TaxRx

Project

submitted in partial fulfillment of the requirement for
the award of the degree of

Bachelor of Technology

In

Computer Science and Engineering

By

ARSHDEEP SINGH
Enroll. No. A50105219074

Under the guidance of

Ms. Neha Bhateja
Assistant professor I
CSE, ASET

Department of Computer Science and Engineering
Amity School of Engineering and Technology Amity
University Haryana
Gurgaon, India
March, 2022

Department of Computer Science and Engineering

Amity School of Engineering and Technology

DECLARATION

I, Arshdeep Singh, A50105219074, student of Bachelor of Technology in Department of Computer Science and Engineering, Amity School of Engineering and Technology, Amity University Haryana, hereby declare that I am fully responsible for the information and results provided in this project report titled "TaxRx" submitted to the Department of Computer Science and Engineering, Amity School of Engineering and Technology, Amity University Haryana, Gurgaon for the partial fulfilment of the requirement for the award of the degree of Bachelor of Technology in Computer Science and Engineering. I have taken care in all respects to honour the intellectual property rights and have acknowledged the contributions of others for using them. I further declare that in case of any violation of intellectual property rights or copyrights, I as a candidate will be fully responsible for the same. My supervisor, Head of department and the Institute should not be held for full or partial violation of copyrights if found at any stage of my degree.

ARSHDEEP SINGH
A50105219074

Department of Computer Science and Engineering

Amity School of Engineering and Technology

CERTIFICATE

This is to certify that the work in the project report entitled “TaxRx” by Arshdeep Singh bearing Enroll. No. A50105219074 is a bonafide record of project work carried out by him under my supervision and guidance in partial fulfilment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering in the Department of Computer Science and Engineering, Amity School of Engineering and Technology, Amity University Haryana, Gurgaon. Neither this project nor any part of it has been submitted for any degree or academic award elsewhere.

Date:

Supervisor:

Ms. Neha Bhateja

Assistant Professor I

Head

Department of Computer Science & Engineering

Amity School of Engineering and Technology

Amity University Haryana, Gurgaon

ACKNOWLEDGMENT

"With hard work, dedication, commitment, efforts, and the right guidance nothing is impossible."

While I was preparing this project report, various information that I found helped me in knowing in the field of web technologies and I am glad that I was able to complete this project and understand many things. Through preparation of this project was an immense learning experience and I build-up many personal qualities during this process like responsibility, determination and punctuality.

I would like to thank to my teachers who supported me all the time, cleared my doubts and to my parents who supported me in every possible manner. I am taking this opportunity to acknowledge and express my deep sense of gratitude to my project supervisor Ms. Neha Bhateja for their expert guidance and invaluable help guiding me throughout the making of this project report.

One more, I would like to thank my friends for their encouragement and help in designing and making my project creative and helping me out with my project issues.

ARSHDEEP SINGH

A50105219074

ABSTRACT

"TaxRx" is a full stack web application that offers statistics and data visualization on invoices and GST return file in the form of tables, interactive line graphs, doughnut graph visualization, create invoice, customer estimate, expense and vendors. A user can register himself to the website and then the user can choose to either send the invoice to someone or can file a GST return. This web page gives eye-catching data visualization by displaying the total expenses and on what category payment was done by using 'Echarts' to create visually appealing, interactive and easy to understand line graph and doughnut graph. A user can perform CRUD(Create, Read, Update, Delete) operations and all his credentials, data is stored in a database "Postgres SQL" and displayed in the form of tables on website. This makes a better user friendly experience.

LIST OF FIGURES

Contents

Chapter 1

INTRODUCTION

1.1 INTRODUCTION TO WEB TECHNOLOGIES

FRONTEND:

JAVASCRIPT

JavaScript (JS) is a lightweight, interpreted, or just-in-time compiled programming language with first-class functions. While it is most well-known as the scripting language for Web pages, many non-browser environments also use it, such as Node.js.

It was introduced in the year 1995 for adding programs to the webpages in the Netscape Navigator browser. With JavaScript, users can build modern web applications to interact directly without reloading the page every time.

Features of JavaScript

There are following features of JavaScript:

1. All popular web browsers support JavaScript as they provide built-in execution environments.
2. JavaScript is an object-oriented programming language that uses prototypes rather thanusing classes for inheritance.
3. It is a case sensitive, light-weighted, and interpreted language.

Figure 1.1.1 JS [8]

INTRODUCTION TO REACT

"A JavaScript library for building user interfaces".

ReactJS is a declarative, efficient, and flexible JavaScript library for building reusable UI components. It is an open-source, component-based front end library responsible only for the view layer of the application.

It was created by Jordan Walker, who was a software engineer at Facebook. It was initially developed and maintained by Facebook and was later used in its products like WhatsApp & Instagram. Facebook developed ReactJS in 2011 in its newsfeed section, but it was released to the public in the month of May 2013.

A ReactJS application is made up of multiple components, each component responsible for outputting a small, reusable piece of HTML code. The components are the heart of all React applications. These Components can be nested with other components to allow complex applications to be built of simple building blocks. ReactJS uses virtual DOM based mechanism to fill data in HTML DOM. The virtual DOM works fast as it only changes individual DOM elements instead of reloading complete DOM every time.

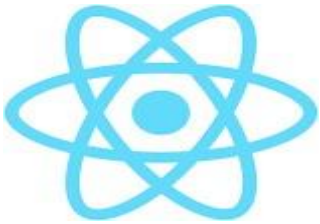

Figure 1.1.2: ReactJS [23]

Why learn ReactJS?

