Internship Report At Paxcom India Pvt. Ltd.

Project report submitted in partial fulfillment of the requirement for the degree of

Bachelor of Technology in

Computer Science and Engineering

By

Vaibhav Thakur (181368)

Under the supervision of

Mr. Vikas Kumar

Department of Computer Science & Engineering and Information Technology

Jaypee University of Information Technology Waknaghat, Solan- 173234, Himachal Pradesh

CERTIFICATE

I hereby declare that the presented report entitled "Internship report at Paxcom India Pvt Ltd" in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering 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 2021 to May 2021 under the supervision of Mr. Vikas Kumar. The matter embodied in the report has not been submitted for the award of any other degree or diploma.

Vaibhav Thakur
181368

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

Vikas Kumar.

Mr. Vikas Kumar
Technical Lead
Paxcom India Pvt. Ltd.

ACKNOWLEDGEMENT

This report is not just a result of hard work by me but there has been a joint contribution by a lot of other people who I would like to thank.

I would like to thank Madhuri Sankhyan, Manager of HR in Talent Acquisition, of Paxcom India (P) Ltd - A Paymentus Company, Delhi for allowing me to do an internship within the organization.

I also would like to thank Mrs. Pallavi Vasishta and all the people that worked along with me at Paxcom India (P) Ltd - A Paymentus Company, Delhi with their patience and openness they created an enjoyable working environment.

I am highly indebted to my manager Mr. Vikas Kumar, for her continuous monitoring, providing details and her assistance in the completion of the training.

I want to express my appreciation for Mr. Pankaj Kumar, my parents and for Jaypee University of Information technology for their kind cooperation and assistance in the completion of the training.

I also thank and appreciate my mentor and colleagues for helping in completing the Internship and people who have voluntarily helped to understand the technologies.

ABSTRACT

The internship was focused on learning various aspects of frontend and backend development on MERN and MEAN stack. During the course of my Internship with Paxcom I was able to learn many new technologies. I created multiple learning projects during my internship period.

The very first project was on DOM manipulation in which by using ES6 JavaScript I built user add, update and delete features. Then the same project I built using JavaScript templating library EJS. Then worked on building an API for CRUD operations using NodeJS and ExpressJS and testing them on POSTMAN. Further I connected them to a frontend built on ReactJS. Also built few POC on redux and using context Library in ReactJS.

I learnt Angular and also did authentication of users using JWT tokens. Features like change password, update credentials, update and delete users were added.

Table of Contents

1. INTRODUCTION **1**

1.1 INTRODUCTION 1

1.2 ABOUT COMPANY 1

1.3 CORE VALUES 2

1.4 EXPERIENCE 2

2. LITERATURE SURVEY **4**

3. SYSTEM DEVELOPMENT **15**

4. PERFORMANCE ANALYSIS **21**

5. CONCLUSION **28**

6. REFERENCES **34**

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

I got placed at Paxcom India Pvt. Ltd. and carried out my 12 weeks of mandatory internship in the company. I was put in with Mr. Vikas Kumar's team. My title during the course of internship was Software Engineer- Trainee. I was mainly responsible for building API and frontend of webapps. The functioning of the team is very smooth due to the core values followed and is also highly productive. We have a good number of projects which are operational simultaneously and the outcomes are really impressive.

1.2 ABOUT COMPANY

Paxcom has the trust of some of the world's major brands and distributors, such as Britannia, Wipro, Mondelez and Pepsico, among others. In addition, Paxcom offers e-commerce solutions for product sales on India's biggest ecommerce sites like amazon, lazada, flipkart, etc. We serve a huge range of options including product intelligence, business intelligence and audit and compliance. We assist customers with data-driven decisions on pricing, advertising strategy, positioning, and other aspects of a product. Since 2019, Paxcom has been a member of the Paymentus Corporation. Paymentus, headquartered in North America, is a leading provider of e-billing and payment solutions. Paymentus enables customers to make real-time payments to businesses through any payment channel. It offers consumers a full and comprehensive payment suite, including premier collaborations such as Amazon Pay, PayPal, and Walmart.
Paxcom makes it easier for both brand owners and dealers to simplify day-to-day operations by focusing on optimizing the market, item content/perceivability, and limited-time methodologies of data-driven examination and bits of knowledge.
Paxcom's goal is to be your trusted team with a diverse range of abilities – your analysis consultant, product expert, advancement administrator, web-based industry researcher, data analyst, and expert specialist.
The organization's mission is to provide beneficial interactions to our customers that will allow them to track their competitors, screen their products, advance their content through

commercial centers, increase perceivability, and promote their offers, among other things. Paxcom was acquired by the Paymentus group a few years back, and the company's name was changed from "Paxcom" to "Paxcom India - A Paymentus Company."
Paymentus was established in 2004 as a result of a desire to change the way people pay their bills. Paymentus has grown from a vision to becoming the leading paperless electronic billing and installment arrangement available, with over 1,500 customers including some of North America's largest billers.
Paymentus consistently strives to grow stronger, faster, increasingly safer, cost-effective charging and installment phase and is identified by Deloitte as one of the fastest growing North American entities in 2011, 2013, 2014, and 2016. We are constantly on the lookout for more value for our customers, both in terms of contracts and management.

1.3 CORE VALUES

There are 3 main core values which helps identify what Paxcom actually is. These are the pillars on which we employees stand on. The huge increase in number of clients has been made possible due to these values - Integrity, Fun and Ownership.
Integrity - this is one of the most important values that employees at Paxcom possess as a person with integrity draws others to them and helps identify one as dependable and trustworthy.
Fun - this is depicted in the Paxcom work culture. We believe in hard work and fun. Events take place from time to time in which all the employees participate with enthusiasm. This helps in increasing productivity and creates a better work environment for everyone.
Ownership - Every employee is responsible for his/her success and failure. Ideas are openly welcomed. Company culture discourages use of Sir/Madam within the office.

1.4 EXPERIENCE

I started my internship with Paxcom India Pvt. Ltd. on 7th February, 2021. I am posted at the New Delhi office but due to the current pandemic we are following work from home. I joined the Web development team and was successfully able to understand the work pattern and structure. I was given multiple technologies to explore and learn during my internship like Javascript, Nodejs, ExpressJS, Git/Github, NoSQL based database like MongoDB, React and Angular. I was also provided with multiple projects to understand the project structure and also have ideas about its flow. I also got to learn soft skills like working in a team, communication within a corporate firm. Because of this training I was able to understand some of the company's products.

CHAPTER 2
LITERATURE SURVEY

During my internship at Paxcom I learnt about various technologies like javascript, typescript, NodeJs, ExpressJs, React, Angular, MongoDB, Github. I also learnt about Postman and React Redux.

Javascript

JavaScript is the most widely used programming language on the planet. The Web's programming language is JavaScript. JavaScript is a simple language to learn.
JavaScript (JS) is a first-class compiled programming language that is lightweight, interpreted, or just-in-time compiled. While it is best known as a scripting language for Web pages, it is also used in a variety of non-browser settings, including Node.js, Apache CouchDB, and Adobe Acrobat. JavaScript is a single-threaded, prototype-based, dynamic language that supports object-oriented, imperative, and declarative (e.g. functional programming) programming styles.

Typescript

TypeScript allows you to write JavaScript in the way you desire. TypeScript is a typed JavaScript superset that compiles to plain JavaScript. TypeScript is a pure object-oriented programming language with classes, interfaces, and statically typed code, similar to C# or Java. TypeScript is used in the popular JavaScript framework Angular 2.0. Understanding TypeScript may assist programmers in writing object-oriented applications that are compiled to JavaScript on both the server and client sides.
TypeScript will be simple to use for programmers who come from an Object-Oriented background. They can create web apps considerably quicker if they know TypeScript, because TypeScript offers strong tooling support.
Microsoft created and maintains the TypeScript programming language. It's a syntactical

superset of JavaScript with the addition of optional static typing. It is designed for the development of large applications and transpiled to JavaScript.

NodeJS

Node.js is a JavaScript runtime environment that is free and cross-platform. It's a well-liked tool for practically any job!

Outside of the browser, Node.js operates the V8 JavaScript engine, which is at the heart of Google Chrome. As a result, Node.js is extremely fast.

Without establishing a new thread for each request, a Node.js software operates in a single process. Node.js' standard library includes a set of asynchronous I/O primitives that prevent JavaScript code from blocking, and libraries in Node.js are often created following non-blocking paradigms, thus blocking behavior is the exception rather than the rule.

Instead of stalling the thread and wasting CPU cycles waiting for Node.js to complete an I/O operation, such as reading from the network, accessing a database, or accessing the filesystem, Node.js will continue the operations when the response comes back.

This allows Node.js to handle thousands of concurrent connections with a single server without adding the overhead of thread concurrency management, which may be a major cause of errors.

Because millions of frontend developers who write JavaScript for the browser can now create server-side code in addition to client-side code without having to learn a new language, Node.js offers a distinct advantage.

The new ECMAScript standards can be utilized without issue in Node.js since you don't have to wait for all of your users' browsers to update - you can choose which ECMAScript version to use by changing the Node.js version, and you can even enable certain experimental features by running Node.js with flags.

ExpressJS

Express.js, or basically Express, is a back end web application structure for Node.js, delivered as free and open-source programming under the MIT License. It is intended for building web applications and APIs. It has been known as the accepted standard server structure for Node.js.

The first creator, TJ Holowaychuk, depicted it as a Sinatra-propelled server, implying that it is generally negligible with many elements accessible as modules. Express is the back-end part of well known improvement stacks like the MEAN, MERN or MEVN stack, along with the MongoDB data set programming and a JavaScript front-end structure or library.

EJS

Embedded Javascript (EJS) is a type of Javascript that is embedded in a webpage. Node.js uses Templating as a templating engine. The template engine makes it possible to develop an HTML template with very little code. It may also inject data into an HTML template on the client side before generating the final HTML. EJS is a basic templating language that uses plain JavaScript to build HTML markup. It also aids in the integration of JavaScript into HTML websites. The default behavior of EJS is to seek for templates to render in the 'views' folder.

MongoDB

MongoDB stores data in JSON-like documents with flexible fields that may change from document to document and data structures that can evolve over time. The document model corresponds to the objects in your application code, allowing you to interact with data more easily. Ad hoc searches, indexing, and real-time aggregation are all useful tools for gaining access to and analyzing your data. Because MongoDB is a distributed database at its heart, it comes with built-in high availability, horizontal scaling, and geographic dispersion. MongoDB is a completely free database.

Postman

Postman is an application programming interface (API) development tool that aids in the creation, testing, and modification of APIs. This utility has almost all of the features that a developer would want. Every month, over 5 million developers utilise it to make API development straightforward and easy. It can make a variety of HTTP queries (GET, POST, PUT, PATCH), save environments for later usage, and convert API to code in a variety of languages (like JavaScript, Python). The URL that we wish to GET, POST, DELETE, etc. is entered into the longest middle input box, which looks like a search bar.

React

React.js was delivered by a computer programmer working for Facebook - Jordane Walke in 2011. Respond is a JavaScript library zeroed in on making decisive UIs (UIs) utilizing a part based idea. It's utilized for taking care of the view layer and can be utilized for web and versatile applications. Responds principal objective is to be broad, quick, decisive, adaptable, and straightforward.

Respond isn't a system, it is explicitly a library. The clarification for this is that React just arrangements with delivering UIs and holds numerous things at the tact of individual tasks. The standard arrangement of apparatuses for making an application utilizing ReactJS is regularly called the stack.

High level libraries, for example, React, create a tree of components in memory identical to the genuine DOM, which frames the virtual DOM in a decisive manner. The virtual DOM is one of the elements that make the system so quick and dependable. JSX utilizes Babel preprocessors to change over HTML-like text in JavaScript documents into JavaScript objects to be parsed.

