ONLINE MEDICAL STORE

Name-shubham Mishra

Registration No-:11715863

Roll no-:RKM012A30

Github link-: https://github.com/shub22mishra/onlinemedicalstore

SUBMITTED TO
NEHA SHARMA MA'AM

Objective

The objective of this project is to provide 24 * 7 medical service to all the people. I have done this project using Node, Express & MongoDB. In this project I have provided some features like Upload Prescription, User Registration, Login, Medicine category and so on.

This is the User Interface of the project

Technology used

- I have used **Node.js** For backend such as for routing.
- **Express.js-:** Node.js Web framework.
- **MongoDB-:** For database
- **Nodemon-:** To do Automatic Refresh. With a small change in code it will refresh our server
- **Joi-:** For Validation
- **EJS-:** Embedded JavaScript Templating. It is a view engine
- **Chai/Mocha-:** to perform basic testing using chai, mocha and chai-http, we will add a script that will invoke our test and we will see pass when tests will pass.

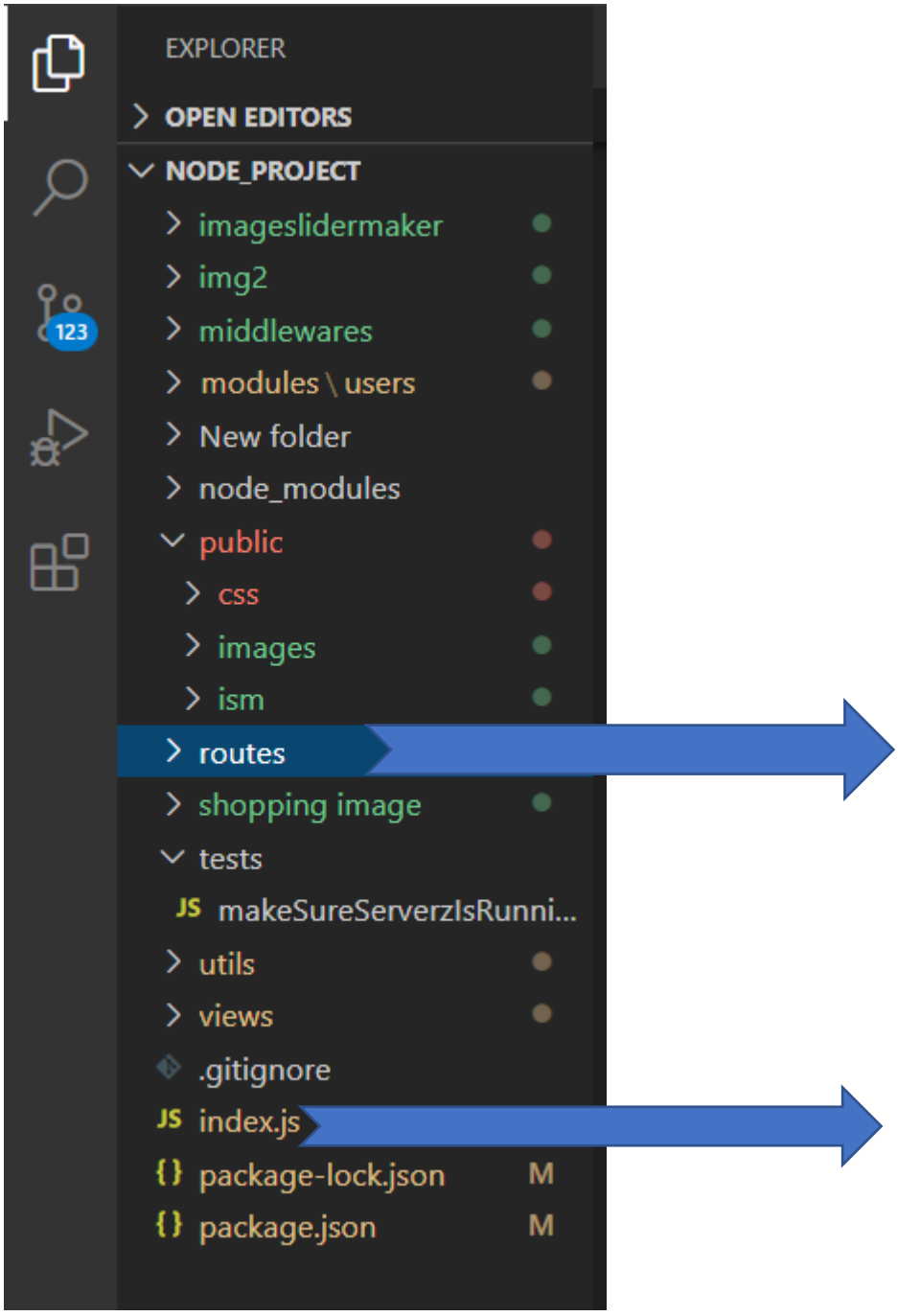
SEVER is Running at localhost:4000

index.js > ...

```
const express = require("express");
const session = require("express-session");
const bodyParser = require("body-parser");
require("./utils/db.config");
const MongoStore = require("connect-mongo")(session);
const mongoDbConnection = require("./utils/db.config");
const passport = require("passport");
require("./utils/authStategies/localStrategy");

const authRoutes = require("./routes/authRoutes");
const app = express();
app.use(express.static("public"));

app.use(bodyParser.urlencoded({ extended: false }));
app.set("view engine", "ejs");
// app.set('trust proxy', 1)
app.use(
  session({
    secret: "788a154e2a8d07c4cafdee4a7d6dff2d90ab2586",
    resave: false,
    saveUninitialized: true,
    cookie: { secure: false },
    store: new MongoStore({ mongooseConnection: mongoDbConnection }),
  })
);
```

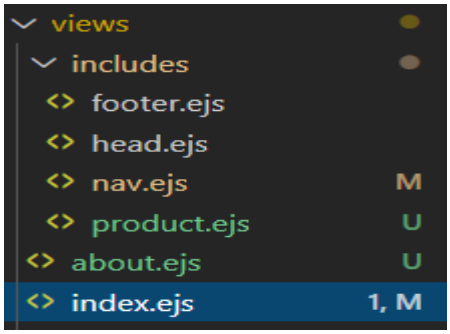
```
app.use(passport.initialize());
app.use(passport.session());

app.use("/", authRoutes);

app.get("/", (req, res) => {
  req.session.views = (req.session.views || 0) + 1
  console.log("User:", req.user);
  return res.render("index");
});

