Feasibility Study and Plan

CS5150: Software Engineering
Cornell University
Ithaca, NY

Under the guidance of
Prof. William Arms
Department of Computer Science

Clients
Rajiv Kumar, Product Owner
Abhijnan Saha, Architect

Ayush Soni (aas427), Gaurav Bang (gsb86), Harshit Manchanda (hm545),
Mukul Shukla (ms3528), Rohith MD (rm977), Shubham Agrawal (sa2279),
Shubham Khandelwal (sk3266), Vidush Vishwanath (vv259)

Table of Contents

Executive Summary

The Bills of Material (BOM) sector is one of the core topics of Product Lifecycle Management area at SAP. The BOM consists of a unique header defining the product and a list of items used to create that product in the manufacturing industry. An example is as follows:

```
Coffee Cup                           -----------> Product (also called Header)
        |--> Plastic Lid             }
        |--> Plastic Straw           }
        |--> Transparent Container   } -----> Items (used to create the product)
        |--> Logo Design             }
        |--> Logo Color              }
```

In the manufacturing industry, there often arises scenarios where one material needs to be replaced with another (Eg: Plastic Lid → Cardboard Lid) due to many factors like stock, cost, location, demand etc. The industries must be ready in these cases to have a supplement back up for a quick replacement so that they do not lose millions of dollars worth of business. Hence arises the need for the BOM-Where Used application. For a brief overview, this application finds all the BOMs where a particular item is used. Similar to the Google search feature, this application too provides a search on items and returns a list of Products/Headers where the item is used as a component. Typical industry scenario returns millions of BOMs and manual replacement of them would take years to complete and hence the mass replacement feature is enabled in the application where the user just checks all the BOMs and replace them in one shot. This functionality already exists.

Our algorithm will advance this framework by taking in the history of customer data and various other features (cost, texture, color, availability, region, etc.) and finally suggesting the best material that could be used to replace the given material by employing Machine Learning. Often in manufacturing industries, the core decision of replacement depends on an uncountable number of factors. To predict the best replacement material is the core idea of our implementation. Additionally, we would try to implement a sequence of best replacements in a hierarchical order.

Clients

Rajiv Kumar (rajiv.kumar01@sap.com): Product Owner
Abhijnan Saha (abhijnan2004@gmail.com): Architect

Resources

Team Composition

The team is proposed to comprise of a Scrum Master, 2 Software Developers in Test (QA team), 3 Full Stack Software Engineers and 2 Machine Learning Engineers.

Front End Technology

As the product requires to demonstrate the ranking of proposed replacement materials with their ranking, we will demonstrate that with a web page built on HTML, CSS, JavaScript as the front-end technologies.

Back End Technology

The backend will comprise of building Python API's that handle the data requests and have the underlying Machine Learning model which analyses the data and sends back the predicted data. The machine learning models will be built using python open source data science and ML libraries such as Scikit-Learn, Numpy, Pandas, and Tensorflow. The data will be hosted on PostgreSQL. Also for version control, we have all the deployment via a GitHub repository.

Where is it hosted? - Min startup cost (Cloud provider)

As the company has its own cloud infrastructure on top of which they want to deploy the application, the development of the product will be based on what the guidelines that are given out by the client. Thus, the cost of deployment should be minimal from our end.

Testing and Release lifecycle

The code will be regularly tested by the Quality Assurance team feature by feature once delivered from the software developers on a bi-weekly basis and after testing, detailed analysis will be sent back for review. Only when given approval from QA it will be sent across to be deployed in the main application. To maintain different versions of the product the whole team will have access to the product GitHub repository. We aim to deliver the product in two versions and the release targets are:
Version 1: Mid March
Version 2: Mid April

Benefits

The project aims to provide the clients of SAP to have quick, better and data-driven recommendations, which will help them to apply to replace material tasks with better efficiency. It will provide SAP, our client, with additional leverage as a new feature in their Bills of Material (BOM) platform. This will reduce the costs and time of material replacement by SAP customers and better profit margins.

Scope

In this project, we will follow the following process for building a machine learning product.

Data Preparation

Since the data is in raw format, we have to pre-process and filter the data. After processing, we convert into a format such that we can easily extract and experiment with new features from the data.

Feature Engineering

To build a machine learning model, we shall be using features such as the recency of change numbers, the current/previous trend in the cost of material, the frequency of previous use, place of manufacture, availability in the current location, etc. Note that the features are not final and are subject to change depending upon the performance, only to serve as an example for the purpose of this report. After the data is prepared, we extract these features and vectorize them as required to build the ML model.

Model Selection

We will experiment with different machine learning algorithms such as Logistic Regression, Naive Bayes, Support Vector Machines, Bagged Decision Trees, and Deep Learning. We will also explore recommender system techniques like collaborative filtering and content based filtering.

