



L OVELY
P ROFESSIONAL
U NIVERSITY

Name :Rakesh Kumar Singh

Reg No:11706551

Roll no:10

Section:KM012

Topic: Article Management System

Project Name :Article Views

Submitted To:

Faculty: Neha Sharma

Website Name

ARTICLE VIEWS

Technology Used

Back End:Node Js,Express Js

Front End:Ejs view Engine

Data Base:MongoDB

Github Link: <https://github.com/rakeshkumar1019/Article-Views>

Coggle.it Link:

<https://coggle.it/diagram/X6ArlfOtsTb1CGj2/t/article-views>

Logical Diagram / Structure view

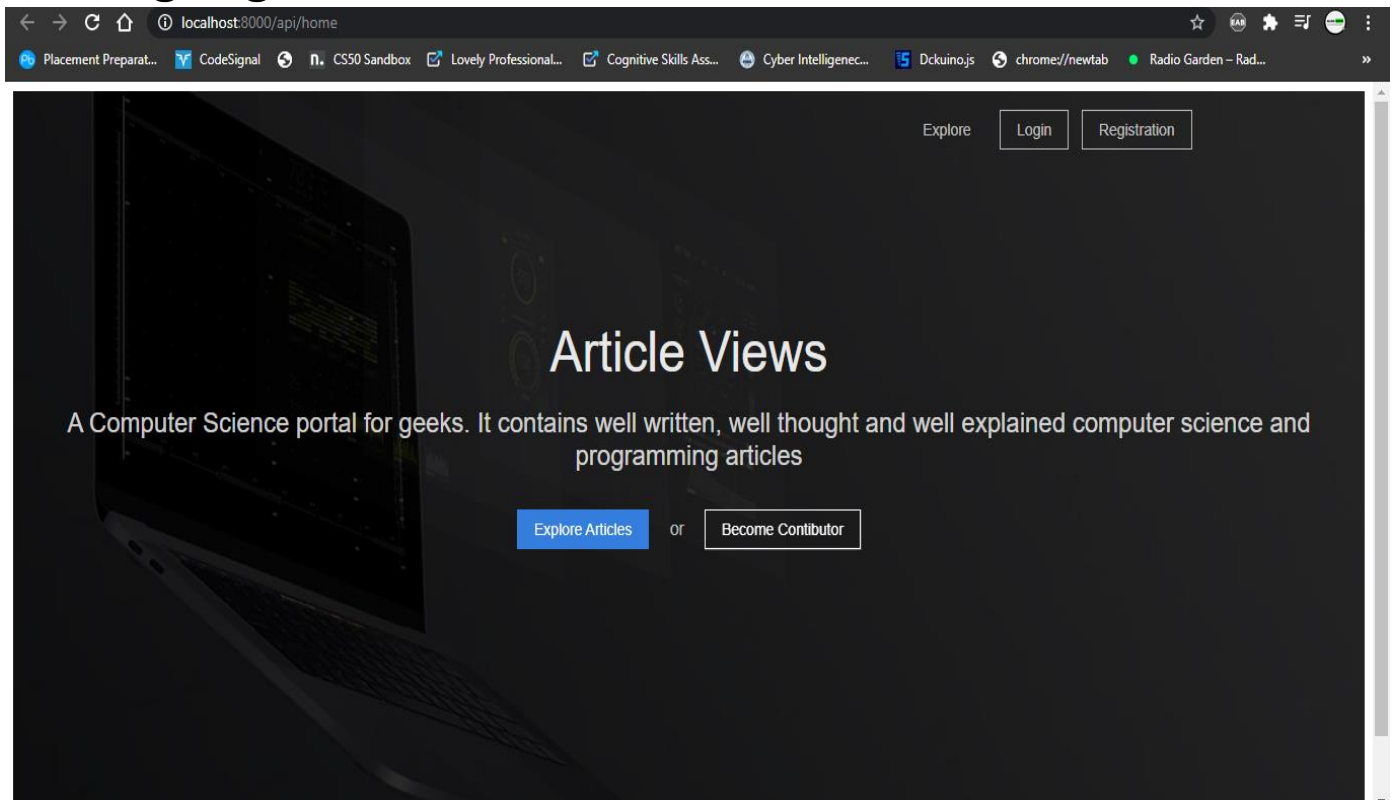


Packa

```
File Edit Selection View Go Run Terminal Help
package.json X
package.json > {} dependencies
5 "main": "app.js",
6 "scripts": {
7   "start": "nodemon app.js"
8 },
9 "keywords": [],
10 "author": "Rakesh Kumar Singh",
11 "license": "ISC",
12 "dependencies": {
13   "body-parser": "^1.19.0",
14   "cookie-parser": "^1.4.4",
15   "cors": "^2.8.5",
16   "dotenv": "^8.2.0",
17   "ejs": "^3.1.5",
18   "express": "^4.17.1",
19   "express-jwt": "^5.3.1",
20   "express-validator": "^6.2.0",
21   "jsonwebtoken": "^8.5.1",
22   "mongoose": "^5.7.7",
23   "morgan": "^1.9.1",
24   "multer": "^1.4.2",
25   "nodemon": "^1.19.4",
26   "npm": "^7.0.7",
27   "uuid": "^3.3.3"
28 }
29
30
```

