

**Mini Project On**

**Parchi**

**By**

**Akash Anil Dubey (2020510018)**

Under the guidance of  
**Internal Supervisor**

**Prof. Nikhita Mangaonkar**



Department of Master Of Computer Application  
Sardar Patel Institute of Technology  
Autonomous Institute Affiliated to Mumbai University  
2022-23

## **CERTIFICATE OF APPROVAL**

This is to certify that the following student

**Akash Anil Dubey (2020510018)**

Have satisfactorily carried out work on the project  
entitled

**“Parchi”**

Towards the fulfilment of project, as laid down  
by

Sardar Patel Institute of Technology  
during year  
2022-23.

Project Guide:  
Prof. Nikhita Mangaonkar

## PROJECT APPROVAL CERTIFICATE

This is to certify that the following students

**Akash Anil Dubey (2020510018)**

Have successfully completed the Project report on

**“Parchi”,**

which is found to be satisfactory and is approved

at

SARDAR PATEL INSTITUTE OF TECHNOLOGY,  
ANDHERI (W), MUMBAI

INTERNAL EXAMINER

EXTERNAL EXAMINER

HEAD OF DEPARTMENT

PRINCIPAL

# Contents

<b>Abstract</b>	<b>i</b>
<b>Objectives</b>	<b>i</b>
<b>List Of Figures</b>	<b>ii</b>
<b>List Of Tables</b>	<b>ii</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Problem Definition . . . . .	1
1.2 Objectives and Scope . . . . .	1
1.2.1 Objectives . . . . .	1
1.2.2 Scope . . . . .	1
1.3 Existing System . . . . .	2
1.4 Proposed System . . . . .	2
1.5 System Requirements . . . . .	3
<b>2 Software Requirement Specification (SRS) and Design</b>	<b>4</b>
2.1 Purpose . . . . .	4
2.2 Definition . . . . .	4
2.3 Overall Description . . . . .	4
2.3.1 Product Functions . . . . .	4
2.4 Modules . . . . .	5
2.4.1 Activity diagram . . . . .	5
2.4.2 Communication Design . . . . .	6
2.4.3 Work Breakdown Structure . . . . .	6
2.4.4 Use-Case . . . . .	7
<b>3 Project Implementation and Testing</b>	<b>9</b>
3.1 Dashboard . . . . .	9
3.2 Home - Add Company . . . . .	10
3.3 Add Client . . . . .	11
3.4 Add Products . . . . .	12
3.5 Invoices . . . . .	13
3.6 Dashboard . . . . .	14
3.7 Invoices . . . . .	16
3.8 Add Client . . . . .	19
3.9 Add Product . . . . .	26
3.10 Add Product . . . . .	32
<b>4 Test Cases</b>	<b>33</b>
<b>5 Limitations</b>	<b>34</b>
<b>6 Future Enhancements</b>	<b>34</b>

<b>7</b>	<b>User Manual</b>	<b>35</b>
<b>8</b>	<b>Bibliography</b>	<b>36</b>
8.1	Web References . . . . .	36

## Abstract

Our Parchi app is designed for any web based device which supports an active internet browser. It will help any individual and will convert the invoice generation digitally.

The User can view and keep track of added invoices and can also edit them. First step is to add customers and clients, along with products. For generating invoice we need to also update the product quantity.

The app will provide vendors the required accurate user invoices for GST and tax clearance.

## Objectives

The web based Application "Parchi" is used

- To provide vendors a user friendly platform for the tax clearance and invoice generation process.
- To add a layer of digital storing and generation of billings for easier stock update and sales statistics.
- To carefully view and update list of customers, products, and respective sales.

## List of Figures

2.4.1Activity Diagram . . . . .	5
2.4.2Communication Diagram . . . . .	6
2.4.3Work Breakdown Structure . . . . .	6
2.4.4Use-Case Diagram . . . . .	7
3.1.1 Dashboard . . . . .	9
3.2.1 Home - Add Company . . . . .	10
3.3.1Add Client . . . . .	11
3.4.1Add Products . . . . .	12
3.5.1Invoices . . . . .	13

## List of Tables

1.5.2 Device Requirements on Client Side . . . . .	3
4.2.1 Use Case Table - Create Profile . . . . .	8
4.2.2 Use Case Table - Add Client . . . . .	8
4.2.3 Use Case Table - Add Products . . . . .	8
4.2.4 Use Case Table - Create Bill . . . . .	8
6.1 Test Case . . . . .	33

# 1 Introduction

## 1.1 Problem Definition

To eliminate redundancy in collecting data for the complete vendor related process of generating invoices and sales billings, and to squash the time gap and miscommunication in the same process.

## 1.2 Objectives and Scope

### 1.2.1 Objectives

The web based application "Parchi" is

- To provide vendors a user friendly platform for the tax clearance and invoice generation process.
- To add a layer of digital storing and generation of billings for easier stock update and sales statistics.
- To carefully view and update list of customers, products, and respective sales.

### 1.2.2 Scope

The respective vendor must provide their details related to the company name, address and contact.

Each vendor has the option of adding 3 different data sets: Client, Company and product. The added details are visible on a virtual dashboard.

Total balance, invoices, products, clients can be stored on the application. The application will save all the data and can be accessed at a later stage on the same browser, as the application stores the data in reference to the browser on the system.



### 1.3 Existing System

Existing system if applicable are not available in a server-less architecture as our defined product. Some of the disadvantages of existing system are as follows :

- Manual record maintenance  
vendors use traditional methods and have to maintain several excel sheets regarding the various available data.
- Absence of multiple factors  
Our project provides features of digital invoice generation, customer and sales records and helps in tax filing of the vendor.

### 1.4 Proposed System

The primary user of the application is the vendor. They can use the application on their devices and their data would be stored on the application which is only accessible to them, the vendor, through their own system, and the local browser.

There is a feature to clear all the data of the application on a single click. This feature enables resetting the whole application data.

Some of the advantages of our system are as follows :

- User friendly interface  
It provides attractive interface to the user for navigation through a dynamic flow of invoice generation process in the app.  
The items of the application are already connected to each other using side navigation bar.
- All the customers and products in one place  
The vendors can view their added data in the dashboard.

