Being able to work with the professional developer let us know how the product is developed and how the professional team members suppose be communicate.

The idea for this project arose out of the necessity to find a long-term solution to the difficulty of properly determining optimal printing press settings. Currently, determining the appropriate settings for HP's industrial printing presses is extremely complicated. It requires professional workers with specific training to be able to understand what settings are optimal for a given print job on a given press.

PROJECT OVERVIEW

With the advisor, regular employees will be able to quickly and efficiently append settings tickets to their print jobs through the use of a ReactJS-based web application.

Employees can specify some basic information about a print job, such as the type of paper to use, the desired quality mode, and the degree of density of information, and pass it off to a Java-based rules engine. The rules engine will process the data and determine the optimal press settings.

The settings advisor will not only recommend the optimal settings for a given job on a given press, but also provide justifications as to why each setting was selected.

Oregon State University

Printing Press Settings Advisor

Automating the selection of settings for large-scale printing jobs on million-dollar printing presses

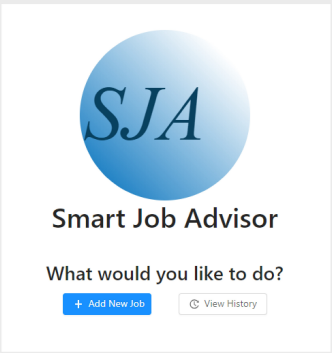


Figure 1: Website homepage

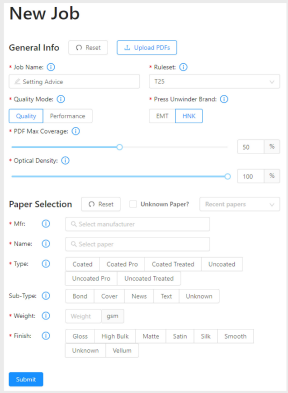


Figure 2: Job creation form

Results

When a user inputs job information in the job creation form and clicks submit, the Smart Job Advisor Engine passes the data to the Rules Engine.

The Rules Engine operates on the data, appends its output, and saves it to the filesystem.

The website queries the filesystem and displays the job results to the user.

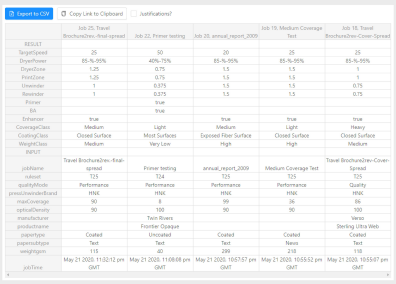


Figure 3: Comparison of previously-run print jobs

PROJECT DESCRIPTION

Our project is called the Smart Job Advisor. It's an program that will help the users of all four HP industrial printing presses decide optimal print job settings.

The front-end, was built with the javascript framework ReactJS. It is a powerful tool that operates quickly and efficiently with large datasets like ours.

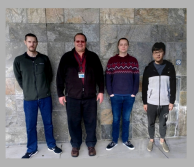
For the back-end, there are three different pieces: the Smart Job Advisor Engine, Filesystem, and the rules engine. Each piece is hosted on Amazon Web Services.

The Smart Job Advisor Engine is a RESTful API that the front-end uses to communicate with the back-end. It sends input data from the front-end to the rules engine, saves the output data from the rules engine and sends it to the filesystem.

The rules engine uses an open-source library called j-easy rules. The library allows us to build our own rules and filter the input based on those rules. The rules are based on research from HP.

The filesystem will act as a respository for the rules files used by the rules engine, as well as the history of all jobs that have been executed.

ACKNOWLEDGEMENTS



We would like to thank our clients, Pieter Van Zee and Ronald Tippetts, as well as all of the hard-working experts at HP for providing us with all the tools and information we need to complete this project.

CONTACT INFORMATION

Team members
Cole Jones
(jonecole@oregonstate.edu)
Kuan-Yu Lai (laik@oregonstate.edu)

Fig. 3: Final Poster for Engineering Expo

9 Project Documentation

9.1 How does the project work?

The Smart Job Advisor arose out of the need for a system that can cut out the middle-man, using a Java-based rules engine (Easy Rules) to provide optimal press settings based on a number of inputs about a print job, such as the type of paper, the ink coverage, whether or not it's in quality mode, what the press unwinder brand is, etc. With the advisor, regular employees will be able to quickly and efficiently append settings tickets to their print jobs through the use of a ReactJS-based web application. The advisor will not only recommend the optimal settings for a given job on a given press, but also provide justifications as to why each setting was selected.

Upon accessing the web tool, users are presented with a choice to either add a new job or view the job history. Choosing to add a new job brings them to a form that asks for some basic information about the job. When the user clicks submit, the information in the form is passed to the SJA Engine. The SJA Engine will then parse the input and send it to the Rules Engine. After the rules engine sends back the recommended settings to the SJA Engine, the SJA Engine will record the recommended settings into a file and send a job ID to the website. The job ID is used to look up the recommended settings for the job from the job history and display its contents to the user.