Hyperparameter Tuning

After we choose a model, we need to tune different parameters such as learning rate for gradient descent, depth of decision trees, hidden layers of the neural network, etc. We will use techniques like grid search to find the best set of hyperparameters which gives us the best performance on our machine learning models.

Deployment

Once the model is tested and ready, we will embed the model in the application, and it will serve recommendations in real time.

Requirements - Technical/Functional/Non-Functional

Technical Requirements

- **Cloud compatible system -** The existing application in the firm for handling Bills of Materials is already using their proprietary cloud. We require our recommendation system to be compatible with the existing cloud in the firm.
- **Client Agnostic Recommendation -** The system to be developed is to be client independent however when applied the client and domain-specific context has to be taken into consideration for providing recommendations.
- **API calls/second -** The recommendation system supports API calls at a maximum of 50 calls/second, based on the computational efficiency of the underlying machine learning algorithm.
- **Dynamic feature addition -** The recommendation system is to be flexible to incorporate any feature dynamically, as required by the client to work well with the existing machine learning algorithm.
- **Memory/CPU Utilization threshold -** 16 GB RAM, 8 core processor, 1.5 to 2.5 GHz clock speed
- **API Payload limit -** The RESTful calls should support 1 MB based JSON size.

Functional Requirements

- **Performance** – The recommendation system API's response time would more generally depend on the computational efficiency of the underlying machine learning algorithms. The estimated response time should be around 20 to 100 ms, throughput should be around 10-20 per second.
- **Scalability -** The recommendation system should be scalable enough to handle around 20 (estimated) API calls and should support calls from all the different client users simultaneously.

- **Availability -** The recommendation system should be available to the users 24X7 and should be well functioning during the time. The maintenance for the same should also be timely handled.
- **Reliability -** We intend to make the recommendation system using the client specific data and we intend to make sure that the machine learning algorithms using this data are well tested to provide reliable and accurate information. We also tend to make sure that the web application is reliable enough to handle payload bursts and memory crash.
- **Recoverability -** We intend to make sure that our system is to be recoverable such that in case of any crash, the underlying system is still consistent. We tend to use similar database interaction APIs like the ones used in the existing application of the firm which also ensures recoverability.
- **Maintainability -** The system should be maintained regularly. Any complaints, crashes or API call failures should be well handled as soon as possible.
- **Security -** The API should be accessible only via the firm's user login authentication and the recommendation system is to be developed in a way that the data of different clients are completely isolated from one another.
- **Manageability -** Manageability includes API health monitoring, logging, alert notifications, new version deployments, and updates. Once the system is deployed, the clients will report issues in form of tickets to the firm.
- **Data Integrity** - We make sure that the client specific data (provided, created and used) will not be disclosed in any form to any external party.
- **Usability** - The end-user clients are supposed to know how to use the API once the system is deployed. The recommendation system basically provides an alternative to the product displayed on the dashboard on the basis of core features of the product and the preferences (learned using ML algorithms) of the end user.

Non-Functional Requirements

- **Hardware and Software:** To develop the system we require 8 machines (PCs for each of the team members supporting Linux or Mac or Windows Operating System) along with code editors (Like sublime text or Eclipse for Java and some python IDEs), JVMs, PostgreSQL database servers (3 at least).
- **Databases:** We intend to use the PostgreSQL database to create and support our data servers.
- **Programming Languages:** We intend to use Java and Python programming languages for developing the system.

Viability

We have a development time of about 2 months. The major work is accumulating, cleaning, validating the data - then testing various ML algorithms. We also have to validate the results, choose the best algorithm and tune it.

