

LOVELY PROFESSIONAL UNIVERSITY

Online Assignment-2 (Project)

Course Code: INT222

Course Title:
Advanced Web Development

Name: Bankar Piyush Sanjay
Reg.No.: 11710240
Roll.No.: RKM013B47

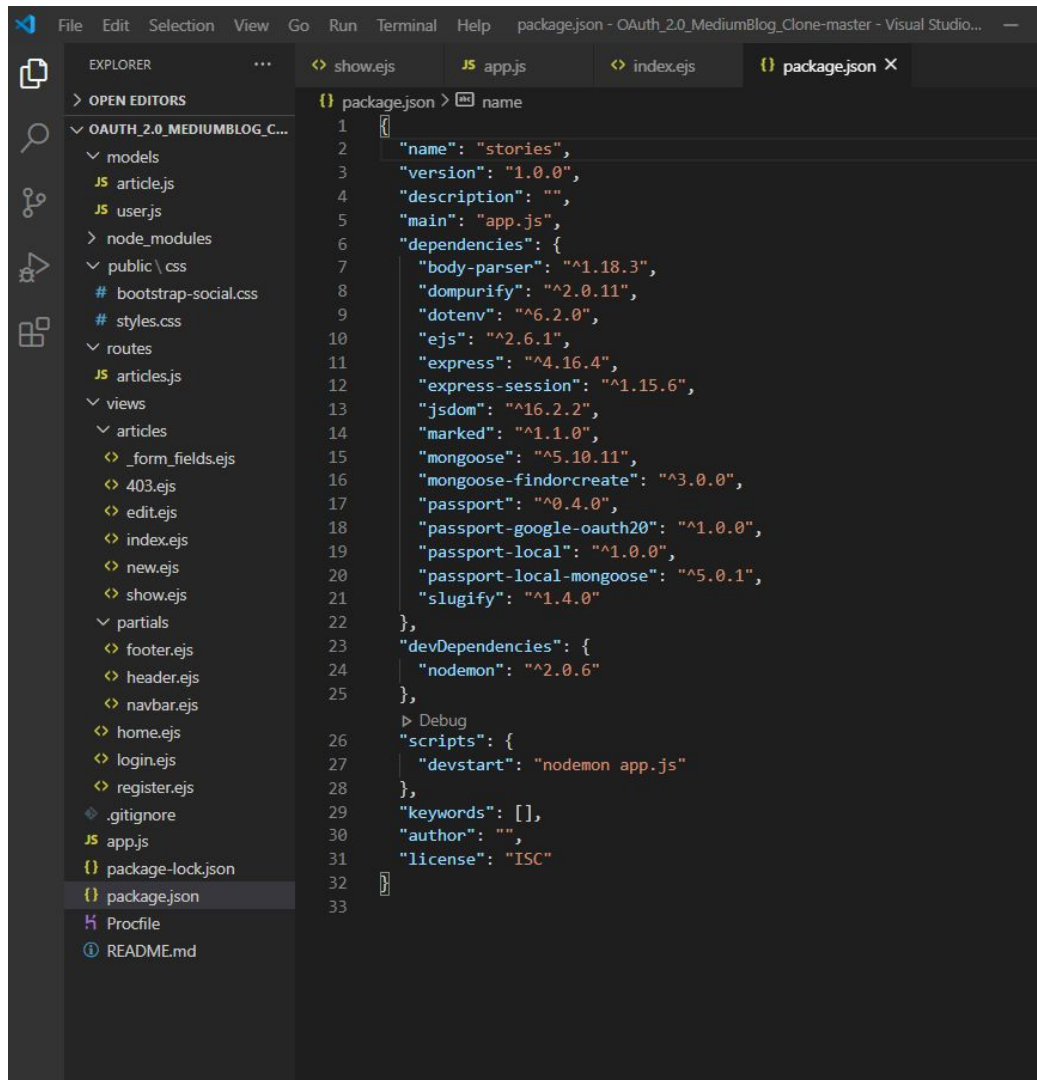
Blogger site (Medium clone)

Github: <https://github.com/piyushb1/mediumClone>

Introduction

The project is based on the concept of multi blogging where multiple people write blogs and articles on the same website. Each user is considered as the author when he/she registered on the website. The website is coded in **NodeJS** environment supported by **ExpressJS** framework and **MongoDB** database. Other supportive file structure and libraries used are mentioned as below:

```
"body-parser": "^1.18.3",    "dompurify": "^2.0.11",    "dotenv": "^6.2.0",    "ejs": "^2.6.1",  
"express": "^4.16.4",    "express-session": "^1.15.6",    "jsdom": "^16.2.2",    "marked": "^1.1.0",  
"mongoose": "^5.10.11",    "mongoose-findorcreate": "^3.0.0",    "passport": "^0.4.0",  
"passport-google-oauth20": "^1.0.0",    "passport-local": "^1.0.0",    "passport-local-mongoose":  
"^5.0.1",    "slugify": "^1.4.0"
```



