



A project report on

AI-Enabled FinTech B2B Order Management Application

Submitted in partial fulfillment of the requirements for the Degree of

B.TECH in Computer Science & Engineering

by

Swagato Bag (1805537)

under the guidance of

Sweta Pati & Soham Das

School of Computer Engineering

Kalinga Institute of Industrial Technology

Deemed to be University

Bhubaneswar

May 2021



CERTIFICATE

This is to certify that the project report entitled “**AI-Enabled FinTech B2B Order Management Application**” submitted by:

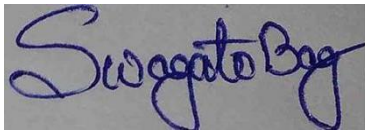
Swagato Bag (1805537)

in partial fulfillment of the requirements for the award of the **Degree of Bachelor of Technology in Discipline of Engineering** is a bonafide record of the work carried out under our guidance from 11th January 2021 to 17th March 2021 and supervision at the School of Computer Engineering, Kalinga Institute of Industrial Technology, Deemed to be University.

The Project was evaluated by us on 04.02.21 (Milestone 1) and 23.03.21 (Milestone 2)

ACKNOWLEDGEMENT

I feel immense pleasure and feel privileged in expressing our deepest and most sincere gratitude to our supervisors **Sweta Pati and Soham Das**, for their excellent guidance throughout our project work. Their kindness, dedication, hard work, and attention to detail have been great inspirations to us. Our heartfelt thanks to you for the unlimited support and patience shown to us. We would particularly like to thank them for all their help in patiently and carefully correcting all our manuscripts. We are also very thankful to **Dr. Jagannath Singh**, Project coordinator (School Of Computer Science & System Engineering), **Dr. Amulya Ratna Swain** Associate Dean (School Of Computer Science Engineering), and **Dr. Bhabani Shankar Prasad Mishra**, Dean (School Of Computer Science Engineering) for their support and suggestions during our course of the project work in the final year of our undergraduate course.

A handwritten signature in blue ink that reads "Swagato Bag". The signature is written in a cursive, flowing style.

Swagato Bag (1805537)

ABSTRACT

The B2B world operates differently from the B2C or C2C world. Businesses work with other businesses on credit. When a buyer business orders something from the seller business, the seller business issues an invoice for that order. This invoice contains various information like the details of the goods purchased and when they should be paid.

Seller business interacts with various businesses and sells goods to all of them at various times. Hence, the seller business needs to keep track of the total amount it owes from all the buyers. This involves keeping track of all invoices from all the buyers. The buyer business needs to clear its amount due before the due date.

However, in real-world scenarios, the invoices are not always cleared i.e. paid in full amount by the due date.

So my project should help the Account Receivables team by collecting payments from customers for their past dues, sending reminders and follow-ups to the customers for payments to be made, and help the company get paid for the services and products supplied.

TABLE OF CONTENTS

	Page
Abstract	4
Table of Contents	5
List of Figures	6
List of Tables	7
CHAPTER 1: INTRODUCTION	8
1.1 AI-Enabled FinTech B2B Order Management Application	8
1.1.1 Introduction	8
1.1.2 Overview	10
1.1.3 Problem Definition	10
1.1.4 Purpose	10
1.1.5 Scope	10
1.1.6 Definition, Acronyms & Abbreviations	11
CHAPTER 2: BACKGROUND/BASIC CONCEPTS	12
2.1 Basic Idea of the project	12
2.1.1 Domain Covered	12
2.1.2 Technology Used	14
2.1.3 The idea behind the project	18
CHAPTER3: PROJECT ANALYSIS/ PROJECT IMPLEMENTATION	19
3.1 Project Implementation	19
CHAPTER 4: RESULTS AND DISCUSSION	30
4.1 Result	30
CHAPTER 5: CONCLUSION & FURTHER WORK	39
5.1 Conclusion	39
5.2 Future Work	39
5.3 Planning And Project Management	39
REFERENCES	41
INDIVIDUAL CONTRIBUTION REPORT	42

LIST OF FIGURES

Figure ID	Figure Title	Page
1.1	Working of a random forest algorithm	15
1.2	Working of Linear Regression	16
1.3	Working of Decision Tree	16
1.4	Working of SVM	17
1.5	Working of XGBoost Regressor	17
1.6	Connection of JDBC with Database with the help of Java	18
1.7	Working of ML Model	21
1.8	The average delay for customers from different business codes	22
1.9	Total_open_amount distribution curve	23
1.10	Delay Distribution Curve	23
1.11	Log Transformation of total open amount curve	24
1.12	scatterplot between total_open_amount and target_value	25
1.13	Heatmap of train dataset	26
1.14	Workflow of Java and ReactJS	27
1.15	ML Accuracy Result	30
1.16	Bucketization	31
1.17	SQLyog	31
1.18	Servlet Mapping	32
1.19	Main UI	33
1.20	Active Buttons	34
1.21	Add Order Button	34
1.22	View Correspondence Button	35
1.23	Edit Button	35
1.24	Delete Button	36
1.25	Predict Button	37
1.26	Working of Flask Server	38

1.27	Gaant Chart for project planning and management	40
------	---	----

LIST OF TABLES

Table ID	Table Title	Page
1.1.6	Definitions, Acronyms, & Abbreviations	11
3.1.1.2	Data and Pre-Processing	21
5.3	Planning And Project Management	35

CHAPTER 1

INTRODUCTION

1.1 AI-Enabled FinTech B2B Order Management Application

1.1.1 INTRODUCTION

An Order Management System is a computer software program that allows businesses to manage their orders and inventory. The Order Management system ensures accurate inventory management by automating the process of entering new inventory. It also tracks sales from various e-commerce platforms such as eBay, Amazon, etc, and informs the owner of the stock of a particular product stoop below the threshold. It automates the entire process of order to cash i.e. order placement, payment reconciliation, fulfillment, and shipment. It can work for both B2B and B2C worlds.

Order management software is also shareable, from the customer service team to the accounting team, the warehouse staff, and the business owner. The best kinds of inventory management systems also have an order management app, allowing you to manage your stock on the go and spot-check your business in real-time.

Effective order management improves the business workflow and increases the likelihood of repeat customers.

1.1.1.1 Order Management System

The Order Management system goes through 6 steps to ensure accurate process and fulfillment of orders placed by the customer. It covers every aspect starting from entering invoices to sales follow-up. The seamless flow of the software ensures faster delivery and reduces the chances of error.

- **The order is placed:** The customer places an order through a third-party sales site, or the owner's website, or over the phone with a live representative. Online, your customers will enter their details on a uniform form, with a choice to have a securely saved preferred payment method. To improve the sales process, make all fields of your online form mandatory so that you've got all the required contact information for the customer up-front. This creates a customer profile and allows your order management system to trace their purchase history, the number of orders, and payment and delivery preferences. This process ensures that the phob=ne number or email id is saved for any follow-up processes or service recovery. The payment is processed, then the order is shipped to the warehouse once your software approves

the fees.

- **Warehouse processing:** Once the order arrives at the warehouse, it's checked by the intake team and therefore the item or items are "picked" from the stock. Having an SKU and barcode for each item increases the accuracy of fulfillment and makes it easier for pickers to easily scan the item and add it to the order.

If there's not enough of the item(s) available to satisfy an order, then a sale order is automatically placed through the order management software. You and therefore the warehouse manager will receive an alert that there could also be a delay in fulfillment. Your customer may receive automatic notification of the delay, and therefore the customer service team can follow up together with your customer.

- **Reconciling the order:** Next, the order is shipped to the accounting department or preferably it should sync automatically together with cloud accounting software, where it's recorded in the A/R ledger. The sale is logged and a receipt is sent to your client. Automating sales ledgers eases the process of auditing, end-of-the-year taxes, and inventory reconciliation.
- **Shipping the order:** As the order is picked from the warehouse the packing team will check for any errors using barcode and SKU. Thereafter, the order is packed carefully and shipped via a third-party delivery system. The customer receives further communication from the order management system that informs the tracking number and estimated delivery time as the order gets shipped. As a store owner, you can also track the progress of shipped orders, which can be helpful if there are special needs orders, such as re-deliveries, VIP orders, or unusually large ones.
- **Post-sales follow-up:** Upon the arrival of the order the system generates an email asking for a review of the product. This email also contains detailed information about how to reach customer service in case of any issues. This helps to avoid the frustration of customers regarding refunds or other issues.

The customer service team oversees this process, thanking the customer for their business or working with them for a refund or replacement.

- **Special order oversight:** The characteristics of a good order management system also include the ability to flag a special order. This may be a return replacement or it could be a VIP order that includes a free thank-you gift or special coupon. If such orders are placed through the system the software flags it with a special code. Thus the customer retention team can monitor the delivery of the product closely for better

accuracy.

1.1.2 OVERVIEW

In this given project we were required to make a full-stack Order Management application backed by a Machine Learning Model to predict the date of payment and aging bucket of an invoice.

1.1.3 PROBLEM DEFINITION

The problem is to create an application for managing B2B order invoices. We have to build a Machine Learning Model to predict the payment date of an invoice when it gets created in the system and categorize the invoice into different buckets based on the predicted payment date, and this whole process should be managed by a beautiful UI.

1.1.4 PURPOSE

In real-world scenarios, the invoices are not always cleared i.e. paid in full amount by the due date. So the purpose of my project is to help the Account Receivables team by collecting payments from customers for their past dues, sending reminders and follow-ups to the customers for payments to be made, and help the company get paid for the services and products supplied

1.1.5 SCOPE

This includes making an ML model, then making a beautiful UI, so that we could add, edit, delete records from the database, and with the help of the ML model, we can predict the payment date.

1.1.6 DEFINITIONS, ACRONYMS, AND ABBREVIATIONS

<u>Term</u>	<u>Description</u>
SPA	Single Page Application
RAD	Rapid Application Development
API	Application Programming Interface
GUI	Graphical User Interface

CHAPTER 2

BACKGROUND

2.1 Basic Idea of the Project

2.1.1 Domain Covered

2.1.1.1 Machine Learning

Machine learning (ML) is defined as the study of computer algorithms that improve their performance and results automatically through long-term experience and by the use of data sets. ML is a subset of artificial intelligence (AI).

Machine learning algorithms build a model with the help of "training data", to make predictions, decisions, or analyses without being explicitly programmed rather, by learning a pattern in the given data. Machine learning can be used in the medical field, filtering emails, and computer vision, where it is very difficult to predict the result or develop conventional algorithms to perform the required tasks.

Data mining is additionally a related field of study, and this focuses on exploratory data analysis through unsupervised learning. In its business application problems, machine learning is also termed predictive analytics.

2.1.1.2 Java

Java is primarily used for internet-based applications. It's an easy and efficient general-purpose language. Java may be an object-oriented, high-level, robust, and secure programming language. Java includes class, objects, inheritance, etc which help within the enhancement of the coding structure and make it flexible and extensible for the programmer.

Why we use Java?

It is easy to learn and simple to use and platform-independent. It is secure, fast and powerful, and open-source, and free as well.

- **Java OOP Concepts:** Object means a real-world entity like a pen, chair, table, computer, watch, etc. Object-Oriented Programming may be a methodology or paradigm to style a program using classes and objects. It simplifies software development and maintenance by providing some concepts:

- | | |
|---------------|-----------------|
| ✧ Object | ✧ Polymorphism |
| ✧ Class | ✧ Abstraction |
| ✧ Inheritance | ✧ Encapsulation |

Any entity that has a state and behavior is understood as an object. For instance, a chair, pen, table, keyboard, bike, etc. It is often physical or logical.

- **Object:** An Object is often defined as an instance of a category. An object contains an address and takes up some space in memory. Objects can communicate without knowing the small print of every other's data or code. The sole necessary thing is the sort of message accepted and therefore the sort of response returned by the objects.

Example: A dog is an object because it has states like color, name, breed, etc. also as behaviors like wagging the tail, barking, eating, etc.

- **Class:** The collection of objects is named class. it's a logical entity. A class also can be defined as a blueprint from which you'll create a private object. Class doesn't consume any space.
- **Inheritance:** When one object acquires all the properties and behaviors of a parent object, it is known as inheritance. It provides code reusability. It is used to achieve runtime polymorphism.
- **Polymorphism:** If one task is performed in different ways, it is known as polymorphism. For example: to convince the customer differently, to draw something, for example, shape, triangle, rectangle, etc.