There are many JavaScript frameworks which are available in the market(like angular, vue), but I have chosen React because the previous frameworks follow the traditional data flow structure, which uses the DOM (Document Object Model). But, ReactJS allows you to divide your entire application into various components. ReactJS still used the same traditional data flow, but it is not directly operating on the browser's Document Object Model (DOM) immediately; instead, it operates on a virtual DOM. It means rather than manipulating the document in a browser after changes to our data, it resolves changes on a DOM built and run entirely in memory.

BACKEND:

INTRODUCTION TO PYTHON

Python is a widely used general-purpose, high level programming language. It was created by Guido van Rossum in 1991 and further developed by the Python Software Foundation. It was designed with an emphasis on code readability, and its syntax allows programmers to express their concepts in fewer lines of code.

Python is a programming language that lets you work quickly and integrate systems more efficiently.

Features of Python:

1. Easy to learn and use
2. Free and open source
3. GUI programming support
4. Large Standard Library

Figure 1.1.3: Python [9]

DJANGO

Django is a high-level Python web framework that enables rapid development of secure and maintainable websites. Built by experienced developers, Django takes care of much of the hassle of web development, so you can focus on writing your app without needing to reinvent the wheel. It is free and open source, has a thriving and active community, great documentation, and many options for free and paid-for support.

Features of Django:

1. Rapid Development
2. Scalable
3. Versatile and secure
4. Vast and supported community
5. Open source

Figure 1.1.4: Django [10]

DATABASE:

POSTGRES SQL

PostgreSQL is used as the primary data store or data warehouse for many web, mobile, geospatial, and analytics applications. PostgreSQL is an advanced, enterprise class open source relational database that supports both SQL (relational) and JSON (non-relational) querying. It is a highly stable database management system

Figure 1.1.5: PostgreSql [11]

API - Application Programming Interface

An API is a set of definitions and protocols for building and integrating application software.

Its act as linkage between an organization or server side database. It connects our web application or software using that linkage that can be enough form of URL, it can be in form of Post and Get request. API has its own feature that can downgrade the API content and limit the user interface:

1. API limitations
2. API authentication.
3. Post and Get Request
4. Update and Delete Request
5. Https errors

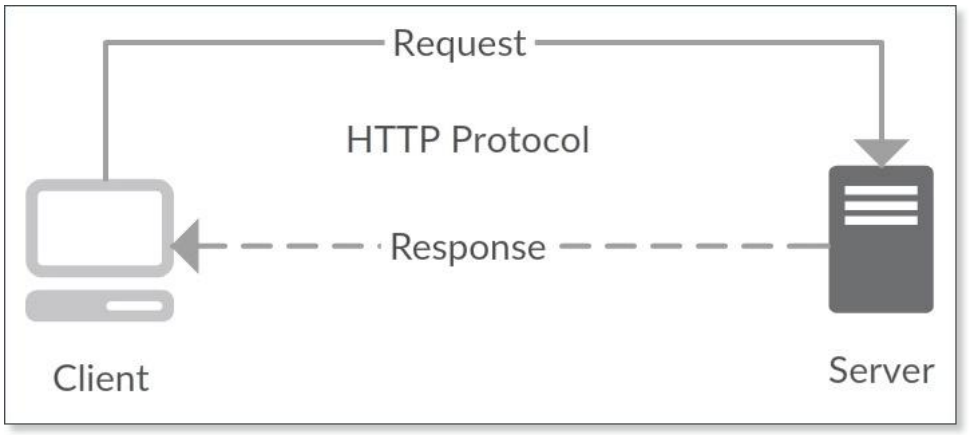

Figure 1.1.6: API [24]

REST API

When a client request is made via a RESTful API, it transfers a representation of the state of the resource to the requester or endpoint. This information, or representation, is delivered in one of several formats via HTTP: JSON (Javascript Object Notation), HTML, XLT, Python, PHP, or plain text.

1.2 AIM and OBJECTIVE

"TaxRx" is a full stack web application which is used to send, store the invoice and file GST return. So to provide a platform where all these things can be done is very hard to find. So we built this user friendly website where everything can be done within frictions of seconds.

The main objective is to provide correct information and better UX:

1. The Purpose of Line Graph and Doughnut Graph are to provide the user better clarity about the data.
2. All the necessary data is being provided on a single page which makes the navigation easy and helps in providing the compact information to the users.
3. It also helps in managing the customers and vendor's data, their expenses, and the invoices data.

1.3 CURRENT SYSTEM

As we consider current system there are very few websites which allow the users to file GST return and manage his personal business in very user friendly way. All these things under one page makes the user very comfortable to use this website and manage his expenditures in a very organized way.

1.4 PROPOSED SYSTEM

To overcome all the above flaws some new and good features are added to it:

1. Provide compact information at one page.
2. Using React, rendering of components (by using react hooks) and data become faster.
3. Added Line Graph (echart.js) and Doughnut Graph for more clarity of data.
4. Data represented in the form of tables which fetches (get) data from the database using a Restful API's.
5. Easy to manage business under one place.
6. Provide better user experience and easy to understand.

Chapter 2

PROJECT BACKGROUND

The previous website of TaxRx was fully static with all the data and credentials stored already on the website. There was no authentication of the user. Every single bit of data was static with no edit and delete operations. Dashboard was also very simple with just 4 cards holding total users, total customers, total invoices, and total estimates with side navbar have home, invoices, users, customers, vendors etc. Only admin was able to login to that website.

