

Project Report

On

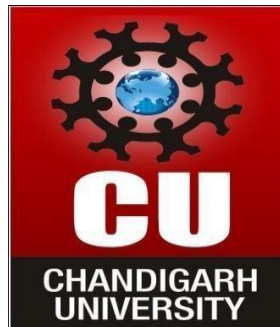
AI-Enabled FinTech B2B Invoice Management Application

Submitted for the requirement of

Project course

BACHELOR OF ENGINEERING

COMPUTER SCIENCE & ENGINEERING



Submitted to:

Er. Richa Dhiman (E11307)

Submitted By:

Rahul Pandey 19BCS2953

Project Teacher Signature

Er. Pooja (E11313)

Supervisor Signature

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
CHANDIGARH UNIVERSITY, GHARUAN
April 2022**

BONAFIDE CERTIFICATE

This is to certify that the work embodied in this Project Report entitled “**AI-Enabled FinTech B2B Invoice Management Application**” being submitted by “**Rahul Pandey**”- UID “**19BCS2953**”, 6 Semester for partial fulfillment of the requirement for the degree of “**Bachelor of Engineering in Computer Science & Engineering**” discipline in “**Chandigarh University**” during the academic session Feb-Jun 2022 is a record of bona fide piece of work, carried out by student under my supervision and guidance in the “**Department of Computer Science & Engineering , Chandigarh University**”.

APPROVED & GUIDED BY:

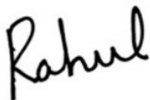
Er. Richa Dhiman (E11307)

ACKNOWLEDGEMENT

I, student of **Bachelor of Engineering in Computer Science & Engineering, 6th Semester** , session: **Feb – June 2022, Chandigarh University**, hereby declare that the work presented in this Project Report entitled “**AI-Enabled FinTech B2B Invoice Management Application** ” is the outcome of my own work, is bona fide and correct to the best of my knowledge and this work has been carried out taking care of Engineering Ethics. The work presented does not infringe any patented work and has not been submitted to any other university or anywhere else for the award of any degree or any professional diploma.

Student details and Signature

Rahul Pandey 19BCS2953



APPROVED & GUIDED BY:

Er. Richa Dhiman (E11307)

Er. Pooja (E11313)

Project Teacher Signature

Supervisor Signature

TABLE OF CONTENTS

Declaration	3
Abstract	5
List of Figures and Tables	7
1. Introduction	8
2. Background	9
2.1 Account receivables Department	
2.2 Problem Statement for ML	
2.3 Problem Statement for Web Application Development	
2.4 React Web App	
2.5 High Level Requirements	
2.5.1 Data Loading in DB	
2.5.2 UI Representation of Data	
2.5.3 AI Support in the application	
2.5.4 DFDs	
3. System Overview	13
3.1 Functional Overview	
3.2 UI Representation	
3.2.1 Add Button	
3.2.2 Edit Button	
3.2.3 Delete Button	
3.3 Programming Stage	
4. Platform Used	28
5. Conclusion	28
Glossary	28
Bibliography	30

LIST OF TABLES

➤ Mandatory Features	2.4.1
➤ Column Names	3.1.1
➤ Columns in UI	3.3.1

LIST OF FIGURES

➤ Level Zero DFD	2.5.4.1
➤ First Level DFD	2.5.4.2
➤ CSV File	3.1.1
➤ Libraries	3.1.2
➤ UI Representation	3.2.1
➤ Add Invoice	3.2.1.1
➤ Edit Invoice	3.2.2.1
➤ Delete Invoice	3.2.3.1 3.2.3.2
➤ Plot	3.3.1
➤ Training Data	3.3.2
➤ Pearson Correlation	3.3.3
➤ Storing Data frame	3.3.4
➤ Bar Chart	3.3.5
➤ Pie Chart	3.3.6

ABSTRACT

This project is a part of the Highway to Highradius program. We are required to build an AI-Enabled FinTech B2B Invoice Management Application.

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. This is known in accounting terminology as “Accounts Receivable”.

“Accounts Receivable represents money owed by entities to the firm on the sale of products or services on credit. In most business entities, accounts receivable is typically executed by generating an invoice and either mailing or electronically delivering it to the customer, who, in turn, must pay it within an established timeframe, called credit terms or payment terms.”

Seller business interacts with various businesses and sells goods to all of them at various times. Hence, the seller business needs to keep track of the total amount it owes from all the buyers. This involves keeping track of all invoices from all the buyers. Each invoice will have various important fields like a payment due date, invoice date, invoice amount, baseline date etc.

The buyer business needs to clear its amount due before the due date. However, in real-world scenarios, the invoices are not always cleared ie. paid in full amount by the due date. The date on which a customer clears the payment for an invoice is called the payment date.

GRAPHICAL ABSTRACT

- UI Representation of Data :-

SI no	Business Code	Customer Number	Clear Date	Business Year	Document Id	Posting Date	Document Create Date	Due Date	Invoice Currency	Document Type	Posting Id	Total Open amount	Baseline Create Date
1	U001	200769623	2020-02-11	2020-01-01	1930438491	2020-01-26	2020-01-25	2020-02-10	USD	RV	1	54273.28	2020-01-26
2	U001	200980828	2019-08-08	2019-01-01	1929646410	2019-07-22	2019-07-22	2019-08-11	USD	RV	1	79656.6	2019-07-22
3	U001	200792734	2019-12-30	2019-01-01	1929873765	2019-09-14	2019-09-14	2019-09-29	USD	RV	1	2253.86	2019-09-14
4	CA02	140105686	0000-00-00	2020-01-01	2960623488	2020-03-30	2020-03-30	2020-04-10	CAD	RV	1	3299.7	2020-03-31
5	U001	200769623	2019-11-26	2019-01-01	1930147874	2019-11-13	2019-11-13	2019-11-28	USD	RV	1	33133.29	2019-11-13

Rows per page: 5 1-5 of 200

[Privacy Policy](#) | © 2022 HighRadius Corporation. All rights reserved.

- Receivables Dashboard Page :-

Add

Business Code Customer Number Clear Date 01/25/2022 Business Year

Document id Posting Date 01/26/2022 Document Create Date 01/25/2022 Due Date 01/25/2022

Invoice Currency Document type Posting Id Total open amount

Baseline Create Date 01/26/2022 Customer Payment Terms Invoice Id

ADD CANCEL

1. Introduction

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. We 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.

The objective of the Web Application Development internship project is:

- To build a Full-stack Invoice Management Application using Reacts, JDBC, Java, 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.

