Internship / Training Report

Major Project report submitted in partial fulfillment of the requirement
for the degree of Bachelor of Technology

in

Computer Science and Engineering/Information Technology

By

Isha Anand-181473

Under the supervision of

Dr. Himanshu Jindal

to

Department of Computer Science & Engineering and Information
Technology
Jaypee University of Information Technology Waknaghat,
Solan-173234, Himachal Pradesh

Certificate

Candidate's Declaration

I hereby declare that the work presented in this report entitled **"Internship /Training Report"** in partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science and Engineering/Information Technology** submitted in the department of Computer Science & Engineering and Information Technology**,** Jaypee University of Information Technology Waknaghat is an authentic record of my own work carried out over a period from February 2022 to May 2022 under the supervision of **Mr. Aman Choudhary** Assistant Manager ,Security Software Development ,RTDS. The matter embodied in the report has not been submitted for the award of any other degree or diploma.

Isha Anand

Isha Anand
181473

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

Himanshu Jindal

Dr. Himanshu Jindal
Assistant Professor
CSE & IT
Dated: 27/05/2022

Aman Choudhary
27/05/22

Aman Choudhary

Assistant Manager

Security SoftwareDevelopment

RTDS

Dated: 27/05/2022

ACKNOWLEDGEMENT

This is a matter of pleasure for me to acknowledge my deep sense of gratitude to Jaypee University and my college, Jaypee university of information technology for giving me an opportunity to explore my abilities via this internship programme. I would like to express my sincere gratitude to our TnP officer, **Mr. Pankaj Kumar** and our faculty Coordinator, **Dr. Nafis U Khan** for this opportunity. I also wish to express my gratitude to my internship supervisor **Dr. Himashu Jindal**, for their valuable guidance and advice in completing this project.

I would like to record my sincere appreciation and gratitude towards all the officials and employees of RTDS, without whose kind assistance, my internship program would not have succeeded. The facts and other vital information provided by them have contributed towards making this report as comprehensive as possible. I am indeed thankful to them.

Last but not the least, I would like to express my sincere thanks to all my family members, friends and well-wishers for their immense support and best wishes throughout the internship duration and the preparation of this report.

I believe that this report will be a valuable asset not only for academic institution, but will also be useful for all those who are interested to learn about internship experiences in global information technology, consulting, and outsourcing company serving clients in multiple locations across the globe.

TABLE OF CONTENTS

List of Abbreviations

1. RTDS - Real Time Data Services
2. SIEM - Security Information and Event Management
3. API - Application Programming Interface
4. DB - Database
5. JS - JavaScript
6. AI -Artificial Intelligence
7. RBF -Radial Basis Function

List Of Figures

ABSTRACT

After the end of 7 th semester, various company visited to our college for the placement of the student, one such company was RTDS, due to my good fortune, I was selected for the Software Developer profile, after selected as SDE I was offered a Full time Offer by RTDS after joining we were given training on various aspects related to our domain which is further discussed in the report. The internship was of around 12 weeks containing various sessions, webinar, online udemy courses, assessment and project.

CHAPTER 1- ORGANISATION

1.1 About the Company

Real Time Data Services (RTDS) is a group of companies thriving in the domain of global information technology by serving clients in the field of Cloud Computing and Communication. The company empowers businesses across the globe with technology solutions that take care of their various operations.

RTDS expertise in cloud computing & IT allows us them to deliver solutions that simplify the business and enhance their performance.

RTDS provide the following services to our clients:

- Accounting and business applications on the cloud
- Server virtualization and consolidation solutions for building private clouds
- Cloud-based communication services to connect all employees and departments on any device
- Provide a lean, scalable, and cost-effective telecommunication solution to help businesses connect to the right people easily

Company Journey

- ➢ RTDS started it’s operation in 2010
- ➢ In 2011 Partnered with data centres in Buffalo, Chicago, Dallas, Seattle, New York, and Phoenix to provide seamless hosting experience and 24*7 Support
- ➢ In 2012 Cloud Telephony services started in USA
- ➢ Became Intuit authorised hosting provider and built strategic partnerships with Salesforce
- ➢ In 2016 Cloud Telephony Services launched in UK
- ➢ In 2017 Felicitated with User Favorite Award at AccountexUSA for QuickBooks Hosting
- ➢ In 2018 Launched Cloud Telephony Services in India

- ➢ In 2019 Achieved subscription of 2000 business accounts in the USA for hosting of accounting and business applications
  Onboarded 1300 small and medium enterprises in India for Cloud Communication Solutions
  In 2019, RTDS group is targeting a revenue of 100 Crores and launching operations in 3 countries

1.2 Mission & Values

Mission

To become the forefront of Tech Global Leaders in Cloud Computing by striving towards excellence in our robust products & Services, providing a competitive edge to our customers.

Values

Striving for Excellence :Believing in consistently improving the quality of services that we offer

Result Orientation :Developing leaders and instilling innovation by creating a result-oriented atmosphere

Integrity:Uncompromising adherence to strong moral and ethical principles and values

Collaboration:We believe in working together and produce unmatched results

1.3 Products & Solution

- Cloud Hosting

  The custom-made cloud solutions that ensures the server specifications and core technicalities of the IT backend.

Cloud accounting solutions for every scale of business.
Cloud POS solutions for retailers ensure efficient inventory management of multiple stores from any location.
Tax-specific cloud solutions offer a secure and centralized environment for tax preparers to prepare and file client's tax returns.

- Cloud Telephony
  Various cloud-based solutions catering to the inbound and outbound calling needs of the
  call centres.
  Business contact numbers with a user-friendly portal to support customer traction and steady business communication.
  Easy to use PBX for an organization or company with various features and customizable options to manage extensions.
  Outbound dialers for outreaching customers with powerful data insights and analytics ensuring utmost agent utilization.

1.4 Corporate Social Responsibility

Ensuring socio-economic development of the community through different participatory and need- based initiatives in the best interest of the poor and deprived sections of the society, so as to help them to become SELF-RELIANT and build a better tomorrow for themselves. Our commitments towards Corporate Social Responsibility include but not limited to, promotion development of under privileged, providing education and training, upskilling the workforce, betterment of the society through respect for universal human rights and the environment, providing medical relief, eradicating hunger, poverty and malnutrition, promoting health care, preventive health care, acting with integrity and accountability and operating responsibly and sustainably

CHAPTER 2- INTERNSHIP

After joining I was assigned to the Security Software Development team where we were working on a new product which is SIEM Security Information and Event Management. We have to learn various Tech stacks and tools and do various hands-on practice and tasks during the training period.

2.1 Tech Stacks:

We introduced to various Technical tools and programming language as a part of our rigorous training program .some of them are mentioned below:

- SIEM
- MonogDB
- Kibana
- Logstash
- ElasticSearch
- JavaScript
- Type Script
- React js
- Postman
- Bitbucket
- Git

Week 1:

In the first week we had our orientation Program where we were introduced to:

- Company Background
- Culture
- Vission & Mission.

We had few interactive session with employees of the firm just to get a better view of the firm.

We had a CEO talk where CEO of the firm himself interactd with the new joiners.

We were assigned to a team and had an interactive session with our manager.

WEEK 2:

Since I was assigned into the security Software team . I was SIEM i.e Security Information and Event Management the product on which we will be working on.

SIEM :

SIEM stands for Security Information and Event Management

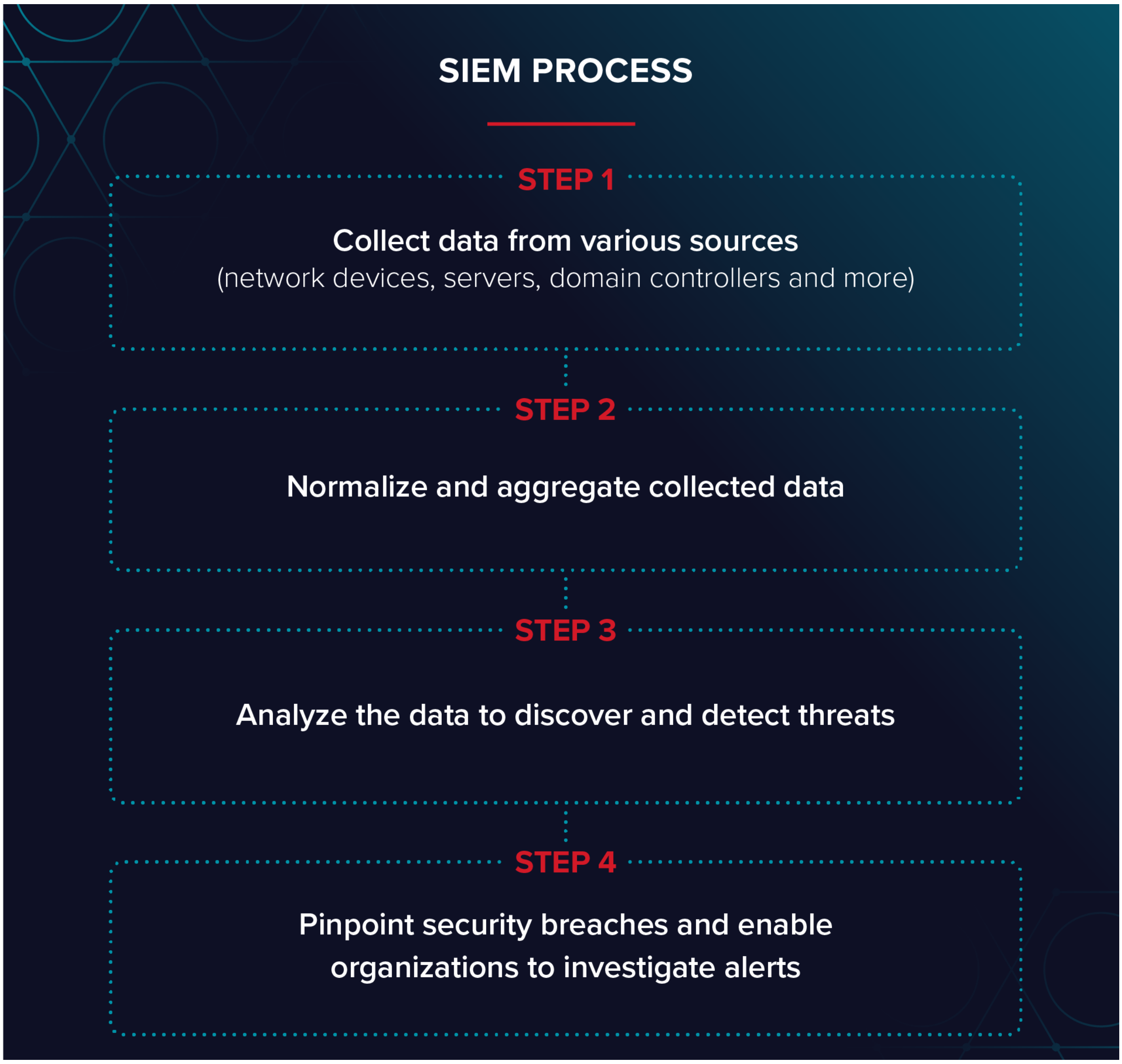
SIEM is a software program that collects and aggregates all the log information generated throughout the organisation's technology infrastructure, from host systems and programs to community and security gadgets together with firewalls and antivirus filters.

SIEM recognizes and classifies occurrences and occasions, as well as examines them. SIEM gives two essential abilities, for example,

Give investigates security-related occurrences and occasions, for example, fruitful and fizzled logins, malware action and other potentially malevolent exercises.

Furthermore, send cautions assuming examination shows that an action goes against foreordained rulesets and demonstrates a potential security issue.

SIEM is an information aggregator, search, and revealing framework. SIEM assembles enormous measures of information from the whole arranged climate, merges and makes that information human accessible. .



SIEM PROCESS

STEP 1

Collect data from various sources
(network devices, servers, domain controllers and more)

STEP 2

Normalize and aggregate collected data

STEP 3

Analyze the data to discover and detect threats

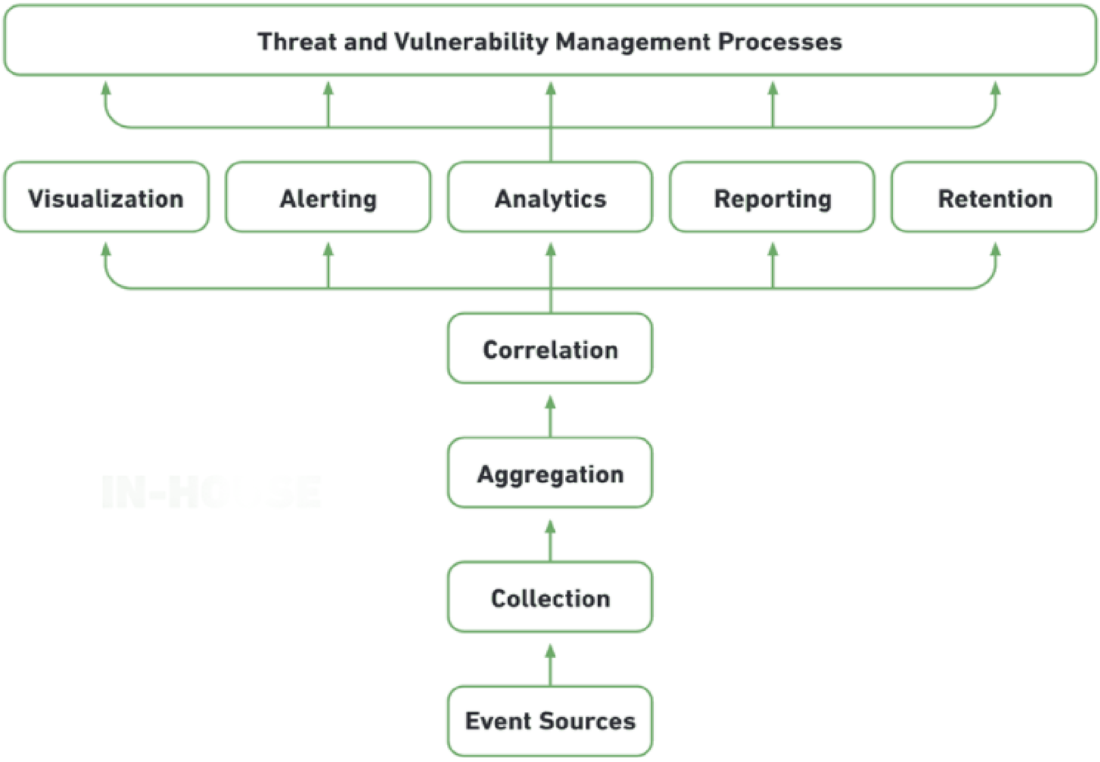
STEP 4

Pinpoint security breaches and enable organizations to investigate alerts

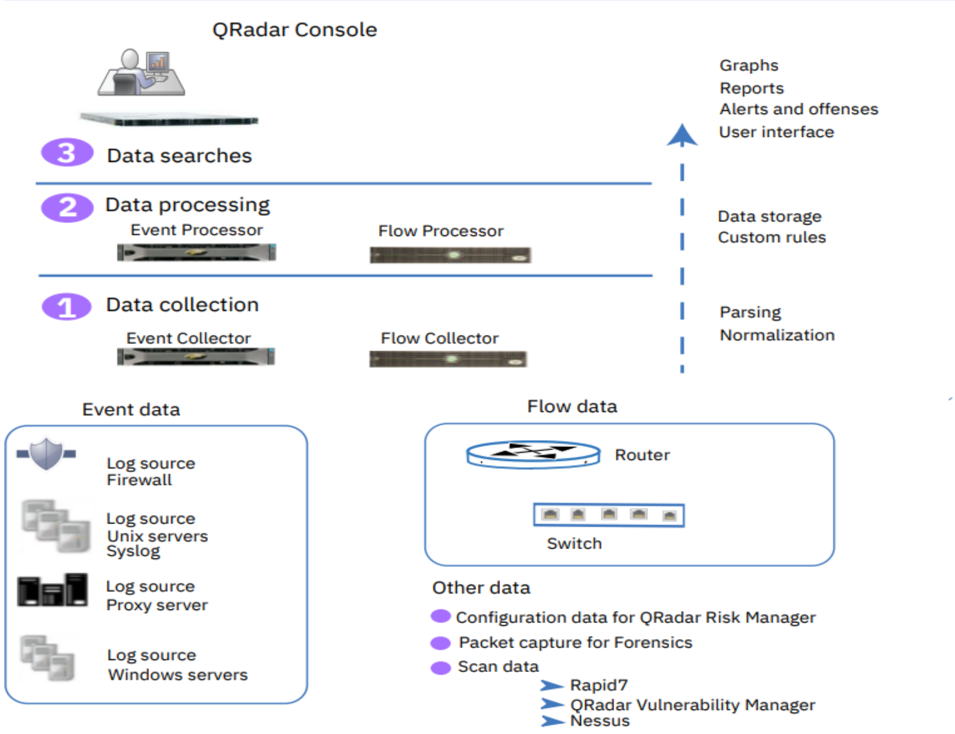
The way to SIEM Success looks something like this:

- Gather logs from standard security sources
- Enhance logs with supplemental information.
- Worldwide Threat Intelligence (Black Lists).
- Human Resource/Internet Download Management.
- Correspond — tracking down the so-called needles in the log bundles.
- Research — follow up and fix.
- The report — Standard Operating Procedures, Service Level Agreements, Trouble Tickets.
- Consolidate — Build white records, new happy.

Various Steps Involved :



The Siem Architecture:



WEEK 3:

Another Database tool we were introduced was MONGO DB.

MONGO DB :

Mongo db is:

- NoSQL database.
- Open-source
- document-oriented database
- written in C++.
- uses BSON to Query database.

Features of MongoDB :

- A Relational data set has a commonplace construction plan that shows number of tables and the connection between these tables, while in MongoDB there is no understanding of relationship

.

- MongoDB is Schema less. It is a report data set in which one assortment holds various records.
- There are no perplexing participates in MongoDB.
- MongoDB is just multiple times quicker than conventional information base frameworks.
- As a report data set, MongoDB makes it simple for engineers to store organized or unstructured information.
- MongoDB is more adaptable and guarantees high and different information accessibility

Database :

In MongoDB, databases hold one or more collections of documents.

If a database does not exist, MongoDB creates the database when you first store data for that database.

```
use myNewDB
```

```
db.myNewCollection1.insertOne( { x: 1 } )
```

The insertOne() operation creates both the database myNewDB and the collection myNewCollection1if they do not already exist.