ge.json

Landing Page



Registration page & Login for Contributor

Registration form

Name

Rakesh Kumar ✓

Last Name

Singh ✓

Email address

srakeshkumar1019@gmail.com ✓

Password

***** ✓

☒ Male ☐ Female

Phone

9182700412 ✓

Login Form

Login form

Email address

srakeshkumar1019@gmail.com ✓

Password

***** ✓

Contibutor Dashboard


← → ↻ 🏠 ⓘ localhost:8000/api/dashboard


🌐 Placement Preparat... 📄 CodeSignal 🌐 CS50 Sandbox 📄 Lovely Professional... 📄 Cognitive Skills Ass... 🌐 Cyber Intelligenec... 📄 Dckuino.js 🌐 chrome://newtab 📄 Radio Garden – Rad...

Article Views Dashboard Profile Create New Articles

Logout

Wellcome To Dashboard





MY Details
Name:Rakesh Kumar Singh
Email:srakeshkumar1019@gmail.com
Gender:male
Phone:9182700412

ABOUT ME
Hello, I am a Full Stack/MERN Stack Developer and confident professional with passion for technology. Has hands on experience in application development.

Resuma

linkedin

SKILLS
MERN /Full Stack Developer
Data Structure & Algorithms
Mysql & MongoDB
Operting System
Networking
C++/C
HTML/CSS


Github

Article Views
A Computer Science portal for geeks. It contains well written, well thought and well explained computer science and programming articles
srakeshkumar1019@gmail.com

Profile Page

Article Views Dashboard Profile Create New Articles

Logout



UPDATE YOUR PROFILE

First Name

Rakesh Kumar

✓

Last Name

Singh

✓

Email

srakeshkumar1019@gmail.com

✓

Gender

☒ Male
☐ Female

Phone

9182700412

✓

Textarea

Hello, I am a Full Stack/MERN Stack Developer and confident professional with passion for technology. Has hands on experience in application development.

✓

Choose Profile Image

Choose File

No file chosen

Update

Article Views
A Computer Science portal for geeks. It contains well written, well thought and well explained computer science and programming articles
srakeshkumar1019@gmail.com

Create New

[Article Views](#) [Dashboard](#) [Profile](#) [Create New](#) [Articles](#) [Logout](#)

Title

Preview

Content

Code

Embed Youtube By URL

Tag One

Tag Two

Tag Three

Choose Article Image


No file chosen

Article Views

A Computer Science portal for geeks. It contains well written, well thought and well explained computer science and programming articles
srakeshkumar1019@gmail.com

List of Articles for Contributors

[Article Views](#) [Profile](#) [Create New](#) [Articles](#) [Logout](#)



ReactJS


An introduction to learn React's Why, What, and How

The Complete Introduction to React Learn about React components with functions and classes. Using JSX. Benefits of components. React Hooks. User events. Taking input from users... jscomplete.com

Last Update
Mon, 02 Nov 2020 07:04:55 GMT

[Read More...](#)

Express.js Tutorial




Learn Express.js Fundamentals With Hands-On

Express.js is a fast and lightweight framework used majorly for web application development and the Node.js Developers all over the world are totally in love with this framework. jscomplete.com

Last Update
Mon, 02 Nov 2020 07:08:13 GMT

[Read More...](#)



An Overview of MongoDB & Mongoose

A Little Historical Context Databases fall into several different categories. The earliest databases, up through the mid-1980's, were so-called CODASYL databases. These organized data into records and related occurrences of different record types to one another using hashes to form a network.

Last Update
Mon, 02 Nov 2020 14:21:22 GMT

[Read More...](#)

