Back End Engineering

Project Report

Semester-V (Batch-2022)

INVOTRACK

(BILLING MANAGEMENT SYSTEM)

Supervised By:

Mr. Raveesh Samkaria

Submitted By:

Saksham Vashisht (2210990768)

Sambhav Yadav (2210990774)

G-12

Department of Computer Science and Engineering
Chitkara University Institute of Engineering & Technology,
Chitkara University, Punjab

ABSTRACT

This report presents the development of Invotrack, an Integrated Store Management System (ISMS) designed to streamline inventory and billing operations across a nationwide network of retail stores. The system addresses common challenges such as inaccurate inventory records, inefficient manual processes, and the complexity of managing role-based access for various levels of management. By leveraging modern full-stack technologies like React, Node.js, Express, and MongoDB, Invotrack provides real-time data synchronization, automated billing, and comprehensive analytics. The report discusses the system's architecture, implementation, testing, and potential future enhancements

1. INTRODUCTION

In today's digital age, managing personal finances has become a critical skill for maintaining financial stability and achieving long-term goals. "Invotrack ," an innovative expense tracker, addresses this need by offering a comprehensive platform for users to track their expenses, set budgets, and analyze their spending habits. With the increasing complexity of financial management, there is a growing demand for accessible tools that simplify the process of budgeting and empower individuals to take control of their finances.

1.1.Objective

The primary objective of "Billing Management System" is to provide users with a centralized and user-friendly platform that facilitates the efficient management of personal finances. The application aims to cater to a wide range of users, from individuals new to budgeting to those with more advanced financial management needs. By offering features such as expense tracking, budget setting, and financial analysis, "Invotrack" seeks to empower users to make informed financial decisions, improve their spending habits, and ultimately achieve financial stability.

1.2 Significance

The significance of "Billing Management System" lies in its ability to democratize access to effective financial management tools. In an era where financial literacy is essential for personal success, many individuals struggle with managing their finances due to a lack of time, resources, or knowledge. "Invotrack" provides a solution by offering an accessible, online platform that allows users to manage their finances at their convenience. Whether users are at home, at work, or on the go, "Invotrack" enables them to track expenses, set budgets, and analyze their financial behavior, fostering a proactive approach to financial management and helping users achieve their financial goals.

PROBLEM STATEMENT

The problem statement for "Invotrack" centers on the challenges individuals face in managing their personal finances effectively. Despite the availability of various budgeting tools and financial apps, many fail to offer a comprehensive, intuitive platform that caters to the diverse needs of users. Common issues include a lack of user-friendly features, inadequate expense tracking, and limited options for financial analysis and budgeting. As a result, individuals often struggle to maintain financial discipline, leading to poor spending habits and financial instability. Therefore, there is a need for a robust expense tracking application that not only provides comprehensive financial management tools but also offers an intuitive user experience, personalized insights, and the flexibility to adapt to users' unique financial situations.

REQUIREMENTS

1. User Authentication: The application must support secure user authentication, including registration, login, and logout functionality. Passwords should be encrypted using secure hashing algorithms.

2. Expense Management: Users should be able to add, edit, and delete expenses, with fields for amount, description, category, and date. The application should also allow users to sort expenses by date and amount.

3. Budget Tracking: The app should allow users to set a monthly budget. If expenses exceed the budget, a warning message should be displayed; otherwise, the remaining budget should be shown.

4. Data Analysis: The app should include features to analyse expenses, such as generating daily, weekly, and monthly totals, and visualizing spending by category through charts.

5. Savings Calculation: The app should calculate potential savings based on user expenses, including the option to calculate future savings with interest over time.

6. User-Friendly Interface: The application should provide an intuitive and user-friendly interface, making it easy for users to navigate and manage their expenses effectively.

WEB DEVELOPMENT TECHNOLOGIES USED

1. Front-End:

- **React :** Used for rendering dynamic HTML content and templating in Node.js.
- **Tailwind:** Used for styling the webpages.React

2. Back-End:

- **Node.js:** Server-side JavaScript runtime used for building the backend of the application.
- **Express.js:** A web application framework for Node.js, utilized to create the server, manage routes, and handle requests and responses.

3. Database:

- **MongoDB:** A NoSQL database used to store and manage user data, expenses, and categories.

4. Authentication & Security:

- **bcrypt:** Used for hashing passwords to ensure secure authentication.
- **JWT (JSON Web Tokens):** Implemented for secure user authentication and session management.