## 1.5 System Requirements

- Hardware Requirements on Client Side

Table 1.5.2: Device Requirements on Client Side

Device	Any device with a supporting internet browser.
--------	--

## **2 Software Requirement Specification (SRS) and Design**

### **2.1 Purpose**

The purpose of our project is to develop a web application that can help vendors and other businesses to carefully segregate all of their relevant client side data and also generate invoices for the sales. The billings will help in tax returns and all of these features are accessible through a server-less architecture from any Internet browser.

### **2.2 Definition**

To build a web Application so the vendors can simplify their tax clearance and invoice generation process.

### **2.3 Overall Description**

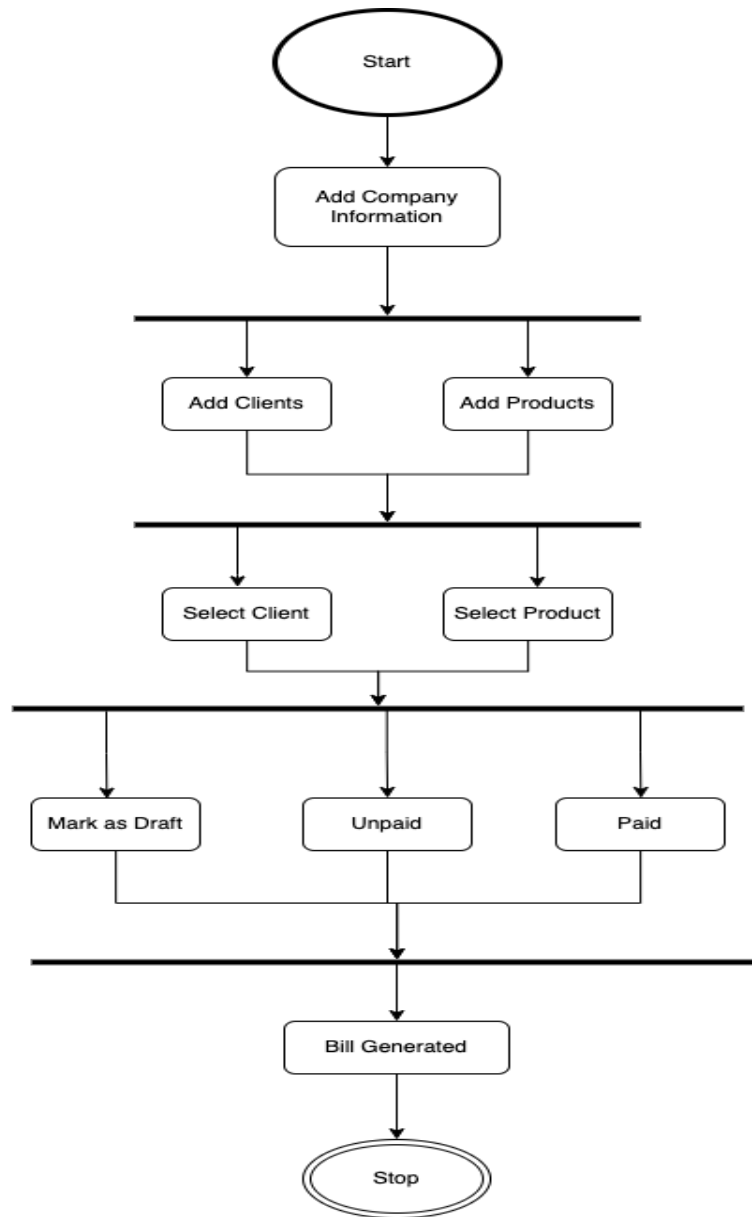
#### **2.3.1 Product Functions**

The product function includes:

1. Authentication: Users would be the vendors, and they can access the application from their browser, which would be saved for future login sessions. There is no external user verification required.
2. Dashboard: This dashboard will have all the added details of Clients, sales and invoices, displayed in a summarized format.
3. Generate bills: Edit bills any time and it can be marked or tagged as paid, or kept in draft.
4. Add sales quantity: This part specifies the sales quantity of any added product.
5. Add Product: It is required to add a new product.
6. Add Client: Add a new client as desired. It requires the name and contact details of the client company.