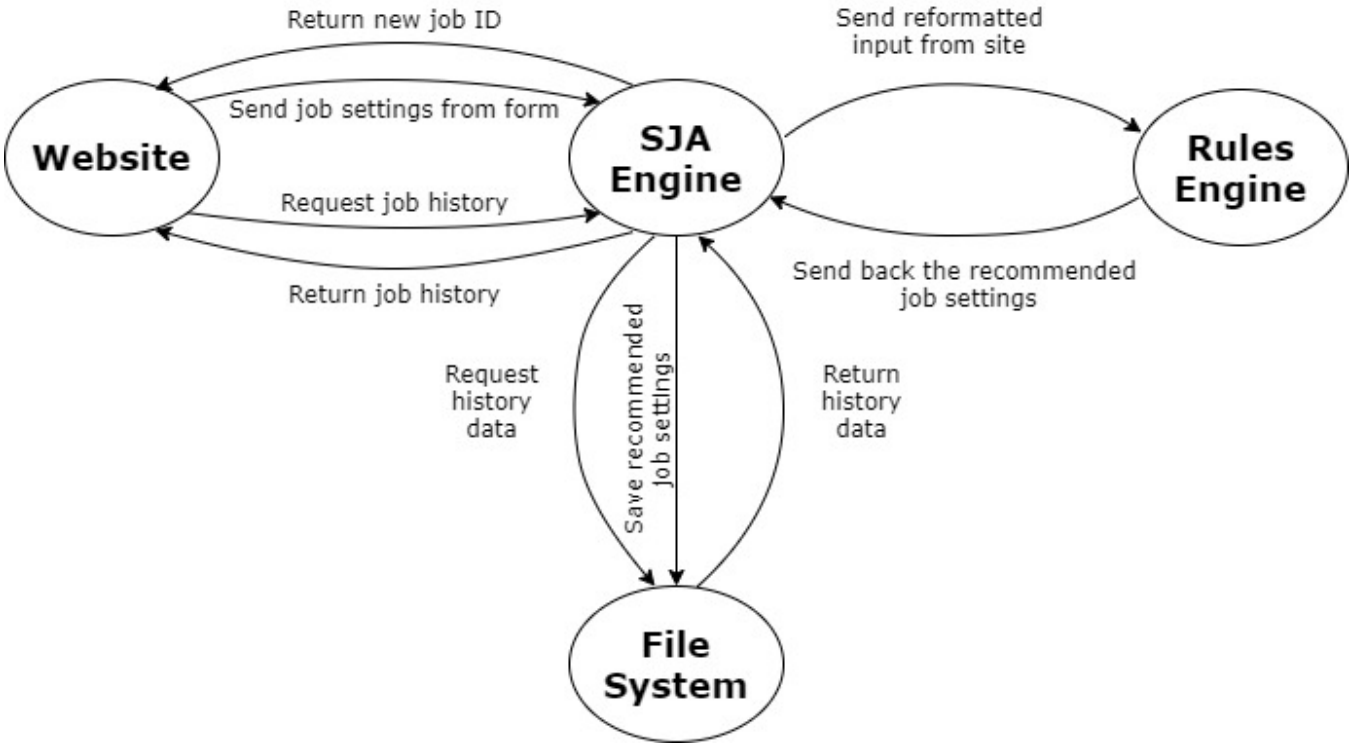


Fig. 4: Project workflow

9.2 Install Guide How to run it

First clone the repository from the Github https://github.com/KuanYuLai/Senior-Project. The code of the front-end is in the sub-folder of the **Website** folder called **SmartJobAdvisor**. The code of the back-end is in the **Back_End** folder. Please run the installation command in the corresponding folder.

9.2.1 Front-End

First, make sure you have Visual Studio 2019 installed to be able to render the site with IIS Express, **OLDER VERSIONS OF VISUAL STUDIO MAY NOT WORK!**

Then, make sure you have Node.js and npm installed. If you don't have Node.js, please download it from this website https://nodejs.org/en/. If you don't have npm, please download it from this website https://www.npmjs.com/get-npm

Now, install the node module with the command:

```
npm install
```

Move into the ClientApp folder:

```
cd ClientApp
```

Install the required node modules in ClientApp folder:

```
npm install
```

Open the **SmartJobAdvisor.sln** file with Visual Studio 2019

Render the site by clicking **IIS Express** at the top of VS (Green 'play' button)

Visual Studio should open a new tab in your browser and display the site.

9.2.2 Back-End

There are two servers in the Back-End which are the SJA Engine and the Rules Engine. The source code of each server is separated into different folder. Please run the installation command inside the corresponding folder.

9.2.2.1 SJA Engine

Before installing the SJA Engine, please make sure you have Node.js and NPM installed in your computer. If you don't have Node.js, please download it from this website https://nodejs.org/en/. If you don't have npm, please download it from this website https://www.npmjs.com/get-npm
First, run the command:

```
npm install
```

to install the required modules.

Then, run:

```
chmod +x eval.sh
```

to add execute permission to the required shell.

Now, start the server with the command:

```
npm start
```

You can now access the server with the URL localhost:8000

9.2.2.2 Rules Engine

First, make sure you have tomcat server installed. After that set up the file serving endpoint to serve the files in the ”rules” folder. Please follow this website to set up the file serviing endpoint https://www.moreofless.co.uk/static-content-web-pages-images-tomcat-outside-war/.

After set up the file serving endpoint. Change the URL in variable **host_url** to

```
localhost:8080/serving_folder_path_name/
```

in file **Launcher.java** and **Job.java**.