In the ideal world, the buyer business should pay back within the stipulated time (ie the Payment Term). However, in the real world, the buyer business seldom pays within their established time frame, and this is where the Account Receivables Department comes into the picture.

- Every business consists of a dedicated Account Receivables Department to collect and track payment of invoices. 3. It consists of an Account receivables team that is responsible for:
 - Collecting payments from customers for their past due to invoices.
 - Sending reminders and follow-ups to the customers for payments to be made.

- Looking after the entire process of getting the cash inflow.
- Help the company get paid for the services and products supplied.

2. Literature Survey

2.1 Account receivables Department:

1. In the ideal world, the buyer business should pay back within the stipulated time (i.e. the Payment Term). However, in the real world, the buyer business seldom pays within their established time frame, and this is where the Account Receivables Department comes into the picture.
2. Every business consists of a dedicated Account receivable. Department to collect and track payment of invoices.
3. It consists of an Account receivables team that is responsible for:
 - Collecting payments from customers for their past due to invoices.
 - Sending reminders and follow-ups to the customers for payments to be made.
 - Looking after the entire process of getting the cash inflow.
 - Help the company get paid for the services and products supplied.

2.2 Problem Statement for ML:

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. We 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.

2.3 Problem Statement for Web Application Development:

The objective of the Web Application Development internship project is:

- To build a Full-stack Invoice Management Application using ReactJs,JDBC, Java, 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.

2.4 React Web App:

The mandatory features are as follows:

MANDATORY FEATURES-
1. UI Creation
2. Grid Creation
3. Grid Data Loading
4. Crud Operation <ul style="list-style-type: none">a) ADDb) EDITc) DELETE
5. Pagination
6. Advanced Search
7. Grid Reloading Button

Table 2.4.1

2.5 High-level requirements of application

2.5.1 Data Loading in DB:

- You will be provided with an invoices dataset which you need to parse, process, and load in the provided database schemas.

2.5.2 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 of some editable fields, adding a new row to the grid, deleting rows from the grid and advance search.

2.5.3 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.

3. Design flow/Process



Figure 2.5.4.1

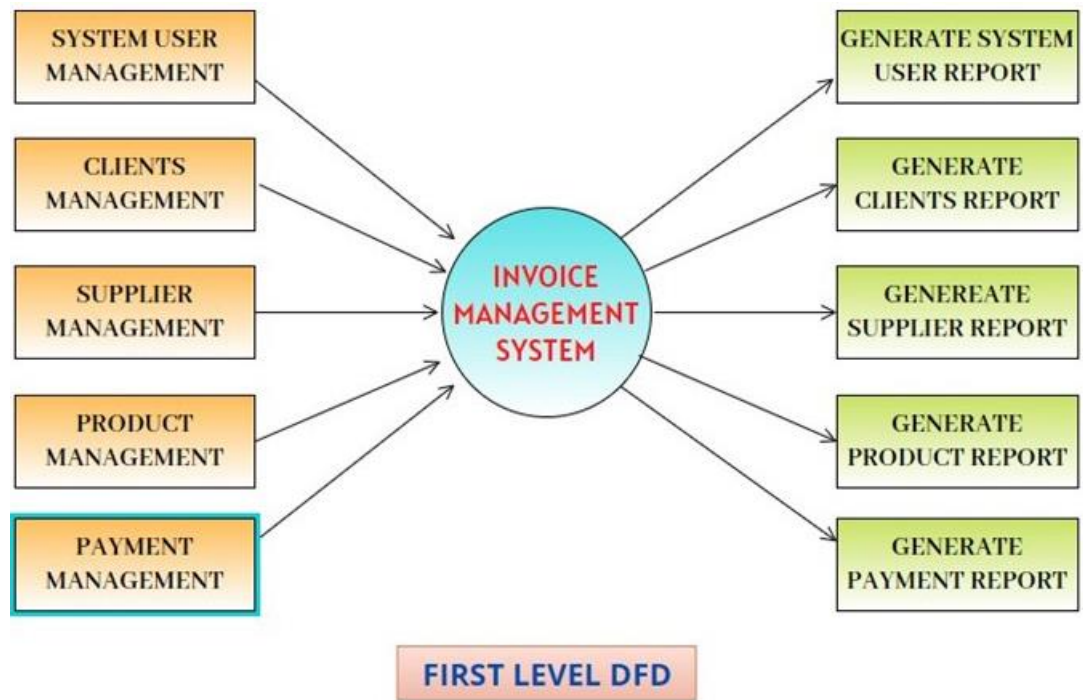


Figure 2.5.4.2

4. Results analysis and validation

4.1 Functional Overview

Below the csv file screenshot:

K17											14514.68	
	A	B	C	D	E	F	G	H	I	J	K	L
1	business_	cust_num	name_cus	clear_date	buisness_	doc_id	posting_d	due_in_d	baseline_	cust_payn	converted	Aging Bucket
2	CA02	1.4E+08	SYSC llc	18:26.5	2020	2.96E+09	00:00.0	00:00.0	00:00.0	CA10	2309.79	0-15
3	U001	2.01E+08	TARG us	52:32.9	2020	1.93E+09	00:00.0	00:00.0	00:00.0	NAA8	11173.02	0-15
4	U001	2E+08	AM	37:07.5	2020	1.93E+09	00:00.0	00:00.0	00:00.0	NAA8	3525.59	0-15
5	U001	2.01E+08	OK system	45:48.8	2020	1.93E+09	00:00.0	00:00.0	00:00.0	NAA8	121105.7	0-15
6	U001	2E+08	DECA corp	20:59.9	2020	1.93E+09	00:00.0	00:00.0	00:00.0	NAM2	3726.06	
7	U001	2.01E+08	TARG assc	52:32.9	2020	1.93E+09	00:00.0	00:00.0	00:00.0	NAA8	5893.01	0-15
8	CA02	1.4E+08	WAL-M co	13:03.7	2020	2.96E+09	00:00.0	00:00.0	00:00.0	CA10	64982.59	0-15
9	U001	2E+08	COAS llc	11:07.6	2020	1.93E+09	00:00.0	00:00.0	00:00.0	NAA8	11380.83	
10	U001	2.01E+08	COST asso	32:28.8	2020	1.93E+09	00:00.0	00:00.0	00:00.0	NAAX	3863.93	
11	U001	2.01E+08	COST llc	32:28.8	2020	1.93E+09	00:00.0	00:00.0	00:00.0	NAAX	74453.01	
12	U001	2.01E+08	DEC corp	36:18.1	2020	1.93E+09	00:00.0	00:00.0	00:00.0	NAM4	138.6	
13	U001	2.01E+08	COST co	32:28.8	2020	1.93E+09	00:00.0	00:00.0	00:00.0	NAAX	32715.47	
14	U001	2E+08	DEC us	20:59.9	2020	1.93E+09	00:00.0	00:00.0	00:00.0	NAM4	174.72	
15	U001	2.01E+08	WAL-MAR	31:49.8	2020	1.93E+09	00:00.0	00:00.0	00:00.0	NAH4	767.78	
16	CA02	1.4E+08	SHOPPE fc	16:35.7	2020	2.96E+09	00:00.0	00:00.0	00:00.0	CA10	1980.72	0-15
17	U001	2.01E+08	WAL-MAR	31:49.8	2020	1.93E+09	00:00.0	00:00.0	00:00.0	NAH4	14514.68	
18	U001	2.01E+08	WAL-MAR	31:49.8	2020	1.93E+09	00:00.0	00:00.0	00:00.0	NAH4	55400.31	
19	U001	2.01E+08	ASSOCIAT	09:41.1	2020	1.93E+09	00:00.0	00:00.0	00:00.0	NAU5	174766.4	0-15
20	U001	2.01E+08	SYSCO sys	52:27.3	2020	1.93E+09	00:00.0	00:00.0	00:00.0	NA32	31512.75	0-15
21	U001	2.01E+08	WAL-MAR	31:49.8	2020	1.93E+09	00:00.0	00:00.0	00:00.0	NAH4	19701.55	
22	U001	2.01E+08	BJ'S trust	34:11.4	2020	1.93E+09	00:00.0	00:00.0	00:00.0	NAA8	132.72	0-15
23	U001	2.01E+08	WAL-MAR	31:49.8	2020	1.93E+09	00:00.0	00:00.0	00:00.0	NAH4	1121.15	
24	U001	2.01E+08	WAL-MAR	31:49.8	2020	1.93E+09	00:00.0	00:00.0	00:00.0	NAH4	11394.84	