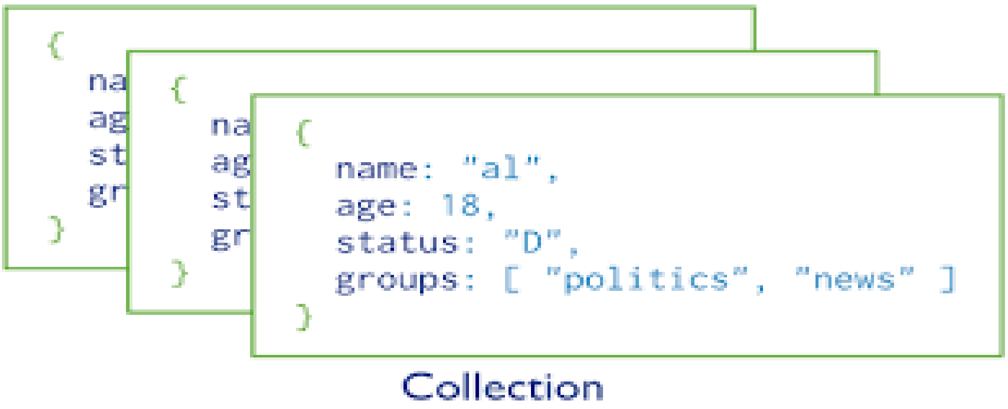
Collection :

Collection is a group of MongoDB documents.

If a collection does not exist, MongoDB creates the collection when we first store data for that collection.

db.myNewCollection2.insertOne({ x: 1 })

db.myNewCollection3.createIndex({ y: 1 })



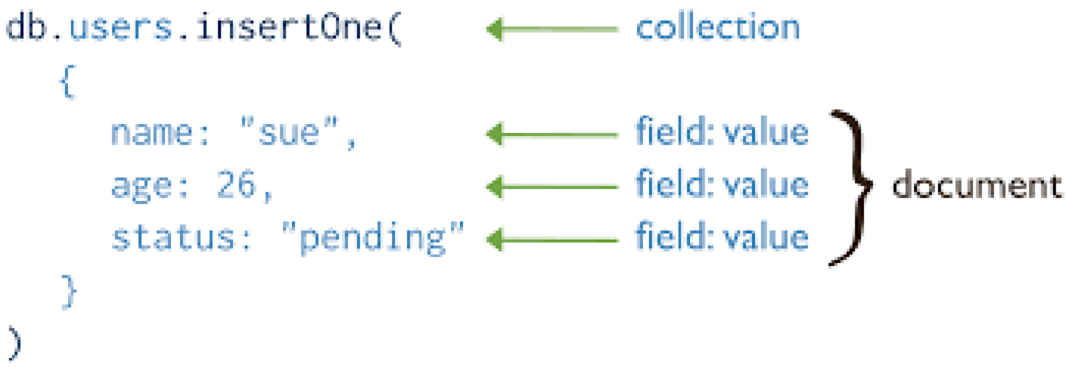
```
{
  name: "al",
  age: 18,
  status: "D",
  groups: [ "politics", "news" ]
}
```

Document :

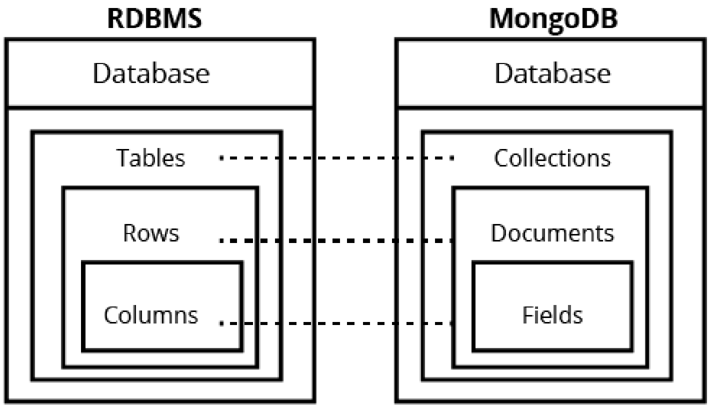
A document is a set of key-value pairs.

documents in the same collection do not need to have the same set of fields or structure,

a collection's documents may hold different types of data.

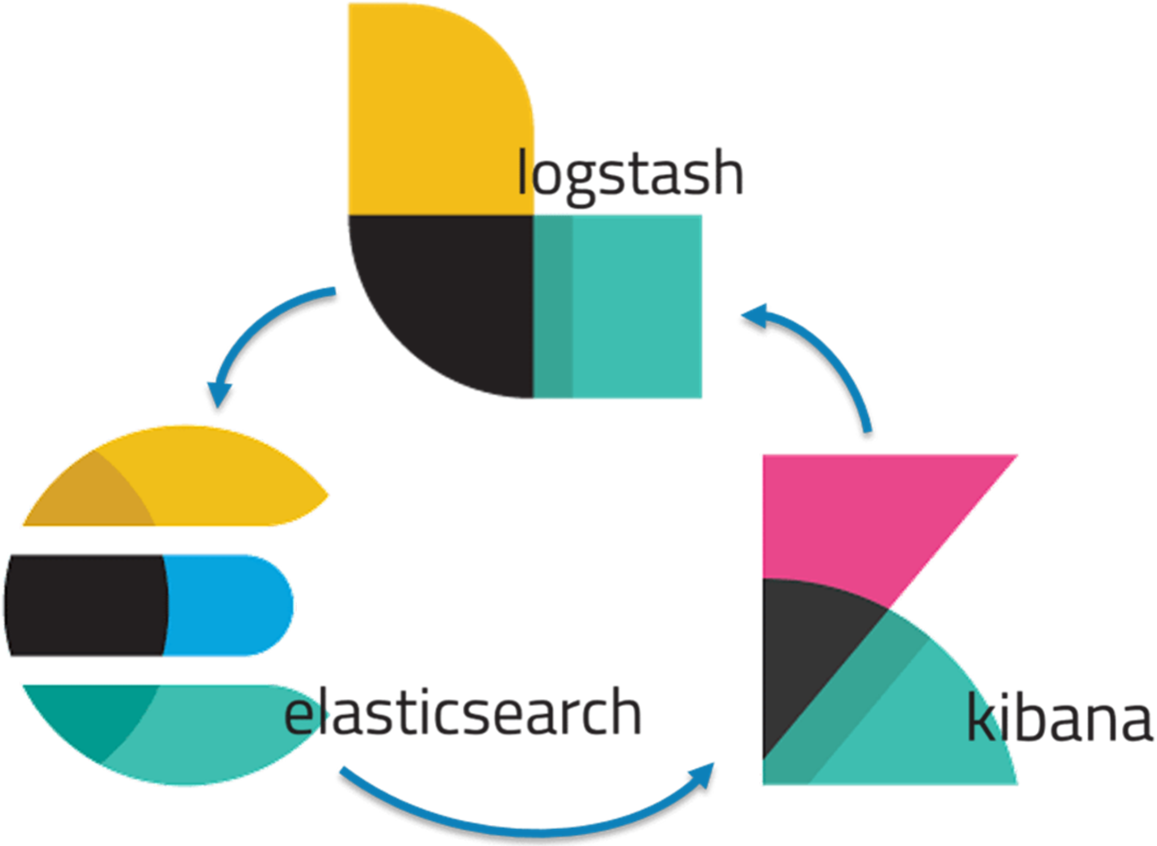


```
db.users.insertOne(
   {
      name: "sue",
      age: 26,
      status: "pending"
   }
)
```


RDBMS
MongoDB
Database
Database
Tables
Collections
Rows
Documents
Columns
Fields

WEEK 4:

We were introduced to ELK Stack where E is for Elastic Search , L stands for Logstash and K for Kibana



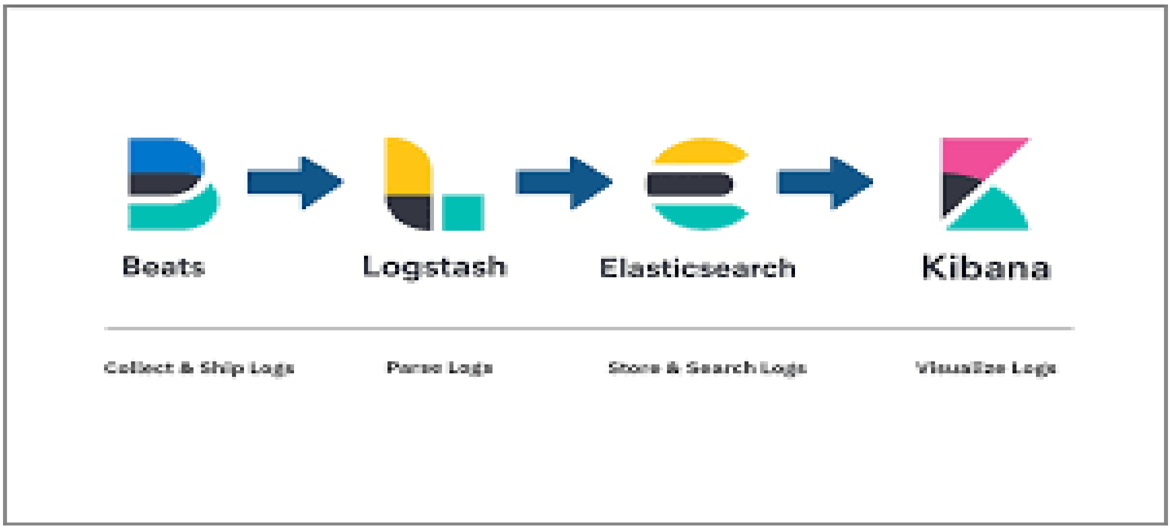
Kibana :

Kibana is a free and open UI that allows you to imagine your Elasticsearch information and explore the Elastic Stack.

Kibana center boats with the works of: histograms, line diagrams, pie graphs, sunbursts, and that's just the beginning. Furthermore, obviously, you can look across your records as a whole.

How Kibana Works :

- Add Data
- Explore Data
- visualise Data
- Model Data Behaviour
- Share the final result



Role of Kibana :

- Enables the searching and interaction with data in Elasticsearch.
- Enables creation and sharing of dynamic dashboards that get updated in real-time.
- Allows performing advanced analytics and creation of reports

Why to use Kibana

- Kibana is the official interface of Elasticsearch
- Kibana addresses various use cases
- Kibana has a strong support community
- Kibana is Open Source

WEEK 5 :

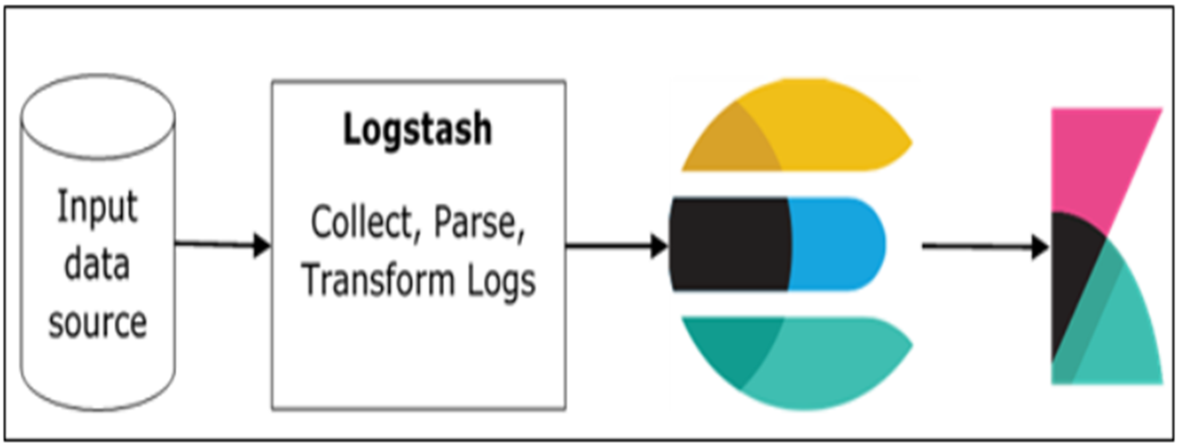
We were introduced to Logstash

Logstash :

Logstash is a device in view of the channel/pipes designs for logs, handling and creating the logs or occasions. It helps in unifying and making continuous examination of logs and occasions from various sources.

Written in Jruby logstash runs over Java Virtual Machine.

It gathers various kinds of information Packets, Events, Transactions, Timestamp Logs ,Data, and so on, from pretty much every kind of source. The information source can be Social information, E-trade, News articles,Internet of Things Game information, Web patterns, Financial information, Mobile gadgets, and so on.

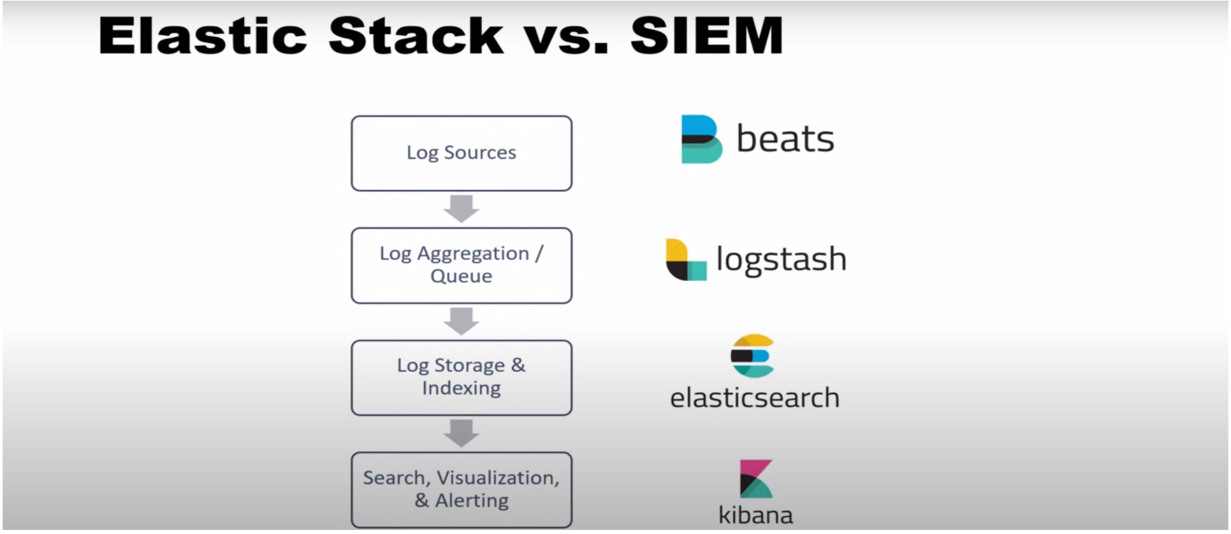


WEEK 6:

We were introduced to Elastic Search

Elastic Search:

Elastic Stack is a collection of opensource items intended to assist clients with taking information from a source and in any configuration and search, break down, and Visualize that information progressively. The item bunch was previously known as ELK Stack,



Elastic Stack vs. SIEM

Log Sources → Log Aggregation / Queue → Log Storage & Indexing → Search, Visualization, & Alerting

beats

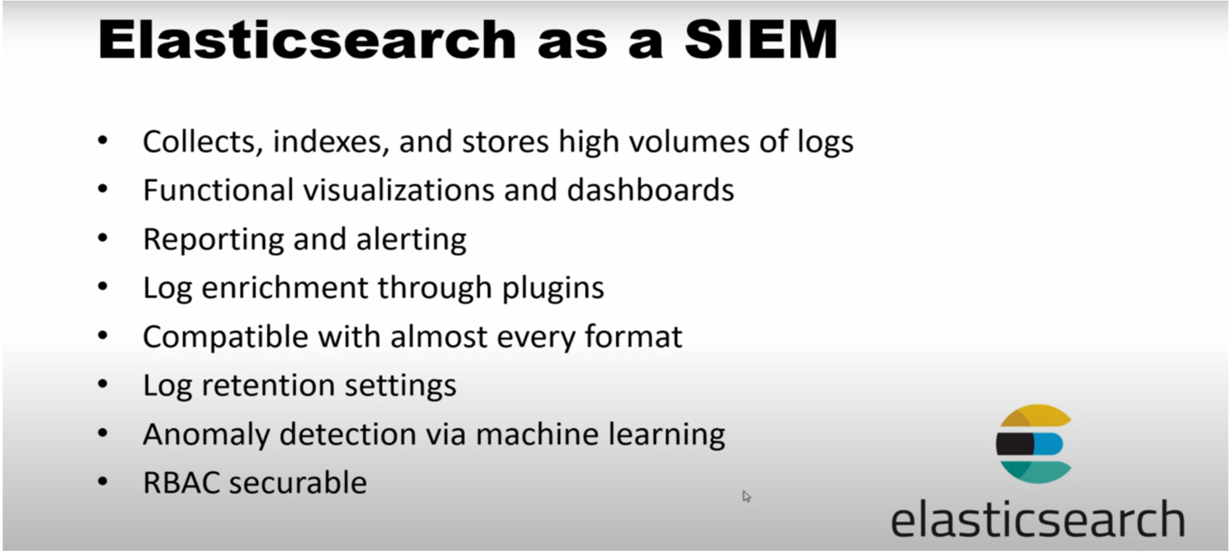
logstash

elasticsearch

kibana

Elasticsearch is a disseminated, free and open Source .

It is considered as analytical engine for a wide range of information, including literary, mathematical, organized, and unstructured. Elasticsearch is based on Apache Lucene and was first delivered in 2010.