Respond doesn't need the utilization of JSX, yet most engineers find that it makes for a more easy to use insight inside the JavaScript code.

Angular

Precise is an open-source, JavaScript structure written in TypeScript. Google keeps up with it, and its main role is to foster single-page applications. As a system, Angular enjoys clear benefits while likewise giving a standard design to engineers to work with. It empowers clients to make huge applications in a viable way.

JavaScript is the most normally utilized client-side prearranging language. It is composed into HTML reports to empower communications with website pages in numerous extraordinary ways. As a generally simple to-learn language with unavoidable help, creating present day applications is appropriate.

Yet, is JavaScript ideal for creating single-page applications that require seclusion, testability, and engineer efficiency? Maybe not.

We now have a plethora of frameworks and libraries to choose from in order to give different options. Angular addresses many, if not all, of the challenges that developers have when utilizing JavaScript alone in front-end web development.

CHAPTER 3
SYSTEM DEVELOPMENT

I created projects during my internship using various technologies mentioned above. The projects were mainly focused on web applications and creating a full stack web app with good UI for better user experience.

3.1 User Application on NodeJS and ReactJs

- **User Crud Operation on Nodejs using Postman**

  CRUD, Express and MongoDB are big words for a person who has never touched any server-side programming in their life.

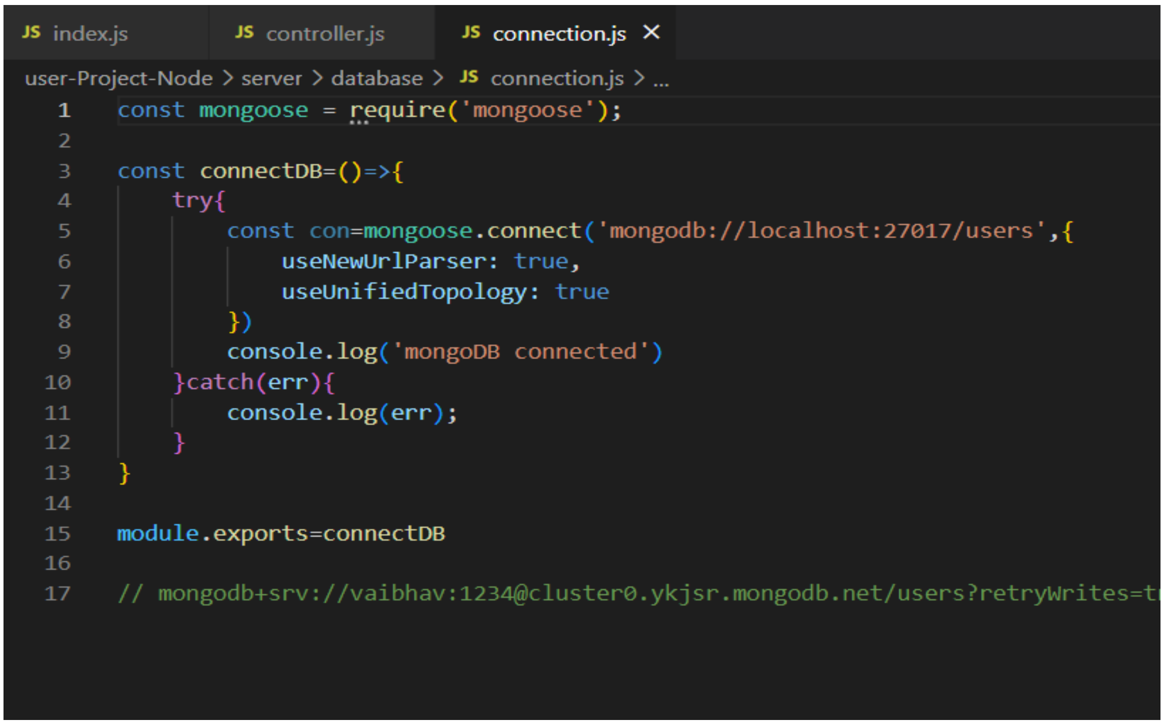
  CRUD is an acronym for Create, Read, Update and Delete. It is a set of operations we get servers to execute(GET, POST, PUT and DELETE). This is what each operation does:

  - **Create (POST) -** Make something
  - **Read (GET)-** Get something
  - **Update (PUT) -** Change something
  - **Delete (DELETE)-** Remove something

The process to start with project on terminal is :

```
1. mkdir node-user-app
2. cd node-user-app
3. npm init
4. npm install express body-parser mongoose --save
```

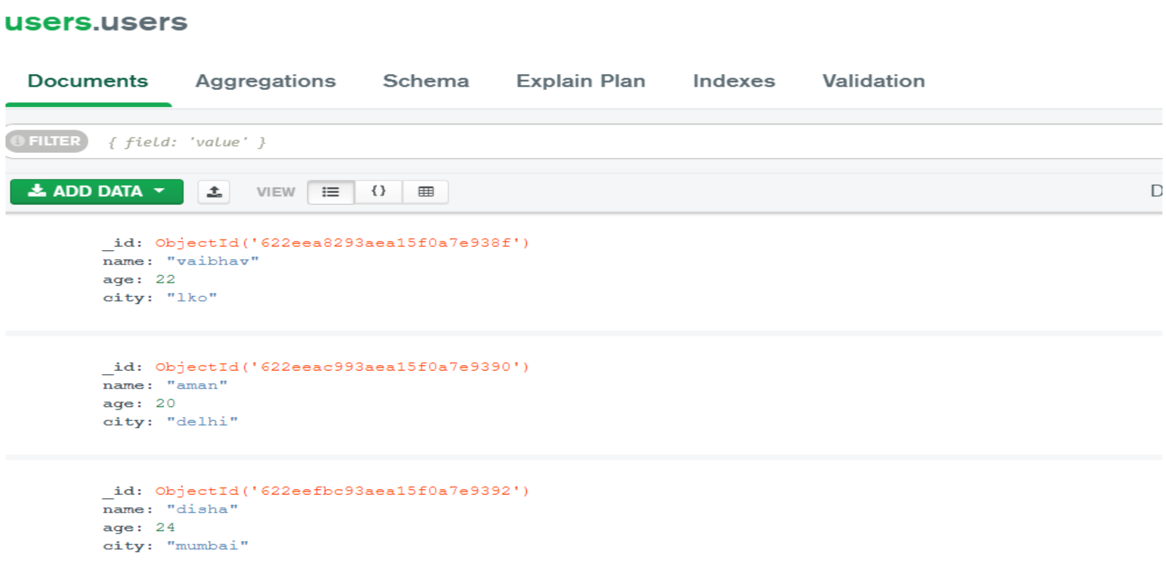
First we write the code for the connection of the database to our code. Using library 'Mongoose' to make code of mongoDB more fast working and creation of schemas much easier



JS index.js
JS controller.js
JS connection.js
user-Project-Node > server > database > JS connection.js > ...

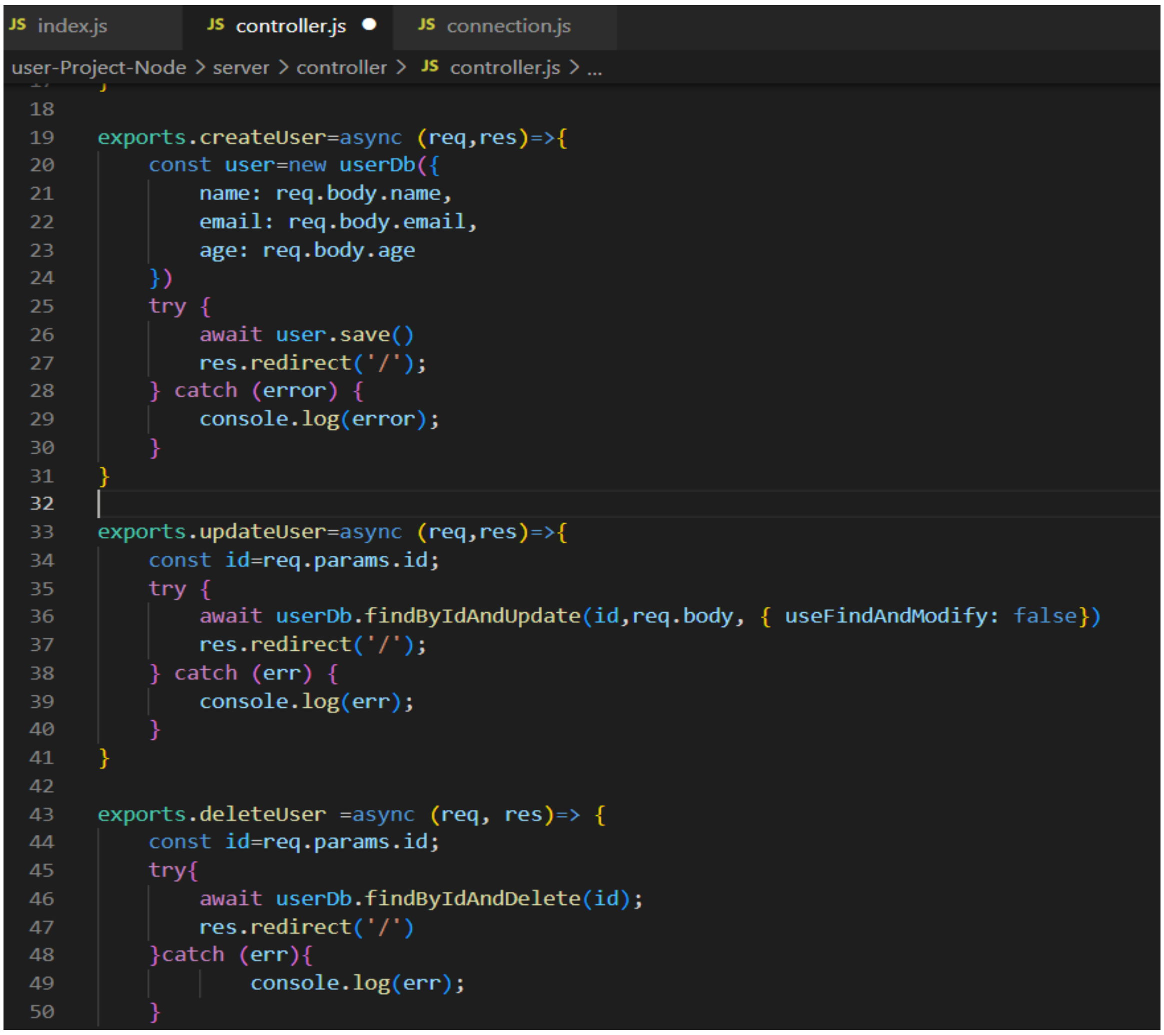
```javascript
const mongoose = require('mongoose');

const connectDB=()=>{
    try{
        const con=mongoose.connect('mongodb://localhost:27017/users',{
            useNewUrlParser: true,
            useUnifiedTopology: true
        })
        console.log('mongoDB connected')
    }catch(err){
        console.log(err);
    }
}

module.exports=connectDB

// mongodb+srv://vaibhav:1234@cluster0.ykjsr.mongodb.net/users?retryWrites=t
```