app.listen(4000, () => {
  console.log("Server running at port 4000");
});

module.exports = app;
```

Project file structure



Home page code and screenshots.

Here I create a view using EJS name as **index.ejs** inside **view** folder.



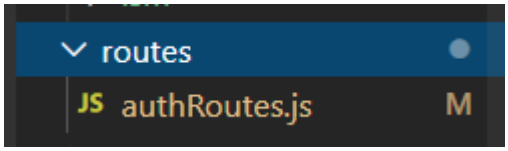
```
<link rel="stylesheet" type="text/css" href="/css/medical.css" ></link>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css"></head>
<link rel="stylesheet" href="ism/css/my-slider.css"/>

<script src="/ism/js/ism-2.2.min.js"></script>


<title>Medical Shop</title>
</head>

<body>
<div class="page">


<div class="header">
<img src="/images/logo.PNG">
<img src="/images/header.jpg" style=" width:1302px;height:100px;">

</div>


<div class="links">


<div class="rightmenu" align="center">
```

```
Server running at port 4000
Connected to MongoDB
(node:12604) DeprecationWarning: collection.ensureIndex is deprecated. Use createIndexes instead.
User: undefined
[nodemon] restarting due to changes...
[nodemon] starting `node index.js`
Server running at port 4000
Connected to MongoDB
(node:13200) DeprecationWarning: collection.ensureIndex is deprecated. Use createIndexes instead.
```

```
<div class="rightmenu" align="center">
  <ul>
  <li><a href="#"/index">Home</a></li>
  <li><a href="/register">Register</a></li>
  <li><a href="/about">ABOUT</a></li>
  <li><a href="/prescription">Upload Prescription</a></li>
  <li><a href="/login">Login</a></li>
  <li><a href="/medicine">Medicine</a></li>
  <li><a href="/medicine">LIFE-STYLE</a></li>
  <li><a href="/medicine">TREATMENTS</a></li>

  </ul>
  </div>

  <input type="text" placeholder="Search for Prescription Medicines & OTC Products..." name="search" align="center">
  <button type="submit" style="color:#FF0000;" ><i class="fa fa-search" ></i></button>




</div>

</div>
```

```
Server running at port 4000
Connected to MongoDB
(node:12604) DeprecationWarning: collection.ensureIndex is deprecated. Use createIndexes instead.
User: undefined
[nodemon] restarting due to changes...
[nodemon] starting `node index.js`
Server running at port 4000
Connected to MongoDB
(node:13200) DeprecationWarning: collection.ensureIndex is deprecated. Use createIndexes instead.
```

