BACKEND OF AN E-COMMERCE WEP APPLICATION USING MERN STACK

A REPORT ON

MINI PROJECT

Submitted by

HIREN KESHVANI – 180570116025

In partial fulfilment for the award of the degreeof

BACHELOR OF ENGINEERING

in

Information Technology Engineering

Faculty of Engineering

Marwadi Education Foundation, Rajkot

Gujarat Technological University, Ahmedabad

October, 2021

Marwadi Education Foundation, Rajkot

Faculty of Engineering

Information Technology Engineering Department

2021

CERTIFICATE

This is to certify that the **Mini Project** entitled **BACKEND OF AN E-COMMERCE WEB APPLICATION USING MERN STACK** has been carried out by **HIREN KESHVANI (180570116025)** under my guidance in partial fulfilment of the degree of Bachelor of Engineering in Information Technology Engineering (7th Semester) of Gujarat Technological University, Ahmedabad during the academic year 2021-22.

Date : 15/06/2021

Internal Guide

Prof. Pratiti Mankodi
Assistant Professor

Head of the Department

Prof. Krunal Vaghela
Head of Department

Acknowledgements

I would like to express my gratitude towards my parent & member of MEFGI for their kind co-operation and encouragement which help me in completion of this project. I would like to express my special gratitude and thanks to Prof.Pratiti Mankodi for giving me such attention and time. I would also like to thank Hitesh Chaudhary Sir to guide me through project.

This Project is totally developed by me with help of people mentioned above. I am also pleased to report Pratiti Madam that i have been working on frontend portion of this project but as of now it occurred me that react is not a small library that i could finish in few days. I will be completing this project by myself in very near future but work i did in these 14 days is very worth and satisfying.