Then we can see data is stored in database

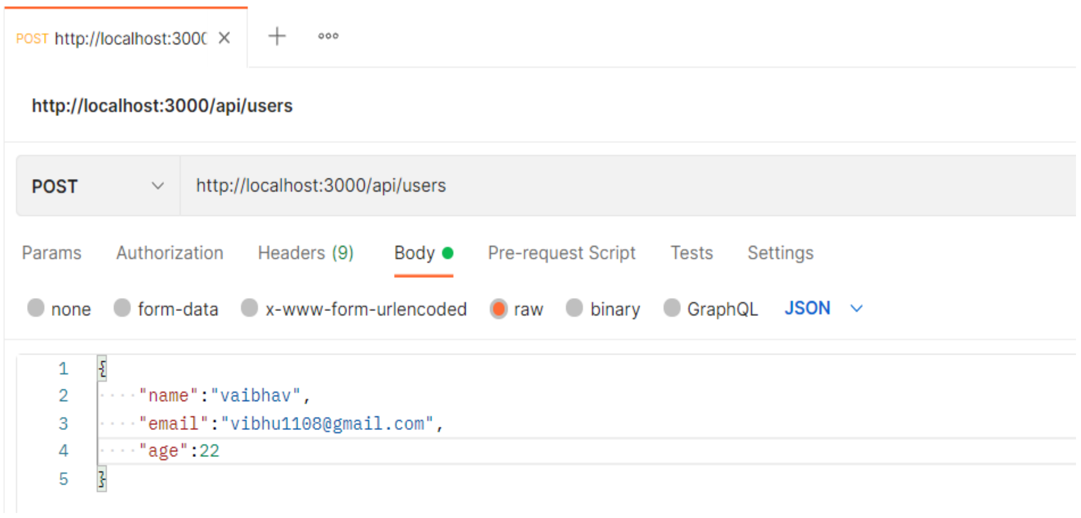


```
_id: ObjectId('622eea8293aea15f0a7e938f')
name: "vaibhav"
age: 22
city: "lko"

_id: ObjectId('622eeac993aea15f0a7e9390')
name: "aman"
age: 20
city: "delhi"

_id: ObjectId('622eefbc93aea15f0a7e9392')
name: "disha"
age: 24
city: "mumbai"
```

After this we write the code for various Crud operations on the server side and handling the API calls by using POSTMAN


index.js
controller.js
connection.js
user-Project-Node > server > controller > controller.js > ...

```
exports.createUser=async (req,res)=>{
    const user=new userDb({
        name: req.body.name,
        email: req.body.email,
        age: req.body.age
    })
    try {
        await user.save()
        res.redirect('/');
    } catch (error) {
        console.log(error);
    }
}

exports.updateUser=async (req,res)=>{
    const id=req.params.id;
    try {
        await userDb.findByIdAndUpdate(id,req.body, { useFindAndModify: false})
        res.redirect('/');
    } catch (err) {
        console.log(err);
    }
}

exports.deleteUser =async (req, res)=> {
    const id=req.params.id;
    try{
        await userDb.findByIdAndDelete(id);
        res.redirect('/')
    }catch (err){
            console.log(err);
    }
```


POST http://localhost:300(

http://localhost:3000/api/users

POST http://localhost:3000/api/users

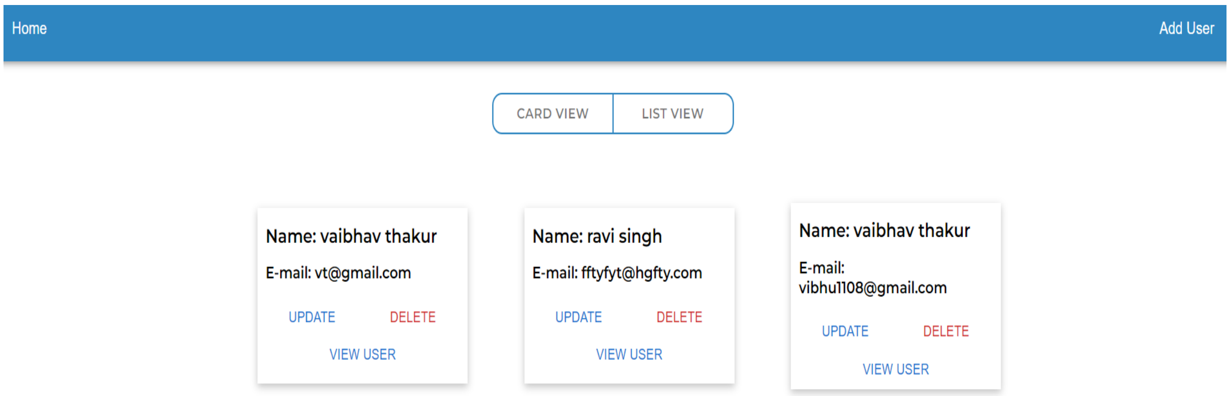
Params Authorization Headers (9) Body ● Pre-request Script Tests Settings

none form-data x-www-form-urlencoded [✓ raw] binary GraphQL JSON

```
{
    "name":"vaibhav",
    "email":"vibhu1108@gmail.com",
    "age":22
}
```

Coming forward to the client side we create a landing page where all users are displayed from the database, with actions button to update, delete and view specific users and when we click on a particular user, the details page opens up with an image and the link directed to is the specific user ID.

Also the page has a toggle for different views we want to have (i.e. cards and table views)

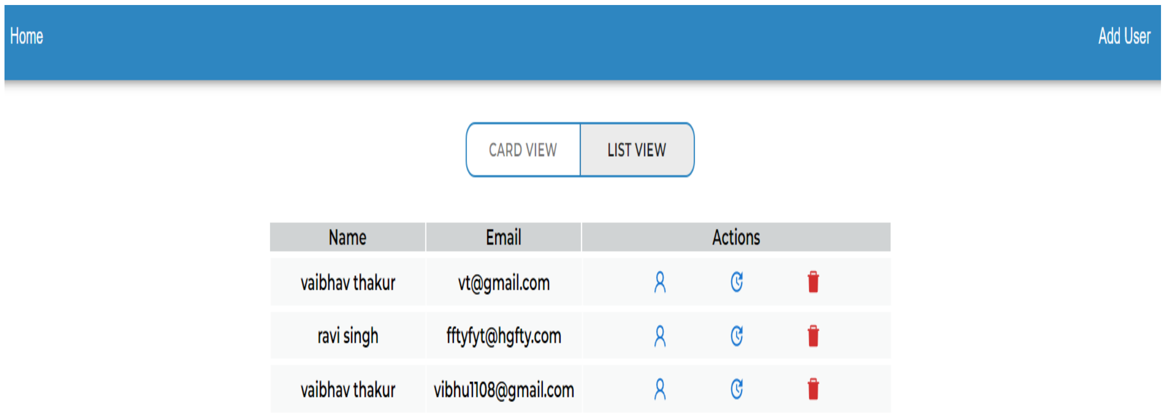

Home Add User

CARD VIEW LIST VIEW

Name: vaibhav thakur
E-mail: vt@gmail.com
UPDATE DELETE
VIEW USER

Name: ravi singh
E-mail: fftyfyt@hgfty.com
UPDATE DELETE
VIEW USER

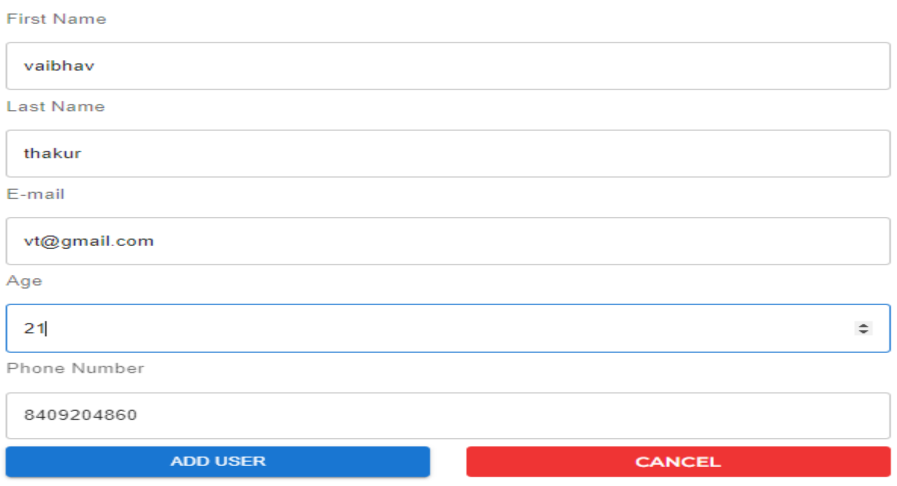
Name: vaibhav thakur
E-mail: vibhu1108@gmail.com
UPDATE DELETE
VIEW USER



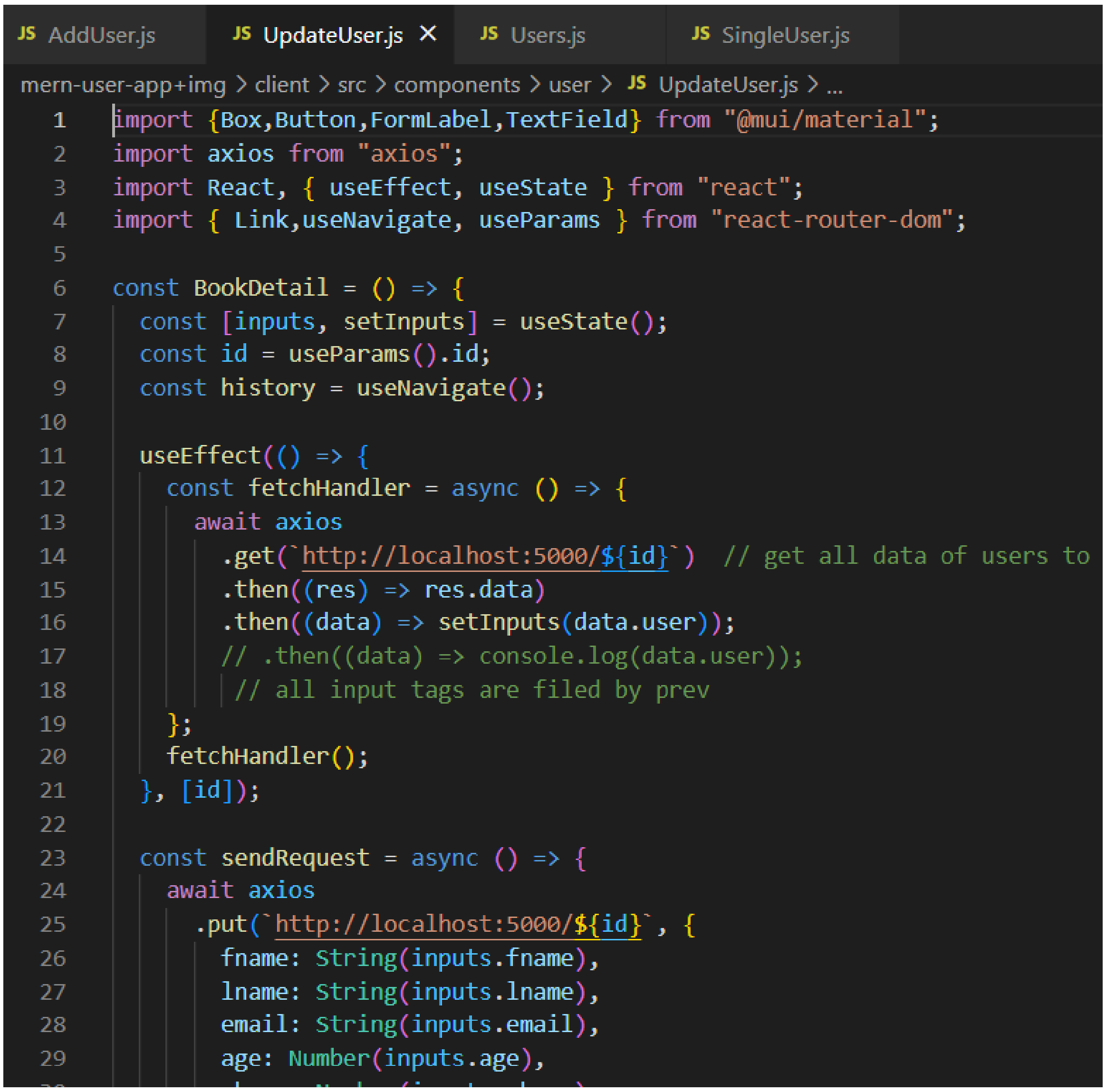
Name	Email	Actions
vaibhav thakur	vt@gmail.com	
ravi singh	fftyfyt@hgfty.com	
vaibhav thakur	vibhu1108@gmail.com	

Now when the user clicks on the AddUser button, the page opens to add the user and when he clicks on the submit button, the user is added and the page is redirected to home.