In Java, we use method overloading and method overriding to achieve polymorphism. Another example can be to speak something; for example, a cat speaks meow, dog barks woof, etc.

- **Abstraction:** Hiding internal details and showing functionality is known as abstraction. For example phone calls, we don't know the internal processing. In Java, we use abstract class and interface to achieve abstraction.
- **Encapsulation:** Binding (or wrapping) code and data together into a single unit are known as encapsulation. For example, a capsule is wrapped with different medicine. A java class is an example of encapsulation. Java bean is the fully encapsulated class because all the data members are private here.
- **Servlet:** This technology is used to create a web application (resides on the server-side and generates a dynamic web page). This technology is robust and scalable because of the java language. There are many interfaces and classes in the Servlet API such as Servlet, GenericServlet, HttpServlet, ServletRequest, ServletResponse, etc.
- **Collections:** Any group of individual objects which are represented as a single unit is known as the collection of the objects. The Collection interface (java.util.Collection)

and Map interface(`java.util.Map`) are the two main“root” interfaces of Java collection classes.

2.1.1.3 ReactJS

ReactJS is an open-source and front-end JavaScript library that is used in large demand. It is mainly used for building user interfaces, especially for SPA. React also allows users to create reusable UI components. Due to its simplicity and flexibility, it is largely preferred by the full stack developer community and is known as the future of web development.

2.1.2 Technology Used

2.1.2.1 Python

Python is termed as a high-level, object-oriented, interpreted programming language with dynamic semantics. It is a dynamic typing language that has built-in data structures, along with dynamic binding. It is used as a scripting language that can connect the existing components. Thus python makes it extremely attractive for RAD. Python also supports immense modules and packages, which helps in program modularity and code reusability. The Python interpreter and the standard libraries are available in source or binary form for all the major platforms like their original websites and numerous 3rd party libraries in GitHub, and thus can be freely distributed. Python has a very easy and simple syntax, which enhances and emphasizes the readability of the code.

2.1.2.2 What are the Python Libraries?

A module is a file with some Python codes in it, and a package is a directory for sub-packages and modules. A Python library is a reusable bunch of code that we can include in our program projects. If we Compare python to languages like C++ or C, Python libraries do not relate to any particular context in Python. Here, a ‘library’ describes a collection of core modules. A library is a collection of modules. A package is a library that can be installed using a package installer like RubyGems or npm. Python Standard Library The Python Standard Library may be a collection of tangible syntax, tokens, and semantics of Python. It comes bundled with core Python distribution. It is written in C language and handles functionalities like Input/Output and other core modules. All these functionalities together make Python the language it is. More than 200 modules are there in the standard library.

- **NumPy:** NumPy stands for Numerical Python. It is the core library that is used for scientific computing in Python. It comprises multidimensional array objects and tools that help in working with these arrays. Numpy Array is a collection of the same type

of indexed values by a tuple of non-negative integers. The number of dimensions is the rank of the array; the shape of an array is a tuple of integers giving the size of the array along each dimension.

- **Pandas:** Panda offers useful, and flexible data structures that make analysis and manipulation of data easy. Pandas help to make the importing and analysis of data much simpler. Pandas are builds on modules like NumPy and matplotlib to give us a single & very convenient place for data analysis and visualization works.
- **matplotlib, seaborn:** These are the most popular visualization library used in python. They include barplot, countplot, histogram, piecharts, and many more. It is a plotting library for the Python programming language and its numerical mathematics extension NumPy.
- **sklearn:** It is the most useful and robust library for ML. It is built upon NumPy, SciPy, and Matplotlib It provides a wide variety of efficient tools for ML and statistical modeling which includes classification, regression, clustering, preprocessing, model selection, and dimensionality reduction.

2.1.2.3 Random Forest Regressor

Random Forest Regression is an ensemble learning method for regression. The ensemble learning method is a technique that combines predictions from multiple decision tree algorithms to make a more accurate prediction than a single model.

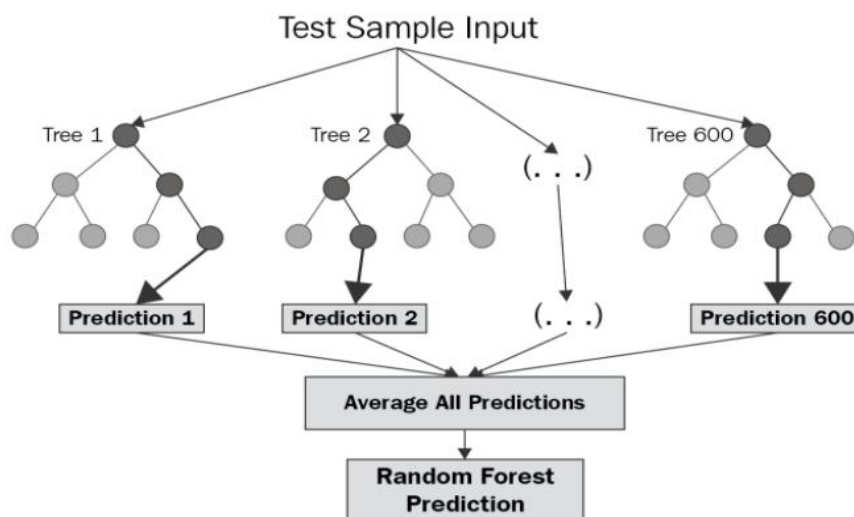


Fig: 1.1 - Working of a random forest algorithm

2.1.2.4 Linear Regression

This type of supervised learning is used in cases where the data used is continuous and is used to predict real-time values like total sales etc. In this algorithm, a relation between independent and dependent variables is set by fitting the best line called regression.

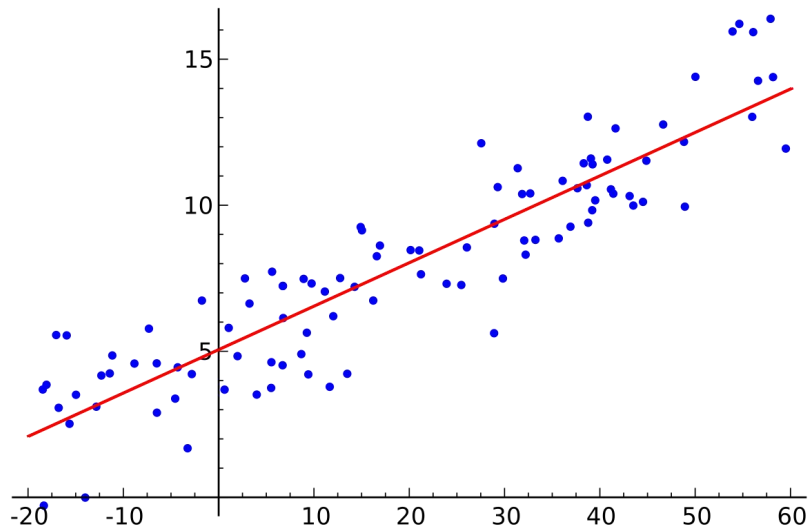


Fig: 1.2 - Working of Linear Regression

2.1.2.5 Decision Tree

This type of supervised learning is used for both categorical and continuous data but it is mainly used for classification problems. Here the data is split into various groups or sets based on different distinct independent variables. Based on the score the best is set of the independent variable is chosen.

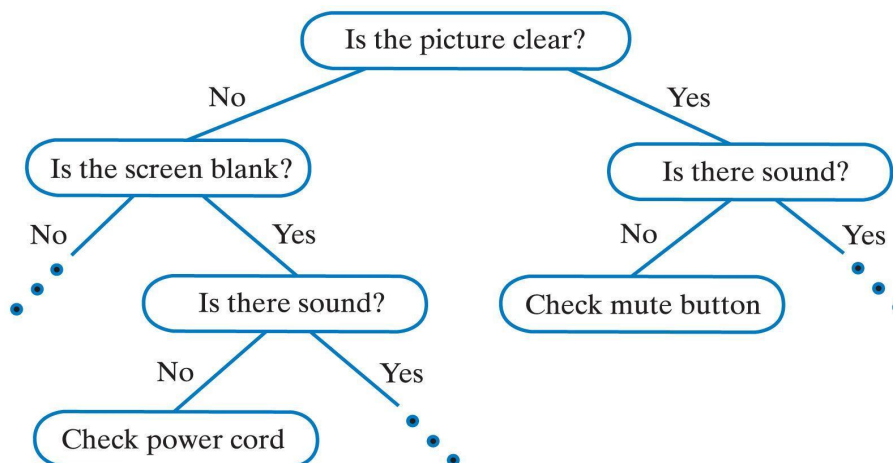


Fig: 1.3 - Working of Decision Tree

2.1.2.5 SVM

In this algorithm, plot n-dimensional space (where n is the number of features you have) with the value of each feature being the value of a particular coordinate. Then some lines are drawn in between the 2 different classified groups. The line from which the 2 closest points from both the groups are the farthest is selected.

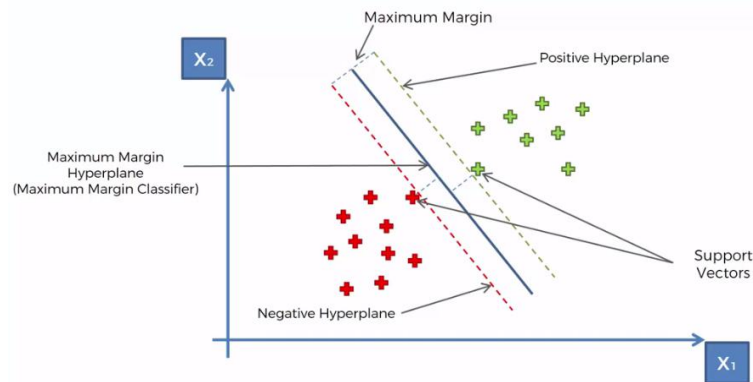


Fig: 1.4 - Working of SVM

2.1.2.6. XGBoost Regressor

Ensemble learning involves training and combining individual models (known as base learners) to get a single prediction. XGBoost is an ensemble learning approach for building supervised regression models. The validity of this statement is often inferred by knowing about its (XGBoost) objective function and base learners. An objective function consists of a loss function and a regularization term. It states the difference between actual values and predicted values, i.e the error in predicted values. The most common loss function in XGBoost for regression problems is reg: linear, which for binary classification is reg: logistics. XGBoost expects to possess the bottom learners which are uniformly bad at the rest so that when all the predictions are combined, bad predictions are canceled out and a will invoice better one sums up to form final good predictions.

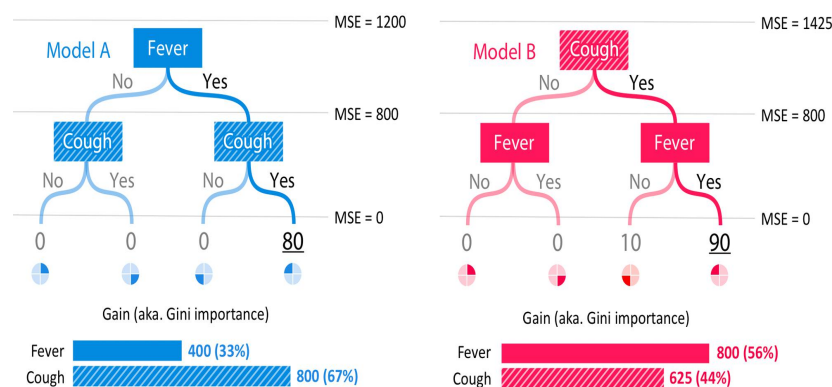


Fig: 1.5 - Working of XGBoost Regressor

2.1.2.4 JDBC

JDBC stands for Java Database Connectivity. It is an API for Java, which helps to connect and access a database. It is used to retrieve, edit, add data to the database when called from the UI by executing a query.