A complete guide to Redux

Redux Toolkit included

localhost:8000/api/dashboard/article/delete/5f9fb05d560c130c383d2e13

Each Article View

Article Views Articles

AN INTRODUCTION TO LEARN REACT'S WHY, WHAT, AND HOW



React is defined as a JavaScript library for building user interfaces. Let's start by talking about the two different parts of this definition. React is a JavaScript "library". It is not exactly a "framework". It is not a complete solution and you will often need to use more libraries with React to form any solution. React does not assume anything about the other parts in any solution. Frameworks serve a great purpose, especially for young teams and startups. When working with a framework, many smart design decisions are already made for you, which gives you a clear path to focus on writing good application-level logic. However, frameworks come with some disadvantages. For experienced developers working on large codebases, these disadvantages are sometimes a deal breaker. Frameworks are not flexible, although some claim to be. A framework usually wants you to code everything a certain way. If you try to deviate from that way, the framework usually ends up fighting you about it. Frameworks are also usually large and full of features. If you need to use only a small piece of them, you have to include the whole thing anyway. Admittedly, this point is changing today but it is still not ideal. Some frameworks are going modular, which I think is great, but I am a big fan of the pure Unix philosophy:

```
ReactDOM.render(  
  React.createElement(  
    'div',  
    null,  
    'Hello React ',  
    React.createElement('input'),  
    React.createElement(  
      'pre',  
      null,  
      new Date().toLocaleTimeString()  
    )  
  )  
)
```

REACT JS

NODE JS

EXPRESS JS

ABOUT AUTHOR



Name :Rakesh Kumar Singh
Email:srakeshkumar1019@gmail.com
Phone:9182700412

Hello, I am a Full Stack/MERN Stack Developer and confident professional with passion for technology. Has hands on experience in application development.

Article Views

Edit /Update Aritcle Page by Contibutor

Article Views
Dashboard
Profile
Create New
Articles
Logout

Update Article

Title

An introduction to learn React's Why, What, and How

Preview

The Complete Introduction to React
Learn about React components with functions and classes. Using JSX. Benefits of

Content

React is defined as a JavaScript library for building user interfaces. Let's start by talking about the two different parts of this definition.

Code

ReactDOM.render(
React.createElement(

Embed Youtube By URL

https://www.youtube.com/embed/Ke90Tje

Tag One

React Js

Tag Two

Node Js

Tag Three

Express Js

Choose Article Image

Choose File No file chosen


Update

Preview

Article Views
A Computer Science portal for geeks. It contains well written, well thought and well explained computer science and programming articles
srakeshkumar1019@gmail.com

EDIT & DELETE ARTICLE BY CONTIBUTOR

Article Views
Profile
Create New
Articles
Logout



ReactJS

An introduction to learn React's Why, What, and How

The Complete Introduction to React Learn about React components with functions and classes. Using JSX. Benefits of components. React Hooks. User events. Taking input from users... jscomplete.com

Last Update


Mon, 02 Nov 2020 07:04:55 GMT

Read More...

Edit

Delete

Express.js Tutorial



Learn Express.js Fundamentals With Hands-On

Express.js is a fast and lightweight framework used majorly for web application development and the Node.js Developers all over the world are totally in love with this framework. jscomplete.com


Last Update

Mon, 02 Nov 2020 07:08:13 GMT

Read More...

Edit

Delete



An Overview of MongoDB & Mongoose

A Little Historical Context Databases fall into several different categories. The earliest databases, up through the mid-1980's, were so-called CODASYL databases. These organized data into records and related occurrences of different record types to one another using hashes to form a network.

Last Update

Mon, 02 Nov 2020 14:21:22 GMT

Read More...

