

A PROJECT REPORT
on
AI-Enabled FinTech B2B
Invoice Management Application

Submitted to
KIIT Deemed to be University

In Partial Fulfilment of the Requirement for the Award of

BACHELOR'S DEGREE IN
COMPUTER SCIENCE &
COMMUNICATION
ENGINEERING
BY

ROHIT JAISWAL 1929235

UNDER THE GUIDANCE OF

TARINI MOHAPATRA
AKASH KUMAR YADAV

Affiliation



SCHOOL OF COMPUTER ENGINEERING
KALINGA INSTITUTE OF INDUSTRIAL TECHNOLOGY
BHUBANESWAR, ODISHA – 751024 May 2020

A PROJECT REPORT
on
AI-Enabled FinTech B2B
Invoice Management Application

Submitted to
KIIT Deemed to be University

In Partial Fulfilment of the Requirement for the Award of

BACHELOR'S DEGREE IN
COMPUTER SCIENCE &
COMMUNICATION
ENGINEERING

BY

ROHIT JAISWAL 1929235

UNDER THE GUIDANCE OF

TARINI MOHAPATRA
AKASH KUMAR YADAV

Affiliation



SCHOOL OF COMPUTER ENGINEERING
KALINGA INSTITUTE OF INDUSTRIAL TECHNOLOGY
BHUBANESWAR, ODISHA -751024 May 2022

KIIT Deemed to be University

School of Computer Engineering
Bhubaneswar, ODISHA 751024



CERTIFICATE

This is to certify that the project entitled

AI-Enabled FinTech B2B Invoice Management Application

submitted by

ROHIT JAISWAL

1929235

is a record of bonafide work carried out by them, in the partial fulfillment of the requirement for the award of Degree of Bachelor of Engineering (Computer Science & Communication Engineering) at KIIT Deemed to be University, Bhubaneswar. This work is done during the years 2022-2023, under our guidance.

Date: 15/ 04/ 2022

Tarini Mohapatra &
Akash Kumar Yadav

Project Mentor

Acknowledgements

We are profoundly grateful to **Tarini Mohapatra & Akash Kumar Yadav** of **Affiliation** for their expert guidance and continuous encouragement throughout to see that this project rights its target from its commencement to its completion.

ROHIT JAISWAL

ABSTRACT

The B2B world is not the same as the B2C or C2C worlds. Businesses work on a credit basis with one another. When a buyer company orders products from a seller company, the seller company sends an invoice to the buyer company. This products invoice comprises a variety of facts, such as the details of the items purchased and when they should be paid. In accounting, this is referred to as "Accounts Receivable." The seller's business interacts with a variety of firms and sells things to them all at different times. As a result, the seller company must keep track of the total amount it owes all of the purchasers. This entails keeping track of all of the purchasers' invoices. Each invoice will have information such as the payment due date, invoice date, invoice amount, baseline date, and so on. The buying company must pay the balance owed before the deadline. However, bills are not always cleared, i.e., paid in full by the due date, in real-world settings. The payment date refers to the day on which a client clears an invoice payment.

In the ideal environment, the buying firm should pay back within the stated period (e.g., the Payment Term) (i.e., the Payment Term). However, in the actual world, the buying firm seldom pays within their set time limit, and this is where the Account Receivables Department enters into the picture. Every firm consists of a separate Account receivables Department to collect and track the payment of bills. It comprises an Account receivables team that is responsible for collecting payments from clients for their previous bills. Sending reminders and follow-ups to the consumers for payments to be made. Looking after the complete procedure of acquiring the cash inflow. Help the firm be compensated for the services and goods given. We will be designing a web application to support the personnel working in the Accounts Receivable divisions in their day-to-day tasks

Keywords: Invoice, Management, Prediction, Web-App, B2B.

Contents

1	Introduction	1
2	Basic Concepts	2
	2.1 Python	2
	2.2 Machine Learning	3
	2.3 Java	8
	2.4 JDBC & Collections	10
	2.5 React	11
3	Problem Statement / Requirement Specifications	15
	3.1 Project Planning	16
	3.2 Project Analysis	17
	3.3 System Design	19
	3.3.1 Design Constraints	19
	3.3.2 System Architecture / Block Diagram	20
4	Implementation	21
	4.1 Methodology / Proposal	21
	4.2 Testing / Verification Plan	27
	4.3 Result Analysis / Screenshots	28
5	Standard Adopted	29
	5.1 Design Standards	29
	5.2 Coding Standards	29
	5.3 Testing Standards	29
6	Conclusion and Future Scope	30
	6.1 Conclusion	30
	6.2 Future Scope	30
	References	31
	Plagiarism Report	31

List of Figures

3.1	Algorithm Accuracies	17
3.2	Project UI Dashboard	18
3.3	Block Diagram of Project	20
4.1	CSV File	24
4.2	Data Inserted in SQL Yog	24
4.3	Receivables Dashboard Page	26
4.4	Regression Model Scores	28
4.5	Final Data Frame	28

Chapter 1

Introduction

An Invoice Management System is a computer software program that allows businesses to manage their orders and inventory. Invoice management systems help ensure more accurate payment management. It comprises an Account receivables team that is responsible for:

- 1) Collecting money from consumers for their prior bills.
- 2) Sending reminders and follow-ups to the consumers for payments to be made.
- 3) Looking after the complete procedure of acquiring the cash inflow.
- 4) Help the firm be compensated for the services and goods given.

Invoice management software is also shareable, from the customer service team to the accounting team, the warehouse staff, and you, 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.

In this project, we will be constructing a web application to support the employees working in the Accounts Receivable departments in their day-to-day tasks. You need to design a web application where the users in the Account Receivable department may,

- 1) View the invoice data from various buyers.
- 2) See various fields/attributes of the invoice(s) from a given buyer.
- 3) Perform Data Pre-processing on the invoice date.
- 4) Get account-level analytics to conveniently see and comprehend data- EDA and Feature Engineering.
- 5) Get a prediction of when the invoice is going to get paid

Chapter 2

Basic Concepts

2.1 Python

Python is object-oriented, high-level programming, interpreted language with dynamic semantics. Python includes high-level built-in data structures, combined with dynamic typing and dynamic binding, which helps python make it incredibly popular for Rapid Application Development, furthermore it is used as a scripting language to link the existing components. Python has an easy and basic syntactic basis which in turn increases and emphasizes the readability of the code and in turn minimizes the cost of program maintenance. Python also offers modules and packages, which foster program modularity and code reusability. The Python interpreter and the standard libraries are available in source or binary form for all main major platforms and may be freely downloaded.

What are the Python Libraries?

A module is a file with some Python code written in it, while a package is a directory containing sub-packages and modules. A Python library is a reusable set of code that we may incorporate into our programs/ projects.

If we Compare python to languages like C++ or C, Python libraries do not connect to any particular environment in Python. Here, a ‘library’ denotes a collection of fundamental components. A library is a collection of modules. A package is a library that can be installed using a package installer like ruby gems or npm.

The Python Standard Library is a collection of precise syntax, tokens, and semantics of Python. It comes packaged with core Python distribution. It is developed in C language and handles capabilities like Input/Output and other basic components. All these capabilities combined make Python the language it is.

More than 200 modules are their standard library. This library ships with Python.

NumPy Arrays -

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 which help in working with these arrays.

NumPy Array is a collection of values in order with the same type and is indexed by a tuple of non-negative integers. The number of dimensions is the rank of the array; the shape of an array depends upon a tuple of integers giving the size of the array along each dimension.

Pandas-

Panda is the most famous Python module for machine learning and data science, which offers useful, powerful, and flexible data structures that make the analysis and manipulation of data easy. Pandas make the importing and analyzing of data much simpler. Pandas is built on modules like NumPy and matplotlib to give us a single & very convenient place for data analysis and visualization works.

Basic Features of Pandas-

Data frame objects help us in keeping track of our data. With pandas' data frame, we can have different data types like (float, int, string, Date-Time, etc) all in one place. Panda's package has built-in functionalities like easy grouping & easy joins of data and rolling windows. It has Good Input/Output capabilities; it can easily pull data from a MySQL database directly into a data frame. With pandas, we can use patsy for R-style syntax in doing regressions.

Pandas Series -

Panda Series is like a 1-D array that is capable of holding data of any data type such as (integer, string, float, python objects, etc.). Panda Series can be created using a constructor.

Syntax: - pandas. Series (data, index, dtype, copy).

2.2 Machine Learning

Machine learning (ML) is described as the study of computer algorithms that improve their performance and outcomes automatically via long-term experience and through the usage of data sets. ML is a subset of artificial intelligence. Machine learning algorithms develop a model which is dependent on sample data for testing, known as "training data", in to generate predictions, judgments or analyses without being explicitly coded for the particular. Machine learning algorithms are utilised in various applications, such as in the medical industry, screening of emails, and computer vision, where it is very difficult to anticipate the result or design traditional algorithms to fulfil the essential tasks. A subset of machine learning is roughly connected to computational statistics, and that focuses on making judgments and predictions using computers; although not all machine learning incorporates statistical learning. The study of mathematical optimization offers the theoretical portion and application areas in the field of machine learning. Data mining is also a similar topic of research, and this focuses on exploratory data analysis through unsupervised learning. In its business application problems, machine learning is often dubbed predictive analytics.

ML MODEL-

While building this model we had gone through many processes:

Step 1. Firstly, we need to import the dataset and also import various data manipulation libraries.

Step 2. We visualized the dataset in the form of a data frame to get a brief idea about the dataset.

Step 3. Identified the target variable which is clear_date.

Step 4. Independent and dependent variables identification and extraction.

Step 5. Handling Missing Values using Null Imputation Techniques.

Step 6. Encoding Categorical Variables for training purposes.

Step 7. Splitting the dataset for train/test and validation.

Step 8. Feature Scaling for improved training using Normalization and Standardisation.

Step 9. Training and Validating ML model.

Step 10. Predict clear_date using test data.