Figure 4.1

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

List of all the fields part of dataset are as follows:	
business_code	
cust_number	
name_customer	
clear_date	
buisness_year	
doc_id	
posting_date	
due_in_date	
baseline_create_date	
cust_payment_terms	
converted_usd	
Aging Bucket	

Table 4.1.1

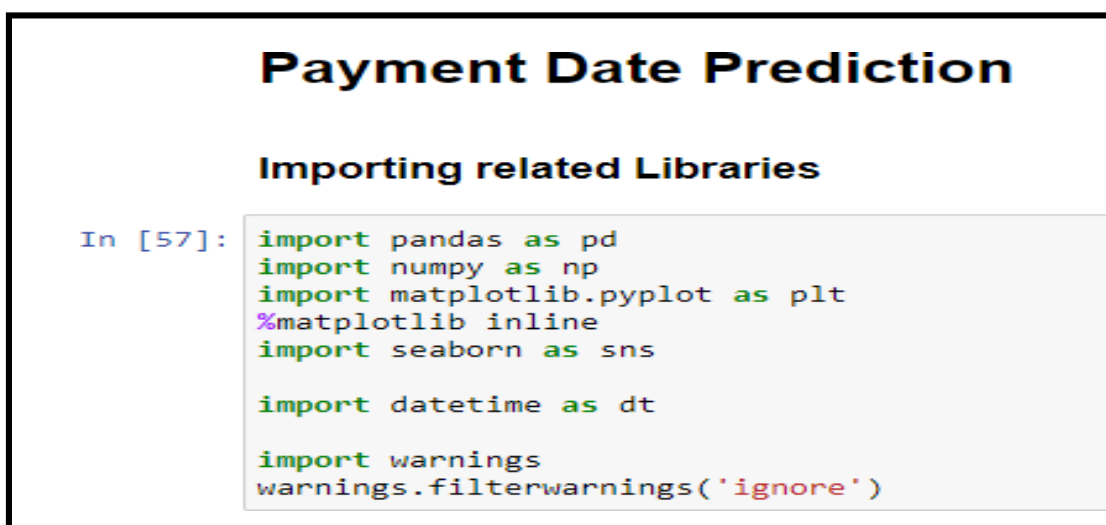


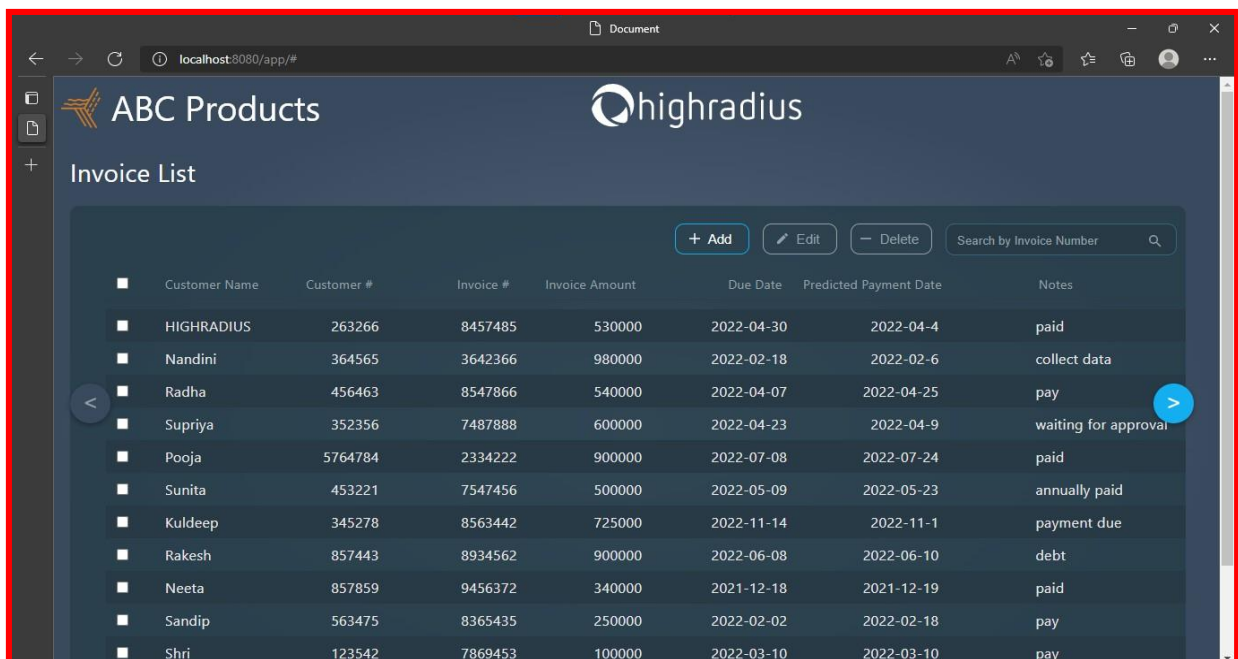
Figure 4.1.2

4.2 UI Representation of the Data

The UI consists of a single screen:

It consists of 2 sections:

- First Section is the header which comprises the ABC Product logo on the left, the Highradius Logo in the middle.
- The second section consists of Predict, Advance Search, Analytics View Add, Delete & Edit, and Search bar.