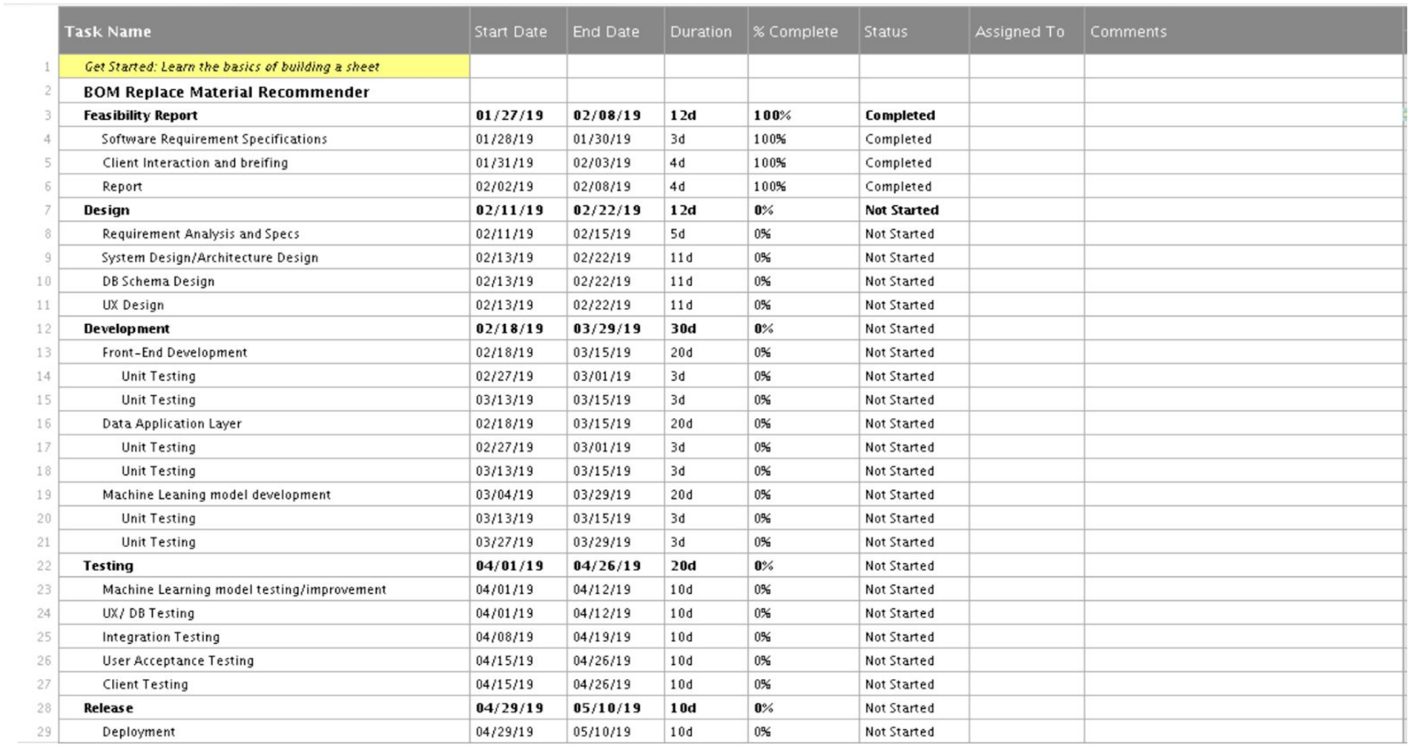
All of the above major tasks must be accomplished in a continuous integration environment. We must also perform regular testing cycles, adopt a streamlined peer review process.

Once the model is ready, we must ensure that the integration with the existing product is seamless. The APIs exposed must conform to the given scalability and performance threshold.

The team voted about the feasibility and various constraints that we have to follow. We are confident about implementing the given technology in the said timelines.

Process & Visibility Plan

As employed and followed by most companies which have been proven effective, the team would follow the AGILE methodology. An Agile Method is a particular approach to project management that is utilized in software development. This method assists teams in responding to the unpredictability of constructing software. It uses incremental, iterative work sequences that are commonly known as sprints to the unpredictability of constructing software. The team would have internal biweekly planning and review meetings scheduled on Monday and Friday of a 2-week sprint with updates every 3 days. The meeting would be scheduled every Saturday with the client starting next week for iterative guidance and learning in the process. Every 2-week sprint would be justified by deliverables as allocated and assigned amongst the team members. The commitment of the number of days taken per task per person is free will within the 2-week sprint. The team would also be responsible for writing their own unit test cases for the deliverable promised at the end of the sprint. As a safety precaution, every team member would stop developing new functionalities by Wednesday of the second week and the remaining two days would be used just to test the deliverable built for that sprint. The testing will be done among the team members and by those who were not involved in the implementation. A designated scrum master would be assigned to monitor, schedule and book the regular updates from the team members. The scrum master would work closely for a “de facto” Product Owner and the Architect of the team to assign, track and guide the deliverables over the entire project. The main Product Owner would be Mr. Rajiv Kumar (SAP), the architect would be Mr. Abhijnan Saha (SAP) and the project guide would be Prof. William Y Arms (Cornell).