The screenshot shows the Visual Studio Code interface with the Explorer sidebar on the left and the main editor area on the right. The Explorer sidebar shows a project structure for 'OAUTH_2_0_MEDIUMBLOG_C...'. The main editor area displays the 'package.json' file, which contains the following content:

```
1 {
2   "name": "stories",
3   "version": "1.0.0",
4   "description": "",
5   "main": "app.js",
6   "dependencies": {
7     "body-parser": "^1.18.3",
8     "dompurify": "^2.0.11",
9     "dotenv": "^6.2.0",
10    "ejs": "^2.6.1",
11    "express": "^4.16.4",
12    "express-session": "^1.15.6",
13    "jsdom": "^16.2.2",
14    "marked": "^1.1.0",
15    "mongoose": "^5.10.11",
16    "mongoose-findorcreate": "^3.0.0",
17    "passport": "^0.4.0",
18    "passport-google-oauth20": "^1.0.0",
19    "passport-local": "^1.0.0",
20    "passport-local-mongoose": "^5.0.1",
21    "slugify": "^1.4.0"
22  },
23  "devDependencies": {
24    "nodemon": "^2.0.6"
25  },
26  "scripts": {
27    "devstart": "nodemon app.js"
28  },
29  "keywords": [],
30  "author": "",
31  "license": "ISC"
32 }
```

EJS

EJS template library is used for encoding routes and html markups easily. Most of the routes are redirected to respective route files like article related and users.

```

JS app.js > ...
1
2   require('dotenv').config();
3   const express = require("express");
4   const bodyParser = require("body-parser");
5   const ejs = require("ejs");
6   const articleRouter = require('./routes/articles')
7   const User = require('./models/user')
8   const Article = require("./models/article")
9   const mongoose = require("mongoose");
10  const session = require('express-session');
11  const passport = require("passport");
12  const passportLocalMongoose = require("passport-local-mongoose");
13
14  const app = express();
15
16  app.use(express.static("public"));
17  app.set('view engine', 'ejs');
18  app.use(bodyParser.urlencoded({
19    extended: false
20  }));
21
22  // DB Config
23  mongoose.connect(process.env.MONGO_URL || 'mongodb://localhost/blog', {
24    useNewUrlParser: true,
25    useUnifiedTopology: true,
26    useCreateIndex: true,
27  });
28
29  //mongodb+srv://admin:n4tI7wwGSM0iQURw@cluster0.jorqq.mongodb.net/blog?retryWri
30
31  mongoose.connection.on('connected', () => {
32    console.log('Mongoose connected');
33  });
34
35
36  app.use(session({
37    secret: "Our little secret.",
38    resave: false,
39    saveUninitialized: false
40  }));
41
42  app.use(passport.initialize());
43  app.use(passport.session());
44
45

```

Routes

The user register and login pages are coded using **passport-local** method and data is fetched from MongoDB database using **mongoose** library.

[Home](#)[Logout](#)

Login

Email

Enter Email

Password

Enter Password

Login

If you don't have Account

Register

[Home](#)[Logout](#)