	Customer Name	Customer #	Invoice #	Invoice Amount	Due Date	Predicted Payment Date	Notes
<input type="checkbox"/>	HIGHRADIUS	263266	8457485	530000	2022-04-30	2022-04-4	paid
<input type="checkbox"/>	Nandini	364565	3642366	980000	2022-02-18	2022-02-6	collect data
<input type="checkbox"/>	Radha	456463	8547866	540000	2022-04-07	2022-04-25	pay
<input type="checkbox"/>	Supriya	352356	7487888	600000	2022-04-23	2022-04-9	waiting for approval
<input type="checkbox"/>	Pooja	5764784	2334222	900000	2022-07-08	2022-07-24	paid
<input type="checkbox"/>	Sunita	453221	7547456	500000	2022-05-09	2022-05-23	annually paid
<input type="checkbox"/>	Kuldeep	345278	8563442	725000	2022-11-14	2022-11-1	payment due
<input type="checkbox"/>	Rakesh	857443	8934562	900000	2022-06-08	2022-06-10	debt
<input type="checkbox"/>	Neeta	857859	9456372	340000	2021-12-18	2021-12-19	paid
<input type="checkbox"/>	Sandip	563475	8365435	250000	2022-02-02	2022-02-18	pay
<input type="checkbox"/>	Shri	123542	7869453	100000	2022-03-10	2022-03-10	pay

Figure 4.2

Add Button

- It is used for adding new field values to the grid.
- The Add button will be in the enabled state if no row is 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 for which values need to be added along with a Cancel and an Add button.
- The user should be able to type in the values, except for the date of the invoice for which there should be a calendar view from where the user is able to select the required date, month, and year.

- The user should fill in all the required fields before adding. If the user tries to click on add before all mandatory fields are filled, the user will not be able to add.

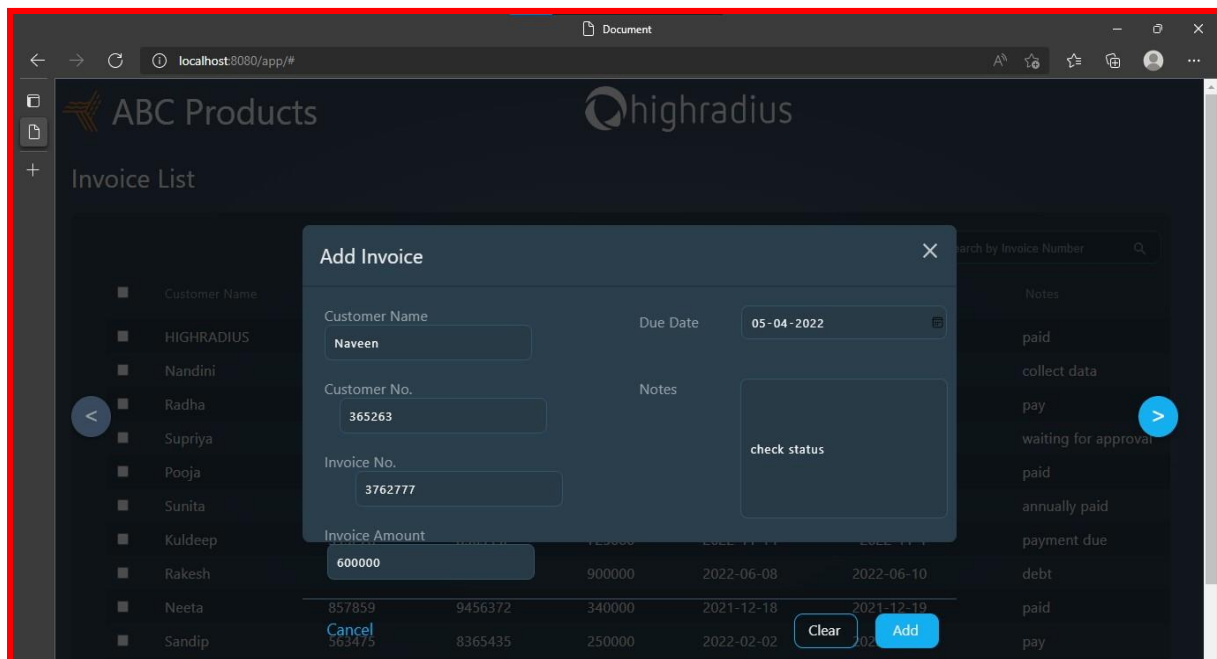


Figure 4.2.1

Edit Button

- It is used for editing the editable field values in the grid.
- Edit button should be disabled at first and should enable only one checkbox is selected
- A user should be able to select a row and then click on the Edit button.
- The fields which can be edited are the Invoice Currency and Customer Payment Terms fields. Without selecting any row, the Edit button should remain disabled.
- On clicking the Edit button, a popup should open up with the column header name and existing value.