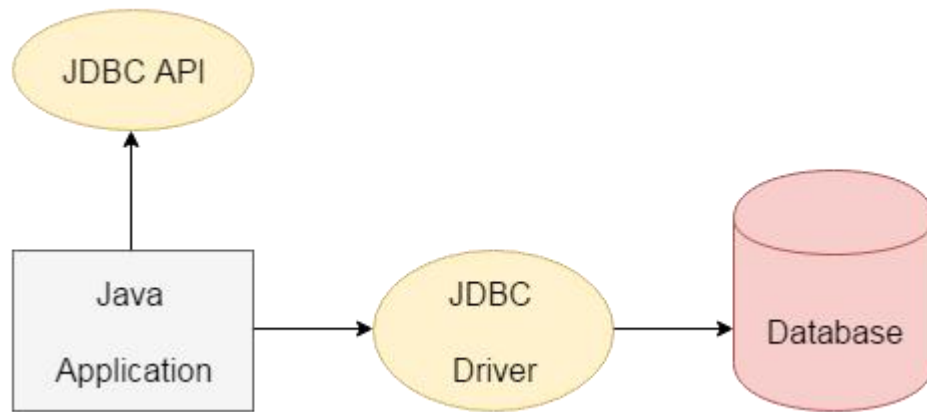


Fig: 1.6 - Connection of JDBC with Database with the help of Java

2.1.3 The idea behind the project

In the Machine Learning part of the project, we received an invoices dataset that contained the past payment information and behaviour of various buyers. Based on the previous payment patterns, the ML model had to predict what will be the date payment is made by the customer for an invoice. The model also predicts which ageing bucket the invoice falls into based on the predicted payment date. In the Web Development part, we parsed, processed, and loaded the given invoices dataset in the provided database schemas. A responsive UI is made which can display the invoice data loaded from the database. The UI supports searching and infinite scrolling operations. The UI supports editing of some editable fields, adding a new row to the grid, deleting rows from the grid, and downloading selected records from the grid in the predefined template(s).

3.1 Project Implementation

The implementation of the project can be broadly divided into two parts:

3.1.1 ML - Data analysis and Modelling: Data analysis and modelling mainly deal with, studying and understanding the underlying patterns of the data, structuring the data, and building some kind of model to estimate the future outcomes based on novel data.

Applying machine learning models to make predictions in business contains two parts: formulating a quantitative model for the business problem and tailoring the machine learning algorithms on the formulated model. Although little work has been done on predicting the outcomes of business invoices, there exists a large amount of analysis that has been performed on the accounts receivable to get a clear picture of the data. In the remaining parts of this section, I will review all the analyses performed and the classic machine learning specifically supervised regression algorithms used here.

3.1.1.1 Problem Formulation

In this project, we ask two questions about a new business invoice when giving instances of historical invoices and outcomes: " Would the invoice payment delay or not? " If it would delay, how long the delay would be? To answer these two questions, we will invoice payment delay, build a regression model that predicts the number of days by which it will get delayed, based on a training set of data containing instances whose outcome is known. We formulate the invoice outcome prediction task as a supervised learning problem: given instances of past invoices and their outcomes, build a model that can predict when a newly issued invoice will be paid, if no advanced actions are taken. And this model shall help us understand the characteristics of delayed invoices and problematic customers. In other words, it doesn't not only identify the payment delay but also evaluates the customers.

3.1.1.2 Data and Pre-Processing

The dataset provided to us consisted of the following attributes

Attribute Name	Attribute Description
cust_number	Customer id
name_customer	Customer Name
clear_date	Date when the invoice is cleared
buisness_year	The business year of the transaction
doc_id	Unique id for invoice
posting_date	Date when the document is posted
document_create_date	First Date of document creation
document_create_date.1	Second Date of document creation
due_in_date	Date when the invoice is supposed to be cleared
invoice_currency	Currency of invoice
document type	type of document
posting_id	--
area_business	--
total_open_amount	The total amount of transactions
baseline_create_date	Final Date of doc creation
cust_payment_terms	Terms on which the customer pays
invoice_id	Unique id for each invoice generated
isOpen	Whether the invoice is still open or not

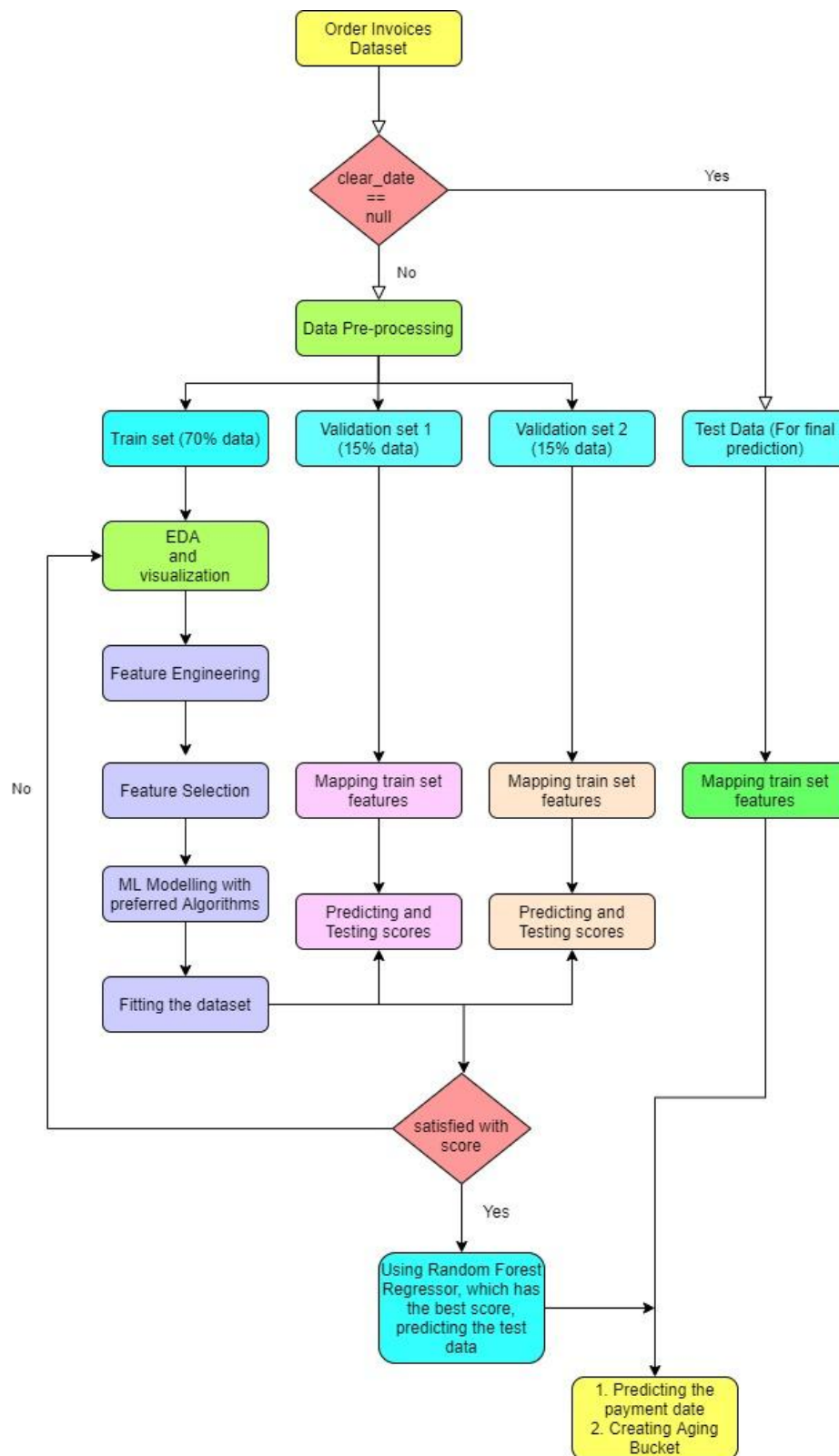


Fig: 1.7 - Working of ML Model

The dataset consisted of 50,000 rows, from which 4094 rows did not have the attribute 'clear_date'. These observations (samples) are considered in the testing set. The other portion of the dataset is divided into training and two validation sets. All the date-related attributes are converted to DateTime format. The supervising attribute or the target variable is named 'label', which interprets the number of days of invoice payment delay. It is calculated by subtracting the due_in_date from the clear_date in terms of days. The training set consisted of 32134 observation data and the two validation sets consisted of 6886 observations each.

3.1.1.3 Analysis of the Features

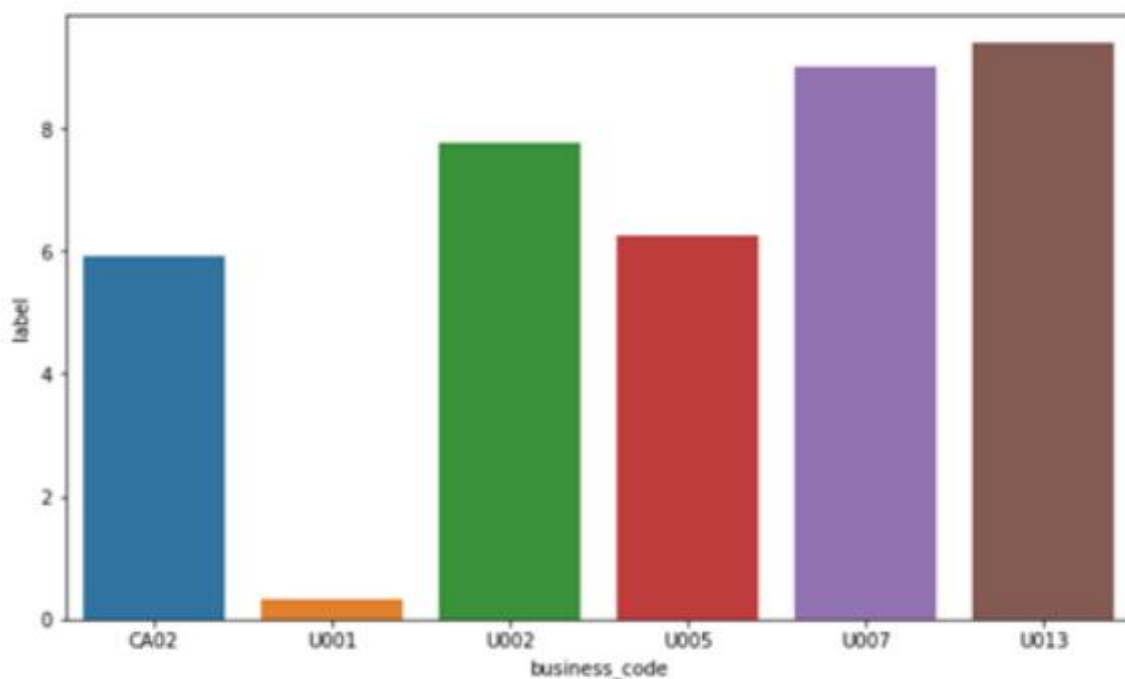


Fig: 1.8 - Average delay for customers from different business codes

The above barplot identified the average delay for customers from different business codes. Here we can see the customers from business code U001, make the minimum delay, and the customers from U013 make the maximum delay.