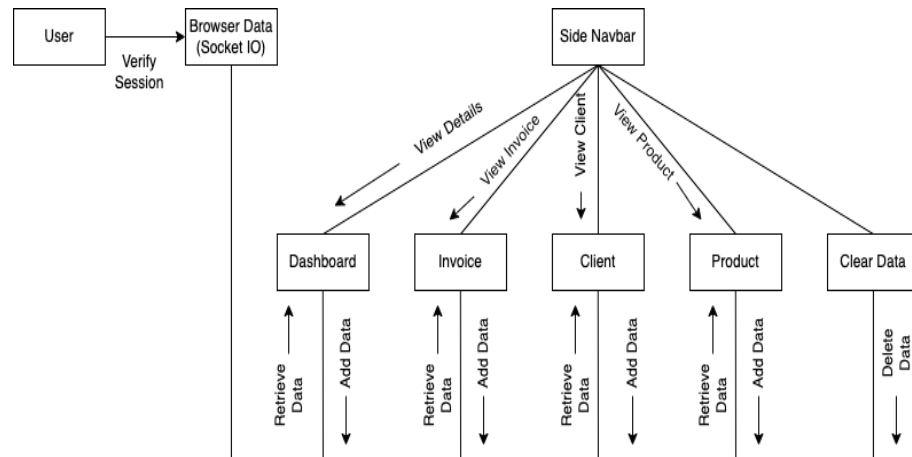
## 2.4 Modules

### 2.4.1 Activity diagram



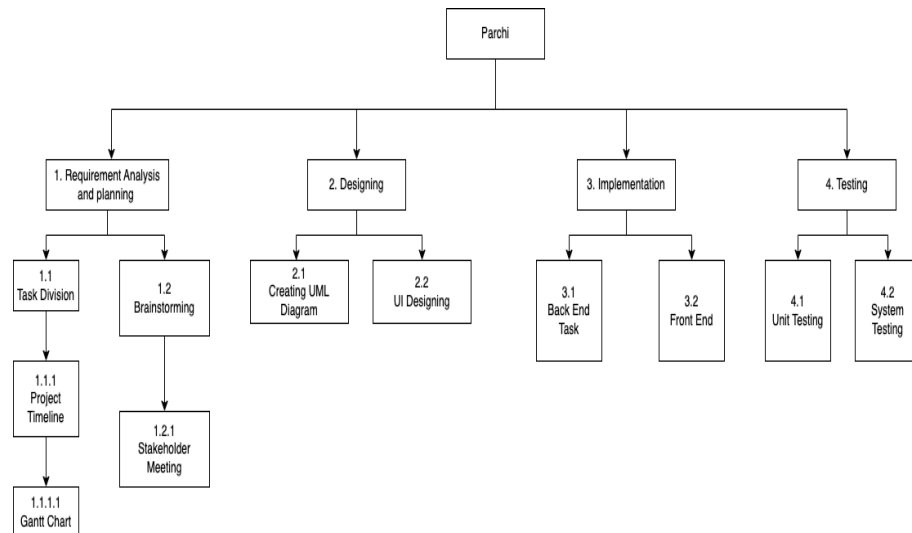
2.4.1: Activity Diagram

### 2.4.2 Communication Design



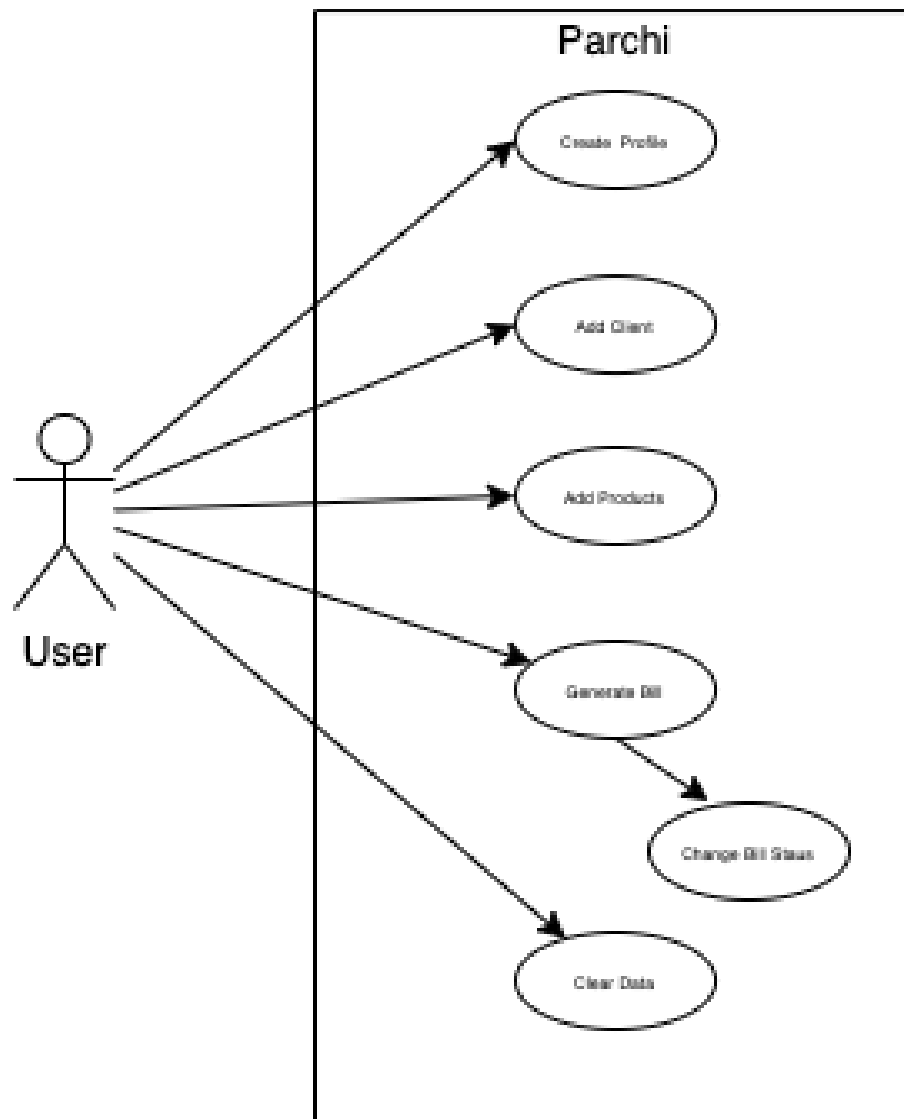
2.4.2: Communication Diagram

### 2.4.3 Work Breakdown Structure



2.4.3: Work Breakdown Structure

## 2.4.4 Use-Case



2.4.4: Use-Case Diagram

Use Cases:

1. Create Profile
2. Add Client
3. Add Products
4. Create Bill

Table 4.2.1: Use Case Table - Create Profile

Use Case ID	1
Use Case Name	Create Profile
Actor	User
Pre-Condition	They must create profile first
Post-Condition	User can edit the profile
Flow of events	Edit company information, update profile and view Billing Data

Table 4.2.2: Use Case Table - Add Client

Use Case ID	2
Use Case Name	Add Client
Actor	User
Pre-Condition	They must create profile first
Post-Condition	User can view or add new clients using their few basic information

Table 4.2.3: Use Case Table - Add Products

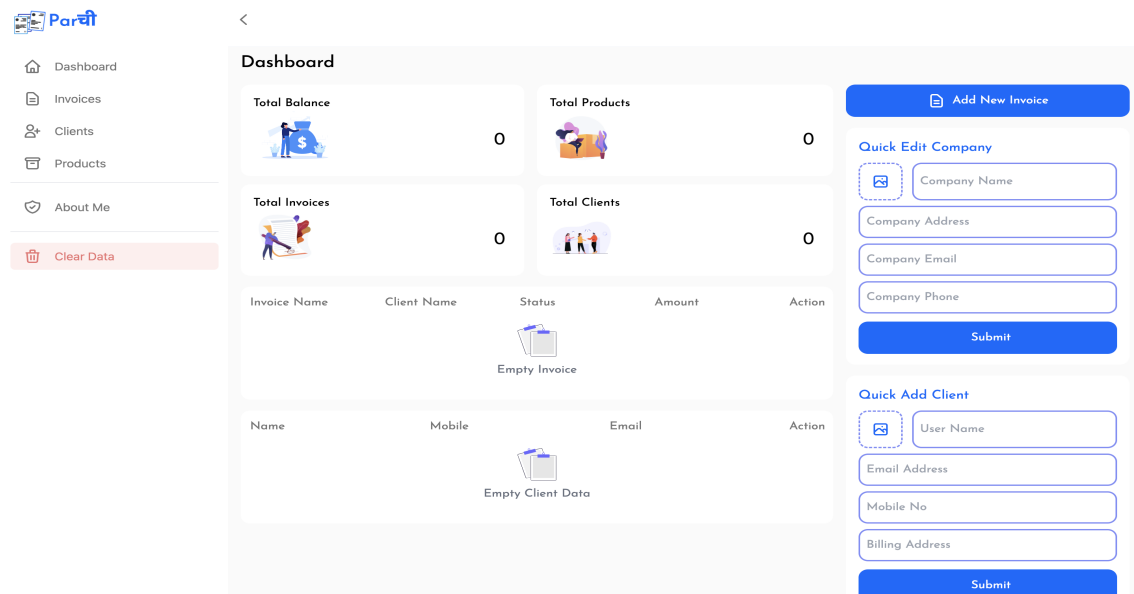
Use Case ID	3
Use Case Name	Add Products
Actor	User
Pre-Condition	Login
Post-Condition	User can add products