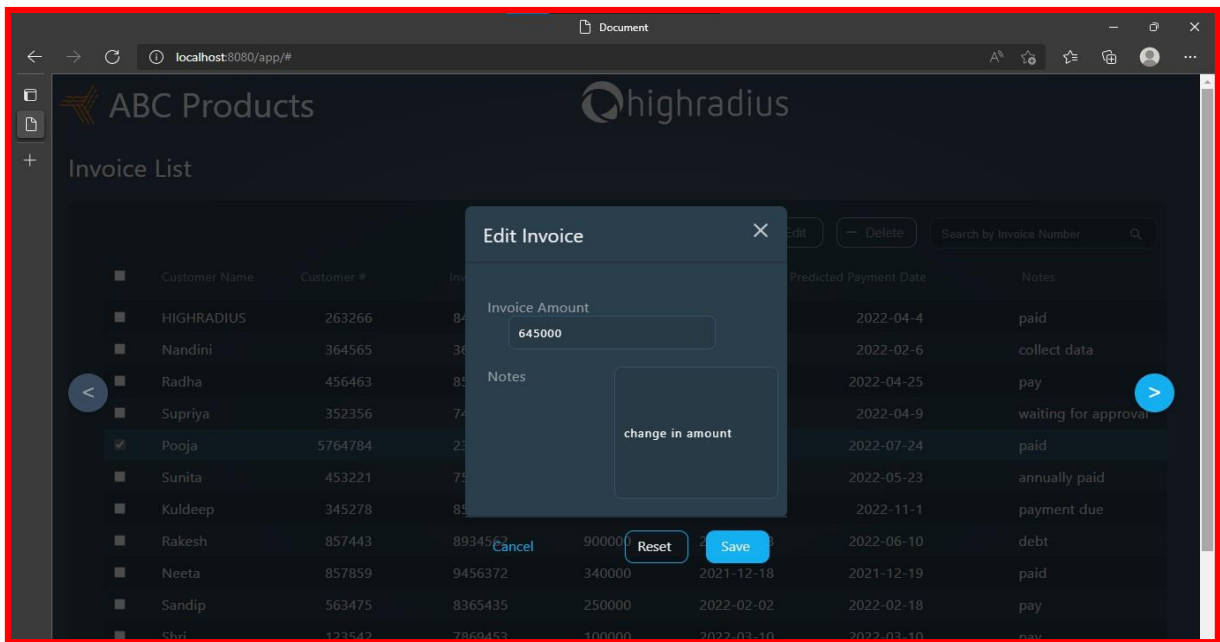


Figure 4.2.2

Delete Button

- Clicking on the delete button will allow the user to delete records from the grid.
- 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.

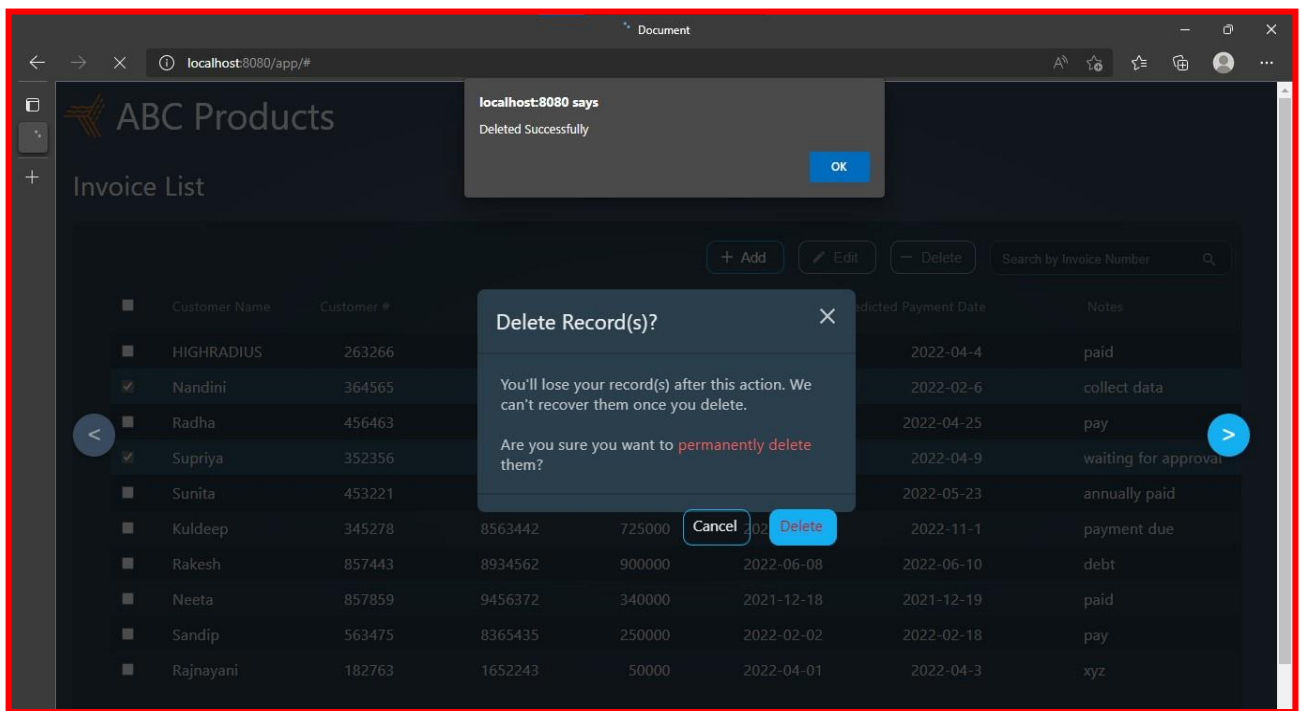


Figure 4.2.3

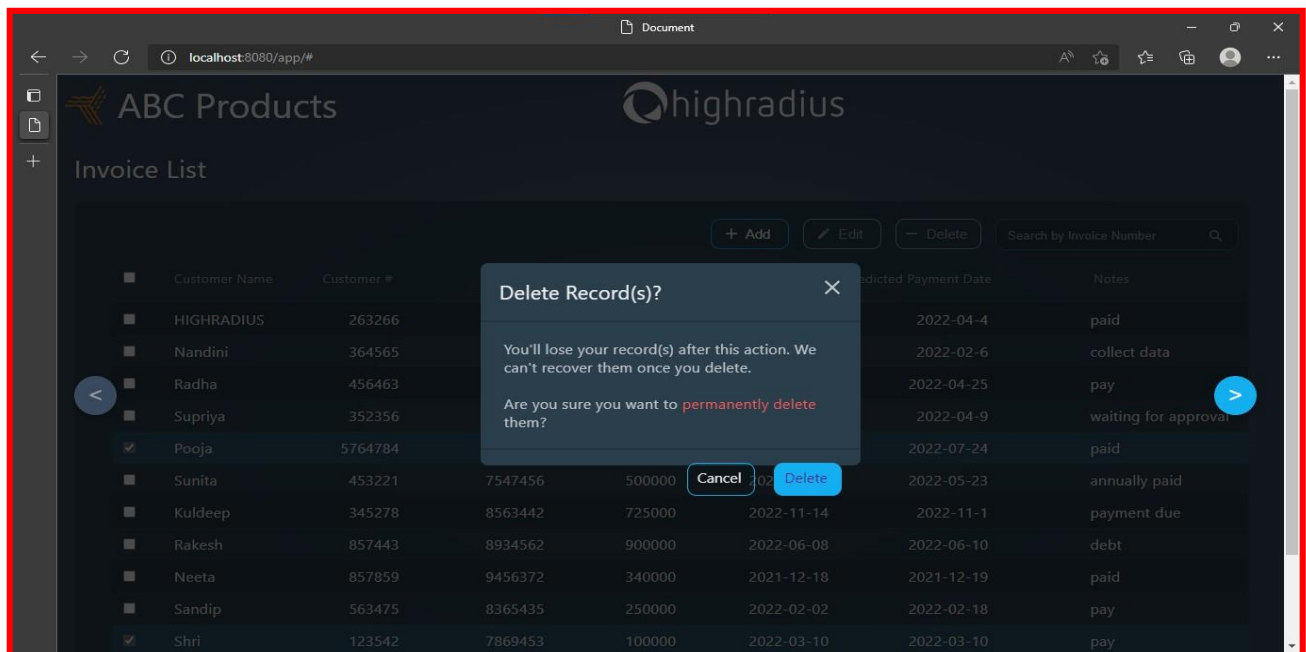


Figure 4.2.4

4.3 Programming Stage

While working with the dataset, we need to work with different lines of code which allows us to drop some columns, getting the data type, splitting data, data type conversion etc.