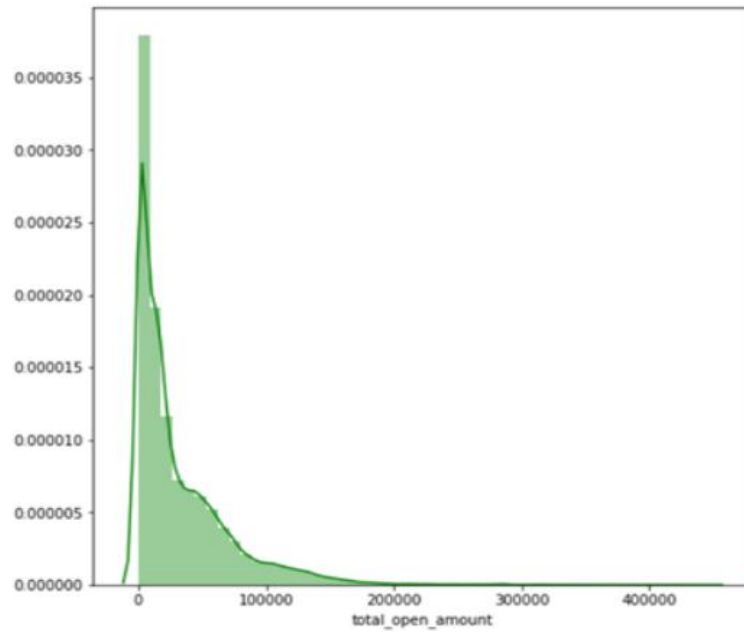


Fig: 1.9 - Total_open_amount distribution curve

- a. From the first plot, we can observe that most of the open amount is in between 0-100,000 i.e. very less number of customers are there whose high amounts are pending

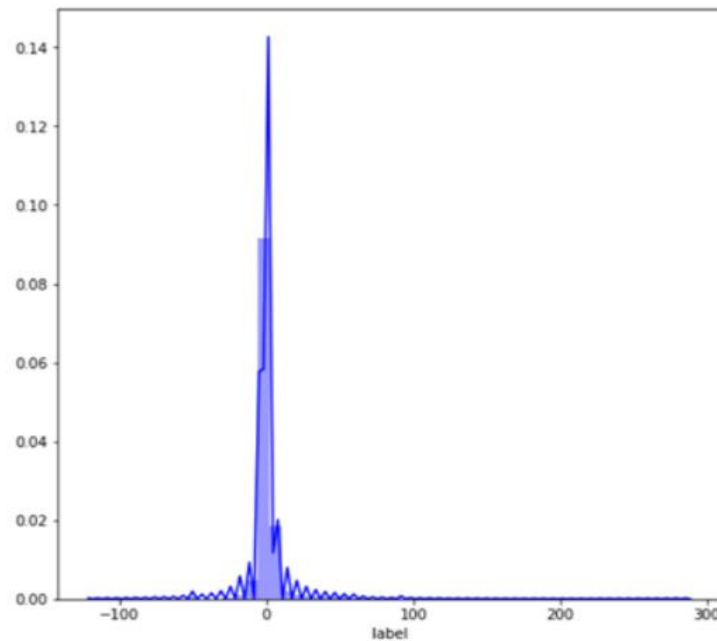


Fig: 1.10 - Delay Distribution Curve

- b. From the second curve, we can observe, most of the customers delay for a period mostly between 0-50.

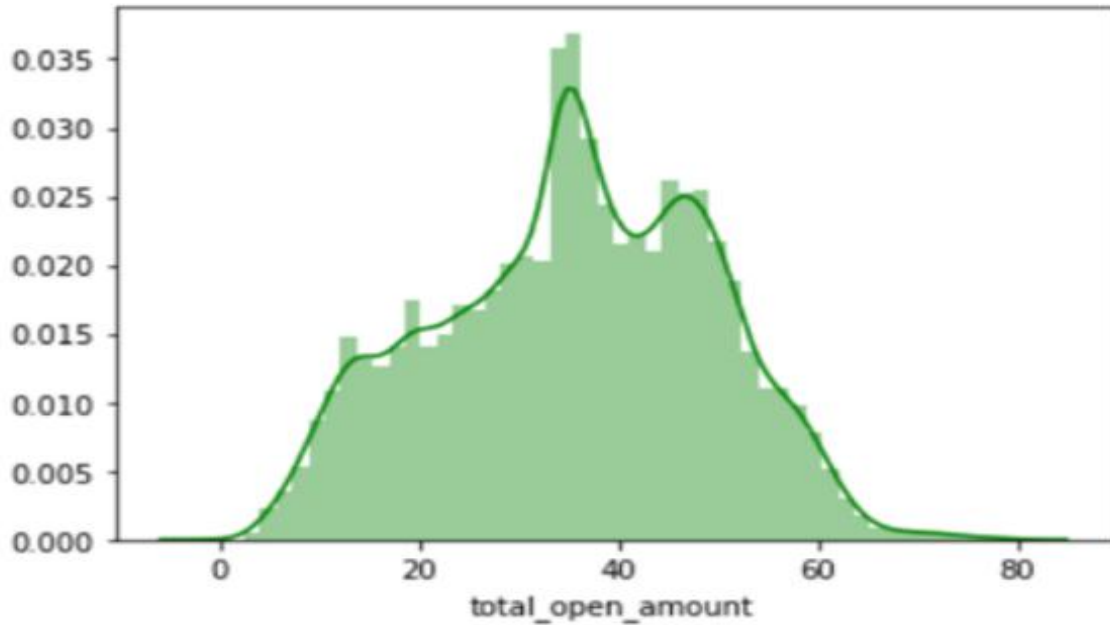


Fig: 1.11 - Log Transformation of total open amount curve

From the above plot, we can observe the distribution of the total open amount curve on log transformation.

3.1.1.4 Feature Engineering

The feature engineering has been done separately for train and validation set data. All the encodings have been performed over the train set and have been mapped to the validation & test set data. Some of the feature engineering steps have been pointed out below:

- Converting the currency to the same unit (USD), as there are few transactions in CAD and few in USD, so there can be unnecessary weighting of observations due to the multiplying factor between USD and CAD. The conversion is done with the help of the formula: $USD = CSD/0.78$
- Label encoding business_code, doc_type
- Mapping number of average delayed days per customer
- Target encoding cust_number, customer terms, this gives us more information about the nature of a specific customer id in delaying the invoice payment.
- Extracting the days, months, and quarters from the dates.

3.1.1.5 Feature Selection

In this section, we drop some of the unnecessary columns.

- clear date – It is already encapsulated in the label column i.e. the target variable.
- name_customer, cust_number, doc_id, posting_id – a too high variance to be considered.
- area_business – All null values. No information
- isOpen, invoice_id – constant attribute.No information.
- All other columns consisting of DateTime values

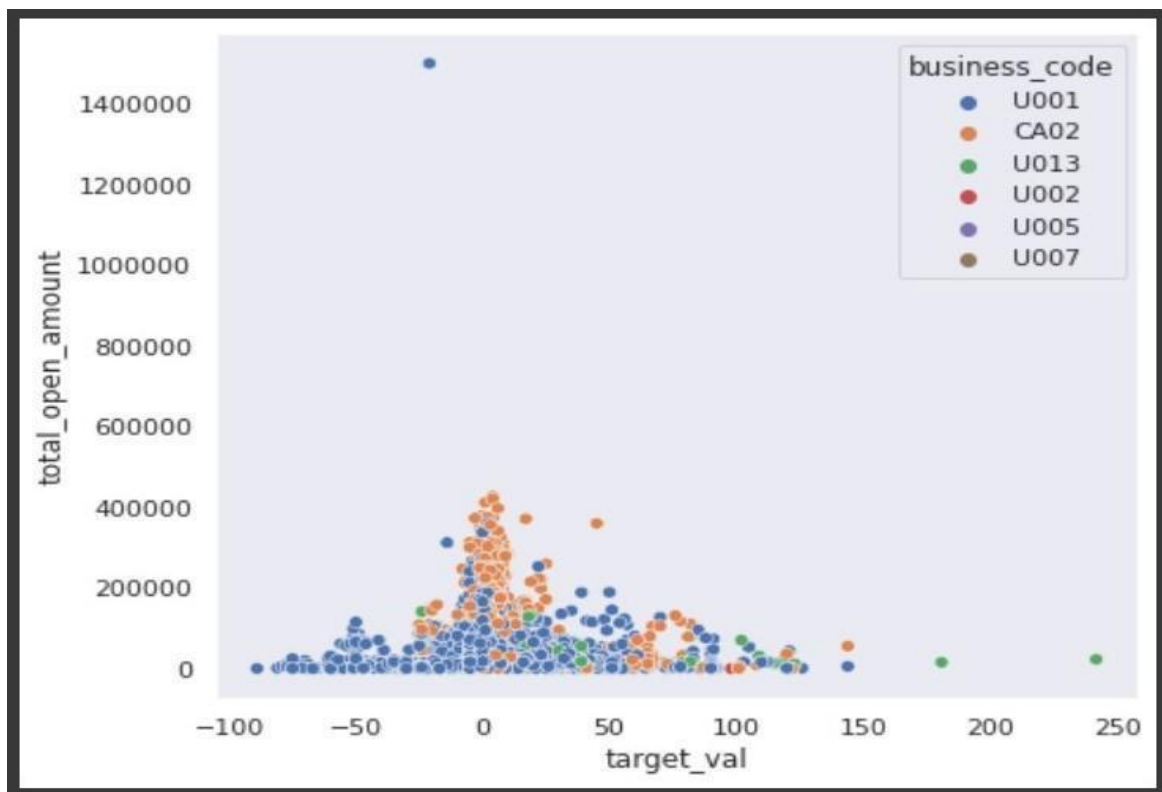


Fig: 1.12 - scatterplot between total_open_amount and target_value

The above graph represents a scatterplot between total_open_amount and target_value (the target column is the delay) with respect to the business_code.

- **Heat map:** A heat map (or heatmap) is a data visualization technique that shows the magnitude of a phenomenon as color in two dimensions. The color variation may be disabled by hue or intensity, giving obvious visual cues to the reader about how the phenomenon is clustered or varies over space.

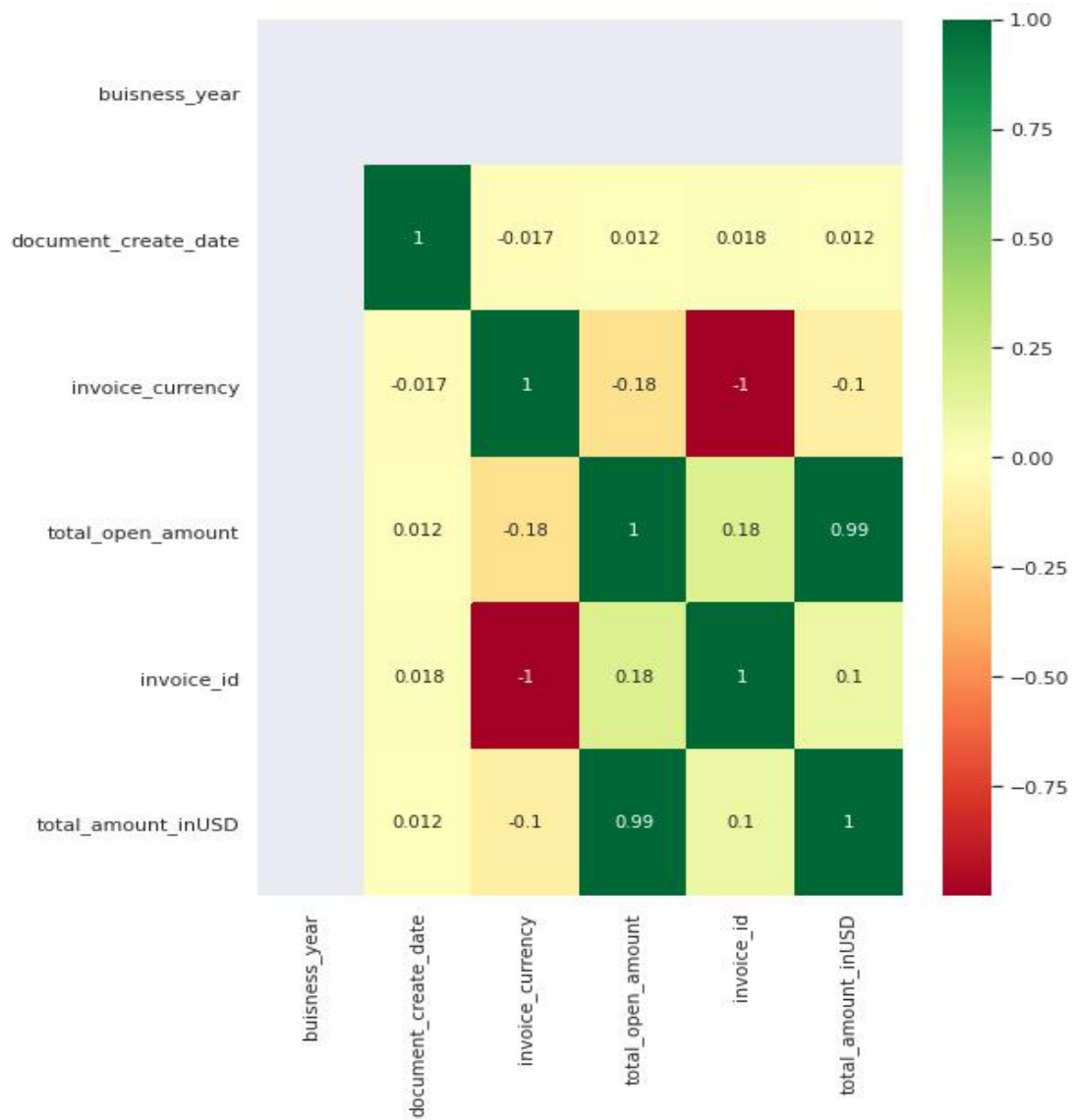


Fig: 1.13 - Heatmap of train dataset

A heatmap based on the training dataset and we can infer from the above plot that the darker the color, the more correlated it is.