Table 4.2.4: Use Case Table - Create Bill

Use Case ID	4
Use Case Name	Create Bill
Actor	User
Pre-Condition	Login
Post-Condition	User can create bill

## 3 Project Implementation and Testing


### 3.1 Dashboard



3.1.1: Dashboard



### 3.2 Home - Add Company

 Parchi

Google

Dashboard

Invoices

Clients

Products

About Me

Clear Data

Dashboard

Total Balance0

Total Products0

Total Invoices0

Total Clients0

Invoice Name	Client Name	Status	Amount	Action
Empty Invoice				

Name	Mobile	Email	Action
Empty Client Data			

Add New Invoice

Quick Edit Company

Google

Mumbai

akash.dubey@spit.ac.in

7741830638

Submit

Quick Add Client

User Name

Email Address

Mobile No

Billing Address

Submit

#### 3.2.1: Home - Add Company

Page 10

### 3.3 Add Client

The screenshot displays the 'Add Client' form in the Parchi application. The interface is divided into several sections:

- Left Sidebar:** Contains navigation links for Dashboard, Invoices, Clients (highlighted), Products, About Me, and a Clear Data button.
- Top Bar:** Shows the Parchi logo and a back arrow.
- Main Content Area:**
  - Advanced Search:** Includes input fields for User Name, User Email, and Mobile Number.
  - Quick Add Client:** Includes input fields for User Name, Email Address, Mobile No, and Billing Address, followed by a Submit button.
  - Table:** A table with columns for Name, Mobile, Email, and Action. The table is currently empty, displaying a message 'Empty Client Data' with a folder icon.

A small inset image in the bottom right corner shows a preview of the application's mobile view.

#### 3.3.1: Add Client

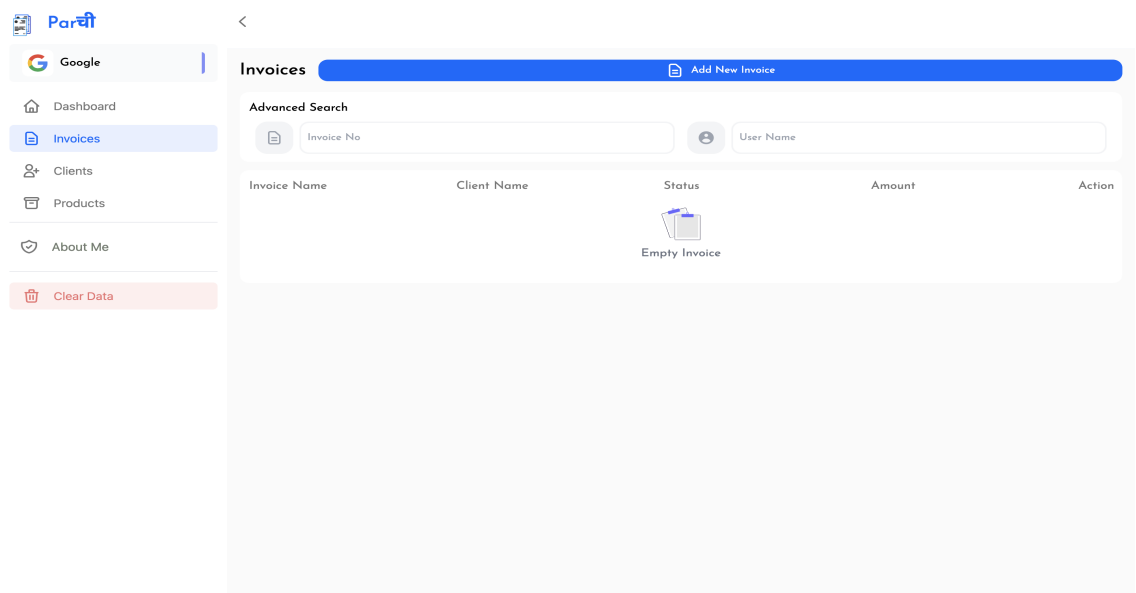
### 3.4 Add Products

The screenshot displays the 'Add Products' form in the Parchi application. The interface is divided into three main sections:

- Sidebar:** Contains navigation links for Dashboard, Invoices, Clients, Products (highlighted), About Me, and a Clear Data button.
- Main Content Area:**
  - Products Header:** Includes an Advanced Search bar with fields for Product ID and Product Name.
  - Table:** A table with columns ProductID, Name, Amount, and Action. The table is currently empty, displaying an 'Empty Data' message.
- Quick Add Product Sidebar:** A sidebar on the right for adding new products, featuring fields for Product ID, Product Name, and Product Amount, along with a Submit button.