Elasticsearch as a SIEM

- Collects, indexes, and stores high volumes of logs
- Functional visualizations and dashboards
- Reporting and alerting
- Log enrichment through plugins
- Compatible with almost every format
- Log retention settings
- Anomaly detection via machine learning
- RBAC securable

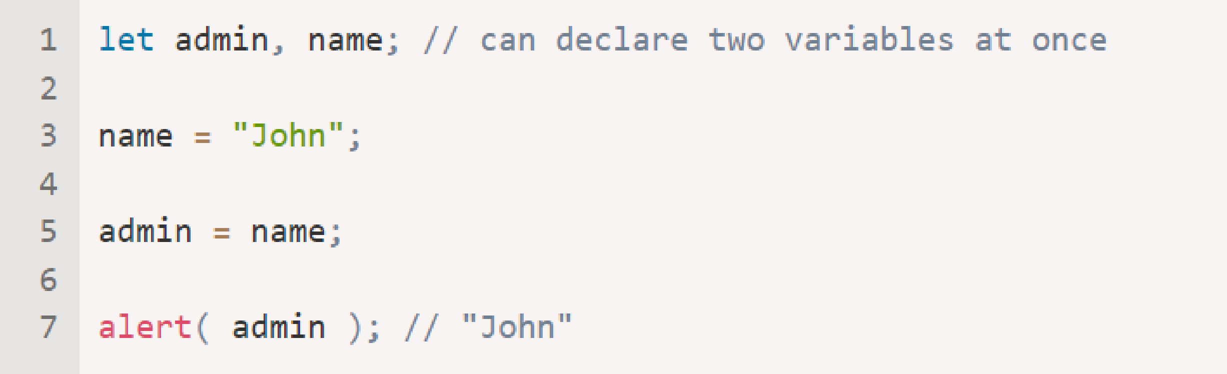
elasticsearch

WEEK 7 - WEEK 10 :

In this Time period we learned about JavaScript and learned about using post man and send GET ,PUT,POST ,DELETE requests

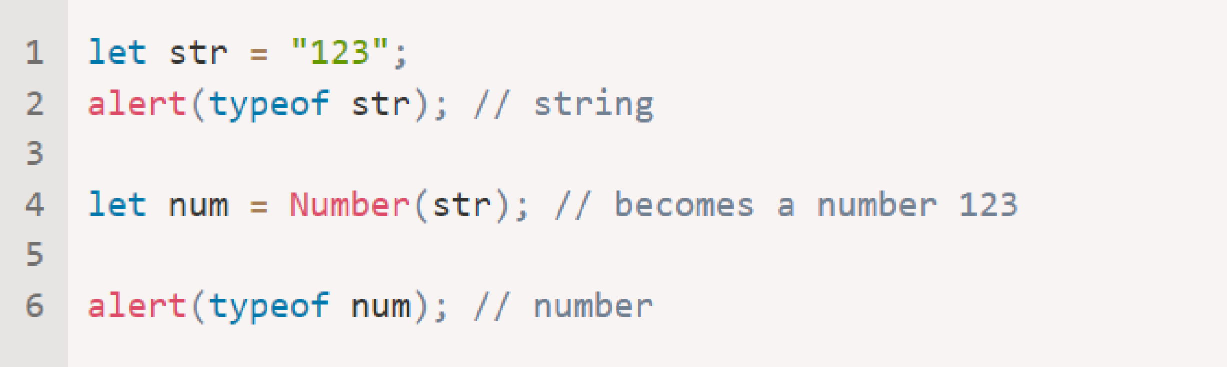
JAVASCRIPT

In javascript var ,Let and const are different ways of creating variables.use let if you want to create a variable that really is variable. Use const if you plan on creating a constant value, so something which you only assign once and never change.var is the old way of creating variable.



```
let admin, name; // can declare two variables at once

name = "John";

admin = name;

alert( admin ); // "John"
```

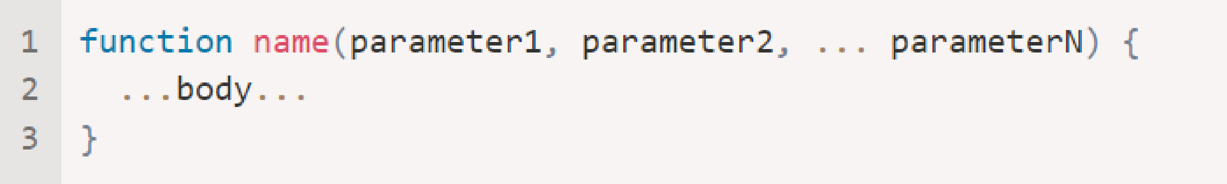
In javascript all the functions and operators naturally convert the value given to then to their right data type.



```
let str = "123";
alert(typeof str); // string

let num = Number(str); // becomes a number 123

alert(typeof num); // number
```

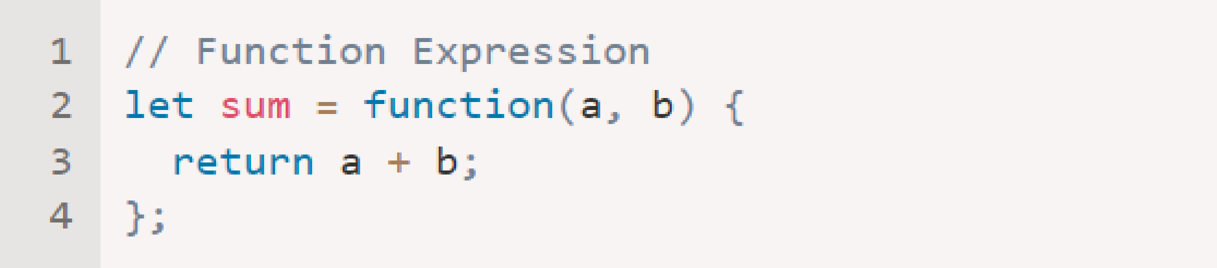
Strict equality check "===" compare data as well as data type ,where as non-strict check "==" only compare data.

Functions in javascript are declared using "function" keyword along with the name of function.



```
function name(parameter1, parameter2, ... parameterN) {
  ...body...
}
```

Function expression is a function created on the right side of the "assignment expression" =: " and assigned to a variable.



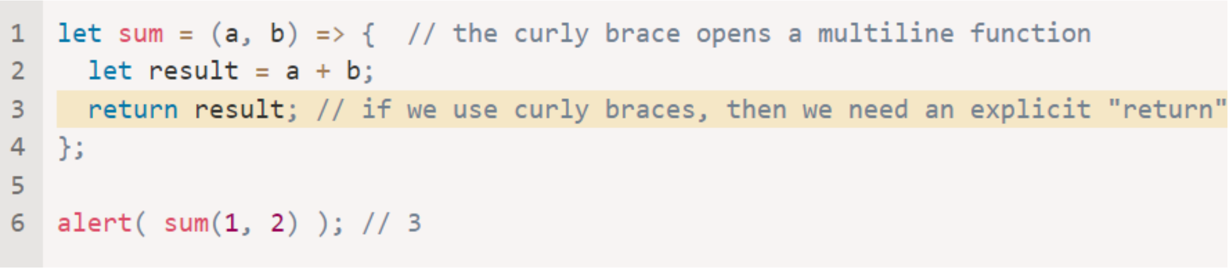
```
// Function Expression
let sum = function(a, b) {
  return a + b;
};
```

A function expression cannot be called earlier than it is defined because it is created at the moment execution reaches it whereas function declaration can be allied earlier in the program .

The arrow function syntax is a bit shorter than the normal syntax since it omits the function keyword .

Arrow Function are similar to function expression but a better version .The function name and arguments are on the the left of =>, function body is on the right.

Evaluated result is returned and stored in the variable on the left side of =>

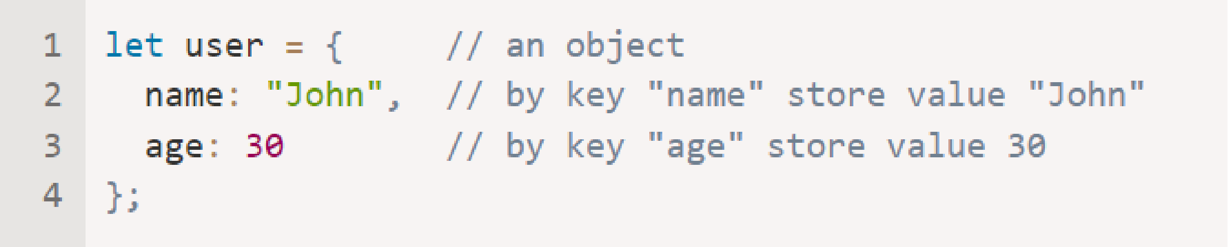


```
let sum = (a, b) => {  // the curly brace opens a multiline function
  let result = a + b;
  return result; // if we use curly braces, then we need an explicit "return"
};

alert( sum(1, 2) ); // 3
```

An object is a data type in Javascript which can be used to store multiple data at once , it can be created using curly braces {} and stores data as key:value pair .

Properties (key:value) can be accessed using dot notation object.key or square bracket notation

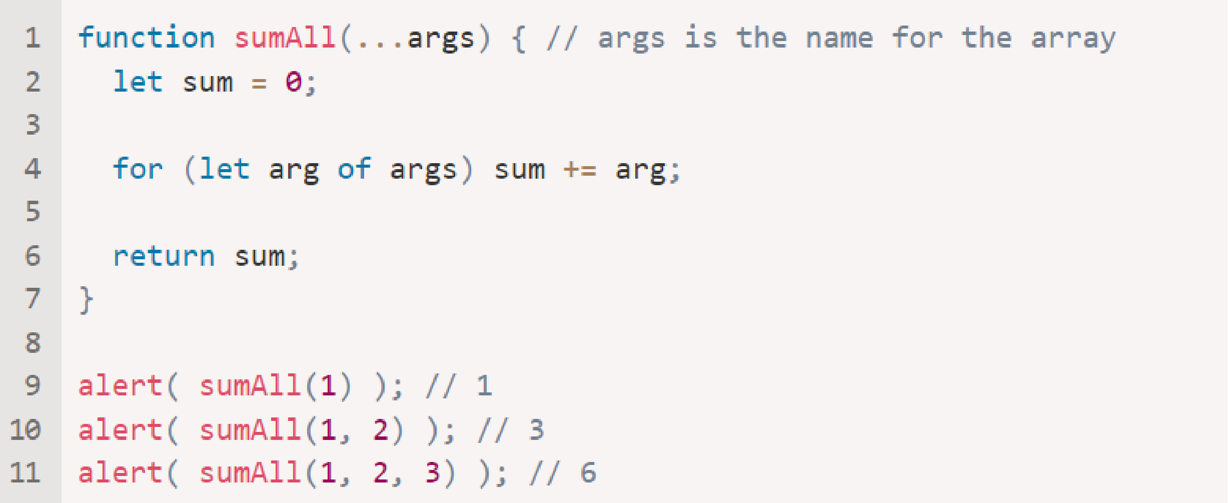
object[key]



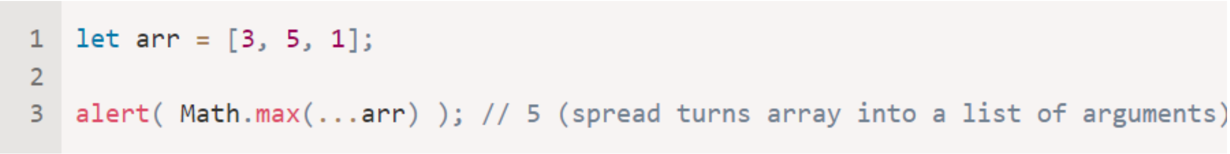
```
let user = {     // an object
  name: "John",  // by key "name" store value "John"
  age: 30        // by key "age" store value 30
};
```

The Spread and the Rest operators.Actually it's only one operator three dots.

Now if we call it spread or rest depends on where we use it, the spread operator is used to split up array elements or object properties.
Rest operator is the same operator but functions differently it merges the properties into one



```
function sumAll(...args) { // args is the name for the array
  let sum = 0;

  for (let arg of args) sum += arg;

  return sum;
}

alert( sumAll(1) ); // 1
alert( sumAll(1, 2) ); // 3
alert( sumAll(1, 2, 3) ); // 6
```



```
let arr = [3, 5, 1];

alert( Math.max(...arr) ); // 5 (spread turns array into a list of arguments)
```

Another core feature of next generation javascript are classes

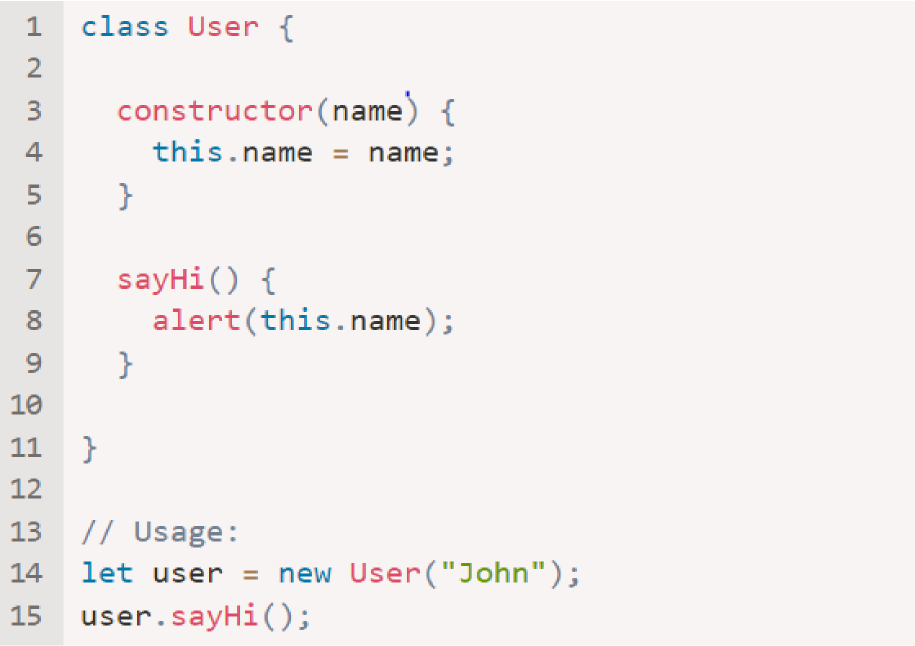
Classes are essentially blueprints for objects.

In our case here for javascript objects. A class is created with the class keyword and a class can have both properties and methods.

Methods are simply functions attached to classes where properties are variables attached to classes.

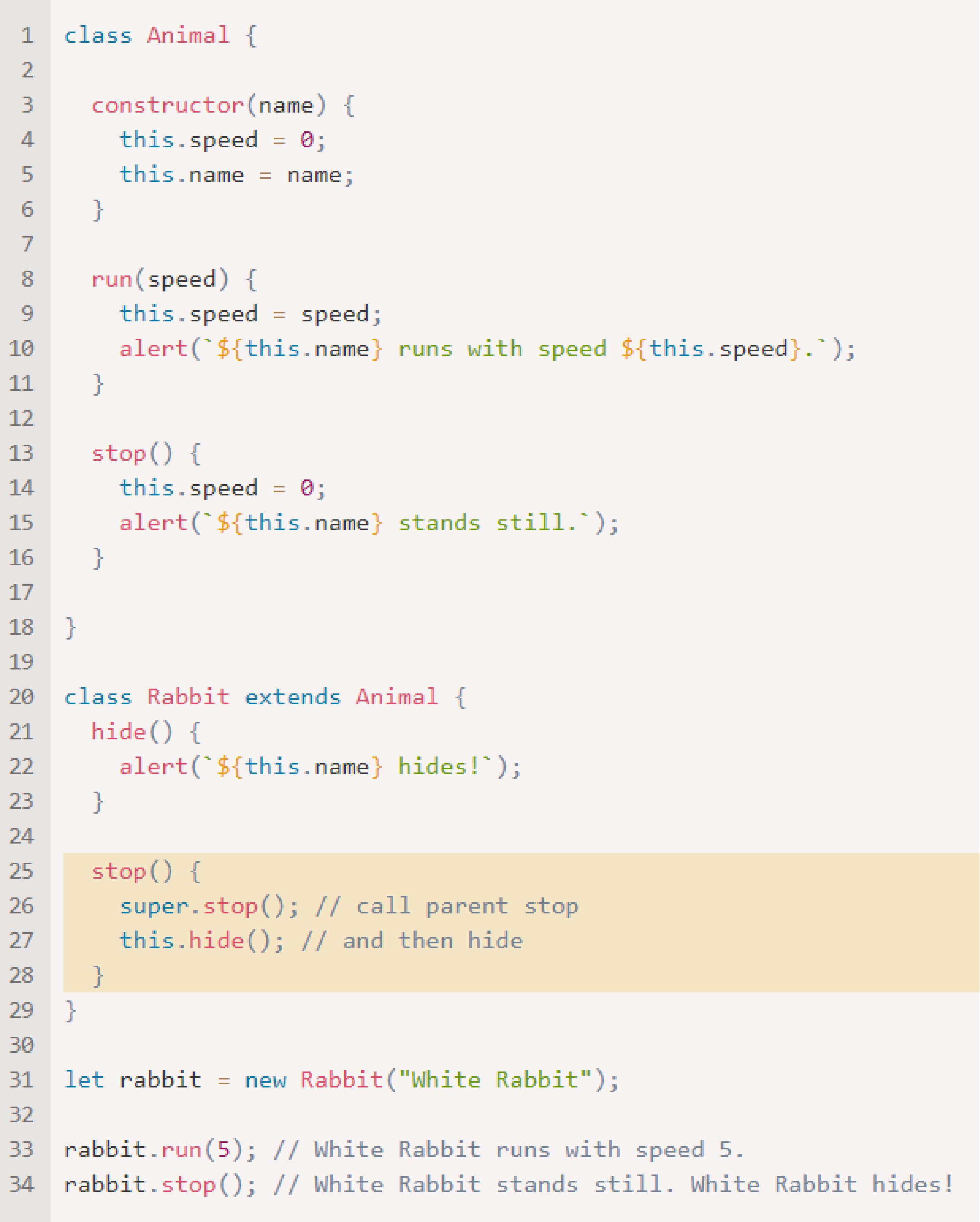
When we add such a property or method a class is instantiated like this with the new keyword.

Constructor is the default function inside a class so we can initialise an object using it . constructor is automatically called with the new keyword.



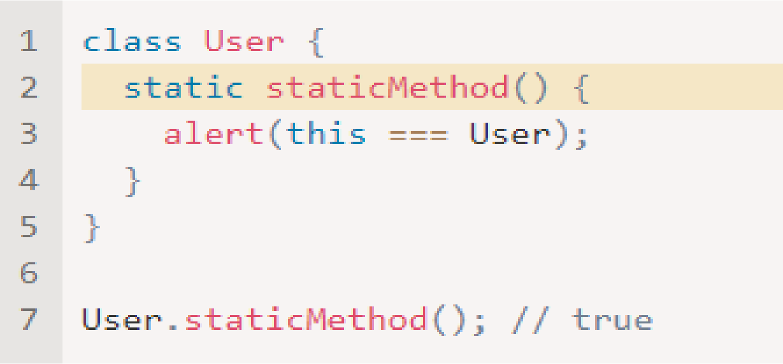
```
class User {

  constructor(name) {
    this.name = name;
  }

  sayHi() {
    alert(this.name);
  }

}

// Usage:
let user = new User("John");
user.sayHi();
```

classes also support inheritance Which means you have another class which you inherit from taking all its properties and methods and potentially adding new properties and methods. We can use the extend keyword to inherit a class.