Task Name	Start Date	End Date	Duration	% Complete	Status	Assigned To	Comments
Get Started: Learn the basics of building a sheet							
BOM Replace Material Recommender							
Feasibility Report	**01/27/19**	**02/08/19**	**12d**	**100%**	**Completed**		
Software Requirement Specifications	01/28/19	01/30/19	3d	100%	Completed		
Client Interaction and breifing	01/31/19	02/03/19	4d	100%	Completed		
Report	02/02/19	02/08/19	4d	100%	Completed		
Design	**02/11/19**	**02/22/19**	**12d**	**0%**	**Not Started**		
Requirement Analysis and Specs	02/11/19	02/15/19	5d	0%	Not Started		
System Design/Architecture Design	02/13/19	02/22/19	11d	0%	Not Started		
DB Schema Design	02/13/19	02/22/19	11d	0%	Not Started		
UX Design	02/13/19	02/22/19	11d	0%	Not Started		
Development	**02/18/19**	**03/29/19**	**30d**	**0%**	Not Started		
Front-End Development	02/18/19	03/15/19	20d	0%	Not Started		
Unit Testing	02/27/19	03/01/19	3d	0%	Not Started		
Unit Testing	03/13/19	03/15/19	3d	0%	Not Started		
Data Application Layer	02/18/19	03/15/19	20d	0%	Not Started		
Unit Testing	02/27/19	03/01/19	3d	0%	Not Started		
Unit Testing	03/13/19	03/15/19	3d	0%	Not Started		
Machine Leaning model development	03/04/19	03/29/19	20d	0%	Not Started		
Unit Testing	03/13/19	03/15/19	3d	0%	Not Started		
Unit Testing	03/27/19	03/29/19	3d	0%	Not Started		
Testing	**04/01/19**	**04/26/19**	**20d**	**0%**	Not Started		
Machine Learning model testing/improvement	04/01/19	04/12/19	10d	0%	Not Started		
UX/ DB Testing	04/01/19	04/12/19	10d	0%	Not Started		
Integration Testing	04/08/19	04/19/19	10d	0%	Not Started		
User Acceptance Testing	04/15/19	04/26/19	10d	0%	Not Started		
Client Testing	04/15/19	04/26/19	10d	0%	Not Started		
Release	**04/29/19**	**05/10/19**	**10d**	**0%**	Not Started		
Deployment	04/29/19	05/10/19	10d	0%	Not Started		