Step 11. Prepare aging bucket by subtracting invoice creation date from predicted clear date. The different buckets were:

- 1) 0-15 days
- 2) 16-30 days
- 3) 31-45 days
- 4) 46-60 days
- 5) Greater than 60 days

Regression Analysis-

In statistical modeling, regression analysis is a set of statistical processes for estimating the relationships between a dependent variable and one or more independent variables. The most common form of regression analysis is linear regression, in which one finds the line that most closely fits the data according to a specific mathematical criterion. For example, the method of ordinary least squares computes the unique line that minimizes the sum of squared differences between the true data and that line. For specific mathematical reasons, this allows the researcher to estimate the conditional expectation of the dependent variable when the independent variables take on a given set of values. Less common forms of regression use slightly different procedures to estimate alternative location parameters or estimate the conditional expectation across a broader collection of non-linear models

Regression analysis is primarily used for two conceptually distinct purposes. First, regression analysis is widely used for prediction and forecasting, where its use has substantial overlap with the field of machine learning. Second, in some situations, regression analysis can be used to infer causal relationships between the independent and dependent variables. Importantly, regressions by themselves only reveal relationships between a dependent variable and a collection of independent variables in a fixed dataset. To use regressions for prediction or to infer causal relationships, respectively, a researcher must carefully justify why existing relationships have predictive power for a new context or why a relationship between two variables has a causal interpretation. The latter is especially important when researchers hope to estimate causal relationships using observational data.

Linear Regression-

Linear regression attempts to model the relationship between two variables by fitting a linear equation to observed data. One variable is considered to be an explanatory variable, and the other is considered to be a dependent variable. For example, a modeler might want to relate the weights of individuals to their heights using a linear regression model.

Before attempting to fit a linear model to observed data, a modeler should first determine whether or not there is a relationship between the variables of interest. This does not necessarily imply that one variable *causes* the other (for example, higher SAT scores do not *cause* higher college grades), but that there is some significant association between the two variables. A scatter plot of the A business based can be a helpful tool in determining the strength of the relationship between two variables. If there appears to be no association between the proposed explanatory and dependent variables (i.e., the scatter plot does not indicate any increasing or decreasing trends), then fitting a linear regression model to the data probably will not provide a useful model. A valuable numerical measure of association between two variables is the correlation concept, which is a value between -1 and 1 indicating the strength of the association of the observed data for the two variables.

A linear regression line has an equation of the form $Y = a + bX$, where X is the explanatory variable and Y is the dependent variable. The slope of the line is b , and a is the intercept (the value of y when $x = 0$).

Regularization-

There are extensions of the training of the linear model called regularization methods. These seek to both minimize the sum of the squared error of the model on the training data (using ordinary least squares) and also reduce the complexity of the model (like the number or absolute size of the sum of all coefficients in the model).

Two popular examples of regularization procedures for linear regression are:

Lasso Regression: where Ordinary Least Squares are modified to also minimize the absolute sum of the coefficients (called L1 regularization).

Ridge Regression: where Ordinary Least Squares are modified to also minimize the squared absolute sum of the coefficients (called L2 regularization).

These methods are effective to use when there is collinearity in your input values and ordinary least squares would overfit the training data.

Now that you know some techniques to learn the coefficients in a linear regression model, let's look at how we can use a model to make predictions on new data.

Support Vector Regression (SVR)-

Support Vector Machines (SVMs) are well known in classification problems. The use of SVMs in regression is not as well documented, however. These types of models are known as Support Vector Regression (SVR). Support Vector Regression (SVR) uses the same principle as SVM, but for regression problems. Let's spend a few minutes understanding the idea behind SVR.

The problem of regression is to find a function that approximates mapping from an input domain to real numbers based on the training sample. So let's now dive deep and understand how SVR works actually. Consider these two red lines as the decision boundary and the green line as the hyperplane. Our objective, when we are moving on with SVR, is to consider the points that are within the decision boundary line. Our best fit line is the hyperplane that has a maximum number of points.

The Support Vector Regression (SVR) uses the same principles as the SVM for classification, with only a few minor differences. First of all, because the output is a real number it becomes very difficult to predict the information at hand, which has infinite possibilities. In the case of regression, a margin of tolerance (epsilon) is set in approximation to the SVM which would have already been requested from the problem. But besides this fact, there is also a more complicated reason, the algorithm is more complicated and therefore to be taken into consideration. However, the main idea is always the same: to minimize error, individualizing the hyperplane which maximizes the margin, keeping in mind that part of the error is tolerated

Decision Tree-

A Decision Tree is a decision-making tool that uses a flowchart-like tree structure or is a model of decisions and all of their possible results, including outcomes, input costs, and utility.

Decision-tree algorithms fall under the category of supervised learning algorithms. It works for both continuous as well as categorical output variables.

The branches/edges represent the result of the node and the nodes have either:

Conditions [Decision Nodes]

Result [End Nodes]

Decision Tree Regression-

Decision tree regression observes the features of an object and trains a model in the structure of a tree to predict data in the future to produce meaningful continuous output. Continuous output means that the output/result is not discrete, i.e., it is not represented just by a discrete, known set of numbers or values

Discrete output example: A weather prediction model that predicts whether or not there'll be rain on a particular day.

Continuous output example: A profit prediction model that states the probable profit that can be generated from the sale of a product.

Here, continuous values are predicted with the help of a decision tree regression model.

Random Forest Regression-

A Random Forest is an ensemble technique capable of performing both regression and classification tasks with the use of multiple decision trees and a technique called Bootstrap Aggregation, commonly known as bagging. Bagging, in the Random Forest method, involves training each decision tree on a different data sample where sampling is done with replacement. The basic idea behind this is to combine multiple decision trees in determining the final output rather than relying on individual decision trees. If you want to read more on Random Forests, I have included some reference links which provide in-depth explanations on this topic.

First, we pass the features(X) and the dependent(y) variable values of the data set, to the method created for the random forest regression model. We then use the grid search cross-validation method from the sklearn library to determine the optimal values to be used for the hyperparameters of our model from a specified range of values. Here, we have chosen the two hyperparameters; *max_depth* and *n_estimators*, to be optimized. According to sklearn documentation, *max_depth* refers to the maximum depth of the tree, and *n_estimators*, the number of trees in the forest. Ideally, you can expect a better performance from your model when there are more trees. However, you must be cautious of the value ranges you specify and experiment using different values to see how your model performs. After creating a random forest regressor object, we pass it to the `cross_val_score()` function which performs K-Fold cross-validation disabled should contain a is given the based on disabled a disabled should download button the matching function has passed button the matching a function has passed provides as an output, an error metric value, which can be used to determine the model performance.

2.3 Java-

Java is a programming language and a platform. Java is a high-level, robust, object-oriented, and secure programming language.

Java was developed by Sun Microsystems (which is now the subsidiary of Oracle) in the year 1995. James Gosling is known as the father of Java. Before Java, its name was Oak. Since Oak was already a registered company, James Gosling and his team changed the Oak name to Java.

Why use Java?

- Java works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc.)
- It is one of the most popular programming languages in the world
- It is easy to learn and simple to use
- It is open-source and free
- It is secure, fast, and powerful
- It has huge community support (tens of millions of developers)
- Java is an object-oriented language that gives a clear structure to programs and allows code to be reused, lowering development costs
- As Java is close to C++ and C#, it makes it easy for programmers to switch to Java or vice versa

Java OOP Concepts-

Object means a real-world entity such as a pen, chair, table, computer, watch, etc. Object-Oriented Programming is a methodology or paradigm to design a program using classes and objects. It simplifies software development and maintenance by providing some concepts:

- Object
- Class
- Inheritance
- Polymorphism
- Abstraction
- Encapsulation

Object-

Any entity that has a state and behaviour is known as an object. For example, a chair, pen, table, keyboard, bike, etc. It can be physical or logical.

An Object can be defined as an instance of a class. An object contains an address and takes up some space in memory. Objects can communicate without knowing the details of each other's data or code. The only necessary thing is the type of message accepted and the type of response returned by the objects.

Example: A dog is an object because it has states like colour, name, breed, etc. as well as behaviours like wagging the tail, barking, eating, etc.

Class-

The collection of objects is called class. It is a logical entity.

A class can also be defined as a blueprint from which you can create an individual object. Class doesn't consume any space.

Inheritance-

When one object acquires all the properties and behaviours 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, a dog barks woof, etc

Abstraction-

Hiding internal details and showing functionality is known as abstraction. For example, in 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 medicines.

A java class is an example of encapsulation. Java bean is the fully encapsulated class because all the data members are private here

2.4 JDBC And Collections-

JDBC-

JDBC stands for Java Database Connectivity. JDBC is a Java API to connect and execute the query with the database. JDBC API uses JDBC drivers to connect with the database. There are four types of JDBC drivers:

- JDBC-ODBC Bridge Driver
- Native Driver
- Network Protocol Driver
- Thin Driver

Purpose of JDBC

The purposes of using JDBC are as follows-

- a) Establish a connection with a data source
- b) Send queries and update statements to the data source
- c) Process the results

The Java application calls JDBC classes and interfaces to submit SQL statements and retrieve results.

The JDBC API is implemented through the JDBC driver. The JDBC Driver is a set of classes that implement the JDBC interfaces to process JDBC calls and return result sets to a Java application. The main objects of JDBC API are the DataSource object, Connection object, Statement, PreparedStatement, and Callable Statement objects.

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.

Before the Collection Framework was introduced, the standard methods for grouping Java objects (or collections) were Arrays or Vectors, or Hash Tables. So, in a way, we can say that collections are synonymous with Arrays, Vectors, or Hash Tables. The different collection objects in java are Array List, LinkedList, Vector, HashSet, Linked Hash Set, Tree Set, HashMap, Tree, Map, Linked HashMap, Hash table.

Purpose of Collections: -

The purposes of using JAVA collections are as follows: -

--Arrays are not resizable. Java Collections Framework provides lots of different useful data types, such as hash sets, HashMap, and array lists which are resizable.

--Java Collections Framework provides abstractions, so you can refer to a list as a List, whether backed by an array list or a linked list.

--New data structures that conform to the standard collection interfaces are by nature reusable. The same goes for new algorithms that operate on objects that implement these interfaces.

