A project report on

Invoice management app, AI enabled, bucket level

Submitted in partial fulfillment of the requirements for the Degree of

B. Tech in COMPUTER SCIENCE AND ENGINEERING

by

UMANG GUPTA (1805084)

under the guidance of

Name of Project Supervisor
Pratyush Sunanda High Radius Technologies
School of Computer Engineering
Kalinga Institute of Industrial Technology
Deemed to be University
Bhubaneswar

January - March 2021

KALINGA INSTITUTE OF INDUSTRIAL TECHNOLOGY

Deemed to be University U/S 3 of the UGC Act, 1956

CERTIFICATE

This is to certify that the project report entitled " **Invoice management app, AI enabled, bucket level"** submitted by

UMANG GUPTA 1805084

in partial fulfillment of the requirements for the award of the **Degree of Bachelor of Technology** in **School of Computer Engineering(CSE)** is a bonafide record of the work carried out under my(our) guidance and supervision at School of Computer Engineering, Kalinga Institute of Industrial Technology, Deemed to be University.

Sangeeta padhy	Upmanyu sarangi
SANGEETA PADHY	UPAMANYU SARANGI
High Radius Technologies	High Radius Technologies

....

The Project was evaluated by us on 31/03/2021

Vipul Roy	Rohit Halder
Subhashree Dash	Subham Mohapatra

ACKNOWLEDGEMENTS

I Umang Gupta , Currently in 6th Semester , 3rd Year , Computer Science and Engineering Have Been Given Work On Project : Invoice management app, AI enabled, bucket level. I Am Very Thankful To High Radius Moderators , High Radius Team , In Machine Learning Phase : Vipul Roy , Rohit Haldar and Sangeeta Padhy Mam , In React+Java Phase : Upmanyu Sarangi , Subham Mohapatra also thankful to Pratyush Sunandan For his great effort and guidance , I Am Very Thankful for the efforts Made by Our Group Coordinator Of High Radius Team Of The High Radius , FIC(Project) Of Kiit University, Program Head Of Computer Science and Engineering and Dean of the School Of Computer Science Engineering Prof. Bhabani Shankar Mishra for all of their sheer determination.

The implementation of the project would have been achievable without the team member's commitment, hard work and never-ending effort.

UMANG GUPTA (Umang Gupta)

ABSTRACT

An Invoice Management Application Which Is An AI Enabled , Bucket Level Which FrontEnd and BackEnd Concept Using The Technology : Machine Learning , React JS and Language : JavaScript , Java. This Invoice Management Application is any Website Which Perform Certain Tasks Such As Tracking The Sales ,Tracking The Orders, inventory, and fulfillment , Calculating The Predicted Payment Date Of Invoice Payment Using Machine Learning Algorithms And For Storing The Data Containing The 50000 Rows And 1 column Using The Concept Of JDBC Connection Concept. And In The FrondEnd Design Concept Using The Java , JavaScript , JSX , REACT Concepts For Give give Better Design Using The Different Colors , Make The Application More Responsiveness Which enables the people To processes and partnerships necessary for products to find their way to the customers who bought them.

The Main Aim Of The Project Should Be To Predict The Invoice Payment Of The B2B. And Also To Build an AI-Enabled FinTech B2B Invoice Management Application , Build a Machine Learning model to predict the date of payment of an invoice.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

Introduction

In an Invoice Mangement App the B2B world operates differently from the B2C or C2C world. Businesses work with other businesses on credit. When a buyer business orders goods from the seller business, the seller business issues an invoice for the same. This invoice for the goods contains various information like the details of the goods purchased and when it should be paid.An Invoice Management System is a computer software program that allows businesses to manage their orders and inventory. Order management systems help ensure more accurate inventory management, automatically entering new inventory into the system, tracking sales through different selling platforms, such as eBay and Amazon, and alerting you, the business owner, when your stock of a particular item drops low enough for a re-order

The objective of the first half of the winter internship project was:

Problem Statement for ML Model:

The objective of the first half of the winter internship project is:

To build a Machine Learning Model to predict the payment date of an invoice when it gets created in the system. Categorize the invoice into different buckets based on predicted payment date.

For example: Let's suppose Walmart wishes to add 20,000 Nike shoes to its sports department. Now, Walmart becomes the buyer business whereas Nike is the seller business. Since this is B2B, the business is done on credit. After successful delivery of the shoes, Nike establishes a Payment term within which Walmart must pay it's total open amount, let's suppose $1.2M. If Walmart is unable to pay within the decided payment term, it becomes the duty of the Account receivables department of Nike to follow up and get Walmart to pay the outstanding amount.

I will be receiving an invoices dataset that contains the past payment information and behaviour of various buyers. Based on the previous payment patterns, the ML model needs to predict what will be the date a payment is made by the customer for an invoice.

The model also needs to predict which aging bucket the invoice falls into based on the predicted payment date.

The objective of the second half of the winter internship project was:

To build a full stack Invoice Management Application using ReactJs, JDBC, Java and JSP.Build a responsive Receivables Dashboard. Visualize Data in the form of grids. Perform Searching operations on the invoices. Edit data in the editable fields of the grid. Download data of selected rows in predefined templates.

Chapter-2

Background

MACHINE LEARNING and REACT - JS

Machine Learning is the field of study that gives computers the capability to learn without being explicitly programmed. ML is one of the most exciting technologies that one would have ever come across. As it is evident from the name, it gives the computer that makes it more similar to humans: The ability to learn. Machine learning is actively being used today, perhaps in many more places than one would expect.

Types of machine learning problems

There are various ways to classify machine learning problems. Here, we discuss the most obvious ones.

1. On basis of the nature of the learning "signal" or "feedback" available to a learning system

- **Supervised learning**: The computer is presented with example inputs and their desired outputs, given by a "teacher", and the goal is to learn a general rule that maps inputs to outputs. The training process continues until the model achieves the desired level of accuracy on the training data. Some real-life examples are:
- **Image Classification:** You train with images/labels. Then in the future you give a new image expecting that the computer will recognize the new object.
- **Market Prediction/Regression:** You train the computer with historical market data and ask the computer to predict the new price in the future.
- **Unsupervised learning**: No labels are given to the learning algorithm, leaving it on its own to find structure in its input. It is used for clustering population in different groups. Unsupervised learning can be a goal in itself (discovering hidden patterns in data).
- **Clustering:** You ask the computer to separate similar data into clusters, this is essential in research and science.
- **High Dimension Visualization:** Use the computer to help us visualize high dimension data.

- **Generative Models:** After a model captures the probability distribution of your input data, it will be able to generate more data. This can be very useful to make your classifier more robust

3.1 Invoice Management Application

Most order management software programs follow a 6-step system, relying on automation to help employees accurately process and fulfill customer orders. When used properly, order management software produces a seamless flow from entering invoices through after sales follow-up. The smoother the software runs, the faster you can fulfill orders, and the less likely your customer receives the wrong item or experiences the frustration of backorders.

1. The order is placed

Your customer places an order through your third-party sales site, your own website, or over the phone with a live representative. Online, your customers will enter their details on a standardized form, with an option to have a securely saved preferred payment method.

To improve the sales process, make all fields of your online form mandatory so that you have all the necessary contact information for the customer up-front. This creates a customer profile and allows your order management system to track their purchase history, the volume of orders, and payment and delivery preferences. It also gives you their phone number and email address, should you need to follow up with service recovery.

The payment is processed, and then the order is sent to the warehouse once your software system approves the charges.

2. Warehouse processing

Once the order arrives at the warehouse, it's checked by the intake team and the item or items are "picked" from the stock. Having a SKU and barcode for every item increases the accuracy of fulfillment and makes it easier for pickers to simply scan the item and add it to the order.

If there isn't enough of the item(s) in stock to fulfill an order, then a purchase order is automatically placed through the order management software. You and the warehouse manager will receive an alert that there may be a delay in fulfillment. Your customer may receive an automatic notification of the delay, and the customer service team can follow up with your customer.

3. Reconciling the order

Next, the order is sent to the accounting department or preferably it should sync automatically with your cloud accounting software, where it's recorded in your A/R ledger. The sale is logged and a receipt sent to your client. Automating your sales ledgers makes it easier for auditing, inventory reconciliation, and end-of-year taxes.

4. Shipping the order

Once the order is picked from the warehouse, your packing team will double-check for accuracy, again using the barcodes and SKU. Then, the order is packed carefully and shipped via a third-party delivery system. Your customer will receive a notice through the order management system that their order has shipped, along with a tracking number and estimated delivery time. 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.

5. Post-sales follow-up

Once the order arrives, the software should generate an automatic email to follow up, asking how they liked the items and ensuring that they received everything accurately. This email should include detailed instructions on how to reach customer service if there are any issues, taking the frustration out of guessing how to obtain a refund if needed.

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

6. Special order oversight

Another aspect of good OMS is 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. When these orders are placed through the system, the software can flag them with a code, allowing you or your customer retention team to personally monitor the order for accuracy.

3.2 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 parametersor 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.

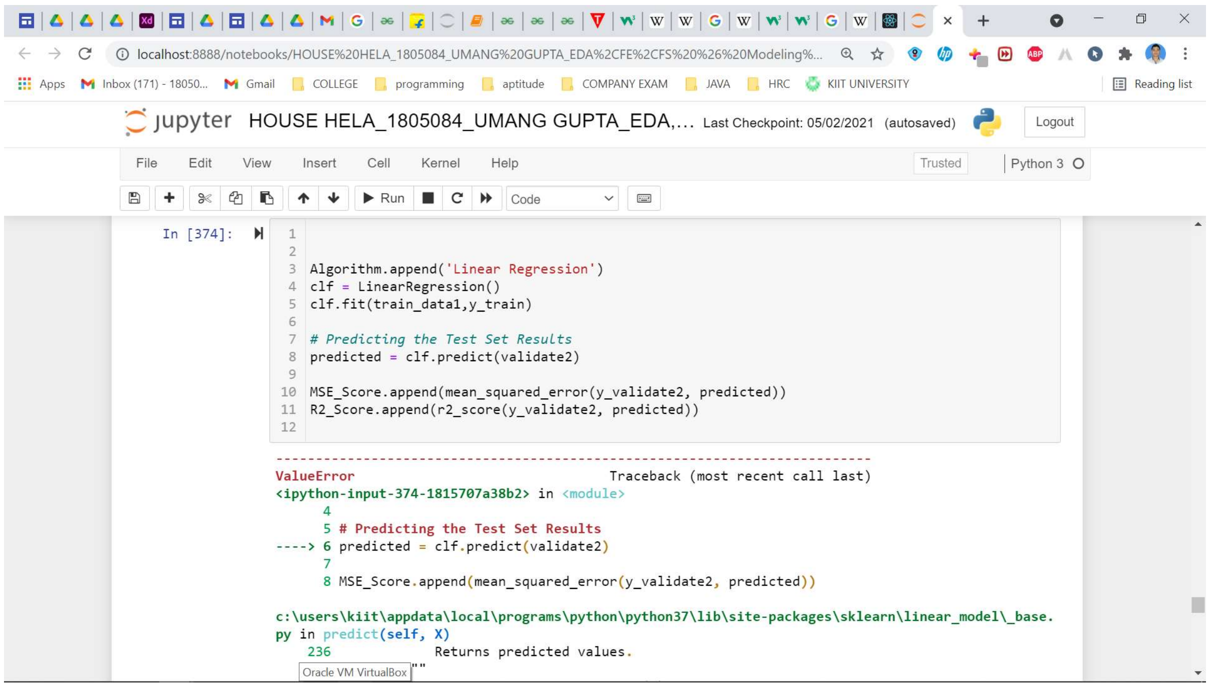
3.2.1 Linear Regression

In statistics, linear regression is a linear approach to modelling the relationship between a scalar response and one or more explanatory variables (also known as dependent and independent variables). The case of one explanatory variable is called *simple linear regression*; for more than one, the process is called multiple linear regression. This term is distinct from multivariate linear regression, where multiple correlated dependent variables are predicted, rather than a single scalar variable.