Now inside route Folder I have done the routing




```
const express = require("express");
const router = express.Router();
const { addUser } = require("../modules/users/service/userService");
const {
  registerSchema,
} = require("../modules/users/validations/authValidation");
const {
  joiErrorFormatter,
  mongooseErrorFormatter,
} = require("../utils/validationFormatter");
const passport = require("passport");

/**
 * Shows page for user registration
 */
router.get("/register", (req, res) => {
  return res.render("register", { message: {}, formData: {}, errors: {} });
});

/**
 * Handles user registration
 */
router.post("/register", async (req, res) => {
  try {
    const validationResult = registerSchema.validate(req.body, {
      abortEarly: false,
    });
```

I have performed automated testing using chai/mocha

tests > JS makeSureServerIsRunning.js > ...

```
const chai = require("chai");
const chaiHttp = require("chai-http");
const app = require("../index");

// Configure chai
chai.use(chaiHttp);
chai.should();

describe("Make sure that status is 200", () => {
  it("should return 200", (done) => {
    chai
      .request(app)
      .get("/")
      .end((err, res) => {
        res.should.have.status(200);
        res.body.should.be.a("object");
        done();
      });
  });
});

describe("Make sure register fails on no data", () => {
  it("should return 400", (done) => {
    chai
      .request(app)
      .post("/register")
      .end((err, res) => {
```

PROBLEMS 25 OUTPUT TERMINAL DEBUG CONSOLE

```
Connected to MongoDB
(node:14536) DeprecationWarning: collection.ensureIndex is deprecated. Use createIndexes instead.
User: undefined
[nodemon] restarting due to changes...
[nodemon] starting `node index.js`
Server running at port 4000
Connected to MongoDB
(node:12604) DeprecationWarning: collection.ensureIndex is deprecated. Use createIndexes instead.
User: undefined
```

There's a

This code states that the return status should be 200. If it return other than 200 . It will throw an error.

2nd Module Registration page

REGISTRATION PAGE

I have created this page using EJS. In this page I have performed Validation using JOI(which is a express middleware for validation).

I have also store all the user detail in Database. I have use MONGODB for the database purpose.

I have also performed password ENCRYPTION for more security purpose. This encrypted password is going to store in database**. For encryption I use node.bcrypt.js**

```
npm install bcrypt
```

```
password: "$2b$10$5cXC8JFg3l1aZwcMZUt30.DcU5EB/OMAdGTepmY2NxZv5h9famrjK"
```