2.5 React-

React is the world’s most popular JavaScript framework maintained by Facebook and a community of developers. It is used for building UI components. Most of the front-end parts of a project can be made efficient with react. It is used mostly because of how simple and efficient it makes for the user to make interactive UI components. It has the following advantages:

1. It is very simple. It helps us to design simple views for each of our states in our project and it renders only the essential components when there is any change in our data.
2. It helps us to keep our states outside the DOM and build complex UIs with the help of encapsulated components that manage their states.
3. We can develop new features without rewriting existing code.

React creates a virtual DOM in memory-In place of changing the browser directly react creates a virtual DOM in memory where it does the necessary manipulating before making any changes to the browser DOM.

React can be learned using HTML and requires NodeJS and NPM to be started. We will be using mostly ES6 for our project

Components-

Components are the building blocks of any react app and multiple components can be seen in a react project. React components serve the same purpose as that JavaScript functions but work in isolation and return HTML through the render() function.

A Higher-Order Component is an advanced technique in React for reusing the component logic. They are a pattern that emerges from React's compositional nature. The use of Higher Order of Components when you are architecturally ready for separating container components from presentation components.

Pure Components in React are the components that do not re-render when the value of state and props have been updated with the same

Values. If the value of the previous state and props and the new state or props is the same the component is not re-rendered. It is used mostly for performance optimization. If your React's component render() function renders the same result given the same props and state you can use React. Pure components for a performance boost in some cases.

Components are like JavaScript Functions-they accept arbitrary inputs and return and return a react element describing what should appear on the screen.

render()-takes input and returns what to display.

Components are divided into two types

Functional Components-

-A functional Component is just a plain JavaScript function that accepts props as input and returns a React element.

-There is no render method used in the Functional Component.

-Known as stateless components because they can simply accept data and display them in some form, they are mainly responsible for rendering UI.

Class components-

-A class component requires you to extend from React. Component and create a render function that returns a React element.

-It must have the render() method returning HTML.

-Known as Stateful components because they can implement logic and state.

-React lifecycle methods can be used inside class components.

HOOKS-

Hooks are the new feature that allows you to use state and other React features without writing a class. Hooks are the functions that "hook into" React state and lifecycle features from function components. It does not work inside classes. If you write a function component, and then you want to add some state to it, previously you do this by converting it to a class. But now you can do it by using a Hook inside the existing function component.

Some regulations to be taken care of while using hooks:

-We should not call hooks inside any loops, functions, or conditions. Hooks should be at the top level of functions.

-You cannot call Hooks from regular JavaScript functions. Instead, you can call Hooks from React function components

STATE-

A state is an instance of a react component class that can be defined as an object of a set of observable properties that control the behaviour of the component. In other words, the state of a component is an object that holds some information that may change over the lifetime of the component

- State of a component should prevail throughout the lifetime, thus we must first have some initial state, to do so we should define the State in the constructor of the component's class.

- State should never be updated explicitly. React uses an observable object as the state that observes what changes are made to the state and helps the component behave accordingly.

- React provides its own method `setState()`. `setState()` method takes a single parameter and expects an object which should contain the set of values to be updated. Once the update is done the method implicitly calls the `render()` method to repaint the page.

- The only time we are allowed to define the state explicitly is in the constructor to provide the initial state.

- React is highly efficient and thus uses asynchronous state updates i.e. React may update multiple `setState()` updates in a single go. Thus, using the value of the current state may not always generate the desired result.

- State updates are independent. The state object of a component may contain multiple attributes and React allows using the `setState()` function to update only a subset of those attributes as well as using multiple `setState()` methods to update each attribute value independently.

PROPS-

It is an object which stores the value of attributes of a tag and works similarly to the HTML attributes. It gives a way to pass data from one component to other components. It is similar to function arguments. Props are passed to the component in the same way as arguments passed in a function. In ReactJs we use props to send data components and every component is treated as a pure JavaScript function where props are equivalent to parameters of pure JavaScript functions.

Props are immutable so we cannot modify the props from inside the component. Inside the components, we can add attributes called props. These attributes are available in the component like this. props and can be used to render dynamic data in our render method. When you need immutable data in the component, you have to add props to `ReactDOM.render()` method in the main.js file of your ReactJS project and used it inside the component which you need.

REDUX-

Redux is a state management tool. With redux, the state of the application is kept in a store, and each component can access any state that it needs from this store. It allows React components to read data from a Redux Store, and dispatch Actions to the Store to update data. Redux helps apps to scale by providing a sensible way to manage the state through a unidirectional data flow model. React Redux is conceptually simple. It subscribes to the Redux store, checks to see if the data which your component wants have changed, and re-renders your component.

So primary reasons for using redux are:

- It keeps up to date with any API changes to ensure yours react components behave similarly throughout.

- encourages good react architecture.

- implements many performance optimizations internally, which allows components to re-render only when it needs to.

The Components of react-redux architecture are

- Store: A Store is a place where the entire state of your application lists. It manages the status of the application and has a dispatch(action) function.

- Action: It is sent or dispatched from the view which are payloads that can be read by Reducers. It is a pure object created to store the information of the user's event. It includes information such as type of action, time of occurrence, location of occurrence, its coordinates, and which state it aims to change.

- Reducer: The reducer reads the payloads from the actions and then updates the store via the state accordingly. It is a pure function to return a new state from the initial state.

AXIOS-

Axios is promise-based, which gives you the ability to take advantage of JavaScript's async and await for more readable asynchronous code.

The GET method requests a representation of the specified resource.

The POST method is used to submit an entity to the specified resource, often causing a change in state or side effects on the server.

.

Axios is designed to handle HTTP requests and responses. It's used more often than Fetch because it has a larger set of features and it supports older browsers.

Chapter 3

Problem Statement / Requirement Specifications

Problem Statement for Machine Learning

As a winter internship project, we will be building a web application to help the people working in the Accounts Receivable departments in their day-to-day activities. You need to build a web application where the users in the Account Receivable department can:

- View the invoice data from various buyers.
- See various fields/attributes of the invoice(s) from a particular buyer.
- Perform Data Pre-processing on the invoice data.
- Get account-level analytics to easily visualize and interpret data- EDA and Feature Engineering.
- Get a prediction of when the invoice is going to get paid

Problem Statement for Web App Development

The objective of the Web Application Development internship project is:

- To build a **Full-stack Invoice Management Application** using ReactJs, JDBC, Java, and Servlets.
- Build a **responsive Receivables Dashboard**.
- **Visualize Data** in the form of grids.
- **Visualize Data** in the form of graphs.
- Perform **Searching** operations on the invoices.
- **Add & Edit data** in the editable fields of the grid.
- **Delete data** of selected rows in predefined templates.

3.1 Project Planning

Specifically, below are the major aspects of the application that needs to be developed. The details for each of the below are provided in the functional overview section.

A. Data Loading in DB

We will be provided with an **invoices dataset** which we need to parse, process, and load in the provided database schemas.

B. UI Representation of the Data

- Build a responsive UI that can display the invoice data loaded from the database.
- The UI should support searching and pagination
- The UI should support editing some editable fields, adding a new row to the grid, deleting rows from the grid, and advanced search.

C. AI Support in the Application

- Add support for predicting the payment date for one or more invoice(s).
- UI should have a button to trigger the prediction of the payment date.
- The payment date needs to be persisted across sessions in the UI.

Estimated Starting Week	Activity	No. of Weeks
1	Basics Of Python.	1
2	Basics of Machine Learning.	1
3	Advance Concepts of Machine Learning and Submission of ML model.	2
5	Basics of Database Management System.	1
6	Introduction to JAVA and HTML	1
7	CSS and JavaScript	1
8	React, Redux, and material UI.	2
10	Project Submission.	Completed

3.2 Project Analysis

Machine Learning Part

For our project, we have analyzed how different components act on our data set to yield results that help us to draw several conclusions. In this chapter of the project report, we will highlight and discuss the results and related inferences.

Delay date prediction and bucketization

After importing our data set and performing exploratory data analysis (EDA), feature engineering (FE), feature selection (FS), and modeling we can conclude that:

Area business is an empty feature and does not support the modeling process.

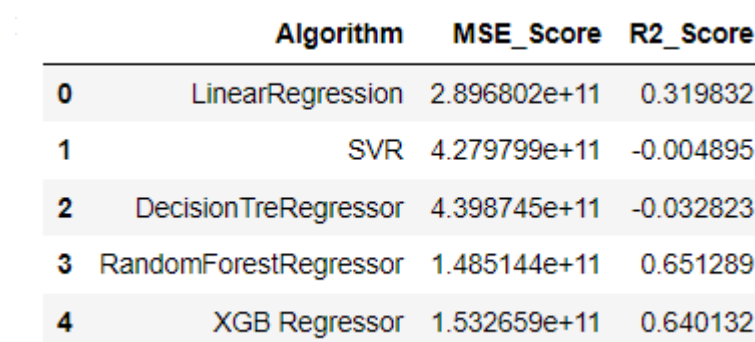
Doc_id and invoice id are just replicas of each other so one of them can be dropped.

Delay date can be calculated by subtracting clear_date from due_in_date.

The delay date shows good correlation with the following features:

'name_customer_target','cust_payment_terms','business_code','cust_number' so the model is trained over these features.