Edit

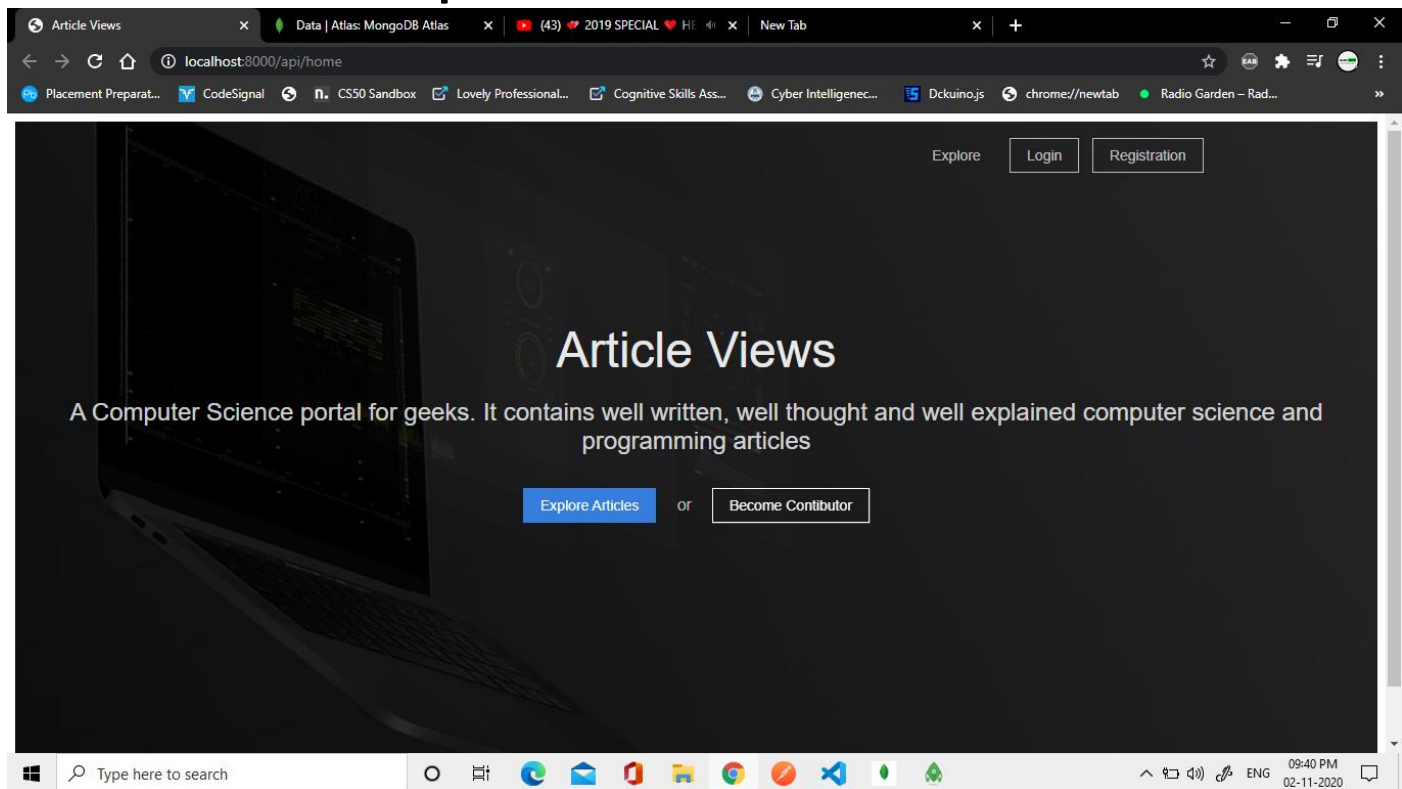
Delete

A complete guide to Redux

Redux Toolkit included

localhost:8000/api/dashboard/article/delete/5f9fb05d560c130c383d2e13

For Normal User-Explore



List of Articles for normal user



ReactJS

An introduction to learn React's Why, What, and How

The Complete Introduction to React Learn about React components with functions and classes. Using JSX. Benefits of components. React Hooks. User events. Taking input from users... jscomplete.com

Last Update

Mon, 02 Nov 2020 07:04:55 GMT

[Read More...](#)

Express.js Tutorial



Learn Express.js Fundamentals With Hands-On

Express.js is a fast and lightweight framework used majorly for web application development and the Node.js Developers all over the world are totally in love with this framework. jscomplete.com

Last Update

Mon, 02 Nov 2020 07:08:13 GMT

[Read More...](#)



An Overview of MongoDB & Mongoose

A Little Historical Context Databases fall into several different categories. The earliest databases, up through the mid-1980's, were so-called CODASYL databases. These organized data into records and related occurrences of different record types to one another using hashes to form a network.

Last Update

Mon, 02 Nov 2020 14:21:22 GMT

[Read More...](#)

A complete guide to Redux

Redux Toolkit included

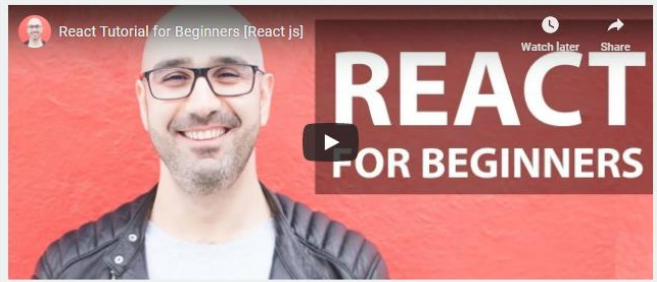


Inside Article

AN INTRODUCTION TO LEARN REACT'S WHY, WHAT, AND HOW



ReactJS



React is defined as a JavaScript library for building user interfaces. Let's start by talking about the two different parts of this definition. React is a JavaScript "library". It is not exactly a "framework". It is not a complete solution and you will often need to use more libraries with React to form any solution. React does not assume anything about the other parts in any solution. Frameworks serve a great purpose, especially for young teams and startups. When working with a framework, many smart design decisions are already made for you, which gives you a clear path to focus on writing good application-level logic. However, frameworks come with some disadvantages. For experienced developers working on large codebases, these disadvantages are sometimes a deal breaker. Frameworks are not flexible, although some claim to be. A framework usually wants you to code everything a certain way. If you try to deviate from that way, the framework usually ends up fighting you about it. Frameworks are also usually large and full of features. If you need to use only a small piece of them, you have to include the whole thing anyway. Admittedly, this point is changing today but it is still not ideal. Some frameworks are going modular, which I think is great, but I am a big fan of the pure Unix philosophy:

form any solution. React does not assume anything about the other parts in any solution. Frameworks serve a great purpose, especially for young teams and startups. When working with a framework, many smart design decisions are already made for you, which gives you a clear path to focus on writing good application-level logic. However, frameworks come with some disadvantages. For experienced developers working on large codebases, these disadvantages are sometimes a deal breaker. Frameworks are not flexible, although some claim to be. A framework usually wants you to code everything a certain way. If you try to deviate from that way, the framework usually ends up fighting you about it. Frameworks are also usually large and full of features. If you need to use only a small piece of them, you have to include the whole thing anyway. Admittedly, this point is changing today but it is still not ideal. Some frameworks are going modular, which I think is great, but I am a big fan of the pure Unix philosophy:

```
ReactDOM.render(
  React.createElement(
    'div',
    null,
    'Hello React ',
    React.createElement('input'),
    React.createElement(
      'pre',
      null,
      new Date().toLocaleTimeString()
    )
  )
)
```

[REACT JS](#)
[NODE JS](#)
[EXPRESS JS](#)

ABOUT AUTHOR



Name :Rakesh Kumar Singh
Email:srakeshkumar1019@gmail.com
Phone:9182700412

Hello, I am a Full Stack/MERN Stack Developer and confident professional with passion for technology. Has hands on experience in application development.

Article Views

A Computer Science portal for geeks. It contains well written, well thought and well explained computer science and programming articles
srakeshkumar1019@gmail.com

Mongo DB Database Users collection

Algo Repo's Org - 20...

Access Manager

Support

Billing

Project 0

Atlas

Realm

Charts

All Clusters

Algo Repo

Network Access

Advanced

+ Create Database

NAMESPACES

views

articles

users

views.users

COLLECTION SIZE: 9998 TOTAL DOCUMENTS: 2 INDEXES TOTAL SIZE: 72KB

Find Indexes Schema Anti-Patterns 0 Aggregation Search Indexes

INSERT DOCUMENT

FILTER {"filter": "example"}

Find Re

QUERY RESULTS 1-2 OF 2

```

_id: ObjectId("5f9e86b10423e629006a147f")
userinfo: "Hello, I am a Full Stack/MERN Stack Developer and confident profession..."
image: "/1604233179211ppp.jpg"
firstname: "Rakesh Kumar"
lastname: "Singh"
email: "srakeshkumar1019@gmail.com"
salt: "bee76f40-1c28-11eb-b65b-27a97e2c5e68"
encry_password: "1a66499edc20bf5ce3d325f4314e4209853828e932844a3b35df04f87ca13fe"
gender: "male"
phone: 9182700412
createdAt: 2020-11-01T09:58:09.214+00:00
updatedAt: 2020-11-01T12:19:39.220+00:00
__v: 0

```

```

_id: ObjectId("5fa014efd7029b2388e5f22a")
userinfo: "Part time contributor at Article Views"
image: "https://image.shutterstock.com/image-vector/contributor-vector-icon-de..."
firstname: "Hande"
lastname: "Ercel"
email: "handeercel@gmail.com"
salt: "1dc15e70-1d16-11eb-880f-392d6d5ce622"

```

Articles Collection

Algo Repo's Org - 20...

Access Manager

Support

Billing

Project 0

Atlas

Realm

Charts

All Clusters

Algo Repo

Database Access

Network Access

Advanced

+ Create Database

NAMESPACES

views

articles

users

views.articles

COLLECTION SIZE: 10.57KB TOTAL DOCUMENTS: 3 INDEXES TOTAL SIZE: 36KB

Find Indexes Schema Anti-Patterns 0 Aggregation Search Indexes

INSERT DOCUMENT

FILTER {"filter": "example"}

Find Re

QUERY RESULTS 1-4 OF 4

```

_id: ObjectId("5f9faf97560c130c383d2e12")
like: Array
author: ObjectId("5f9e86b10423e629006a147f")
title: "An introduction to learn React's Why, What, and How"
preview: "The complete introduction to React"
content: "Learn about React components with ..."
content: "React is defined as a JavaScript library for building user interfaces...."
ReactDOM.render(
  React.createElement(
    code: 'div',
    null,
    '...'
  )
)
video_URL: "https://www.youtube.com/embed/Ke90Tje7V50"
tag_one: "React Js"
tag_two: "Node Js"
tag_three: "Express Js"
article_image: "/16043006954551_dLaDL-15N0iprzMOpM7zQ.png"
createdAt: 2020-11-02T07:04:55.478+00:00
updatedAt: 2020-11-02T07:04:55.478+00:00
__v: 0

```

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")
const ejs = require("ejs")

//my routes
const authRoutes = require("./routes/auth")
const dashboardRoutes = require("./routes/dashboard")
const profileRoutes = require("./routes/profile")
const articleRoutes = require("./routes/article")