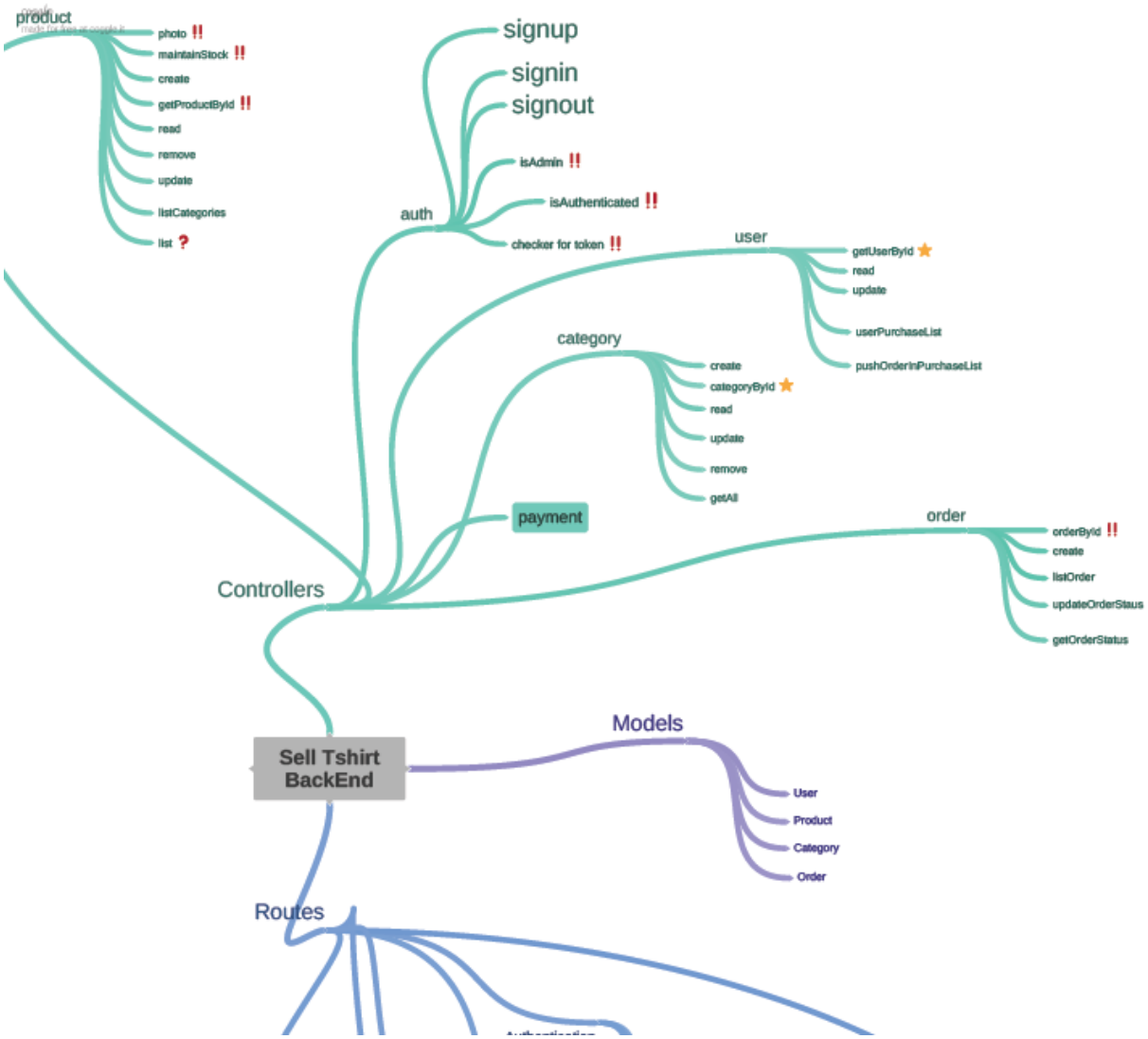
Abstract

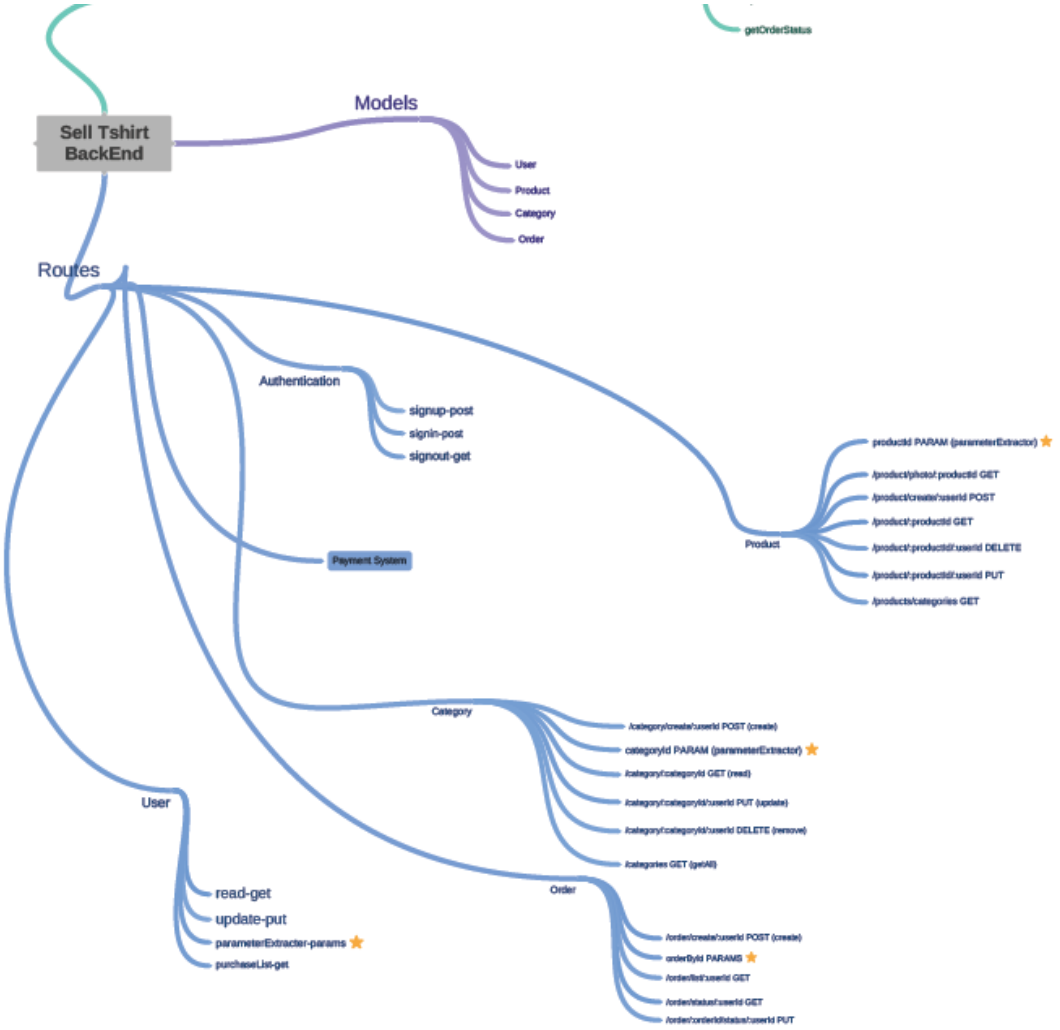
This Project is Backend of a T-Shirt Selling Store. As a son of clothesman i did not think of anything else to sell but Clothes. This Project is an implementation of how E-Commerce projects developed in real life and how behind the scene database, route, authentication and all the backend stuff works.

Selling electronic gadgets, lightning devices and clothing is common nowadays at larger scale but this project is an initiative to bring that facility at local level with minimum affordable price possible. Most of the tools used in this development are open source so there isn't much price concern for customer.

MERN Stack might be more easy to develop application at this scale and also recommended one. Behaviour aspects of javascript has been very fruitful for developers to make the process quick and reliable than other methods. Learning curve is very flexible and one can grasp more things with basic knowledge of JS. This project can satisfy need of any business at required level, whether it's grocery, clothes or sports item.