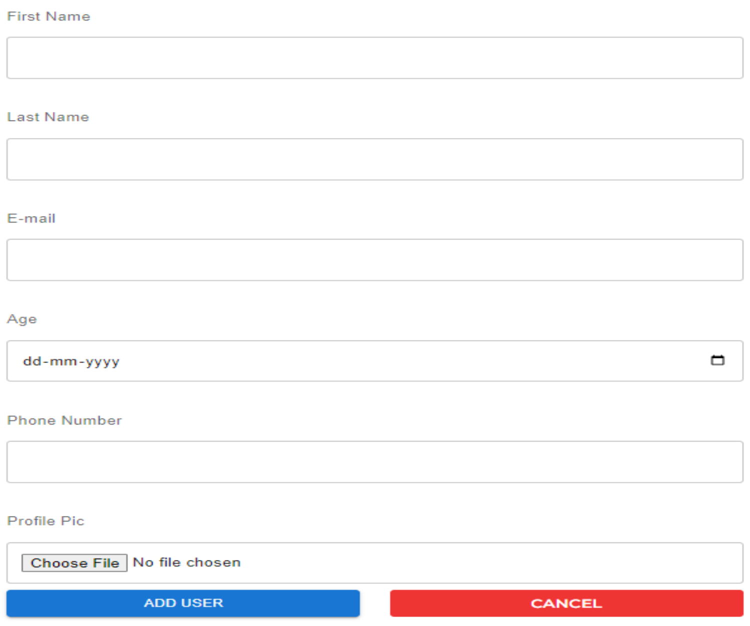
In a similar way in the UpdateUser route, the user inputs are auto populated from existing values and we can change them by overriding its value.

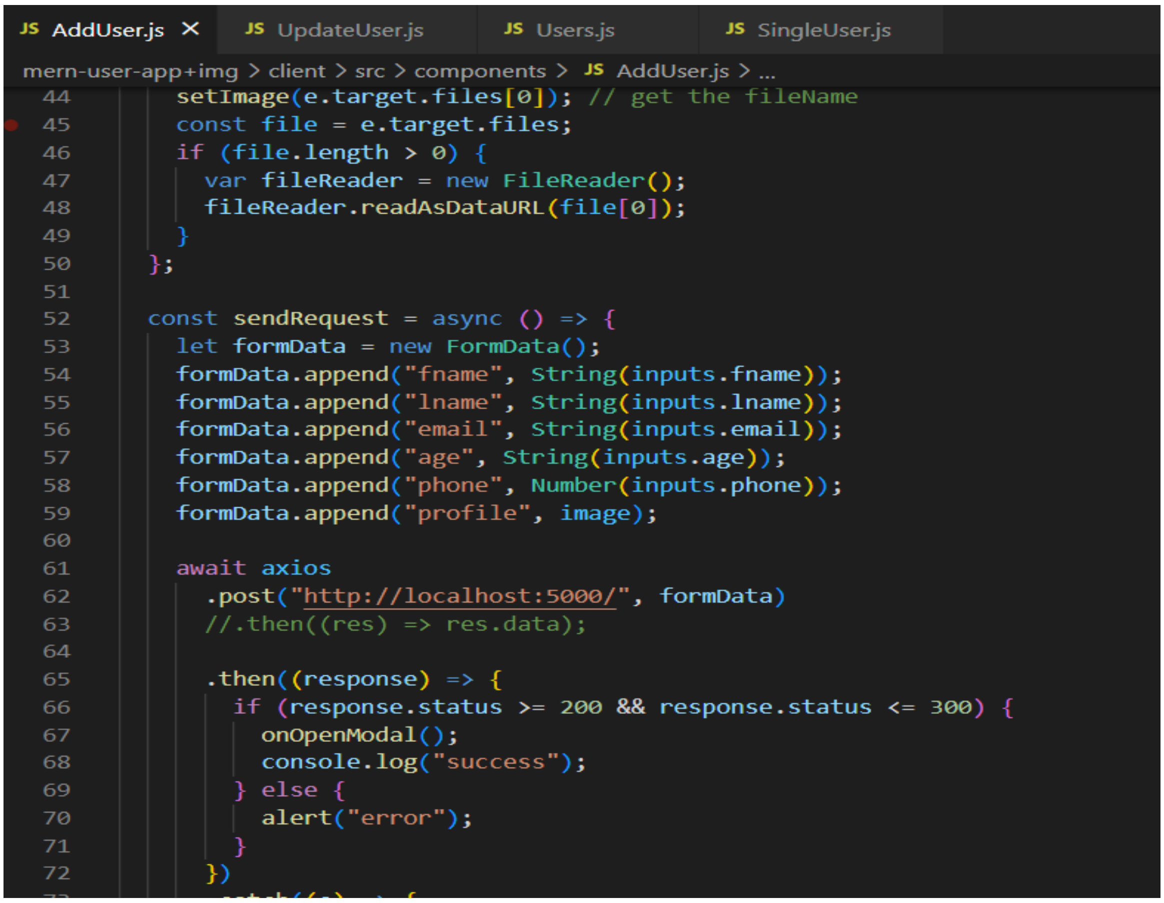


First Name	vaibhav
Last Name	thakur
E-mail	vt@gmail.com
Age	21
Phone Number	8409204860

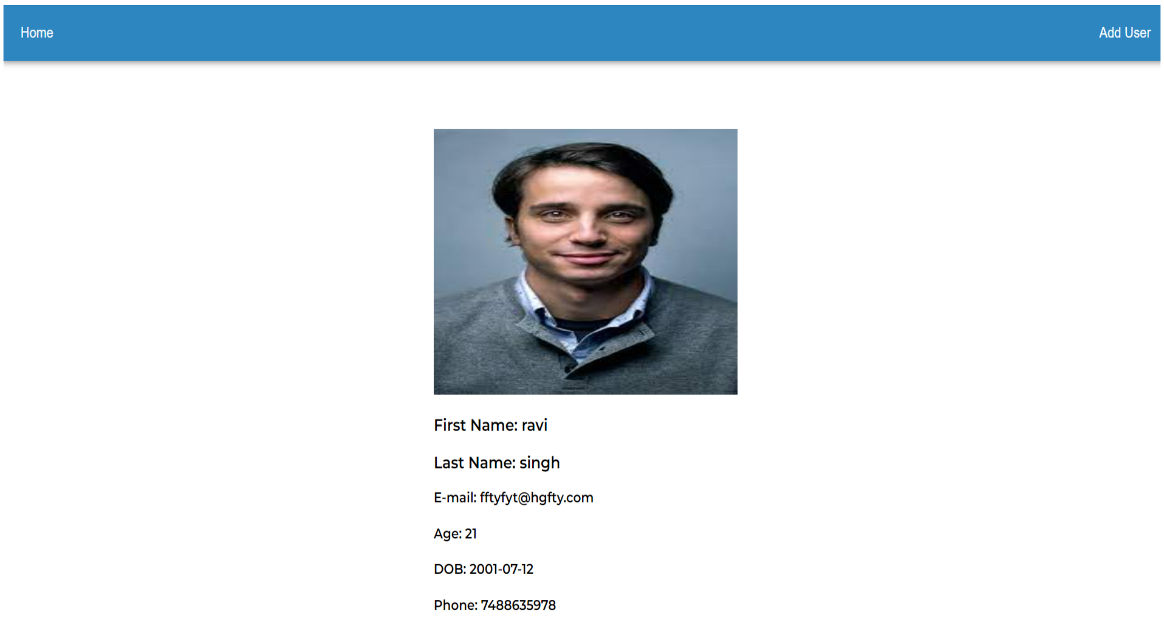

AddUser.js
UpdateUser.js
Users.js
SingleUser.js
mern-user-app+img > client > src > components > user > UpdateUser.js > ...

```javascript
import {Box,Button,FormLabel,TextField} from "@mui/material";
import axios from "axios";
import React, { useEffect, useState } from "react";
import { Link,useNavigate, useParams } from "react-router-dom";

const BookDetail = () => {
  const [inputs, setInputs] = useState();
  const id = useParams().id;
  const history = useNavigate();

  useEffect(() => {
    const fetchHandler = async () => {
      await axios
        .get(`http://localhost:5000/${id}`)  // get all data of users to
        .then((res) => res.data)
        .then((data) => setInputs(data.user));
        // .then((data) => console.log(data.user));
         // all input tags are filed by prev
    };
    fetchHandler();
  }, [id]);

  const sendRequest = async () => {
    await axios
      .put(`http://localhost:5000/${id}`, {
        fname: String(inputs.fname),
        lname: String(inputs.lname),
        email: String(inputs.email),
        age: Number(inputs.age),
```




AddUser.js
UpdateUser.js
Users.js
SingleUser.js
mern-user-app+img > client > src > components > AddUser.js > ...

```
    setImage(e.target.files[0]); // get the fileName
    const file = e.target.files;
    if (file.length > 0) {
      var fileReader = new FileReader();
      fileReader.readAsDataURL(file[0]);
    }
  };

  const sendRequest = async () => {
    let formData = new FormData();
    formData.append("fname", String(inputs.fname));
    formData.append("lname", String(inputs.lname));
    formData.append("email", String(inputs.email));
    formData.append("age", String(inputs.age));
    formData.append("phone", Number(inputs.phone));
    formData.append("profile", image);

    await axios
      .post("http://localhost:5000/", formData)
      //.then((res) => res.data);

      .then((response) => {
        if (response.status >= 200 && response.status <= 300) {
          onOpenModal();
          console.log("success");
        } else {
          alert("error");
        }
      })
```



First Name: ravi

Last Name: singh

E-mail: fftyfyt@hgfty.com

Age: 21

DOB: 2001-07-12

Phone: 7488635978

3.2 User Authentication using JSON web Token (JWT)

JSON Web Token (JWT) is an open norm (RFC 7519) that characterizes a minimized and independent way for safely communicating data between parties as a JSON object. This data can be confirmed and trusted in light of the fact that it is carefully marked. JWTs can be marked utilizing confidentiality (with the HMAC calculation) or a public/confidential key pair utilizing RSA or ECDSA.

In spite of the fact that JWTs can be encoded to likewise give mystery between parties, we will zero in on marked tokens. Marked tokens can confirm the trustworthiness of the cases held inside it, while scrambled tokens conceal those cases from different gatherings. At the point when tokens are marked utilizing public/confidential key matches, the mark likewise confirms that main the party holding the confidential key is the one that marked it.JSON Web Token (JWT) is an open norm (RFC 7519) that characterizes a minimized and independent way for safely communicating data between parties as a JSON object. This data can be confirmed and trusted on the grounds that it is carefully marked. JWTs can be marked utilizing confidentiality (with the HMAC calculation) or a public/confidential key pair utilizing RSA or ECDSA.

In spite of the fact that JWTs can be encoded to likewise give mystery between parties, we will zero in on marked tokens. Marked tokens can check the trustworthiness of the cases held inside it, while encoded tokens conceal those cases from different gatherings. At the point when tokens are marked utilizing public/confidential key matches, the mark likewise ensures that the party holding the confidential key is the one that marked it.