Here password get encrypted

code and screenshot

```
        <div class="form-group">
          <label for="password">Password<sup class="text-danger">*</sup></label>
          <input
            type="password"
            name="password"
            class="form-control <%= errors.password ? 'is-invalid' : '' %>"
            id="password"
            required
          />
          <% if (errors.password) { %>
          <div class="invalid-feedback"><%= errors.password[0] %></div>
          <% } %>
        </div>

        <div class="form-group">
          <label for="repeat_password"
            >Repeat Password<sup class="text-danger">*</sup></label
          >
          <input
            type="password"
            name="repeat_password"
            class="form-control"
            id="repeat_password"
            required
          />
```

```
<body>
  <%- include('includes/nav'); %>
  <div class="container mt-5">
    <h1>Registration</h1>
    <% if (message.type) { %>
    <div
      class="alert alert-<%= message.type === 'success' ? 'success' : 'danger' %>"
      role="alert"
    >
      <%= message.body %>
    </div>
    <% } %>
    <form method="POST" action="/register">
      <div class="form-group">
        <label for="name">Name<sup class="text-danger">*</sup></label>
        <input
          type="text"
          class="form-control <%= errors.name ? 'is-invalid' : '' %>"
          id="name"
          name="name"
          value="<%= formData.name %>"
          maxlength="64"
          minlength="2"
          required
        />
```

```
    <% if (errors.name) { %>
    <div class="invalid-feedback"><%= errors.name[0] %></div>
    <% } %>
  </div>

  <div class="form-group">
    <label for="email"
      >Email address<sup class="text-danger">*</sup></label
    >
    <input
      type="email"
      class="form-control <%= errors.email ? 'is-invalid' : '' %>"
      id="email"
      name="email"
      aria-describedby="emailHelp"
      value="<%= formData.email %>"
      maxlength="128"
      required
    />
    <% if (errors.email) { %>
    <div class="invalid-feedback"><%= errors.email[0] %></div>
    <% } %>
    <small id="emailHelp" class="form-text text-muted"
      >We'll never share your email with anyone else.</small
    >
  </div>
```

FOR VALIDATION

```
modules > users > validations > JS authValidation.js > [e] registerSchema
1   const Joi = require("@hapi/joi");
2
3   const registerSchema = Joi.object({
4     name: Joi.string().replace(/[0-9]/g, "").min(2).max(64).required(),
5
6     password: Joi.string().required(),
7
8     repeat_password: Joi.ref("password"),
9
10    email: Joi.string()
11      .trim()
12      .lowercase()
13      .email({ minDomainSegments: 2, tlds: { allow: ["com", "net", "in"] } }),
14  }).with("password", "repeat_password");
15
16  module.exports = { registerSchema };
17
```

With the help of this code I validate name as well as password.

If user password is not matched with conform password it will throw error

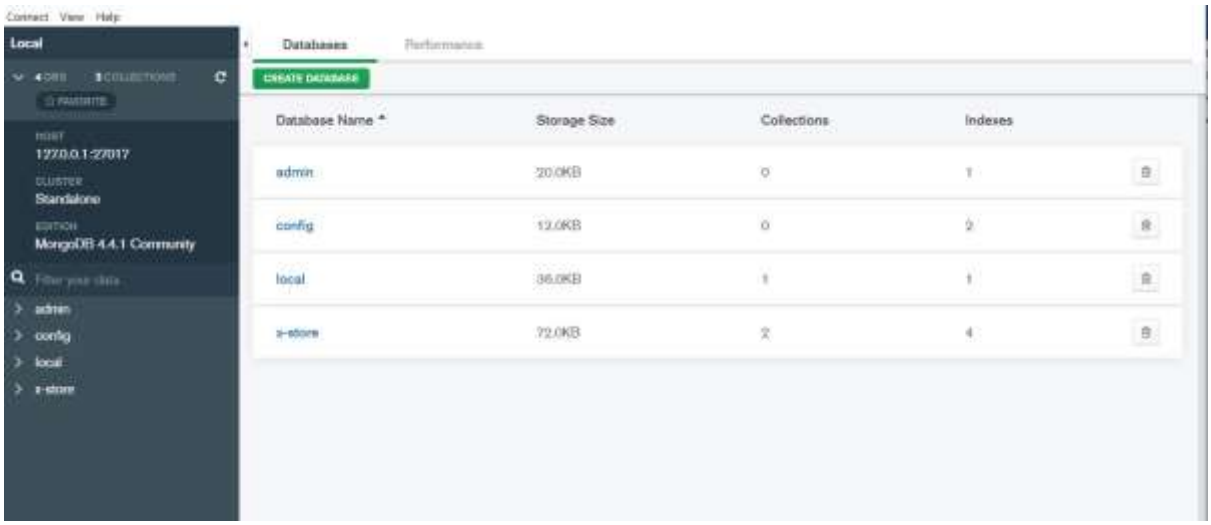
If the detail entered by user is correct