3.1.1.6 Modelling (Supervised Regression Learning)

Before modeling the validation set has been divided into two sets –

- val1 – for hyperparameter tuning
- val2 – for testing the accuracy of the model on unseen data

For modeling, we have considered 5 models namely –

- XGBoost
- Gradient Boosting
- Decision Tree
- Random Forest
- Light Gradient Boosting

3.1.2 Java + ReactJS - application development: The application development part is related to building a web-based application that consists of a frontend dashboard displaying the data in a tabular form with different options like add, delete, edit and predict, connected to a backend database. We have created a beautiful user interface to add, edit, delete, print, and most importantly, predict order invoices with ReactJS. For this we have followed certain steps :

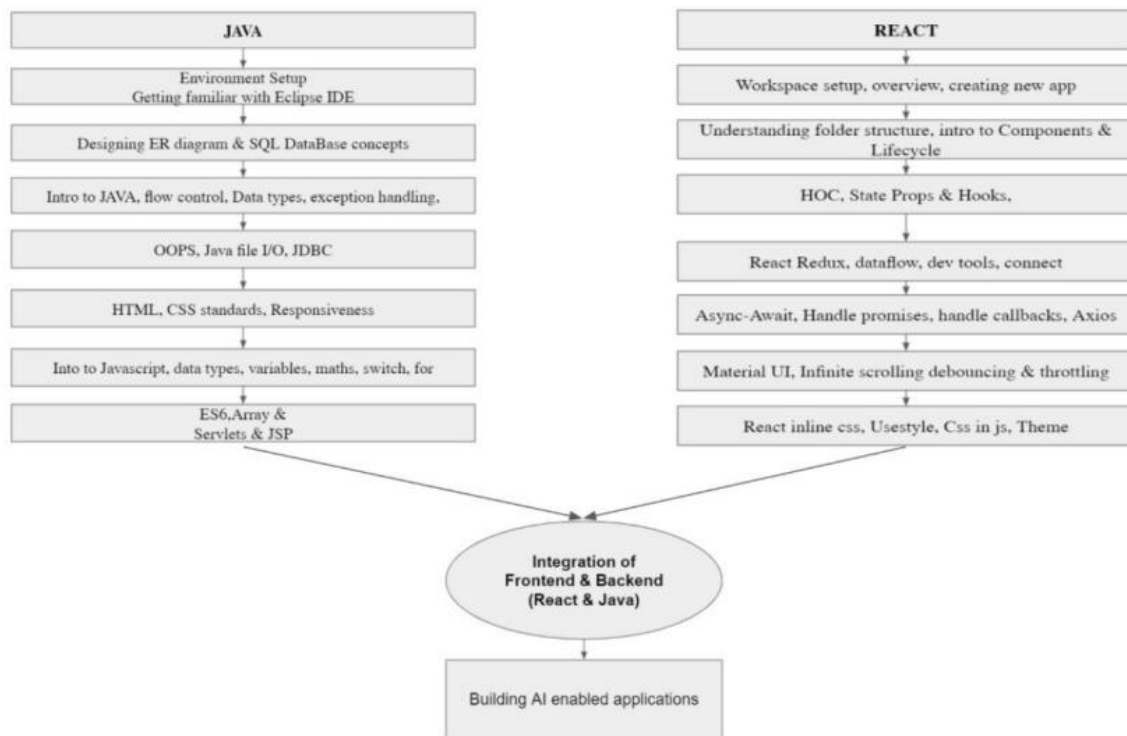


Fig: 1.14 - Workflow of Java and ReactJS

3.1.2.1 Loading of Data in the Database

The data was loaded in the database using SQLYOG and all tables and queries were created and successfully displayed. This was done by creating a raw CSV reader module from scratch. This module can read CSV files and can process the data to input into the database with the help of the JDBC Driver class.

3.1.2.2 Creation of the Backend

The backend portion was done in Java. The first JDBC connection was established with SQL. Then servlets were created for every functionality.

- **Add servlet** - POST request from the frontend with parameters such as invoice amount, notes, date, etc and pass them to the SQL database to add a new order to the database
- **Edit Servlet** - POST request from the frontend with parameters such as doc_id to identify the invoice in addition to the parameters which need to be changed.
- **Delete Servlet** - Delete the selected invoices from the database, by passing their respective doc_id's to identify them in the database.
- **Search Servlet** - Get the invoice number from the frontend and pass them as an HTTP request and search through the database and return it to the frontend again.
- **Data Display Servlet** - Display the table of invoices to the frontend UI.

3.1.2.3 Creating a Responsive Dashboard

A responsive dashboard was made in React Js. The main page had a Head Section, composed of the company logo and account name logo, and then the Grid Section, consisting of a panel having Add, Delete, Edit, and search button. The Predicted Payment Date will remain blank and selecting one or multiple rows and then clicking on predict will get the payment date populated. Whenever multiple rows are selected the add button will remain activated and the user can type in the value he/she wants to insert. Also, the user has to fill all the required fields otherwise insertion won't happen. Clicking on the edit button permits the user to edit all the editable columns, but only when one row is selected.

With the help of the delete button, the user can delete one or more pre-existing data.

The search button helps to look for particular data. The View Correspondence button will help to print the invoices of one or more orders. All these functionalities were carried out by using material-UI and establishing a connection with the backend.

3.1.2.4 Flask Integration

Finally, integration of the ML model was done with help of the flask module. First, one .pickle file was generated then using some lines of python script flask integration was done. As same as the servlet used as an API for the functionalities, for the prediction also some POST request was made using this .pickle file.

And finally, the project is complete.

```

export function prediction(submittingData) {
  submittingData = {"id": "1805537", "data": submittingData};
  console.log("In prediction function: ", submittingData);
  return axios.post(
    FLASK_URL,
    {},
    {
      headers: { 'Content-Type': 'application/json' },
      params: { data: submittingData }
    }
  ).then((res) => {
    console.log(res.data);
    return res.data;
  });
}

```

```

{
  "id": "<ROLL_NO>",
  "data": [
    {
      .....
    },
    {
      .....
    }
  ]
}

```

Like other servlets, in this way we accessed the flask server (left image) for the model prediction. The json data sent to the server was quite like this (right image).

Note : Some time, we may face one problem like this:

```

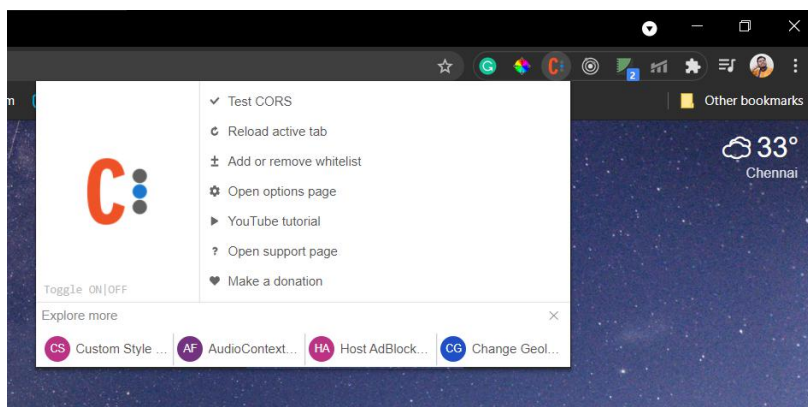
✖ Access to XMLHttpRequest at 'http://localhost:8080/1805537/select_all_1805537:1
1?page=2' from origin 'http://localhost:3000' has been blocked by CORS policy:
No 'Access-Control-Allow-Origin' header is present on the requested resource.

Error: Network Error
    at createError (createError.js:16)
    at XMLHttpRequest.handleError (xhr.js:84)

✖ ▶ GET http://localhost:8080/1805537/select_all?page=2
net::ERR_FAILED

```

This happens when we want to access cross origin requests. As the frontend UI is running on localhost:3000 port, but the backend servlets are running in 8080 port, browser does not allow this cross origin request. So we have to explicitly switch on this service via this chrome extension. Now we can run our project.



CHAPTER 4

RESULTS

4.1 Machine Learning Model Results (Backend)

The final accuracy for all the models on the training set, validation set 1, validation set 2 have been listed in the table below.

	train score	val1 score	val2 score	mse	model
0	0.613453	0.472818	0.645630	34.657654	XGBoost
1	0.582278	0.570999	0.738099	25.614128	Decision Tree
2	0.679106	0.546105	0.718473	27.533574	Ligh Gradient Boosting
3	0.637799	0.500607	0.659822	33.269671	Gradient Boosting
4	0.936800	0.751315	0.722642	27.125892	Random Forest

Fig: 1.15 - ML Accuracy Result

As the 'Random Forest' yielded the highest accuracy in the table, so we selected this algorithm for predicting the training dataset, which we have already segregated at first, the data with 'clear_data' == null.

Basically, we predicted the delay i.e. our target column, but the problem statement states something else, now our job is to add this delay to the due date column to predict the approximate predict date.

After creating predict date column in the test data, we were asked to categorize the invoice into different buckets based on the predicted payment date(delay) that is if the delay is within 15 days then it will be placed in bucket1 if the delay is more than 15 days and within 30 days then it will be placed in Bucket 2 and so on.

After categorizing the data we are creating a new column in our dataset (Aging bucket) and storing the bucket list in that column for further reference.

After the modelling was done, we stored the best model (with the Random Forest Regressor) in a .pkl file, so that we can use this file when we'll predict the result.

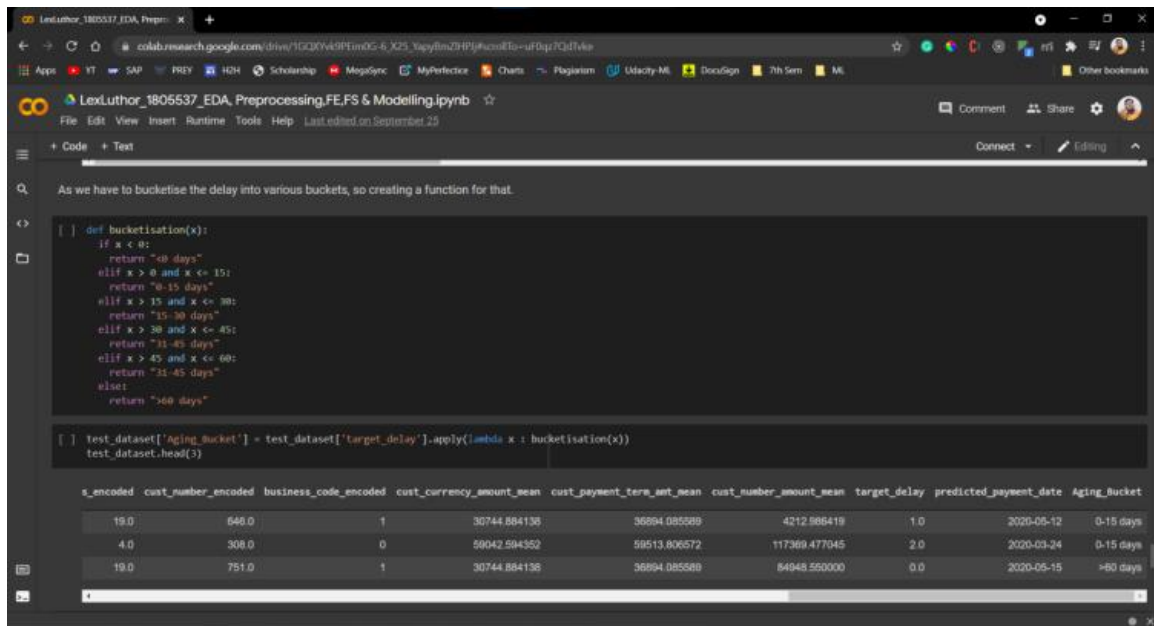


Fig: 1.16 - Bucketization

4.2 Java & SQL (Backend)

As the ML part was done, we started filling our databases with the present data (i.e. from .CSV). But we cannot put the raw data into the database, so we made a CSV reader from scratch to preprocess the raw data and make it, database readable.