5. Data Visualization:

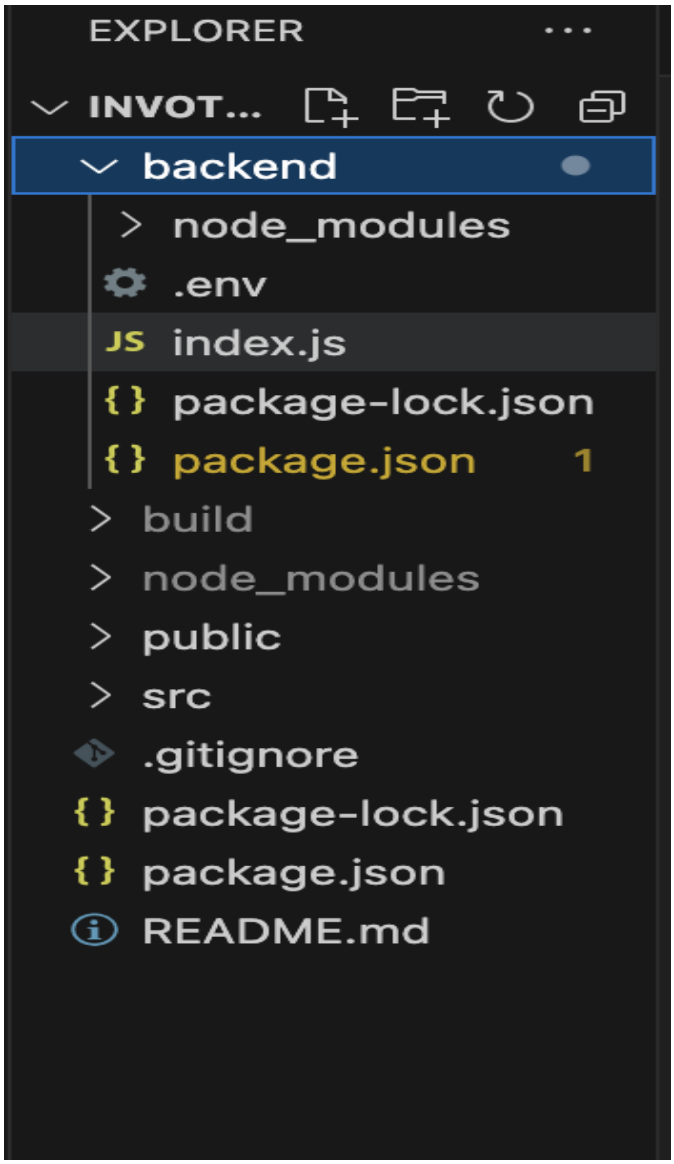
- **Chart.js:** Libraries used for creating charts and graphs to visualize expense data, providing insights into spending habits.

PROPOSED DESIGN

The proposed design for the Expense Tracker app features a user-friendly interface with intuitive navigation and responsive design for accessibility across all devices. The app will include a visually appealing layout with clear expense management tools and high-quality graphics. Users will benefit from organized categories, powerful search and filtering options, and comprehensive financial analysis tools. The design aims to enhance user experience by simplifying expense tracking and providing valuable insights into spending patterns.

CODE

- **Folder Structure:**



- **Index.js(Backend)**

backend > JS index.js > ...

```javascript
const express = require("express");
const mongoose = require("mongoose");
const cors = require("cors");
const dotenv = require("dotenv");

dotenv.config();

const app = express();

app.use(cors());
app.use(express.json());

mongoose.connect(process.env.MONGODB_URL, {
  useNewUrlParser: true,
  useUnifiedTopology: true,
})
Tabnine: Edit | Test | Explain | Document | Ask
.then(() => console.log("Connected to MongoDB"))
Tabnine: Edit | Test | Explain | Document | Ask
.catch((error) => console.log("Error connecting to MongoDB:", error));

const userSchema = new mongoose.Schema({
  firstName: String,
  lastName: String,
  email: String,
  contact: String,
  address1: String,
  address2: String,
});

const User = mongoose.model("User", userSchema);

Tabnine: Edit | Test | Explain | Document | Ask
app.post("/api/form", async (req, res) => {
  try {
    const newUser = new User(req.body);
    await newUser.save();
    res.status(201).json(newUser);
  } catch (error) {
    res.status(500).json({ error: "Failed to create user" });
  }
});

const PORT = process.env.PORT || 3001;
Tabnine: Edit | Test | Explain | Document | Ask
app.listen(PORT, () => console.log(`Server running on port ${PORT}`));
```