In linear regression, the relationships are modeled using linear predictor functions whose unknown model parameters are estimated from the data. Such models are called linear models. Most commonly, the conditional mean of the response given the values of the explanatory variables (or predictors) is assumed to be an affine function of those values; less commonly, the conditional median or some other quantile is used. Like all forms of regression analysis, linear regression focuses on the conditional probability distribution of the response given the values of the predictors, rather than on the joint probability distribution of all of these variables, which is the domain of multivariate analysis.

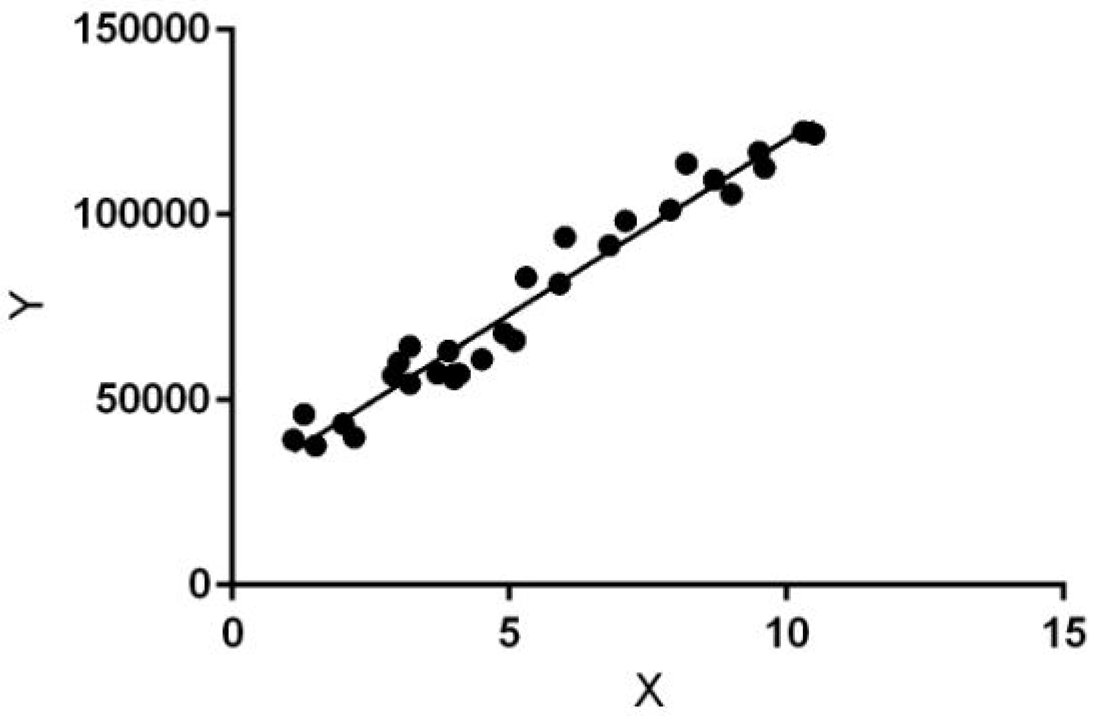
Linear regression was the first type of regression analysis to be studied rigorously, and to be used extensively in practical applications. This is because models which depend linearly on their unknown parameters are easier to fit than models which are non-linearly related to their parameters and because the statistical properties of the resulting estimators are easier to determine.

Linear Regression is a machine learning algorithm based on **supervised learning**. It performs a **regression task**. Regression models a target prediction value based on independent variables. It is mostly used for finding out the relationship between variables and forecasting. Different regression models differ based on – the kind of relationship between dependent and independent variables, they are considering and the number of independent variables being used.



```python


Algorithm.append('Linear Regression')
clf = LinearRegression()
clf.fit(train_data1,y_train)

# Predicting the Test Set Results
predicted = clf.predict(validate2)

MSE_Score.append(mean_squared_error(y_validate2, predicted))
R2_Score.append(r2_score(y_validate2, predicted))

```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-374-1815707a38b2> in <module>
      4 
      5 # Predicting the Test Set Results
----> 6 predicted = clf.predict(validate2)
      7 
      8 MSE_Score.append(mean_squared_error(y_validate2, predicted))

c:\users\kiit\appdata\local\programs\python\python37\lib\site-packages\sklearn\linear_model\_base.
py in predict(self, X)
    236             Returns predicted values.
                    """
```



Linear regression performs the task to predict a dependent variable value (y) based on a given independent variable (x). So, this regression technique finds out a linear relationship between x (input) and y(output). Hence, the name is Linear Regression. In the figure above, X (input) is the work experience and Y (output) is the salary of a person. The regression line is the best fit line for our model.

Hypothesis function for Linear Regression :

$$y = \theta_1 + \theta_2 . x$$

While training the model we are given :
x: input training data (univariate – one input variable(parameter))
y: labels to data (supervised learning)

When training the model – it fits the best line to predict the value of y for a given value of x. The model gets the best regression fit line by finding the best θ1 and θ2 values.
θ1: intercept
θ2: coefficient of x

Once we find the best θ1 and θ2 values, we get the best fit line. So when we are finally using our model for prediction, it will predict the value of y for the input value of x.

How to update θ1 and θ2 values to get the best fit line ?

Cost Function (J):
By achieving the best-fit regression line, the model aims to predict y value such that the error difference between predicted value and true value is minimum. So, it is very important to update the θ1 and θ2 values, to reach the best value that minimize the error between predicted y value (pred) and true y value (y).

$$minimize \frac{1}{n} \sum_{i=1}^{n} (pred_i - y_i)^2$$

$$J = \frac{1}{n} \sum_{i=1}^{n} (pred_i - y_i)^2$$

Cost function(J) of Linear Regression is the **Root Mean Squared Error (RMSE)** between predicted y value (pred) and true y value (y).

Gradient Descent**:**
To update θ1 and θ2 values in order to reduce Cost function (minimizing RMSE value) and achieving the best fit line the model uses Gradient Descent. The idea is to start with random θ1 and θ2 values and then iteratively updating the values, reaching minimum cost.

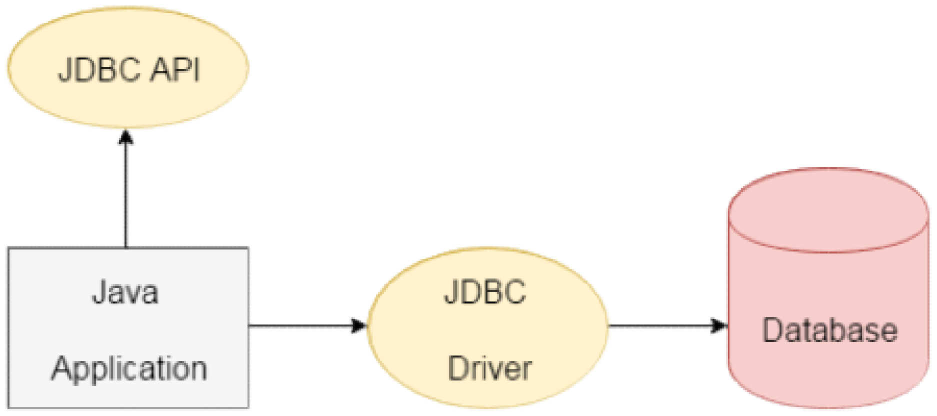
JAVA JDBC

JDBC stands for Java Database Connectivity. JDBC is a Java API to connect and execute the query with the database. It is a part of JavaSE (Java Standard Edition). 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, and
- Thin Driver

We have discussed the above four drivers in the next chapter.

We can use JDBC API to access tabular data stored in any relational database. By the help of JDBC API, we can save, update, delete and fetch data from the database. It is like Open Database Connectivity (ODBC) provided by Microsoft.



The current version of JDBC is 4.3. It is the stable release since 21st September, 2017. It is based on the X/Open SQL Call Level Interface. The **java.sql** package contains classes and interfaces for JDBC API. A list of popular interfaces of JDBC API are given below:

- Driver interface
- Connection interface
- Statement interface
- PreparedStatement interface
- CallableStatement interface
- ResultSet interface
- ResultSetMetaData interface
- DatabaseMetaData interface
- RowSet interface

A list of popular classes of JDBC API are given below:

- DriverManager class
- Blob class
- Clob class
- Types class

Why Should We Use JDBC

Before JDBC, ODBC API was the database API to connect and execute the query with the database. But, ODBC API uses ODBC driver which is written in C language (i.e. platform dependent and unsecured). That is why Java has defined its own API (JDBC API) that uses JDBC drivers (written in Java language).

We can use JDBC API to handle database using Java program and can perform the following activities:

1. Connect to the database
2. Execute queries and update statements to the database
3. Retrieve the result received from the database.

REACT JS

React is a declarative, efficient, and flexible JavaScript library for building user interfaces. It's 'V' in MVC. ReactJS is an open-source, component-based front-end library responsible only for the view layer of the application. It is maintained by Facebook.

React uses a declarative paradigm that makes it easier to reason about your application and aims to be both efficient and flexible. It designs simple views for each state in your application, and React will efficiently update and render just the right component when your data changes. The declarative view makes your code more predictable and easier to debug.

Reasons to learn React: React is a declarative, efficient, and flexible JavaScript library for building user interfaces. React.js is an open-source, component-based front-end library responsible only for the view layer of the application. It is an MVC architecture-based library that plays the role of "C" which means control.

React uses a declarative paradigm that makes it easier to reason about your application and aims to be both efficient and flexible. It designs simple views for each state in your application, and React will efficiently update and render just the right component when your data changes. The declarative view makes your code more predictable and easier to debug.

Features of React.js: There are unique features are available on React because that it is widely popular.

- **Use JSX:** It is faster than normal JavaScript as it performs optimizations while translating to regular JavaScript. It makes it easier for us to create templates.
- **Virtual DOM:** Virtual DOM exists which is like a lightweight copy of the actual DOM. So for every object that exists in the original DOM, there is an object for that in React Virtual DOM. It is exactly the same, but it does not have the power to directly change the layout of the document. Manipulating DOM is slow, but manipulating Virtual DOM is fast as nothing gets drawn on the screen.
- **One-way Data Binding:** This feature gives you better control over your application.
- **Component:** A Component is one of the core building blocks of React. In other words, we can say that every application you will develop in React will be made up of pieces called components. Components make the task of building UIs much easier. You can see a UI broken down into multiple individual pieces called components and work on them independently and merge them all in a parent component which will be your final UI.
- **Performance:** React.js use JSX, which is faster compare to normal JavaScript and HTML. Virtual DOM is a less time taking procedure to update webpages content.

HTML(HYPER TEXT MARKUP LANGUAGE)

The **HyperText Markup Language**, or **HTML** is the standard markup language for documents designed to be displayed in a web browser. It can be assisted by technologies such as Cascading Style Sheets (CSS) and scripting languages such as JavaScript.

Web browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document.

HTML elements are the building blocks of HTML pages. With HTML constructs, images and other objects such as interactive forms may be embedded into the rendered page. HTML provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes and other items. HTML elements are delineated by *tags*, written using angle brackets. Tags such as <**img** /> and <**input** /> directly introduce content into the page. Other tags such as <**p**> surround and provide information about document text and may include other tags as sub-elements. Browsers do not display the HTML tags, but use them to interpret the content of the page.

HTML can embed programs written in a scripting language such as JavaScript, which affects the behavior and content of web pages. Inclusion of CSS defines the look and layout of content. The World Wide Web Consortium (W3C), former maintainer of the

HTML and current maintainer of the CSS standards, has encouraged the use of CSS over explicit presentational HTML since 1997.