Register

Name

Enter Name

Email

Enter Email

Password

Enter Password

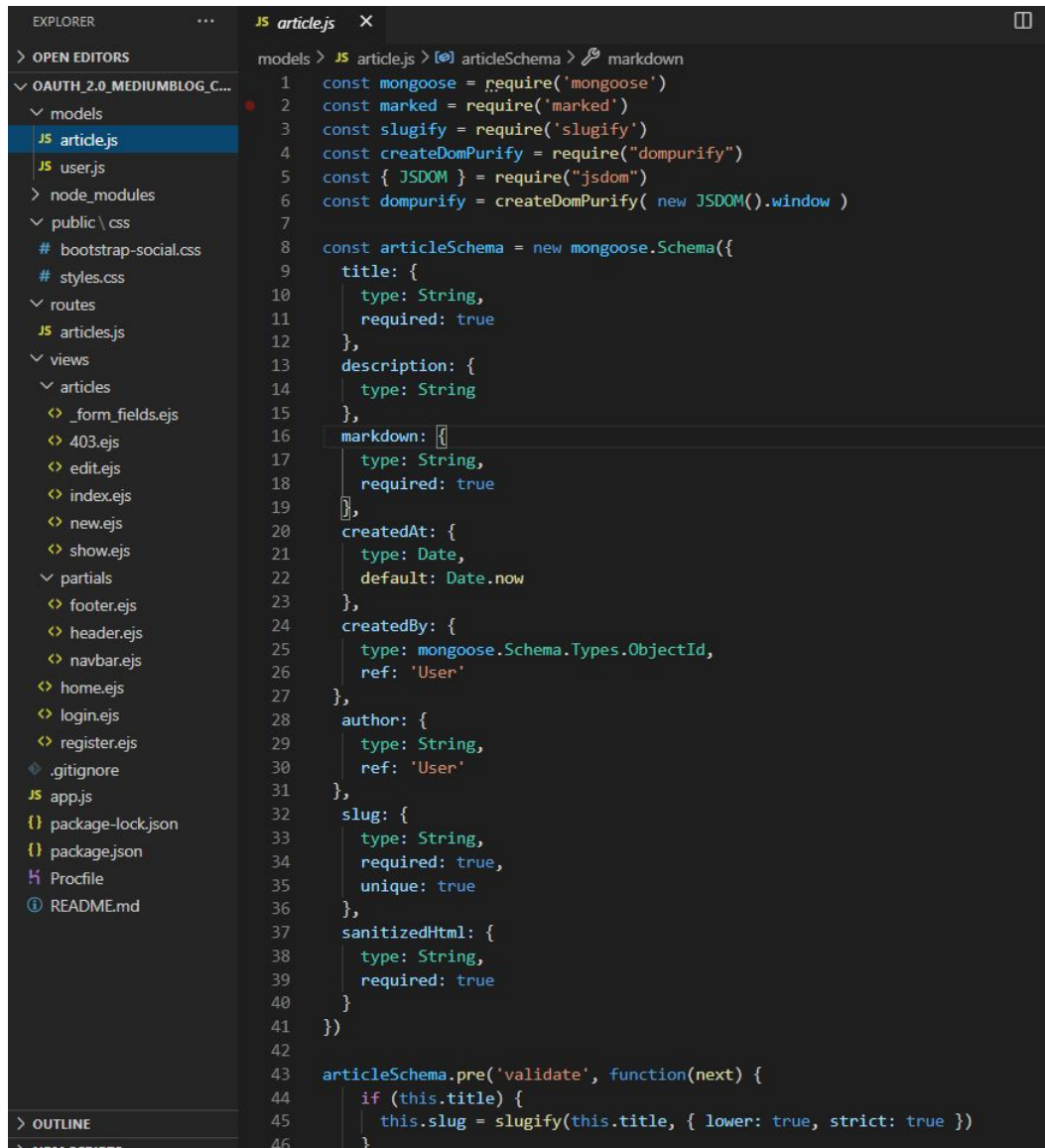
Register

Repeating codes like layouts, css properties and header files are stored in single file in **partials** and then retrieved for later use in each page of website. This way a code for navigation bars , footers and forms need not be written multiple times.

Models

Each type of data is stored in MongoDB database. The data types are given the predefined Schema using mongoose library like **users** and **articles**.

The libraries like **dompurify** and **slugify** are used to create a unique and simple urls for each article. The time and author of each article is recorded in it as soon as it is created.



```
1  const mongoose = require('mongoose')
2  const marked = require('marked')
3  const slugify = require('slugify')
4  const createDomPurify = require("dompurify")
5  const { JSDOM } = require("jsdom")
6  const dompurify = createDomPurify( new JSDOM().window )
7
8  const articleSchema = new mongoose.Schema({
9    title: {
10      type: String,
11      required: true
12    },
13    description: {
14      type: String
15    },
16    markdown: {
17      type: String,
18      required: true
19    },
20    createdAt: {
21      type: Date,
22      default: Date.now
23    },
24    createdBy: {
25      type: mongoose.Schema.Types.ObjectId,
26      ref: 'User'
27    },
28    author: {
29      type: String,
30      ref: 'User'
31    },
32    slug: {
33      type: String,
34      required: true,
35      unique: true
36    },
37    sanitizedHtml: {
38      type: String,
39      required: true
40    }
41  })
42
43  articleSchema.pre('validate', function(next) {
44    if (this.title) {
45      this.slug = slugify(this.title, { lower: true, strict: true })
46    }
47  })
```

Views

The repetitive code of pages like articles is coded only once. And then the data of each article is retrieved from the database when it is required.