- **App.js:**

src > JS App.js > ...

```
import { useState } from "react";
import { Routes, Route } from "react-router-dom";
import Topbar from "./scenes/global/Topbar";
import Sidebar from "./scenes/global/Sidebar";
import Dashboard from "./scenes/dashboard";
import Team from "./scenes/team";
import Invoices from "./scenes/invoices";
import Contacts from "./scenes/contacts";
import Bar from "./scenes/bar";
import Form from "./scenes/form";
import Line from "./scenes/line";
import Pie from "./scenes/pie";
import FAQ from "./scenes/faq";
import Geography from "./scenes/geography";
import { CssBaseline, ThemeProvider } from "@mui/material";
import { ColorModeContext, useMode } from "./theme";
import Calendar from "./scenes/calendar/calendar";

Tabnine: Edit | Test | Explain | Document | Ask
function App() {
  const [theme, colorMode] = useMode();
  const [isSidebar, setIsSidebar] = useState(true);
```

```
  return (
    <ColorModeContext.Provider value={colorMode}>
      <ThemeProvider theme={theme}>
        <CssBaseline />
        <div className="app">
          <Sidebar isSidebar={isSidebar} />
          <main className="content">
            <Topbar setIsSidebar={setIsSidebar} />
            <Routes>
              <Route path="/" element={<Dashboard />} />
              <Route path="/team" element={<Team />} />
              <Route path="/contacts" element={<Contacts />} />
              <Route path="/invoices" element={<Invoices />} />
              <Route path="/form" element={<Form />} />
              <Route path="/bar" element={<Bar />} />
              <Route path="/pie" element={<Pie />} />
              <Route path="/line" element={<Line />} />
              <Route path="/faq" element={<FAQ />} />
              <Route path="/calendar" element={<Calendar />} />
              <Route path="/geography" element={<Geography />} />
            </Routes>
          </main>
        </div>
      </ThemeProvider>
    </ColorModeContext.Provider>
  );
}

export default App;
```

- **index.js**

src > JS index.js > ...

```
import React from "react";
import ReactDOM from "react-dom/client";
import "./index.css";
import App from "./App";
import { BrowserRouter } from "react-router-dom";

const root = ReactDOM.createRoot(document.getElementById("root"));
root.render(
  <React.StrictMode>
    <BrowserRouter>
      <App />
    </BrowserRouter>
  </React.StrictMode>
);
```

- **Theme.js**

```
src > JS theme.js > ...
import { createContext, useState, useMemo } from "react";
import { createTheme } from "@mui/material/styles";

// color design tokens export
export const tokens = (mode) => ({
  ...(mode === "dark"
    ? {
        grey: {
          100: "#e0e0e0",
          200: "#c2c2c2",
          300: "#a3a3a3",
          400: "#858585",
          500: "#666666",
          600: "#525252",
          700: "#3d3d3d",
          800: "#292929",
          900: "#141414",
        },
        primary: {
          100: "#d0d1d5",
          200: "#a1a4ab",
          300: "#727681",
          400: "#1F2A40",
          500: "#141b2d",
          600: "#101624",
          700: "#0c101b",
          800: "#080b12",
          900: "#040509",
        },
        greenAccent: {
          100: "#dbf5ee",
          200: "#b7ebde",
          300: "#94e2cd",
          400: "#70d8bd",
          500: "#4cceac",
          600: "#3da58a",
          700: "#2e7c67",
          800: "#1e5245",
          900: "#0f2922",
        },
```