```
<html>
<head>
<title>Page Title</title>
</head>
<body>

<h1>This is a Heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

CSS(CASCADING STYLE SHEETS)

Cascading Style Sheets (**CSS**) is a style sheet language used for describing the presentation of a document written in a markup language such as HTML.[1] CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript.[2]

CSS is designed to enable the separation of presentation and content, including layout, colors, and fonts.[3] This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, enable multiple web pages to share formatting by specifying the relevant CSS in a separate .css file which reduces complexity and repetition in the structural content as well as enabling the .css file to be cached to improve the page load speed between the pages that share the file and its formatting.

Separation of formatting and content also makes it feasible to present the same markup page in different styles for different rendering methods, such as on-screen, in print, by voice (via speech-based browser or screen reader), and on Braille-based tactile devices. CSS also has rules for alternate formatting if the content is accessed on a mobile device.[4]

The name *cascading* comes from the specified priority scheme to determine which style rule applies if more than one rule matches a particular element. This cascading priority scheme is predictable.

The CSS specifications are maintained by the World Wide Web Consortium (W3C). Internet media type (MIME type) text/css is registered for use with CSS by RFC 2318 (March 1998). The W3C operates a free CSS validation service for CSS documents.

JAVASCRIPT(JS)

JavaScript often abbreviated as **JS**, is a programming language that conforms to the ECMAScript specification.[9] JavaScript is high-level, often just-in-time compiled, and multi-paradigm. It has curly-bracket syntax, dynamic typing, prototype-based object-orientation, and first-class functions.

Alongside HTML and CSS, JavaScript is one of the core technologies of the World Wide Web.[10] Over 97% of websites use it client-side for web page behavior,[11] often incorporating third-party libraries.[12] All major web browsers have a dedicated JavaScript engine to execute the code on the user's device.

As a multi-paradigm language, JavaScript supports event-driven, functional, and imperative programming styles. It has application programming interfaces (APIs) for working with text, dates, regular expressions, standard data structures, and the Document Object Model (DOM).

The ECMAScript standard does not include any input/output (I/O), such as networking, storage, or graphics facilities. In practice, the web browser or other runtime system provides JavaScript APIs for I/O.

JavaScript engines were originally used only in web browsers, but they are now core components of other software systems, most notably servers and a variety of applications.

Although there are similarities between JavaScript and Java, including language name, syntax, and respective standard libraries, the two languages are distinct and differ greatly in design.

```
<html>

<body>

<h2>My First JavaScript</h2>

<button type="button"

onclick="document.getElementById('demo').innerHTML = Date()">

Click me to display Date and Time.</button>

<p id="demo"></p>

</body>

</html>
```

JSX(JAVASCRIPT LIBRARY)

This funny tag syntax is neither a string nor HTML.

It is called JSX, and it is a syntax extension to JavaScript. We recommend using it with React to describe what the UI should look like. JSX may remind you of a template language, but it comes with the full power of JavaScript.

JSX produces React "elements". We will explore rendering them to the DOM in the next section. Below, you can find the basics of JSX necessary to get you started.

React embraces the fact that rendering logic is inherently coupled with other UI logic: how events are handled, how the state changes over time, and how the data is prepared for display.

Instead of artificially separating technologies by putting markup and logic in separate files, React separates concerns with loosely coupled units called "components" that contain both. We will come back to components in a further section, but if you're not yet comfortable putting markup in JS, this talk might convince you otherwise.

React doesn't require using JSX, but most people find it helpful as a visual aid when working with UI inside the JavaScript code. It also allows React to show more useful error and warning messages.

```
const name = 'Josh Perez';const element = <h1>Hello, {name}</h1>;
             .render(
          ,
             .getElementById('root'));
```

Chapter-3

Project Implementation

In this given project we were required to make a full stack Invoice/Bill Management application and also to build a Machine Learning Model to predict the date of payment and aging bucket of an invoice.

Technologies Used :

1) Machine Learning using Python
2) Java,SQL(Backend)
3) HTML,CSS,Javascript,React(Frontend)
4) Flask

Objective of Machine Learning :

1) To build a Machine Learning Model to predict the payment date of an invoice when it gets created in the system.
2) Categorize the invoice into different buckets based on predicted payment date.

About Dataset:

1) business_code - company code of the account
2) cust_number - customer number given to all the customers of the Account
3) name_customer - name of the customer.
4) cust_number - Each customer has a number that uniquely identifies it.
5) business_code - company code of the account
6) cust_number - customer number given to all the customers of the Account
7) name_customer - name of the customer.
8) cust_number - Each customer has a number that uniquely identifies it.
9) document_create_date - The date on which the invoice document was created

10) document_create_date_norm - Normalised date of the invoice document

11) posting_id - key indicator to identify whether an AR item is an invoice.

12) due_in_date - The date on which the customer is expected to clear an invoice

13) invoice_id - Unique number assigned when a seller creates an Invoice.

14) baseline_create_date - The date on which the Invoice was created.

15) total_open_amount - The amount that is yet to be paid for that invoice

16) invoice_amount - The total amount for that invoice.

17) cust_payment_terms -Business terms and agreements be

18) invoice_currency - The currency of the invoice amount in the document for the invoice.

19) doc_id - It is also an unique identifier of an invoice and is a primary key.

20) total_open_amount - open amount of an invoice tween customers and accounts on discounts and days of payment

21) area_business - Business area in sap is defined as an organisational area within the financial accounting module.

22) clear_date - The date on which the customer clears an invoice, or in simple terms, they make the full payment.

23) is_open_invoice - indicator of whether an invoice is open or closed.

ML MODEL:

While building this model we had gone through many process:

Step 1. Firstly we needed 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 to training purposes.

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

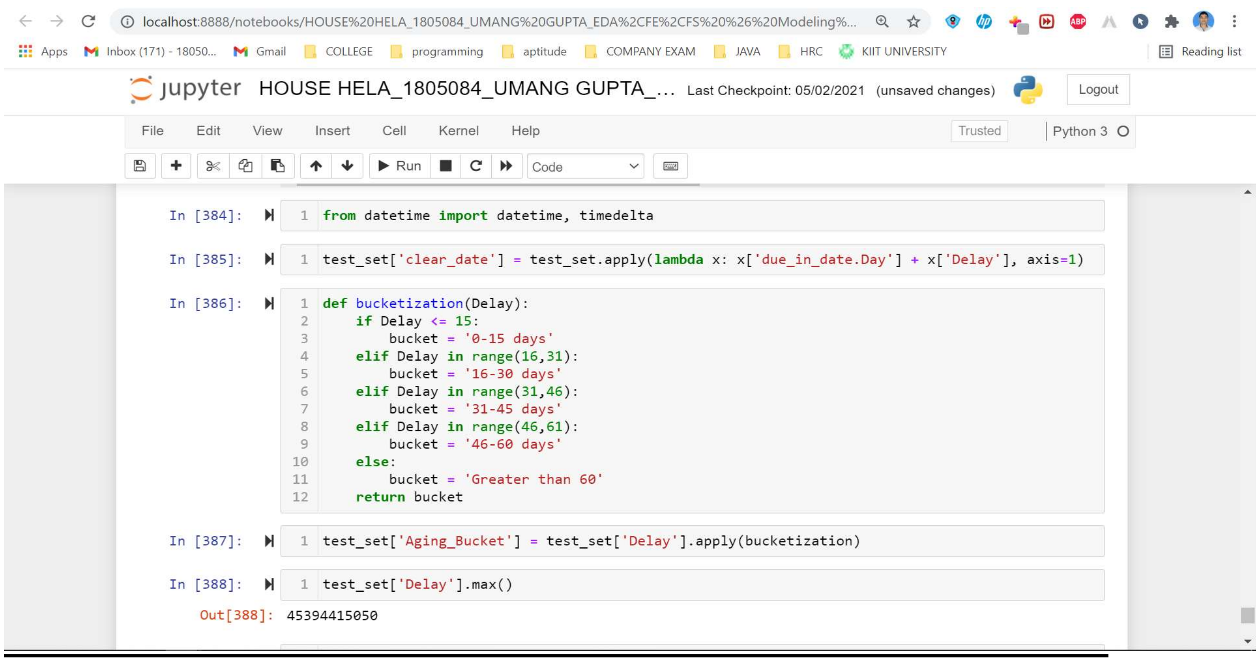
Step 8. Feature Scaling for improved training using Normalization and Standardisation.

Step 9. Training and Validating ML model.

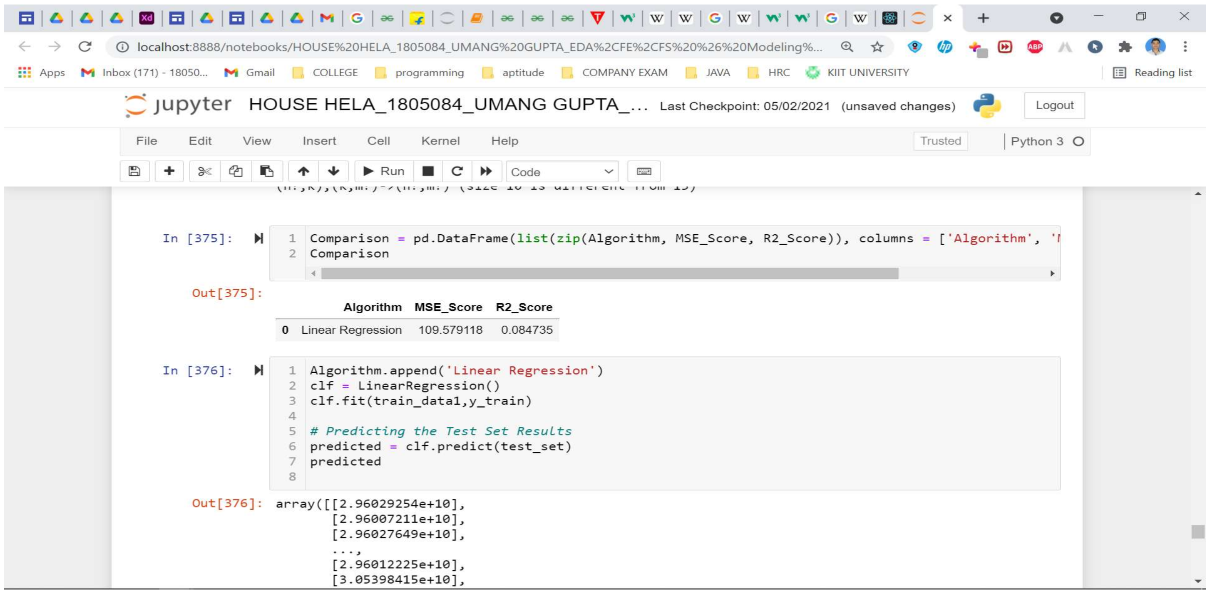
Step 10. Predict clear_date using test data.