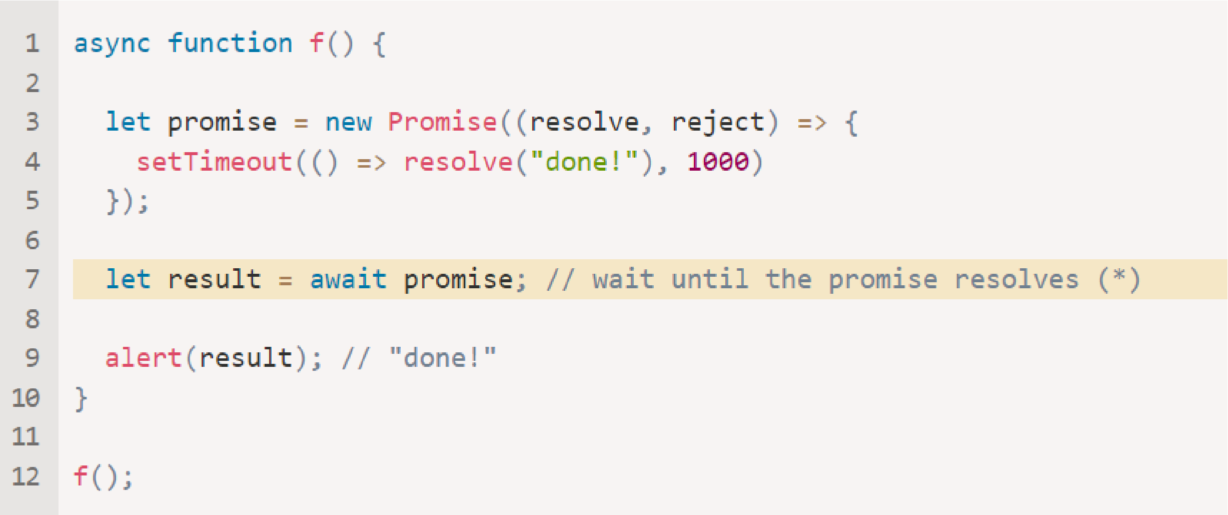
```
class Animal {

  constructor(name) {
    this.speed = 0;
    this.name = name;
  }

  run(speed) {
    this.speed = speed;
    alert(`${this.name} runs with speed ${this.speed}.`);
  }

  stop() {
    this.speed = 0;
    alert(`${this.name} stands still.`);
  }

}

class Rabbit extends Animal {
  hide() {
    alert(`${this.name} hides!`);
  }

  stop() {
    super.stop(); // call parent stop
    this.hide(); // and then hide
  }
}

let rabbit = new Rabbit("White Rabbit");

rabbit.run(5); // White Rabbit runs with speed 5.
rabbit.stop(); // White Rabbit stands still. White Rabbit hides!
```

Static method are defined using static keyword , static methods are defined for the whole class instead for a particular object, static methods can be accessed outside the class without creating object by directly using the class name to call the static method.



```
class User {
  static staticMethod() {
    alert(this === User);
  }
}

User.staticMethod(); // true
```

Async/Await



```
async function f() {

  let promise = new Promise((resolve, reject) => {
    setTimeout(() => resolve("done!"), 1000)
  });

  let result = await promise; // wait until the promise resolves (*)

  alert(result); // "done!"
}

f();
```

WEEK 11-16 :

I was assigned into the Front End team where my responsibility was to develop the User Interface and website of SIEM . So We were given training on React Js and Typescript

REACT JS :

React a JavaScript library for building user interfaces.
React embraces a concept called components.
React is all about components.Components are in the end just a combination of HTML code, CSS code for styling and possibly JavaScript code for some logic.
And you don't have to reuse a component to make it a component. It's only one of its traits that it is reusable.

Each UIs can be separated into parts ,Into holders tables , structures and fastens ,datagrids ,navbars , input components. Each web applications can be separated into parts like this. Respond is about these parts. You fabricate these singular parts and afterward you tell React how to create them together into a last UI. Respond embraces this idea of parts due to that reusability angle and in light of the fact that it permits us to isolate our interests.
The two ideas are significant ideas in programming overall. Having reusable structure blocks assists us with keeping away from reiteration, and in programming overall it is great on the off chance that we don't rehash the same thing. Having a detachment of worries assists us with keeping our code base little and reasonable as opposed to having one huge record which holds all the HTML code and all the JavaScript rationale.

For the whole UI we have little isolated units, these parts where each part has one clear concern, one concentration.React simply gets that idea of components also, of isolating code across components also, makes an interpretation of it to the front end web application world where we constructed a whole UI by parting our code into various parts which we then can blend and match as the need might arise to.

React Hooks are the Functionalities which "guide into" React state and lifecycle highlights from work parts.Hooks are allowed only inside react functional components , they don’t work in class components.

The main Hooks are :

- useEffect
- useState
- useReducer
- useContext

Typescript :

TypeScript is a specifically programming language that expands on JavaScript, giving you better tooling at any scale.

It is a strongly typed programming language and is considered as superset of JavaScript and adds discretionary static composing to the language.

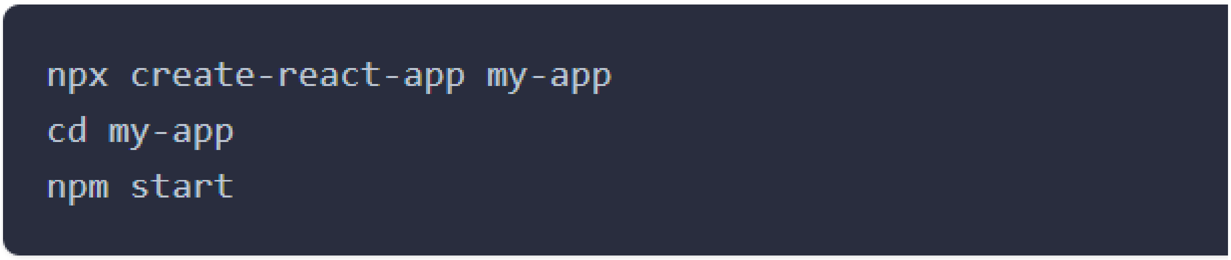
Every file in typescript is save with.ts extension

Chapter -3 System Development

Now using the above tech stacks I made projects during my internship. So in my internship my task was to design the user interface for our product , which consist of login ,logout ,register,jwt authentication ,user crud,roles crud ,policies crud operations .

So starting with the installation of libraries ,since we were creating our project in react so we need to install Node.js into our system and then npm

Next step is creating our react application using the following command :

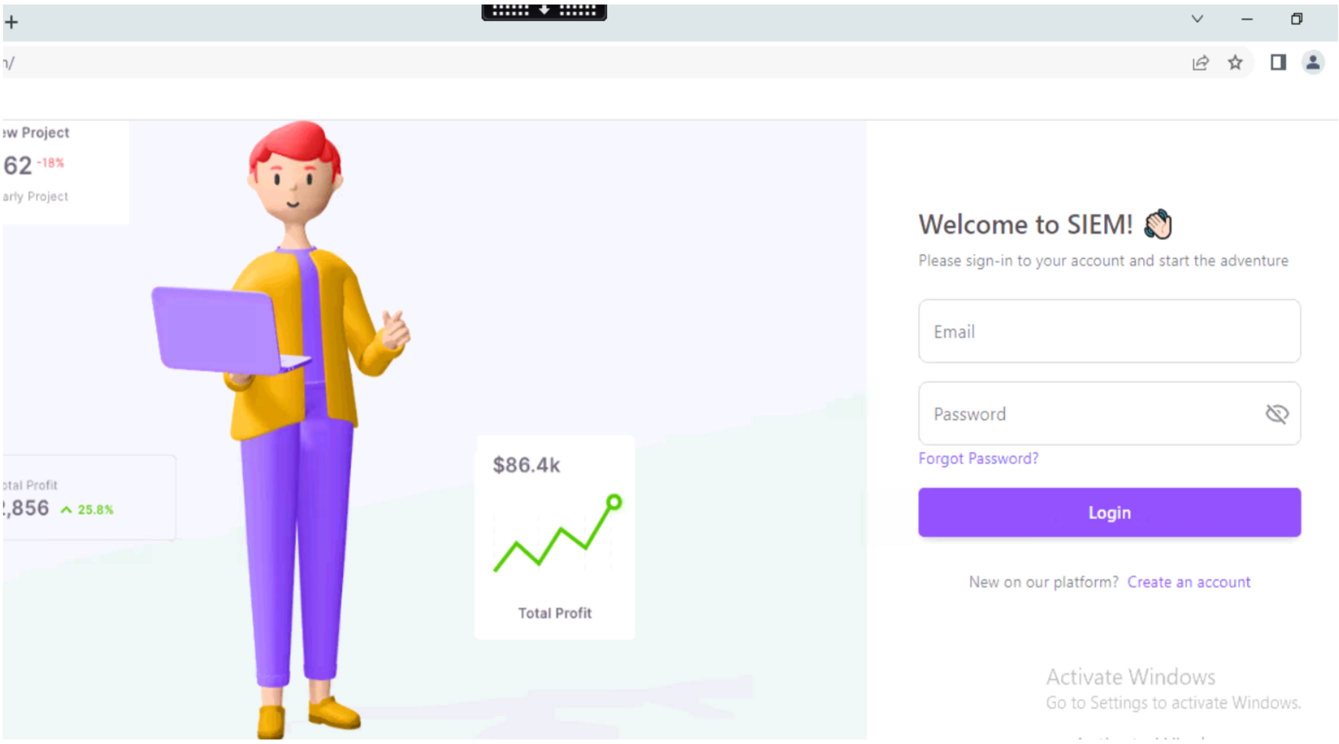


```
npx create-react-app my-app
cd my-app
npm start
```

Login Page :

So first we begin with creating our Login page UI for our Web Application ,
So our Login page has a side form on right side of page and illustration on left.
The login Form has two fields : Email & Passwords ,then there is a login button which will submit the form and a link beneath for creating   account which on clicking will load the create account page.

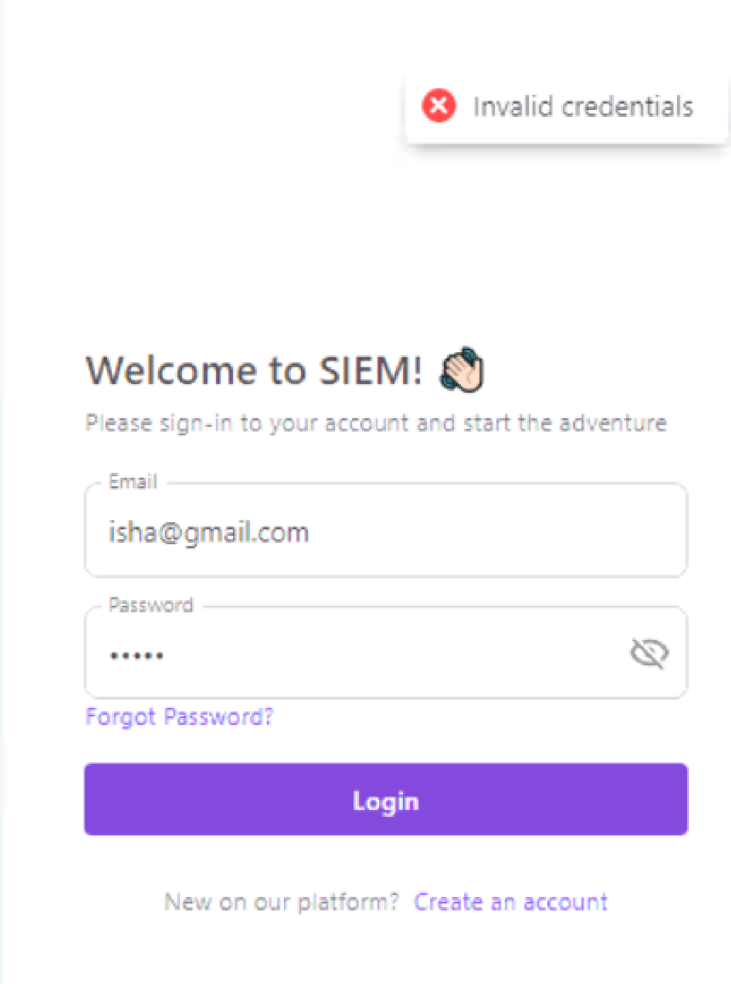
So this is   what Login page looks like :



Welcome to SIEM! 👋

Please sign-in to your account and start the adventure

We have applied validation on email to check if email is valid or not and every field is required ,The page will automatically pop up a message if we enter a wrong email or password. which is not stored in our database



Invalid credentials

Welcome to SIEM!

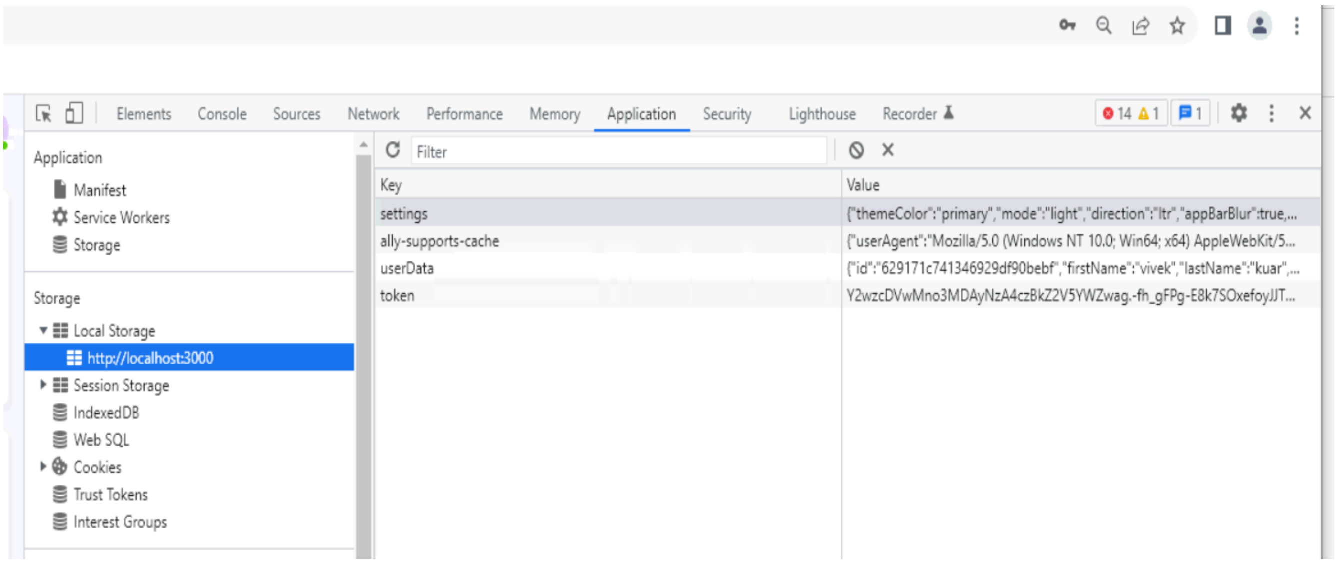
Please sign-in to your account and start the adventure

Email: isha@gmail.com

Password: •••••

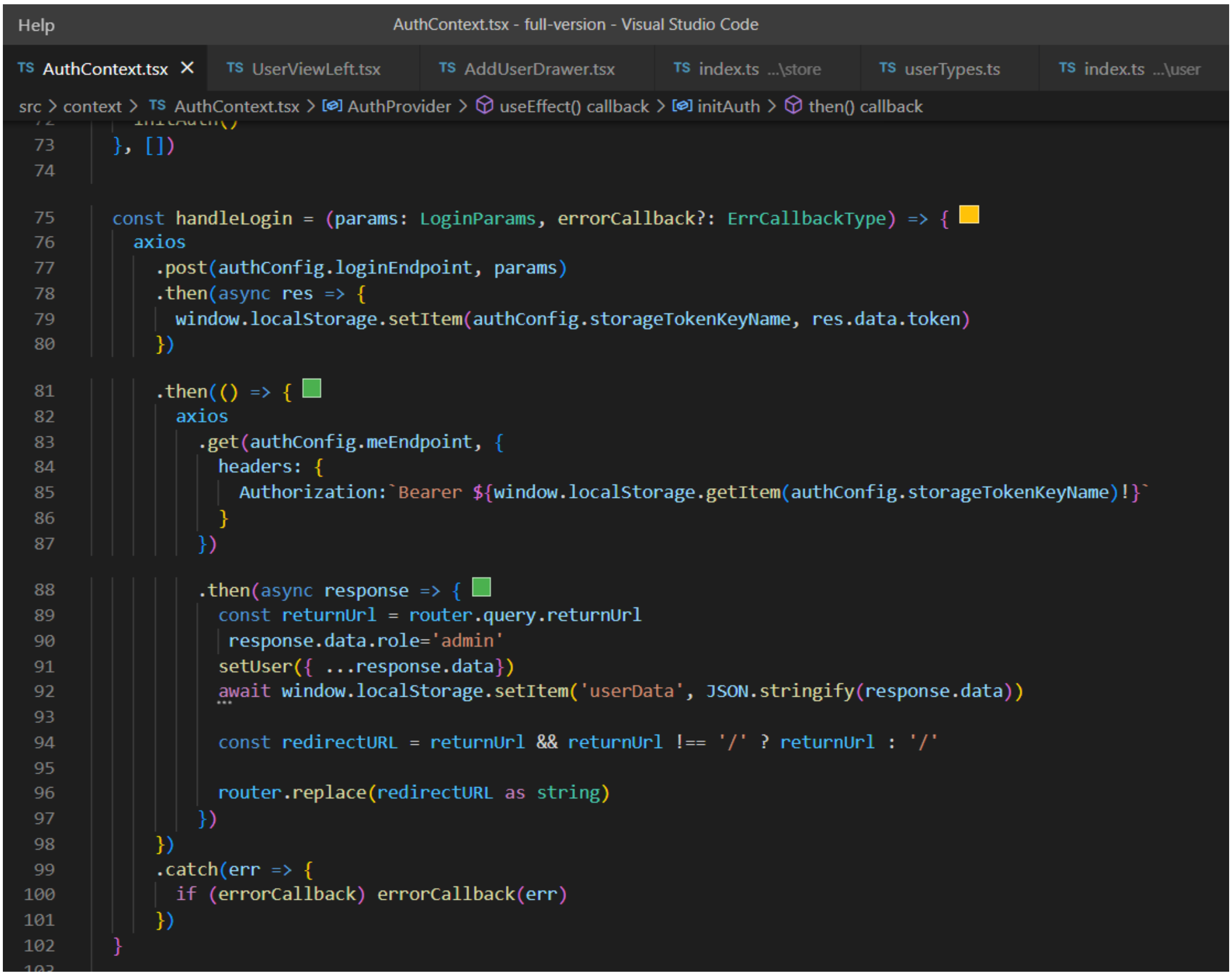
Token authentication:

On clicking the login button a request is sent to the Login api which on successful login return and authentication token in response which is stored in our local storage and is unique for every user.