What we did was to convert that static website into fully functional dynamic website where at first there is Email based authentication where the user needs to verify his account then only, he can login to the website. Once the user is login in, he/she is asked to provide the information regarding his business as the whole purpose of this website was to help businesses manage their expenses and keep a track of vendors, customers, invoices, and his monthly expenditure. Then he can choose whether he wants to file GST return or send someone his invoice. Once he reaches the dashboard he can have a look of his vendors, customers, invoices, and his monthly expenditure. He can also have a look where he spent his money this month and how much he has spent this month compared to the previous month.

We used Django for the backend, PostgreSQL for database and ReactJs for frontend. In this website a user can add any details, edit it or delete it any time he wants which makes this website a dynamic website. We need data visualization tools such as line chart and doughnut chart to make website more realistic and make it more attractive and pleasant to look at, which was not there in the previous version of this website.

Chapter 3

TOOLS & TECHNOLOGIES USED

3.1 WIREFRAMING TOOL

Wire framing is a process of designing the overview or a blue-print of products to establish the structure and flow of possible design solutions for your project.

3.2 APACE ECHARTS

Apache ECharts is a free, powerful charting and visualization library offering an easy way of adding intuitive, interactive, and highly customizable charts to your commercial products. It is written in pure JavaScript and based on zrender, which is a whole new lightweight canvas library.

3.3 DJOSER

Djoser library provides a set of Django Rest Framework views to handle basic actions such as registration, login, logout, password reset and account activation. It works with custom user model.

3.4 DJANGO REST FRAMEWORK

Django REST framework is a powerful and flexible toolkit for building Web APIs.

Figure 3.7.1: Django Rest Framework [18]

3.5 MATERIAL-UI - A React UI framework

Material-UI is simply a library that allows us to import and use different components to create a user interface in our React applications.

Material UI can provide you pre-styled components.

3.6 PSYCOPG

Psycopg is the most popular PostgreSQL database adapter for the Python programming language. Its main features are the complete implementation of the Python DB API 2.0 specification and the thread safety (several threads can share the same connection). It was designed for heavily multi-threaded applications that create and destroy lots of cursors and make a large number of concurrent "INSERT"s or "UPDATE"s.

Figure 3.5.1: Psycopg [16]

3.7 REACT BOOTSTRAP

React-Bootstrap is a component-based library that provides native Bootstrap components as pure React components. Instead of utilizing JavaScript source and plugins from the CDN, it converts all JavaScript to react and bundles all components together.

Chapter 4

SOFTWARE AND HARDWARE REQUIREMENTS

Hardware configuration:

Processor: i5 or i7

RAM: 2 GB

Free Space required on hard disk: 500 MB

Software requirement:

Any Text editor (VS Code, Sublime, etc).

Nodejs version >= 11 & yarn/npm

Django & Djoser

Web browser

Software requirement: (Client end)

HTTP supported Web browser

Web connectivity

Nodejs version >= 11 & yarn/npm

Chapter 5

DESIGN OF A PROJECT

5.1 WIREFRAMING

Wire framing is a process of designing the overview or a blue-print of products to establish the structure and flow of possible design solutions for your project.

FRONTEND -

The design of a frontend is divided into multiple components:

1. Header: It includes the toggle button, search, notification and user profile area.
2. Statistics Cards: There will be 4 info cards each display the data of total users, customers, invoices and the total estimates.
3. Line Graph: Displays the data of total expenses in the form of Line graph which shows the data of current year and previous year.
4. Doughnut Graph: Display the data of expenses i.e on what category the user spent his money.
5. Shortcut Card: Card which displays the shortcut to add invoice, add customer, add vendor or to invite a guest person.
6. Side Navbar: Shows the links such as dashboard, users, expenses, signup and pricing plan.
7. Footer: Show the copyright and author name.