JWT should be used when :

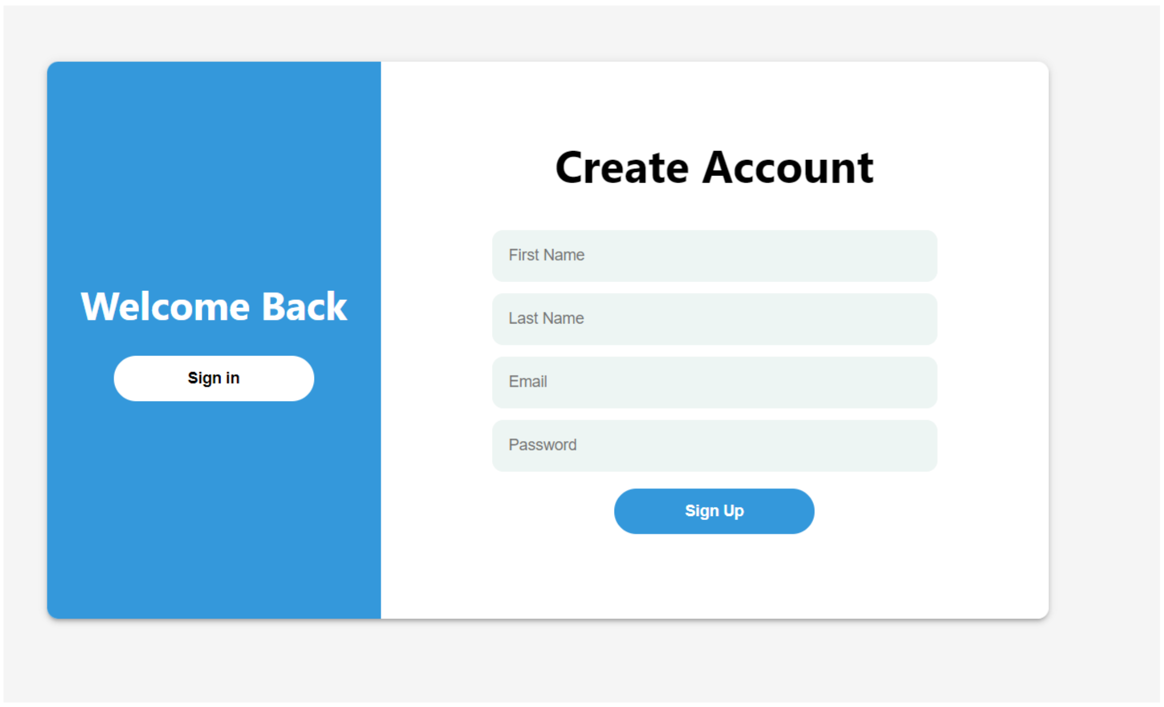
- **Authorization** is the most typical case in which JWT is used. Each subsequent request will contain the JWT once the user has signed in, allowing the user to access routes, services, and resources that are authorized with that token. Single Sign On is a popular feature that makes use of JWT nowadays due to its low overhead and ability to be utilized across many domains.

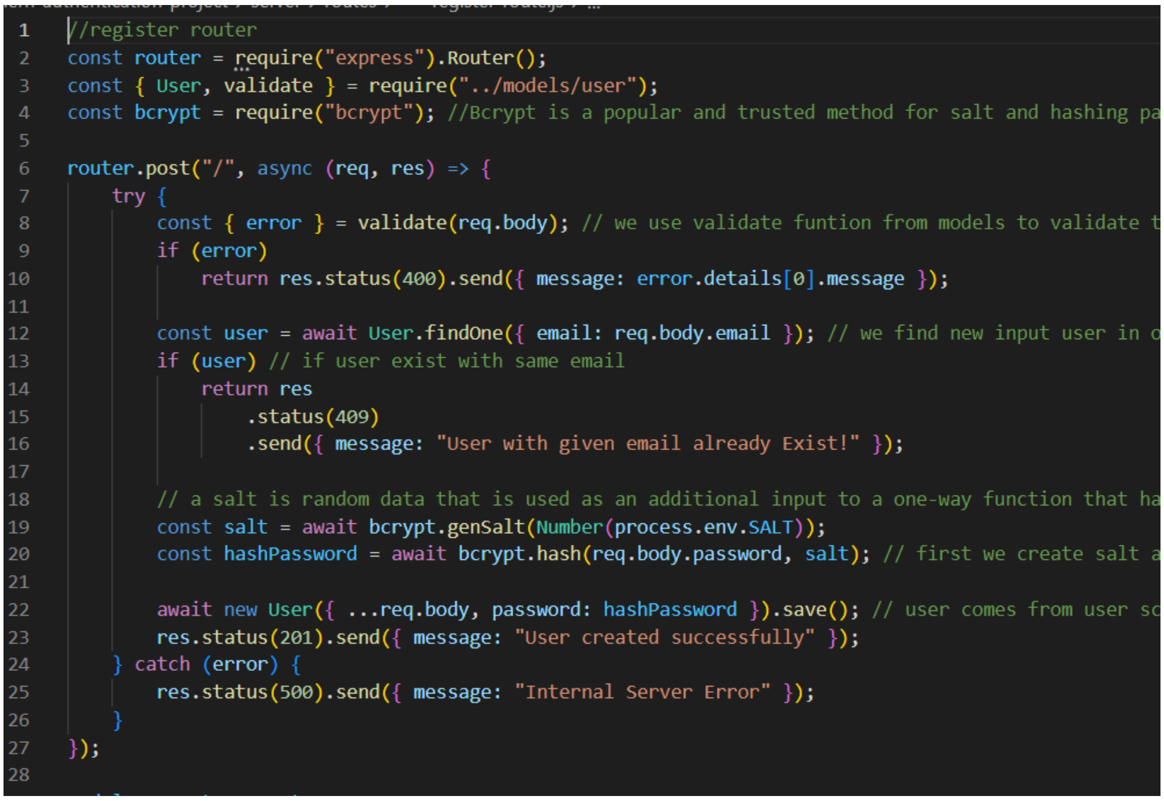
- **Information Exchange**: JSON Web Tokens are a safe mechanism to send data between two parties. You can be sure the senders are who they say they are because JWTs may be signed—for example, with public/private key pairs. You may also check that the content hasn't been altered because the signature is calculated based on the header and payload.

JSON Web Tokens are made up of three pieces that are separated by dots (.). They are:

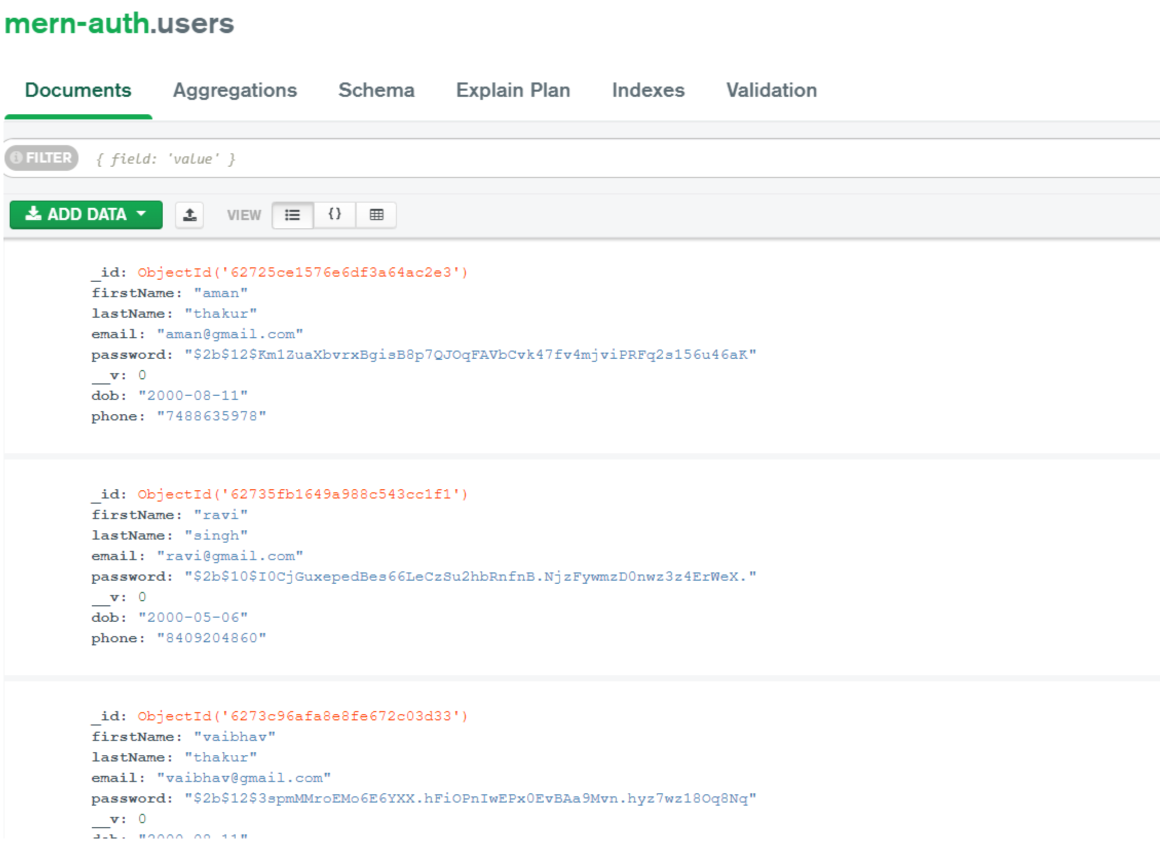
- Header
- Payload
- Signature

First the user comes to the registration page and after registration its credentials are stored in the database. Also the password is stored as in hash form by crypting so no one can see it.



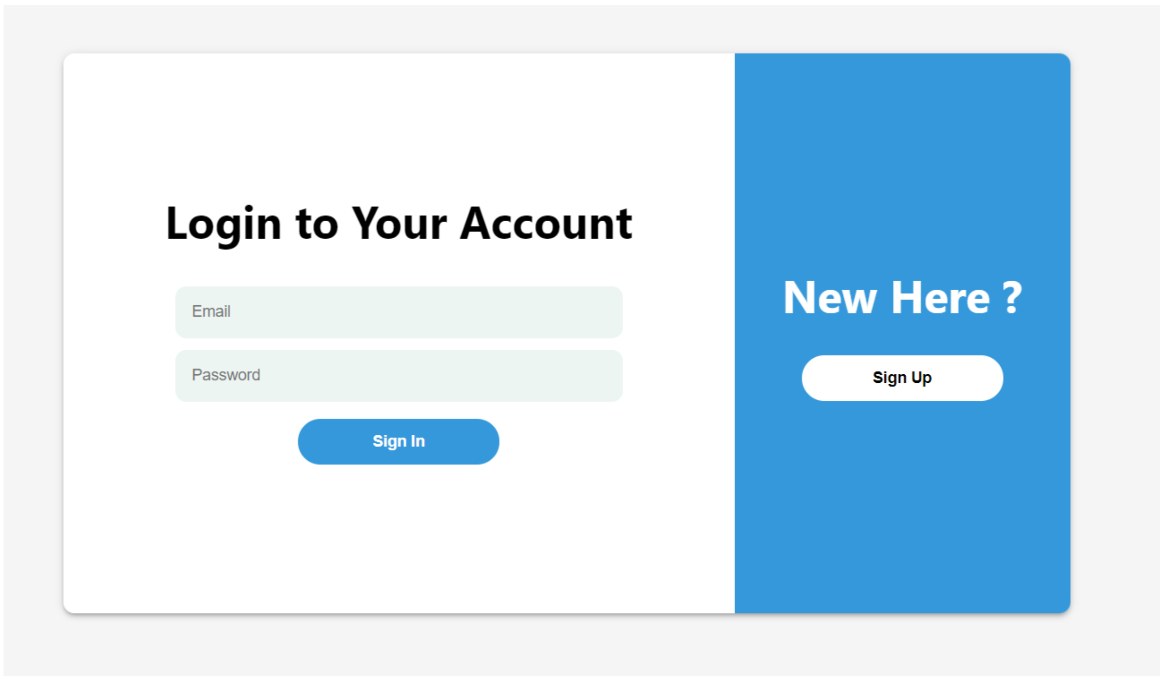


```
//register router
const router = require("express").Router();
const { User, validate } = require("../models/user");
const bcrypt = require("bcrypt"); //Bcrypt is a popular and trusted method for salt and hashing pa

router.post("/", async (req, res) => {
    try {
        const { error } = validate(req.body); // we use validate funtion from models to validate t
        if (error)
            return res.status(400).send({ message: error.details[0].message });

        const user = await User.findOne({ email: req.body.email }); // we find new input user in o
        if (user) // if user exist with same email
            return res
                .status(409)
                .send({ message: "User with given email already Exist!" });

        // a salt is random data that is used as an additional input to a one-way function that ha
        const salt = await bcrypt.genSalt(Number(process.env.SALT));
        const hashPassword = await bcrypt.hash(req.body.password, salt); // first we create salt a

        await new User({ ...req.body, password: hashPassword }).save(); // user comes from user sc
        res.status(201).send({ message: "User created successfully" });
    } catch (error) {
        res.status(500).send({ message: "Internal Server Error" });
    }
});