List of Figures


product
photo
maintainStock
create
getProductById
read
remove
update
listCategories
list
signup
signin
signout
isAdmin
isAuthenticated
checker for token
auth
user
getUserById
read
update
userPurchaseList
pushOrderInPurchaseList
category
create
categoryById
read
update
remove
getAll
payment
order
orderById
create
listOrder
updateOrderStaus
getOrderStatus
Controllers
Sell Tshirt BackEnd
Models
User
Product
Category
Order
Routes


getOrderStatus
Sell Tshirt BackEnd
Models
User
Product
Category
Order
Routes
Authentication
signup-post
signin-post
signout-get
Payment System
Product
productId PARAM (parameterExtractor)
/product/photo/:productId GET
/product/create/:userId POST
/product/:productId GET
/product/:productId/:userId DELETE
/product/:productId/:userId PUT
/products/categories GET
Category
/category/create/:userId POST (create)
categoryId PARAM (parameterExtractor)
/category/:categoryId GET (read)
/category/:categoryId/:userId PUT (update)
/category/:categoryId/:userId DELETE (remove)
/categories GET (getAll)
User
read-get
update-put
parameterExtracter-params
purchaseList-get
Order
/order/create/:userId POST (create)
orderById PARAMS
/order/list/:userId GET
/order/status/:userId GET
/order/:orderId/status/:userId PUT

Contents for Project Report

1. Introduction

1.1 Project Description

This Project is an E-Commerce Web Application developed using MERN Stack. As a final product it is a t-shirt selling store developed for local business. I have developed the backend portion of application as of now. I am working on the frontend of the same and project is fully fledged.

1.2 Project Scope

Scope of this project is to all local to intermediate businesses who want to take their business online with somewhat of investment. This project has no use at large scale as per I coded but we can expand this project as we want to and at scale we want to. But it common scenario in india that most of the businessmen don't bother to make it online but they tie up with flipkart or amazon like platforms to sell product by giving commission.

1.3 Project Requirements

requirements of this projects is mentioned as below

IDE(VS Code)

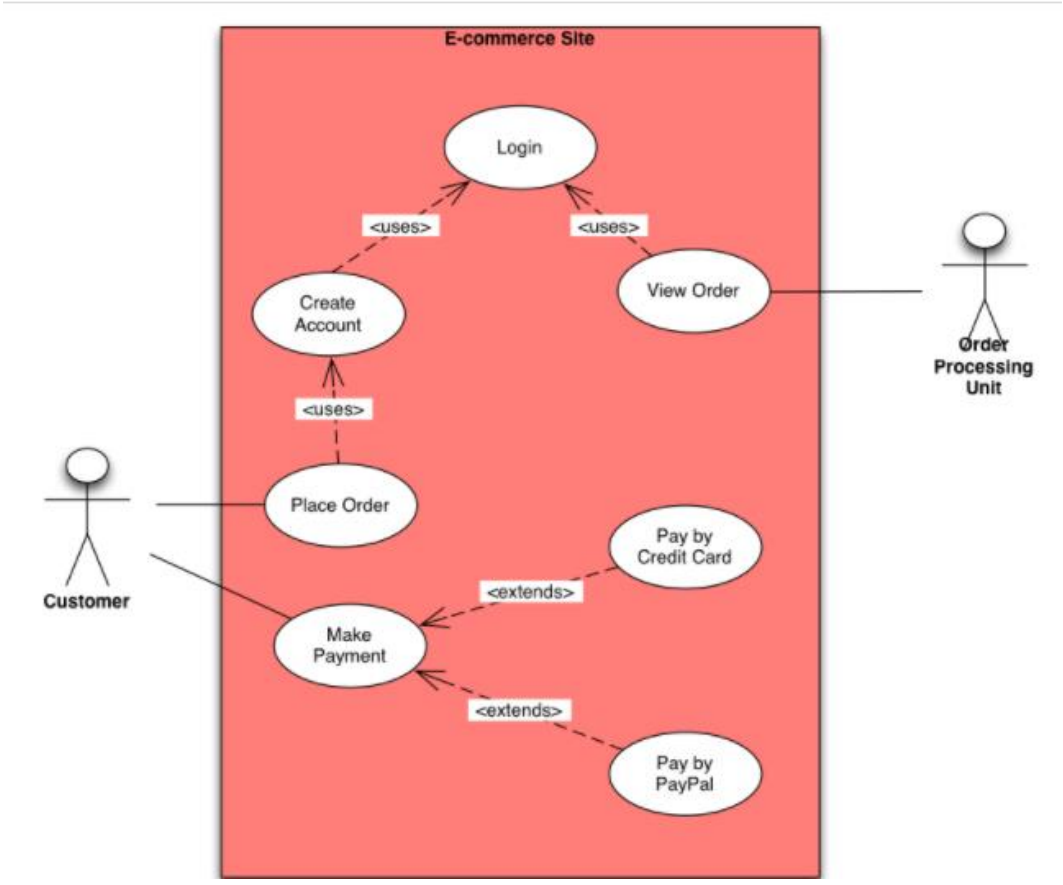
MERN Stack Developer

Bandwidth on any platform(aws/azure/digital ocean)