Tool used here: app.diagram.net

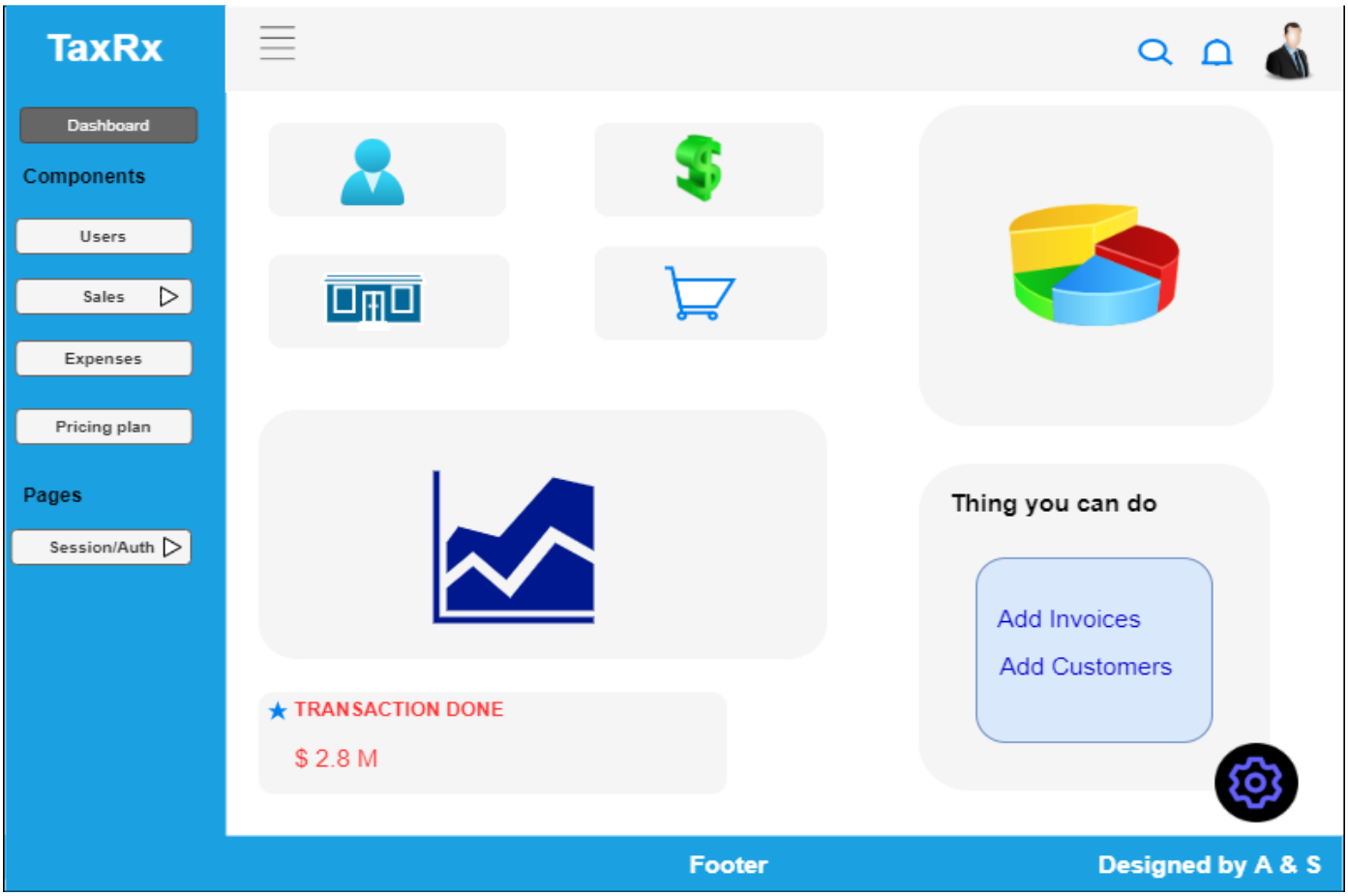

Figure 5.1.1: Wireframe dashboard layout

BACKEND & DATABASE –

The design of a frontend is divided into multiple components:

1. Authentication
2. Users
3. Customers
4. Vendors
5. Expenses
6. Invoices
7. GST Form

1	REGISTATION OF USER	
	1 USER NAME	
	2 NUMBER	
	3 EMAIL ID	

Figure 5.1.2: Fields for Authentication

2	SELF PROFILE		
	1 TRADE NAME / BUSINESS NAME		
	2 GST NO	IF AVAILBLE	
	3 OFFICE ADDRESS		
	4 STATE		
	5 STAE CODE		
	6 NUMBER		
	7 EMAIL ID		
	8 PAN NO		
	9 AUTHORISED SIGNATORY PERSON'S NAME		
	10 BANK DETAILS		

Figure 5.1.3: Fields for User Registration

3	CUSTOMER REGISTAION	
	1 NAME OF CUSTOMER	
	2 ADDRESS OF CUSTOMER	
	3 GST NUMBER	IF AVAILBLE
	4 PAN NO	
	5 EMAIL ID	
	6 NUMBER	
	7 BANK DETAILS	

Figure 5.1.4: Fields for Customer Registration

GST SERVICES				
1	REGISTRATION	999	2499	5999
		1. GST REGISTRATION	1. GST REGISTRATION	1. GST REGISTRATION
		2. 1 MONTH RETURN FILLING	2. 3 MONTHS RETURN FILLING	2. 12 MONTHS RETURN FILLING
		3. 1 MONTHS PERSONAL ASSISTANCE	3. 3 MONTHS PERSONAL ASSISTANCE	3. 12 MONTHS PERSONAL ASSISTANCE
2	GST RETURN FILLING	599	2999	4999
		1. 1 MONTH RETURN FILLING	1. 6 MONTHS RETURN FILLING	1. 12 MONTHS RETURN FILLING
			2. 6 MONTH PERSONAL ASSISTANCE	2. 12 MONTH PERSONAL ASSISTANCE

Figure 5.1.5: GST Price Plans

Chapter 6

IMPLEMENTATION

Implementation will be divided into various components as per the wireframe:

6.1 BACKEND –

- django-admin startproject my_app
- python manage.py createsuperuser
- python manage.py startapp my-app
- python manage.py runserver

DEPENDENCIES INSTALLED

```
asgiref==3.5.0
certifi==2021.10.8
cffi==1.15.0
charset-normalizer==2.0.11
coreapi==2.3.3
coreschema==0.0.4
cryptography==36.0.1
defusedxml==0.7.1
Django==3.2.10
django-cors-headers==3.11.0
django-templated-mail==1.1.1
djangorestframework==3.13.1
djangorestframework-simplejwt==4.8.0
djoser==2.1.0
idna==3.3
itypes==1.2.0
Jinja2==3.0.3
MarkupSafe==2.0.1
oauthlib==3.2.0
psycopg2==2.9.3
psycopg2-binary==2.9.3
pycparser==2.21
PyJWT==2.3.0
python3-openid==3.2.0
pytz==2021.3
requests==2.27.1
requests-oauthlib==1.3.1
six==1.16.0
social-auth-app-django==4.0.0
social-auth-core==4.2.0
sqlparse==0.4.2
tzdata==2021.5
uritemplate==4.1.1
urllib3==1.26.8
```

Figure 6.1.1: Dependencies installed using pip install

6.1.1 AUTHENTICATION

```python
from djoser.serializers import UserCreateSerializer
from django.contrib.auth import get_user_model
User = get_user_model()

class UserCreateSerializer(UserCreateSerializer):
    class Meta(UserCreateSerializer.Meta):
        model = User
        fields = ('id', 'email', 'name', 'password', 'phone')
```

Figure 6.1.1.1: Serializers file for Authentication

Serializers file which is responsible to convert objects to data types which is easily understandable by javascript or front end frameworks.