```

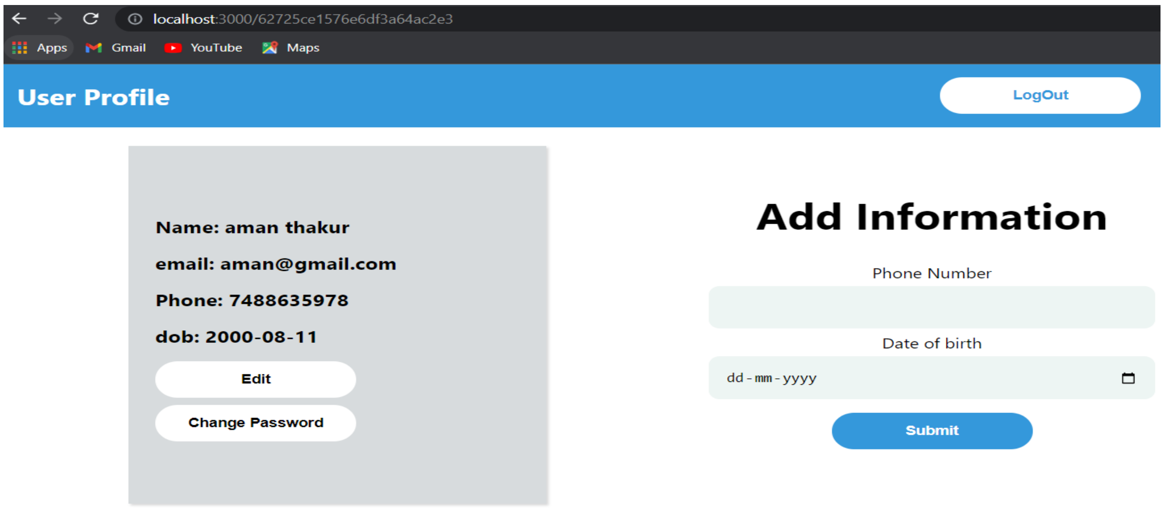


```
_id: ObjectId('62725ce1576e6df3a64ac2e3')
firstName: "aman"
lastName: "thakur"
email: "aman@gmail.com"
password: "$2b$12$Km1ZuaXbvrxBgisB8p7QJOqFAVbCvk47fv4mjviPRFq2s156u46aK"
__v: 0
dob: "2000-08-11"
phone: "7488635978"

_id: ObjectId('62735fb1649a988c543cc1f1')
firstName: "ravi"
lastName: "singh"
email: "ravi@gmail.com"
password: "$2b$10$I0CjGuxepedBes66LeCzSu2hbRnfnB.NjzFywmzD0nwz3z4ErWeX."
__v: 0
dob: "2000-05-06"
phone: "8409204860"

_id: ObjectId('6273c96afa8e8fe672c03d33')
firstName: "vaibhav"
lastName: "thakur"
email: "vaibhav@gmail.com"
password: "$2b$12$3spmMMroEMo6E6YXX.hFiOPnIwEPx0EvBAa9Mvn.hyz7wz18Oq8Nq"
__v: 0
```

After successful registration, the user goes to the login page, where he enters his/her email and password. There is a validation to check that if that email exist or not then decrypt the password to compare it.



When user logins, that specific user page opens up with its unique id and a token is stored in local storage and when user logouts that token in terminated.

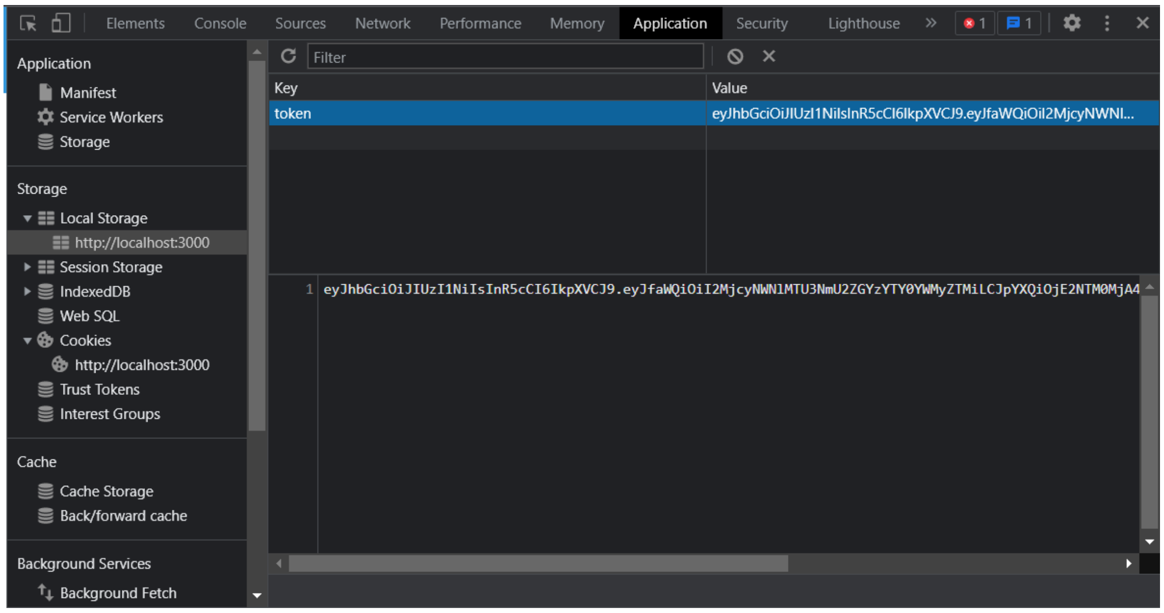

localhost:3000/62725ce1576e6df3a64ac2e3

Name: aman thakur

email: aman@gmail.com

Phone: 7488635978

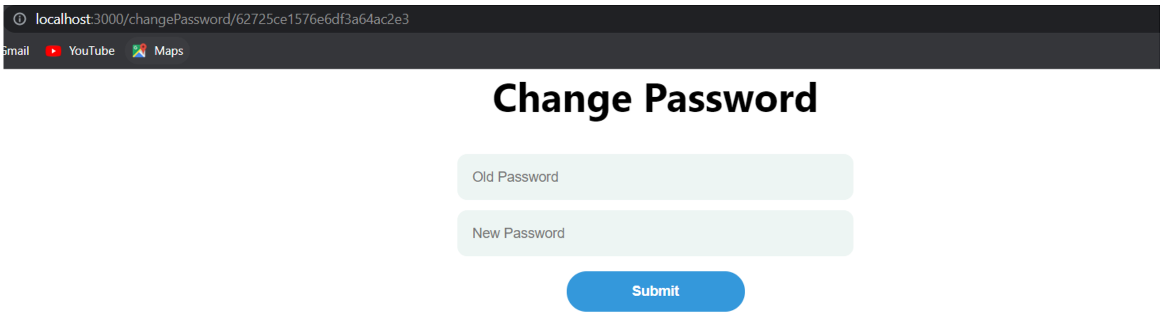
dob: 2000-08-11

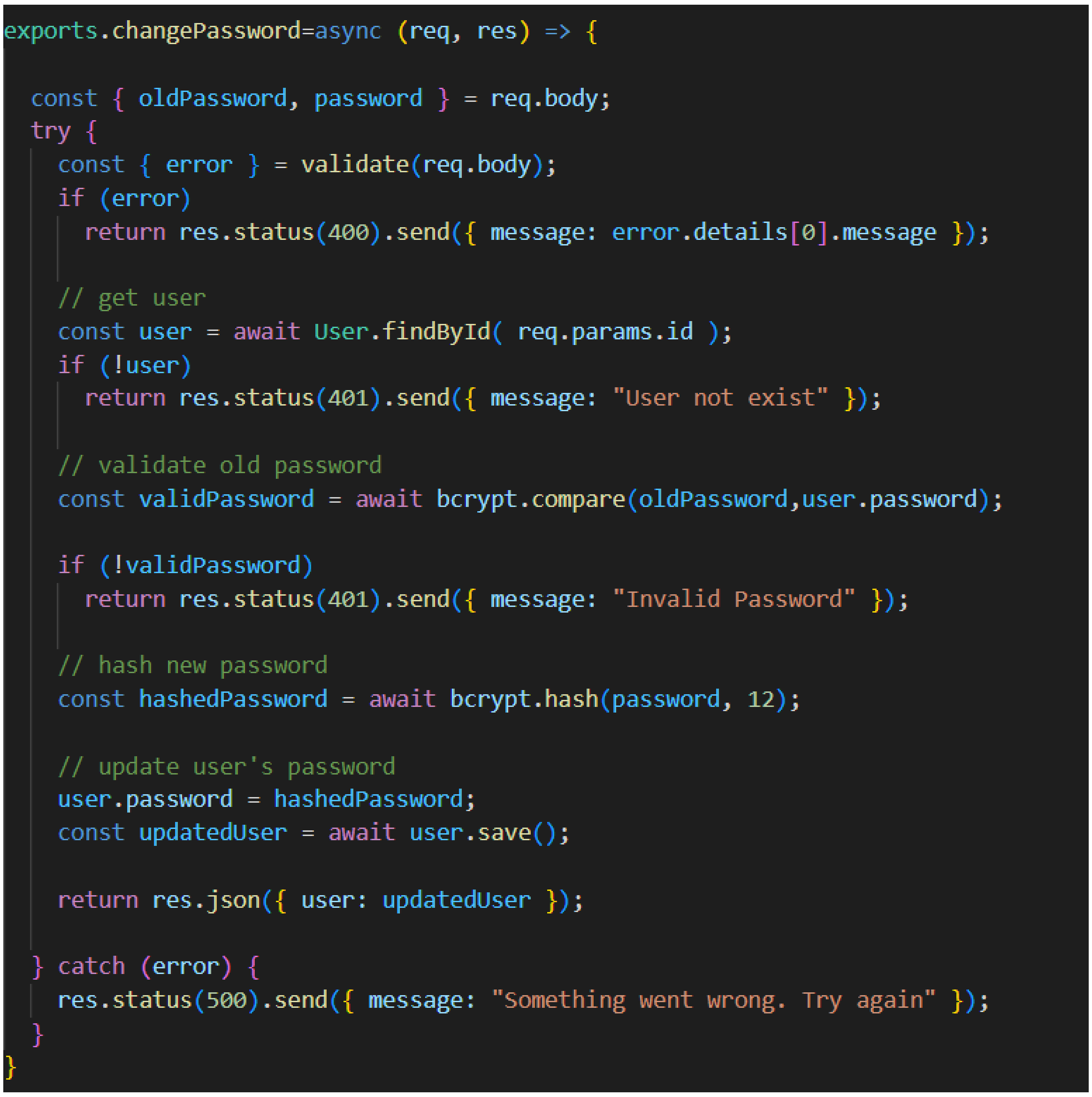


Key	Value
token	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJfaWQiOiI2MjcyNWNl...