Now run the server with command:

```
mvn tomcat7:run
```

in the **Test_Project** folder and access it with the URL localhost:8080

9.3 User Guide

On the main page, click the **Add New Job** button to create a new job. Either upload a PDF to the website or select the **PDF Max Coverage** by yourself. The website can handle multiple PDF upload at once but the evaluation will be based on the same settings except for the **PDF Max Coverage**. After that choose the settings in the form. The paper selection section will filter the option after you choose the manufacturer. Only the supported option can be selected. Click the **Submit** button after all of the information is filled.

The website will jump to the job result page after the recommended settings are generated. You can use the recommended settings in the printing press and see the reason why the engine chose these values. Uncheck the **Justification** on top to toggle the description.

9.4 API Documentation

9.4.1 SJA Engine

GET

- **/paper-db**: Return the data of the paper database file.
- **/job-history**: Return all records of the documented settings.
- **/job-history/id**: Return specific documented settings.

POST

- **/new-job**: Takes the input of the user settings and forward to the rules engine. Send back the rules engine output with the given user settings. Both input and output are in JSON format.

DELETE

- **/job-history/remove/[id, id, ...]**: Take a list of jobID separated by comma and delete them.
- **/test/removeAll**: Remove all of the job history records. This endpoint is for testing purposes.

9.4.2 Rules Engine

POST

- **/new**: Takes the input from the SJA Engine and uses the given rules in the ruleset subject of the input to generate the result. The input is case-sensitive.

10 Recommended Technical Resources for Learning More

10.1 Helpful Websites

1) Ant Design Documentation — Excellent documentation for the front-end's UI framework, Ant Design. Just be sure to change the version (in the top right) from 4.X to 3.X, since we're using a slightly older (but still supported) version.
2) Apache Tomcat Documentation — Official document usually gives the most detailed instruction and explanation.
3) Amazon EC2 Documentation — Amazon's documentation covers any kind of situation you might face so it's very helpful when it comes to setting up their EC2 server.
4) Stack Overflow — As always, stackoverflow is immensely helpful for any CS-related questions (but you probably already knew that).
5) HP PageWide Presses — Useful for learning more about the HP printing presses for which we were developing this application.

10.2 Helpful Books

N/A

10.3 Helpful People

- Scott Fairbanks - Talked with our clients about machine-learning and its limitations. He helped narrow the scope of the project, communicate expectations to our clients (like how much time per week we should spend on the project), and restructure the project to better fit within our schedules and abilities.
- Prasad Tadepalli - Professor Tadepalli gave us helpful advice on whether we should use Machine Learning to solve our problem. He also compared the pros and cons of the Machine Learning and decision tree to us. In his opinion, machine learning is not the best solution because we want to be able to control the output.

11 Conclusions and Reflections

11.1 What technical information did you learn?

11.1.1 Cole Jones

I learned a good amount about how complicated the process of finding appropriate settings for an industrial printing press can be. I didn't know there were so many different variables involved in the process, like paper type, the chemical content of the paper, the desired speed at which the user wants the press to run, etc. All of this information was compiled by experts at HP, which showed me that there's a whole division at HP specifically for figuring stuff like this out.

11.1.2 Kuan-Yu Lai

I learned a lot of technical information in this project. First is, how to set up different kinds of servers using AWS and how to operate and maintain them. Second is, how complicated is the HP industrial printing press. The settings are hundreds of times more complicated than the home printer which requires a well-trained expert to operate a single printer. Last is, how to convert research results into the program.

11.2 What non-technical information did you learn?

11.2.1 Cole Jones

Over the course of this project, I was able to expand my skills with React and javascript. I'm sure that I know more about web design than I did before this capstone. I also learned a lot about the importance of documentation, and how to relay information that's in my head to another user,specifically someone who is not at all familiar with the project. Our clients taught me how to make swimlanes and other diagrams to explain the overall flow of the web application,something that proved useful over the course of the last three terms.

11.2.2 Kuan-Yu Lai

Communication and documentation are extremely important to make efficient progress, especially documentation. Since the project needs to be view by other people and our client, having clear documentation is very important. Also, the documentation can also keep track of your progress, makes sure everything is on track.

11.3 What have you learned about project work?

11.3.1 Cole Jones

I learned that the hardest part about project work is not the implementation itself, but choosing what technologies you want to use beforehand. At the beginning of this project, we had an idea for a technology we wanted to use, but as we uncovered more information about that project and what our clients wanted out of it, we changed technologies more than once.

11.3.2 Kuan-Yu Lai

I learned that a good plan leads to a good project. To generate a good plan, it requires deep research to find the best component of your solution. Then, assemble these components and generate a plan that helps you develop your project efficiently.

11.4 What have you learned about project management?

11.4.1 Cole Jones

I learned that time management is very important. Just because I had plenty of time to work on something didn't mean I was able to just put it aside until right before it needed to get done,because there were many times where my partner's work on the backend of the site required changes on the front-end. I learned that coordinating this work can be very difficult, especially when dealing with a 15 hour time difference. I also learned that it can be difficult to work without a plan, and that laying out a design for the interaction between the front and back-ends was helpful in keeping us on track while we worked on separate components.

11.4.2 Kuan-Yu Lai

Keep everything organize even at the beginning because it's hard to reorganize when the project becomes larger and larger. Also, be sure to use a version control system because you want to make sure your work won't break the previous working version of the project.