6.1.2 DATABASE CREATION

```python
from django.db import models

# Create your models here.
class Start(models.Model):
    StartId = models.AutoField(primary_key=True)
    fname = models.CharField('First name' ,max_length = 100)
    lname = models.CharField('Last name' ,max_length = 100)
    business = models.CharField('Business Name' ,max_length = 100)
    address = models.CharField('Address' ,max_length = 100)
    state = models.CharField('State' ,max_length = 100)
    code = models.CharField('Code', max_length=6)
```

Figure 6.1.2.1:Models File for User Registration

Models file where we provide the essential fields and what type of data they are storing. Each model maps to single database table.

```python
    elif request.method == 'POST':
        start_data = JSONParser().parse(request)
        serializer = StartSerializer(data=start_data)
        if serializer.is_valid():
            serializer.save()
            return JsonResponse("Added Successfully", safe=False)
        return JsonResponse("Failed to Add", safe=False)
```

Figure 6.1.2.2: POST Method API

This is the API call method function for POST. This API is called when we need to POST data in database.

6.2 DATABASE –

- Install postgresql from https://www.postgresql.org/
- Create database and set password for your database

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': 'postgres',
        'USER': 'postgres',
        'PASSWORD': 'password12',
        'HOST': 'localhost',
        'PORT': '5432'
    }
}
```

Figure 6.2.1: Postgres Database

Form Data

```
export class Usersform extends Component {
    constructor(props) {
        super(props)
        this.handleSubmit = this.handleSubmit.bind(this)
    }
    handleSubmit(event) {
        event.preventDefault()
        fetch('http://localhost:8000/api/start/', {
            method: 'POST',
            headers: {
                Accept: 'application/json',
                'Content-Type': 'application/json',
            },
            body: JSON.stringify({
                StartId: null,
                fname: event.target.fname.value,
                lname: event.target.lname.value,
                business: event.target.business.value,
                address: event.target.address.value,
                state: event.target.state.value,
                code: event.target.code.value,
            }),
        })
            .then((res) => res.json())
            .then(
                (result) => {
                    alert(result)
                    window.location.reload(true)
                },
                (error) => {
                    alert('Failed')
                }
            )
    }
```

Figure 6.2.2: Method to POST data in Database

It is used to post the data in database when the user fills the particular fields. Data gets stored in the database.

Tables Data

```
export class PaginationTable extends Component {
    constructor(props) {
        super(props)
        this.state = { deps: [], addStartShow: false, editStartShow: false }
    }

    refreshList() {
        fetch('http://localhost:8000/invoice/')
            .then((response) => response.json())
            .then((data) => {
                this.setState({ deps: data })
            })
    }

    componentDidMount() {
        this.refreshList()
    }
    deleteDep(VendorId) {
        if (window.confirm('Are you sure?')) {
            fetch('http://localhost:8000/invoice/' + VendorId, {
                method: 'DELETE',
                header: {
                    Accept: 'application/json',
                    'Content-Type': 'application/json',
                },
            })
```

Figure 6.2.3: Method to GET data from Database

It is used to fetch the data from the database and also helps to delete data. Data is deleted on the basis of ID.

6.3 FRONTEND –

To create a new React project:

- npx create-react-app my-app
- cd my-app
- npm start

DEPENDENCIES INSTALLED

Figure 6.3.1: Dependencies installed using npm

6.3.1 ON-BOARDING PAGES (FRONTEND)

Get Started Page:

The page is build using ReactJS, new business register themselves as new user and their business by filling up the form to proceed to next page. Without the user registration the one can't proceed to next page.

```
<Form.Group controlId="fname">
    <Form.Control
        //  className=" mb-2 p-2"
        style={{
            width: '80%',
            marginBottom: '16px',
            borderRadius: '20px',
            marginTop: '5px',
        }}
        size="lg"
        type="text"
        placeholder="First Name"
        name="Name"
        required
    />
</Form.Group>
```

Figure 6.3.1.1: Implementation of Form

To create the form "Form-Group" & "Form-Label" React Bootstrap is used.

On Boarding Page:

After registering itself and their business, they proceed to this page and in this page we have provided the users two option "Send Invoices" which direct the user to main dashboard and help to get the information of its details and second "File GST Return" this direct the user to pricing plan page where user can opt for any plan and file GST.

6.3.2 DASHBOARD IMPLEMENTATION (FRONTEND)

Dashboard Theme is build using Material UI kit. It include the sidebar navigation which provide the react routes to all the components including the form, tables, cards, & the pricing plan.

The “Grid”, “Icon”, “Box”, “Tooltip”, etc is imported from material ui.