//DBCONNECTION
mongoose.connect(process.env.DATABASE,
  {
    useUnifiedTopology: true,
    useNewUrlParser: true,
    useCreateIndex: true
  }).then(() => {
    console.log("DB CONNECTED")
  })
// Middleware
app.use(bodyParser.json())
app.use(bodyParser.urlencoded({ extended: false }))
app.use(cookieParser())
app.use(cors())
app.use(express.static('uploads'))
app.set('view engine', 'ejs');

//my routes
app.use("/api", authRoutes)
app.use("/api", dashboardRoutes)
app.use("/api", profileRoutes)
app.use("/api", articleRoutes)
//PORT
const port = process.env.PORT || 8000
app.listen(port, () => {
  console.log(`app is running at ${port}`)
```

```
})
```

Model>article.js

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

let articleSchema = new mongoose.Schema({
  title: {
    type: String,
  },
  preview: {
    type: String,
  },
  content: {
    type: String,
  },
  video_URL: {
    type: String,
    trim: true,
  },
  article_image: {
    type: String,
    trim: true,
  },
  author: {
    type: ObjectId,
    ref: 'User'
  },
  code: {
    type: String,
  },
  tag_one: {
    type: String,
    trim: true,
  },
  tag_two: {
    type: String,
    trim: true,
  },
  tag_three: {
    type: String,
```

```

        trim: true,
    },
},
{ timestamps: true }
)

module.exports = mongoose.model("Article", articleSchema)

```

Model>user Scehma

```

const mongoose = require("mongoose")
const crypto = require("crypto")
const uuidv1 = require('uuid/v1')
var userSchema = new mongoose.Schema({
  firstname: {
    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,
    default: "Part time contributor at Article Views "
  },
  encry_password: {
    type: String,
    required: true,
  },

```



```

    phone: {
      type: Number,
      required: true,
      maxlength: 10,
    },
    gender: {
      type: String,
      required: true,
    },
    image: {
      type: String,
      default: "https://image.shutterstock.com/image-
vector/contributor-vector-icon-design-transparent-260nw-406989082.jpg"
    },
    salt: String,
  },
  { timestamps: true }
)

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

userSchema.methods = {
  authenticate: function (plainpassword) {
    return this.securePassword(plainpassword) === this.encry_password
  },
  securePassword: function (plainpassword) {
    if (!plainpassword) return "";
    try {
      return crypto.createHmac('sha256', this.salt)
        .update(plainpassword)
        .digest('hex')
    } catch (err) {
      return ""
    }
  }
}

```

```

        } catch (err) {
            return "";
        }
    }
}
module.exports = mongoose.model("User", userSchema)

```

Routes>auth.js

```

const { check, validationResult } = require('express-validator')
const express = require("express")
const router = express.Router()
const { isSignedIn, signout, signup, signin, registrationForm, loginForm,
  home } = require("../controllers/auth")

router.get("/registration", registrationForm)
router.get("/login", loginForm)
router.get("/home", home)

router.post("/signup", [
    check("firstname", "firstname should be at least 3 char").isLength({
min: 3 }),
    check("email", "email should be valid").isEmail(),
    check("password", "password should be at least 3 char").isLength({ mi
n: 1 }),
    check("phone", "phone no should be 10 digit").isLength({ max: 10 }),
], signup)

router.post("/signin", [
    check("email", "email is required").isEmail(),
    check("password", "password is required").isLength({ min: 3 }),
], signin)

// router.get("/testroute", isSignedIn, (req, res) => {
//     res.json({
//         message: "succesfull"
//     })
// })

router.get("/signout", signout)

```

```
module.exports = router
```

Routes > dashboard.js

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

const { dashboard } = require("../controllers/dashboard")

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

module.exports = router
```

Routes > profile.js

```
const { check, validationResult } = require('express-validator')
const path = require("path")
var multer = require('multer');
const express = require("express")
const { profileForm, updateProfile } = require("../controllers/profile")
const router = express.Router()

var storage1 = multer.diskStorage({
  destination: function (req, file, callback) {
    callback(null, 'uploads/');
  },
  filename: function (req, file, callback) {
    callback(null, '/' + Date.now() + file.originalname);
  }
});

var upload = multer({
  storage: storage1,
  fileFilter: function (req, file, callback) {
    var ext = path.extname(file.originalname);
    if (ext !== '.png' && ext !== '.jpg' && ext !== '.gif' && ext !==
    '.jpeg') {
      return callback(new Error('Only images are allowed'))
    }
    callback(null, true)
  },
```