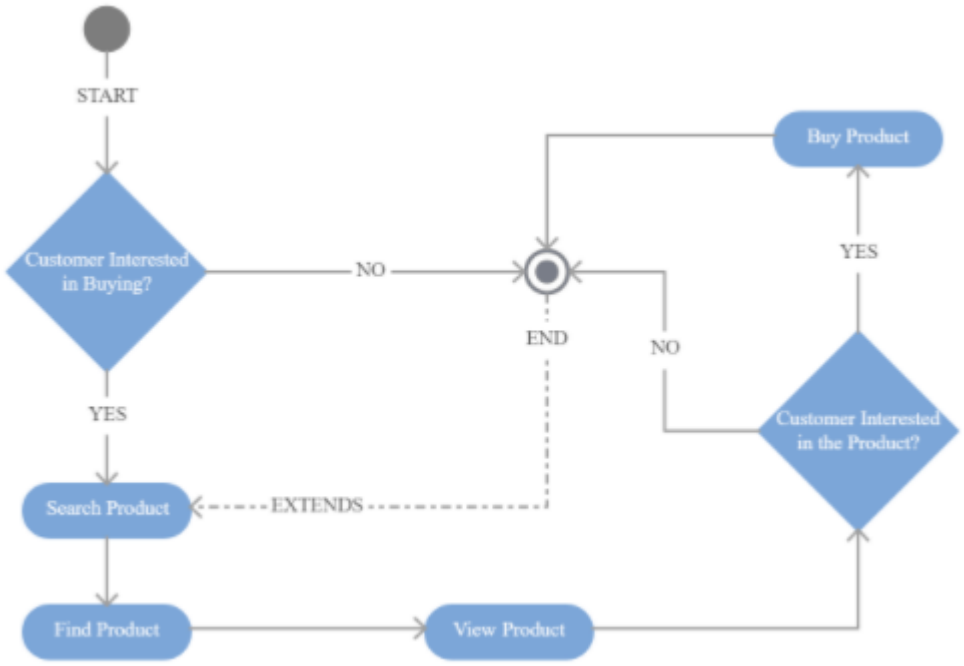
Regular maintainance

2.Project Analysis

2.1 Use-Case Diagram

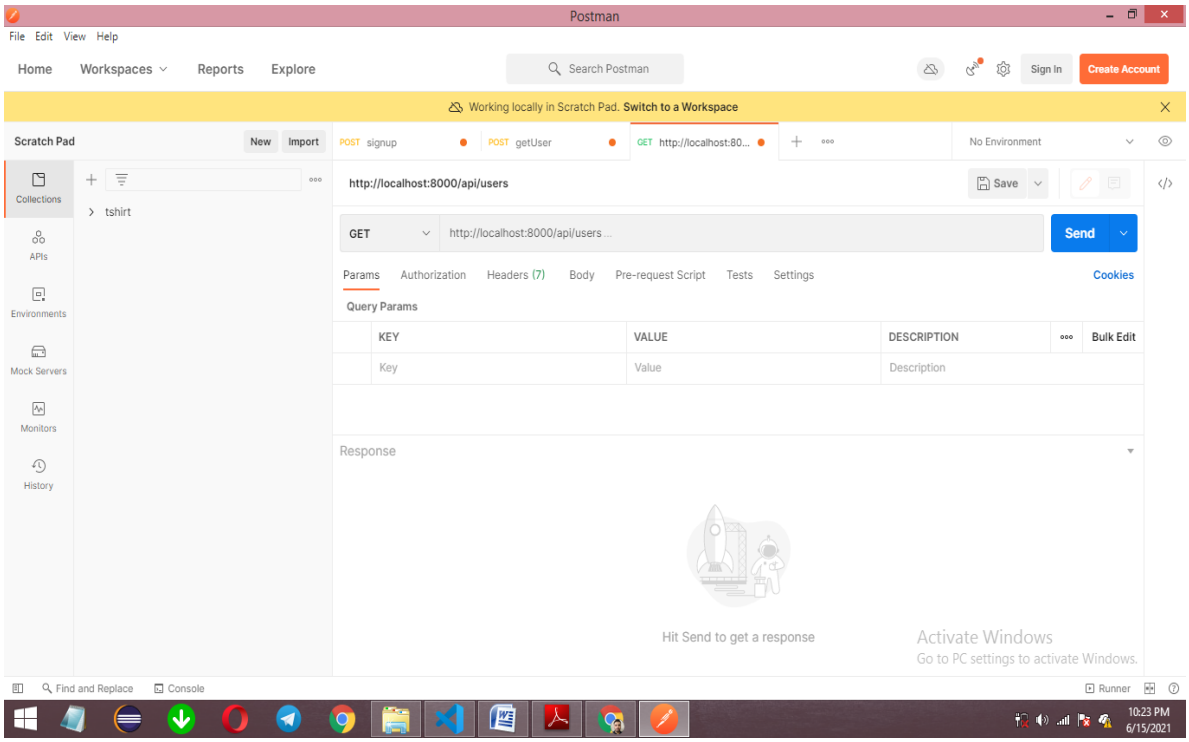


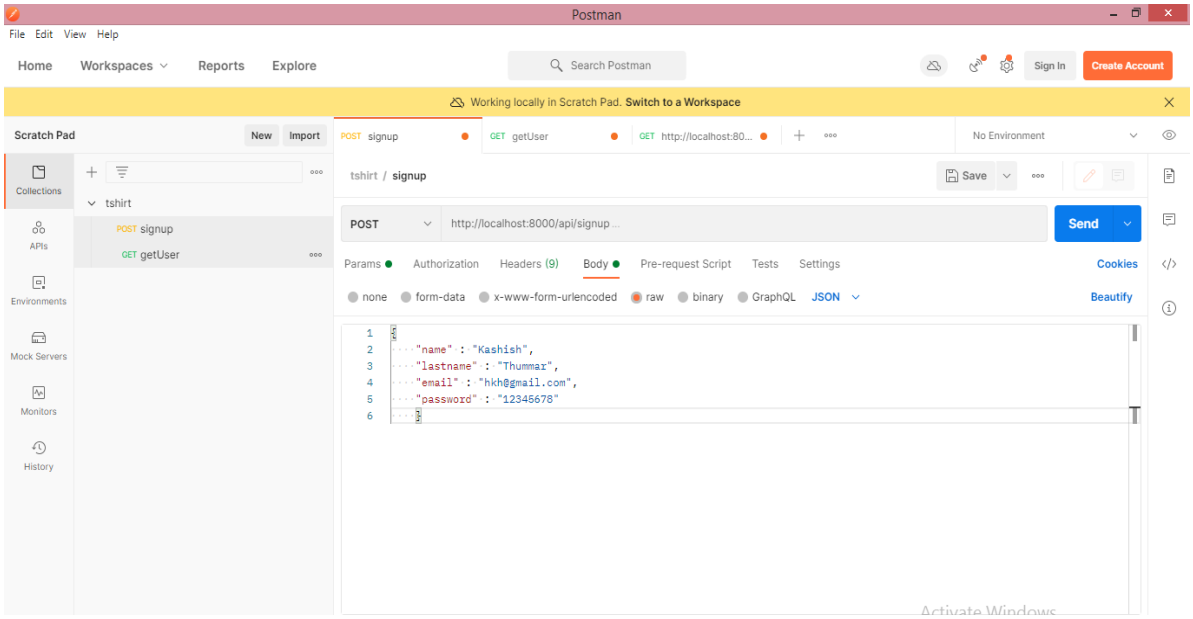
2.1 ACTIVITY DIAGRAM

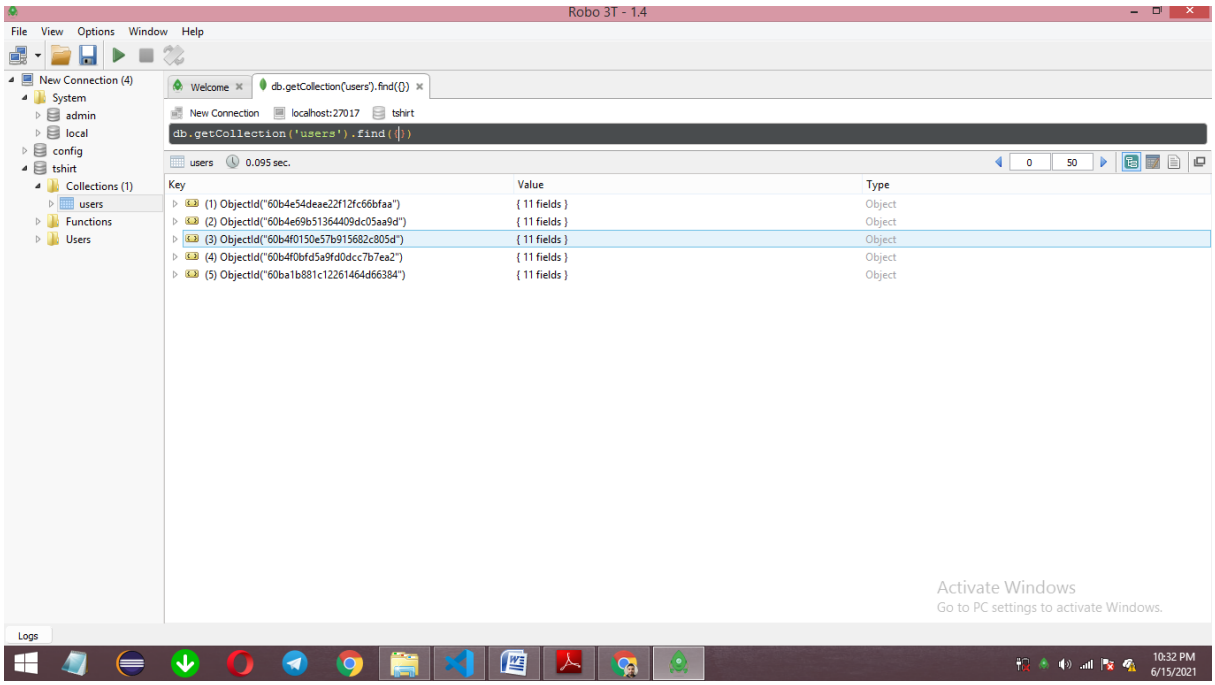


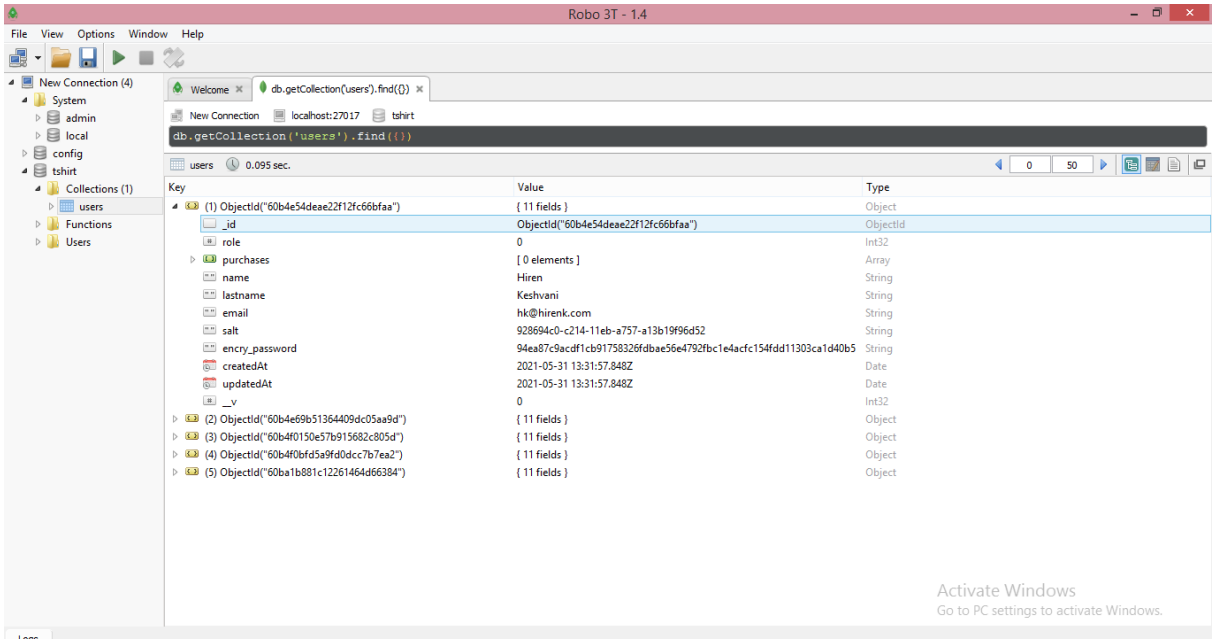
3. Project Implementation

3.1 Project Layouts(GUI Screenshot)






Robo 3T - 1.4
File View Options Window Help
New Connection (4)
System
admin
local
config
tshirt
Collections (1)
users
Functions
Users
Welcome
db.getCollection('users').find({})
New Connection
localhost:27017
tshirt
db.getCollection('users').find({})
users
0.095 sec.
Key
Value
Type
(1) ObjectId("60b4e54deae22f12fc66bfaa")