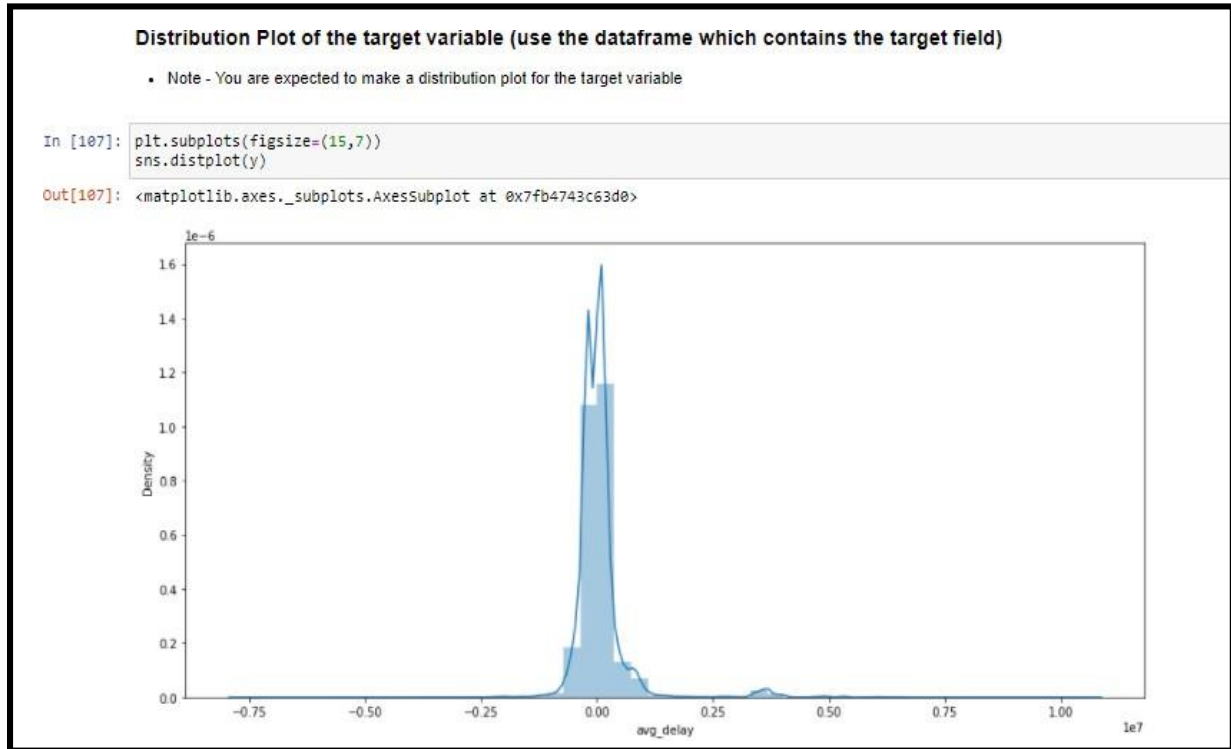


Figure 4.3.1

Display and describe the X_train dataframe

```
In [112]: X_train
```

```
Out[112]:
```

	business_code	cust_number	name_customer	buisness_year	doc_id	posting_date	due_in_date	baseline_create_date	cust_payment_terms	conv
29528	CA02	0140104249	SOB associates	2019.0	2.980521e+09	2019-01-08	2018-12-24	2018-12-14	CA10	1
412	U001	0200708844	WINC co	2019.0	1.928721e+09	2019-02-01	2018-12-29	2018-12-14	NAA8	
26350	U001	0200769623	WAL-MAR trust	2019.0	1.928543e+09	2018-12-30	2019-01-14	2018-12-30	NAH4	
15167	U001	0200728979	BJS corporation	2019.0	1.928540e+09	2018-12-30	2019-01-14	2018-12-30	NAA8	
37703	U001	0200769623	WAL-MAR in	2019.0	1.928543e+09	2018-12-30	2019-01-14	2018-12-30	NAH4	
...
17776	U001	0200729942	SA Illo	2019.0	1.929812e+09	2019-08-30	2019-09-14	2019-08-30	NAA8	
36823	U001	0200769623	WAL-MAR foundation	2019.0	1.929817e+09	2019-08-30	2019-09-14	2019-08-30	NAH4	
48532	U001	0200792293	UNIFIE foundation	2019.0	1.929816e+09	2019-08-30	2019-09-14	2019-08-30	NAA8	1
36043	U001	0200704858	WAKE us	2019.0	1.929809e+09	2019-08-30	2019-09-14	2019-08-30	NAA8	
32093	U001	0200708844	WINC associates	2019.0	1.929807e+09	2019-08-30	2019-09-14	2019-08-30	NAA8	