Key	Value
settings	{"themeColor":"primary","mode":"light","direction":"ltr","appBarBlur":true,...
ally-supports-cache	{"userAgent":"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/5...
userData	{"id":"629171c741346929df90bebf","firstName":"vivek","lastName":"kuar",...
token	Y2wzcDVwMno3MDAyNzA4czBkZ2V5YWZwag.-fh_gFPg-E8k7SOxefoyJJT...

So this the code for handle login where we are sending GET request using axios to the login API along with parameter entered through form and waiting for response using async and await.


Help
AuthContext.tsx - full-version - Visual Studio Code
TS AuthContext.tsx ×
TS UserViewLeft.tsx
TS AddUserDrawer.tsx
TS index.ts ...\store
TS userTypes.ts
TS index.ts ...\user
src > context > TS AuthContext.tsx > AuthProvider > useEffect() callback > initAuth > then() callback

```
    initAuth()
  }, [])

  const handleLogin = (params: LoginParams, errorCallback?: ErrCallbackType) => {
    axios
      .post(authConfig.loginEndpoint, params)
      .then(async res => {
        window.localStorage.setItem(authConfig.storageTokenKeyName, res.data.token)
      })

      .then(() => {
        axios
          .get(authConfig.meEndpoint, {
            headers: {
              Authorization:`Bearer ${window.localStorage.getItem(authConfig.storageTokenKeyName)!}`
            }
          })

          .then(async response => {
            const returnUrl = router.query.returnUrl
             response.data.role='admin'
            setUser({ ...response.data})
            await window.localStorage.setItem('userData', JSON.stringify(response.data))

            const redirectURL = returnUrl && returnUrl !== '/' ? returnUrl : '/'

            router.replace(redirectURL as string)
          })
      })
      .catch(err => {
        if (errorCallback) errorCallback(err)
      })
  }
```

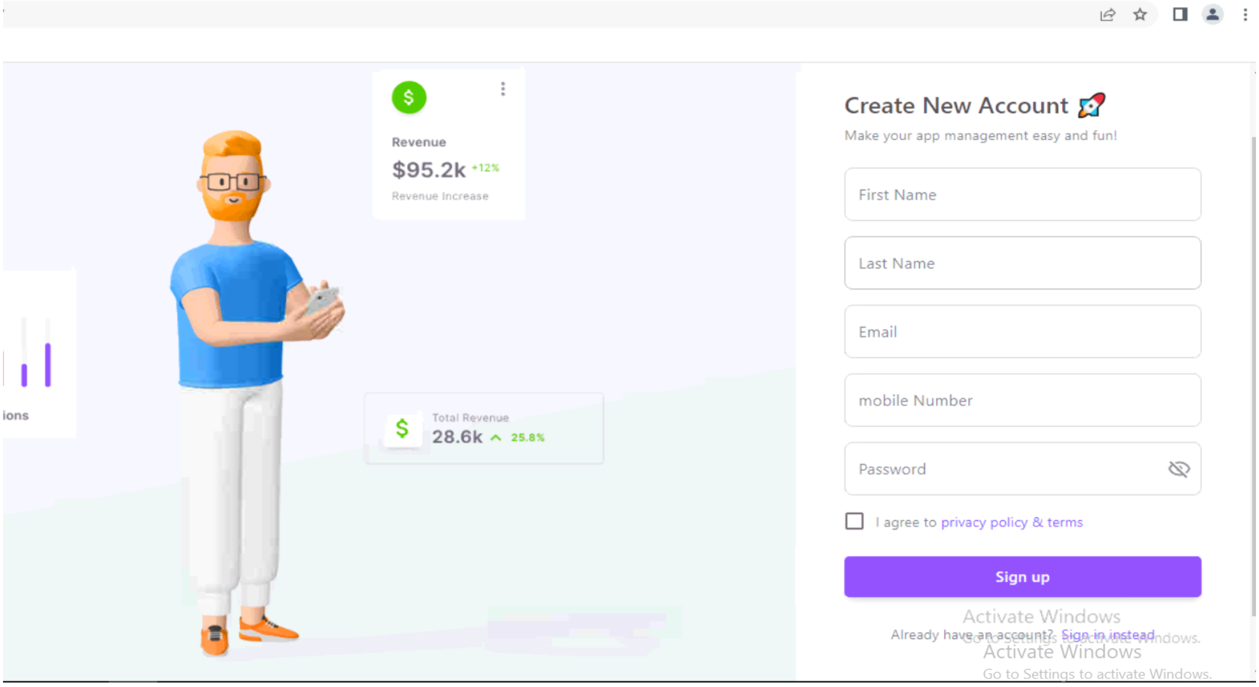
Register page:

Now we will be designing our register page or create account page .

For our register page we will have a similar form on the right side of page and left side will be illustration.

Our Register Form consist of Five fields i.e first name ,last name ,email ,mobile,password and all fields are mandatory along with a check box for agreeing terms and policies.

So this what it looks like:



Create New Account

Make your app management easy and fun!

First Name

Last Name

Email

mobile Number

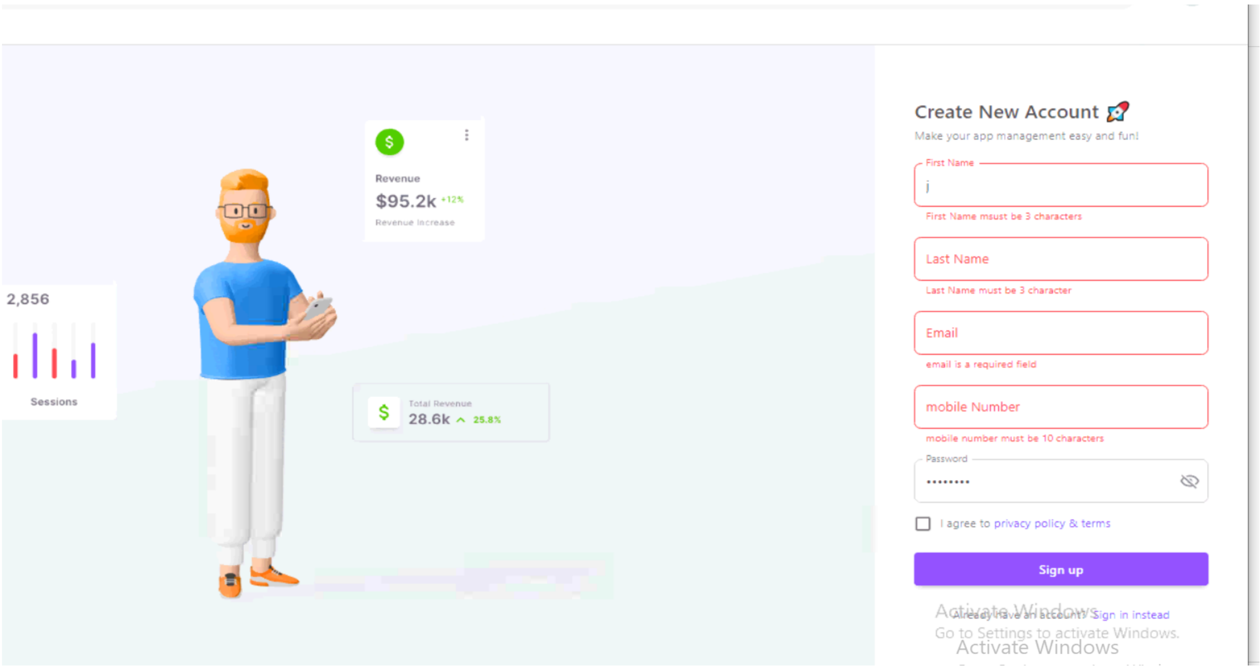
Password

□ I agree to privacy policy & terms

Sign up

Already have an account? Sign in instead

Register page also has validation over all the fields:



Create New Account

Make your app management easy and fun!

First Name: j

First Name msust be 3 characters

Last Name

Last Name must be 3 character

Email

email is a required field

mobile Number

mobile number must be 10 characters

Password: ••••••••

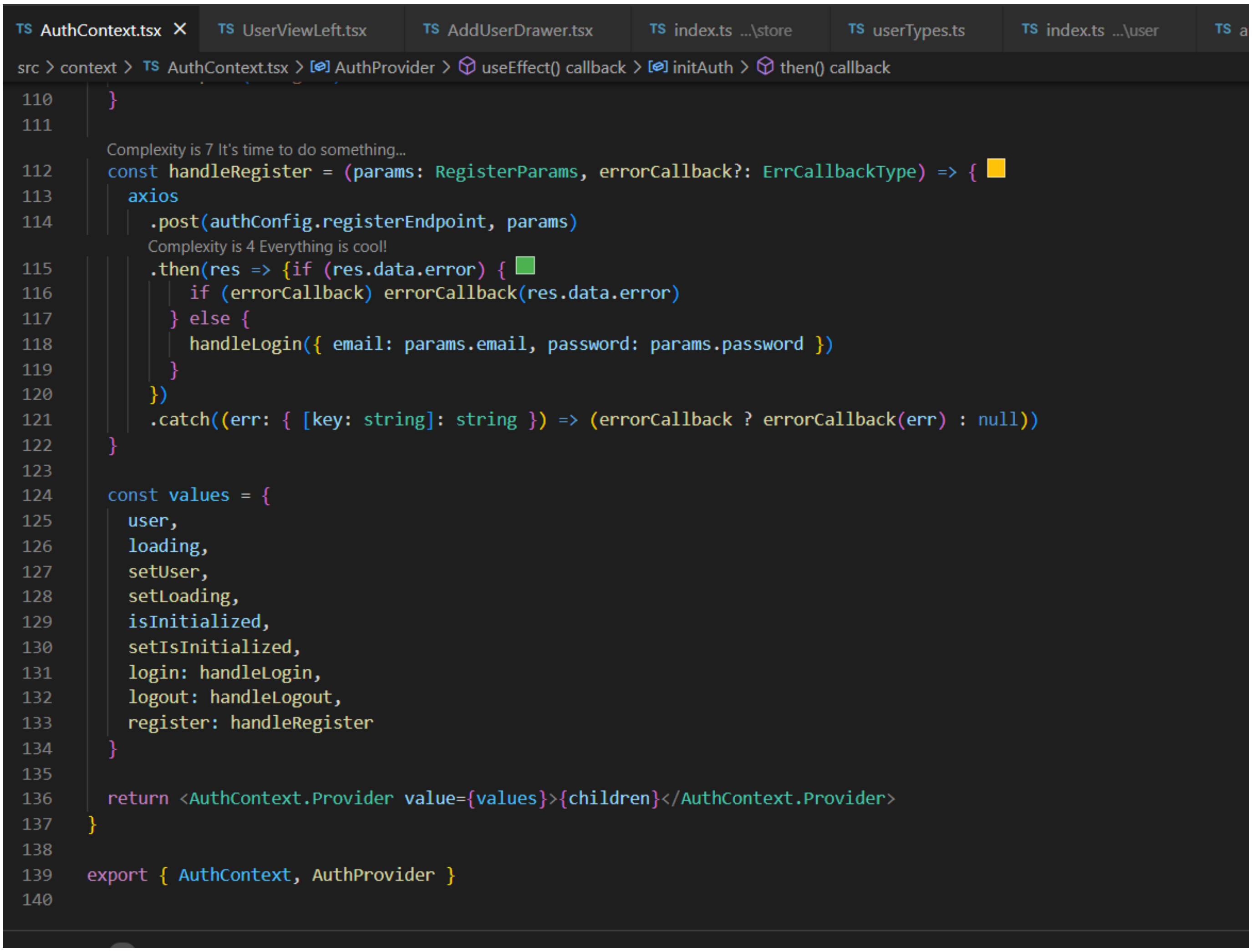
□ I agree to privacy policy & terms

Sign up

Already have an account? Sign in instead

On clicking the Sign up button a request is sent to the register api which on successful login return and authentication token in response which is stored in our local storage and is unique for every user.

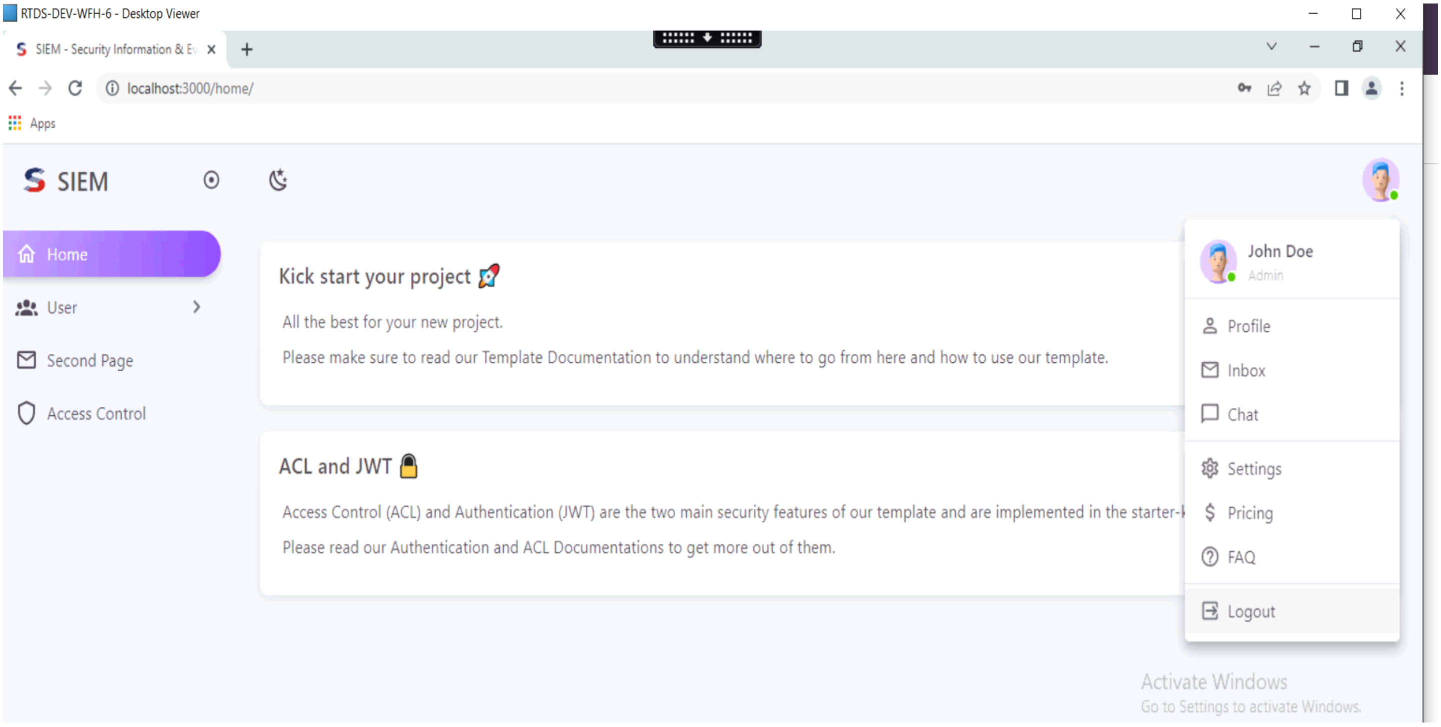
So we have our code for register where we are using axios for sending POST request to register API and have async and await response



```
  }

  Complexity is 7 It's time to do something...
  const handleRegister = (params: RegisterParams, errorCallback?: ErrCallbackType) => { 🟨
    axios
      .post(authConfig.registerEndpoint, params)
      Complexity is 4 Everything is cool!
      .then(res => {if (res.data.error) { 🟩
          if (errorCallback) errorCallback(res.data.error)
        } else {
          handleLogin({ email: params.email, password: params.password })
        }
      })
      .catch((err: { [key: string]: string }) => (errorCallback ? errorCallback(err) : null))
  }

  const values = {
    user,
    loading,
    setUser,
    setLoading,
    isInitialized,
    setIsInitialized,
    login: handleLogin,
    logout: handleLogout,
    register: handleRegister
  }

  return <AuthContext.Provider value={values}>{children}</AuthContext.Provider>
}

export { AuthContext, AuthProvider }
```

Logout :

Now for logout we have created a logout button in the side user menu



Kick start your project 🚀

All the best for your new project.

Please make sure to read our Template Documentation to understand where to go from here and how to use our template.

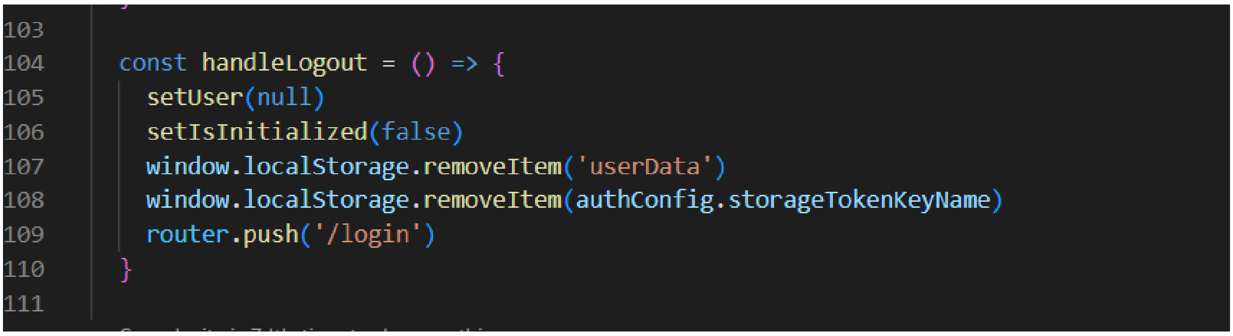
ACL and JWT 🔒

Access Control (ACL) and Authentication (JWT) are the two main security features of our template and are implemented in the starter-k

Please read our Authentication and ACL Documentations to get more out of them.