{ 11 fields }
Object
(2) ObjectId("60b4e69b51364409dc05aa9d")
{ 11 fields }
Object
(3) ObjectId("60b4f0150e57b915682c805d")
{ 11 fields }
Object
(4) ObjectId("60b4f0bfd5a9fd0dcc7b7ea2")
{ 11 fields }
Object
(5) ObjectId("60ba1b881c12261464d66384")
{ 11 fields }
Object
Activate Windows
Go to PC settings to activate Windows.
Logs
10:32 PM
6/15/2021


Robo 3T - 1.4
File View Options Window Help
New Connection (4)
System
admin
local
config
tshirt
Collections (1)
users
Functions
Users
Welcome
db.getCollection('users').find({})
New Connection
localhost:27017
tshirt
db.getCollection('users').find({})
users
0.095 sec.
Key
Value
Type
(1) ObjectId("60b4e54deae22f12fc66bfaa")
{ 11 fields }
Object
_id
ObjectId("60b4e54deae22f12fc66bfaa")
ObjectId
role
0
Int32
purchases
[0 elements]
Array
name
Hiren
String
lastname
Keshvani
String
email
hk@hirenk.com
String
salt
928694c0-c214-11eb-a757-a13b19f96d52
String
encry_password
94ea87c9acdf1cb91758326fdbae56e4792fbc1e4acfc154fdd11303ca1d40b5
String
createdAt
2021-05-31 13:31:57.848Z
Date
updatedAt
2021-05-31 13:31:57.848Z
Date
__v
0
Int32
(2) ObjectId("60b4e69b51364409dc05aa9d")
{ 11 fields }
Object
(3) ObjectId("60b4f0150e57b915682c805d")
{ 11 fields }
Object
(4) ObjectId("60b4f0bfd5a9fd0dcc7b7ea2")
{ 11 fields }
Object
(5) ObjectId("60ba1b881c12261464d66384")
{ 11 fields }
Object
Activate Windows
Go to PC settings to activate Windows.