3.4.1: Add Products

### 3.5 Invoices



#### 3.5.1: Invoices

### 3.6 Dashboard

```

import React from "react";
import LottieMoney from "../LotiIcon/LottieMoney";
import LottieProduct from "../LotiIcon/LottieProduct";
import LottieInvoice from "../LotiIcon/LottieInvoice";
import LottiePersons from "../LotiIcon/LottiePersons";
import { useSelector } from "react-redux";
import { getAllClientsSelector } from "../../store/clientSlice";
import { getAllProductSelector } from "../../store/productSlice";
import {
  getAllInvoiceSelector,
  getTotalBalance,
} from "../../store/invoiceSlice";
import NumberFormat from "react-number-format";

function DashboardWidgets() {
  const clients = useSelector(getAllClientsSelector);
  const products = useSelector(getAllProductSelector);
  const totalBalance = useSelector(getTotalBalance);
  const allInvoices = useSelector(getAllInvoiceSelector);

  return (
    <>
      <div className="flex flex-wrap">
        <div className="w-full mb-3 md:w-1/2">
          <div className="p-4 bg-white rounded-xl md:mr-4 hover:shadow-sm">
            <div className="font-title">Total Balance</div>
            <div className="flex justify-between items-center">
              <div className="h-30">
                <LottieMoney loop className="h-20"/>
              </div>
              <div className="h-30">
                <div className="text-2xl mr-2">
                  <NumberFormat
                    value={totalBalance}
                    className=""
                    displayType={"text"}
                    thousandSeparator={true}
                    renderText={(value, props) => <span {...props}>{value}</span>}
                  />
                </div>
              </div>
            </div>
          </div>
        </div>
      </div>
    </>
  );
}

```

```

</div>
<div className="w-full mb-3 md:w-1/2">
  <div className="p-4 bg-white rounded-xl hover:shadow-sm">
    <div className="font-title">Total Products</div>
    <div className="flex justify-between items-center">
      {*_Icon*_}
      <div className="h-30">
        <LottieProductLoop className="h-20"/>
      </div>
      {*_IconFinished*_}
      <div className="text-2xl mr-2">
        <NumberFormat
          value={products?.length}
          className=""
          displayType={"text"}
          thousandSeparator={true}
          renderText={(value, _props) => <span {..._props}>{value}</span>}
        />
      </div>
    </div>
  </div>
</div>
</div>
<div className="w-full mb-3 md:w-1/2">
  <div className="p-4 bg-white rounded-xl md:mr-4 hover:shadow-sm">
    <div className="font-title">Total Invoices</div>
    <div className="flex justify-between items-center">
      {*_Icon*_}
      <div className="h-30">
        <LottieInvoiceLoop className="h-20"/>
      </div>
      {*_IconFinished*_}
      <div className="text-2xl mr-2">
        <NumberFormat
          value={allInvoices?.length}
          className=""
          displayType={"text"}
          thousandSeparator={true}
          renderText={(value, _props) => <span {..._props}>{value}</span>}
        />
      </div>
    </div>
  </div>
</div>
</div>
<div className="w-full mb-3 md:w-1/2">
  <div className="p-4 bg-white rounded-xl hover:shadow-sm">
    <div className="font-title">Total Clients</div>

```

```

    <div className="flex justify-between items-center">
    { /* Icon */ }
    <div className="h-30">
    <LottiePersons loop className="h-20" />
    </div>
    { /* Icon Finished */ }
    <div className="text-2xl mr-2">
    <NumberFormat
    value={clients?.length}
    className=""
    displayType="text"
    thousandSeparator={true}
    renderText={(value, props) => <span {...props}>{value}</span>}
    />
    </div>
    </div>
    </div>
    </div>
    </div>
    </div>
    </div>
  </>
  );
}

```

```
export default DashboardWidgets;
```

### 3.7 Invoices

```

import React, { useCallback, useState, useEffect } from "react";
import { motion } from "framer-motion";
import { toast } from "react-toastify";
import { useSelector, useDispatch } from "react-redux";

import {
  getIsInvoiceConfirmModal,
  setConfirmModalOpen,
  setIsConfirm,
} from "../store/invoiceSlice";
import CheckCircleIcon from "../Icons/CheckCircleIcon";

function InvoiceConfirmModal(props) {
  const dispatch = useDispatch();
  const isOpenConfirmModal = useSelector(getIsInvoiceConfirmModal);
  const [animate, setAnimate] = useState(true);

```

```

    const onConfirmModal = useCallback(() => {
      dispatch(setIsConfirm(true));
      dispatch(setConfirmModalOpen(false));
      toast.success("Successfully Action", {
        position: "bottom-center",
        autoClose: 2000,
      });
    }, [dispatch]);

    const onCancelHandler = useCallback(() => {
      dispatch(setConfirmModalOpen(false));
      dispatch(setIsConfirm(false));
    }, [dispatch]);

    useEffect(() => {
      if (isOpenConfirmModal !== false) {
        setAnimate(true);
      } else {
        setAnimate(false);
      }
    }, [isOpenConfirmModal]);

    return isOpenConfirmModal !== false ? (
      <motion.div
        className="modal-container"
        aria-labelledby="modal-title"
        role="dialog"
        aria-modal="true"
        initial={{
          opacity: 0,
        }}
        animate={{
          opacity: animate ? 1 : 0,
        }}
        transition={{
          type: "spring",
          damping: 18,
        }}
      >
        <div className="relative">
          <div className="fixed inset-0 bg-gray-500 bg-opacity-75 transition-opacity"></div>

          <div className="fixed z-10 inset-0 overflow-y-auto">
            <div className="flex items-end sm:items-center justify-center min-h-full p-4 text-
            <div className="relative bg-white rounded-lg text-left overflow-hidden shadow-xl