```
import { Grid, Card, Icon, IconButton, Tooltip } from '@mui/material'
```

Figure 6.3.2.1: Imported from Material UI

```
<Grid item xs={12} md={6}>
    <StyledCard elevation={6}>
        <ContentBox>
            <Icon className="icon">group</Icon>
            <Box ml="12px">
                <Small>TOTAL USERS</Small>
                <Heading>{props.count}</Heading>
            </Box>
        </ContentBox>
        <Tooltip title="View Details" placement="top">
            <IconButton>
                <Link to="/users">
                    <Icon>arrow_right_alt</Icon>
                </Link>
            </IconButton>
        </Tooltip>
    </StyledCard>
</Grid>
```

Figure 6.3.2.2: Stat Cards implementation

For better data visualization, we have added two graphs “Doughnut” and “Line” Graph. It’s designed with material ui and created using “Echarts”.

Doughnut Graph

import ReactEcharts from 'echarts-for-react';

```
<ReactEcharts
    style={{ height: height }}
    option={{
        ...option,
        color: [...color],
    }}
/>
```

Figure 6.3.2.3: Implementation of ReactEcharts component

This Code is responsible for showcasing the data inside the Graph.

It will also have some default code from react echarts to form the graph which includes the scales.

Line Chart

import ReactEcharts from 'echarts-for-react';

It will also have some default code from react echarts to form the Line Chart which includes the scales (xAxis, yAxis).

On xAxis – Months On yAxis – Values

6.3.3 TABLE & FORM IMPLEMENTATION (FRONTEND)

This section is the most important part of the website. All the users, customers, vendors can add their data by filling up the forms and able to see the other person details/ information in table.

```
<SimpleCard title="Create Vendor">
    <SimpleVendForm />
</SimpleCard>
<Box py="12px" />
<SimpleCard title="Current Vendors">
    <VendorTable />
</SimpleCard>
```

Figure 6.3.3.1: Form and Table display

Form Component:

The forms are designed using React Bootstrap.

There are 4 categories of forms, the user can fill the form to get there product displayed to other user and interact with other businesses.

The entered details by the user get stored using a value or Control Id which take the string input and store in state and later connect all the submitted information to the database and display it on table.

Table Component:

The tables are designed using React Bootstrap.

```
{deps.map((dep) => (
    <TableRow key={dep.VendorId}>
        <TableCell>{dep.VendorId}</TableCell>
        <TableCell>{dep.Name}</TableCell>
        <TableCell>{dep.Address}</TableCell>
        <TableCell>{dep.GST}</TableCell>
        <TableCell>{dep.Email}</TableCell>
        <TableCell>{dep.Pan}</TableCell>
        <TableCell>{dep.Contact}</TableCell>
        <TableCell>{dep.Bank}</TableCell>
        <TableCell>
```

Figure 6.3.3.2: Table connected to DB using API

We have also provided with Edit and Delete functionality. The user can edit its data and delete it whenever they want.

```
<EditVendor
    show={this.state.editStartShow}
    onHide={editStartClose}
    Id={Id}
    name={name}
    add={add}
    gst={gst}
    email={email}
    Pan={Pan}
    contact={contact}
    acc={acc}
/>
```

Figure 6.3.3.3: Edit and delete

editStartClose() is func which is setting the editStartShow to false at default.

Inside the body we have added the form.

It is also connected to the database. And it is taking the default value from previous entered data using props.

```
<Form.Group controlId="VendorId">
        <Form.Label>Sr. No.</Form.Label>
        <Form.Control
            type="text"
            name="id"
            required
            defaultValue={this.props.Id}
            placeholder="ID"
        />
    </Form.Group>
```

Figure 6.3.3.4: Modal Body form

On submitting the form it will update your data in database and also on the visual table.

The API is using the id to connect the state to DB.