22995 rows x 10 columns

Figure 4.3.2

Out[141]: <matplotlib.axes._subplots.AxesSubplot at 0x7fb403110f10>

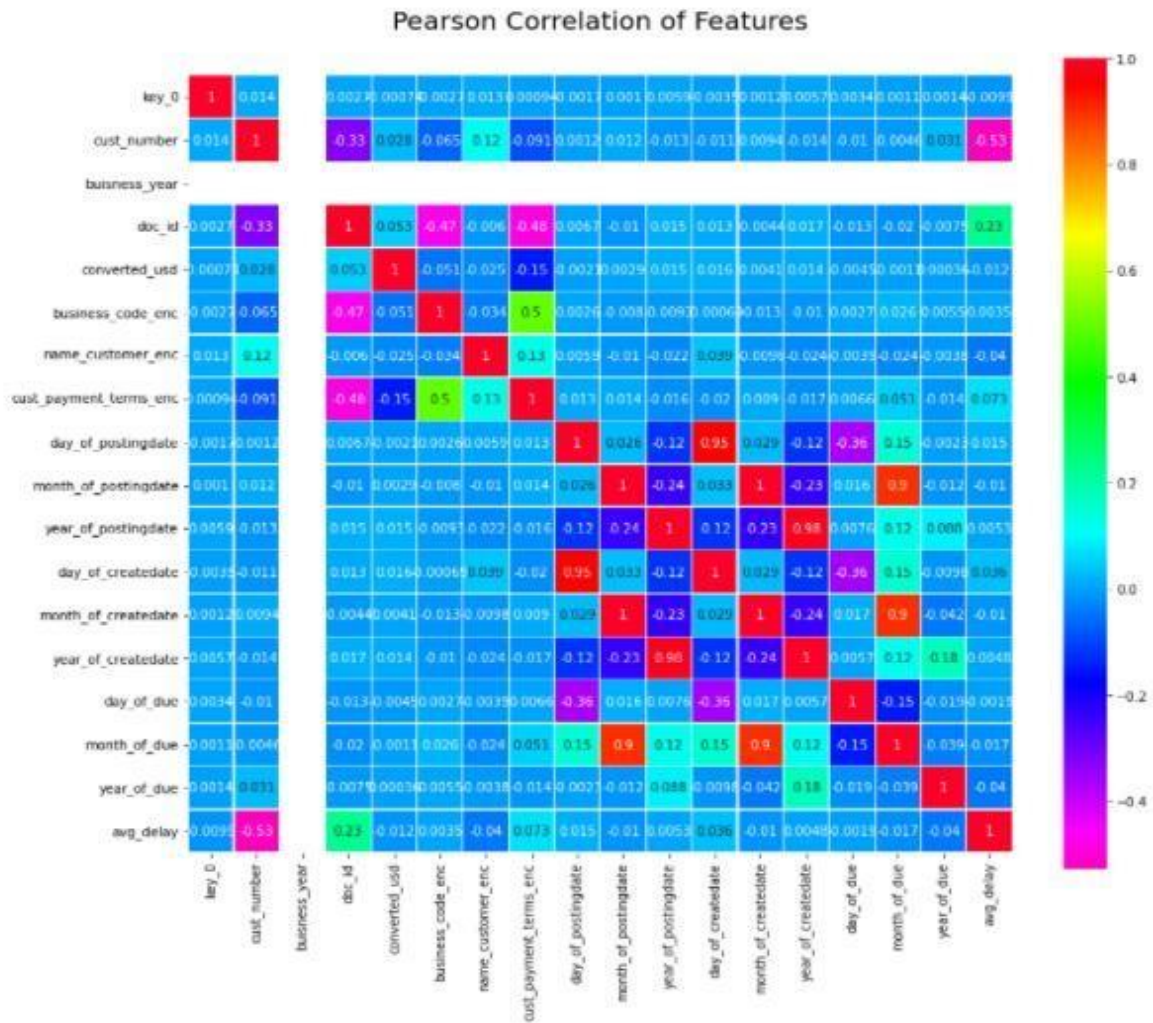


Figure 4.3.3

Following are the columns displayed in the UI:

Customer name
Customer id
Invoice number
Invoice amount
Due date
Predicted Payment date
Note

Table 4.3.4



Figure 4.3.5

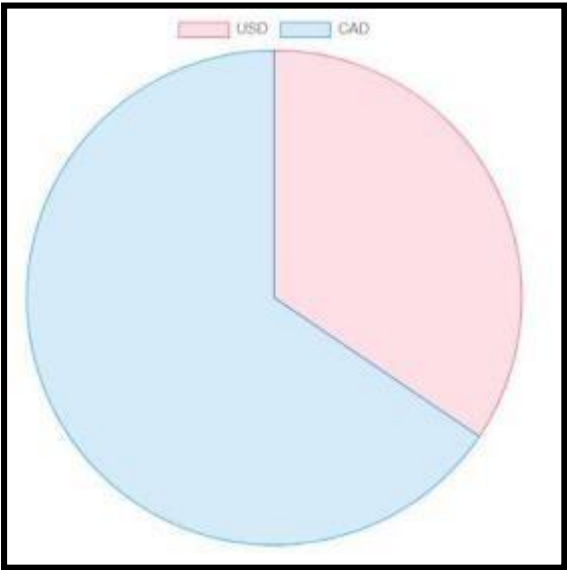


Figure 4.3.6

Following are the java files which contain the code for the application. I have used eclipse IDE for working with these files.

AddInvoice.java

```
import java.io.IOException;
import java.sql.Connection;
import java.sql.*;
import java.io.*;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 * Servlet implementation class AddInvoice
 */
@WebServlet("/AddInvoice")
public class AddInvoice extends HttpServlet {
    private static final long serialVersionUID = 1L;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public AddInvoice() {
        super();
        // TODO Auto-generated constructor stub
    }

    /**
     * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse
    response)
     */
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        // TODO Auto-generated method stub
        response.getWriter().append("Served at: ").append(request.getContextPath());
    }

    /**
     * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse
    response)
     */
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        // TODO Auto-generated method stub

        try {
            String customerName = request.getParameter("custName");
```

```

        String customerId = request.getParameter("custId");
        String invoiceId = request.getParameter("invNo");
        String invoiceAmt = request.getParameter("invAmt");
        String dueDate = request.getParameter("dueDa");
        String notes = request.getParameter("not");
        Connection con = DBConnection.createConnect();

        int no=((int) (Math.random()*(30 - 1))) + 1;
        String prirate=dueDate.substring(0,8)+no;
        //System.out.println("Post establishing a DB connection -=====> "
+prirate);

        String query = "INSERT INTO mytable (name_customer, cust_number,
invoice_id, total_open_amount, due_in_date,predicted_clear_date, notes) values (?, ?, ?, ?, ?,
?,?)";

        PreparedStatement st = con.prepareStatement(query);
        st.setString(1, customerName);
        st.setString(2, customerId);
        st.setString(3, invoiceId);
        st.setString(4, invoiceAmt);
        st.setString(5, dueDate);
        st.setString(6, prirate);
        st.setString(7, notes);
        int qq=st.executeUpdate();
        //System.out.println("qq "+qq);
        con.close();
        //System.out.println("Pendn - ");

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

```

DeleteInvoice.java

```

import java.io.IOException;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.io.*;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 * Servlet implementation class DeleteInvoice
 */