3.2 Project Code

PACKAGE.JSON

```
{
  "name": "projbackend",
  "version": "1.0.0",
  "description": "",
  "main": "app.js",
  "scripts": {
    "start": "nodemon app.js"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "body-parser": "^1.19.0",
    "cookie-parser": "^1.4.4",
    "cors": "^2.8.5",
    "dotenv": "^8.2.0",
    "express": "^4.17.1",
    "express-jwt": "^5.3.1",
    "express-validator": "^6.2.0",
    "formidable": "^1.2.1",
    "i": "^0.3.6",
    "jsonwebtoken": "^8.5.1",
    "lodash": "^4.17.15",
    "mongoose": "^5.7.7",
    "morgan": "^1.9.1",
    "nodemon": "^1.19.4",
```

```
    "npm": "^6.13.0",
    "uuid": "^3.3.3"
  }
}
```

APP.JS

```
require("dotenv").config();

const mongoose = require("mongoose");
const express = require("express");
const app = express();
const bodyParser = require("body-parser");
const cookieParser = require("cookie-parser");
const cors = require("cors");

//My Routes
const authRoutes = require("./routes/auth");
const userRoutes = require("./routes/user");

//DB Connection
mongoose
  .connect(process.env.DATABASE, {
    useNewUrlParser: true,
    useUnifiedTopology: true,
    useCreateIndex: true
  })
```

```
  .then(() => {
    console.log("DB CONNECTED");
  });

  //Middlewares
  app.use(bodyParser.json());
  app.use(cookieParser());
  app.use(cors());

  //My Routes
app.use("/api", authRoutes);
app.use("/api", userRoutes);



//PORT
const port = process.env.PORT || 8000;

//Starting a Server
app.listen(port, () => {
  console.log(`app is running at ${port}`);
});
```

CONTROLLER FILES

AUTH.JS

```
const User = require("../models/user");
```

```
const { check, validationResult } = require('express-
validator');
var jwt = require('jsonwebtoken');
var expressJwt = require('express-jwt');

exports.signup = (req,res) => {

    const errors = validationResult(req);

    if(!errors.isEmpty()){
        return res.status(422).json({
         error : errors.array()[0].msg

        });
 }


    const user = new User(req.body)
    user.save((err,user) =>{
        if(err){
            return res.status(400).json({
             err : "NOT ABLE TO SAVE USER IN THE DATA
BASE"
            });
        }
        res.json({
            name : user.name,
            email : user.email,
            id : user._id
```

```
        });
    });
};

exports.signin = (req,res) => {
    const errors = validationResult(req);

    const{email,password} = req.body;

    if(!errors.isEmpty()){
        return res.status(422).json({
         error : errors.array()[0].msg

        });
 }

 User.findOne({email}, (err, user) =>{
     if(err || !user){
        return  res.status(400).json({
             error : "user does not exist in database
"
         })
     }

     if(!user.authenticate(password)){
         return res.status(401).json({
             error : "email and password do not match
"
```

```
        })

    }
    //token created
    const token = jwt.sign({_id: user._id}, process.
env.SECRET);
    //put token in cookie
    res.cookie("token", token, {expire: new Date() +
9999});

    //send response to front end
    const {_id, name, email, role} = user;
    return res.json({token, user: {_id, name, email,
 role}});
 });

};

exports.signout = (req,res) => {
    res.clearCookie("token");
    res.json({
        message : "User Signed Out Successfully"
    });
};

//protected routes
exports.isSignedIn = expressJwt({
    secret : process.env.SECRET,
```

```
    userProperty : "auth"
});


//custom middlewares

exports.isAuthenticated = (req, res, next) => {
let checker = req.profile && req.auth && req.profile.
_id == req.auth._id;

if(!checker){
 return res.status(403).json({
     error : "ACCESS DENIED"
 });
}
    next();
}

exports.isAdmin = (req, res, next) => {
    if(req.profile.role === 0) {
        return res.status(403).json({
            error : "You are not an Admin, Access Den
ied"
        });
    }


    next();
}
```

USER.JS

```
const User = require("../models/user");

exports.getUserById = (req, res, next, id) => {
    User.findById(id).exec((err, user) => {
        if(err || !user){
        return res.status(400).json({
            error: "No User Was Found IN Database"
        });
        }
        req.profile = user;
        next();
    });
};

exports.getUser = (req,res) =>{
    req.profile.salt = undefined;
    req.profile.encry_password = undefined;
    req.profile.createdAt = undefined;
    req.profile.updatedAt = undefined;

    return res.json(req.profile)
}

exports.updateUser = (req, res) => {
    User.findByIdAndUpdate(
        {_id : req.profile._id},
        {$set : req.body},
```

```
        {new : true, useFindAndModify : false},
        (err, user) => {
          if(err){
              return res.status(400).json({
                  error : "you are not authorized to
update this information "
              });
          }
          user.salt = undefined;
          user.encry_password = undefined;
          res.json(user)
        }
    );

};
```

ROUTES FILE

AUTH.JS

```
var express = require("express");
var router = express.Router();
const { check } = require('express-validator');


const {signout,signup,signin, isSignedIn} = require("
../controllers/auth");


router.post(
```

```
    "/signup",
    [
    check("name","name should be atleast 3 character"
).isLength({ min: 3 }),
    check("email","email is required").isEmail(),
    check("password","password should be atleast 8 ch
aracters").isLength({ min: 8 })
    ],
signup
);

router.post(
    "/signin",
    [
    check("email","email is required").isEmail(),
    check("password","password field is required").is
Length({ min: 1 })
    ],
signin
);


router.get("/signout",signout);



module.exports = router;
```