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



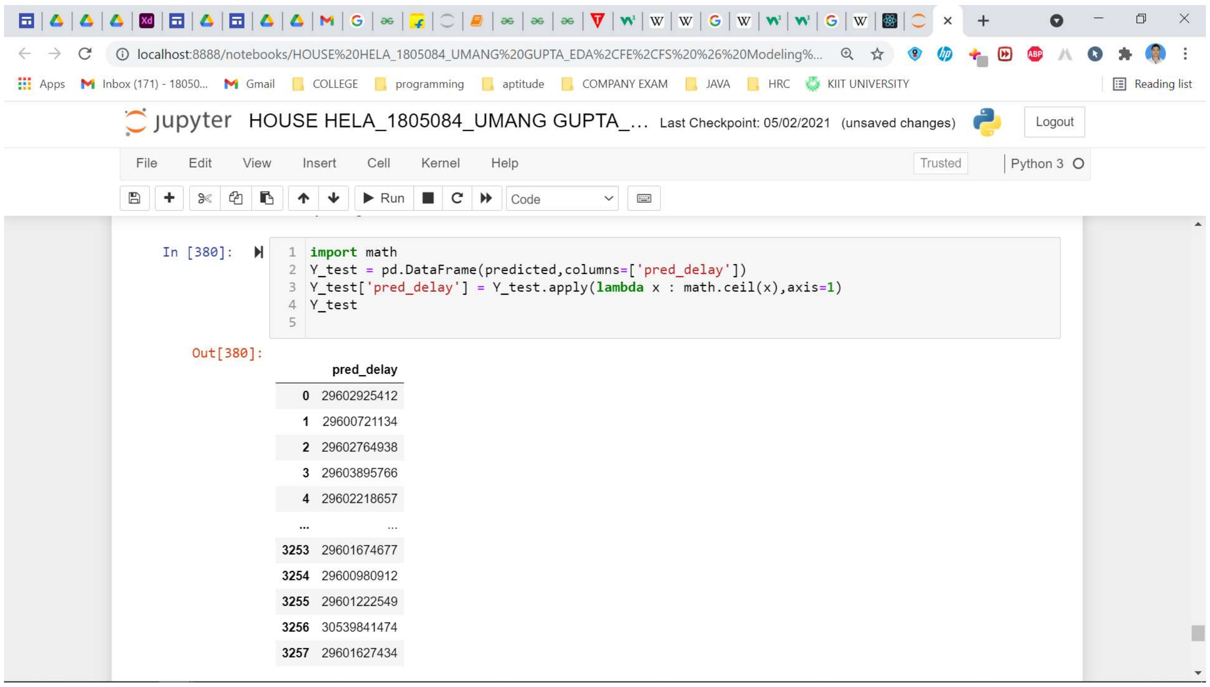
```
In [384]: from datetime import datetime, timedelta

In [385]: test_set['clear_date'] = test_set.apply(lambda x: x['due_in_date.Day'] + x['Delay'], axis=1)

In [386]: def bucketization(Delay):
              if Delay <= 15:
                  bucket = '0-15 days'
              elif Delay in range(16,31):
                  bucket = '16-30 days'
              elif Delay in range(31,46):
                  bucket = '31-45 days'
              elif Delay in range(46,61):
                  bucket = '46-60 days'
              else:
                  bucket = 'Greater than 60'
              return bucket

In [387]: test_set['Aging_Bucket'] = test_set['Delay'].apply(bucketization)

In [388]: test_set['Delay'].max()

Out[388]: 45394415050
```



```
(n?,k),(k,m?)->(n?,m?) (size 16 is different from 15)

In [375]: Comparison = pd.DataFrame(list(zip(Algorithm, MSE_Score, R2_Score)), columns = ['Algorithm', 'I
          Comparison

Out[375]:
```

	Algorithm	MSE_Score	R2_Score
0	Linear Regression	109.579118	0.084735

```
In [376]: Algorithm.append('Linear Regression')
          clf = LinearRegression()
          clf.fit(train_data1,y_train)

          # Predicting the Test Set Results
          predicted = clf.predict(test_set)
          predicted


Out[376]: array([[2.96029254e+10],
                 [2.96007211e+10],
                 [2.96027649e+10],
                 ...,
                 [2.96012225e+10],
                 [3.05398415e+10],
```



```
In [380]:
import math
Y_test = pd.DataFrame(predicted,columns=['pred_delay'])
Y_test['pred_delay'] = Y_test.apply(lambda x : math.ceil(x),axis=1)
Y_test

Out[380]:
      pred_delay
   0  29602925412
   1  29600721134
   2  29602764938
   3  29603895766
   4  29602218657
 ...          ...
3253  29601674677
3254  29600980912
3255  29601222549
3256  30539841474
3257  29601627434
```

Objective of Java and React :

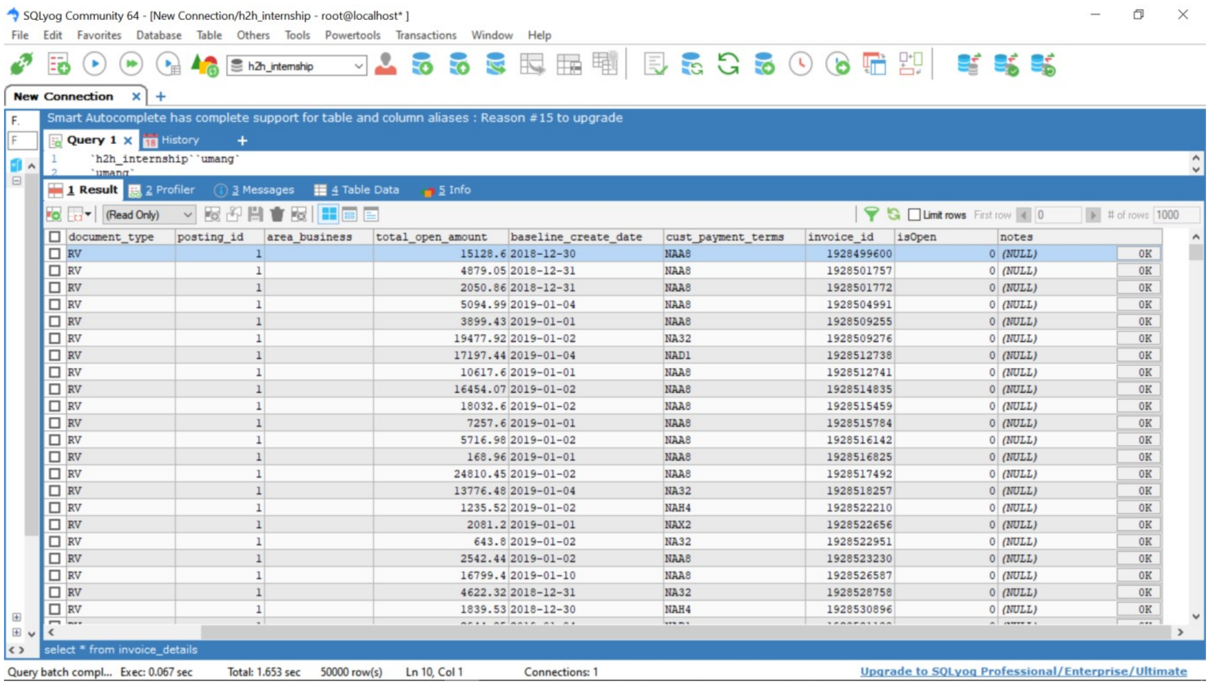
Backend Design:

Data Loading in the Database:

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


SQLyog Community 64 - [New Connection/h2h_internship - root@localhost*]

Smart Autocomplete has complete support for table and column aliases : Reason #15 to upgrade

```
`h2h_internship``umang`
`umang`
```

document_type	posting_id	area_business	total_open_amount	baseline_create_date	cust_payment_terms	invoice_id	isOpen	notes	
RV	1		15128.6	2018-12-30	NAA8	1928499600	0	(NULL)	OK
RV	1		4879.05	2018-12-31	NAA8	1928501757	0	(NULL)	OK
RV	1		2050.86	2018-12-31	NAA8	1928501772	0	(NULL)	OK
RV	1		5094.99	2019-01-04	NAA8	1928504991	0	(NULL)	OK
RV	1		3899.43	2019-01-01	NAA8	1928509255	0	(NULL)	OK
RV	1		19477.92	2019-01-02	NA32	1928509276	0	(NULL)	OK
RV	1		17197.44	2019-01-04	NAD1	1928512738	0	(NULL)	OK
RV	1		10617.6	2019-01-01	NAA8	1928512741	0	(NULL)	OK
RV	1		16454.07	2019-01-02	NAA8	1928514835	0	(NULL)	OK
RV	1		18032.6	2019-01-02	NAA8	1928515459	0	(NULL)	OK
RV	1		7257.6	2019-01-01	NAA8	1928515784	0	(NULL)	OK
RV	1		5716.98	2019-01-02	NAA8	1928516142	0	(NULL)	OK
RV	1		168.96	2019-01-01	NAA8	1928516825	0	(NULL)	OK
RV	1		24810.45	2019-01-02	NAA8	1928517492	0	(NULL)	OK
RV	1		13776.48	2019-01-04	NA32	1928518257	0	(NULL)	OK
RV	1		1235.52	2019-01-02	NAH4	1928522210	0	(NULL)	OK
RV	1		2081.2	2019-01-01	NAX2	1928522656	0	(NULL)	OK
RV	1		643.8	2019-01-02	NA32	1928522951	0	(NULL)	OK
RV	1		2542.44	2019-01-02	NAA8	1928523230	0	(NULL)	OK
RV	1		16799.4	2019-01-10	NAA8	1928526587	0	(NULL)	OK
RV	1		4622.32	2018-12-31	NA32	1928528758	0	(NULL)	OK
RV	1		1839.53	2018-12-30	NAH4	1928530896	0	(NULL)	OK

```
select * from invoice_details
```

Query batch compl... Exec: 0.067 sec Total: 1.653 sec 50000 row(s) Ln 10, Col 1 Connections: 1

Upgrade to SQLyog Professional/Enterprise/Ultimate

Servlet Creation:

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

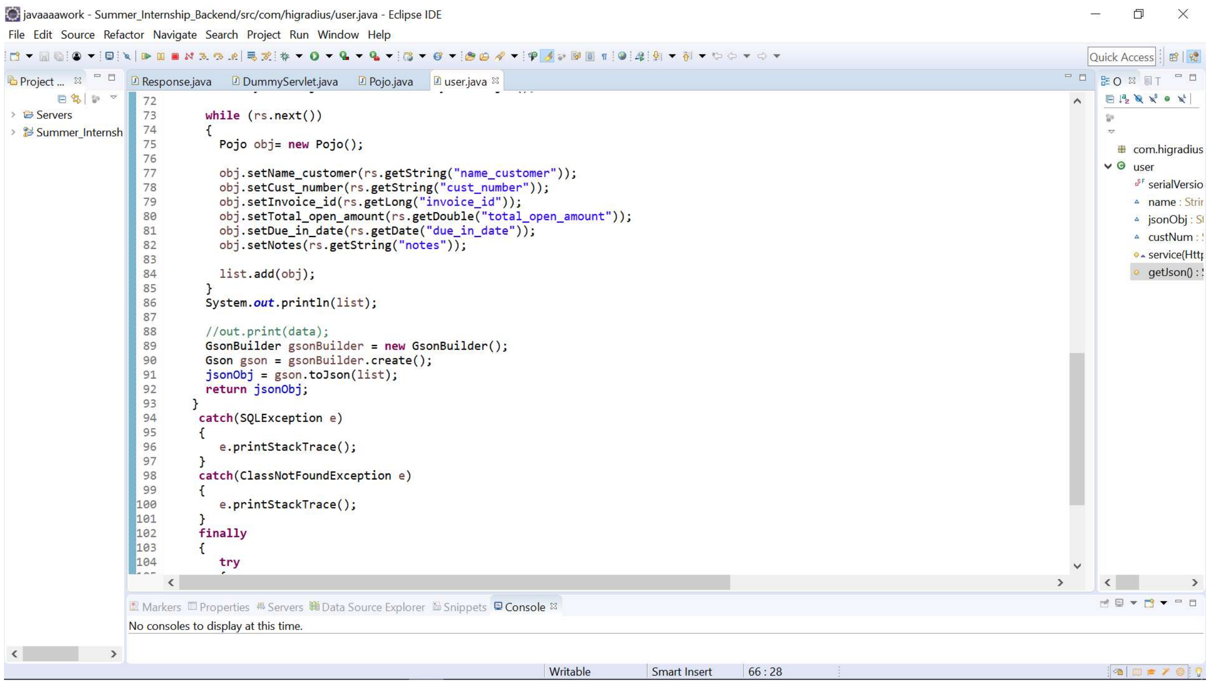
1) Add servlet - Get a POST request from the frontend with parameters such as invoice amount ,notes,date , etc and pass them to the SQL database.

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 by passing 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 a 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.

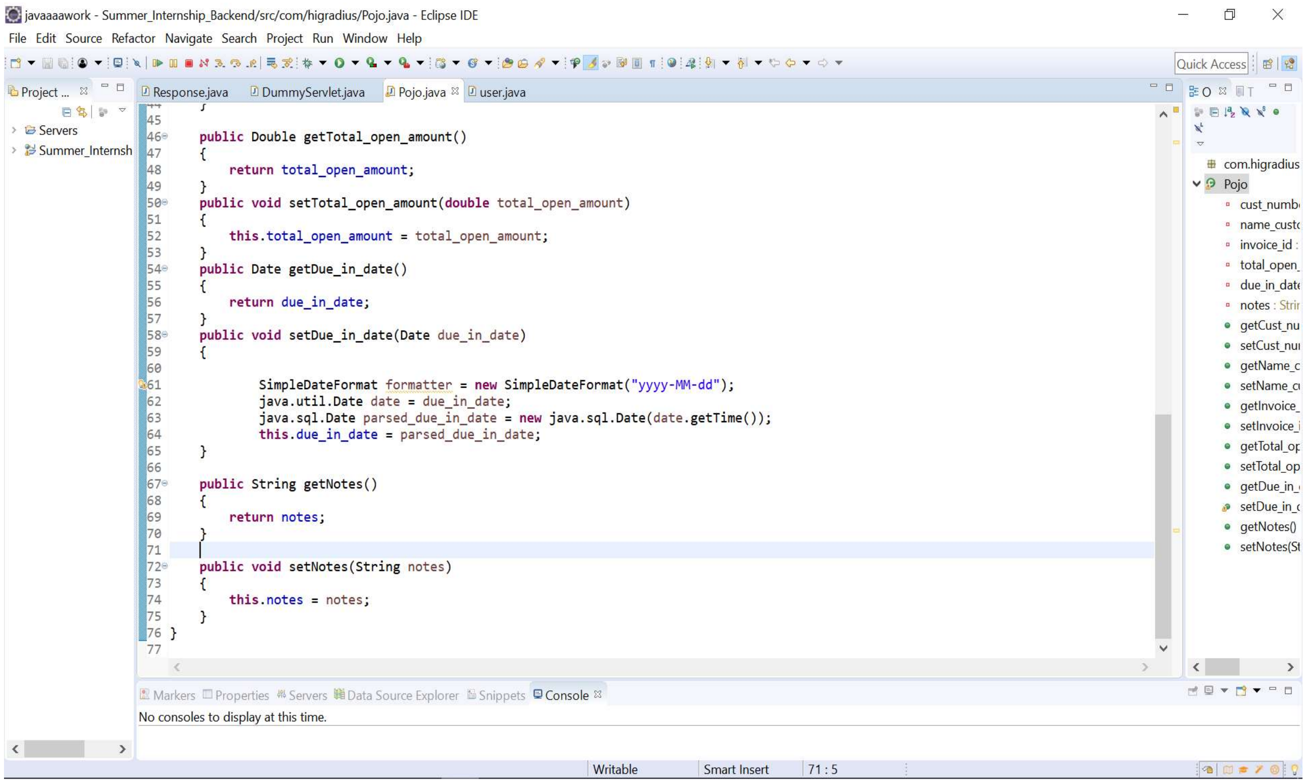

javaaaawork - Summer_Internship_Backend/src/com/higradius/user.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

```
     while (rs.next())
     {
       Pojo obj= new Pojo();

       obj.setName_customer(rs.getString("name_customer"));
       obj.setCust_number(rs.getString("cust_number"));
       obj.setInvoice_id(rs.getLong("invoice_id"));
       obj.setTotal_open_amount(rs.getDouble("total_open_amount"));
       obj.setDue_in_date(rs.getDate("due_in_date"));
       obj.setNotes(rs.getString("notes"));

       list.add(obj);
     }
     System.out.println(list);

     //out.print(data);
     GsonBuilder gsonBuilder = new GsonBuilder();
     Gson gson = gsonBuilder.create();
     jsonObj = gson.toJson(list);
     return jsonObj;
   }
    catch(SQLException e)
    {
       e.printStackTrace();
    }
    catch(ClassNotFoundException e)
    {
       e.printStackTrace();
    }
    finally
    {
       try
```

No consoles to display at this time.

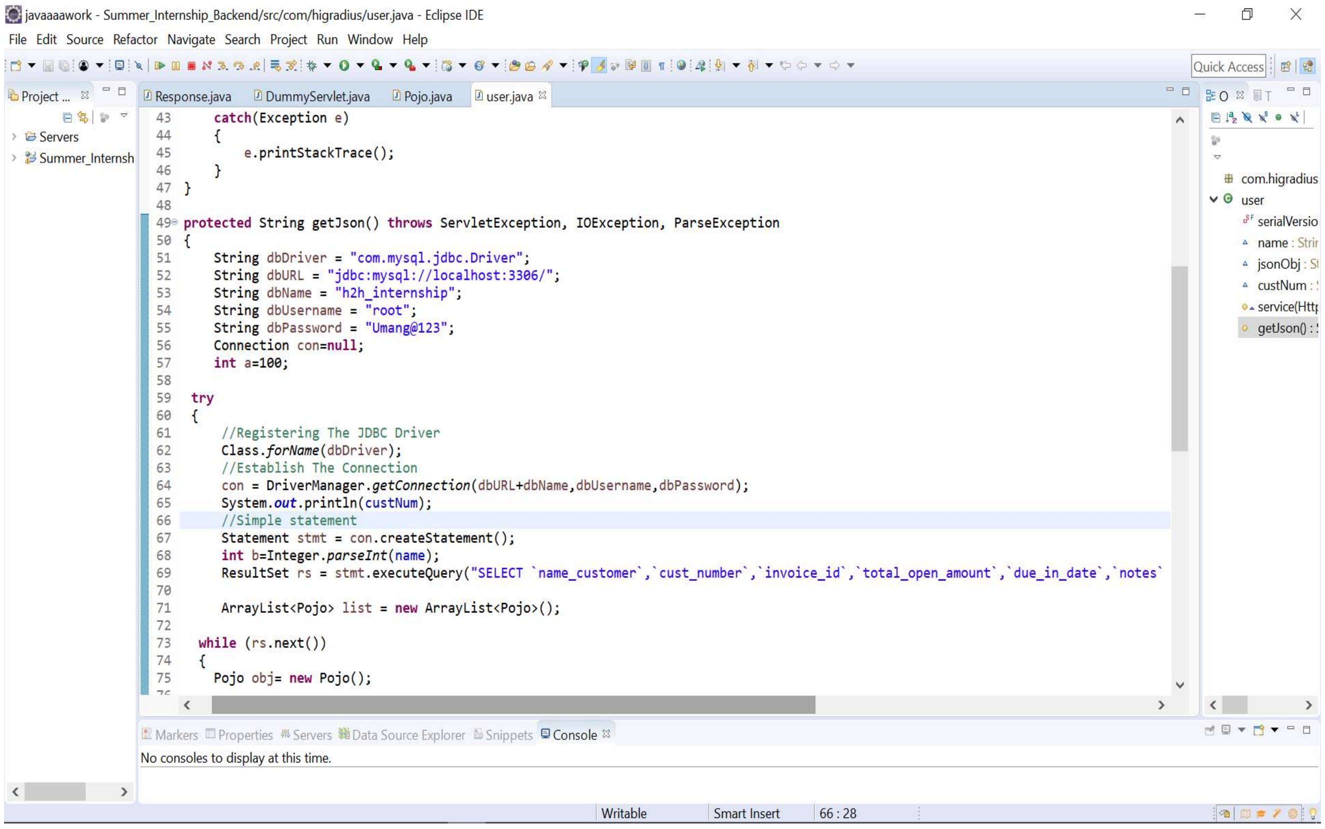
Writable Smart Insert 66 : 28



```
    }

    public Double getTotal_open_amount()
    {
        return total_open_amount;
    }
    public void setTotal_open_amount(double total_open_amount)
    {
        this.total_open_amount = total_open_amount;
    }
    public Date getDue_in_date()
    {
        return due_in_date;
    }
    public void setDue_in_date(Date due_in_date)
    {

            SimpleDateFormat formatter = new SimpleDateFormat("yyyy-MM-dd");
            java.util.Date date = due_in_date;
            java.sql.Date parsed_due_in_date = new java.sql.Date(date.getTime());
            this.due_in_date = parsed_due_in_date;
    }

    public String getNotes()
    {
        return notes;
    }

    public void setNotes(String notes)
    {
        this.notes = notes;
    }
}
```

```
No consoles to display at this time.
```