On clicking the logout button we are deleting our authentication token from localStorage and redirecting back to the login page.

This is what handle logout code looks like:



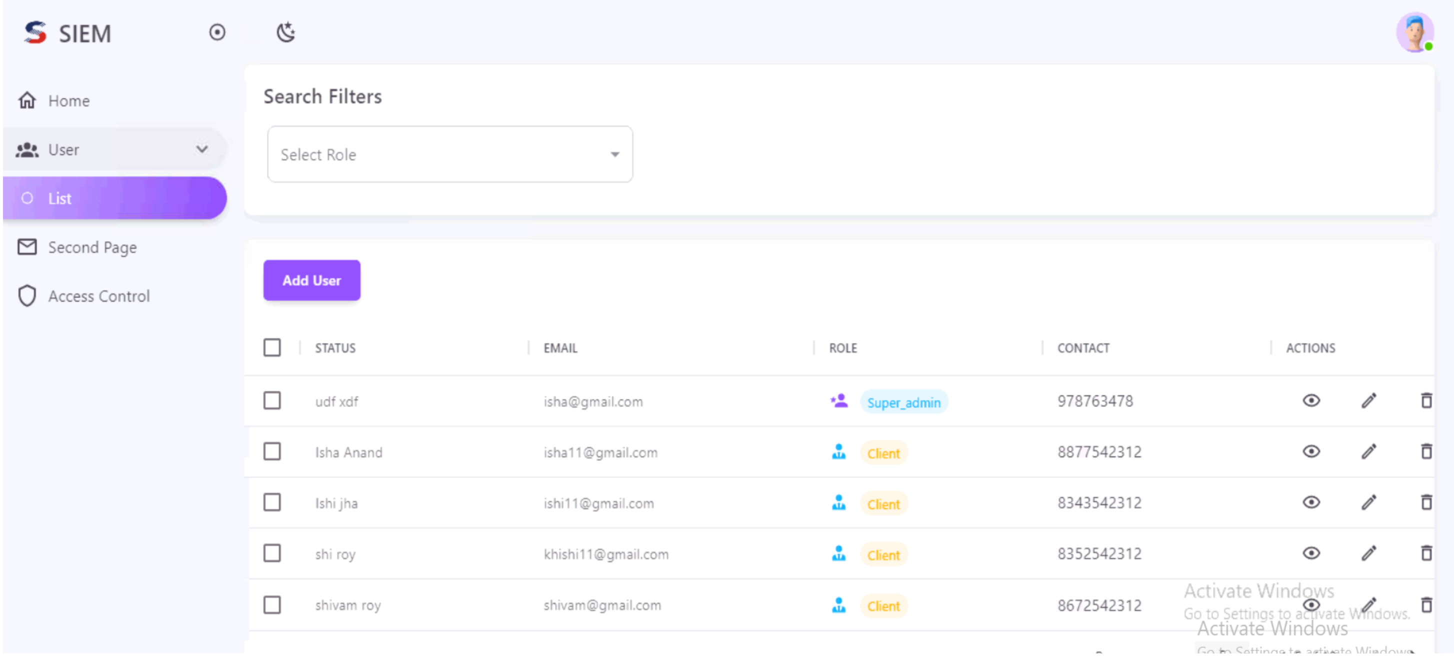
```
  const handleLogout = () => {
    setUser(null)
    setIsInitialized(false)
    window.localStorage.removeItem('userData')
    window.localStorage.removeItem(authConfig.storageTokenKeyName)
    router.push('/login')
  }
```

User Page:

Now we will be performing crud on user data i.e will be performing create ,read ,update ,delete on the list of users that have registered on our portal

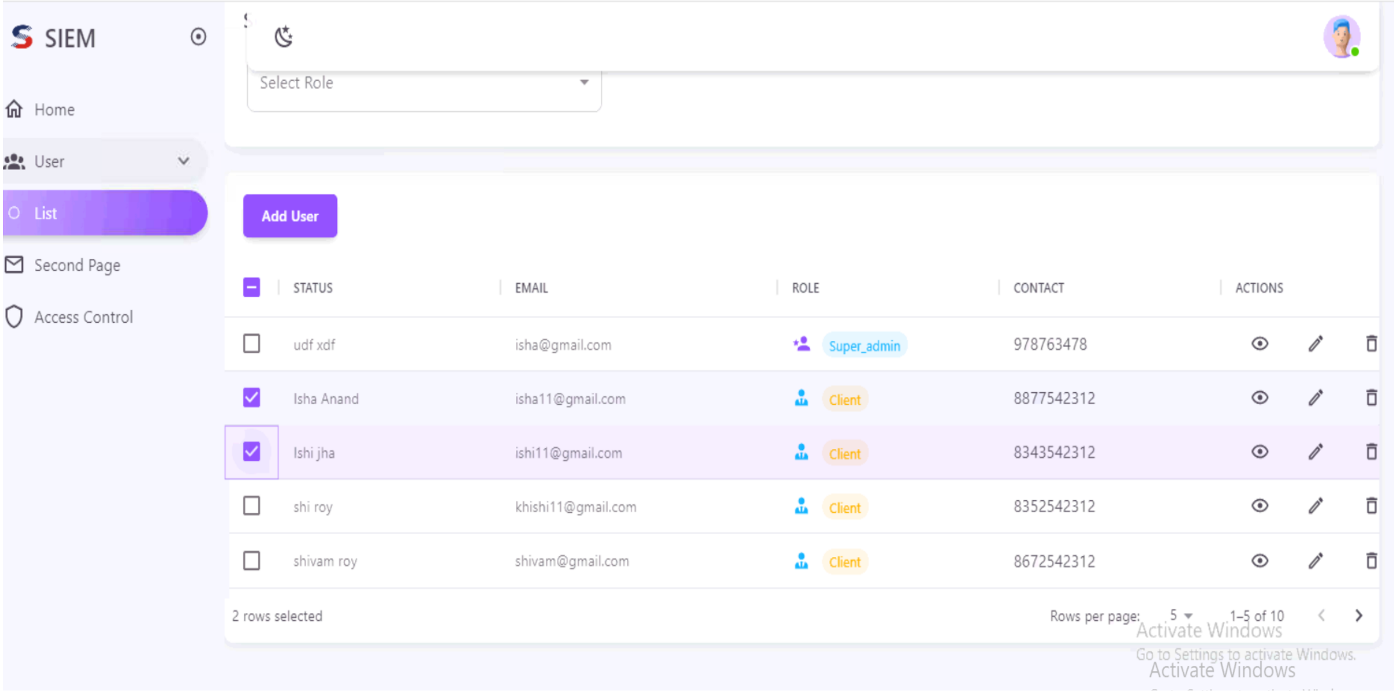
Begin with the fetching data from backend and displaying list of users over the table

So here we are displaying Name ,Email ,Role ,Contact ,Actions (view ,edit ,delete)



□	STATUS	EMAIL	ROLE	CONTACT	ACTIONS
□	udf xdf	isha@gmail.com	Super_admin	978763478	
□	Isha Anand	isha11@gmail.com	Client	8877542312	
□	Ishi jha	ishi11@gmail.com	Client	8343542312	
□	shi roy	khishi11@gmail.com	Client	8352542312	
□	shivam roy	shivam@gmail.com	Client	8672542312	

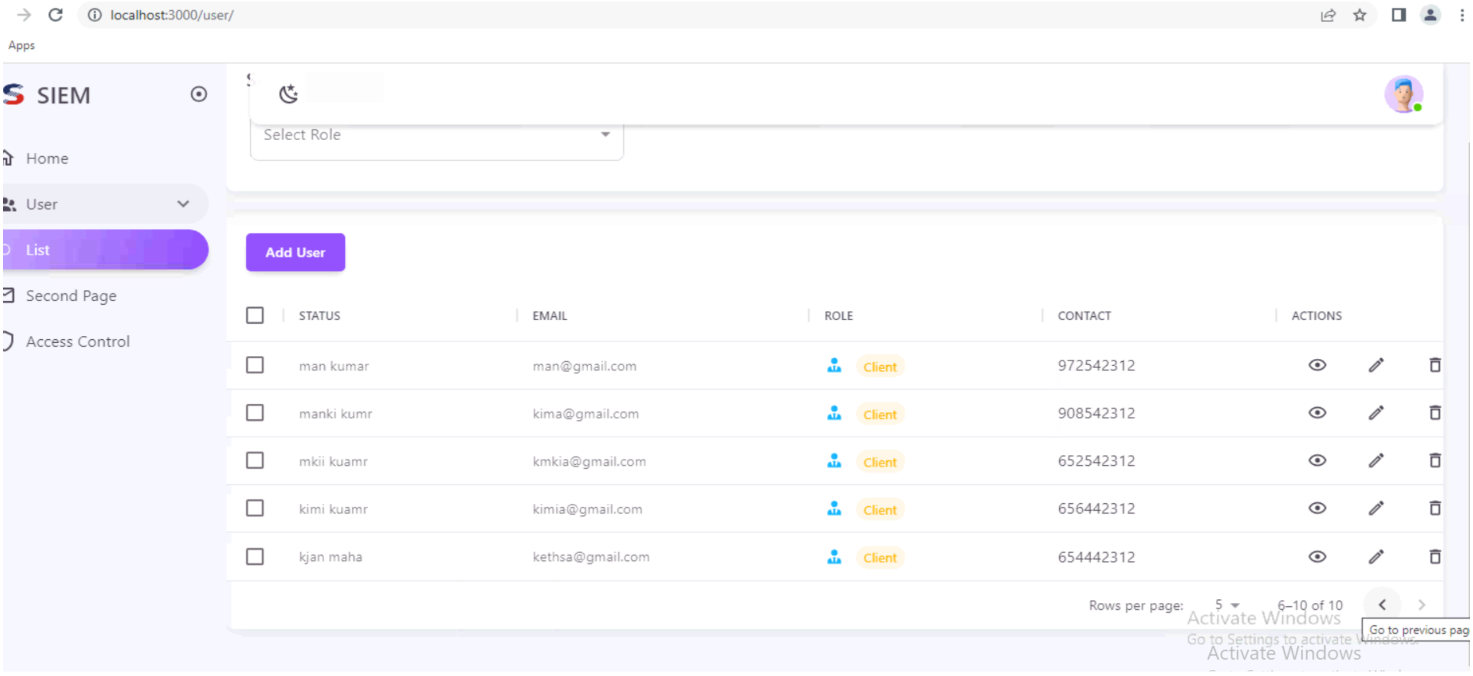
We have applied pagination to our table where we can change data on change page no or changing rows per page



▬	STATUS	EMAIL	ROLE	CONTACT	ACTIONS
□	udf xdf	isha@gmail.com	Super_admin	978763478	
☑	Isha Anand	isha11@gmail.com	Client	8877542312	
☑	Ishi jha	ishi11@gmail.com	Client	8343542312	
□	shi roy	khishi11@gmail.com	Client	8352542312	
□	shivam roy	shivam@gmail.com	Client	8672542312	

2 rows selected

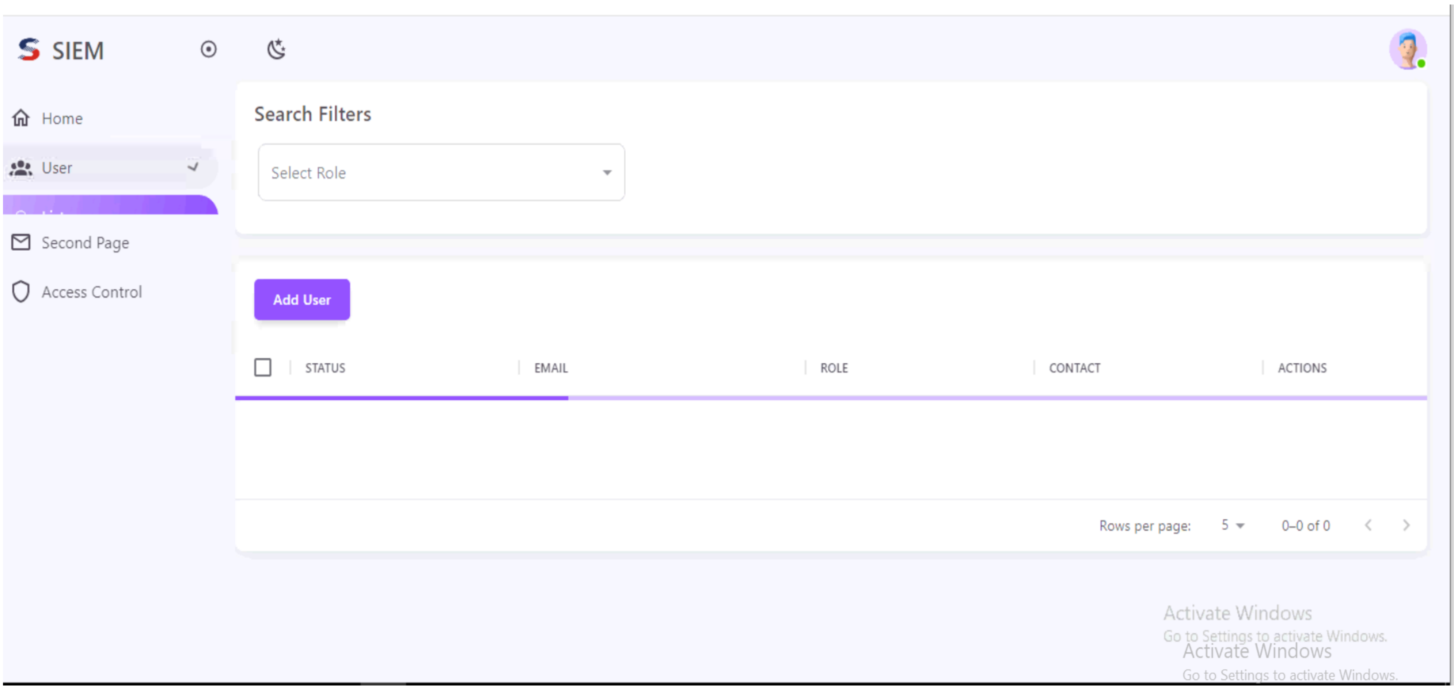
Rows per page: 5 1–5 of 10



☐	STATUS	EMAIL	ROLE	CONTACT	ACTIONS
☐	man kumar	man@gmail.com	Client	972542312	
☐	manki kumr	kima@gmail.com	Client	908542312	
☐	mkii kuamr	kmkia@gmail.com	Client	652542312	
☐	kimi kuamr	kimia@gmail.com	Client	656442312	
☐	kjan maha	kethsa@gmail.com	Client	654442312	

Rows per page: 5 6–10 of 10

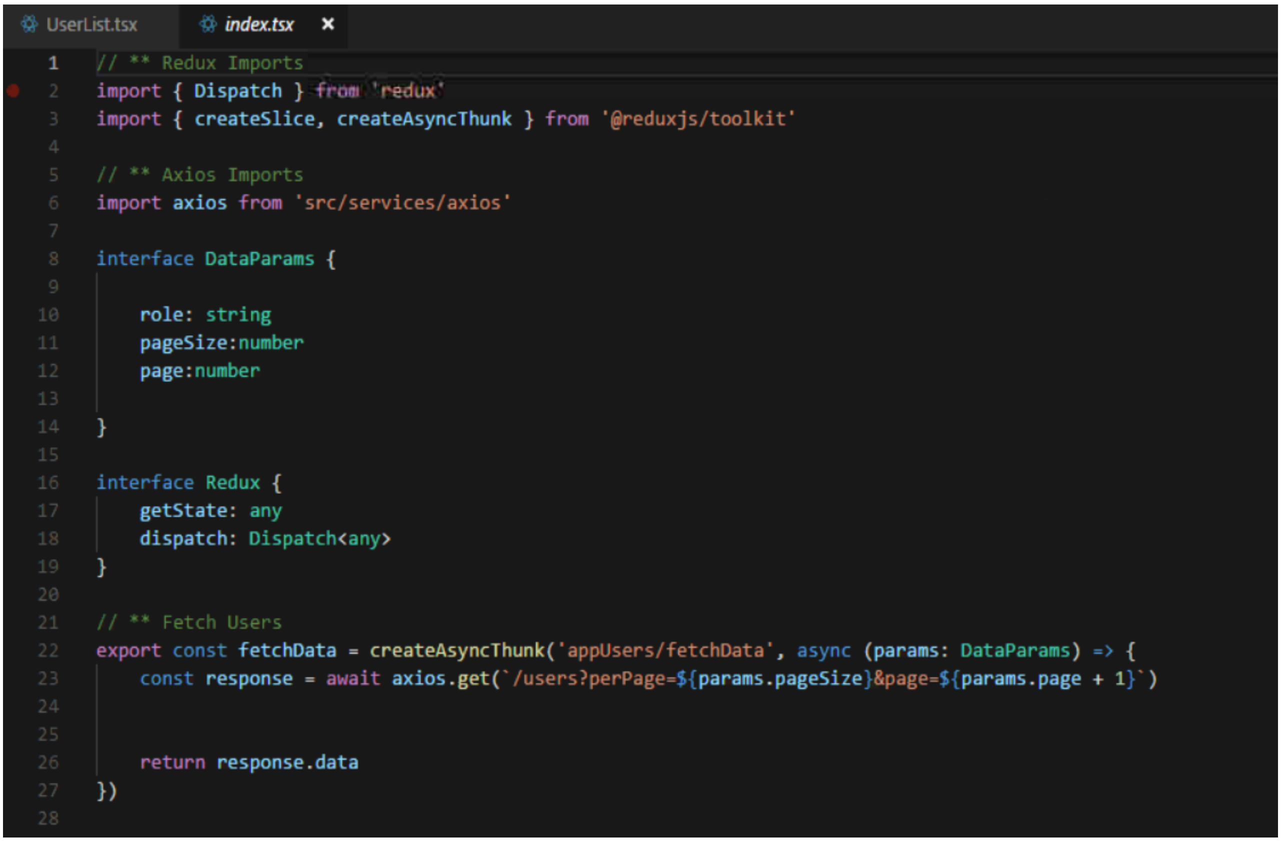
We also implemented loader in our table so the time till data is loading loader will run in table