USER.JS

```
const express = require("express");
const router = express.Router();

const { getUserById, getUser, updateUser } = require(
"../controllers/user");
const { isSignedIn, isAuthenticated, isAdmin} = requi
re("../controllers/auth");

router.param("userId", getUserById);

router.get("/user/:userId",isSignedIn, isAuthenticate
d,  getUser);
router.put("/user/:userId", isSignedIn, isAuthenticat
ed, updateUser);


module.exports = router;
```

DOTENV FILE

DATABASE = mongodb://localhost:27017/tshirt
SECRET = andresdefonollosa

MODEL FILES
CATEGORY.JS

```
const mongoose = require("mongoose");

const categorySchema = new mongoose.Schema({
   name:{
```

```
        type : String,
        trim : true,
        required : true,
        maxlength : true,
        unique : true
}
},
{timestamps : true}
);

module.exports = mongoose.model("Category",categorySc
hema);
```

ORDER.JS

```
const mongoose = require("mongoose");
const {ObjectId} = mongoose.Schema;

const ProductCartSchema = new mongoose.Schema({
    product : {
        type : ObjectId,
        ref : "Product"
    },

    name : String,
    count : Number,
    price : Number
});
```

```
const ProductCart = mongoose.model("ProductCart",Prod
uctCartSchema)

const  OrderSchema = new mongoose.Schema(
    {
        products : [ProductCartSchema],
        transaction_id : {},
        amount : { type : Number},
        address : String,
        updated : Date,
        user : {
            type : ObjectId,
            ref : "User"
        }
    },
    {timestamps : true}
);

const Order = mongoose.model("Order",OrderSchema)

module.exports = {Order,ProductCart};
```

PRODUCT.JS

```
const mongoose = require("mongoose");
const {ObjectId} = mongoose.Schema;

const productSchema = new mongoose.Schema({

    name : {
```

```
        type : String,
        trim : true,
        required : true,
        maxlength : 32
    },

    description :{
        type : String,
        trim : true,
        required : true,
        maxlength : 2000
    },

    price : {
        type : Number,
        required : true,
        maxlength : 32,
        trim : true
    },

    category : {
         type : ObjectId,
         ref : "Category",
         required : true
    },

    stock : {
        type : Number
```

```
    },

    sold : {
        type : Number,
        default : 0
    },

    photo : {
        data : Buffer,//there are many other ways to
store this such as s3 bucket of aws,firebase
        contentType : String
    }
},
{timestamps : true}
);

module.exports = mongoose.model("Product",productSche
ma);
```

USER.JS

```
var mongoose = require("mongoose");
const crypto = require("crypto");
const uuidv1 = require("uuid/v1");

var userSchema = new mongoose.Schema({
  name: {
    type: String,
    required: true,
    maxlength: 32,
```

```
    trim: true
  },
  lastname: {
    type: String,
    maxlength: 32,
    trim: true
  },
  email: {
    type: String,
    trim: true,
    required: true,
    unique: true
  },
  userinfo: {
    type: String,
    trim: true
  },
  encry_password: {
    type: String,
    required: true
  },
  salt: String,
  role: {
    type: Number,
    default: 0
  },
  purchases: {
    type: Array,
```

```
    default: []
  }
},
{timestamps : true}
);

userSchema
  .virtual("password")
  .set(function(password) {
    this._password = password;
    this.salt = uuidv1();
    this.encry_password = this.securePassword(passwor
d);
  })
  .get(function() {
    return this._password;
  });

userSchema.methods = {
  authenticate: function(plainpassword) {
    return this.securePassword(plainpassword) === thi
s.encry_password;
  },

  securePassword: function(plainpassword) {
    if (!plainpassword) return "";
    try {
      return crypto
```

```
        .createHmac("sha256", this.salt)
        .update(plainpassword)
        .digest("hex");
    } catch (err) {
      return "";
    }
  }
};

module.exports = mongoose.model("User", userSchema);
```

4 SUMMARY

4.1 CONCLUSION

At the end it's fair to say that project nowadays like E-Commerce will be easy to create by modern cutting edge technologies and javascript frameworks like react,express, node backend run-time environment. Also, now people will save their commission money they gave to big platforms.Cost of this project is not as other methods or stacks

4.2 FUTURE SCOPE

This Project will get more market share as internet and websites crash starts in india in near future. E-Commerce will get more common in next decade and may be with the stack i have used here and also many methodologies.

Instructions for creating the report

Line Spacing: 1.5

Printing Margin: 3 cm Left Margin

2.5 cm all Side Margin (Top + Bottom + Right)

Font: Times New Roman **only**.

Font size:

(Text should start from next line after Title.)

MAIN TITLE: 16 BOLD (Alignment: Left) (Title Case)

SUB TITLE: 14 BOLD (Alignment: Left) (Title Case)

MATTER: 12 Normal (Alignment: Justify)

All paragraphs must start **without** 'tab'.

Two line spacing between paragraphs.

Start new Chapter from new page.

No blank area at the end of each page except last page of chapter.

DO NOT CHANGE THE SEQUENCE OF THE REPORT.

- Cover Page & Title Page
- Certificate from College (Individual)
- Acknowledgements
- Abstract
- Abstract
- List of Tables
- List of Figures
- Table of Content
- Chapters