11.5 What have you learned about working in teams?

11.5.1 Cole Jones

I learned that even though you segment your work (I focused on the front-end website while my partner focused on the back-end control engine) you're never going to only focus on that partition. There were many times in which the way I designed my website influenced the way my partner needed to design the backend, and vice versa. I also learned that communication with a timely response is very important. There were a few times that I was late to act on a received message, much to the annoyance of my partner, who had to wait until I made a change to continue on his own work.

11.5.2 Kuan-Yu Lai

Lack of communication will waste a lot of time since different people will have different coding styles and ideas. When two programs are having conflicts, it takes a lot of time to merge it together. Also, you still have to understand what your teammate is doing even you are not responsible for it. This will be helpful when team members are debugging while they merge their work together.

11.6 If you could do it all over, what would you do differently?

11.6.1 Cole Jones

I would spend more time planning before implementing the code. Also, I would have skipped trying to work with machine-learning entirely, since we wasted a lot of time settling on the technology before we actually got around to working with it. We spent weeks getting Drools to work and learning about it, before quickly switching over to our current technology, JEasy Rules Engine. Since we had spent all of Fall term figuring this out, if we had skipped right to JEasy we would have had several more weeks to work on the project.

11.6.2 Kuan-Yu Lai

The documentation part and the timeline will be the two main things that I want to improve. Document what you change and what feature of your program is developed and working can save a lot of time for us. As for the timeline, I often underestimate the time I need to implement the feature. This cause the team to push our schedule later several times. Therefore, estimating the time I need precisely will make the project development progress smoother.

APPENDIX A

ESSENTIAL CODE LISTINGS

A.1 Front-End

```
/* Fetch the paper database from the server. */
fetchPaperDatabase = async () => {
    /* Call database to request paperDatabase object. */
    await fetch(ServerURL + "paper-db/", {
        method: "GET",
        mode: 'cors',
        headers: {
            'Accept': 'application/json',
        }
    }).then(async (res) => {
        await res.json().then((data) => {
            this.setState({ paperDatabase: data.paperdb });
        });
    }).catch(err => {
        this.fetchError("fetch paper database");
    });
}
```

Fig. 5: The function that fetches the paper database from the back-end

```
<Table
    rowKey="jobID"
    rowSelection={rowSelection}
    dataSource={tableData}
    columns={tableColumns}
    style={{ width: tableWidth, maxWidth: '100%' }}
    scroll={{ y: '60vh', x: tableWidth - 100 }}
    pagination={{ defaultPageSize: 10, showQuickJumper: true, showSizeChanger: true, pageSizeOptions: ['10', '20', '30'] }}
    size={windowWidth <= 500 ? "small" : "large"}
    bordered
    onRow={(record, rowIndex) => {
        /* Allows rows to be selected by clicking on them,
         * instead of having to click the rowselection checkbox. */
        return {
            onClick: () => {
                var tempRowKeys = [...this.state.selectedRowKeys];

                if (tempRowKeys.indexOf(record.jobID) === -1)
                    tempRowKeys.push(record.jobID)
                else
                    tempRowKeys.splice(tempRowKeys.indexOf(record.jobID), 1);

                this.setState({ selectedRowKeys: tempRowKeys });
                this.forceUpdate();
            }
        }
    }}
/>
```

Fig. 6: The JSX that generates a table with click-based row selection

```
/* Gathers and validates form data, then makes a POST call to the rules engine. */
handleSubmit = e => {
    e.preventDefault();
    this.props.form.validateFields(async (err, values) => {
        if (!err) {
            /* This is here so the values of the form can be seen in the console for debugging. */
            console.log('Received values of form: ', values);

            /* Force weightgsm to be an int on the off chance it's passed as a string. */
            values.weightgsm = parseInt(values.weightgsm);

            /* Set cookies for next time. */
            this.setCookies(values);

            /* Open the modal for submitting files, only if files were added. */
            if (this.state.fileList.length > 0) {
                this.setState({ formValues: values }, () => {
                    this.setState({ showSubmitModal: true });
                });
            } else {
                /* Call database to post form data. */
                await fetch(ServerURL + 'new-job/', {
                    method: 'POST',
                    mode: 'cors',
                    body: JSON.stringify(values),
                    headers: {
                        'Accept': 'application/json',
                        'Content-Type': 'application/json',
                    }
                }).then(async (res) => {
                    await res.json().then((data) => {
                        this.setState({
                            createdID: data.id,
                            jobCreated: true
                        });
                    });
                }).catch(() => {
                    this.fetchError("submit job");
                });
            }
        }
    });
};
```

Fig. 7: The function that handles POSTing the job to the back-end

```
<Row gutter={20}>
    <Col span={12}>
        <Form.Item
            style={{ marginBottom: -5 }}
            label={
                <>
                    <span>Quality Mode:</span>
                    <Popover content={qualityModeInfo} title="Quality Mode" placement="bottom">
                        <Icon style={{ fontSize: 18, color: 'dodgerblue', position: 'relative', left: 8, bottom: -2 }} type="info-circle" />
                    </Popover>
                </>
            }
        >
            {getFieldDecorator('qualityMode', {
                rules: [{ required: true }],
                initialValue: cookies.get('qualityMode') || "Quality",
            })(
                <Radio.Group className={Style.formItemPaper}>
                    <Radio.Button value="Quality">Quality</Radio.Button>
                    <Radio.Button value="Performance">Performance</Radio.Button>
                </Radio.Group>
            )}
        </Form.Item>
    </Col>
    <Col span={12}>
        <Form.Item
            style={{ marginBottom: -5 }}
            label={
                <>
                    <span>Press Unwinder Brand:</span>
                    <Popover content={pressUnwinderBrandInfo} title="Press Unwinder Brand" placement="bottom">
                        <Icon style={{ fontSize: 18, color: 'dodgerblue', position: 'relative', left: 8, bottom: -2 }} type="info-circle" />
                    </Popover>
                </>
            }
        >
            {getFieldDecorator('pressUnwinderBrand', {
                rules: [{ required: true }],
                initialValue: cookies.get('pressUnwinderBrand') || "EMT",
            })(
                <Radio.Group className={Style.formItemPaper}>
                    <Radio.Button value="EMT">EMT</Radio.Button>
                    <Radio.Button value="HNK">HNK</Radio.Button>
                </Radio.Group>
            )}
        </Form.Item>
    </Col>
</Row>
```