src > JS theme.js > tokens > <unknown>

```
39            900: "#012922",
40          },
41          redAccent: {
42            100: "#f8dcdb",
43            200: "#f1b9b7",
44            300: "#e99592",
45            400: "#e2726e",
46            500: "#db4f4a",
47            600: "#af3f3b",
48            700: "#832f2c",
49            800: "#58201e",
50            900: "#2c100f",
51          },
52          blueAccent: {
53            100: "#e1e2fe",
54            200: "#c3c6fd",
55            300: "#a4a9fc",
56            400: "#868dfb",
57            500: "#6870fa",
58            600: "#535ac8",
59            700: "#3e4396",
60            800: "#2a2d64",
61            900: "#151632",
62          },
63        }
64      : {
65          grey: {
66            100: "#141414",
67            200: "#292929",
68            300: "#3d3d3d",
69            400: "#525252",
70            500: "#666666",
71            600: "#858585",
72            700: "#a3a3a3",
73            800: "#c2c2c2",
74            900: "#e0e0e0",
75          },
76          primary: {
77            100: "#040509",
78            200: "#080b12",
79            300: "#0c101b",
80            400: "#f2f0f0", // manually changed
81            500: "#141b2d",
82            600: "#1F2A40",
83            700: "#727681",
84            800: "#a1a4ab",
85            900: "#d0d1d5",
```

```
29            },
30            greenAccent: {
31              100: "#dbf5ee",
32              200: "#b7ebde",
33              300: "#94e2cd",
34              400: "#70d8bd",
35              500: "#4cceac",
36              600: "#3da58a",
37              700: "#2e7c67",
38              800: "#1e5245",
39              900: "#0f2922",
40            },
41            redAccent: {
42              100: "#f8dcdb",
43              200: "#f1b9b7",
44              300: "#e99592",
45              400: "#e2726e",
46              500: "#db4f4a",
47              600: "#af3f3b",
48              700: "#832f2c",
49              800: "#58201e",
50              900: "#2c100f",
51            },
52            blueAccent: {
53              100: "#e1e2fe",
54              200: "#c3c6fd",
55              300: "#a4a9fc",
56              400: "#868dfb",
57              500: "#6870fa",
58              600: "#535ac8",
59              700: "#3e4396",
60              800: "#2a2d64",
61              900: "#151632",
62            },
63          }
64        : {
65            grey: {
66              100: "#141414",
67              200: "#292929",
68              300: "#3d3d3d",
69              400: "#525252",
70              500: "#666666",
71              600: "#858585",
72              700: "#a3a3a3",
73              800: "#c2c2c2",
74              900: "#e0e0e0",
```

```
98          redAccent: {
99            100: "#2c100f",
100           200: "#58201e",
101           300: "#832f2c",
102           400: "#af3f3b",
103           500: "#db4f4a",
104           600: "#e2726e",
105           700: "#e99592",
106           800: "#f1b9b7",
107           900: "#f8dcdb",
108         },
109         blueAccent: {
110           100: "#151632",
111           200: "#2a2d64",
112           300: "#3e4396",
113           400: "#535ac8",
114           500: "#6870fa",
115           600: "#868dfb",
116           700: "#a4a9fc",
117           800: "#c3c6fd",
118           900: "#e1e2fe",
119         },
120       }),
121 });
122
123 // mui theme settings
124 export const themeSettings = (mode) => {
125   const colors = tokens(mode);
126   return {
127     palette: {
128       mode: mode,
129       ...(mode === "dark"
130         ? {
131             // palette values for dark mode
132             primary: {
133               main: colors.primary[500],
134             },
135             secondary: {
136               main: colors.greenAccent[500],
137             },
138             neutral: {
139               dark: colors.grey[700],
140               main: colors.grey[500],
141               light: colors.grey[100],
142             },
143             background: {
```

```
// mui theme settings
export const themeSettings = (mode) => {
  const colors = tokens(mode);
  return {
    palette: {
      mode: mode,
      ...(mode === "dark"
        ? {
            // palette values for dark mode
            primary: {
              main: colors.primary[500],
            },
            secondary: {
              main: colors.greenAccent[500],
            },
            neutral: {
              dark: colors.grey[700],
              main: colors.grey[500],
              light: colors.grey[100],
            },
            background: {
              default: colors.primary[500],
            },
          }
        : {
            // palette values for light mode
            primary: {
              main: colors.primary[100],
            },
            secondary: {
              main: colors.greenAccent[500],
            },
            neutral: {
              dark: colors.grey[700],
              main: colors.grey[500],
              light: colors.grey[100],
            },
            background: {
              default: "#fcfcfc",
            },
          }),
    },
    typography: {
      fontFamily: ["Source Sans Pro", "sans-serif"].join(","),
      fontSize: 12,
      h1: {
```

```
169          fontFamily: ["Source Sans Pro", "sans-serif"].join(","),
170          fontSize: 40,
171        },
172        h2: {
173          fontFamily: ["Source Sans Pro", "sans-serif"].join(","),
174          fontSize: 32,
175        },
176        h3: {
177          fontFamily: ["Source Sans Pro", "sans-serif"].join(","),
178          fontSize: 24,
179        },
180        h4: {
181          fontFamily: ["Source Sans Pro", "sans-serif"].join(","),
182          fontSize: 20,
183        },
184        h5: {
185          fontFamily: ["Source Sans Pro", "sans-serif"].join(","),
186          fontSize: 16,
187        },
188        h6: {
189          fontFamily: ["Source Sans Pro", "sans-serif"].join(","),
190          fontSize: 14,
191        },
192      },
193    };
194  };
195
196  // context for color mode
197  export const ColorModeContext = createContext({
198    toggleColorMode: () => {},
199  });
200
201  export const useMode = () => {
202    const [mode, setMode] = useState("dark");
203
204    const colorMode = useMemo(
205      () => ({
206        toggleColorMode: () =>
207          setMode((prev) => (prev === "light" ? "dark" : "light")),
208      }),
209      []
210    );
211
212    const theme = useMemo(() => createTheme(themeSettings(mode)), [mode]);
213    return [theme, colorMode];
214  };
```