Writable   Smart Insert   71 : 5



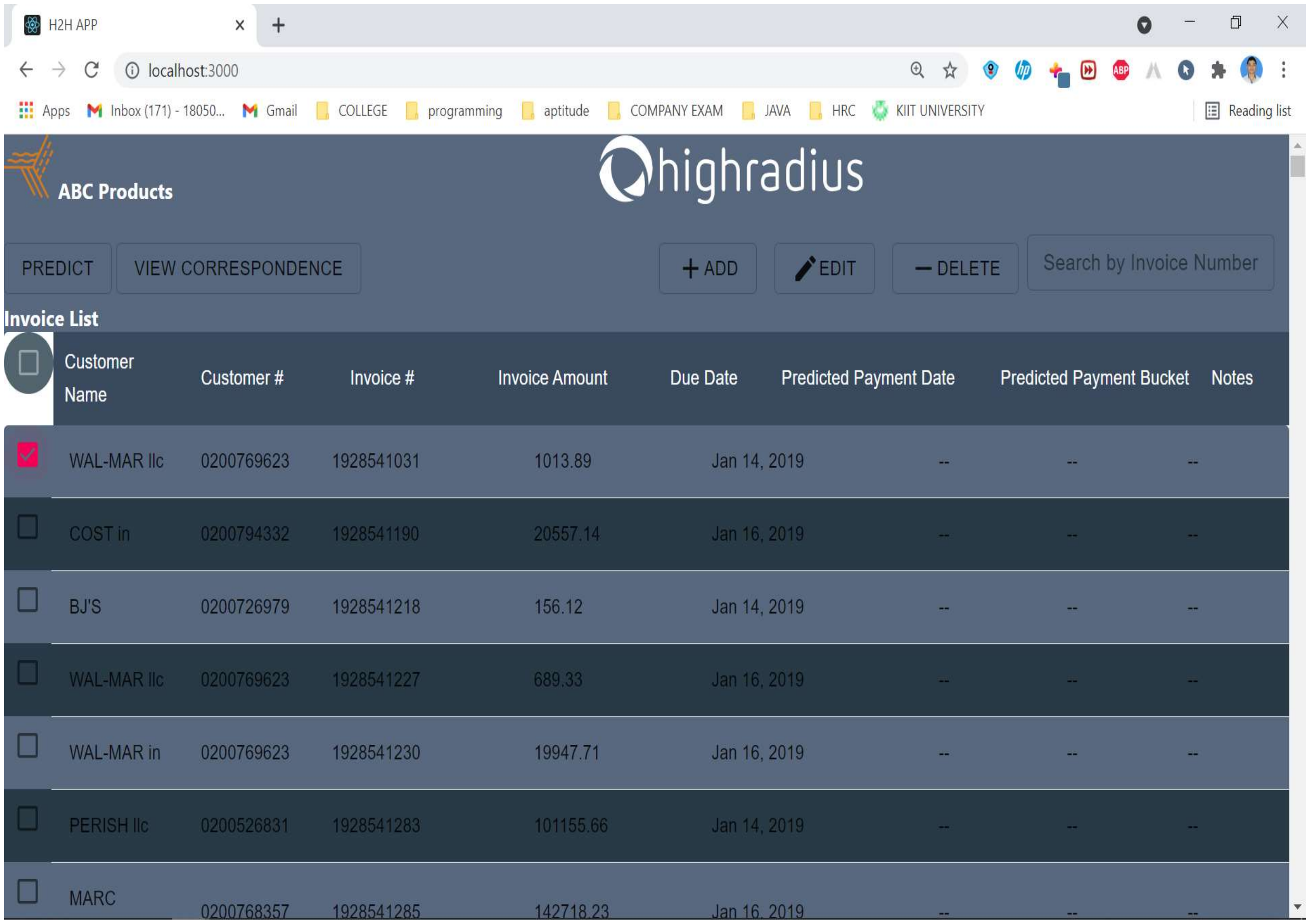
```
    catch(Exception e)
    {
        e.printStackTrace();
    }
}

protected String getJson() throws ServletException, IOException, ParseException
{
    String dbDriver = "com.mysql.jdbc.Driver";
    String dbURL = "jdbc:mysql://localhost:3306/";
    String dbName = "h2h_internship";
    String dbUsername = "root";
    String dbPassword = "Umang@123";
    Connection con=null;
    int a=100;

 try
 {
     //Registering The JDBC Driver
     Class.forName(dbDriver);
     //Establish The Connection
     con = DriverManager.getConnection(dbURL+dbName,dbUsername,dbPassword);
     System.out.println(custNum);
     //Simple statement
     Statement stmt = con.createStatement();
     int b=Integer.parseInt(name);
     ResultSet rs = stmt.executeQuery("SELECT `name_customer`,`cust_number`,`invoice_id`,`total_open_amount`,`due_in_date`,`notes`

     ArrayList<Pojo> list = new ArrayList<Pojo>();

  while (rs.next())
  {
    Pojo obj= new Pojo();
```

```
No consoles to display at this time.
```

Writable   Smart Insert   66 : 28

Frontend Desigm:

UI Representation of the Data:



Invoice List

	Customer Name	Customer #	Invoice #	Invoice Amount	Due Date	Predicted Payment Date	Predicted Payment Bucket	Notes
✓	WAL-MAR llc	0200769623	1928541031	1013.89	Jan 14, 2019	--	--	--
☐	COST in	0200794332	1928541190	20557.14	Jan 16, 2019	--	--	--
☐	BJ'S	0200726979	1928541218	156.12	Jan 14, 2019	--	--	--
☐	WAL-MAR llc	0200769623	1928541227	689.33	Jan 16, 2019	--	--	--
☐	WAL-MAR in	0200769623	1928541230	19947.71	Jan 16, 2019	--	--	--
☐	PERISH llc	0200526831	1928541283	101155.66	Jan 14, 2019	--	--	--
☐	MARC	0200768357	1928541285	142718.23	Jan 16, 2019	--	--	--

Receivables Dashboard Page

It consists of 2 sections:

1. Header

2. Grid Panel Section

1. Header Section

The header consists of :

- Account name logo <ABC Products> on the left
- The HighRadius 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 hello bro !!

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

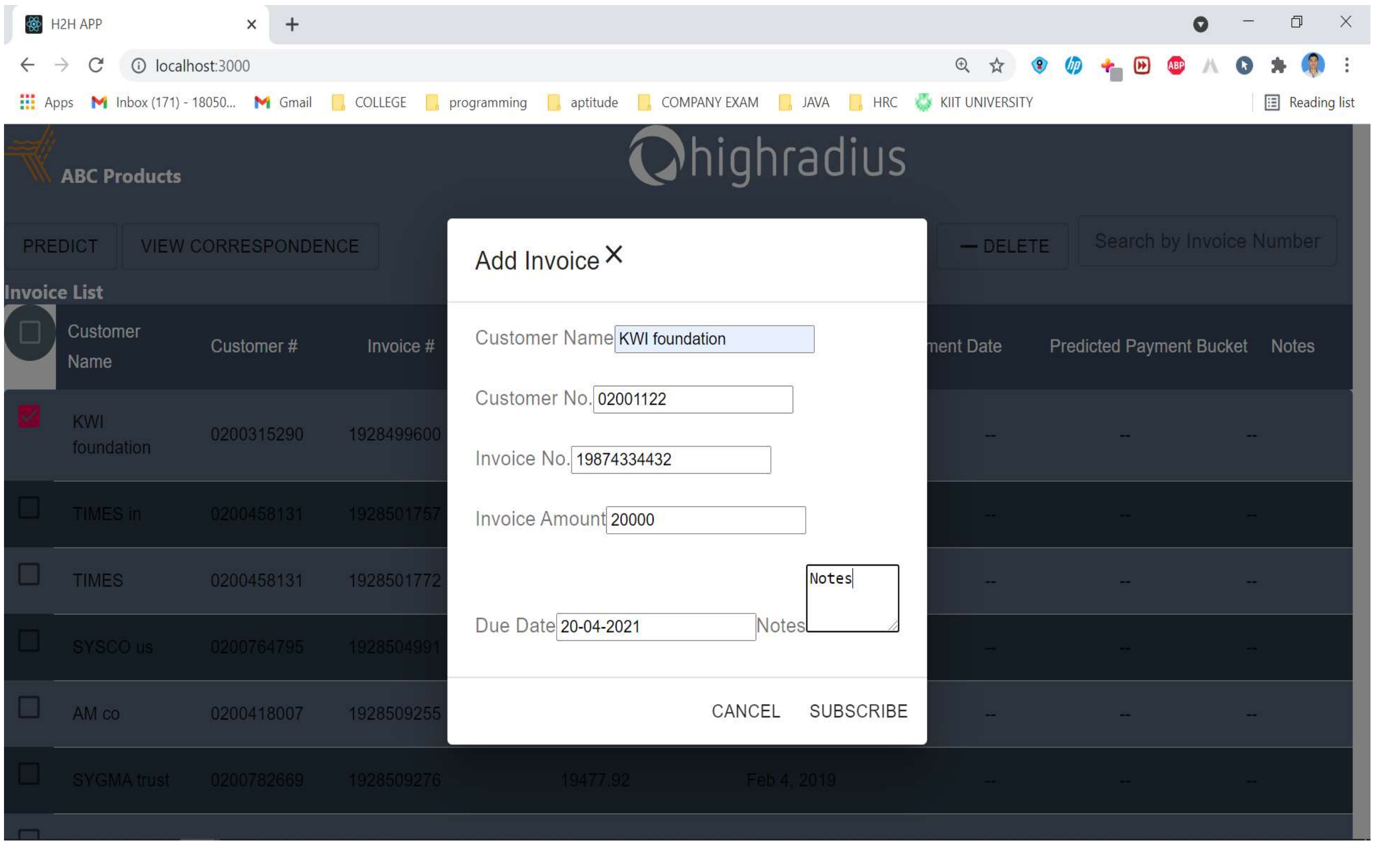
Delete Button Functionality - Deletes an invoice in the database

View Correspondence Button Functionality - Helps in viewing an invoice or a group of invoices in detail and exporting it.

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.