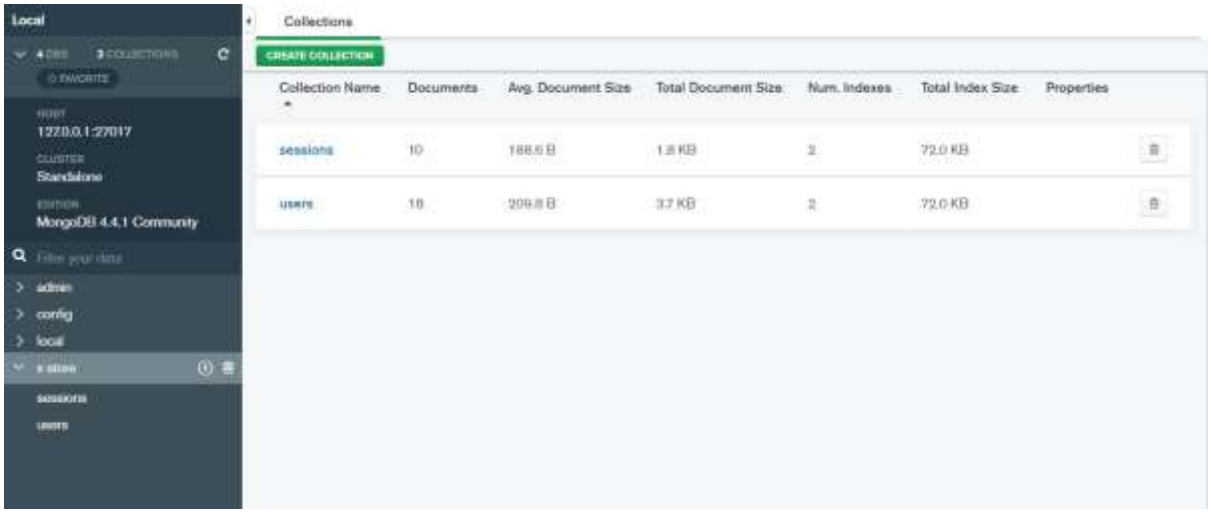
DATA CONNECTIVITY(MONGODB)

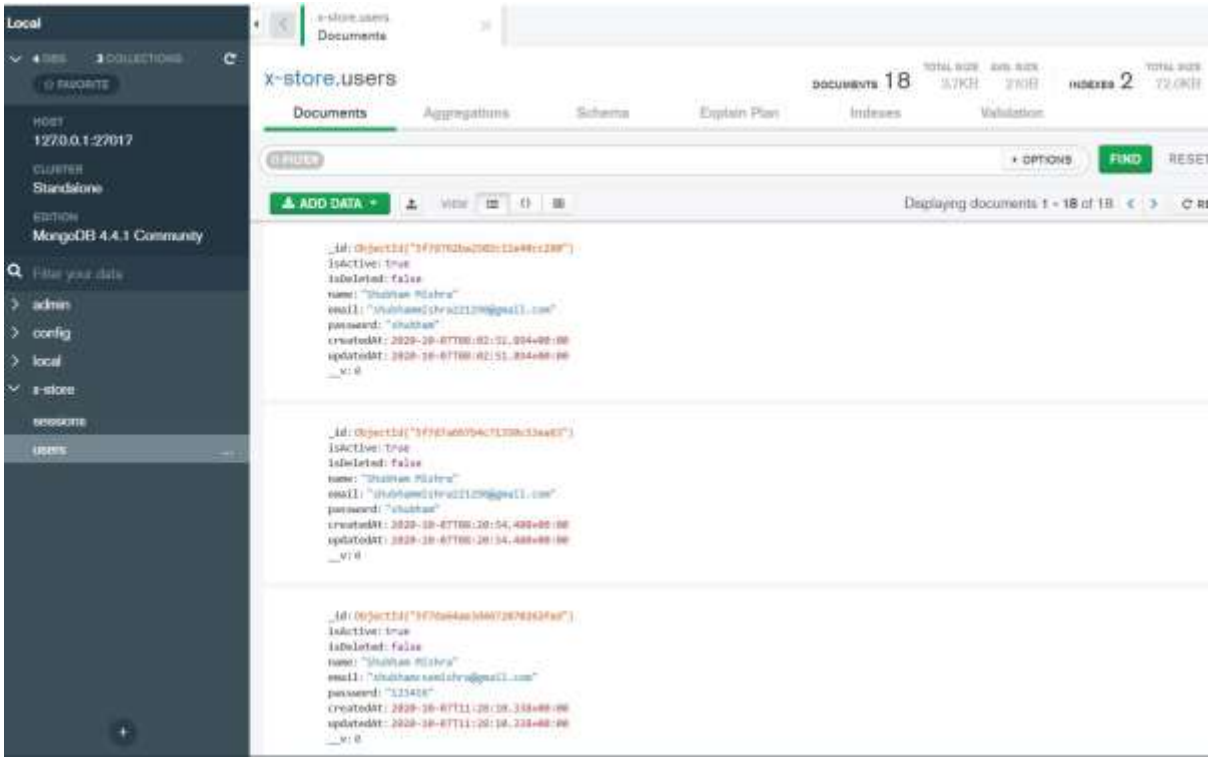
utils > JS db.config.js > ...