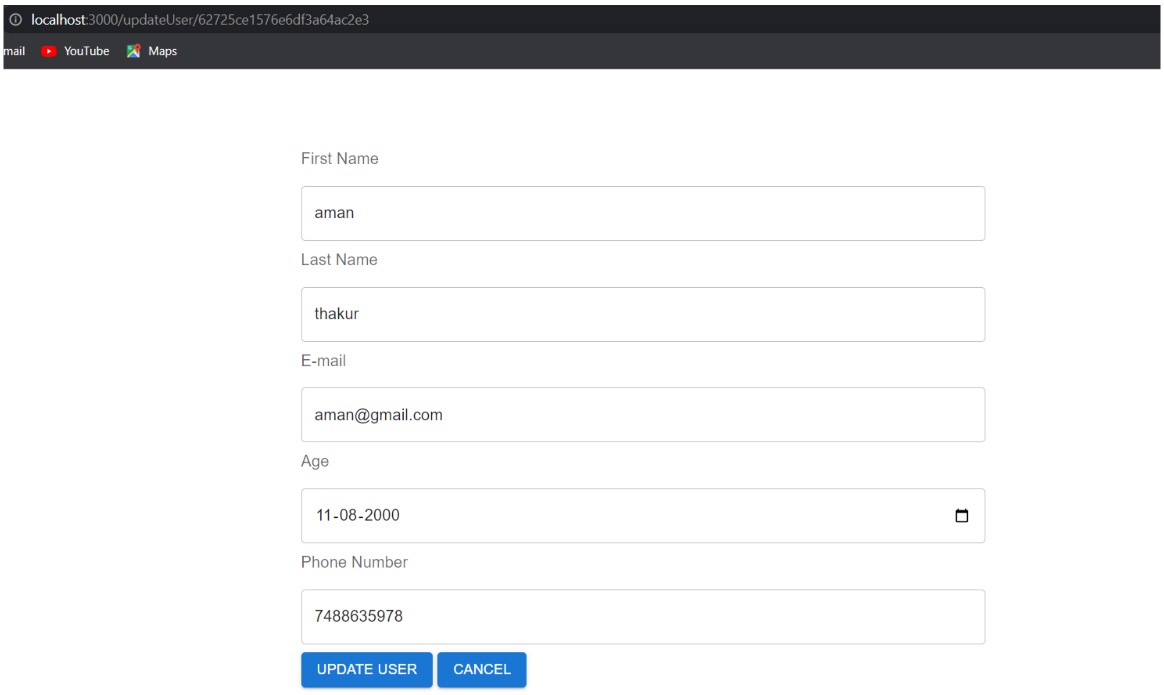
```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJfaWQiOiI2MjcyNWNlMTU3NmU2ZGYzYTY0YWMyZTMiLCJpYXQiOjE2NTM0MjA4
```

Also functionalities like change password and update user password is also added to enhance user experience.





```
exports.changePassword=async (req, res) => {

  const { oldPassword, password } = req.body;
  try {
    const { error } = validate(req.body);
    if (error)
      return res.status(400).send({ message: error.details[0].message });

    // get user
    const user = await User.findById( req.params.id );
    if (!user)
      return res.status(401).send({ message: "User not exist" });

    // validate old password
    const validPassword = await bcrypt.compare(oldPassword,user.password);

    if (!validPassword)
      return res.status(401).send({ message: "Invalid Password" });

    // hash new password
    const hashedPassword = await bcrypt.hash(password, 12);

    // update user's password
    user.password = hashedPassword;
    const updatedUser = await user.save();

    return res.json({ user: updatedUser });

  } catch (error) {
    res.status(500).send({ message: "Something went wrong. Try again" });
  }
}
```

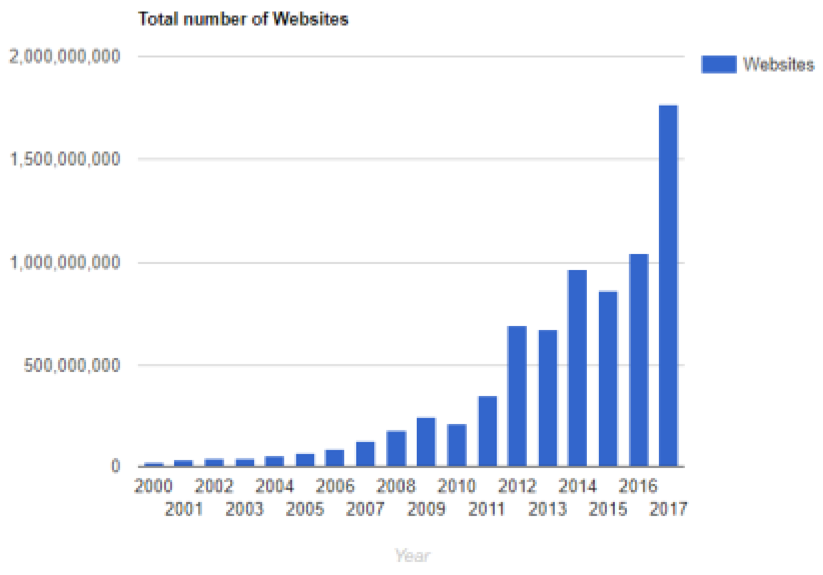

localhost:3000/updateUser/62725ce1576e6df3a64ac2e3

First Name	aman
Last Name	thakur
E-mail	aman@gmail.com
Age	11-08-2000
Phone Number	7488635978

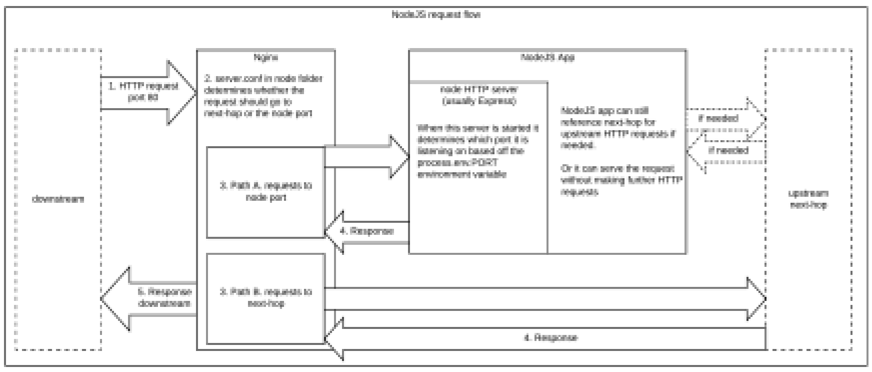
CHAPTER 4
PERFORMANCE ANALYSIS

4.1 USER APP PERFORMANCE ANALYSIS

To lay out the web application MERN stack was utilized. The web application was created utilizing the MERN stack utilizing Mongoose and MongoDB information base. Chrome designer instruments were used while testing involving revival devices for reproduction. The accompanying area talks about MERN stack parts and their execution. A. NodeJS Node.js is written in C ++ language, which is a JS working environment. Node.js is a JS runtime environment. Node.js utilizes the Google Chrome V8 motor for good execution. Node.js utilizes occasion driven, nonconcurrent programming callback works, and planned utilizing single-string engineering. Node.js configuration utilizes the occasion driven as the key center idea for its current circumstance, which provided us with the different number of APIs that are occasion based and nonconcurrent in nature which has helped us in building the site utilizing node.js for our back-end improvement.

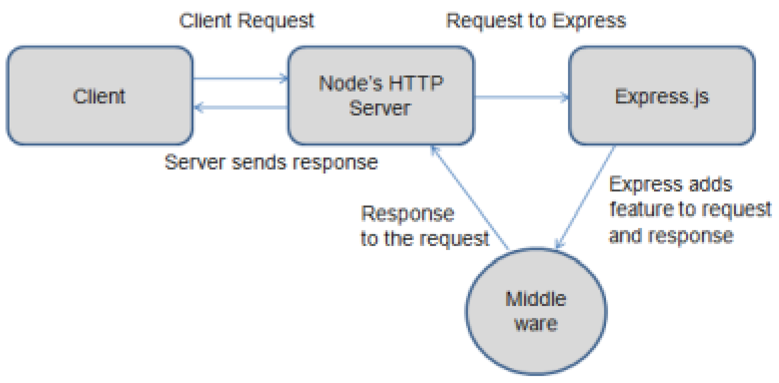


Because Express is a Node.js framework, we used it. During the development of the application, we discovered that instead of generating a slew of node modules and writing the code in NodeJS, Express made writing the back-end code and putting it into an organized fashion more simpler and easier. Express aided us in the construction of the web apps and APIs we needed for our project since it supports a large number of middlewares, making the code simpler and easier to develop. The most significant advantages of adopting Express in our application are asynchronous programming and single-threaded architecture. A comprehensive API is required for our application. To begin our express project, we created a new folder, and the procedures were as follows: we had to add a command to the command line to initialize the package. a file in json


NodeJS request flow
downstream
1. HTTP request port 80
Nginx
2. server.conf in node folder determines whether the request should go to next-hop or the node port
3. Path A. requests to node port
3. Path B. requests to next-hop
NodeJS App
node HTTP server (usually Express)
When this server is started it determines which port it is listening on based off the process.env.PORT environment variable
NodeJS app can still reference next-hop for upstream HTTP requests if needed.
Or it can serve the request without making further HTTP requests.
if needed
if needed
4. Response
4. Response
5. Response downstream
upstream next-hop

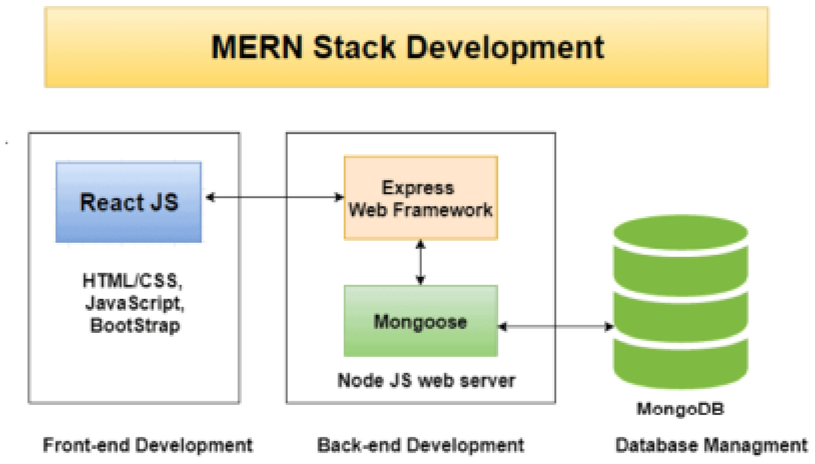
We involved React.JS for building our UI for the web application, as it is utilized for the improvement of a solitary page application since it can deliver powerfully changing information at a fast rate. Respond permits designers to code in JS and make User Interface parts. We concentrated on virtual DOM objects in React.JS, which we executed in our task. Any progressions we made in our online business web application made the whole User-Interface re-render the virtual DOM. This permits us to look at the likely distinction between the DOM Object and Virtual DOM. We utilized JSX, It made our code more straightforward and less complex to write in React application.[4] React.JS utilizes Components. Parts are the structure blocks of User-Interface wherein every part had a rationale connected with our online business application and it added to the by and large

Our web application's user interface. Components may be reused, which made our web application code easier to understand by other developers and improved the overall efficiency of the online site.