```
fetch('http://localhost:8000/vendor/')
.then((response) => response.json())
.then((data) => { this.setState({ deps: data })  })
```

Chapter 7

SCREENSHOTS

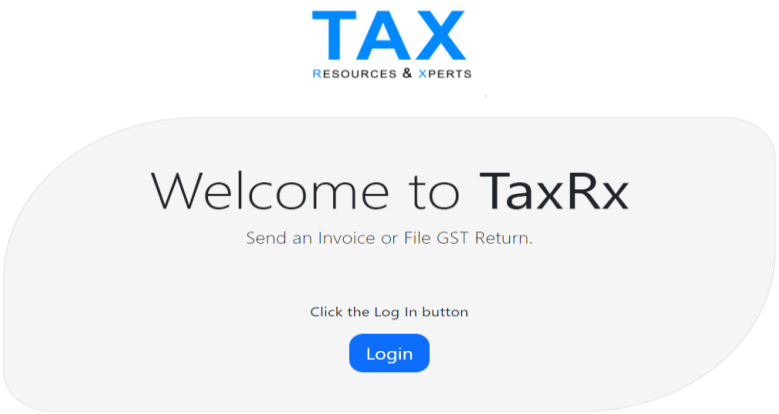

Figure 7.1: Front Page

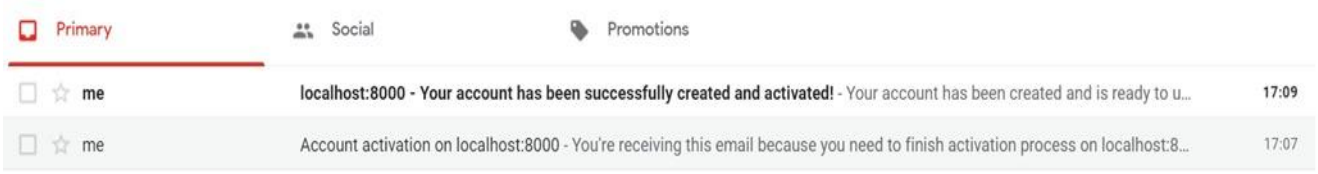

Figure 7.2: Confirm the Email

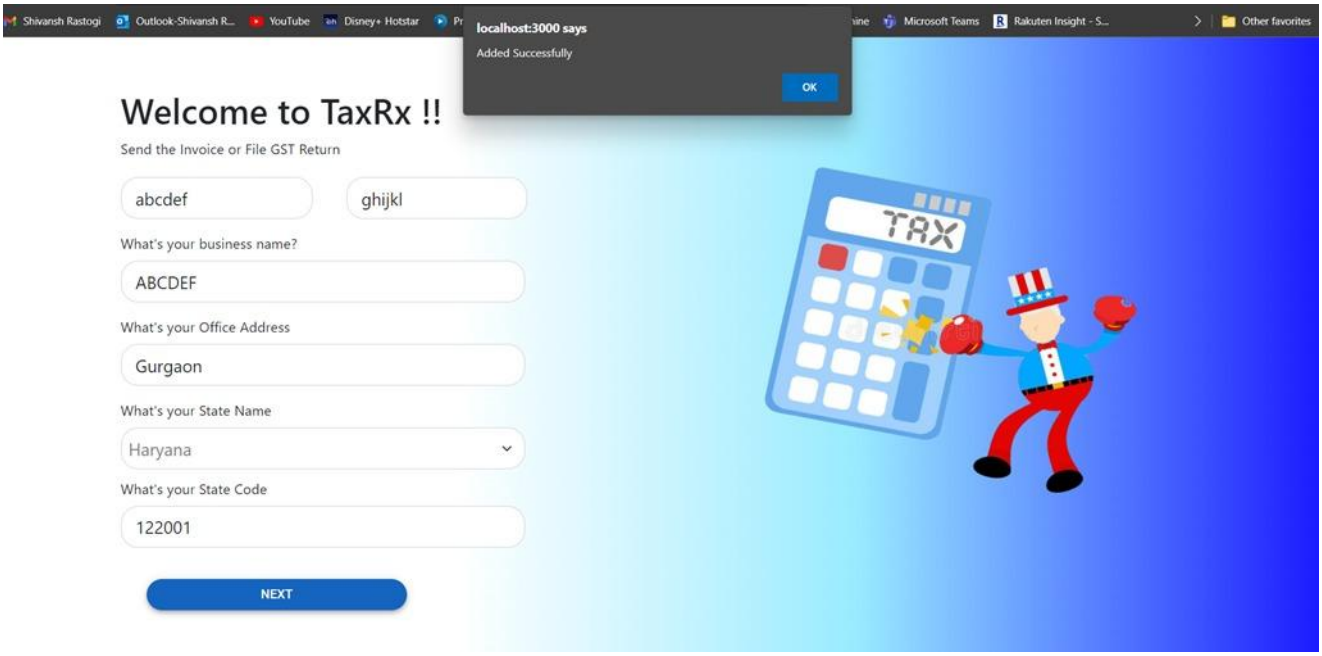

Figure 7.3: Launchpad Page

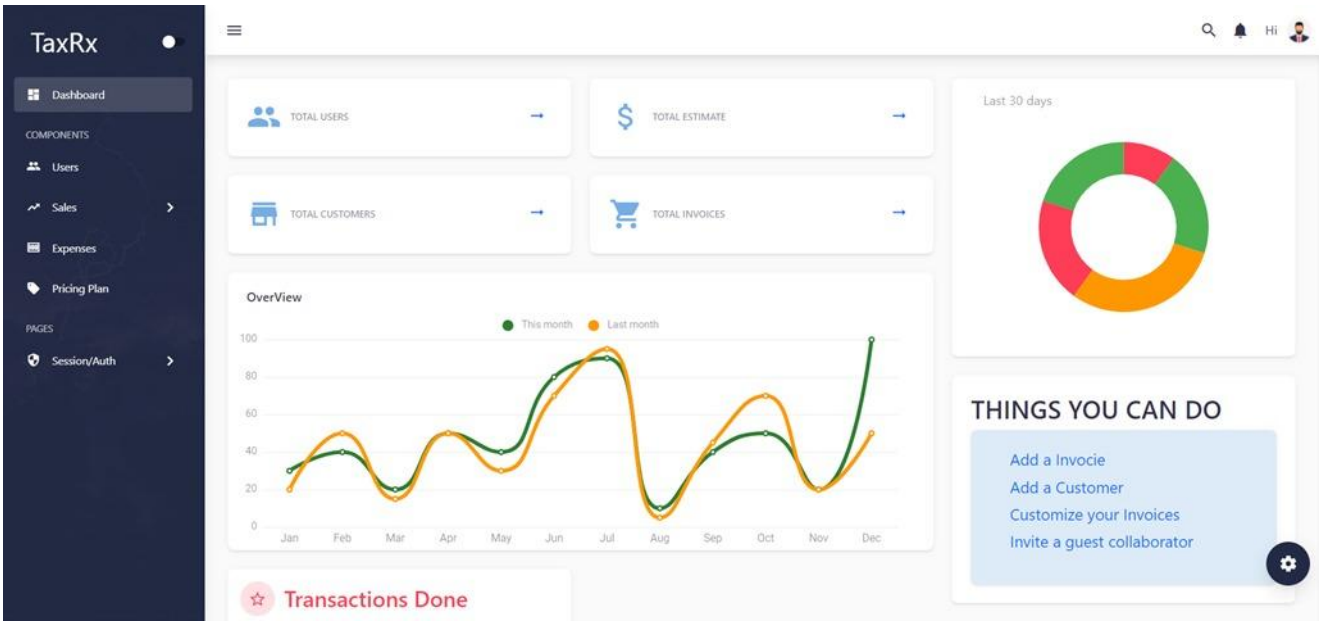

Figure 7.4: Dashboard

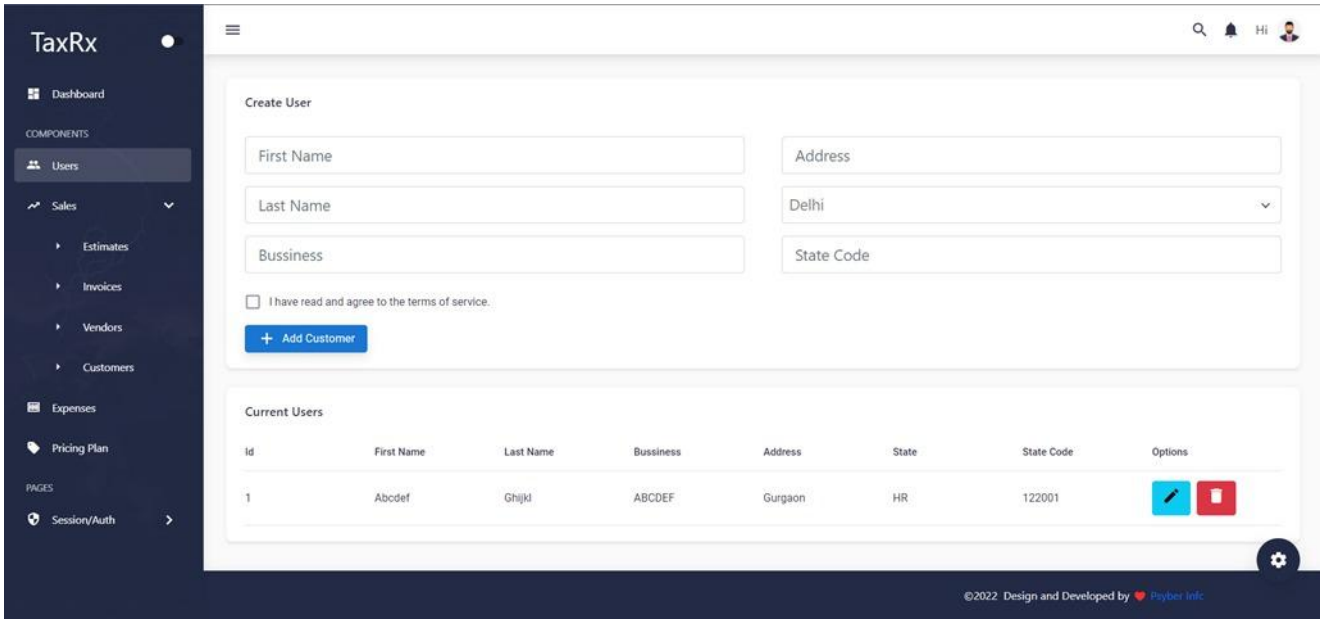

Figure 7.5: Form & Table

Chapter 8

SCOPE OF A PROJECT

"TaxRx" is a full stack web application which is used to send, store the invoice and file GST return. So to provide a platform where all these things can be done is very hard to find.

It will provide the Young Businesses a platform which helps to bridge the gap between the customers and vendors. Management of customers and vendor's data will become convenient.

For more improvements in the future we are planning to provide a secure and authenticated payment gateway for both the credit/debit card or UPI payment gateway.

In future we can implement a structure which will maintain users activity session and based on that the system will recommend them the usage of their website for the good.

Technologies used like "echarts" that provide great data visualization will expand more in future for better understanding of data.

Chapter 9

CONCLUSION

As, the previous website of TaxRx was fully static with all the data and credentials stored already on the website. There was no authentication of the user. Dashboard was also very simple with just 4 cards holding total users, total customers, total invoices, and total estimates with side navbar have home, invoices, users, customers, vendors etc. Only admin was able to login to that website.