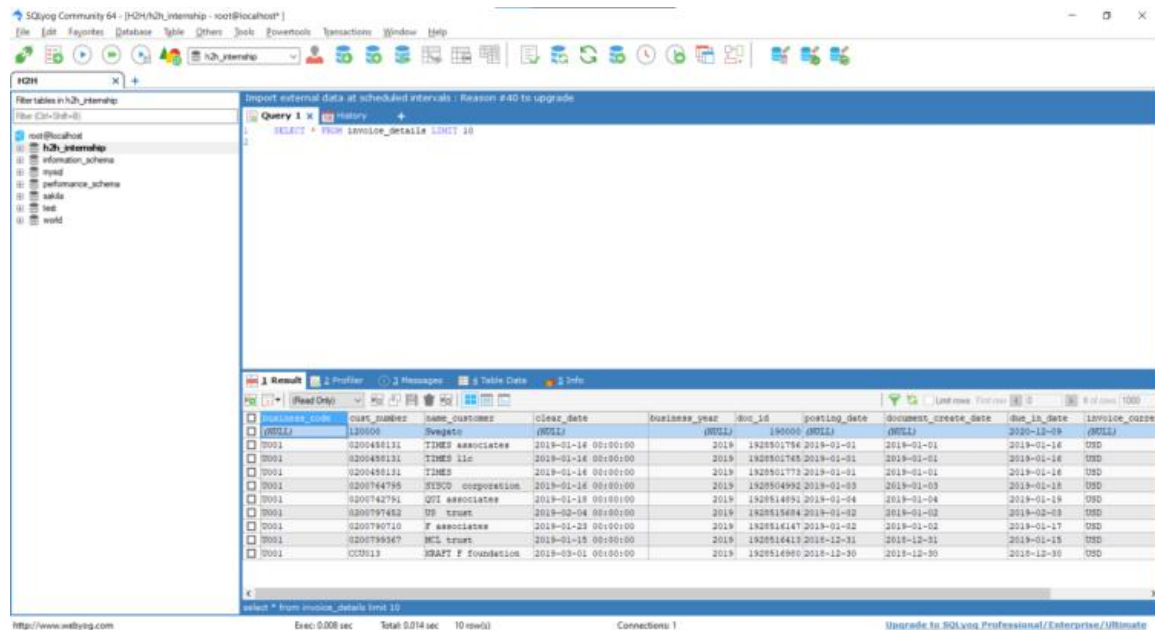


Fig: 1.17 - SQLyog

So we automated the process with code and the above was the result. Now, as this part was done, we started with creating the Add, Edit, Delete and Search servlets. We created the servlets using JDBC and servlet mapping was done, so that we can call each servlet uniquely.

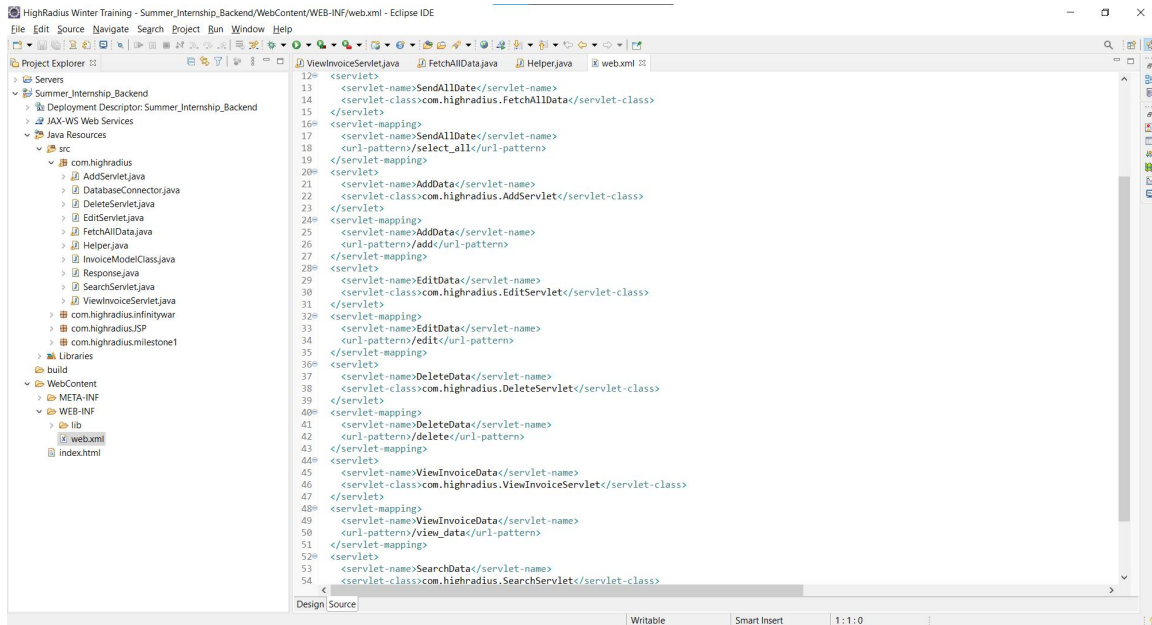


Fig: 1.18 - Servlet Mapping

List of Servlets:

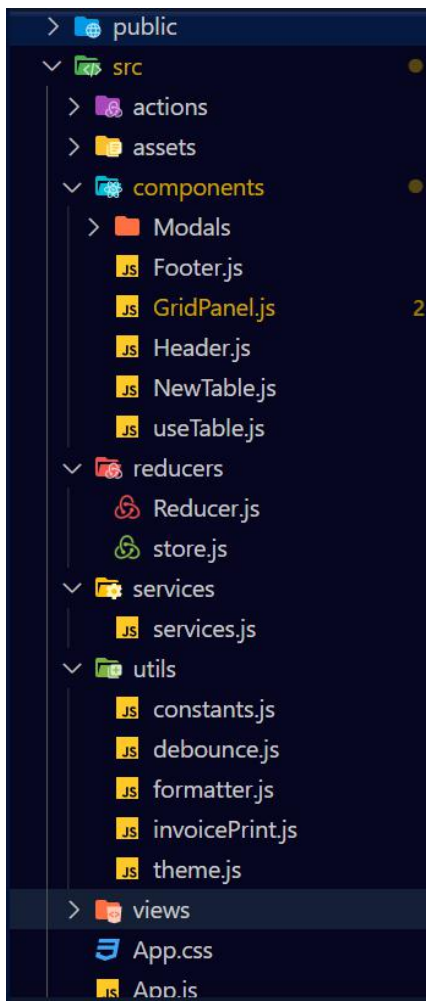
- **http://localhost:3000/1805537/select_all** - fetch all data
- **http://localhost:3000/1805537/add** - add data to database
- **http://localhost:3000/1805537/edit** - edit one order
- **http://localhost:3000/1805537/delete** - delete orders
- **http://localhost:3000/1805537/view_data** - for view correspondence
- **http://localhost:3000/1805537/search** - searching data

```
public class InvoiceModelClass {
    private String businessCode;
    @Expose private String customerNumber;
    @Expose private String customerName;
    private Timestamp clearDate;
    private int businessYear;
    @Expose private long docID;
    private Date postingDate;
    private Date documentCreateDate;
    @Expose private Date dueInDate;
    @Expose @SerializedName("orderCurrency") private String invoiceCurrency;
    private String documentType;
    private byte postingID;
    private String areaBusiness;
    @Expose @SerializedName("invoiceAmount") private double totalOpenAmount;
    @Expose private Date baselineCreateDate;
    private String customerPaymentTerms;
    @Expose @SerializedName("orderId") private long invoiceID;
    private byte isOpen;
    @Expose private String notes;
}
```

When sending the data to the frontend, we don't need all the data there, so @Expose keyword is added to send only those particular attributes through servlets.

4.3 Web Application Results (Frontend)

Here is the directory structure of the VS Code or the frontend part.



- action folder is used to place the action function for setting up the redux store.
- assets folder contains all the images and the CSS files.
- component folder contains the main UI structure of the application. For ex. - header portion, middle grid portion, tables and all the buttons which are needed like Add, Edit, View Correspondence, Predict and delete. The UI of Search Bar is also present.
- reducer folder contains the files needed for setting up the redux store.
- services folder contains all the API like “fetchAll”, “addDataAPI”, “editDataAPI”, “deleteDataAPI”, “viewDataAPI”, “prediction” and “search”
- utils contains all the utility functions, constant values, invoice printing code and theme set up.

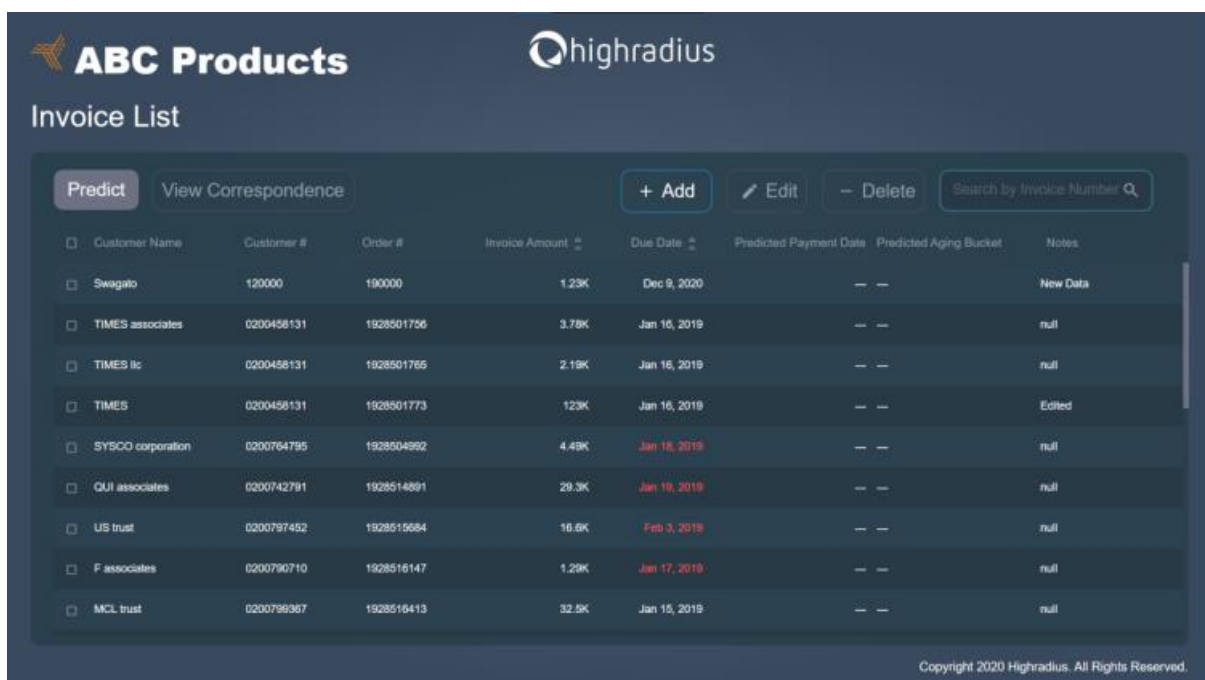


Fig: 1.19 - Main UI

The main UI page consists of the HighRadius logo in the upper middle, the company logo in the upper left corner, and the required buttons in the next subsequent portion and at last, the table UI which contains the dataset.