After performing the Linear Regression and other regression algorithms, the results obtained is



	Algorithm	MSE_Score	R2_Score
0	LinearRegression	2.896802e+11	0.319832
1	SVR	4.279799e+11	-0.004895
2	DecisionTreRegressor	4.398745e+11	-0.032823
3	RandomForestRegressor	1.485144e+11	0.651289
4	XGB Regressor	1.532659e+11	0.640132

Fig 3.1: Algorithm Accuracies

Java Servlets

Java Servlet helps us to connect our database to our user interface. All the data is fetched with the help of servlets and placed in the user interface (UI). All the servers which were made are:

- addServlet
- FetchServlet
- updateServlet
- deleteServlet

User Interface with React

After having our servlets ready we build our User Interface using react and Material UI

The final User Interface looks like this

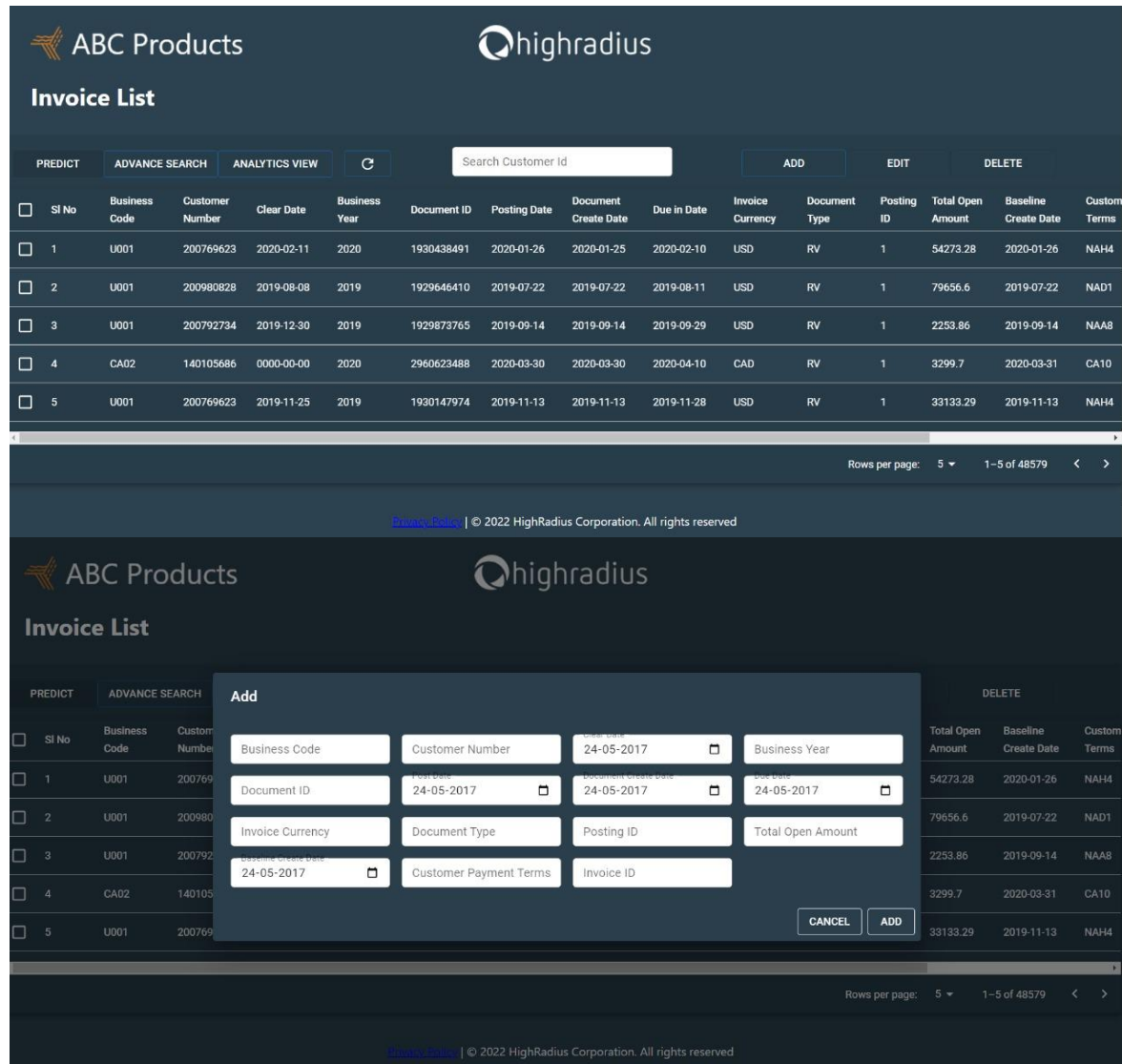


Fig: 3.2: Project UI dashboard

3.3 System Design

3.3.1 Design Constraints/ Tools Used

1. PYTHON: Python is an interpreted high-level general-purpose programming language. To build the machine learning prediction model we used the following libraries- Pandas, NumPy, Seaborn, Scikit-learn.

2 HTML: Hypertext Markup Language is the fundamental markup language that helped us build the website for this project. It allowed us to create and structure sections for the web page. Being a markup language, HTML does not provide dynamic functionality instead its salient features are its ability to format and organize documents.

1. CSS: Cascading Style Sheets is a design language that allows users to present web pages in a beautiful manner using different colors and styles for texts, backgrounds, headlines, and paragraphs.

2. JavaScript: JavaScript, being a dynamic computer programming language, allows interaction between client-side script and user which makes the webpage dynamic. JavaScript also has object-oriented programming capabilities. When used for the frontend, JavaScript provides a platform to make the webpage more interactive such as image carousels and converting menus to dropdowns, and making any page responsive.

3. JSP: Java Server Pages belong to Java web technologies and are server-side technologies that help in creating dynamic and platform-independent web pages. Using this, Java code can be inserted into HTML.

4. JDK: The Java Development Kit (JDK) is a software development environment used for developing Java applications and applets. It includes the Java Runtime Environment (JRE), an interpreter/loader (java), a compiler (javac), an archiver (jar), a documentation generator (Javadoc) and other tools needed in Java development.

5. JRE: The Java Runtime Environment, or JRE, is a software layer that runs on top of a computer's operating system software and provides the class libraries and other resources that a specific Java program needs to run.

6. Apache Tomcat: Apache Tomcat is and is an open-source software where Java Servlet, Java Server Pages, Java Expression Language, and WebSocket technologies can be implemented. Tomcat provided us with a pure Java system environment where Java code could be run.

7. SQL yog: SQL yog Ultimate is the most powerful manager, admin, and GUI tool for MySQL, combining the features of MySQL Query Browser and MySQL GUI tools in a single intuitive interface. SQL yog is a fast, easy-to-use, and compact graphical tool for managing your MySQL databases.

8. Apache Tomcat: Apache Tomcat is an open-source software where Java Servlet, Java Server Pages, Java Expression Language, and WebSocket technologies can be implemented. Tomcat provided us with a pure Java system environment where Java code could be run.

9. SQLyog: SQLyog Ultimate is the most powerful manager, admin, and GUI tool for MySQL, combining the features of MySQL Query Browser and MySQL GUI tools in a single intuitive interface. SQLyog is a fast, easy-to-use, and compact graphical tool for managing your MySQL databases.

10. Eclipse: Eclipse is an integrated development environment (IDE) used in computer programming. It contains a base workspace and an extensible plug-in system for customizing the environment. Eclipse is written mostly in Java and its primary use is for developing Java applications.

11. Visual Studio Code: Visual Studio Code is a source code editor that can be used with a variety of programming languages, including Java, JavaScript, Go, Node.js, Python, and C++. Instead of a project system, it allows users to open one or more directories, which can then be saved in workspaces for future reuse.

12. JSX: JSX stands for JavaScript XML. It is a JavaScript syntax extension wherein the developer can write HTML codes in JavaScript. React processes this extended syntax into JavaScript calls. This enables the functionalities of HTML in the application to be used with JavaScript. This syntax extension makes the code easier to write and debug as it.

3.3.2 Block Diagram

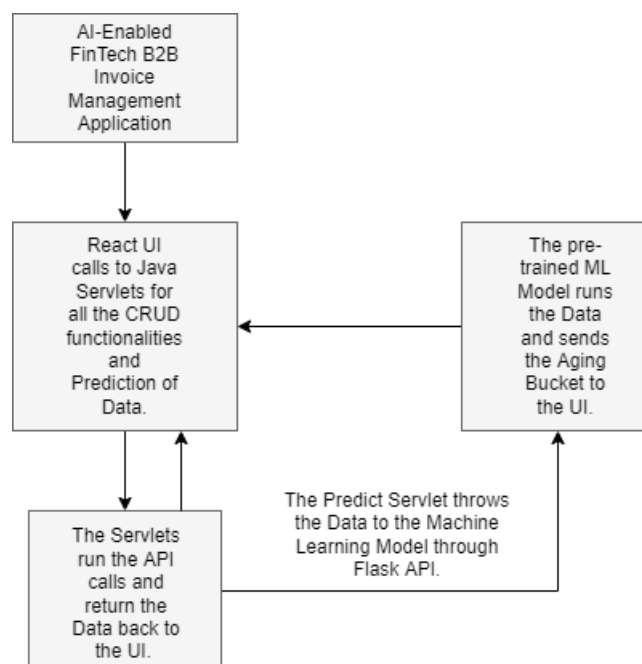


Fig 3.3: Block diagram of project

Chapter 4

Implementation

4.1 Methodology OR Proposal

About Dataset

For our project, I used an invoices dataset that contains the past payment information and behavior of various buyers. The dataset was in csv format and the dataset contained the following columns: -

- - Company ID
- - Document Number
- - Business Code
- - Create Year
- - Document type
- - Customer Number
- - Customer Name
- - Document Create Date
- - Posting date
- - Due In Date
- - Order Create Date
- - Invoice ID
- - Baseline Create Date
- - Total Open Amount

Feature Engineering

All the features are categorized in parts, then all the features which comes under the category will be sequenced and followed by the reason of using it. Now we have to represent the categories for which categorical columns need to be created. We use three methods to create categorical column: -

1. Label Encoding: Map each category/ unique value to a numeric value such as A:1,B:2,C:3

2. One hot encoding: Here, we map each category to a vector that contains 1 and 0 denoting the presence of the feature or not. The number of vectors depends on the categories which we have in our dataset. For high cardinality features, this method produces a lot of columns that slow down the learning of the model significantly.

3. Target Encoding: Here, features are replaced with a blend of the posterior probability of the target for the given particular categorical value and the prior probability of the target over all the training data. Also, they are not generated For the test data. We usually save the target encodings obtained from the training data set and use the same encodings to encode features in the test data set.

Feature selection

Feature selection is one of the most important processes in machine learning algorithms. It uses a greedy approach and evaluates all possible combinations of features to give us the best score. There are different approaches to go for feature selection i.e. manually, wrapper method and embedded method.

A correlation matrix was used to find the best possible features manually, and business_code, cust_number, cust_payment_terms and the difference between due_in_date and the baseline_create_date was chosen as the appropriate features.

After that the model was evaluated. Three main metrics are used for model evaluation :-

1. Accuracy
2. Precision
3. Recall

For the test set feature calculations, we predict one prediction at a time and then that prediction was used to calculate the delay

Modelling

Decision Tree, Linear Regression, Support vector machines(SVMs) and Random Forest Regressor were used with different feature sets and the models were evaluated on the basis of R2_score and RMSE_score. After the evaluation the models were used to predict when the payments were cleared by the customers.