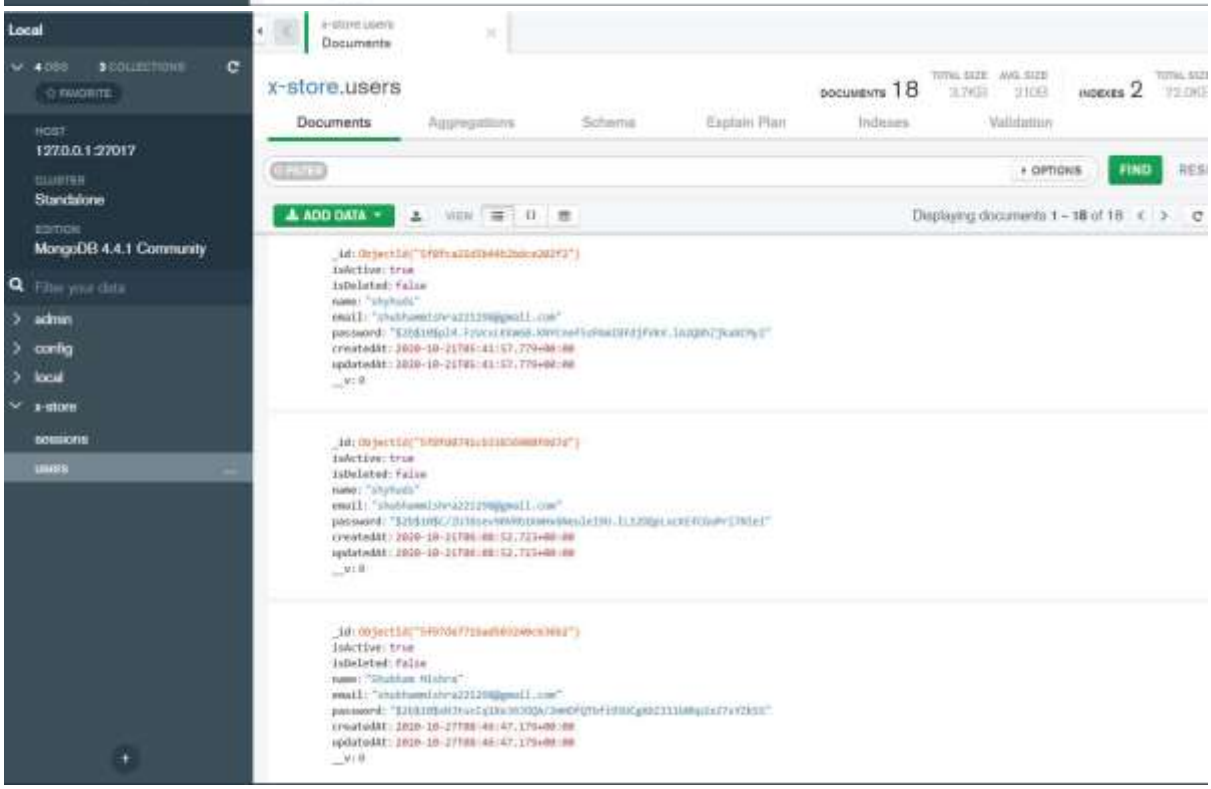
```
1   const mongoose = require("mongoose");
2   mongoose.connect("mongodb://localhost:27017/x-store", {
3     useNewUrlParser: true,
4     useUnifiedTopology: true,
5   });
6
7   mongoose.connection.once("open", () => {
8     console.log("Connected to MongoDB");
9   });
10  module.exports = mongoose.connection;
11
```