Add Invoice

Customer Name KWI foundation

Customer No. 02001122

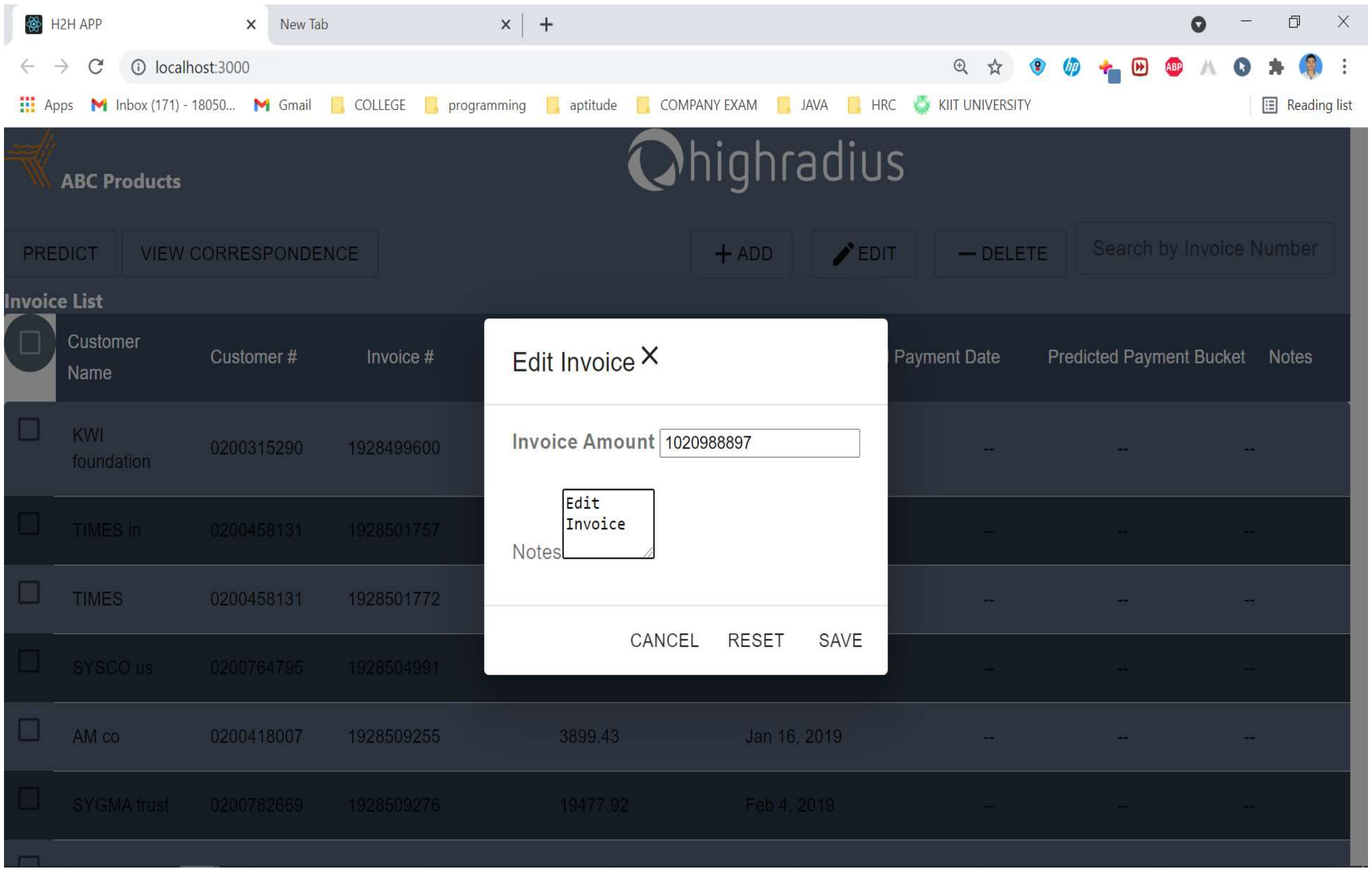
Invoice No. 19874334432

Invoice Amount 20000

Due Date 20-04-2021

Notes Notes

	Customer Name	Customer #	Invoice #
✓	KWI foundation	0200315290	1928499600
	TIMES in	0200458131	1928501757
	TIMES	0200458131	1928501772
	SYSCO us	0200764795	1928504991
	AM co	0200418007	1928509255
	SYGMA trust	0200782669	1928509276



Invoice List

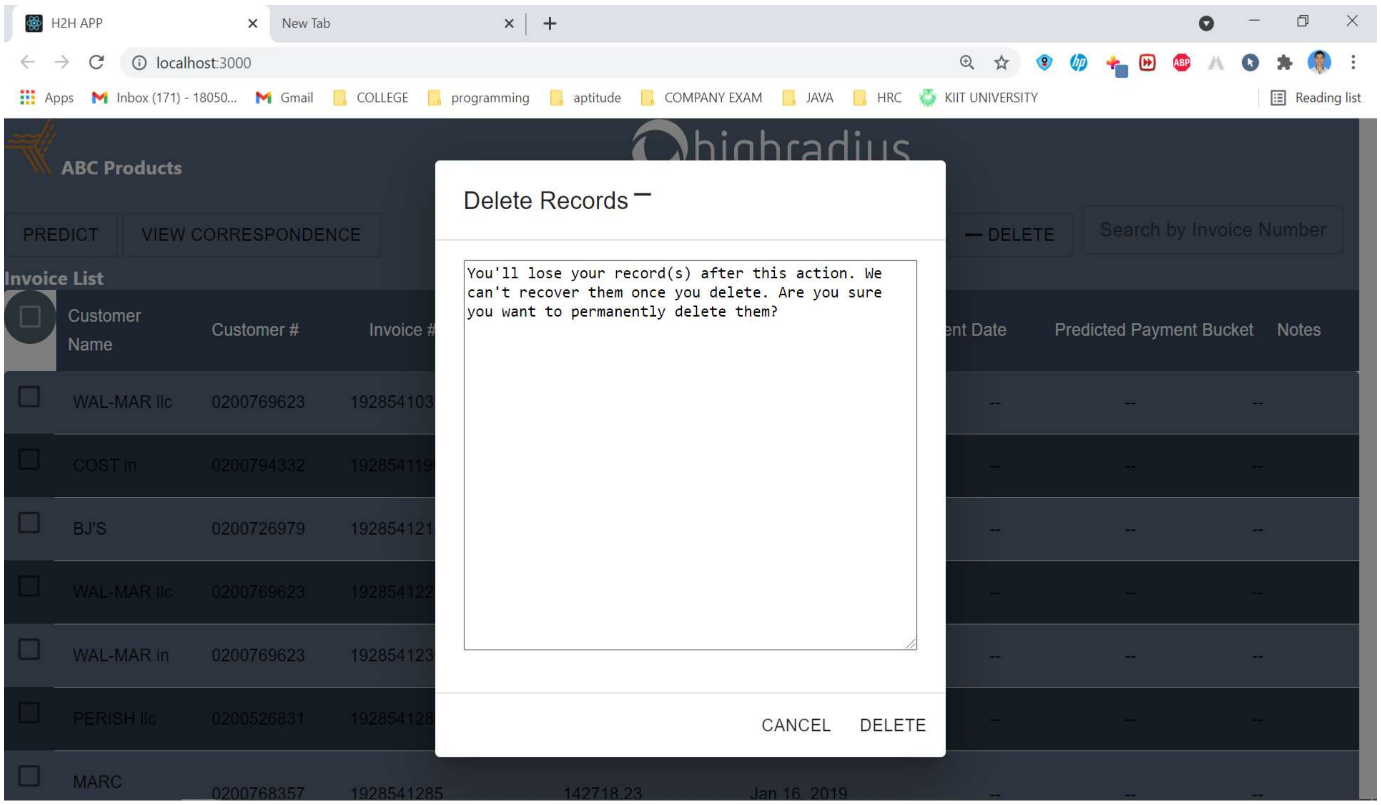
Customer Name	Customer #	Invoice #			Payment Date	Predicted Payment Bucket	Notes
KWI foundation	0200315290	1928499600			--	--	--
TIMES in	0200458131	1928501757			--	--	--
TIMES	0200458131	1928501772			--	--	--
SYSCO us	0200764795	1928504991			--	--	--
AM co	0200418007	1928509255	3899.43	Jan 16, 2019	--	--	--
SYGMA trust	0200782669	1928509276	19477.92	Feb 4, 2019	--	--	--

Edit Invoice ×

Invoice Amount 1020988897

Notes
```
Edit
Invoice
```

CANCEL RESET SAVE


Delete Records

You'll lose your record(s) after this action. We can't recover them once you delete. Are you sure you want to permanently delete them?

Problems faced while doing the project

1) The Frontend Part was quite difficult since I was a complete beginner in this field.

2) Understanding how to forward actions from front-end to the back-end was quite difficult.

3) Infinite Loading was quite challenging.

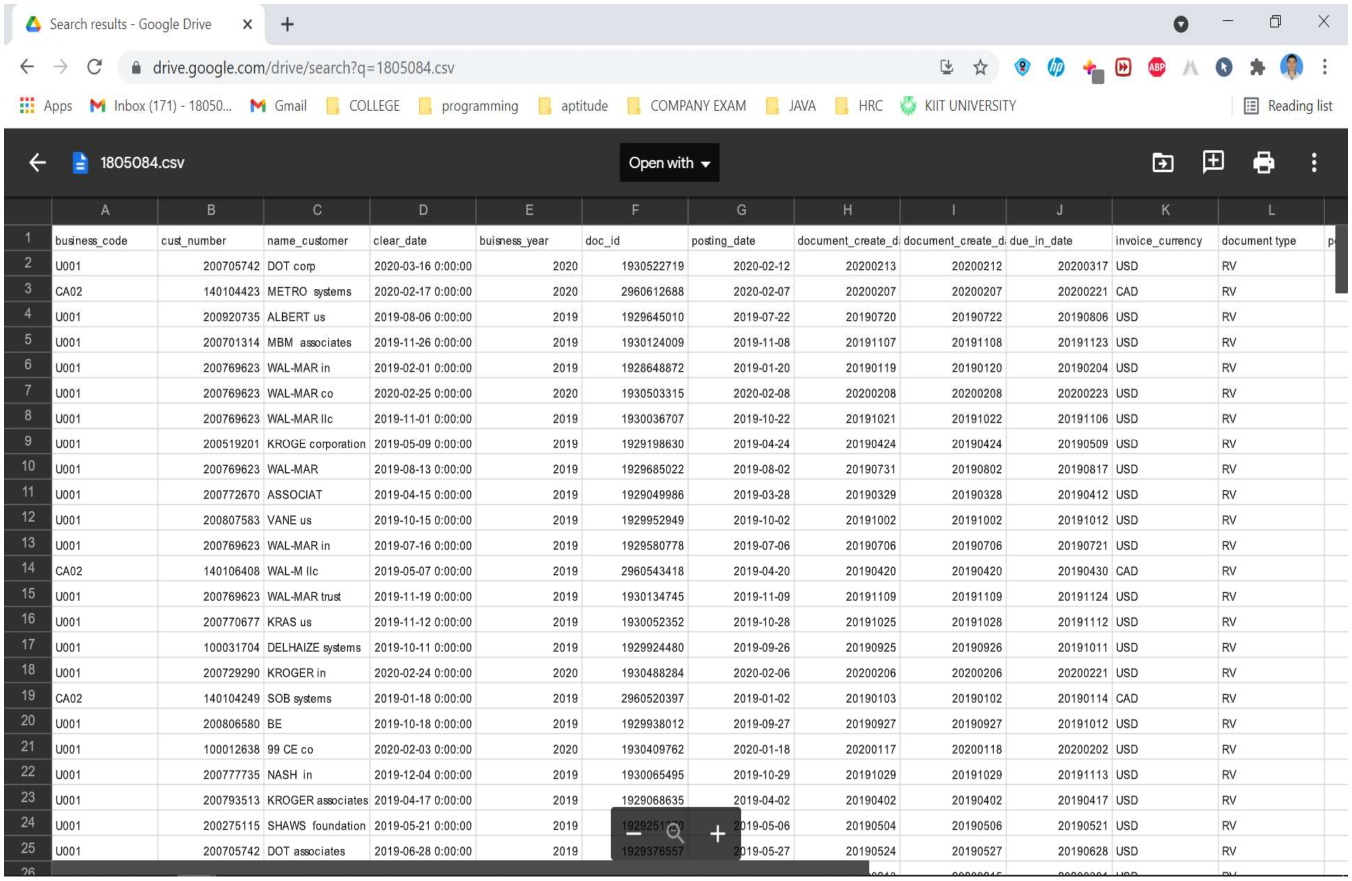
4) UI design was also difficult to implement since I am very new to Javascript. Logical interfacing of the functionalities was the most challenging work.

Chapter 4

Results & Discussion

Below is the sample CSV file screenshot.