Objective of Java and React:

Concept Details

1. HTTP Methods are as following:

- GET request: The GET method is used to get a representation of a resource. Requests made with the GET method should only return data.
- Post request: The POST method is used to submit an object to the defined resource, which often causes the server's state to alter or has side effects

- PUT request: All current representations of the target resource are replaced with the request payload when using the PUT process.

- DELETE request: The DELETE method deletes the specified resources.

2. Callbacks: -

Callbacks are functions that are transferred to another function as an argument. Once an incident has occurred or a specific mission has been completed. In asynchronous code, it's frequently used. Later on, callback functions are named.

3. Promises -: A promise is an entity that has the potential to generate a single value in the future: either a resolved value or a reason why it hasn't been resolved yet (e.g., a network error occurred). A promise may be fulfilled, refused, or pending in one of three ways. Users may use callbacks to manage the fulfilled value or the reason for rejection when using Promise.

A promise has 3 stages-

- Fulfilled- if the request is fulfilled
- Rejected- if the request is rejected
- Pending- the request is not yet fulfilled

These were the functions that were used to make the API request to get required data from the database.

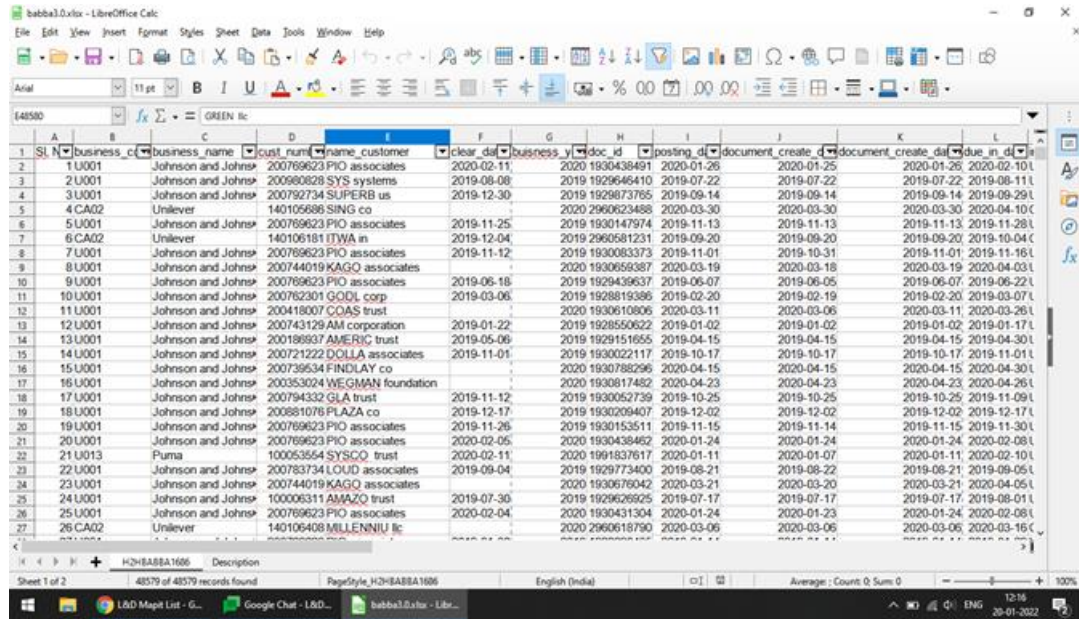
Backend:

Data Loading in the Database:

Step 1: Execute the SQL script for the creation of the table.

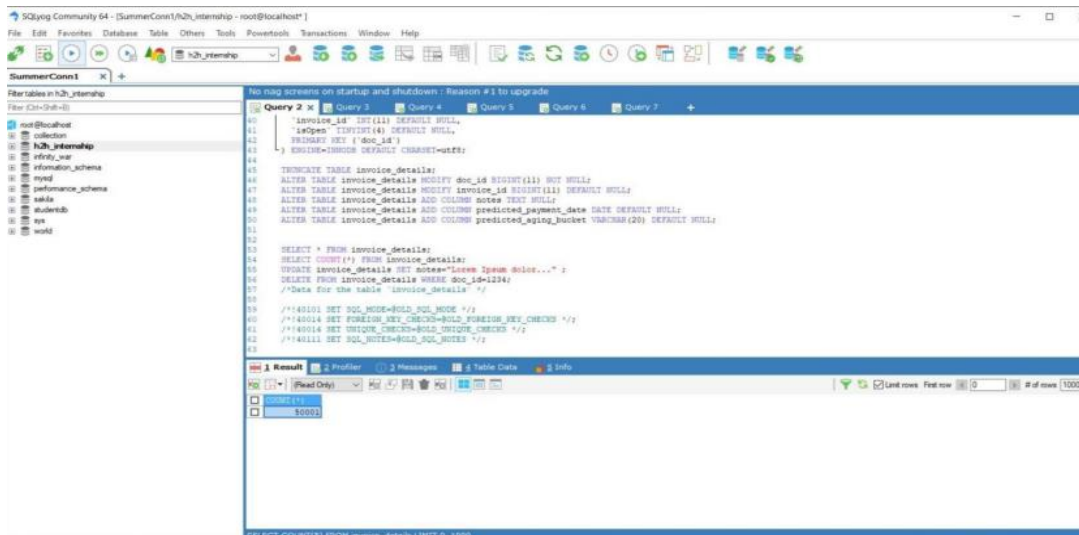
Step 2: Read the CSV datasheet using a CSV reader and stored information.

Step 3: We used a JDBC driver and also made a POJO class which helped us load the datasheet into the database in batches



Sl. No.	business_id	business_name	cust_num	name_customer	clear_date	business_yr	doc_id	posting_date	document_create_date	document_create_date	due_in_date
1	U001	Johnson and Johnson	200769623	PIO associates	2020-02-11	2020	1930438491	2020-01-26	2020-01-26	2020-02-10	
2	U001	Johnson and Johnson	200980528	SYS systems	2019-08-08	2019	1929646410	2019-07-22	2019-07-22	2019-08-11	
3	U001	Johnson and Johnson	200792734	SUPERB us	2019-12-30	2019	1929873785	2019-09-14	2019-09-14	2019-09-29	
4	CA02	Unilever	140105686	SING co		2020	2960623488	2020-03-30	2020-03-30	2020-04-10	
5	U001	Johnson and Johnson	200769623	PIO associates	2019-11-25	2019	1930147974	2019-11-13	2019-11-13	2019-11-28	
6	CA02	Unilever	140106181	ITWA in	2019-12-04	2019	2960581231	2019-09-20	2019-09-20	2019-10-04	
7	U001	Johnson and Johnson	200769623	PIO associates	2019-11-12	2019	1930083373	2019-11-01	2019-10-31	2019-11-16	
8	U001	Johnson and Johnson	200744019	KAGQ associates		2020	1930656387	2020-03-19	2020-03-18	2020-04-03	
9	U001	Johnson and Johnson	200769623	PIO associates	2019-06-18	2019	1929439637	2019-06-07	2019-06-05	2019-06-22	
10	U001	Johnson and Johnson	200762301	GODL corp	2019-03-06	2019	1928819386	2019-02-20	2019-02-19	2019-03-07	
11	U001	Johnson and Johnson	200418007	COAS trust		2020	1930610806	2020-03-11	2020-03-06	2020-03-26	
12	U001	Johnson and Johnson	200743129	AM corporation	2019-01-22	2019	1928550622	2019-01-02	2019-01-02	2019-01-17	
13	U001	Johnson and Johnson	200198937	AMERJC trust	2019-05-06	2019	1929151655	2019-04-15	2019-04-15	2019-04-30	
14	U001	Johnson and Johnson	200721222	DOLLA associates	2019-11-01	2019	1930022117	2019-10-17	2019-10-17	2019-11-01	
15	U001	Johnson and Johnson	200739534	FINDLAY co		2020	1930788296	2020-04-15	2020-04-15	2020-04-30	
16	U001	Johnson and Johnson	200353024	WEGMAN foundation		2020	1930817482	2020-04-23	2020-04-23	2020-04-26	
17	U001	Johnson and Johnson	200794332	GLA trust	2019-11-12	2019	1930062739	2019-10-25	2019-10-25	2019-11-09	
18	U001	Johnson and Johnson	200881076	PLAZA co	2019-12-17	2019	1930209407	2019-12-02	2019-12-02	2019-12-17	
19	U001	Johnson and Johnson	200769623	PIO associates	2019-11-26	2019	1930153511	2019-11-15	2019-11-14	2019-11-30	
20	U001	Johnson and Johnson	200769623	PIO associates	2020-02-05	2020	1930438482	2020-01-24	2020-01-24	2020-02-08	
21	U013	Puma	100053564	SYSCQ trust	2020-02-11	2020	1991837617	2020-01-11	2020-01-07	2020-02-10	
22	U001	Johnson and Johnson	200783734	LOUD associates	2019-09-04	2019	1929773400	2019-08-21	2019-08-21	2019-09-05	
23	U001	Johnson and Johnson	200744019	KAGQ associates		2020	1930678042	2020-03-21	2020-03-20	2020-04-05	
24	U001	Johnson and Johnson	100006311	AMAZO trust	2019-07-30	2019	1929626925	2019-07-17	2019-07-17	2019-08-01	
25	U001	Johnson and Johnson	200769623	PIO associates	2020-02-04	2020	1930431304	2020-01-24	2020-01-23	2020-01-24	
26	CA02	Unilever	140106408	MILLENNIUM llc		2020	2960618790	2020-03-06	2020-03-06	2020-03-16	