Fig. 8: An example of the layout of a form item

```
/* Second function called in componentDidMount().
 * Generates form data and appends file. POSTs the formdata to the /analyze-pdf endpoint.
 * Saves returned maxCoverage value and sets status flag for job to indicate completion of analysis. */
analyzePDF = async (index) => {
    const { fileList, fileStatus, fileCoverage } = this.state;

    /* Generate a form data and apply the file with 'pdf' tag. */
    const formData = new FormData();
    formData.append('pdf', fileList[index]);

    /* POST the file to the /analyze-pdf endpoint, capture returned value. */
    await fetch(ServerURL + 'analyze-pdf/', {
        method: 'POST',
        mode: 'cors',
        body: formData,
    }).then(async (res) => {
        await res.json().then((data) => {
            /* Make copies of current state arrays. */
            let tempCoverage = [...fileCoverage];
            let tempStatus = [...fileStatus];

            /* Capture coverage value and set status tag in copied arrays. */
            tempCoverage[index] = parseInt(data.Coverage);
            tempStatus[index] = 1;

            /* Save the changes to the actual state arrays. */
            this.setState({
                fileCoverage: tempCoverage,
                fileStatus: tempStatus,
            });
        });
    }).catch((err) => {
        console.log("ERROR: ", err)
    });

    /* Rebuild modal content to refresh view and update status. */
    this.buildModalContent();
}
```

Fig. 9: The function that sends the user's PDF(s) to the back-end to analyze and obtain a maximum ink coverage value

A.2 Back-End

```
//Parse the input from the front-end
function output_parser(input){
        var basic_rule = ["job-rules", "paper-rule"];
        const T24 = ["BA-rule", "primer-rule"];
        const T25 = ["Enhancer-rule"];
        var json = {};

        console.log("Name:" + input.jobName + " Ruleset: " + input.ruleset);

        //Building output data object
        json.jobName = input.jobName;
        json.qualityMode = input.qualityMode;
        json.CoverageSize= input.maxCoverage;
        json.opticalDensity= input.opticalDensity;
        json.paperType = input.papertype;
        json.papersubType = input.papersubtype;
        json.weightgsm = input.weightgsm;
        json.finish = input.finish;
        json.pressUnwinderBrand = input.pressUnwinderBrand;
        json.ruleclass = input.ruleset;
        json.ruleset = new Array();

        //Decide what files to fire in the rules engine
        if (input.ruleset == "T24")
            basic_rule = basic_rule.concat(T24);
        else if (input.ruleset == "T25")
            basic_rule = basic_rule.concat(T25);

        json.ruleset = json.ruleset.concat(basic_rule);

        return json;
}
```

Fig. 10: The function parse the input from the front-end into what rules engine wants

```
//Evaluating PDF by executing shell command
function PDF_evaluation(callback){
    console.log("Start evaluating PDF");
    execFile('./eval.sh', ['upload/pdf', 'grey'], (err, stdout, stderr) => {
        if (err){
            console.log("Error occur");
            callback(err);
        }
        console.log("Finish evaluating");
        console.log(`stdout: ${stdout}`);
        callback(stdout);
    });
}
```