- **Invoices (invoices.js)**

src > scenes > invoices > index.js > ...

```jsx
import { Box, Typography, useTheme } from "@mui/material";
import { DataGrid } from "@mui/x-data-grid";
import { tokens } from "../../theme";
import { mockDataInvoices } from "../../data/mockData";
import Header from "../../components/Header";

const Invoices = () => {
  const theme = useTheme();
  const colors = tokens(theme.palette.mode);
  const columns = [
    { field: "id", headerName: "ID" },
    {
      field: "name",
      headerName: "Name",
      flex: 1,
      cellClassName: "name-column--cell",
    },
    {
      field: "phone",
      headerName: "Phone Number",
      flex: 1,
    },
    {
      field: "email",
      headerName: "Email",
      flex: 1,
    },
    {
      field: "cost",
      headerName: "Cost",
      flex: 1,
      renderCell: (params) => (
        <Typography color={colors.greenAccent[500]}>
          ${params.row.cost}
        </Typography>
      ),
    },
    {
      field: "date",
      headerName: "Date",
      flex: 1,
    },
  ];

  return (
    <Box m="20px">
```

```
44
45    return (
46      <Box m="20px">
47        <Header title="INVOICES" subtitle="List of Invoice Balances" />
48        <Box
49          m="40px 0 0 0"
50          height="75vh"
51          sx={{
52            "& .MuiDataGrid-root": {
53              border: "none",
54            },
55            "& .MuiDataGrid-cell": {
56              borderBottom: "none",
57            },
58            "& .name-column--cell": {
59              color: colors.greenAccent[300],
60            },
61            "& .MuiDataGrid-columnHeaders": {
62              backgroundColor: colors.blueAccent[700],
63              borderBottom: "none",
64            },
65            "& .MuiDataGrid-virtualScroller": {
66              backgroundColor: colors.primary[400],
67            },
68            "& .MuiDataGrid-footerContainer": {
69              borderTop: "none",
70              backgroundColor: colors.blueAccent[700],
71            },
72            "& .MuiCheckbox-root": {
73              color: `${colors.greenAccent[200]} !important`,
74            },
75          }}
76        >
77          <DataGrid checkboxSelection rows={mockDataInvoices} columns={columns} />
78        </Box>
79      </Box>
80    );
81  };
82
83  export default Invoices;
```

- **Calender (calender.js)**

src > scenes > calendar > JS calendar.js > ...

```javascript
import { useState } from "react";
import FullCalendar from "@fullcalendar/react";
import { formatDate } from "@fullcalendar/core";
import dayGridPlugin from "@fullcalendar/daygrid";
import timeGridPlugin from "@fullcalendar/timegrid";
import interactionPlugin from "@fullcalendar/interaction";
import listPlugin from "@fullcalendar/list";
import {
  Box,
  List,
  ListItem,
  ListItemText,
  Typography,
  useTheme,
} from "@mui/material";
import Header from "../../components/Header";
import { tokens } from "../../theme";

const Ca const theme: Theme
  const theme = useTheme();
  const colors = tokens(theme.palette.mode);
  const [currentEvents, setCurrentEvents] = useState([]);

  const handleDateClick = (selected) => {
    const title = prompt("Please enter a new title for your event");
    const calendarApi = selected.view.calendar;
    calendarApi.unselect();

    if (title) {
      calendarApi.addEvent({
        id: `${selected.dateStr}-${title}`,
        title,
        start: selected.startStr,
        end: selected.endStr,
        allDay: selected.allDay,
      });
    }
  };

  const handleEventClick = (selected) => {
    if (
      window.confirm(
        `Are you sure you want to delete the event '${selected.event.title}'`
      )
    ) {
      selected.event.remove();
```