```



```
export default InvoiceConfirmModal;
```

### 3.8 Add Client

```
import React, { useCallback, useEffect, useMemo, useState } from "react";
import { useSelector, useDispatch } from "react-redux";
import { motion } from "framer-motion";
import {
  getIsOpenClientSelector,
  getAllClientsSelector,
  setClientSelector,
  setOpenClientSelector,
} from "../../store/clientSlice";
import {
  defaultTdStyle,
  defaultTdActionStyle,
  defaultTdWrapperStyle,
  defaultTdContent,
  defaultTdContentTitleStyle,
  defaultSearchStyle,
} from "../../constants/defaultStyles";
import ReactPaginate from "react-paginate";
import Button from "../../Button/Button";

// Example items, to simulate fetching from another resources.
const itemsPerPage = 6;
const emptySearchForm = {
  name: "",
  email: "",
  mobileNo: "",
};

function ClientChooseModal() {
  const dispatch = useDispatch();
  const allClients = useSelector(getAllClientsSelector);
  const openModal = useSelector(getIsOpenClientSelector);

  const [animate, setAnimate] = useState(true);
  const [searchForm, setSearchForm] = useState(emptySearchForm);
  const [currentItems, setCurrentItems] = useState(null);
  const [pageCount, setPageCount] = useState(0);
  const [itemOffset, setItemOffset] = useState(0);

  const clients = useMemo(() => {
```

```

        let filterData = allClients.length > 0 ? [...allClients].reverse() : [];
        if (searchForm.name?.trim()) {
            filterData = filterData.filter((client) =>
                client.name.includes(searchForm.name)
            );
        }

        if (searchForm.email?.trim()) {
            filterData = filterData.filter((client) =>
                client.email.includes(searchForm.email)
            );
        }

        if (searchForm.mobileNo?.trim()) {
            filterData = filterData.filter((client) =>
                client.mobileNo.includes(searchForm.mobileNo)
            );
        }

        return filterData;
    }, [allClients, searchForm]);

    // Invoke when user click to request another page.
    const handleClick = (event) => {
        const newOffset = (event.selected * itemsPerPage) % clients.length;
        setItemOffset(newOffset);
    };

    const handlerSearchValue = useCallback((event, keyName) => {
        const value = event.target.value;

        setSearchForm((prev) => {
            return { ...prev, [keyName]: value };
        });

        setItemOffset(0);
    }, []);

    const onCancelHandler = useCallback(() => {
        dispatch(setOpenClientSelector(false));
    }, [dispatch]);

    const handleSelect = useCallback(
        (item) => {
            dispatch(setClientSelector(item.id));
        }
    );

```

```

        setTimeout(() => {
            dispatch(setOpenClientSelector(false));
        }, 50);
    },
    [dispatch]
);

    useEffect(() => {
        const endOffset = itemOffset + itemsPerPage;
        setCurrentItems(clients.slice(itemOffset, endOffset));
        setPageCount(Math.ceil(clients.length / itemsPerPage));
    }, [clients, itemOffset]);

    useEffect(() => {
        if (openModal) {
            setAnimate(true);
        } else {
            setAnimate(false);
        }
    }, [clients, openModal]);

    return openModal ? (
        <motion.div
            className="modal-container"
            aria-labelledby="modal-title"
            role="dialog"
            aria-modal="true"
            initial={{
                opacity: 0,
            }}
            animate={{
                opacity: animate ? 1 : 0,
            }}
            transition={{
                type: "spring",
                damping: 18,
            }}
        >
            <div className="relative">
                <div className="fixed inset-0 bg-gray-500 bg-opacity-75 transition-opacity"></div>

                <div className="fixed z-10 inset-0 overflow-y-auto">
                    <div className="flex justify-center min-h-full p-4 text-center">
                        <div className="relative bg-white rounded-lg text-left overflow-hidden shadow-xl"
                            <div className="bg-white px-4 pt-5 pb-4 sm:p-6 sm:pb-4 flex-1">
                                <div className="rounded-xl px-3 py-3 mb-3">

```

```

<div className="font-title_mb-2">AdvancedSearch</div>
<div className="flex_w-full flex-col_sm:flex-row">
  <div className="mb-2_sm:mb-0_sm:text-left_text-default-color flex flex-r
  <div className="h-12_w-12 rounded-2xl_bg-gray-100_mr-2 flex justify-ce
  <svg
    xmlns="http://www.w3.org/2000/svg"
    className="h-6_w-6 text-gray-400"
    viewBox="0 0 20 20"
    fill="currentColor"
  >
    <path
      fillRule="evenodd"
      d="M18 10a8 8 0 1 1-16 0 8 8 0 0 1 16 0zm-6-3a2 2 0 1 1-4 0 2 2 0 0 1
      clipRule="evenodd"
    >/>
  </svg>
</div>
<input
  autoComplete="nope"
  value={searchForm.name}
  placeholder="User Name"
  className={defaultSearchStyle}
  onChange={(e) => handlerSearchValue(e, "name")}
/>
</div>
<div className="mb-2_sm:mb-0_sm:text-left_text-default-color flex flex-r
  <div className="h-12_w-12 rounded-2xl_bg-gray-100_mr-2 flex justify-ce
  <svg
    xmlns="http://www.w3.org/2000/svg"
    className="h-6_w-6 text-gray-400"
    fill="none"
    viewBox="0 0 24 24"
    stroke="currentColor"
    strokeWidth={2}
  >
    <path
      strokeLinecap="round"
      strokeLinejoin="round"
      d="M3 8l7.89 5.26a2 2 0 0 2 .22 0L21 8M5 19h14a2 2 0 0 2-2V7a2 2
    >/>
  </svg>
</div>
<input
  autoComplete="nope"
  value={searchForm.email}
  placeholder="User Email"

```

```

        className={defaultSearchStyle}
        onChange={(e) => handlerSearchValue(e, "email")}
    />
</div>
<div className="mb-2 sm:mb-0 sm:text-left text-default-color flex flex-1"
    >
    <div className="h-12 w-12 rounded-2xl bg-gray-100 mr-2 flex justify-c
    >
    <svg
    xmlns="http://www.w3.org/2000/svg"
    className="h-6 w-6 text-gray-400"
    fill="none"
    viewBox="0 0 24 24"
    stroke="currentColor"
    strokeWidth={2}
    >
    <path
    strokeLinecap="round"
    strokeLinejoin="round"
    d="M12 18h.01M8 21h8a2 2 0 0 0 2-2H8a2 2 0 0 0 2-2v2"
    />
    </svg>
    </div>
    <input
    autoComplete="nope"
    value={searchForm.mobileNo}
    placeholder="Mobile Number"
    className={defaultSearchStyle}
    onChange={(e) => handlerSearchValue(e, "mobileNo")}
    />
    </div>
</div>
</div>
</div>

<div className="sm:bg-white rounded-xl sm:px-3 sm:py-3">
    <div className="hidden sm:flex invisible sm:visible w-full flex-col sm:fl
    <div className="sm:text-left text-default-color font-title flex-1">
        Name
    </div>
    <div className="sm:text-left text-default-color font-title flex-1">
        Mobile
    </div>
    <div className="sm:text-left text-default-color font-title flex-1">
        Email
    </div>
    <div className="sm:text-left text-default-color font-title sm:w-11">
        Action
    </div>