☐	STATUS	EMAIL	ROLE	CONTACT	ACTIONS

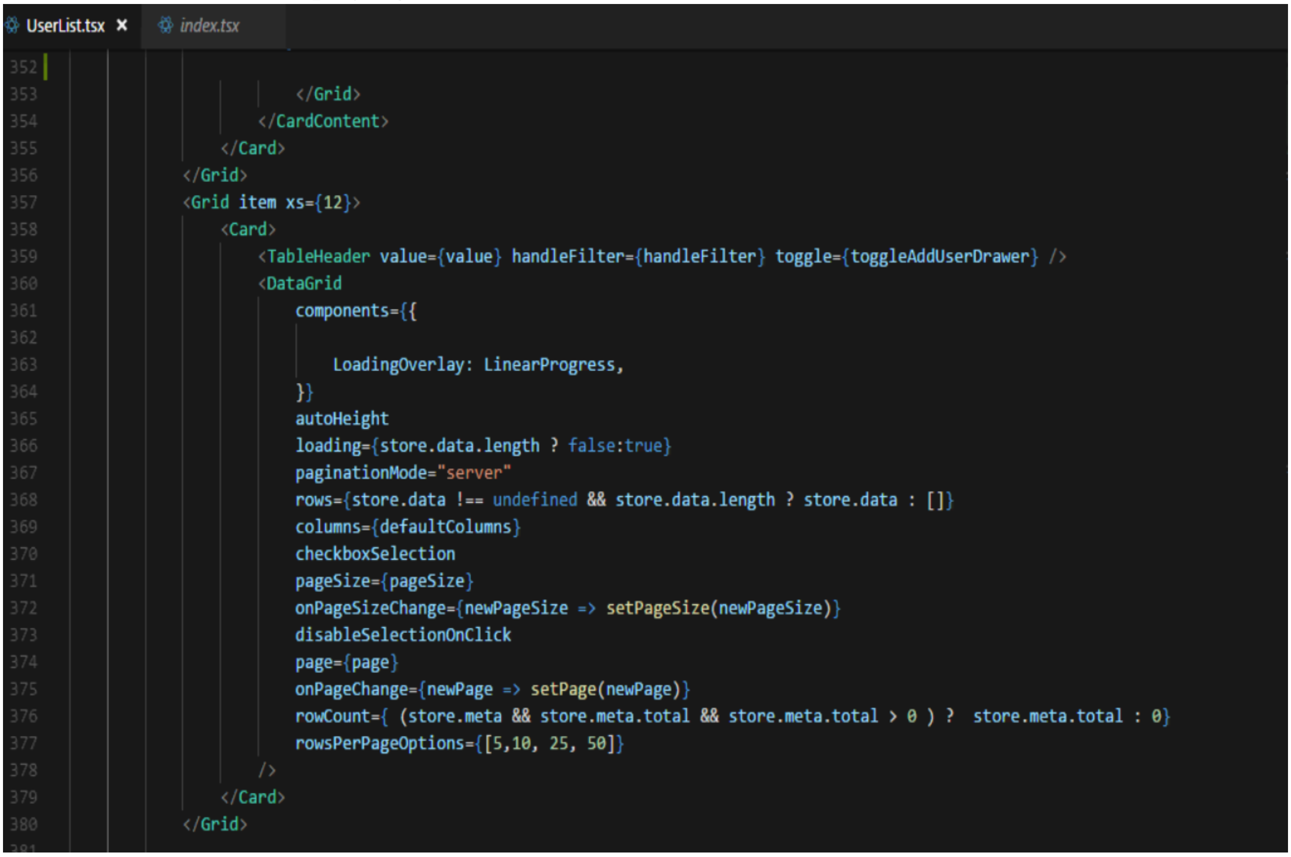
Rows per page: 5 0–0 of 0

So this the api GET request we are sending in order to fetch user list from backend



```tsx
// ** Redux Imports
import { Dispatch } from 'redux'
import { createSlice, createAsyncThunk } from '@reduxjs/toolkit'

// ** Axios Imports
import axios from 'src/services/axios'

interface DataParams {

    role: string
    pageSize:number
    page:number

}

interface Redux {
    getState: any
    dispatch: Dispatch<any>
}

// ** Fetch Users
export const fetchData = createAsyncThunk('appUsers/fetchData', async (params: DataParams) => {
    const response = await axios.get(`/users?perPage=${params.pageSize}&page=${params.page + 1}`)


    return response.data
})

```

Our table that we are displaying in UI:

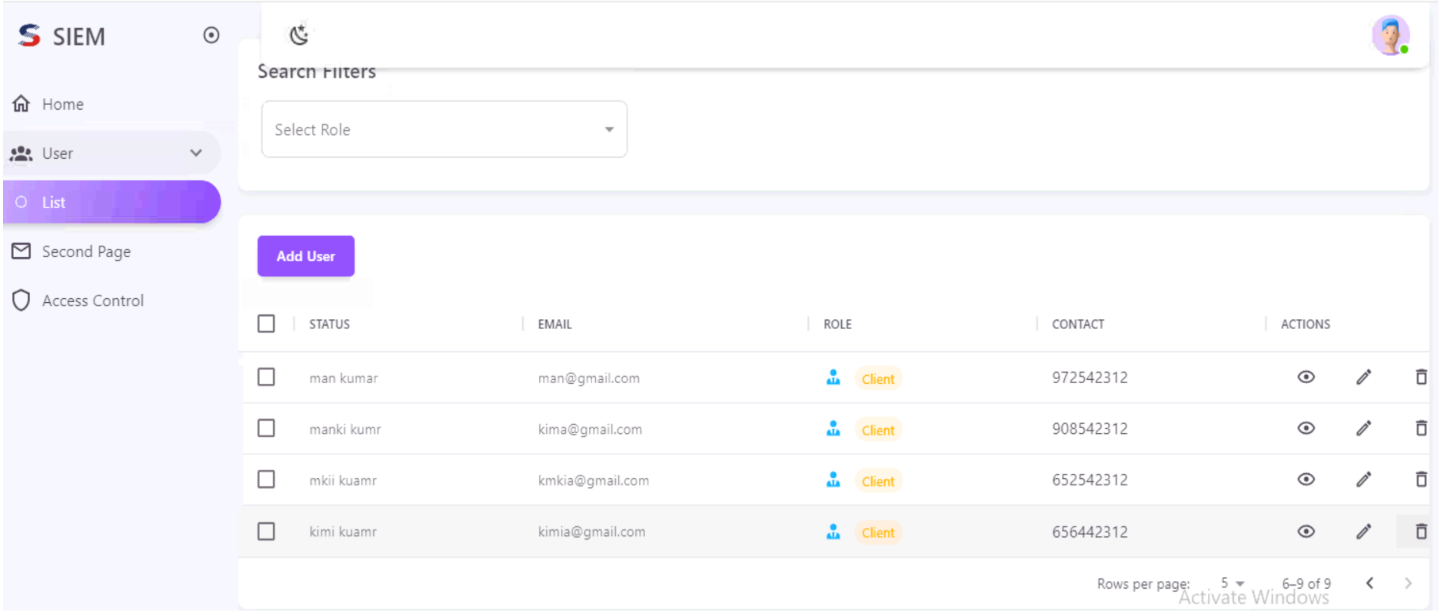


```tsx

            </Grid>
          </CardContent>
        </Card>
      </Grid>
      <Grid item xs={12}>
        <Card>
          <TableHeader value={value} handleFilter={handleFilter} toggle={toggleAddUserDrawer} />
          <DataGrid
            components={{

              LoadingOverlay: LinearProgress,
            }}
            autoHeight
            loading={store.data.length ? false:true}
            paginationMode="server"
            rows={store.data !== undefined && store.data.length ? store.data : []}
            columns={defaultColumns}
            checkboxSelection
            pageSize={pageSize}
            onPageSizeChange={newPageSize => setPageSize(newPageSize)}
            disableSelectionOnClick
            page={page}
            onPageChange={newPage => setPage(newPage)}
            rowCount={ (store.meta && store.meta.total && store.meta.total > 0 ) ?  store.meta.total : 0}
            rowsPerPageOptions={[5,10, 25, 50]}
          />
        </Card>
      </Grid>
```

Now we will perform the Delete operation on the table so on applying delete our row dispapper ,

To apply delete on a particular row we need to click on the delete button at that particular row.

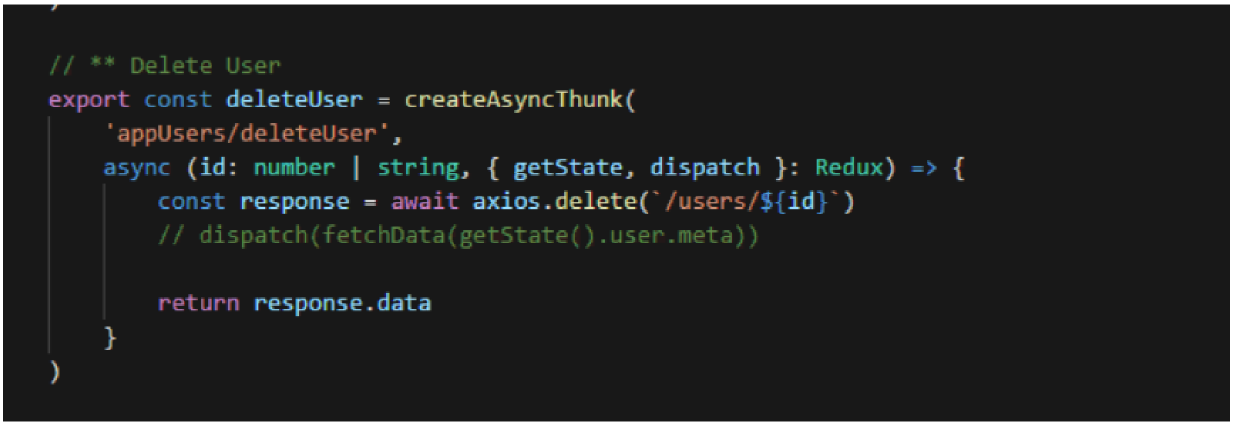
As show earlier there were 10 record but now there are 9 only.



□	STATUS	EMAIL	ROLE	CONTACT	ACTIONS
□	man kumar	man@gmail.com	Client	972542312	
□	manki kumr	kima@gmail.com	Client	908542312	
□	mkii kuamr	kmkia@gmail.com	Client	652542312	
□	kimi kuamr	kimia@gmail.com	Client	656442312	

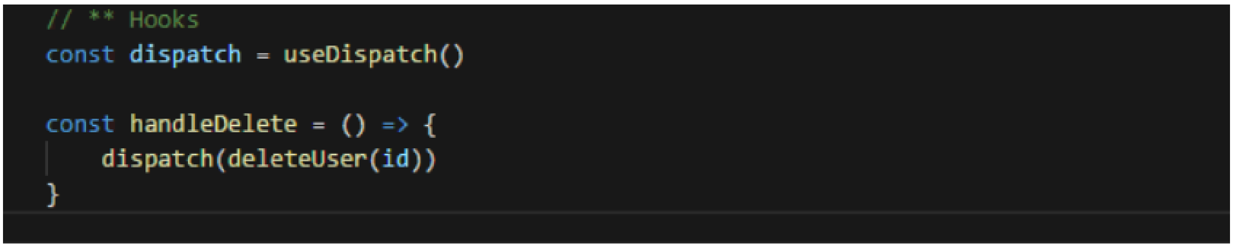
Rows per page: 5 6–9 of 9

Below code represent the delete request :



```
// ** Delete User
export const deleteUser = createAsyncThunk(
    'appUsers/deleteUser',
    async (id: number | string, { getState, dispatch }: Redux) => {
        const response = await axios.delete(`/users/${id}`)
        // dispatch(fetchData(getState().user.meta))

        return response.data
    }
)
```

And the handle Delete Function:



```
// ** Hooks
const dispatch = useDispatch()

const handleDelete = () => {
    dispatch(deleteUser(id))
}
```

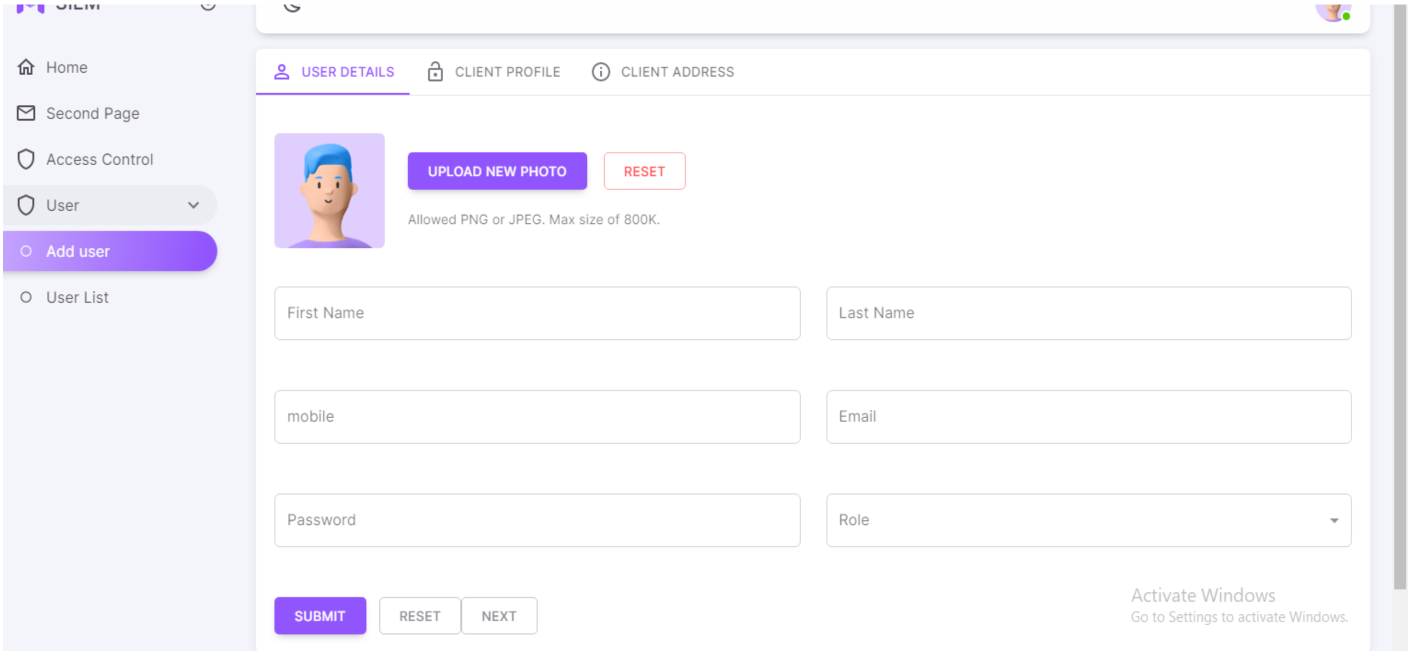
Add User:

Now we will move towards adding user to user table:

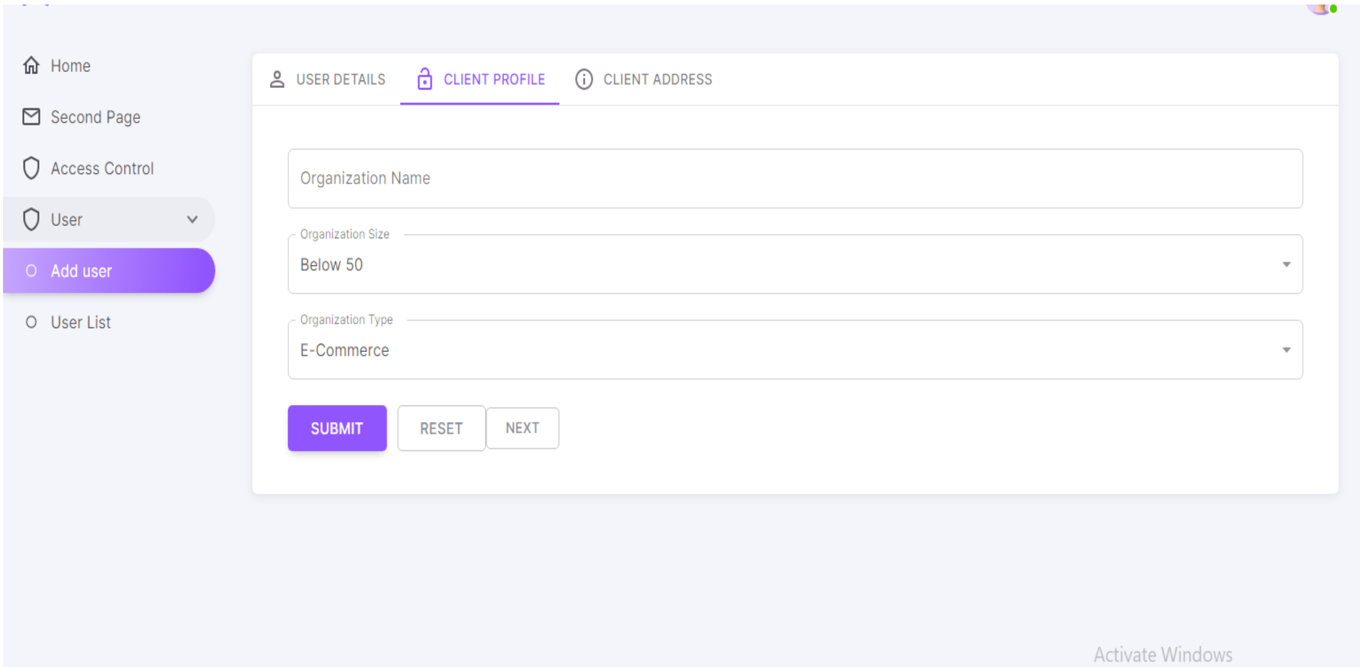
For adding our user we had an Add User button on the user table so clicking on add user button will lead to our add user form .