Fig. 11: The function handle the evaluation of the uploaded PDF file

APPENDIX B

WEBSITE PHOTOS

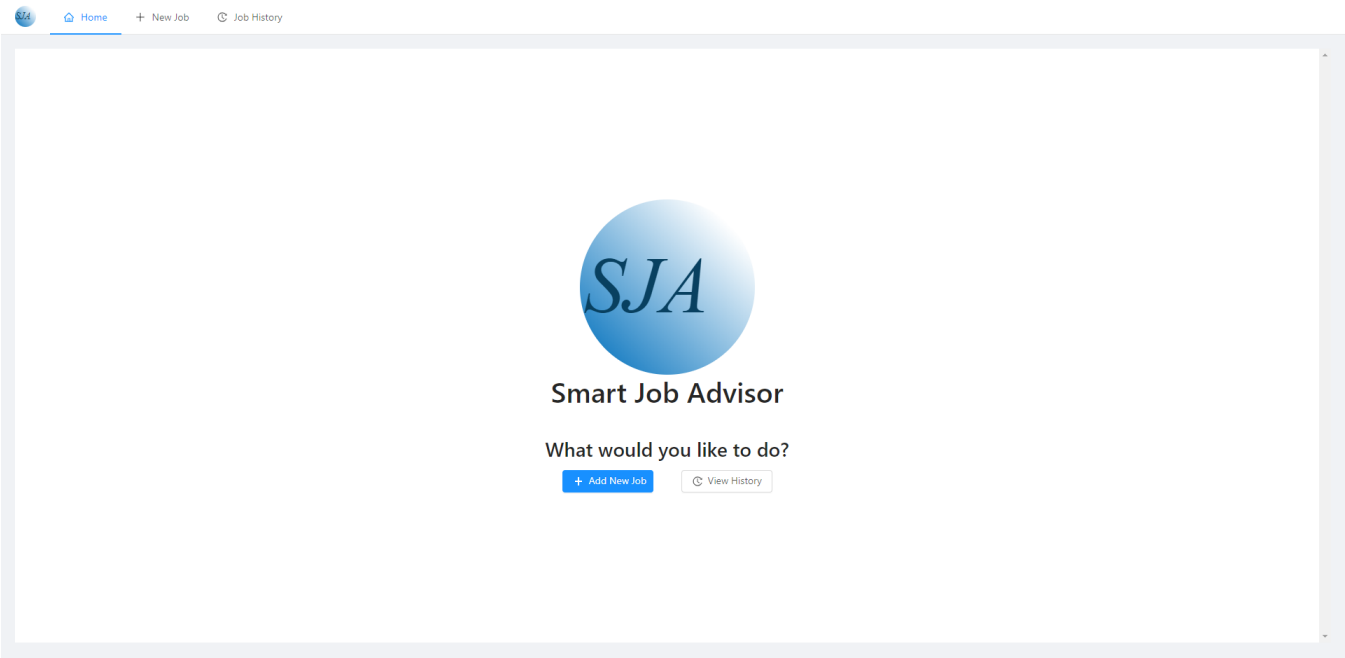


Fig. 12: Main page of the SJA website

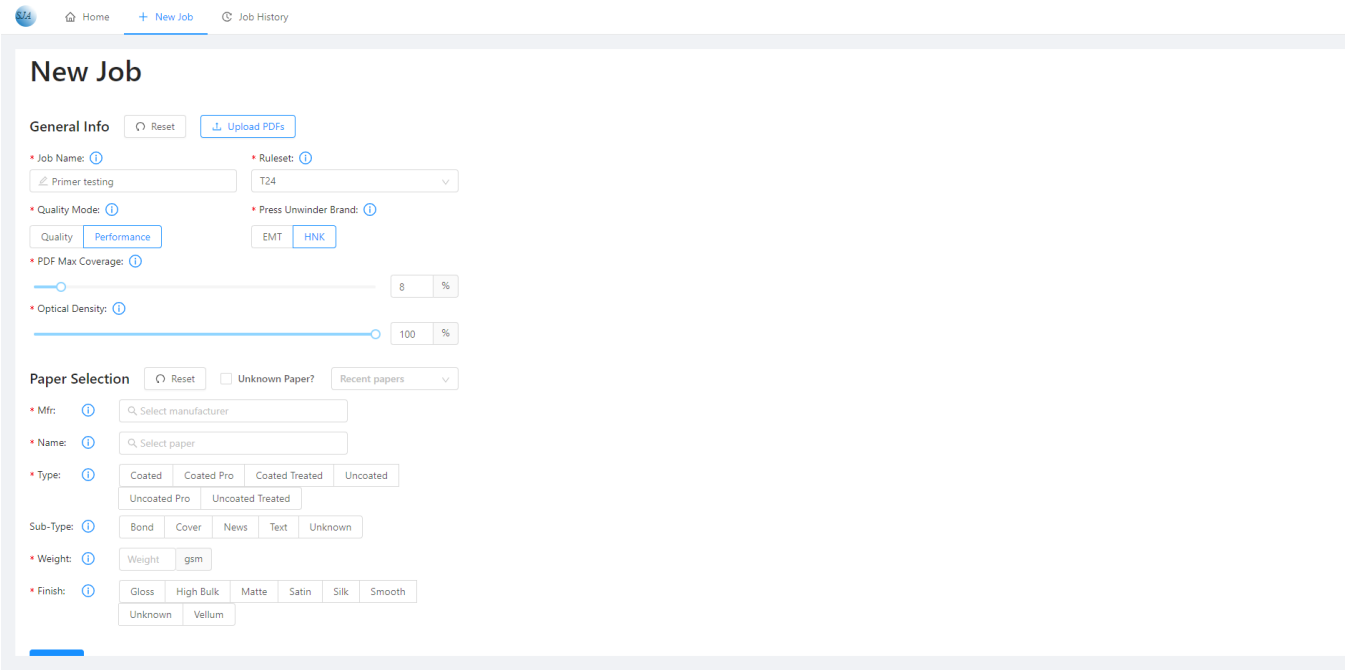
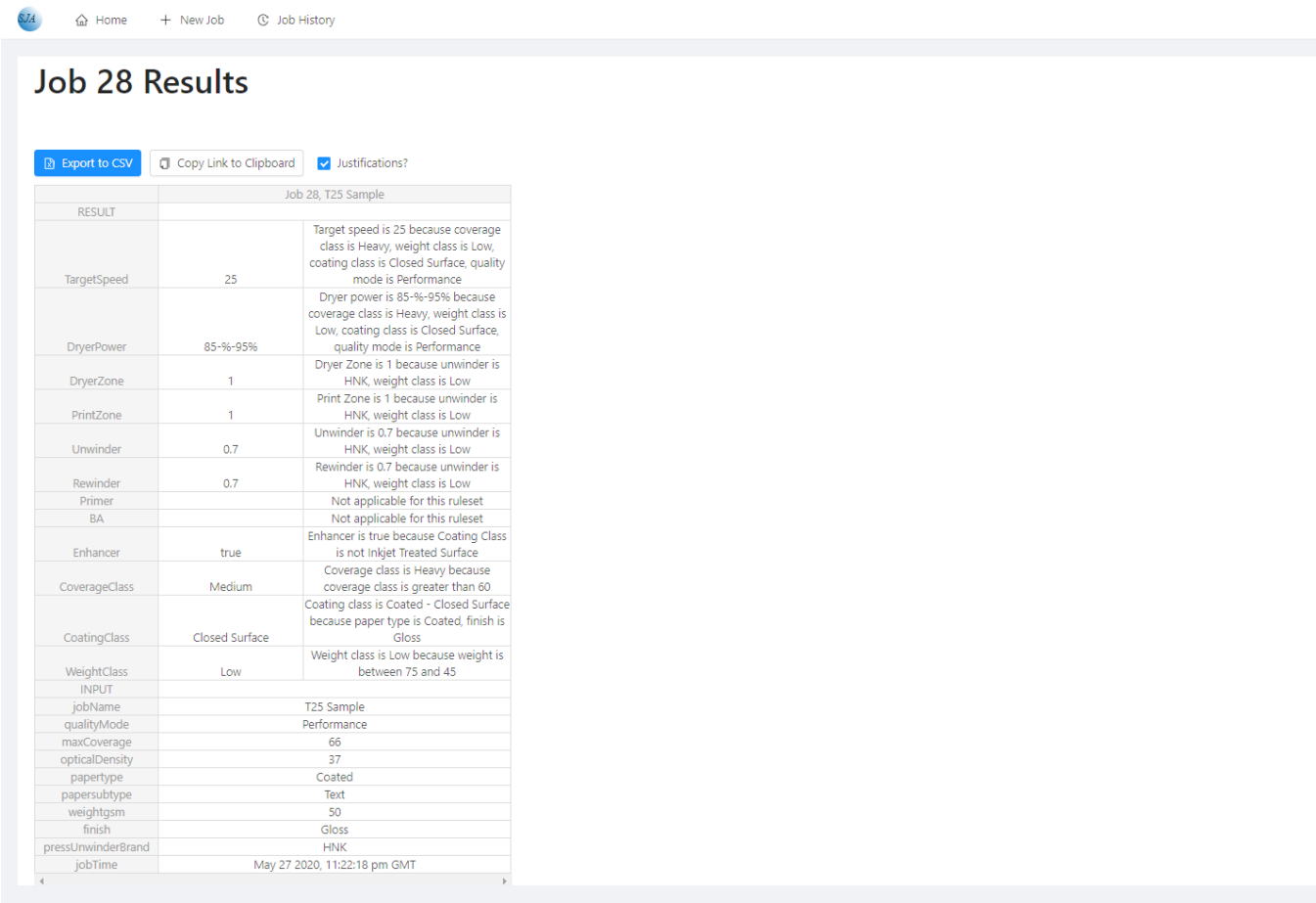