Fig 4.1: CSV file



```

-- Query 2
1. CREATE TABLE `invoice_details` (
  `invoice_id` INT(11) NOT NULL,
  `business_id` INT(11) NOT NULL,
  `cust_num` INT(11) NOT NULL,
  `name_customer` VARCHAR(255) NOT NULL,
  `clear_date` DATE NOT NULL,
  `business_yr` INT(11) NOT NULL,
  `doc_id` VARCHAR(255) NOT NULL,
  `posting_date` DATE NOT NULL,
  `document_create_date` DATE NOT NULL,
  `document_create_date` DATE NOT NULL,
  `due_in_date` DATE NOT NULL,
  PRIMARY KEY (`invoice_id`),
  INDEX (`business_id`, `cust_num`, `name_customer`, `clear_date`, `business_yr`, `doc_id`, `posting_date`, `document_create_date`, `document_create_date`, `due_in_date`))
2. TRUNCATE TABLE `invoice_details`;
3. ALTER TABLE `invoice_details` MODIFY `doc_id` VARCHAR(255) NOT NULL;
4. ALTER TABLE `invoice_details` MODIFY `invoice_id` INT(11) NOT NULL;
5. ALTER TABLE `invoice_details` ADD COLUMN `notes` TEXT NOT NULL;
6. ALTER TABLE `invoice_details` ADD COLUMN `predicted_payment_date` DATE NOT NULL;
7. ALTER TABLE `invoice_details` ADD COLUMN `predicted_aging_bucket` VARCHAR(255) NOT NULL;
8. SELECT * FROM `invoice_details`;
9. SELECT COUNT(*) FROM `invoice_details`;
10. UPDATE `invoice_details` SET `notes`="Lorem Ipsum dolor..." ;
11. DELETE FROM `invoice_details` WHERE `doc_id`=1234;
12. --Data for the table `invoice_details` --
13. /**40001 SET SQL_MODE="NO_AUTO_VALUE_ON_ZERO" */
14. /**40004 SET FOREIGN_KEY_CHECKS=0; */
15. /**40014 SET UNION_CHECKS=0; */
16. /**40011 SET SQL_MODE="NO_AUTO_VALUE_ON_ZERO" */
17.
-- Query 3
1. INSERT INTO `invoice_details` (
  `invoice_id`,
  `business_id`,
  `cust_num`,
  `name_customer`,
  `clear_date`,
  `business_yr`,
  `doc_id`,
  `posting_date`,
  `document_create_date`,
  `document_create_date`,
  `due_in_date`,
  `notes`,
  `predicted_payment_date`,
  `predicted_aging_bucket`
) VALUES (
  1,
  1,
  1,
  'Johnson and Johnson',
  '2020-02-11',
  2020,
  '1930438491',
  '2020-01-26',
  '2020-01-26',
  '2020-01-26',
  '2020-02-10',
  'PIO associates',
  '2020-02-10',
  '2020-02-10'
);
2. INSERT INTO `invoice_details` (
  `invoice_id`,
  `business_id`,
  `cust_num`,
  `name_customer`,
  `clear_date`,
  `business_yr`,
  `doc_id`,
  `posting_date`,
  `document_create_date`,
  `document_create_date`,
  `due_in_date`,
  `notes`,
  `predicted_payment_date`,
  `predicted_aging_bucket`
) VALUES (
  2,
  1,
  1,
  'Johnson and Johnson',
  '2019-08-08',
  2019,
  '1929646410',
  '2019-07-22',
  '2019-07-22',
  '2019-07-22',
  '2019-08-11',
  'SYS systems',
  '2019-08-11',
  '2019-08-11'
);
3. INSERT INTO `invoice_details` (
  `invoice_id`,
  `business_id`,
  `cust_num`,
  `name_customer`,
  `clear_date`,
  `business_yr`,
  `doc_id`,
  `posting_date`,
  `document_create_date`,
  `document_create_date`,
  `due_in_date`,
  `notes`,
  `predicted_payment_date`,
  `predicted_aging_bucket`
) VALUES (
  3,
  1,
  1,
  'Johnson and Johnson',
  '2019-12-30',
  2019,
  '1929873785',
  '2019-09-14',
  '2019-09-14',
  '2019-09-14',
  '2019-09-29',
  'SUPERB us',
  '2019-09-29',
  '2019-09-29'
);
4. INSERT INTO `invoice_details` (
  `invoice_id`,
  `business_id`,
  `cust_num`,
  `name_customer`,
  `clear_date`,
  `business_yr`,
  `doc_id`,
  `posting_date`,
  `document_create_date`,
  `document_create_date`,
  `due_in_date`,
  `notes`,
  `predicted_payment_date`,
  `predicted_aging_bucket`
) VALUES (
  4,
  2,
  1,
  'Unilever',
  '2020-03-30',
  2020,
  '2960623488',
  '2020-03-30',
  '2020-03-30',
  '2020-03-30',
  '2020-04-10',
  'SING co',
  '2020-04-10',
  '2020-04-10'
);
5. INSERT INTO `invoice_details` (
  `invoice_id`,
  `business_id`,
  `cust_num`,
  `name_customer`,
  `clear_date`,
  `business_yr`,
  `doc_id`,
  `posting_date`,
  `document_create_date`,
  `document_create_date`,
  `due_in_date`,
  `notes`,
  `predicted_payment_date`,
  `predicted_aging_bucket`
) VALUES (
  5,
  1,
  1,
  'Johnson and Johnson',
  '2019-11-25',
  2019,
  '1930147974',
  '2019-11-13',
  '2019-11-13',
  '2019-11-13',
  '2019-11-28',
  'PIO associates',
  '2019-11-28',
  '2019-11-28'
);
6. INSERT INTO `invoice_details` (
  `invoice_id`,
  `business_id`,
  `cust_num`,
  `name_customer`,
  `clear_date`,
  `business_yr`,
  `doc_id`,
  `posting_date`,
  `document_create_date`,
  `document_create_date`,
  `due_in_date`,
  `notes`,
  `predicted_payment_date`,
  `predicted_aging_bucket`
) VALUES (
  6,
  2,
  1,
  'Unilever',
  '2019-12-04',
  2019,
  '2960581231',
  '2019-09-20',
  '2019-09-20',
  '2019-09-20',
  '2019-10-04',
  'ITWA in',
  '2019-10-04',
  '2019-10-04'
);
7. INSERT INTO `invoice_details` (
  `invoice_id`,
  `business_id`,
  `cust_num`,
  `name_customer`,
  `clear_date`,
  `business_yr`,
  `doc_id`,
  `posting_date`,
  `document_create_date`,
  `document_create_date`,
  `due_in_date`,
  `notes`,
  `predicted_payment_date`,
  `predicted_aging_bucket`
) VALUES (
  7,
  1,
  1,
  'Johnson and Johnson',
  '2019-11-12',
  2019,
  '1930083373',
  '2019-11-01',
  '2019-10-31',
  '2019-10-31',
  '2019-11-16',
  'PIO associates',
  '2019-11-16',
  '2019-11-16'
);
8. INSERT INTO `invoice_details` (
  `invoice_id`,
  `business_id`,
  `cust_num`,
  `name_customer`,
  `clear_date`,
  `business_yr`,
  `doc_id`,
  `posting_date`,
  `document_create_date`,
  `document_create_date`,
  `due_in_date`,
  `notes`,
  `predicted_payment_date`,
  `predicted_aging_bucket`
) VALUES (
  8,
  1,
  1,
  'Johnson and Johnson',
  '2020-03-19',
  2020,
  '1930656387',
  '2020-03-19',
  '2020-03-18',
  '2020-03-18',
  '2020-04-03',
  'KAGQ associates',
  '2020-04-03',
  '2020-04-03'
);
9. INSERT INTO `invoice_details` (
  `invoice_id`,
  `business_id`,
  `cust_num`,
  `name_customer`,
  `clear_date`,
  `business_yr`,
  `doc_id`,
  `posting_date`,
  `document_create_date`,
  `document_create_date`,
  `due_in_date`,
  `notes`,
  `predicted_payment_date`,
  `predicted_aging_bucket`
) VALUES (
  9,
  1,
  1,
  'Johnson and Johnson',
  '2019-06-07',
  2019,
  '1929439637',
  '2019-06-07',
  '2019-06-05',
  '2019-06-05',
  '2019-06-22',
  'PIO associates',
  '2019-06-22',
  '2019-06-22'
);
10. INSERT INTO `invoice_details` (
  `invoice_id`,
  `business_id`,
  `cust_num`,
  `name_customer`,
  `clear_date`,
  `business_yr`,
  `doc_id`,
  `posting_date`,
  `document_create_date`,
  `document_create_date`,
  `due_in_date`,
  `notes`,
  `predicted_payment_date`,
  `predicted_aging_bucket`
) VALUES (
  10,
  1,
  1,
  'Johnson and Johnson',
  '2019-02-20',
  2019,
  '1928819386',
  '2019-02-20',
  '2019-02-19',
  '2019-02-19',
  '2019-03-07',
  'GODL corp',
  '2019-03-07',
  '2019-03-07'
);
11. INSERT INTO `invoice_details` (
  `invoice_id`,
  `business_id`,
  `cust_num`,
  `name_customer`,
  `clear_date`,
  `business_yr`,
  `doc_id`,
  `posting_date`,
  `document_create_date`,
  `document_create_date`,
  `due_in_date`,
  `notes`,
  `predicted_payment_date`,
  `predicted_aging_bucket`
) VALUES (
  11,
  1,
  1,
  'Johnson and Johnson',
  '2020-03-11',
  2020,
  '1930610806',
  '2020-03-11',
  '2020-03-06',
  '2020-03-06',
  '2020-03-26',
  'COAS trust',
  '2020-03-26',
  '2020-03-26'
);
12. INSERT INTO `invoice_details` (
  `invoice_id`,
  `business_id`,
  `cust_num`,
  `name_customer`,
  `clear_date`,
  `business_yr`,
  `doc_id`,
  `posting_date`,
  `document_create_date`,
  `document_create_date`,
  `due_in_date`,
  `notes`,
  `predicted_payment_date`,