This add user form is a three step form where for adding user we have to first fill the user details ,then its profile and then address , each page has a save next and reset button , where the save button will save the data and we can open it any time further and next will move to next page of the user add form and reset will clear all the fields.

which looks like this:



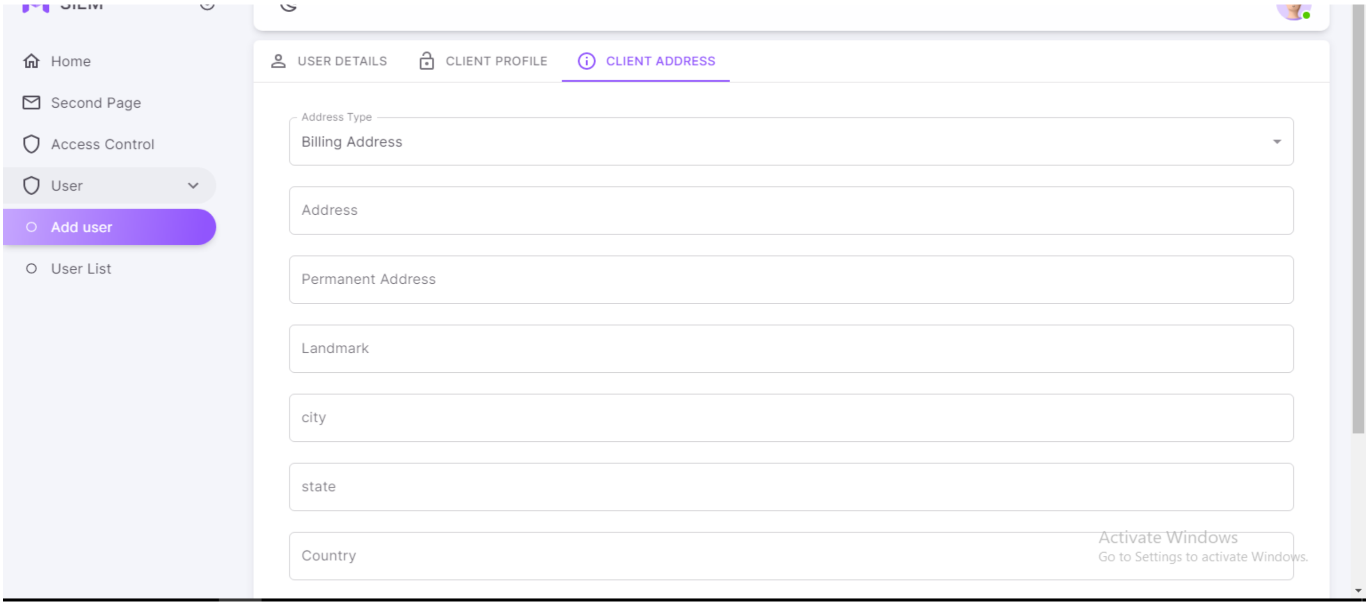
Allowed PNG or JPEG. Max size of 800K.



Organization Name

Organization Size
Below 50

Organization Type
E-Commerce



Address Type
Billing Address

Address

Permanent Address

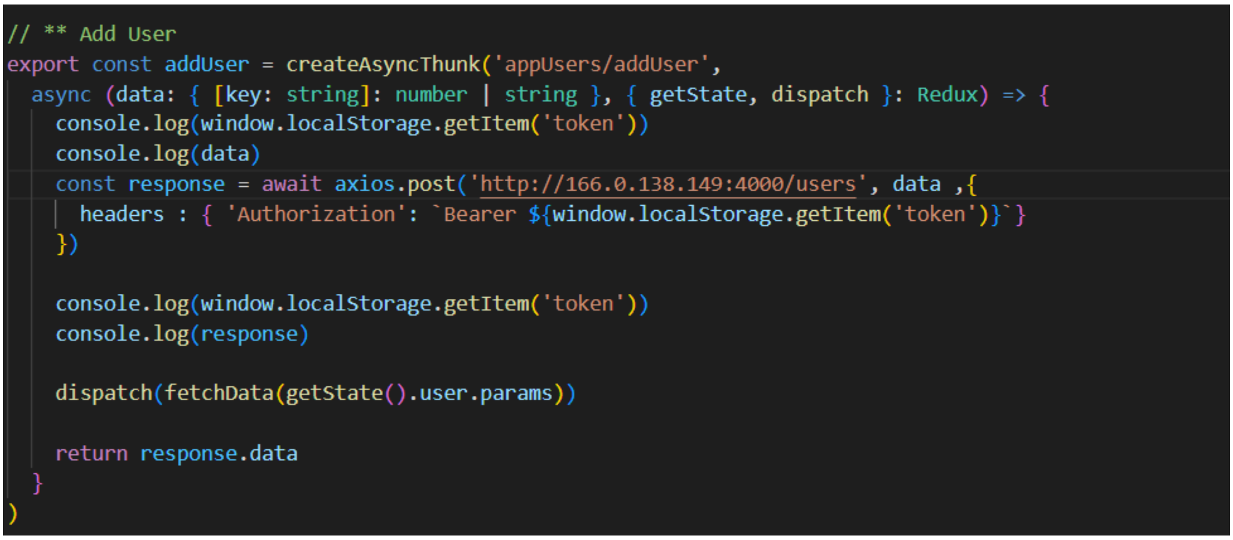
Landmark

city

state

Country

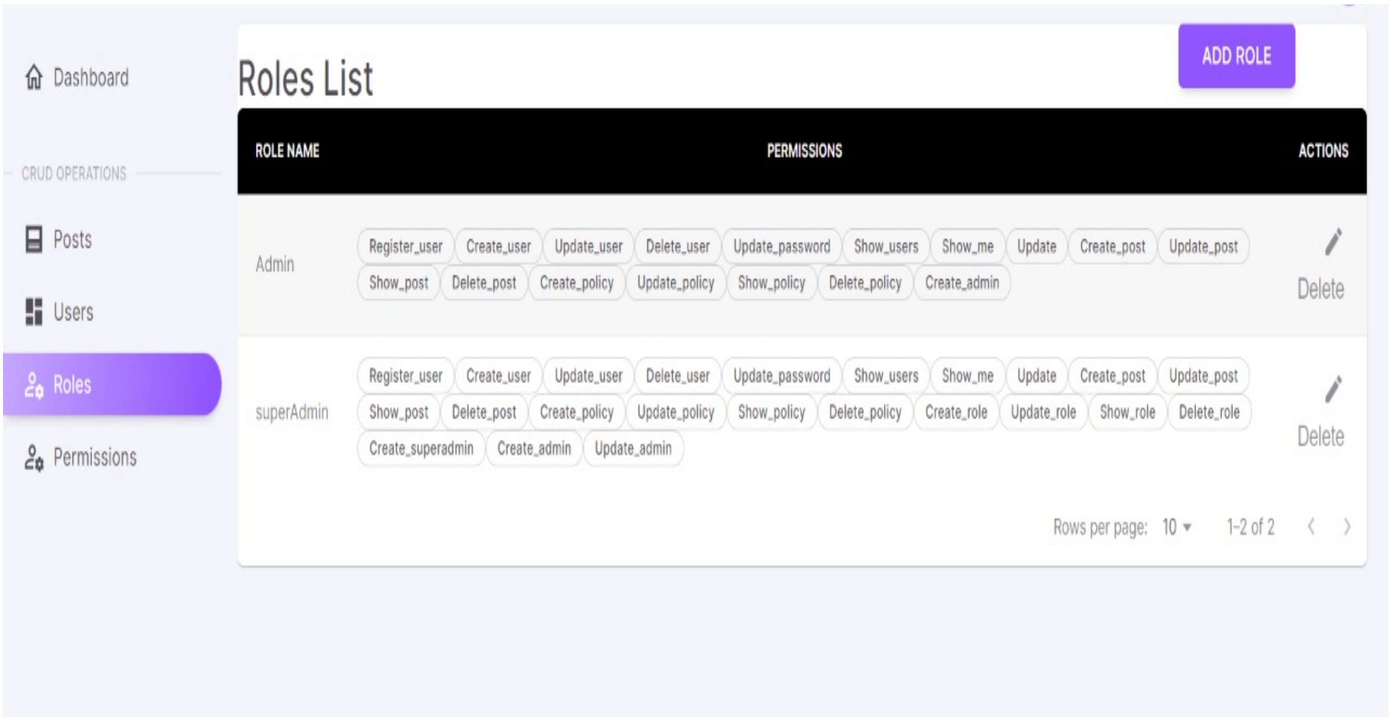
Code of add user where we are sending post request to api along with th parameters filled in the form Which looks like:



```
// ** Add User
export const addUser = createAsyncThunk('appUsers/addUser',
  async (data: { [key: string]: number | string }, { getState, dispatch }: Redux) => {
    console.log(window.localStorage.getItem('token'))
    console.log(data)
    const response = await axios.post('http://166.0.138.149:4000/users', data ,{
      headers : { 'Authorization': `Bearer ${window.localStorage.getItem('token')}`}
    })

    console.log(window.localStorage.getItem('token'))
    console.log(response)

    dispatch(fetchData(getState().user.params))

    return response.data
  }
)
```

Roles:

Now we will move towards our next module which is roles Begin with the fetching data from So here we are displaying Role Name ,Permissions ,Actions (view ,edit ,delete) backend and displaying list of users over the table



Roles List

ROLE NAME	PERMISSIONS	ACTIONS
Admin	Register_user Create_user Update_user Delete_user Update_password Show_users Show_me Update Create_post Update_post Show_post Delete_post Create_policy Update_policy Show_policy Delete_policy Create_admin	Delete
superAdmin	Register_user Create_user Update_user Delete_user Update_password Show_users Show_me Update Create_post Update_post Show_post Delete_post Create_policy Update_policy Show_policy Delete_policy Create_role Update_role Show_role Delete_role Create_superadmin Create_admin Update_admin	Delete

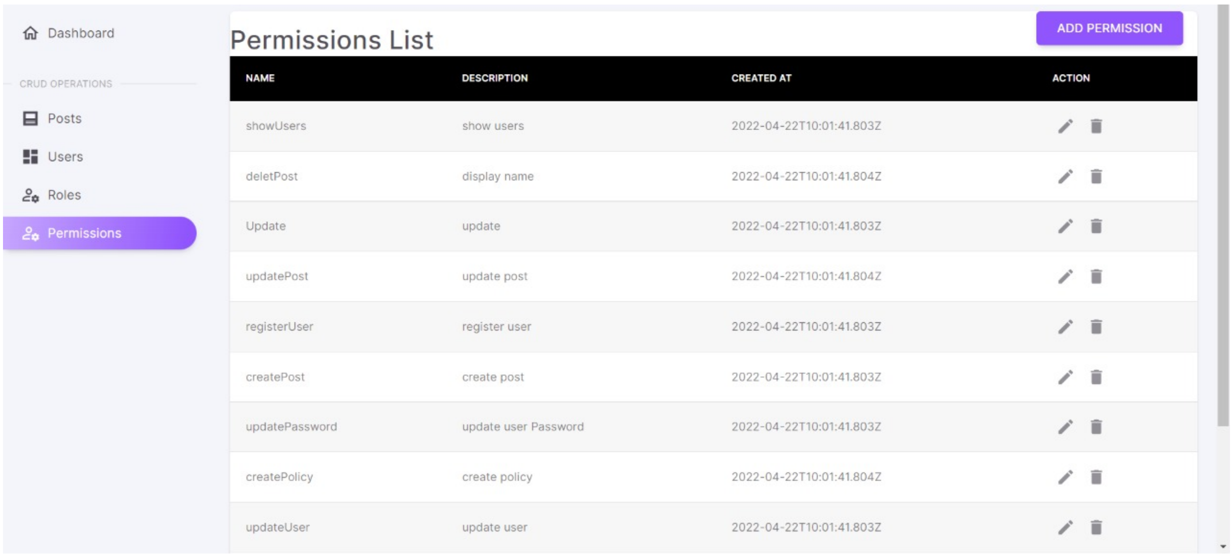
Rows per page: 10 ▾ 1-2 of 2 ‹ ›

Permissions :

Now we will move towards our next module which is Permissions table

Begin with the fetching data from So here we are displaying Name ,Description ,Created At ,Actions (view ,edit ,delete)

backend and displaying list of users over the table



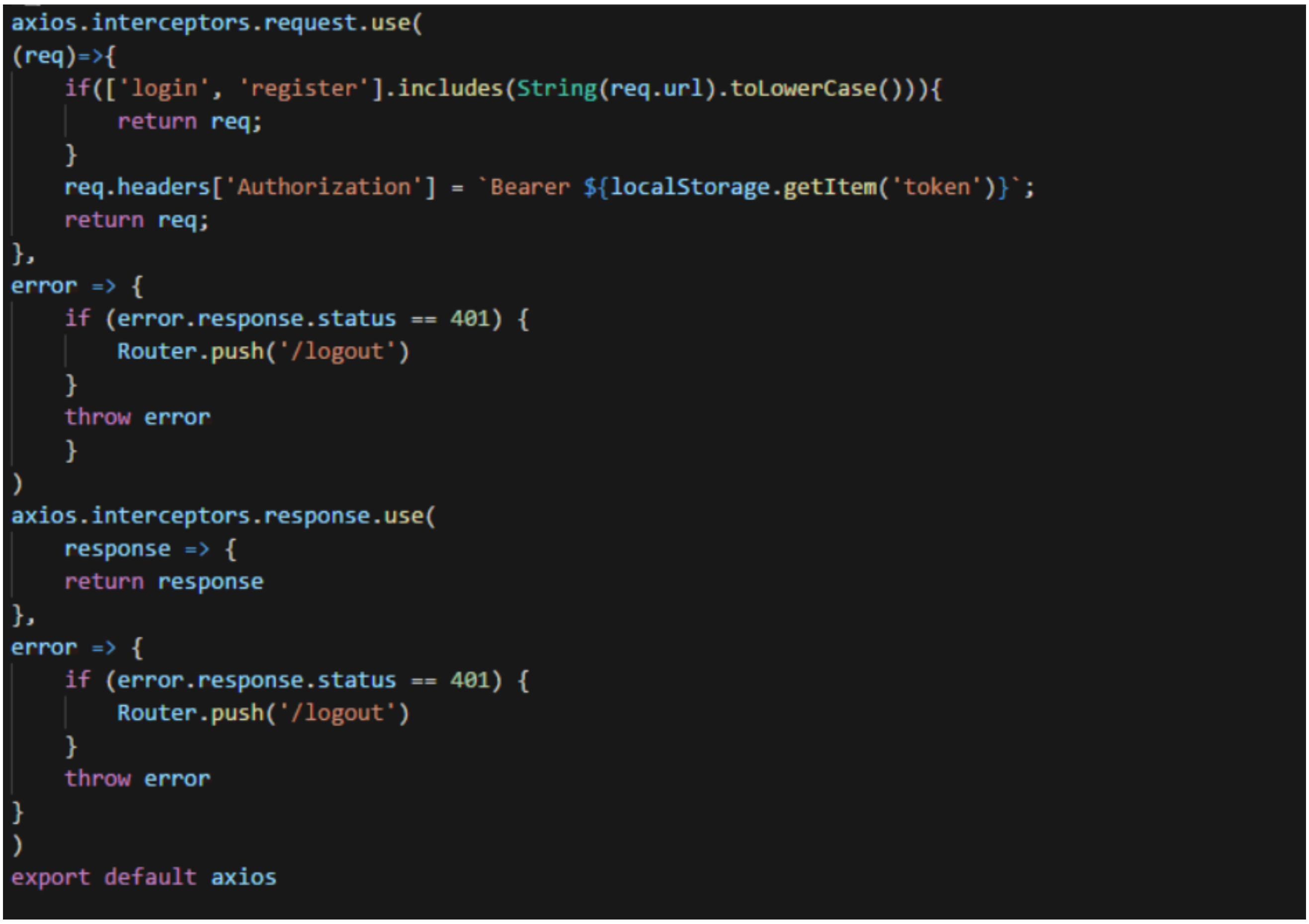
Permissions List

NAME	DESCRIPTION	CREATED AT	ACTION
showUsers	show users	2022-04-22T10:01:41.803Z	✎ 🗑
deletPost	display name	2022-04-22T10:01:41.804Z	✎ 🗑
Update	update	2022-04-22T10:01:41.803Z	✎ 🗑
updatePost	update post	2022-04-22T10:01:41.804Z	✎ 🗑
registerUser	register user	2022-04-22T10:01:41.803Z	✎ 🗑
createPost	create post	2022-04-22T10:01:41.803Z	✎ 🗑
updatePassword	update user Password	2022-04-22T10:01:41.803Z	✎ 🗑
createPolicy	create policy	2022-04-22T10:01:41.804Z	✎ 🗑
updateUser	update user	2022-04-22T10:01:41.803Z	✎ 🗑

Interceptor:

We created request and response interceptor to monitor and log our requests and responses.

By implementing Interceptor we can handle error in our request and response and we can pass the authorisation token through interceptor in every request instead of passing individually in every request.



```
axios.interceptors.request.use(
(req)=>{
    if(['login', 'register'].includes(String(req.url).toLowerCase())){
        return req;
    }
    req.headers['Authorization'] = `Bearer ${localStorage.getItem('token')}`;
    return req;
},
error => {
    if (error.response.status == 401) {
        Router.push('/logout')
    }
    throw error
    }
)
axios.interceptors.response.use(
    response => {
    return response
},
error => {
    if (error.response.status == 401) {
        Router.push('/logout')
    }
    throw error
}
)
export default axios
```

Chapter -4 Conclusion

I am still on the way doing my training and soon will start working on the live project with the RTDS and I have learned so much from this internship offered by the internship, which really helped me in shaping my personality and equipping me with the knowledge of this technologies. I will give my best in doing the internship project.

I would like to thank in advance to the mentor , trainer and my manager Mr. Aman Choudhary who guided me through the whole journey of my internship in RTDS and solved all my doubts during the internship. The Coaches, Sme , Mentor and trainer were all of good nature and at every moment helped me when I was doing wrong and shaped me during my whole internship. Specially the my mentor gave his more effort during the internship and passed our all query to the higher authority in the company whether it was not understanding a topic ,bug resolving , technical issue faced or providing extra time to complete the work.

I would highly recommend my juniors to prepare well for the offer in RTDS and get the internship opportunity because RTDS is a great company in the information technology field and there is a lot to learn .

I like thank you my TNP officer Mr. Pankaj Kumar and Faculty member Dr. Nafis U khan sir for their support and hard work during the whole placement process because I know how complex is the management of the placement drive.

CHAPTER -5 REFERENCES

- https://developer.mozilla.org/en-US/docs/Web
- https://www.w3schools.com/xml/dom_nodetype.asp
- https://nodejs.org
- https://www.mongodb.com