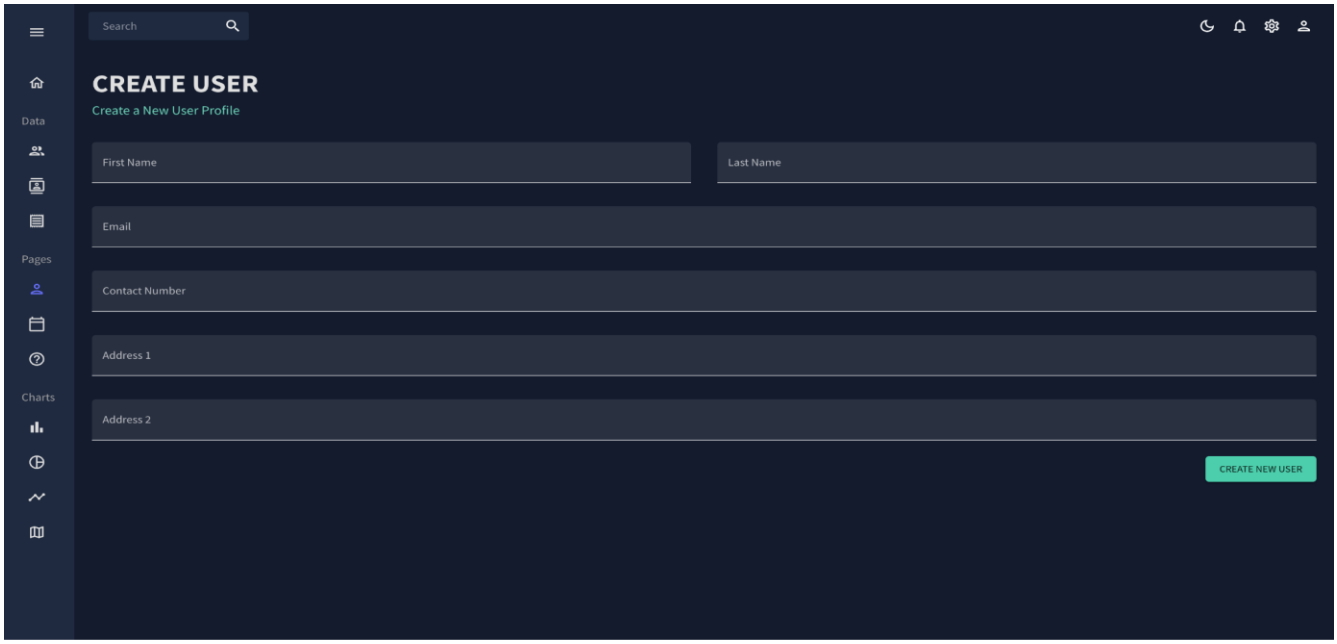
```
50  return (
51    <Box m="20px">
52      <Header title="Calendar" subtitle="Full Calendar Interactive Page" />
53
54      <Box display="flex" justifyContent="space-between">
55        {/* CALENDAR SIDEBAR */}
56        <Box
57          flex="1 1 20%"
58          backgroundColor={colors.primary[400]}
59          p="15px"
60          borderRadius="4px"
61        >
62          <Typography variant="h5">Events</Typography>
63          <List>
64            {currentEvents.map((event) => (
65              <ListItem
66                key={event.id}
67                sx={{
68                  backgroundColor: colors.greenAccent[500],
69                  margin: "10px 0",
70                  borderRadius: "2px",
71                }}
72              >
73                <ListItemText
74                  primary={event.title}
75                  secondary={
76                    <Typography>
77                      {formatDate(event.start, {
78                        year: "numeric",
79                        month: "short",
80                        day: "numeric",
81                      })}
82                    </Typography>
83                  }
84                />
85              </ListItem>
86            ))}
87          </List>
88        </Box>
89
90        {/* CALENDAR */}
91        <Box flex="1 1 100%" ml="15px">
92          <FullCalendar
93            height="75vh"
94            plugins={[
95              dayGridPlugin,
```