Fig 1: Project Overview, Deliverables, Milestones and Tasks split by days

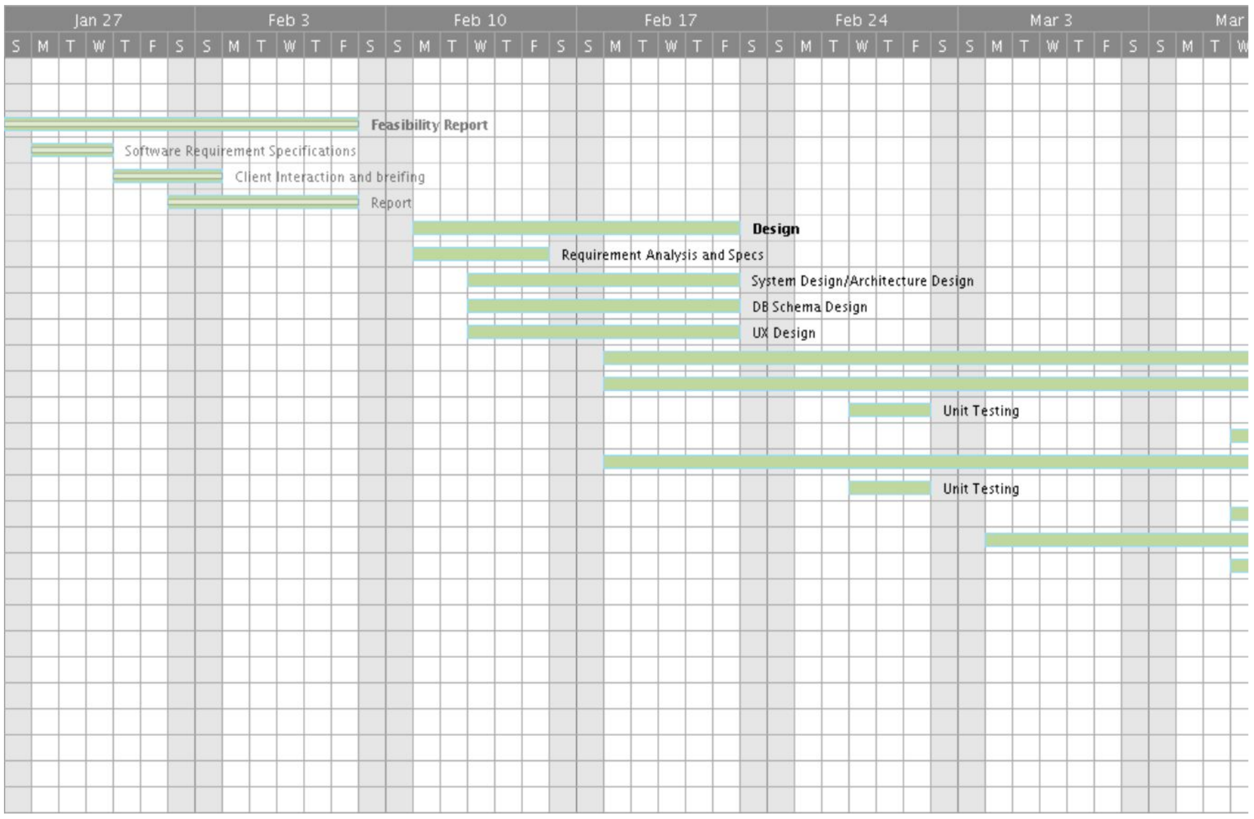


Fig 2: Feasibility Report, Design and Development

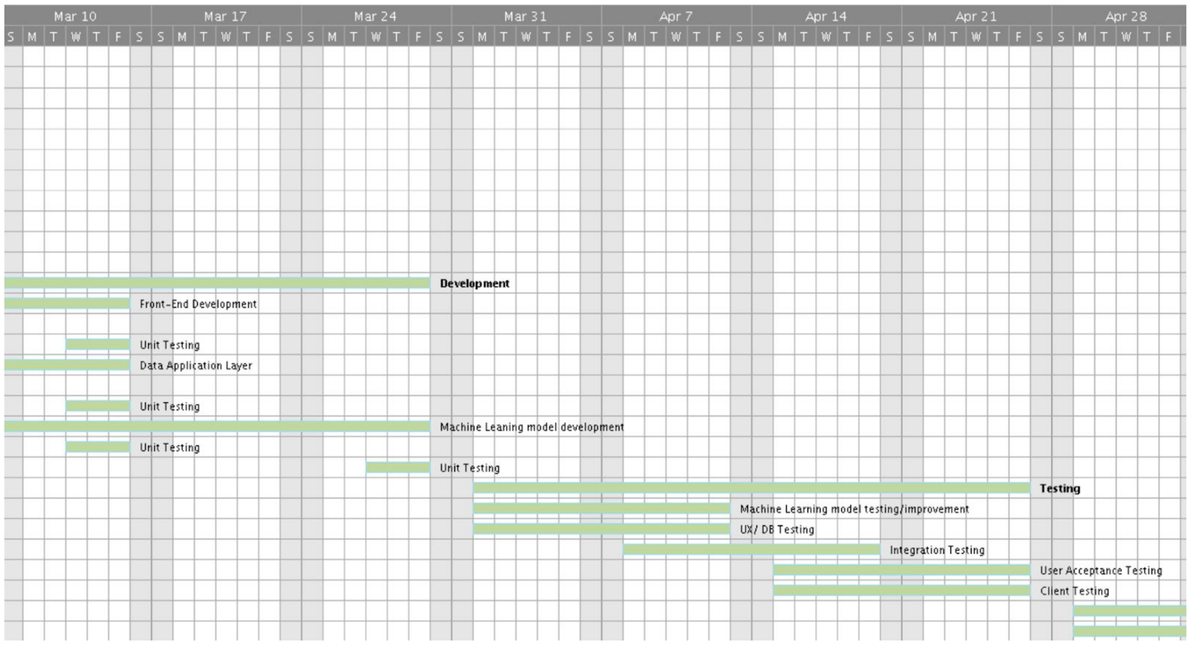


Fig 3: Development, Testing and Deployment

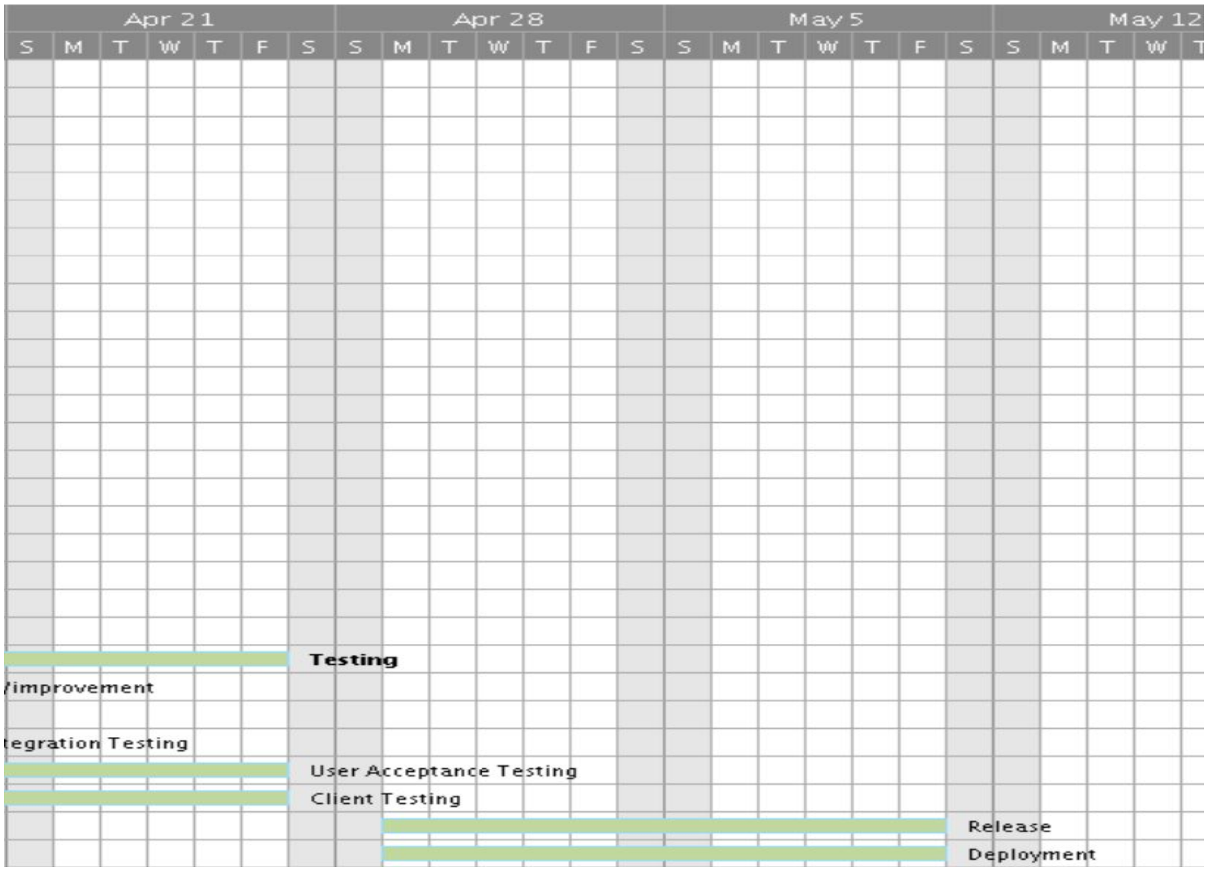


Fig 4: Testing and Deployment

Deliverables

Non-technical

The different aspects of non-technical deliverables required to complete the project are listed below:

1) **Scrum Meetings** - Have scrum meetings every 3-4 days to discuss the progress of different aspects of the projects and to discuss future deliverables.
2) **Client Meetings** - Discussion with client every week/fortnight to discuss the system integration specifications, code quality standard, expected accuracy etc.
3) **Written Documentation** - This aspect involves any documentation expected by the client and Software Engineering course reports (like the Feasibility Report, Progress Report, Presentation, and Demonstration).
4) **Architecture Diagram** - The architecture diagram would be a representation of the different components required in the pipeline. It would go through various iterations subject to the requirement of the client, cloud integration and implementation standard.
5) **Software Requirements Specification** - The document would fully describe the expected behavior of the end-product, helping to make sure that the client's expectations are consistent with the product in development. Both functional and nonfunctional requirements such as performance goals and description of quality attributes shall be included in the SRS.

Technical

The main technical deliverables revolve around the 3 components of the project which are Front-End Design/Machine Learning model/Database Design. On a topic level scope, the main technical deliverables are as listed below :

1) **Front-End:** Tool Kits,SDKs,Forms
2) **Machine Learning:** ML models with the best accuracies
3) **Back-End:** Database tables on PostgreSQL

The most important thing to note is that the clients would migrate the database and the front end into the inhouse technologies. The Machine Learning models would be delivered and used as a plugin in the existing software.