For our project, we selected MongoDB, a document-oriented database. MongoDB is a database where each record is in the form of a document. MongoDB turns our JSON data into a binary version that can be stored and searched more effectively behind the scenes on the server. To query the database, MongoDB employs BSON. MongoDB saves BSON format both internally and across the network, although this does not imply that it is a JSON database. Any data may be represented in JSON format and saved natively in MongoDB, as well as retrieved in JSON format. We may argue that MongoDB is versatile because it allows users to construct schema, databases, and tables, among other things, based on our research and implementation.

Because MongoDB is a document-oriented database, indexing documents is simple. As a result, it is able to respond at a faster rate. MongoDB is a scalable database. We managed huge data in the MongoDB database by separating it into a layered described structure. MongoDB is a database server that allows several databases to be run simultaneously.


MERN Stack Development
React JS
HTML/CSS, JavaScript, BootStrap
Express Web Framework
Mongoose
Node JS web server
MongoDB
Front-end Development
Back-end Development
Database Managment

4.2 USER AUTHENTICATION WITH JWT PERFORMANCE ANALYSIS

The jsonwebtoken is the de-facto standard in the Node.js ecosystem, so here's the golden question: how does it perform?
In both cases, the hottest frames are in jsonwebtoken package and its direct dependency, jwa (via jws). Let's see how we can improve their performance.

Node's crypto module performs the JWT signing and verifying operations. This is currently the fastest implementation available. So, we analyzed the jsonwebtoken, jws and jwa source code to see what we could do to improve performance and how.

While the jsonwebtoken implementation is robust and effective, we found a fundamental problem; all operations are orchestrated by the jsonwebtoken package using the jws and jwa packages. This is not generally a problem, but as each of the three packages are developed as standalone packages, it is. This causes the repetition of many operations (such as input type and format validations) as input must be validated at each layer. In some cases, such as the stream interface of jws, which is not used by jsonwebtoken, there is also unwanted and unused overhead.

Finally, the jsonwebtoken resides in the public API it exposes. Each time a signing, decoding or verification is performed, the same set of options is provided and validated. Even though each operation has minimal impact on each request, they add up and result in slower operations (especially in a single-threaded environment like Node.js).

The split implementation also poses a problem when sending pull requests (PR) to change the code; PRs should be sent to each of the three packages, and applied and released together to ensure the changes work correctly. This is generally difficult but is even more difficult in this case as jsonwebtoken, jws and jwa do not have a common maintainer. Therefore, using a PR to improve the existing packages was not viable. Our solution was to write a new package, fast-jwt.

The purpose of fast-jwt is to improve jsonwebtoken performance while keeping the same features and a similar API. To do this, we established the following architecture principles:

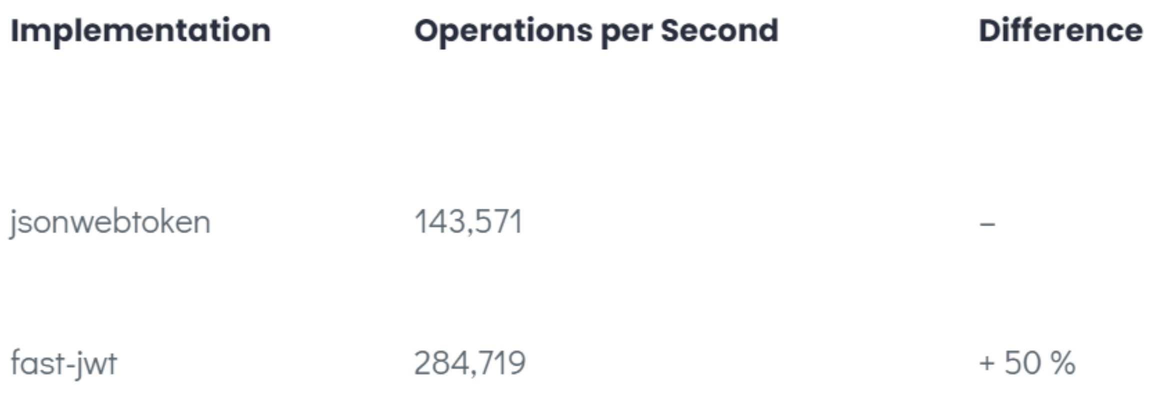
1. **Minimize the number of external dependencies:** except for the cache layer and a couple of small cryptographic utilities, fast-jwt has no external dependencies. This ensures the code is easily maintained and data flow can be followed.
2. **Use factory pattern and single ahead options verification:** fast-jwt uses the factory pattern to create the signer, decoder and verifier functions. This ensures that all options (with the exception of the key, which might be fetched at execution time, depending on the options passed) are validated only once and only during the startup phase.
3. **Small public API:** the public fast-jwt API consists of three factory functions (one for each operation) with a specific set of options.

Caching

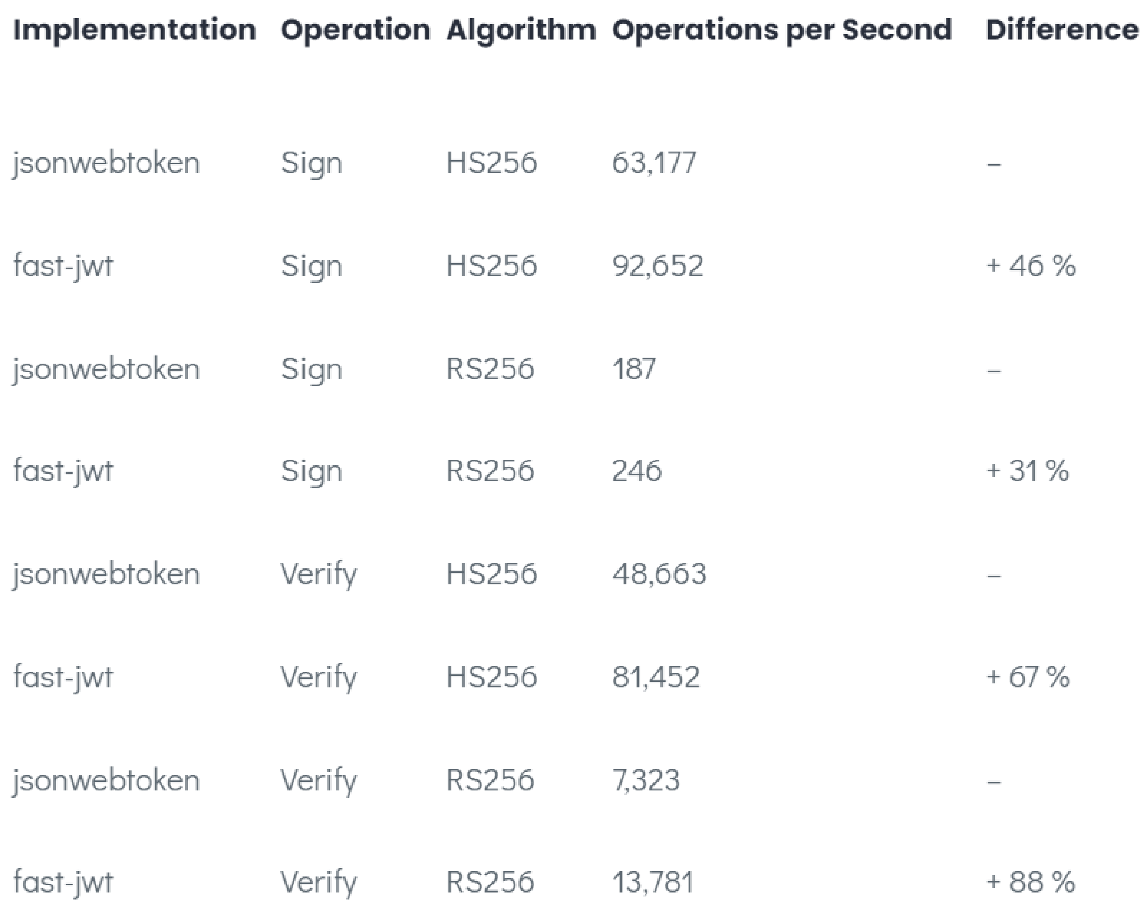
We introduced the use of caching in verify operations while developing fast-jwt to further improve performance further.

Most of the time, servers tend to process the same tokens. When verifying tokens, servers perform the same operations on the same data all the time (as typically the same user uses the same token in multiple time-close requests).

fast-jwt uses mnemonist to add a Least Recently Used (LRU) cache to all factories. Verified tokens are always cached. If verification fails, the error is also cached and the operation is not retried. Caching considers the time-sensitive claims of the token (iat, nbf and exp) and makes sure the verification is retried after a token becomes valid or after a token expires.



Implementation	Operations per Second	Difference
jsonwebtoken	143,571	–
fast-jwt	284,719	+ 50 %



Implementation	Operation	Algorithm	Operations per Second	Difference
jsonwebtoken	Sign	HS256	63,177	–
fast-jwt	Sign	HS256	92,652	+ 46 %
jsonwebtoken	Sign	RS256	187	–
fast-jwt	Sign	RS256	246	+ 31 %
jsonwebtoken	Verify	HS256	48,663	–
fast-jwt	Verify	HS256	81,452	+ 67 %
jsonwebtoken	Verify	RS256	7,323	–
fast-jwt	Verify	RS256	13,781	+ 88 %

Fast-jet is an experimental library to check if we could improve the performance in JWT verification. As for any experimental features, we are eager to receive feedback. We do not plan to move restify-jwt orfastify-auth0-verify to this module yet as jsonwebtoken is more stable and secure.

CHAPTER 5
CONCLUSION

5.1 CONCLUSION

I am still on the way doing my internship with the Paxcom and I have learned so much from this internship offered by the internship, rally helped me in shaping my personality and equipping me with the knowledge of this technologies. My Final internship project is still remaining with paxcom internship and I will give my best in doing the internship project.

I would like to thank in advance to the manager, colleagues and mentor of Paxcom who guided me through the whole journey of my internship in Paxcom and solved all my doubts during the internship. The manager and mentor 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 solved my all query in the project whether it was related to the reattempt of the assessment, technical issue faced in the assessment or providing extra time to complete the work.

I would highly recommend my juniors to prepare well for the offer in the paxcom and get the internship opportunity from paxcom because paxcom is one of the top fortune companies in the information technology field. I would 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 the management of the placement drive is.

5.2 FUTURE SCOPE

The future scope of the project is to implement both of them in an angular framework and learn angular in a more fast and efficient way.

Also implement session and cookie authentication instead of JSON Web Tokens and explore that field.

References

1) https://developer.mozilla.org/en-US/docs/Web

2) https://www.w3schools.com/xml/dom_nodetype.asp

3) https://www.ijert.org/research/performance-optimization-using-mern-stack-on-web-app lication-IJERTV10IS060239.pdf

Vaibhav Thakur

ORIGINALITY REPORT

23%	15%	3%	22%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Segi University College Student Paper	3%
2	Submitted to Kazakh-British Technical University Student Paper	2%
3	ijarcs.info Internet Source	2%
4	www.ir.juit.ac.in:8080 Internet Source	2%
5	Submitted to Kaplan College Student Paper	2%
6	Submitted to Lebanese University Student Paper	1%
7	Submitted to Istanbul Aydin University Student Paper	1%
8	www.interviewbit.com Internet Source	1%
9	Submitted to Lovely Professional University Student Paper	1%