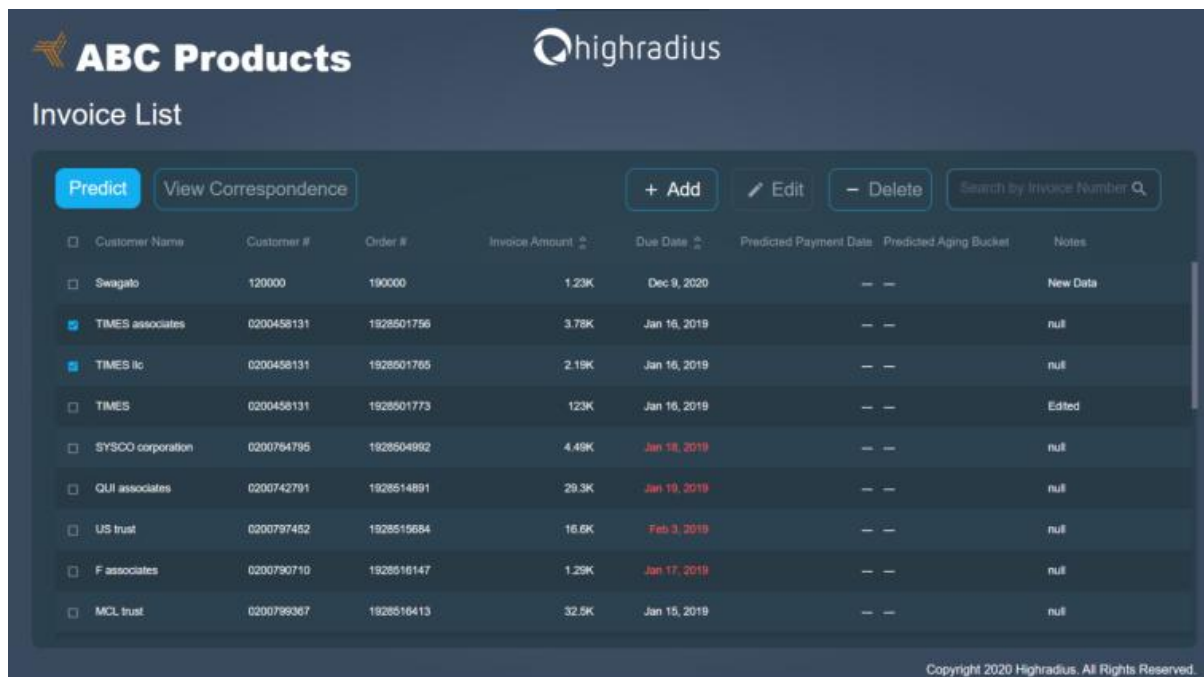


Fig: 1.20 - Active Buttons

If one row is ticked, then we can do Predict, View Correspondence, Add, Delete, Search operation, but cannot do edit operation, as we cannot edit two orders parallely.

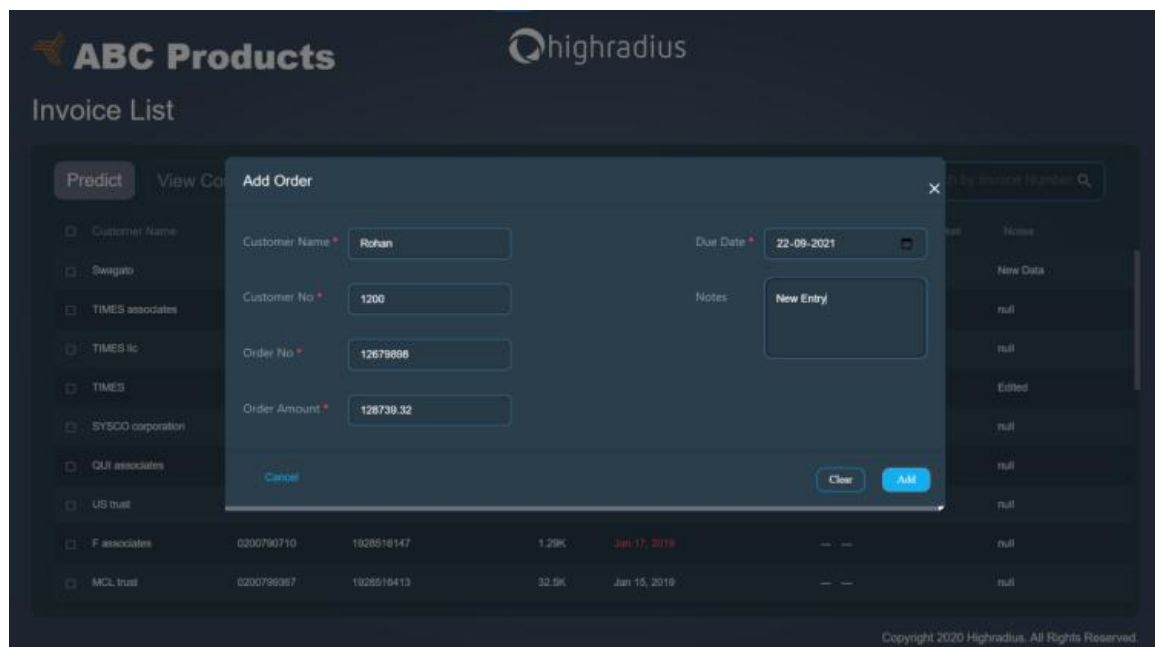


Fig: 1.21 - Add Order Button

Add Button Functionality: Clicking on the add button will allow the user to add data to the editable fields. The add button will remain in a disabled state by default. After the user adds every row, the add button gets enabled. The window should contain a Clear and an Add button. The user should be able to edit the values. Once the user clicks on the save button, the new values should be displayed in the UI and should remain persistent

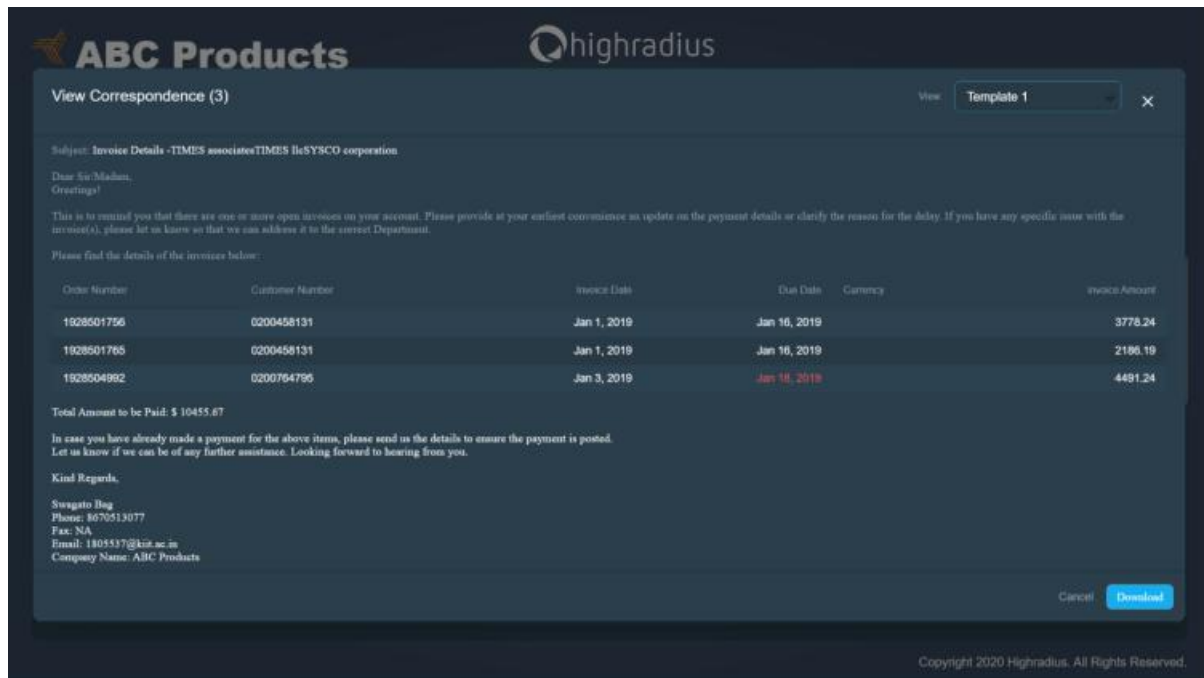


Fig: 1.22 - View Correspondence

View Correspondence Button Functionality: Clicking on the view correspondence button will allow the user to view and generate PDF's of the selected invoices from the grid. The view correspondence button will remain in disabled state by default. When the user selects one or more rows, the view correspondence button gets enabled. Clicking on the View Correspondence button displays a popup window on the screen. The window should contain a default template with the data of selected rows auto-populated in it along with a Cancel and a Download button. The account name in the template should be dynamically filled. The total amount to be paid should be the sum of the open amounts of the auto populated invoices in the template. The sender's details should be static values. On clicking the Download button, a file containing the data from the selected rows in the selected template should be downloaded in PDF format.

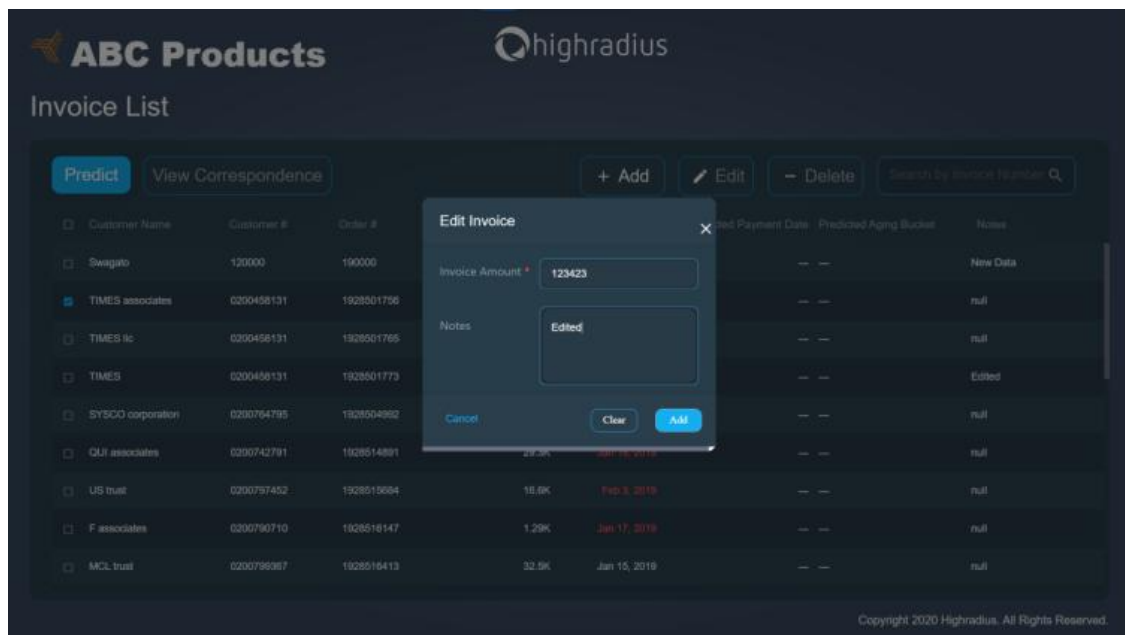


Fig: 1.23 - Edit Button

Edit Button Functionality: Clicking on the edit button will allow the user to modify the editable fields in the data. The editable columns are Invoice Amount and Notes. The edit button will remain in a disabled state by default. When the user selects one row, the Edit button gets enabled. If a user selects multiple rows, the edit button will remain disabled. Clicking on the Edit button displays a popup window on the screen. The window should contain the Invoice Amount and Notes headers along with a Cancel and a Save button. The user should be able to edit the values. Once the user clicks on the save button, the new values should be displayed in the UI and should remain persistent.