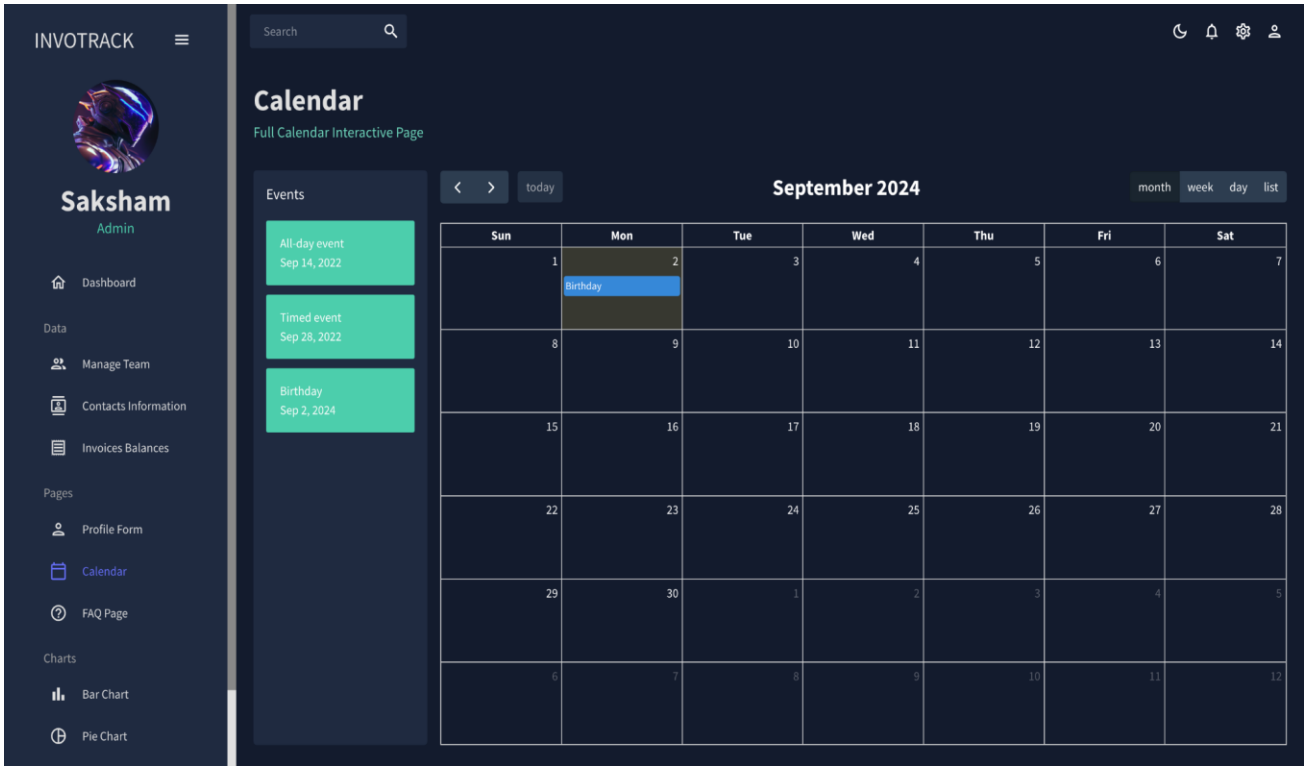
```
        {/* CALENDAR */}
        <Box flex="1 1 100%" ml="15px">
          <FullCalendar
            height="75vh"
            plugins={[
              dayGridPlugin,
              timeGridPlugin,
              interactionPlugin,
              listPlugin,
            ]}
            headerToolbar={{
              left: "prev,next today",
              center: "title",
              right: "dayGridMonth,timeGridWeek,timeGridDay,listMonth",
            }}
            initialView="dayGridMonth"
            editable={true}
            selectable={true}
            selectMirror={true}
            dayMaxEvents={true}
            select={handleDateClick}
            eventClick={handleEventClick}
            eventsSet={(events) => setCurrentEvents(events)}
            initialEvents={[
              {
                id: "12315",
                title: "All-day event",
                date: "2022-09-14",
              },
              {
                id: "5123",
                title: "Timed event",
                date: "2022-09-28",
              },
            ]}
          />
        </Box>
      </Box>
    );
  };

  export default Calendar;
```