```

```

</div>

<div>
    {currentItems}&&
    currentItems.map((client) => (
        <div className={defaultTdWrapperStyle} key={client.id}>
            <div className={defaultTdStyle}>
                <div className={defaultTdContentTitleStyle}>
                    Name
                </div>
                <div className={defaultTdContent}>
                    {client.image ? (
                        <img
                            className="object-cover h-10 w-10 rounded-2xl"
                            src={client.image}
                            alt={client.name}
                        />
                    ) : (
                        <span className="h-10 w-10 rounded-2xl bg-gray-100 flex justify-center items-center">
                            <svg
                                xmlns="http://www.w3.org/2000/svg"
                                className="h-6 w-6 text-gray-400"
                                viewBox="0 0 20 20"
                                fill="currentColor"
                            >
                                <path
                                    fillRule="evenodd"
                                    d="M18 10a8 8 0 11-16 0 8 8 0 0116 0zm-6-3a2 2 0 11-4 0 2 2 0 014 0z"
                                    clipRule="evenodd"
                                />
                            </svg>
                        </span>
                    )}
                </div>
            </div>
        </div>
    )
    <div className={defaultTdStyle}>
        <div className={defaultTdContentTitleStyle}>
            Mobile
        </div>
        <div className={defaultTdContent}>
            <span className="whitespace-nowrap text-ellipsis overflow-hidden">
                {client.mobileNo}
            </span>
        </div>
    </div>

```

```

</span>
</div>
</div>
<div className={defaultTdStyle}>
  <div className={defaultTdContentTitleStyle}>
    Email
  </div>
  <div className={defaultTdContent}>
    <span className="whitespace-nowrap text-ellipsis overflow-hidden">
      {client.email}{" "}
    </span>
  </div>
</div>
<div className={defaultTdActionStyle}>
  <div className={defaultTdContentTitleStyle}>
    Action
  </div>
  <div className={defaultTdContent}>
    <Button
      size="sm"
      block={1}
      onClick={() => handleSelect(client)}
    >
      Select
    </Button>
  </div>
</div>
</div>
))}

{clients.length > 0 && (
  <ReactPaginate
    className="inline-flex items-center space-x-px mt-2"
    previousLinkClassName="py-2 px-3 ml-0 leading-tight text-gray-500 bg-white"
    nextLinkClassName="py-2 px-3 ml-0 leading-tight text-gray-500 bg-white"
    pageLinkClassName="py-2 px-3 ml-0 leading-tight text-gray-500 bg-white"
    breakLinkClassName="py-2 px-3 ml-0 leading-tight text-gray-500 bg-white"
    activeLinkClassName="py-2 px-3 text-blue-600 bg-blue-50 border border-gray-200"
    breakLabel="..."
    onPageChange={handlePageClick}
    pageRangeDisplayed={1}
    pageCount={pageCount}
    previousLabel="<"
    nextLabel=">"
    renderOnZeroPageCount={null}
  />
)}

```



### 3.9 Add Product

Page 26

```

    }_from_ "../store/productSlice";
    import_ ProductIDIcon_ from_ "../Icons/ProductIDIcon";
    import_ ProductIcon_ from_ "../Icons/ProductIcon";

    //_Example_items,_to_simulate_fetching_from_another_resources.
    const_ itemsPerPage_ =_ 6;
    const_ emptySearchForm_ =_{
      _name:_ "",
      _email:_ "",
      _mobileNo:_ "",
    };

    function_ ProductChoosenModal()_{
      _const_ dispatch_ =_ useDispatch();
      _const_ allProducts_ =_ useSelector(getAllProductSelector);
      _const_ openModal_ =_ useSelector(getIsOpenProductSelector);

      _const_ [animate,_ setAnimate]_ =_ useState(true);
      _const_ [searchForm,_ setSearchForm]_ =_ useState(emptySearchForm);
      _const_ [currentItems,_ setCurrentItems]_ =_ useState(null);
      _const_ [pageCount,_ setPageCount]_ =_ useState(0);
      _const_ [itemOffset,_ setItemOffset]_ =_ useState(0);

      _const_ products_ =_ useMemo(()_ =>_{
        _let_ filterData_ =_ allProducts.length_ >_ 0_ ?_ [...allProducts].reverse()_ :_ [];
        _if_(searchForm.name?.trim())_{
          _filterData_ =_ filterData.filter((product)_ =>
            _product.name.includes(searchForm.name)
          );
        }

        _if_(searchForm.productID?.trim())_{
          _filterData_ =_ filterData.filter((product)_ =>
            _product.productID.includes(searchForm.productID)
          );
        }

        _return_ filterData;
      },[_allProducts,_ searchForm]);

      //_Invoke_when_user_click_to_request_another_page.
      _const_ handleClick_ =_(event)_ =>_{
        _const_ newOffset_ =_(event.selected*_itemsPerPage)%_products.length;
        _setItemOffset(newOffset);
      };