Data entered by the user is stored in the database named as x-store;

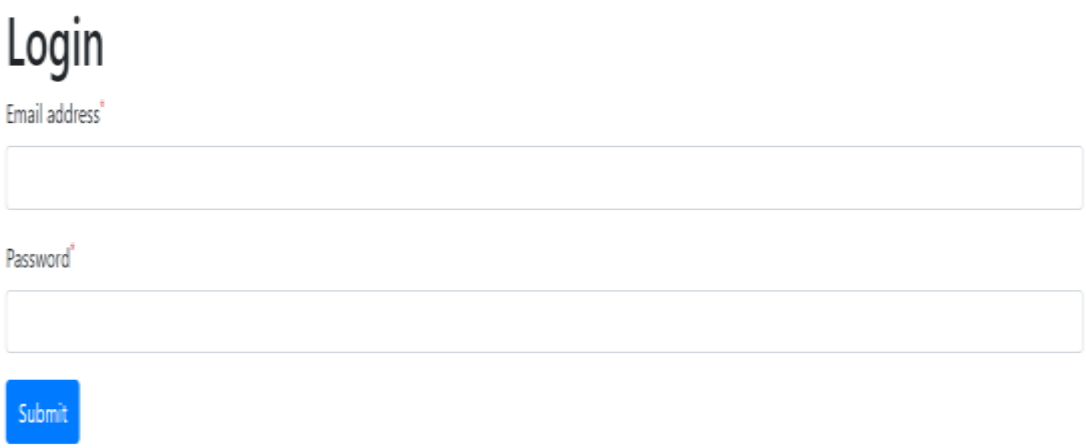





Local
x-store.users
Documents
x-store.users
DOCUMENTS 18
INDEXES 2
Documents
Aggregations
Schema
Explain Plan
Indexes
Validation
+ OPTIONS
FIND
ADD DATA
Displaying documents 1 - 18 of 18
HOST
127.0.0.1:27017
CLUSTER
Standalone
EDITION
MongoDB 4.4.1 Community
admin
config
local
x-store
sessions
users

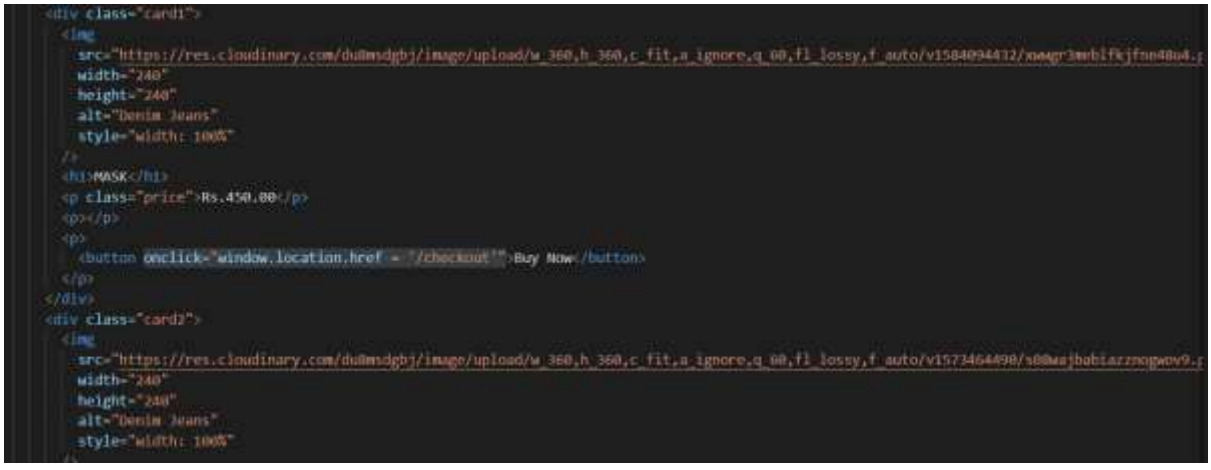

Local
x-store.users
Documents
x-store.users
DOCUMENTS 18
INDEXES 2
Documents
Aggregations
Schema
Explain Plan
Indexes
Validation
+ OPTIONS
FIND
ADD DATA
Displaying documents 1 - 18 of 18
HOST
127.0.0.1:27017
CLUSTER
Standalone
EDITION
MongoDB 4.4.1 Community
admin
config
local
x-store
sessions
users