Fig. 13: Job page of the SJA website



	Job 28, T25 Sample	
RESULT		
TargetSpeed	25	Target speed is 25 because coverage class is Heavy, weight class is Low, coating class is Closed Surface, quality mode is Performance
DryerPower	85-%-95%	Dryer power is 85-%-95% because coverage class is Heavy, weight class is Low, coating class is Closed Surface, quality mode is Performance
DryerZone	1	Dryer Zone is 1 because unwinder is HNK, weight class is Low
PrintZone	1	Print Zone is 1 because unwinder is HNK, weight class is Low
Unwinder	0.7	Unwinder is 0.7 because unwinder is HNK, weight class is Low
Rewinder	0.7	Rewinder is 0.7 because unwinder is HNK, weight class is Low
Primer		Not applicable for this ruleset
BA		Not applicable for this ruleset
Enhancer	true	Enhancer is true because Coating Class is not Inkjet Treated Surface
CoverageClass	Medium	Coverage class is Heavy because coverage class is greater than 60
CoatingClass	Closed Surface	Coating class is Coated - Closed Surface because paper type is Coated, finish is Gloss
WeightClass	Low	Weight class is Low because weight is between 75 and 45
INPUT		
jobName	T25 Sample	
qualityMode	Performance	
maxCoverage	66	
opticalDensity	37	
papertype	Coated	
papersubtype	Text	
weightgsm	50	
finish	Gloss	
pressUnwinderBrand	HNK	
jobTime	May 27 2020, 11:22:18 pm GMT	