All the Columns of the CSV file need to be loaded into the DB.


Search results - Google Drive
drive.google.com/drive/search?q=1805084.csv
1805084.csv
Open with

	A	B	C	D	E	F	G	H	I	J	K	L
1	business_code	cust_number	name_customer	clear_date	buisness_year	doc_id	posting_date	document_create_d	document_create_d	due_in_date	invoice_currency	document type
2	U001	200705742	DOT corp	2020-03-16 0:00:00	2020	1930522719	2020-02-12	20200213	20200212	20200317	USD	RV
3	CA02	140104423	METRO systems	2020-02-17 0:00:00	2020	2960612688	2020-02-07	20200207	20200207	20200221	CAD	RV
4	U001	200920735	ALBERT us	2019-08-06 0:00:00	2019	1929645010	2019-07-22	20190720	20190722	20190806	USD	RV
5	U001	200701314	MBM associates	2019-11-26 0:00:00	2019	1930124009	2019-11-08	20191107	20191108	20191123	USD	RV
6	U001	200769623	WAL-MAR in	2019-02-01 0:00:00	2019	1928648872	2019-01-20	20190119	20190120	20190204	USD	RV
7	U001	200769623	WAL-MAR co	2020-02-25 0:00:00	2020	1930503315	2020-02-08	20200208	20200208	20200223	USD	RV
8	U001	200769623	WAL-MAR llc	2019-11-01 0:00:00	2019	1930036707	2019-10-22	20191021	20191022	20191106	USD	RV
9	U001	200519201	KROGE corporation	2019-05-09 0:00:00	2019	1929198630	2019-04-24	20190424	20190424	20190509	USD	RV
10	U001	200769623	WAL-MAR	2019-08-13 0:00:00	2019	1929685022	2019-08-02	20190731	20190802	20190817	USD	RV
11	U001	200772670	ASSOCIAT	2019-04-15 0:00:00	2019	1929049986	2019-03-28	20190329	20190328	20190412	USD	RV
12	U001	200807583	VANE us	2019-10-15 0:00:00	2019	1929952949	2019-10-02	20191002	20191002	20191012	USD	RV
13	U001	200769623	WAL-MAR in	2019-07-16 0:00:00	2019	1929580778	2019-07-06	20190706	20190706	20190721	USD	RV
14	CA02	140106408	WAL-M llc	2019-05-07 0:00:00	2019	2960543418	2019-04-20	20190420	20190420	20190430	CAD	RV
15	U001	200769623	WAL-MAR trust	2019-11-19 0:00:00	2019	1930134745	2019-11-09	20191109	20191109	20191124	USD	RV
16	U001	200770677	KRAS us	2019-11-12 0:00:00	2019	1930052352	2019-10-28	20191025	20191028	20191112	USD	RV
17	U001	100031704	DELHAIZE systems	2019-10-11 0:00:00	2019	1929924480	2019-09-26	20190925	20190926	20191011	USD	RV
18	U001	200729290	KROGER in	2020-02-24 0:00:00	2020	1930488284	2020-02-06	20200206	20200206	20200221	USD	RV
19	CA02	140104249	SOB systems	2019-01-18 0:00:00	2019	2960520397	2019-01-02	20190103	20190102	20190114	CAD	RV
20	U001	200806580	BE	2019-10-18 0:00:00	2019	1929938012	2019-09-27	20190927	20190927	20191012	USD	RV
21	U001	100012638	99 CE co	2020-02-03 0:00:00	2020	1930409762	2020-01-18	20200117	20200118	20200202	USD	RV
22	U001	200777735	NASH in	2019-12-04 0:00:00	2019	1930065495	2019-10-29	20191029	20191029	20191113	USD	RV
23	U001	200793513	KROGER associates	2019-04-17 0:00:00	2019	1929068635	2019-04-02	20190402	20190402	20190417	USD	RV
24	U001	200275115	SHAWS foundation	2019-05-21 0:00:00	2019	19292512[illegible]0	2019-05-06	20190504	20190506	20190521	USD	RV
25	U001	200705742	DOT associates	2019-06-28 0:00:00	2019	1929376557	2019-05-27	20190524	20190527	20190628	USD	RV

The **Predicted Payment Date** and **Predicted Aging Bucket columns will remain blank** bydefault.

Clicking on the **Predict button** after selecting the one or more rows **will auto populatethe two columns with the values derived from the ML model**.

The grid will also have a **Select all and Deselect All functionality** to select one or morerecords.

The past due invoices should have the due date marked in red in the grid.

The third portion is the **vertical scroll bar**. The grid should support infinite scrolling.

A loading icon animation should be displayed while the records are loading.

The grid should be **sortable** by two columns- **due date** and **invoice amount** in **ascendingand descending order.**

<u>**Functionalities in Detail:**</u>

AddButtonFunctionality:

The Add button will remain in **enabled state** if no rows are selected.
Whenever one or more rows are selected, the Add button will still remain activated.
After clicking on the Add button, a pop up window will appear with all the fields forwhich values need to be added along with a Cancel and a Save button.
The user should be able to **type in the values,** except for the due date of the invoice forwhich there should be a calendar view from where the user is able to select the requireddate, month and year.
The user should fill all the required fields before adding. If the user tries to click on addbefore all mandatory fields are filled, an error message will be displayed.
Once the user clicks on the add button after all mandatory fields are filled, the newvalues should be displayed in the UI and should remain persistent.

<u>**Edit ButtonFunctionality:**</u>

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 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 shouldcontain the Invoice Amount and Notes headers along with the existing data values forboth, 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 UIand should remain persistent.

<u>DeleteButtonFunctionality:</u>

Clicking on the **delete button** will allow the user to **delete records** from the grid.
The delete button will **remain in disabled state** by default.
When the **user selects one or more rows**, the delete button gets enabled.
A pop up should be displayed on clicking delete to confirm that the user wants to delete the selected records permanently.
Once the user clicks on the delete button, the row(s) should be removed from the grid in the UI and should remain persistent.a

ViewCorrespondenceButtonFunctionality:

SingleTemplateinViewCorrespondenceButton(Level1):

Clicking on the **view correspondence button** will allow the user to **view and generatepdfs ofthe 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 getsenabled.
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 autopopulated in it along witha Cancel and a Download button.
The account name in the template should be dynamically filled. The total amount to bepaid should be the sum of the open amounts of the auto populated invoices in thetemplate.
The senders details should be static values.
On clicking the Download button, a file containing the data from the selected rows inthe selected template should be downloaded in pdf format.

MultipleTemplate sinViewCorrespondenceButton(Level2):

The user should be able to select from multiple template options from a drop downmenu.
Any template when selected should contain the data of the selected records autopopulated in it.
On clicking the download button, the downloaded file should contain the data in theformat of the template being viewed by the user at the time of clicking the downloadbutton.

PredictButtonFunctionality:

The Predict button will remain in **disabled state** if no rows are selected.
Whenever one or more rows are selected, the Predict button will be activated.
After clicking on the Predict button, the Predicted Payment Date and Predicted AgingBucket will be populated for the respective records.

SearchBarFunctionality:

The user should be able to type the invoice number in the search bar.
All invoices with matching invoice numbers should be filtered in the grid and thematching portion should be highlighted.
If no such invoice is present, a message should be displayed stating “No Results Found”.
The search bar should implement the debouncing concept. Debouncing allows afunction that is bound to an event to fire only once after a specific amount of time has passed.

Chapter 5

Conclusion & Future Scope

5.1 Conclusion

To build a Machine Learning Model to predict the Payment Date of an Invoice Date when it gets created in the system. Categorize the invoice into different buckets based on predicted payment date.

I will be receiving an invoices Dataset that contains the past payment information and behaviour of various buyers. Based on the previous payment patterns, the ML model needs to predict what will be the date a payment is made by the customer for an invoice. The model also needs to predict which aging bucket the invoice falls into based on the predicted payment date.

To build a full stack Invoice Management Application using ReactJs, JDBC, Java and JSP.Build a responsive Receivables Dashboard. Visualize Data in the form of grids. Perform Searching operations on the invoices. Edit data in the editable fields of the grid. Download data of selected rows in predefined templates.

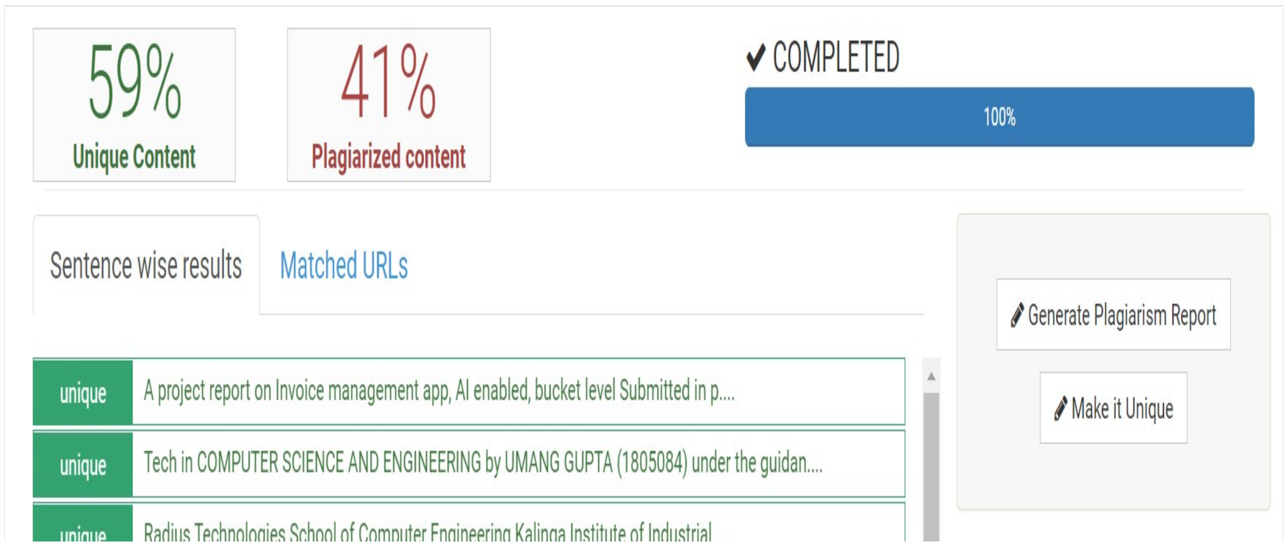
5.2 Future Scope

The Project - ” Invoice Management Application , AI , Bucket Level ” Which We Have Designed Using The React JS , Machine Learning Model , JavaScripts . This Project Deals With The B2B(Business To Business) IT HUB Industry In Which Better Predict The Invoice Payment Date Payment Date Of An Invoice Payment , With The Help OF When Two Business Companies Buys The Product.

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

The simplest definition of accounts receivable is money owed to an entity by its customers. Correspondingly, the amount not yet received is credit and, of course, the amount still owed past the due date is Account receivables.

PLAGARISM REPORT:



59%
Unique Content

41%
Plagiarized content

✔ COMPLETED

100%

Sentence wise results | Matched URLs

unique	A project report on Invoice management app, AI enabled, bucket level Submitted in p....
unique	Tech in COMPUTER SCIENCE AND ENGINEERING by UMANG GUPTA (1805084) under the guidan....
unique	Radius Technologies School of Computer Engineering Kalinga Institute of Industrial