```

```

@WebServlet("/DeleteInvoice")
public class DeleteInvoice extends HttpServlet {
    private static final long serialVersionUID = 1L;
    public DeleteInvoice() {
        super();
        // TODO Auto-generated constructor stub
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        // TODO Auto-generated method stub
        response.getWriter().append("Served at: ").append(request.getContextPath());
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        // TODO Auto-generated method stub
        try {

            String fieldValue = request.getParameter("idList");
            int field = Integer.parseInt(fieldValue);

            Connection con = DBConnection.createConnect();
            String query = "DELETE FROM mytable WHERE FIELD1 = ?";

            PreparedStatement st = con.prepareStatement(query);
            st.setInt(1, field);
            st.executeUpdate();
            con.close();

        }

        catch(Exception e) {

        }

    }
}

```

EditInvoice.java

```

import java.io.IOException;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.io.*;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;

```



```

import javax.servlet.http.HttpServletResponse;

/**
 * Servlet implementation class EditInvoice
 */
@WebServlet("/EditInvoice")
public class EditInvoice extends HttpServlet {
    private static final long serialVersionUID = 1L;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public EditInvoice() {
        super();
        // TODO Auto-generated constructor stub
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        // TODO Auto-generated method stub
        response.getWriter().append("Served at: ").append(request.getContextPath());
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        // TODO Auto-generated method stub
        try {
            String fieldValue = request.getParameter("uniqId");
            int field = Integer.parseInt(fieldValue);
            String newInAmt = request.getParameter("inamt");
            float newInvoiceAmt = Float.parseFloat(newInAmt);
            String newNotes = request.getParameter("nn");

            Connection con = DBConnection.createConnect();
            String query = "UPDATE mytable SET total_open_amount = ?, notes =
? WHERE FIELD1 = ?";

            PreparedStatement st = con.prepareStatement(query);
            st.setFloat(1, newInvoiceAmt);
            st.setString(2, newNotes);
            st.setInt(3, field);

            st.executeUpdate();
            con.close();
        }

        catch(Exception e) {

        }}

```

SearchInvoice.java

```
import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import com.google.gson.Gson;
import com.google.gson.GsonBuilder;

@WebServlet("/SearchInvoice")
public class SearchInvoice extends HttpServlet {
    private static final long serialVersionUID = 1L;
    public SearchInvoice() {
        super();
        // TODO Auto-generated constructor stub
    }
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        // TODO Auto-generated method stub
        /*response.setContentType("application/json");
        response.setCharacterEncoding("UTF-8");

        PrintWriter out = response.getWriter();

        int rowCount = 12;*/

        try {
            Connection con = DBConnection.createConnect();

            String searchInvoice = request.getParameter("searchInvoice");
            String page = request.getParameter("page");
            Statement st = con.createStatement();
            String sql_statement = "SELECT * FROM mytable WHERE invoice_id="
+ ""+searchInvoice+""; // + "%' LIMIT " + page + "," + rowCount;
            ResultSet rs = st.executeQuery(sql_statement);
            ArrayList<Response> data = new ArrayList<>();
            while(rs.next()) {
                Response inv = new Response();
                inv.setCustomerName(rs.getString("name_customer"));
                inv.setCustomerId(rs.getString("cust_number"));
                inv.setInvoiceId(rs.getString("invoice_id"));
                inv.setDueDate(rs.getString("due_in_date"));
                inv.setInvoiceAmt(rs.getFloat("total_open_amount"));
                data.add(inv);
            }
        }
    }
}
```

```

    }
    Gson gson = new GsonBuilder().serializeNulls().create();
    String invoices = gson.toJson(data);
    response.setContentType("application/json");
    try {
        response.getWriter().write(invoices); //getWriter()
returns a PrintWriter object that can send character text to the client.
    }
    catch(IOException e)
    {
        e.printStackTrace();
    }
    rs.close();
    st.close();
    con.close();

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

protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    // TODO Auto-generated method stub
    doGet(request, response);
}

}

```

DBConnection.java

```

import java.sql.*;
import java.sql.SQLException;

public class DBConnection {
    public static Connection createConnect() {
        Connection con = null;
        String url = "jdbc:mysql://localhost:3306/hrcdb";
        String uname = "root";
        String pass = "08#raj";

        try {
            try {
                Class.forName("com.mysql.cj.jdbc.Driver");
            }
            catch (ClassNotFoundException e)
            {
                e.printStackTrace();
            }
            con = DriverManager.getConnection(url, uname, pass);
            System.out.println("Post establishing a DB connection - " +con);
        }
    }
}

```

```

    }
    catch(SQLException e)
    {
        System.out.println("Error Occurred");
        e.printStackTrace();
    }
    return con;
}
}

```

5. Conclusions and future work

Being a part of this summer internship and learning to build this project is in itself a great opportunity. I have learnt a lot from the classes that were being conducted by HighRadius and applied those concepts while building the project. There were ups and downs in the journey but such hurdles have motivated me to think and come up with a solution, as a result of which I was able to complete the project.

5.1 Platform used

For the implementation of this phase of the project I have used Jupyter Notebook and worked with python. I have also used excel in order to store data sets and data frames in .csv format. After the first phase I have used eclipse IDE to work with .java, .js (javaScript), .xml, HTML and CSS files. I found that eclipse is a good choice of platform as it provides us various options such as switching the workspace, using servlets of our own choice, importing and exporting files if required, etc.

5.2 Glossary

- **Invoice** - A document that is issued by a seller to a buyer when some goods are purchased. The fields which can be part of the invoice are defined below.
- **Advanced Search** - A pop-up window, which depicts the illustration that enables the user to search with single or multiple parameter values from the grid.
- **Predict** - The predict button is used as a tool to predict the Payment Date of each invoice.

- B2B - Business to Business
- B2C-Business to Consumer
- C2C - Consumer to Consumer
- **Payment Terms** - These indicate the period within which payments should be made and how. These terms are usually included in the invoices generated by companies and sent to customers.

6. Bibliography

[1] "Servlet Tutorial", javatpoint, 2019. [Online]. Available at :- <https://www.javatpoint.com/servlet-tutorial/>

[2] "Python Programming Language - GeeksforGeeks", *GeeksforGeeks*, [Online]. Available at :- <https://www.geeksforgeeks.org/python-programming-language/>

[3] "Java JDBC Tutorial – javatpoint", 2019. [Online]. Available at :- <https://www.javatpoint.com/java-jdbc/>

[4] "H2H Tech Track Project Requirement Specification", High Radius, 2022. [Online]. Available at :- <https://sites.google.com/view/h2htechtrackgroup2/prs?authuser=0/>

[5] "H2H Tech Track Reading Content", *High Radius*, 2022. [Online]. Available at :- <https://sites.google.com/view/h2htechtrackgroup2/reading-content/>