Fig. 14: Job results page of the SJA website

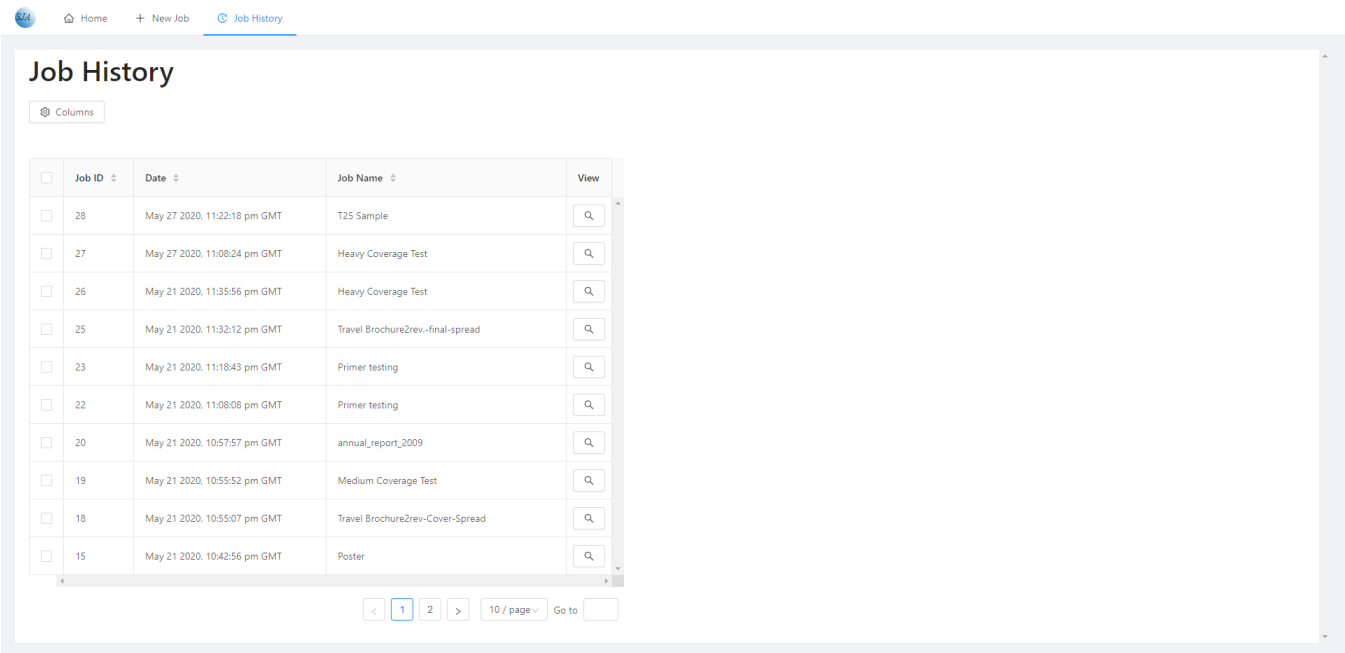


Fig. 15: History page of the SJA website

APPENDIX C

RESPONSE TO CODE REVIEW CRITICISMS

C.1 Front-End

Criticism	Action
"You need an about page to teach the user how to use the site."	Instead of adding an about page, info popups were added to the New Job form to explain what each field means. I think this is enough, since the users of the site are going to have a vague idea of what all the options mean.
"I would like to see more comments..."	Many comments were added to the code.
"I did not see unit tests."	We determined that unit testing was not worth our time, although this is something we may explore between the code freeze and the expo, just for our clients
"The presenters said that they are waiting for a new ruleset from their employer, so it seems like once that is handled the requirements will be met."	New ruleset was given to us by our client and fully implemented.
"It's possible that refactoring could be done, because there are some very long functions."	Some code refactoring was done to make functions more efficient.
"I am sure there are possible ways to improve the code, but I think the current react code is efficient enough that the marginal benefit of rewriting it is not worth the extra time."	Some code refactoring was done to make functions more efficient. Some of the more complicated functions were not rewritten because of time constraints.
"I did not see any unit tests for the react code. However, I am not a strong believer in unit tests for react code as I don't really see the benefit in it, so I wouldn't recommend the team spend time on it anyways."	Unit tests were not added for the UI. This is something we considered doing for our clients after the code freeze, but I did not think it was worth the time to do it before then.
"In JobResults.js, in the fetchJob function, it uses both Async/Await and .then(). I know that's really picky, but it really stuck out to me."	Attempted to fix this by only using Async/Await or .then(), but it only works with both. Without Async/Await, the react code moves along before the function can finish. Without .then(), the API call moves along before it finishes. I decided to leave the code as-is, since it doesn't really hurt anything to use both Async/Await and .then().

C.2 Back-End

Criticism	Action
”The names of variables and classes seemed very explicit in what they meant. I would like to see more comments, especially in the more complicated parts of the backend code.”	More comments added for both back-end servers.
”I saw some sections of the code labeled “old”. I’m assuming the comment label means these bits of code are no longer useful, so they should be taken out.”	Old code removed after the code clean up.
”I did not see unit tests. During the presentation Postman was used and I saw a lot of tests, but there did not seem to be any structure among them. I would recommend exporting a collection of Postman tests and storing it on the repo for other people to use to test the program.”	Sample Postman test case added in the ReadMe file.
”The presenters said that they are waiting for a new ruleset from their employer, so it seems like once that is handled the requirements will be met.”	New ruleset file updated.
”I could not find any obvious abstractions, or a way to write functionally equivalent code. Its possible that refactoring could be done, because there are some very long functions.”	Helper functions added to reduce the duplicated code.
”Setting up tomcat was a challenge. I know I need a tomcat file servelet. The ReadMe for the RulesEngine states I must “move all the Json and YML files to that folder to serve it to the server”. What files must I copy into the /static folder for tomcat? That is the part I am stuck on.”	Update the instruction to make it more clear.