  `predicted_aging_bucket`
) VALUES (
  12,
  1,
  1,
  'Johnson and Johnson',
  '2019-01-02',
  2019,
  '1928550622',
  '2019-01-02',
  '2019-01-02',
  '2019-01-02',
  '2019-01-17',
  'AM corporation',
  '2019-01-17',
  '2019-01-17'
);
13. INSERT INTO `invoice_details` (
  `invoice_id`,
  `business_id`,
  `cust_num`,
  `name_customer`,
  `clear_date`,
  `business_yr`,
  `doc_id`,
  `posting_date`,
  `document_create_date`,
  `document_create_date`,
  `due_in_date`,
  `notes`,
  `predicted_payment_date`,
  `predicted_aging_bucket`
) VALUES (
  13,
  1,
  1,
  'Johnson and Johnson',
  '2019-04-15',
  2019,
  '1929151655',
  '2019-04-15',
  '2019-04-15',
  '2019-04-15',
  '2019-04-30',
  'AMERJC trust',
  '2019-04-30',
  '2019-04-30'
);
14. INSERT INTO `invoice_details` (
  `invoice_id`,
  `business_id`,
  `cust_num`,
  `name_customer`,
  `clear_date`,
  `business_yr`,
  `doc_id`,
  `posting_date`,
  `document_create_date`,
  `document_create_date`,
  `due_in_date`,
  `notes`,
  `predicted_payment_date`,
  `predicted_aging_bucket`
) VALUES (
  14,
  1,
  1,
  'Johnson and Johnson',
  '2019-10-17',
  2019,
  '1930022117',
  '2019-10-17',
  '2019-10-17',
  '2019-10-17',
  '2019-11-01',
  'DOLLA associates',
  '2019-11-01',
  '2019-11-01'
);
15. INSERT INTO `invoice_details` (
  `invoice_id`,
  `business_id`,
  `cust_num`,
  `name_customer`,
  `clear_date`,
  `business_yr`,
  `doc_id`,
  `posting_date`,
  `document_create_date`,
  `document_create_date`,
  `due_in_date`,
  `notes`,
  `predicted_payment_date`,
  `predicted_aging_bucket`
) VALUES (
  15,
  1,
  1,
  'Johnson and Johnson',
  '2020-04-15',
  2020,
  '1930788296',
  '2020-04-15',
  '2020-04-15',
  '2020-04-15',
  '2020-04-30',
  'FINDLAY co',
  '2020-04-30',
  '2020-04-30'
);
16. INSERT INTO `invoice_details` (
  `invoice_id`,
  `business_id`,
  `cust_num`,
  `name_customer`,
  `clear_date`,
  `business_yr`,
  `doc_id`,
  `posting_date`,
  `document_create_date`,
  `document_create_date`,
  `due_in_date`,
  `notes`,
  `predicted_payment_date`,
  `predicted_aging_bucket`
) VALUES (
  16,
  1,
  1,
  'Johnson and Johnson',
  '2020-04-23',
  2020,
  '1930817482',
  '2020-04-23',
  '2020-04-23',
  '2020-04-23',
  '2020-04-26',
  'WEGMAN foundation',
  '2020-04-26',
  '2020-04-26'
);
17. INSERT INTO `invoice_details` (
  `invoice_id`,
  `business_id`,
  `cust_num`,
  `name_customer`,
  `clear_date`,
  `business_yr`,
  `doc_id`,
  `posting_date`,
  `document_create_date`,
  `document_create_date`,
  `due_in_date`,
  `notes`,
  `predicted_payment_date`,
  `predicted_aging_bucket`
) VALUES (
  17,
  1,
  1,
  'Johnson and Johnson',
  '2019-10-25',
  2019,
  '1930062739',
  '2019-10-25',
  '2019-10-25',
  '2019-10-25',
  '2019-11-09',
  'GLA trust',
  '2019-11-09',
  '2019-11-09'
);
18. INSERT INTO `invoice_details` (
  `invoice_id`,
  `business_id`,
  `cust_num`,
  `name_customer`,
  `clear_date`,
  `business_yr`,
  `doc_id`,
  `posting_date`,
  `document_create_date`,
  `document_create_date`,
  `due_in_date`,
  `notes`,
  `predicted_payment_date`,
  `predicted_aging_bucket`
) VALUES (
  18,
  1,
  1,
  'Johnson and Johnson',
  '2019-12-02',
  2019,
  '1930209407',
  '2019-12-02',
  '2019-12-02',
  '2019-12-02',
  '2019-12-17',
  'PLAZA co',
  '2019-12-17',
  '2019-12-17'
);
19. INSERT INTO `invoice_details` (
  `invoice_id`,
  `business_id`,
  `cust_num`,
  `name_customer`,
  `clear_date`,
  `business_yr`,
  `doc_id`,
  `posting_date`,
  `document_create_date`,
  `document_create_date`,
  `due_in_date`,
  `notes`,
  `predicted_payment_date`,
  `predicted_aging_bucket`
) VALUES (
  19,
  1,
  1,
  'Johnson and Johnson',
  '2019-11-15',
  2019,
  '1930153511',
  '2019-11-15',
  '2019-11-14',
  '2019-11-14',
  '2019-11-30',
  'PIO associates',
  '2019-11-30',
  '2019-11-30'
);
20. INSERT INTO `invoice_details` (
  `invoice_id`,
  `business_id`,
  `cust_num`,
  `name_customer`,
  `clear_date`,
  `business_yr`,
  `doc_id`,
  `posting_date`,
  `document_create_date`,
  `document_create_date`,
  `due_in_date`,
  `notes`,
  `predicted_payment_date`,
  `predicted_aging_bucket`
) VALUES (
  20,
  1,
  1,
  'Johnson and Johnson',
  '2020-01-24',
  2020,
  '1930438482',
  '2020-01-24',
  '2020-01-24',
  '2020-01-24',
  '2020-02-08',
  'PIO associates',
  '2020-02-08',
  '2020-02-08'
);
21. INSERT INTO `invoice_details` (
  `invoice_id`,
  `business_id`,
  `cust_num`,
  `name_customer`,
  `clear_date`,
  `business_yr`,
  `doc_id`,
  `posting_date`,
  `document_create_date`,
  `document_create_date`,
  `due_in_date`,
  `notes`,
  `predicted_payment_date`,
  `predicted_aging_bucket`
) VALUES (
  21,
  13,
  1,
  'Puma',
  '2020-01-11',
  2020,
  '1991837617',
  '2020-01-11',
  '2020-01-07',
  '2020-01-07',
  '2020-02-10',
  'SYSCQ trust',
  '2020-02-10',
  '2020-02-10'
);
22. INSERT INTO `invoice_details` (
  `invoice_id`,
  `business_id`,
  `cust_num`,
  `name_customer`,
  `clear_date`,
  `business_yr`,
  `doc_id`,
  `posting_date`,
  `document_create_date`,
  `document_create_date`,
  `due_in_date`,
  `notes`,
  `predicted_payment_date`,
  `predicted_aging_bucket`
) VALUES (
  22,
  1,
  1,
  'Johnson and Johnson',
  '2019-08-21',
  2019,
  '1929773400',
  '2019-08-21',
  '2019-08-21',
  '2019-08-21',
  '2019-09-05',
  'LOUD associates',
  '2019-09-05',
  '2019-09-05'
);
23. INSERT INTO `invoice_details` (
  `invoice_id`,
  `business_id`,
  `cust_num`,
  `name_customer`,
  `clear_date`,
  `business_yr`,
  `doc_id`,
  `posting_date`,
  `document_create_date`,
  `document_create_date`,
  `due_in_date`,
  `notes`,
  `predicted_payment_date`,
  `predicted_aging_bucket`
) VALUES (
  23,
  1,
  1,
  'Johnson and Johnson',
  '2020-03-21',
  2020,
  '1930678042',
  '2020-03-21',
  '2020-03-20',
  '2020-03-20',
  '2020-04-05',
  'KAGQ associates',
  '2020-04-05',
  '2020-04-05'
);
24. INSERT INTO `invoice_details` (
  `invoice_id`,
  `business_id`,
  `cust_num`,
  `name_customer`,
  `clear_date`,
  `business_yr`,
  `doc_id`,
  `posting_date`,
  `document_create_date`,
  `document_create_date`,
  `due_in_date`,
  `notes`,
  `predicted_payment_date`,
  `predicted_aging_bucket`
) VALUES (
  24,
  1,
  1,
  'Johnson and Johnson',
  '2019-07-17',
  2019,
  '1929626925',
  '2019-07-17',
  '2019-07-17',
  '2019-07-17',
  '2019-08-01',
  'AMAZO trust',
  '2019-08-01',
  '2019-08-01'
);
25. INSERT INTO `invoice_details` (
  `invoice_id`,
  `business_id`,
  `cust_num`,
  `name_customer`,
  `clear_date`,
  `business_yr`,
  `doc_id`,
  `posting_date`,
  `document_create_date`,
  `document_create_date`,
  `due_in_date`,
  `notes`,
  `predicted_payment_date`,
  `predicted_aging_bucket`
) VALUES (
  25,
  1,
  1,
  'Johnson and Johnson',
  '2020-01-24',
  2020,
  '1930431304',
  '2020-01-24',
  '2020-01-23',
  '2020-01-23',
  '2020-01-24',
  'PIO associates',
  '2020-01-24',
  '2020-01-24'
);
26. INSERT INTO `invoice_details` (
  `invoice_id`,
  `business_id`,
  `cust_num`,
  `name_customer`,
  `clear_date`,
  `business_yr`,
  `doc_id`,
  `posting_date`,
  `document_create_date`,
  `document_create_date`,
  `due_in_date`,
  `notes`,
  `predicted_payment_date`,
  `predicted_aging_bucket`
) VALUES (
  26,
  2,
  1,
  'Unilever',
  '2020-03-06',
  2020,
  '2960618790',
  '2020-03-06',
  '2020-03-06',
  '2020-03-06',
  '2020-03-16',
  'MILLENNIUM llc',
  '2020-03-16',
  '2020-03-16'
);