RESULTS

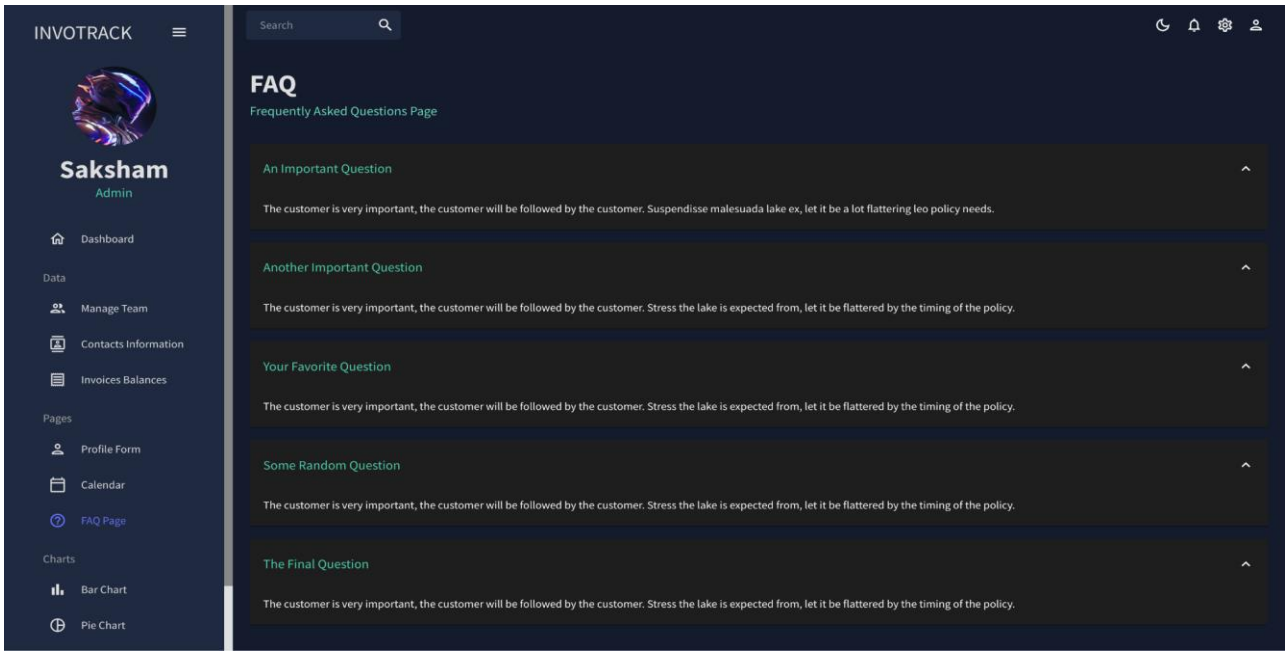
- **HOME PAGE**

- **ADDING USERS**




INVOTRACK
Search
Saksham
Admin
Dashboard
Data
Manage Team
Contacts Information
Invoices Balances
Pages
Profile Form
Calendar
FAQ Page
Charts
Bar Chart
Pie Chart
Calendar
Full Calendar Interactive Page
Events
All-day event
Sep 14, 2022
Timed event
Sep 28, 2022
Birthday
Sep 2, 2024
today
September 2024
month
week
day
list
Sun
Mon
Tue
Wed
Thu
Fri
Sat
Birthday

FAQ PAGE


INVOTRACK
Search
Saksham
Admin
Dashboard
Data
Manage Team
Contacts Information
Invoices Balances
Pages
Profile Form
Calendar
FAQ Page
Charts
Bar Chart
Pie Chart
FAQ
Frequently Asked Questions Page
An Important Question
The customer is very important, the customer will be followed by the customer. Suspendisse malesuada lake ex, let it be a lot flattering leo policy needs.
Another Important Question
The customer is very important, the customer will be followed by the customer. Stress the lake is expected from, let it be flattered by the timing of the policy.
Your Favorite Question
The customer is very important, the customer will be followed by the customer. Stress the lake is expected from, let it be flattered by the timing of the policy.
Some Random Question
The customer is very important, the customer will be followed by the customer. Stress the lake is expected from, let it be flattered by the timing of the policy.
The Final Question
The customer is very important, the customer will be followed by the customer. Stress the lake is expected from, let it be flattered by the timing of the policy.

REFERENCES

- **Front-End ->**
  1. **React Documentation** - https://www.npmjs.com/package/ejs
  2. **CSS Documentation** - https://css.com/docs
- **Back-End ->**
  1. **Express.js Documentation** - https://expressjs.com/
  2. **Node.js Documentation** - https://nodejs.org/en/learn
- **User Authentication ->**
  1. **Bcrypt.js** - https://www.npmjs.com/package/bcryptjs
  2. **JSON Web Token (JWT)** - https://jwt.io/introduction/
- **Database Management ->**
  1. **MongoDB** - https://docs.mongodb.com/
- **Graph Visualization ->**
  1. **Chart.js Documentation** - https://www.chartjs.org/docs/latest/