Once the user is successful registered .it will redirected to Login Page

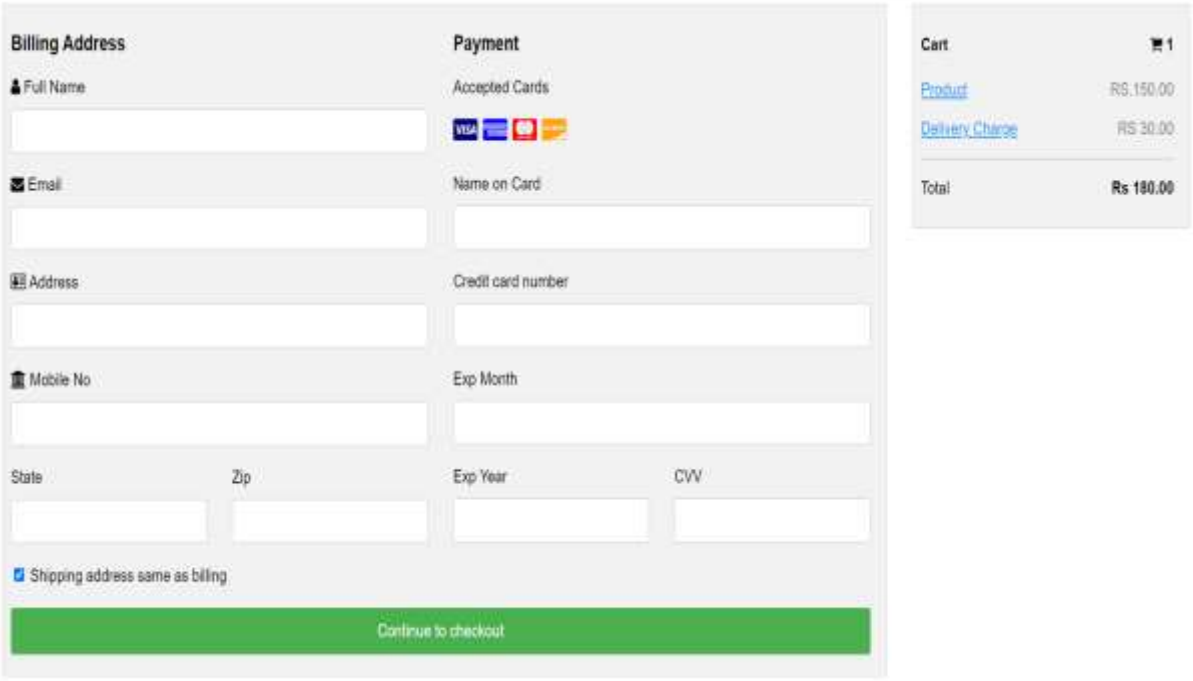


Now there is a product page where user can buy medicine

Code screenshot



After click on buy now it will be redirected to Checkout form




```
<div class="col-25">
  <div class="container">
    <h4>Cart <span class="price" style="color:black"><i class="fa fa-shopping-cart"></i> <b>1</b></span></h4>
    <p><a href="#">Product </a> <span class="price"> RS.150.00</span></p>
    <p><a href="#">Delivery Charge</a> <span class="price">RS 30.00</span></p>

    <hr>
    <p>Total <span class="price" style="color:black"><b>Rs 180.00</b></span></p>
  </div>
 </div>
</div>
```

After final checkout it will redirect to

Thanks for shopping with us

You will receive your medicine as soon as possible.

Stay safe stay healthy

About us page

Upload Prescription Page

----------End----------