```

```
const handleSelect = useCallback(
  (item) => {
    dispatch(setProductSelector(item.id));

    setTimeout(() => {
      dispatch(setOpenProductSelector(false));
    }, 50);
  },
  [dispatch]
);

const onCancelHandler = useCallback(() => {
  dispatch(setOpenProductSelector(false));
}, [dispatch]);

const handlerSearchValue = useCallback((event, keyName) => {
  const value = event.target.value;

  setSearchForm((prev) => {
    return { ...prev, [keyName]: value };
  });

  setItemOffset(0);
}, []);

useEffect(() => {
  // Fetch items from another resources.
  const endOffset = itemOffset + itemsPerPage;
  setCurrentItems(products.slice(itemOffset, endOffset));
  setPageCount(Math.ceil(products.length / itemsPerPage));
}, [products, itemOffset]);

useEffect(() => {
  if (openModal) {
    setAnimate(true);
  } else {
    setAnimate(false);
  }
}, [products, openModal]);

return openModal ? (
  <motion.div
    className="modal-container"
    aria-labelledby="modal-title"
    role="dialog"
    aria-modal="true"
```

```

        initial={{
            opacity: 0,
        }}
        animate={{
            opacity: animate ? 1 : 0,
        }}
        transition={{
            type: "spring",
            damping: 18,
        }}
    >
    <div className="relative">
        <div className="fixed inset-0 bg-gray-500 bg-opacity-75 transition-opacity"></div>

        <div className="fixed z-10 inset-0 overflow-y-auto">
            <div className="flex justify-center min-h-full p-4 text-center">
                <div className="relative bg-white rounded-lg text-left overflow-hidden shadow-xl"
                <div className="bg-white px-4 pt-5 pb-4 sm:p-6 sm:pb-4 flex-1">
                    <div className="rounded-xl px-3 py-3 mb-3">
                        <div className="font-title mb-2">Advanced Search</div>
                        <div className="flex w-full flex-col sm:flex-row">
                            <div className="mb-2 sm:mb-0 sm:text-left text-default-color flex flex-1">
                                <div className="h-12 w-12 rounded-2xl bg-gray-100 mr-2 flex justify-center">
                                    <ProductIDIcon />
                                </div>
                                <input
                                    autoComplete="nope"
                                    value={searchForm.productId}
                                    placeholder="Product ID"
                                    className={defaultSearchStyle}
                                    onChange={(e) => handlerSearchValue(e, "productId")}
                                />
                            </div>
                            <div className="mb-2 sm:mb-0 sm:text-left text-default-color flex flex-1">
                                <div className="h-12 w-12 rounded-2xl bg-gray-100 mr-2 flex justify-center">
                                    <ProductIcon />
                                </div>
                                <input
                                    autoComplete="nope"
                                    value={searchForm.name}
                                    placeholder="Product Name"
                                    className={defaultSearchStyle}
                                    onChange={(e) => handlerSearchValue(e, "name")}
                                />
                            </div>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </div>

```

```

</div>

<div className="sm:bg-white rounded-xl sm:px-3 sm:py-3">
  <div className="hidden sm:flex invisible sm:visible w-full flex-col sm:flex-direction-reverse">
    <div className="sm:text-left text-default-color font-title flex-1">
      ProductID
    </div>
    <div className="sm:text-left text-default-color font-title flex-1">
      Name
    </div>
    <div className="sm:text-left text-default-color font-title flex-1">
      Amount
    </div>
    <div className="sm:text-left text-default-color font-title sm:w-11">
      Action
    </div>
  </div>

  <div>
    {currentItems &&
      currentItems.map((product) => (
        <div className={defaultTdWrapperStyle} key={product.id}>
          <div className={defaultTdStyle}>
            <div className={defaultTdContentTitleStyle}>
              ProductID
            </div>
            <div className={defaultTdContent}>
              {product.image ? (
                <img
                  className="object-cover h-10 w-10 rounded-2xl"
                  src={product.image}
                  alt={product.name}
                />
              ) : (
                <span className="h-10 w-10 rounded-2xl bg-gray-100 flex justify-center items-center">
                  <ProductIcon />
                </span>
              )}
            <span className="whitespace-nowrap text-ellipsis overflow-hidden">
              {product.productId || "#"}
            </span>
          </div>
        </div>
      )}

    <div className={defaultTdStyle}>
      <div className={defaultTdContentTitleStyle}>

```

```

Name
</div>
<div className={defaultTdContent}>
  <span className="whitespace-nowrap text-ellipsis overflow-hidden">
    {product.name}
  </span>
</div>
</div>

<div className={defaultTdStyle}>
  <div className={defaultTdContentTitleStyle}>
    Amount
  </div>
  <div className={defaultTdContent}>
    <span className="whitespace-nowrap text-ellipsis overflow-hidden">
      {product.amount}
    </span>
  </div>
</div>

<div className={defaultTdActionStyle}>
  <div className={defaultTdContentTitleStyle}>
    Action
  </div>
  <div className={defaultTdContent}>
    <Button onClick={() => handleSelect(product)}>
      Select
    </Button>
  </div>
</div>
</div>
))}

{products.length > 0 && (
  <ReactPaginate
    className="inline-flex items-center space-x-px mt-2"
    previousLinkClassName="py-2 px-3 ml-0 leading-tight text-gray-500 bg-white"
    nextLinkClassName="py-2 px-3 ml-0 leading-tight text-gray-500 bg-white"
    pageLinkClassName="py-2 px-3 ml-0 leading-tight text-gray-500 bg-white"
    breakLinkClassName="py-2 px-3 ml-0 leading-tight text-gray-500 bg-white"
    activeLinkClassName="py-2 px-3 text-blue-600 bg-blue-50 border border-gray-200"
    breakLabel="..."
    onPageChange={handlePageClick}
    pageRangeDisplayed={1}
    pageCount={pageCount}
    previousLabel="<"

```

### 3.10 Add Product

## 4 Test Cases

Table 6.1: Test Case

Test Case ID	Test Case Name	Test Data	Expected Output	Actual Output	Result
1	User enters email for registration	Enters the correct Email	Account Created Successful	Home Page	Pass
2	User Add previoud date for invoice	Enter the details	Prompt error	Prompt error	Pass
3	User add product details	Add product	Added Successfully	Add Product Page	Pass
4	Add Client Data	Invalid mail of client	Prompt error	Prompt error	Pass



## 5 Limitations

- It needs internet to be accessed.
- It does not have the features to login from another device.
- Data can't be shared within the application.
- It does not have a feature to add or remove GST.
- Graphical representation of total sales is not available.

## 6 Future Enhancements

- Automatic report generation for monthly / quarterly / yearly sales.
- Fill GST directly from app.
- Share Bill directly over mail or pdf.
- Graphical representation of total sales.

## 7 User Manual

### **Part 1 – Create Profile**

User can visit the site and create their profile using their basic information. They didn't have to take any actions after entering the data.

### **Part 2 – Add Product**

User needs to visit to the Add Product page and visit

### **Part 3 – Add Client**

User can add modify or delete the clients from the dashboard or from the Add Client Page

### **Part 4 – Invoice**

The user can generate the bill from invoice tab, while generating the invoice the user needs to select the product and the client from the existing the list

### **Part 5 – Clear Data**

User can clear their data by hitting on the clear data button

## 8 Bibliography

### 8.1 Web References

- [1.] <https://reactjs.org/tutorial/tutorial.html>
- [2.] <https://socket.io/docs/v4/server-api/>
- [3.] <https://www.youtube.com/user/SocketIO>
- [4.] <https://stackoverflow.com/>
- [5.] <https://www.draw.io/>
- [6.] <https://www.geeksforgeeks.org/>