High-Level Architecture

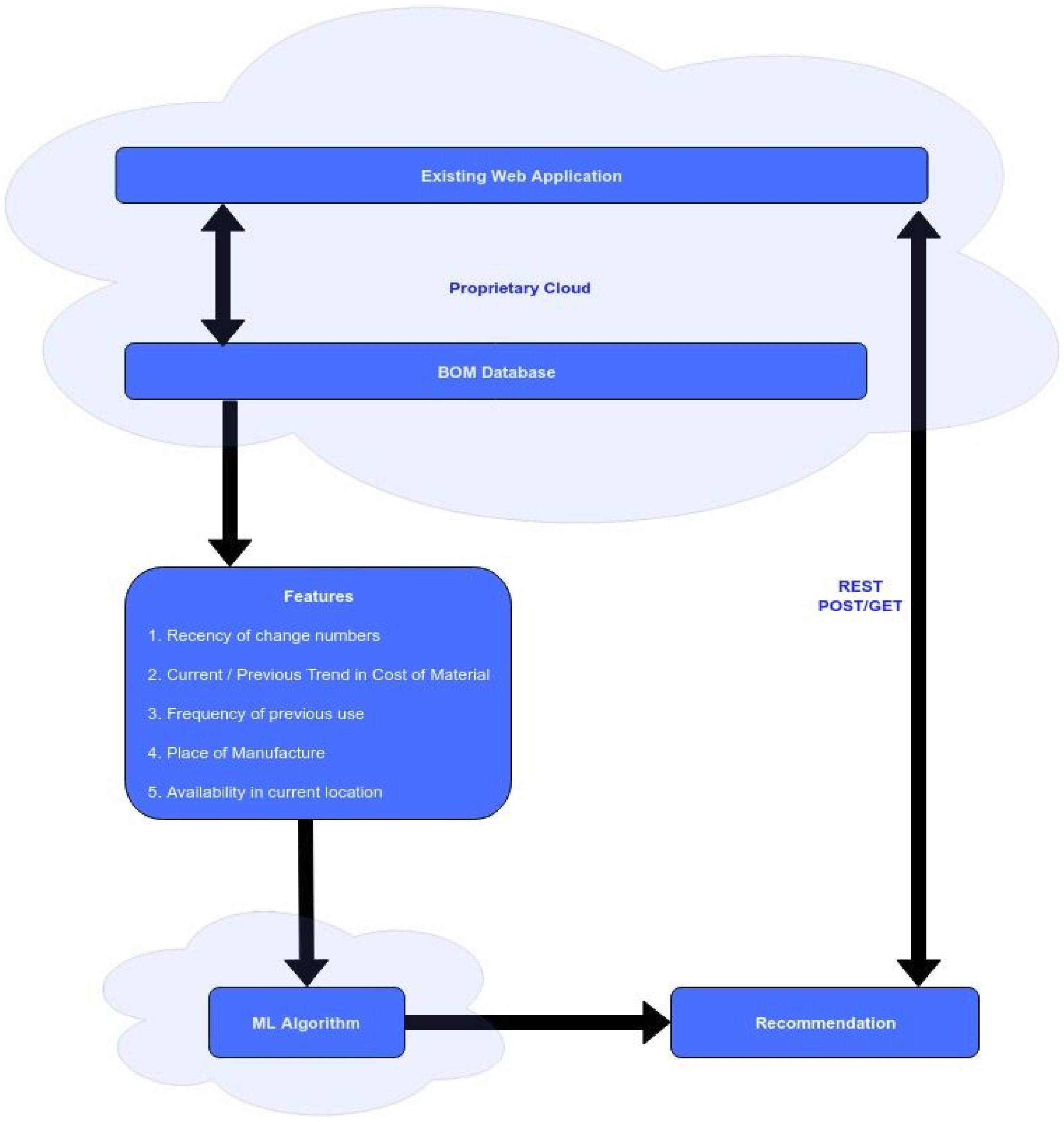

Existing Web Application
Proprietary Cloud
BOM Database
Features
1. Recency of change numbers
2. Current / Previous Trend in Cost of Material
3. Frequency of previous use
4. Place of Manufacture
5. Availability in current location
REST
POST/GET
ML Algorithm
Recommendation

Fig 5: High Level Architecture diagram

Risk Analysis

Several possible risks associated with the project have been brainstormed and are outlined below. Additionally, a path for risk mitigation, which will be followed is provided for each outlined risk.

Volatile Requirements

During the course of this project, it is possible that requirements may change as our product comes into fruition.

Risk Mitigation: To reduce the risk associated with changing requirements, we will ensure that contact with the client is maintained on a weekly basis and periodic check-ins are performed. Furthermore, the agile software development process will be used in an iterative fashion. This way, we will have a product to present to the client relatively early on and will understand the client's concerns & feedback and be able to respond accordingly.

Integration with Larger System

It is expected that this system will be incorporated into a larger, prebuilt system by the client. This poses potential risks with respect to the scale of data that the project must operate on, which may vary significantly between development by the team and use by the client. Furthermore, the clients end-users consist of potentially several different organizations. It is important that all end-users can apply the methods in this project successfully.

Risk Mitigation: In order to ensure that the developed project scales with large amounts of data, we may have the clients run a preliminary version of our product. If such a version performs up to expectations, further developments may be iteratively tested as well. This way, the end product is guaranteed to perform at levels of expectation. To allow several different organizations to use the product easily, we will obtain assurance from the client that all end-users for the product will have input-data that is identical in format. With such assurance, the methods in this project should be interchangeable between end-user organizations.

Access to reliable data

In order to comply with confidentiality requirements provided by the client, we will not have access to the data that this framework will be expected to run and produce results on.