Our current application “TaxRx” comes handy as it provides the user to file GST return or send the invoice. With the better user experience and modern interface. It offers statistics and data visualization on invoices and GST return file in the form of tables, interactive line graphs, doughnut graph visualization, create invoice, customer estimate, expense and vendors. A user can register himself to the website and then the user can choose to either send the invoice to someone or can file a GST return. This web page gives eye-catching data visualization by displaying the total expenses and on what category payment was done by using ‘Echarts’ to create visually appealing, interactive and easy to understand line graph and doughnut graph. A user can perform CRUD(Create, Read, Update, Delete) operations and all his credentials, data is stored in a database “Postgres SQL” and displayed in the form of tables on website. This makes a better user friendly experience.

REFERENCES

1. https://www.django-rest-framework.org/
2. https://react-bootstrap.github.io/
3. https://github.com/mui/material-ui
4. https://www.djangoproject.com/
5. https://react-bootstrap.github.io/
6. https://mui.com/
7. https://www.computerhope.com/jargon/j/javascript.htm
8. https://en.wikipedia.org/wiki/React_(JavaScript_library)
9. https://en.wikipedia.org/wiki/Python_(programming_language)
10. https://www.twilio.com/blog/2018/05/build-chat-python-django-applications-programmable-chat.html
11. https://www.ovhcloud.com/asia/public-cloud/postgresql/
12. https://zapier.com/learn/apis/chapter-2-protocols/
13. https://taxrx.in/
14. https://echarts.apache.org/en/index.html
15. https://www.freecodecamp.org/news/meet-your-material-ui-your-new-favorite-user-interface-library-6349a1c88a8c/
16. https://github.com/psycopg
17. https://www.educative.io/blog/react-bootstrap-tutorial
18. https://www.django-rest-framework.org/
19. For HTML, CSS, JS, ReactJS: https://en.wikipedia.org/wiki/
20. Wireframe: https://app.diagrams.net/
21. https://www.w3schools.com/jsref/met_loc_reload.asp#:~:text=Window%20location.&text=The%20reload()%20method%20reloads,reload%20button%20in%20your%20browser.
22. https://restfulapi.net/
23. https://snipcart.com/blog/javascript-practice-exercises
24. https://novicedeveloper.com/what-is-rest-api/