```
    limits: {
      // fileSize: 1024 * 1024
    }
  });

router.get('/dashboard/profile/:id', profileForm)
router.post("/dashboard/profile/update", upload.single('image'), updateProfile)

module.exports = router
```

Routes>user

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

const { getUserById, getUser, updateUser, userPurchaseList } = require("../controllers/user")
const { isSignedIn } = require("../controllers/auth")

router.param("userId", getUserById)

router.get("/user/:userId", isSignedIn, getUser)

router.put("/user/:userId", isSignedIn, updateUser)

module.exports = router
```

Routes>article

```
const express = require("express")
var multer = require('multer');
const path = require("path")

const router = express.Router()
const { deleteArticleForm, showArticleAdmin, createArticle, articleSave, showArticle, showOneArticle, editArticleForm, updateArticle } = require("../controllers/article")
```

```

var storage1 = multer.diskStorage({
  destination: function (req, file, callback) {
    callback(null, 'uploads/');
  },
  filename: function (req, file, callback) {
    callback(null, '/' + Date.now() + file.originalname);
  }
});

var upload = multer({
  storage: storage1,
  fileFilter: function (req, file, callback) {
    var ext = path.extname(file.originalname);
    if (ext !== '.png' && ext !== '.jpg' && ext !== '.gif' && ext !==
'.jpeg') {
      return callback(new Error('Only images are allowed'))
    }
    callback(null, true)
  },
  limits: {
    // fileSize: 1024 * 1024
  }
});

router.get("/dashboard/article/new", createArticle)
router.post("/dashboard/article/save", upload.single('article_image'), ar
ticleSave)
router.get("/dashboard/article/:article_id", showOneArticle)
router.get("/dashboard/article/show/all", showArticle)
router.get("/dashboard/article/show/all/admin", showArticleAdmin)
router.get("/dashboard/article/edit/:id", editArticleForm)
router.get("/dashboard/article/delete/:id", deleteArticleForm)
router.post("/dashboard/article/update", upload.single('article_image'),
updateArticle)
module.exports = router

```

Contoller>auth.js

```

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

```

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

exports.home = (req, res) => {
  res.sendFile("index.html", { root: './public' })
}

exports.registrationForm = (req, res) => {
  res.sendFile("registration.html", { root: './public' })
}

exports.loginForm = (req, res) => {
  res.sendFile("login.html", { root: './public' })
}

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 DB"
      })
    }
    res.redirect("/api/login");
  })
}

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 email does not exists"
        })
    }

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

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

    //send response to front end
    const { _id, firstname, email, role } = user
    res.redirect("/api/dashboard");

})

}

exports.signout = (req, res) => {
    res.clearCookie("token")
    res.clearCookie("id")
    res.clearCookie("islogdinSuccess")

    res.redirect("/api/login")
}

```

```
//protected routes
exports.isSignedIn = expressJwt({
  secret: process.env.SECRET,
  userProperty: "auth"
})
```

Controller>dashbaord

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

exports.dashboard = (req, res) => {
  if (req.cookies.islogdinSuccess) {
    let id = req.cookies.id
    User.findById(id).exec((err, profile) => {
      if (err || !profile) {
        return res.status(400).json({
          error: "No profile was found"
        })
      }
      profile.salt = undefined
      profile.encyr_password = undefined
      res.render('dashboard', { profile: profile })
    })
  } else {
    res.redirect("/api/login")
  }
}
```

Controller >profile.js

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

exports.profileForm = (req, res) => {
  if (req.cookies.islogdinSuccess) {
    let id = req.params.id
    User.findById(id).exec((err, profile) => {
      if (err || !profile) {
        return res.status(400).json({
          error: "No profile was found"
        })
      }
    })
  }
}
```



```

        })
    }
    profile.salt = undefined
    profile.encyr_password = undefined
    res.render("profile", { profile: profile })
  })
} else {
  res.redirect("/api/login")
}
}

exports.updateProfile = (req, res) => {
  if (req.cookies.islogdinSuccess) {
    let id = req.cookies.id
    User.findOneAndUpdate({ _id: id }, {
      $set: {
        image: req.file.filename,
        firstname: req.body.firstname,
        lastname: req.body.lastname,
        email: req.body.email,
        gender: req.body.gender,
        phone: req.body.phone,
        userinfo: req.body.userinfo,
      }
    }).exec((err, profile) => {
      if (err) {
        return res.status(400).json({
          error: "No user was found in DB and not updated"
        })
      }
      res.redirect("/api/dashboard")
    })
  } else {
    res.redirect("/api/login")
  }
}
}

```

Controller >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 DB"
      })
    }

    req.profile = user
    next()
  })
}

exports.getUser = (req, res) => {
  req.profile.salt = undefined
  req.profile.ency_password = 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"
        })
      }
      user.salt = undefined
      user.ency_password = undefined
      res.json(user)
    }
  )
}
```