```

Fig 4.2: Data Inserted in SQLYog

Servlets:

It is a technology that resides at the server side and is responsible for generating dynamic web pages is known as a servlet, in other words it helps create web applications.

There are many interfaces and classes in the Servlet API such as Servlet, GenericServlet, HttpServlet, ServletRequest, ServletResponse, etc.

A servlet is used to extend the capabilities of servers that host applications accessed by means of a request-response programming model. Servlets are mainly used to extend the applications hosted by web servers, however, they can respond to other types of requests too. For such applications, HTTP-specific servlet classes are defined by Java Servlet technology

Servlet Life Cycle

The Servlet life cycle mainly includes the following four stages,

- Loading a Servlet
- Initializing the Servlet
- Request handling.
- Destroying the Servlet

Servlet Creation:

So after the UI is made some actions (add, edit, delete etc.) need to perform. So using the help of the servlets given below we can make those actions happen.

- 1) Add servlet - Get a POST request from the frontend with parameters such as invoice amount, notes, date, etc., and pass them to the SQL database.
- 2) Edit Servlet - GET a 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.
- 3) Delete Servlet - Delete the selected invoices from the database bypassing their respective doc_id's to identify them in the database.
- 4) Search Servlet - Get the invoice number from the frontend and pass them as an HTTP request using Axios to the backend and search through the database and return it to the frontend again.
- 5) Data Display Servlet - Display the table of invoices to the frontend.

Frontend:

UI Representation of the Data:

Receivables Dashboard Page

SI No	Business Code	Customer Number	Clear Date	Business Year	Document ID	Posting Date	Document Create Date	Due in Date	Invoice Currency	Document Type	Posting ID	Total Open Amount	Baseline Create Date	Custom Terms
1	U001	200769623	2020-02-11	2020	1930438491	2020-01-26	2020-01-25	2020-02-10	USD	RV	1	54273.28	2020-01-26	NAH4
2	U001	200980828	2019-08-08	2019	1929646410	2019-07-22	2019-07-22	2019-08-11	USD	RV	1	79656.6	2019-07-22	NAD1
3	U001	200792734	2019-12-30	2019	1929873765	2019-09-14	2019-09-14	2019-09-29	USD	RV	1	2253.86	2019-09-14	NAA8
4	CA02	140105686	0000-00-00	2020	2960623488	2020-03-30	2020-03-30	2020-04-10	CAD	RV	1	3299.7	2020-03-31	CA10
5	U001	200769623	2019-11-25	2019	1930147974	2019-11-13	2019-11-13	2019-11-28	USD	RV	1	33133.29	2019-11-13	NAH4

Fig 4.3: Receivables Dashboard Page

It consists of 2 sections:

1. Header
2. Grid Panel Section

1. Header Section

The header consists of:

- Account name logo on the left
- The Company Logo in the center.

2. Grid Panel Section

The Grid panel section will be divided into 4 portions:

- The header of the grid will have a Predict button on the top left corner followed by a View Correspondence Button, an Add Button, an Edit Button, a Delete Button and a Search Bar.
- The name of the grid that is Invoice List will be mentioned in the top left corner of the grid.
- The second portion is the table with customer invoice data as rows and the columns.

List of all the columns to be represented on the UI are as follows:

1. Checkbox
2. Customer Name
3. Customer Number (Customer #)
4. Invoice Number (Invoice #)
5. Invoice Amount
6. Due Date
7. Predicted Payment Date
8. Predicted Aging Bucket
9. Notes

Functionalities implemented in React in Detail:

Add Button Functionality - Adds an Invoice to the existing database

Edit Button Functionality- Edits an invoice in the database

Delete Button Functionality - Deletes an invoice in the database

Analytics View - Helps to view chart of details according to delay date.

Predict Button Functionality - Predict the clearing date of an invoice. So after clicking on the predict button it will populate the Predicted Payment Date and Predicted Aging Bucket.

Search Bar Functionality - Search an invoice by the invoice number.

Infinite Scrolling Functionality - As we know that loading 50000 data at same time could cause problems in the UI. To remove that problem we used the concept of Infinite Scrolling where the data would be loaded in batches.

Problems faced while doing the project:

- 1) In the machine learning model it was tough to choose or make the features which would increase the accuracy of the model.
- 2) Implementing Redux in React was tough.
- 3) Faced much difficulty while implementing the various functionalities in the UI .
- 4) Infinite Loading was quite challenging. UI design was also difficult to implement since I am very new to JavaScript. Logical interfacing of the functionalities was the most challenging work.

4.2 Testing OR Verification Plan

After project work is complete, it must have some verification criterion so that we can decide whether the project satisfactorily completed or not. This is called Testing or verification. For example, in software development, some test case must be included and used to verify the outcome of the project. The testing and verification for the ML Model is defined by the accuracy scores of various standards for different regression models and their corresponding values.

The tests used were:

- MSE
- RMSE
- R2

	Algorithm	MSE_Score	R2_Score
0	LinearRegression	2.896802e+11	0.319832
1	SVR	4.279799e+11	-0.004895
2	DecisionTreeRegressor	4.398745e+11	-0.032823
3	RandomForestRegressor	1.485144e+11	0.651289
4	XGB Regressor	1.532659e+11	0.640132

Fig 4.4: Regression Model Scores

In our project, as we are creating an application to find whether bill payment will be delayed or not, the testing and verification of that is resolved by the predicted values of the “aging_bucket” column of the grid. The verification is done in the backend as the model predicts the previously null values and reverts it back to the console to be displayed on the grid. It is then repeated for different values to assure the working and thus the solution to the statement is verified.

4.3 Result Analysis and Screenshots

After the completion of the project, I was able to build the predictive model as well as the B2B application for the use of the company.

For the machine learning model, I was able to acquire an R2 score of 0.69 for XG Boost Regressor. On comparing it with the other regression models, it was found that the XG Boost Regressor performed the best among all of them. Hence the XG Boost Regression model was applied to the validation set and the result which was obtained was found to be 0.69 for the R2 score.

Finally, the features were mapped onto the final test set and were fed to the predictive model. A final data frame with the predicted dates and the aging buckets was created. A snippet of which is shown below.

business_code	cust_number	name_customer	clear_date	buisness_year	doc_id	posting_date	due_in_date	baseline_create_date	cust_paymen
CA02	0140105686	SYSC Ilc	2020-04-20 05:08:42.125000	2020	2.960623e+09	2020-03-30	2020-04-10	2020-03-31	
U001	0200744019	TARG us	2020-04-10 13:45:49.750000	2020	1.930659e+09	2020-03-19	2020-04-03	2020-03-19	
U001	0200418007	AM	2020-03-27 01:21:47.375000	2020	1.930611e+09	2020-03-11	2020-03-26	2020-03-11	
U001	0200739534	OK systems	2020-04-27 02:16:44.453125	2020	1.930788e+09	2020-04-15	2020-04-30	2020-04-15	
U001	0200353024	DECA corporation	2020-04-23 14:12:27.671875	2020	1.930817e+09	2020-04-23	2020-04-26	2020-04-16	

ame_customer	clear_date	buisness_year	doc_id	posting_date	due_in_date	baseline_create_date	cust_payment_terms	converted_usd	Aging Bucket
SYSC Ilc	2020-04-20 05:08:42.125000	2020	2.960623e+09	2020-03-30	2020-04-10	2020-03-31	CA10	2309.79	0-15
TARG us	2020-04-10 13:45:49.750000	2020	1.930659e+09	2020-03-19	2020-04-03	2020-03-19	NAA8	11173.02	0-15
AM	2020-03-27 01:21:47.375000	2020	1.930611e+09	2020-03-11	2020-03-26	2020-03-11	NAA8	3525.59	0-15
OK systems	2020-04-27 02:16:44.453125	2020	1.930788e+09	2020-04-15	2020-04-30	2020-04-15	NAA8	121105.65	NaN
DECA corporation	2020-04-23 14:12:27.671875	2020	1.930817e+09	2020-04-23	2020-04-26	2020-04-16	NAM2	3726.06	NaN

Fig 4.5: Final Dataframe

Chapter 5

Standards Adopted

5.1 Design/ Standards

In all the engineering streams, there are predefined design standards are present such as IEEE, ISO etc.

- Design and development stages
- Design and development inputs
- Design and development controls
- Design and development outputs
- Design and development changes

5.2 Coding Standards

Coding standards are collections of coding rules, guidelines, and best practices. Few of the coding standards are:

1. Write as few lines as possible.
2. Use appropriate naming conventions.
3. Segment blocks of code in the same section into paragraphs.
4. Use indentation to marks the beginning and end of control structures. Clearly specify the code between them.
5. Don't use lengthy functions. Ideally, a single function should carry out a single task.

5.3 Testing/Verification Standards

There are some ISO and IEEE standards for quality assurance and testing of the product.

- Design complies with and is traceable to inputs
- Design complies with client requirements
- Design complies with standards, guides and codes
- Design complies with local and statutory requirements
- Constructability and maintainability review has been completed
- Design is economical and value-for-money

Chapter 6

Conclusion & Future Scope

6.1 Conclusion

An invoice management process starts when you receive the supplier invoice. The invoice should then be: identified, categorized, filed and matched to a purchase order (PO) by the person responsible for the order. In this project, I have built an AI-Enabled FinTech B2B Order Management Application using a regression model to predict whether payment of the passed function will be delayed. The accuracy of the model turned out to be 90%. The same has been deployed using Flask Framework and React JS for the frontend

This Project dealt with the design and development process of context-adaptive Web-application. During the whole process, I was able to learn and implement many new technologies. I also got the knowledge of how Business to Business work operates. The project helped me to build a complete understanding of a working full-stack application. Also by integrating a machine learning model to my project ,I was able to enhance the capabilities of my project and make it as industrial as possible.

6.2 Future Scope

The accuracy of the model can be increased and the same can be deployed in the cloud to make it available to a larger audience
AI-Enabled

Organizations, firms, and even individuals may keep track of the payment of their orders with the aid of AI Enabled Fintech B2B INVOICE MANAGEMENT Application, regardless of the big number of clients. This will not only save time, but will also provide a functional environment for an organization's and/or workplace's proper functioning.

We can work on making the application more user-friendly and try to build new features for the machine learning model to improve the accuracy. We can also work on making the site responsive for every device or resolution.

References

- [1] <https://www.geeksforgeeks.org/data-preprocessing-machine-learning-python/>
- [2] <https://reactjs.org/>
- [3] <https://www.javatpoint.com/servlet-tutorial>
- [4] [How to Choose a Feature Selection Method For Machine Learning \(machinelearningmastery.com\)](#)
- [5] [Java Database Connectivity\(JDBC\) \(knowledgehut.com\)](#)

PLAGIARISM REPORT



PLAGIARISM SCAN REPORT