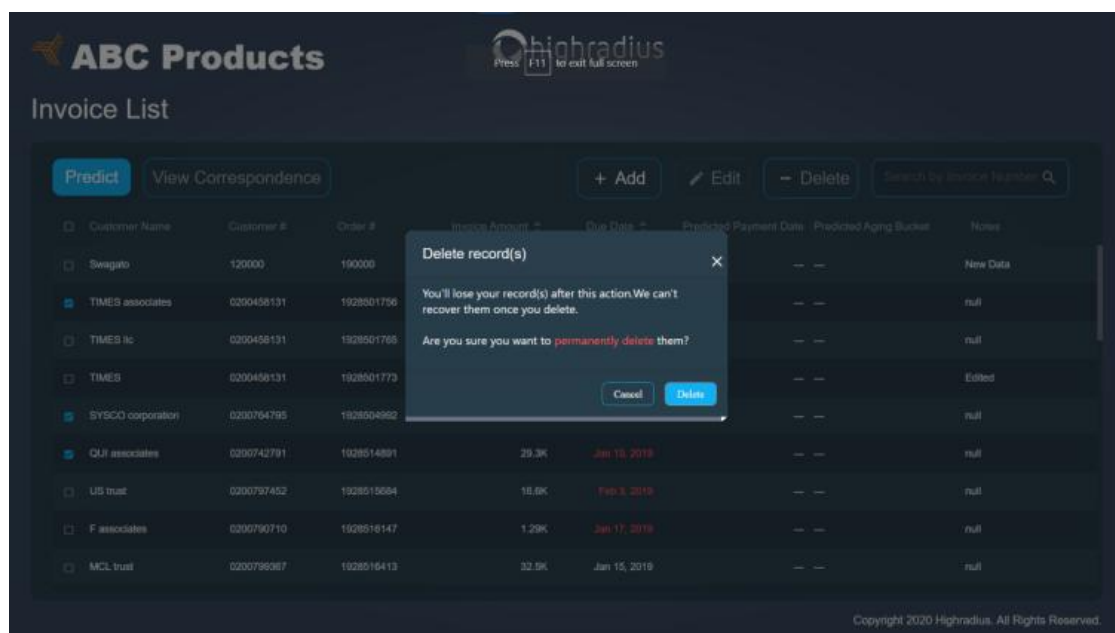


Fig: 1.24 - Delete Button

Delete Button Functionality: Clicking on the delete button will allow the user to delete one or multiple data. The delete button will remain in a disabled state by default. After the user adds ticks to a minimum one row, the delete button gets enabled. The window should contain a Cancel and a Delete button. Once the user clicks on the Delete button, the values should be deleted in the UI and should remain persistent

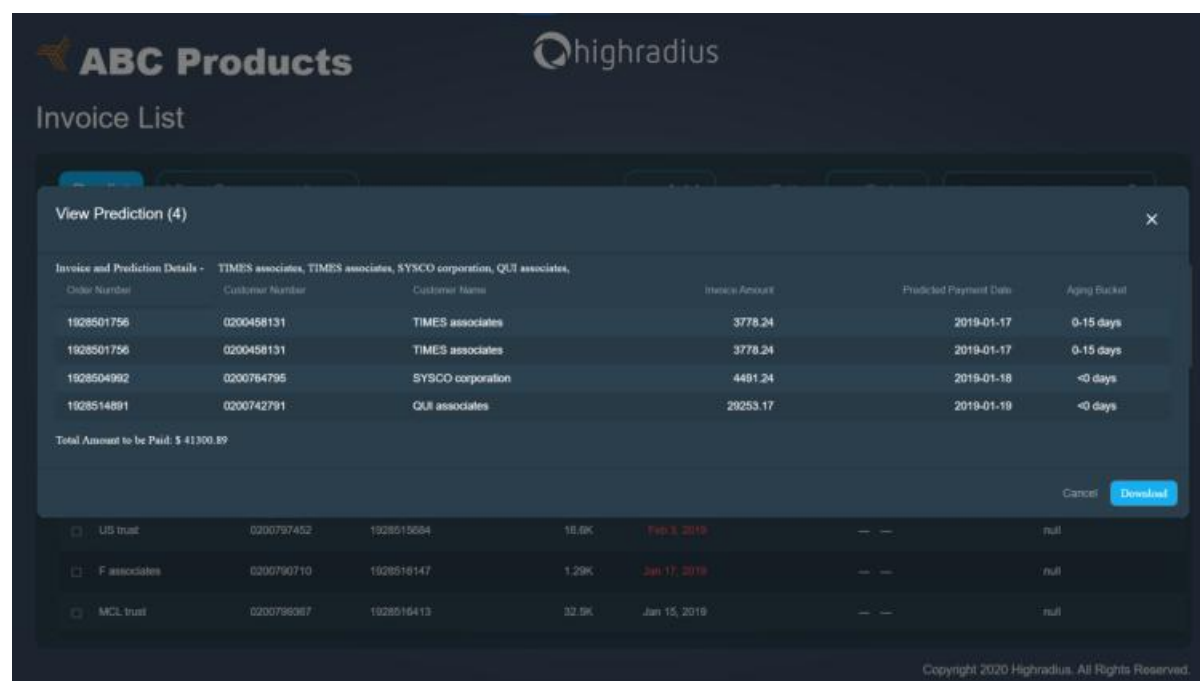


Fig: 1.25 - Predict Button

Predict Button Functionality: Clicking on the predict button will allow the user to predict the probable payment Date for one or multiple data. The predict button will remain in a disabled state by default. After the user adds ticks to a minimum one row, the predict button gets enabled. The window should contain a Cancel and a Download button. On clicking the Download button, a file containing the data from the selected rows in the selected template should be downloaded in PDF format. The PDF would look like this.

View prediction (3)

Order Number	Customer Number	Customer Name	Invoice Amount	Predicted Payment Date	Aging Bucket
1928501756	0200458131	TIMES associates	3778.24	2019-01-17	0-15 days
1928504992	0200764795	SYSCO corporation	4491.24	2019-01-18	<0 days
1928515684	0200797452	US trust	16584.08	2019-02-04	0-15 days

Total Amount to be Paid: \$ 24853.56

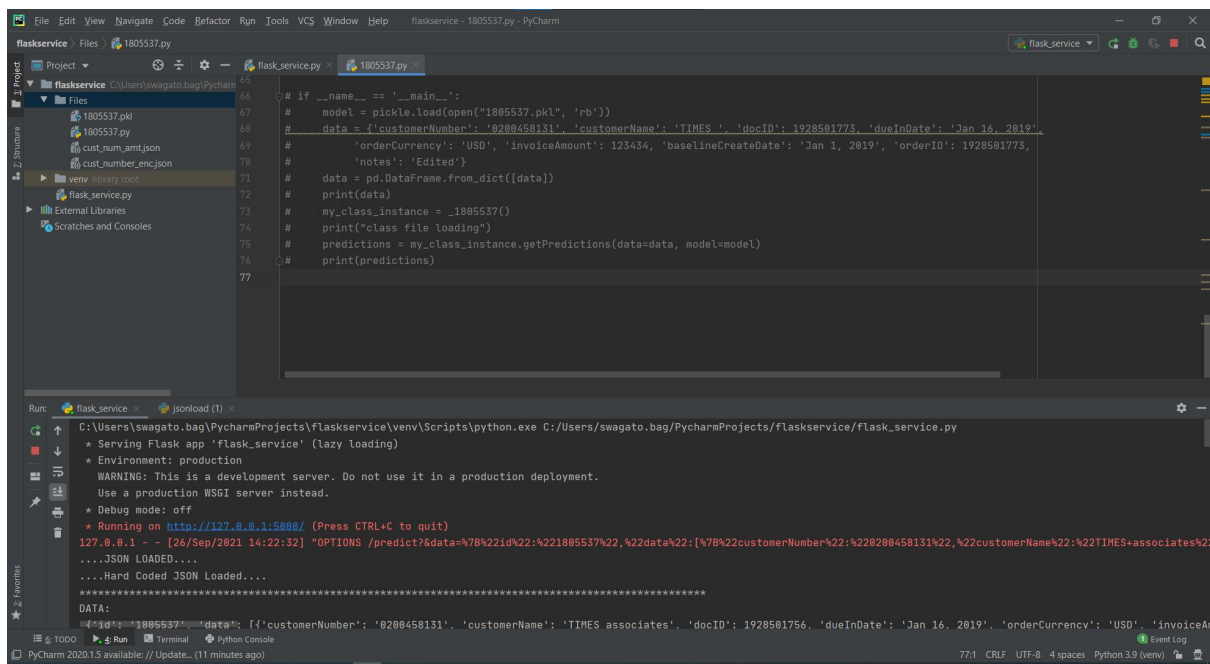


Fig: 1.26 - Working of Flask Server

Flask server was setup in pycharm IDE for the convenience purpose. First we have created a virtual environment for the same to install extra libraries that we need for the purpose, like flask library for the local server, joblib for loading the pickle file and others.

The local server address was set up to be <http://127.0.0.1:5000/predict> to be used for the flask server.

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1 Conclusion

During the tenure of my training at HighRadius, I had exposure to various technical languages which helped in shaping my analytical and problem-solving skills as well as provide knowledge that could be applicable in real-time projects as well. By making the Order Management System I was able to learn how a flow of code works and how to establish connections between different languages.

5.2 Future Work

With the help of AI-Enabled Fintech B2B Order Management Application the organizations, companies, and in fact, individuals can keep track of the payment of their orders irrespective of a large number of clients. This will not only reduce time but also provide a functional manner for the proper functioning of an organization and/or workplace.

5.3 Planning And Project Management

Activity	Starting week	Number of weeks
Python Training	2 nd Week January	1
Data processing and Exploration	3 rd Week January	1.5
Feature Engineering and Selection	4 th Week January	0.5
Modeling (End of Milestone 1)	1 st Week February	1
SQL & Java Training	2 nd Week February	1
HTML, CSS, JS Training	3 rd Week February	0.5
Intro to React Training, States & Props, Life Cycle Methods	3 rd Week February	2
Hooks, Redux, Axios	1 st Week March	0.5
Material UI, Debouncing, Throttling	2 nd Week March	0.5
Styling, Themes, JSS (End of Milestone 2)	3 rd Week March	0.5

Table 5.1 showing details about project planning and management

Task	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9
Python Training									
Data Processing & Exploration									
Feature Engineering and Selection									
Modelling (End of Milestone 1)									
SQL & Java Training									
HTML, CSS, JS Training									
Intro to React Training, States & Props, Life Cycle Methods									
Hooks, Redux, Axios									
Material UI, Debouncing, Throttling									
Styling, Themes, JSS (End of Milestone 2)									

Fig: 1.27 - Gaant Chart for project planning and management

REFERENCES

- [1] Various articles on <https://towardsdatascience.com>
- [2] <https://www.tutorialspoint.com/servlets/index.htm>
- [3] https://www.w3schools.com/react/showreact.asp?filename=demo2_react_test
- [4] <https://www.javatpoint.com/machine-learning>
- [5] <https://www.javascripttutorial.net/>
- [6] <https://openliberty.io/guides/rest-client-reactjs.html>
- [7] <https://docs.python.org>

INDIVIDUAL CONTRIBUTION REPORT:

AI-Enabled FinTech B2B Order Management Application

SWAGATO BAG

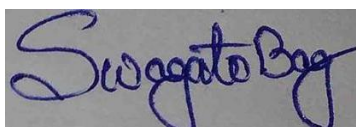
1805537

Contribution and findings:

The best finding of the project is understanding the actual scenario of B2B operations as well as how B2B is different from B2C and C2C. The work structure and the difference between the ideal world and real-world business come into the picture where we get to know that in the ideal world, the buyer business should pay back within the stipulated time (i.e. the Payment Term). However, in the real world, the buyer business seldom pays within their established time frame, and this is where the Account Receivables Department comes into the picture. Every big company finds it difficult to keep a track of their payments and here Account receivables team helps in doing their job by

- Collecting payments from customers for their past invoices.
- Sending reminders and follow-ups to the customers for payments to be made.
- Looking after the entire process of getting the cash inflow.
- Help the company get paid for the services and products supplied.

The other findings while working with the dataset was encoding the features and forming new features as it was a verse dataset with all kinds of data to parse. So it was a challenging task but after handling all the data we got confidence that we work with any type of dataset with ease. As it was a vast dataset parsing data from the database was another challenge for us and as a result, got to know many new approaches and new classes in java which turned out to be very useful and are going to help us shortly. Last but not least in UI development using Reactjs and MaterialUi helped us to get a real-life idea of the professional work although over-writing the themes in MaterialUi was a bit of a pain and making it responsive was another challenge. And the best thing that we got to learn from React was the redux as it made our work simpler and the idea of hooks and props also helped a lot. All in all, it was a complete package which helped us to learn a lot of new skills and the best part is it helped us to explore our potential.



Swagato Bag