Risk Mitigation: The first part of this project will involve creating a reliable dataset to mimic what the framework will be expected to run on. It is a hard requirement that the format of our self-made dataset and that of the final dataset that the algorithm runs on is exactly the same. After the development of the dataset, it will be sent to the client for verification. Subsequent development of the system will be pursued only after the client has provided written assurance of the consistency between the test dataset and the self-made dataset.

Limited Time

Perhaps the hardest requirement of this project is the time requirement. It is absolutely imperative that all parts of the project including design, coding, testing, and client feedback are complete by the end of the semester.

Risk Mitigation: Although it is not possible to increase the time required for the project, we will minimize this risk through adhering as tightly as possible to the process guidelines (in the Process section).

Business Consideration

Copyright:

For the following paragraphs, the term "the team" refers to the entity formed by all eight individuals in this project as listed on page 1 of the feasibility report.

It is understood that after the completion of this project, the team will provide an unrestricted copyright license to the client. This license covers all use of the product, integration into a larger system as deemed by the client, and the ability to further develop the product as the client sees fit. The team reserves the right to showcase the project as their own work in ways including, but not limited to presenting demos of the project, listing the project on resumes, & showing the project to future prospective employers.

It is not expected that any trademark will be developed during the course of this project.

Trade Secrets

There is no expected exchange of trade secrets between the client and the team. In order to maintain the confidentiality of the client's data and end-user data, the team will not use any client provided data or methods. Therefore, all methods necessary for this project will be developed by the members of the teams themselves.

Patents

It is not expected that any work performed for this project will be eligible for a patent. However, if such an unforeseen opportunity arises in the future, the team reserves all rights to the intellectual property and subsequent patents from the work completed within the scope of this project.

Alternatives Algorithms

A recommendation/ranking algorithm will be used to perform the core task of finding the best possible replacement of a given item. The algorithm is expected to produce a ranked list of items, with the items higher up on the list (or the ones that rank best) expected to be the best replacement of the item under consideration. The following are the algorithms that could tentatively be used for the task of ranking/recommendation:

1) **Logistic Regression**: Logistic regression, (usually used for classification) is often used in machine-learned ranking. The algorithm learns from the training data by assigning weights to features, where the feature weights in a way tell about the importance of that particular feature (the magnitude of the weight tells about how much the feature influences the score of the item, and the sign of the weight tells whether the feature positively or negatively affects with the score). Once the model is trained, the test data is scored by using the previously trained weights to perform a dot product operation with the feature values to obtain a score. The sample with the highest score is ranked best, while the lowest score is ranked worst.
   Mathematically, logistic regression makes the assumption that the distribution of the label given the features takes the exponential form. The algorithm performs well with a significant amount of data but is prone to overfitting when the amount of data is less. Some hyperparameters to control the algorithm are class weights and regularization parameter.
2) **Random Forest**: A Random forest or in other words is a bagged decision tree. It is also quite popular due to its easy hyperparameters optimization. It has two parameters namely the depth which is set to √dimension_of_features. In fact, they don't require any preprocessing and their values can be of any units. So, for the case of Bill of Materials, this method becomes highly relevant and as the model works independently of the company and its measurement units. Also, it is fairly easy to train and has standard hyperparameters with respect to any company.
3) **Collaborative Filtering**: It is a technique which is used mainly for item-item recommender systems with sparse datasets. Often only certain parameters might be relevant while comparing two items and hence this method becomes useful. This is especially true in-case of Bill of Materials wherein we might only want to compare features where the customer has valid values and common means of comparison between items. It is mainly based on techniques like cosine similarity and then ranking based mean, median or any other ranking algorithm.

Future Consideration

If the above functionalities are delivered as promised well ahead of time with the approval from the client, the team would like to take up another similar requirement from the same client. Similar to recommending the best replace material, another requirement is to recommend an alternative to a material. The database for the item (cardboard cup) stores information for "Alternative Material". For more in-depth clarification, replace material is functionality that replaces a particular item across BOMs. For example: (Plastic cup ---> Cardboard cup). Note that the plastic cup as an item in the BOM is being replaced with a cardboard cup. On similar lines, the second requirement goes in depth per item. For example: the cardboard cup can be in itself made from many different types of cardboard (light, dark, thick, thin, polystyrene etc). Using this field, the team would like to create a similar algorithm to determine the best alternative material that can be used.

Conclusion

The entire process of working with a client through different stages along with thorough documentation should give us a good idea of a real-world software engineering project and its requirements. The documentation not only serves as a running guide to the project but shall also help to understand the details to anyone alien to the current working group. This would make it easy for future enhancements in the project.

This feasibility report acts as written documentation of the thorough discussion with the client company, their expectations and our deliverables (technical and non-technical).