Controller >article.js

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

exports.createArticle = (req, res) => {
  if (req.cookies.islogdinSuccess) {
    res.render("article", { id: req.cookies.id })
  } else {
    res.redirect("/api/login")
  }
}

exports.articleSave = (req, res) => {
  if (req.cookies.islogdinSuccess) {
    let articleBody = {
      ...req.body,
      article_image: req.file.filename,
    }
    // console.log(articleBody)
    const article = new Article(articleBody)
    article.save((err, article) => {
      if (err) {
        return res.status(400).json({
          err: "NOT able to save article in DB"
        })
      }
      res.redirect("/api/dashboard/article/" + article._id)
    })
  } else {
    res.redirect("/api/login")
  }
}

exports.showOneArticle = (req, res) => {
  let article_id = req.params.article_id
  Article.findById(article_id).exec((err, article) => {
    if (err || !article) {
```

```

        return res.status(400).json({
          error: "No article was found"
        })
      }
      let id = article.author
      User.findById(id).exec((err, profile) => {
        if (err || !profile) {
          return res.status(400).json({
            error: "No profile was found"
          })
        }
        res.render("article_one", { article: article, profile: profile })
      })
    })
  })
}

exports.showArticle = (req, res) => {
  Article.find({}).exec((err, article) => {
    if (err) {
      res.status(400).json({
        err: "NOT able to find any articles"
      })
    }
    // console.log(req.cookies)
    res.render("article_all", { article: article })
  })
}

exports.showArticleAdmin = (req, res) => {
  if (req.cookies.isloggedinSuccess) {
    Article.find({}).exec((err, article) => {
      if (err) {
        res.status(400).json({
          err: "NOT able to find any articles"
        })
      }
    })
  }
}

```

```

        // console.log(req.cookies)
        res.render("article_all_admin", { article: article })
    })
} else {
    res.redirect("/api/login")
}
}

exports.editArticleForm = (req, res) => {
    if (req.cookies.islogdinSuccess) {
        // console.log(req.params)
        let id = req.params.id

        Article.findById(id, function (err, article) {
            if (err || !article) {
                res.status(400).json({
                    err: "article is NOT avialable"
                })
            }
            res.render('articleUpdate', { article: article })

        })
    } else {
        res.redirect("/api/login")
    }
}

exports.updateArticle = (req, res) => {
    if (req.cookies.islogdinSuccess) {
        const updatearticle = req.body
        updatearticle.article_image = req.file.filename
        const {
            _id,
            title,
            preview,
            content,
            code,
            article_image,
            video_URL,
            author,

```

```

        tag_one,
        tag_two,
        tag_three
    } = updatearticle
    Article.findByIdAndUpdate(_id, {
        title: title,
        preview: preview,
        content: content,
        code: code,
        article_image: article_image,
        video_URL: video_URL,
        author: author,
        tag_one: tag_one,
        tag_two: tag_two,
        tag_three: tag_three,
    }, (err, article) => {
        if (err) {
            res.status(400).json({
                err: "artcile is NOt edited"
            })
        }
        res.redirect("/api/dashboard/article/" + article._id)
    })
} else {
    res.redirect("/api/login")
}
}

exports.deleteArticleForm = (req, res) => {
    if (req.cookies.islogdinSuccess) {
        const id = req.params.id
        Article.deleteOne({ _id: id }, (err, quiz) => {
            if (err) {
                res.status(400).json({
                    err: "Article question cannot deleted"
                })
            }
            res.redirect("/api/dashboard/article/show/all/admin")
        })
    }
}

```

```

    } else {
      res.redirect("/api/login")
    }
  }
}

```

Folder Architecture

The screenshot shows a Windows File Explorer window titled 'views'. The left sidebar displays the 'Quick access' pane with various locations like Desktop, Downloads, Documents, Pictures, whatsapp, all, controllers, Screenshots, stacks, This PC, 3D Objects, Desktop, Downloads, Music, Pictures, Videos, Local Disk (C:), and Network. The main pane shows a list of files and folders within the 'views' directory. The 'controllers' folder is selected.

Name	Date modified	Type	Size
controllers	02-11-2020 07:14 AM	File folder	
models	02-11-2020 06:57 AM	File folder	
node_modules	01-11-2020 03:55 PM	File folder	
public	02-11-2020 08:29 PM	File folder	
routes	02-11-2020 07:14 AM	File folder	
uploads	02-11-2020 08:15 PM	File folder	
views	02-11-2020 07:37 PM	File folder	
.env	31-10-2020 12:29 PM	ENV File	1 KB
app.js	02-11-2020 07:17 AM	JavaScript File	2 KB
package.json	02-11-2020 09:34 PM	JSON File	1 KB
package-lock.json	01-11-2020 03:55 PM	JSON File	142 KB

The taskbar at the bottom shows the Windows Start button, a search bar, and several pinned applications including Edge, Mail, Teams, File Explorer, Chrome, and Word. The system clock indicates the time is 10:49 PM on 02-11-2020.