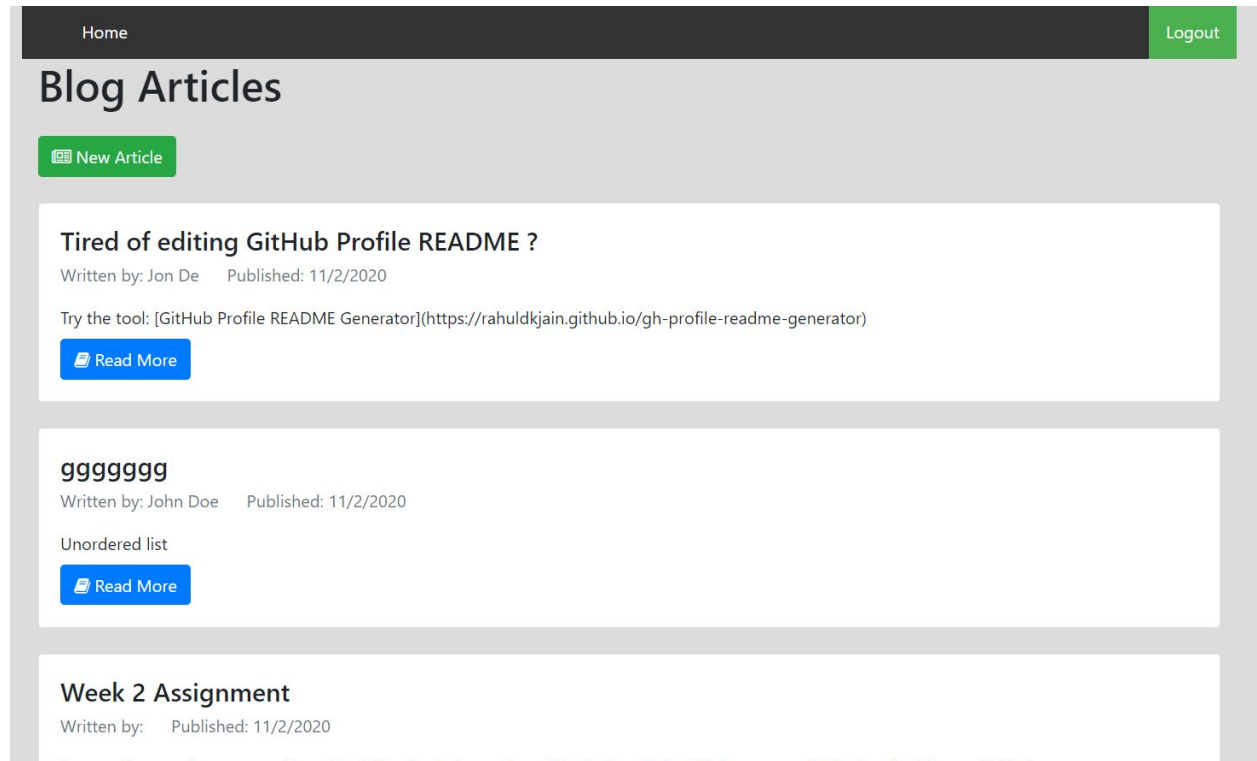
```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css" integrity="sha384-Gn5384x17Cky7gpJzUD/rVd3KmHCzK1sYg4mIweZ2kOPv22kJ339dd/pX6b28Zo">
7   <meta http-equiv="X-UA-Compatible" content="ie=edge">
8   <link href="https://stackpath.bootstrapcdn.com/font-awesome/4.7.0/css/font-awesome.min.css" rel="stylesheet" integrity="sha384-v3ynxlr6uM2hsYsn4XEwhMIYxtJNVSR5aPWEMvSzLiaQb2yqqHlL9wys7QpqX9sIhR8b">
9   <title><%= article.title %></title>
10
11 </head>
12 <body>
13   <%- include('../partials/navbar') %>
14   <div class="container">
15
16     <h1 class="mb-1"><%= article.title %></h1>
17     <div class="text-muted mb-2"> Written by
18       <%= article.author %>
19     </div>
20     <div>
21       <span style="text-align: right;">Published :
22       <%= article.createdAt.toLocaleDateString() %>
23     </span>
24   </div>
25
26   <a href="/articles/edit/<%= article.id %>" class="btn btn-info"> <i class="fa fa-pencil aria-hidden="true"></i> Edit</a>
27   <form action="/articles/delete/<%= article.id %>" method="POST" class="d-inline">
28     <button type="submit" class="btn btn-danger"> <i class="fa fa-trash aria-hidden="true"></i> Delete</button>
29   </form>
30
31   <div>
32     <%= article.sanitizedHtml %>
33   </div>
34 </div>
35 </body>
36 </html>
```

Authentication

The website can only be used after you login. All the articles published are visible to everyone. But only the authors of articles are authorized to **edit** or **delete** them. The authentication is done by verifying the author element in “article” to the `_id` element of currently logged in person in the session.

HomePage

After you login, you will be redirected to the front page where all the articles will be displayed in order of latest article first.



This will display the title, intro, date of published and the author of article in cascade manner to be read.

