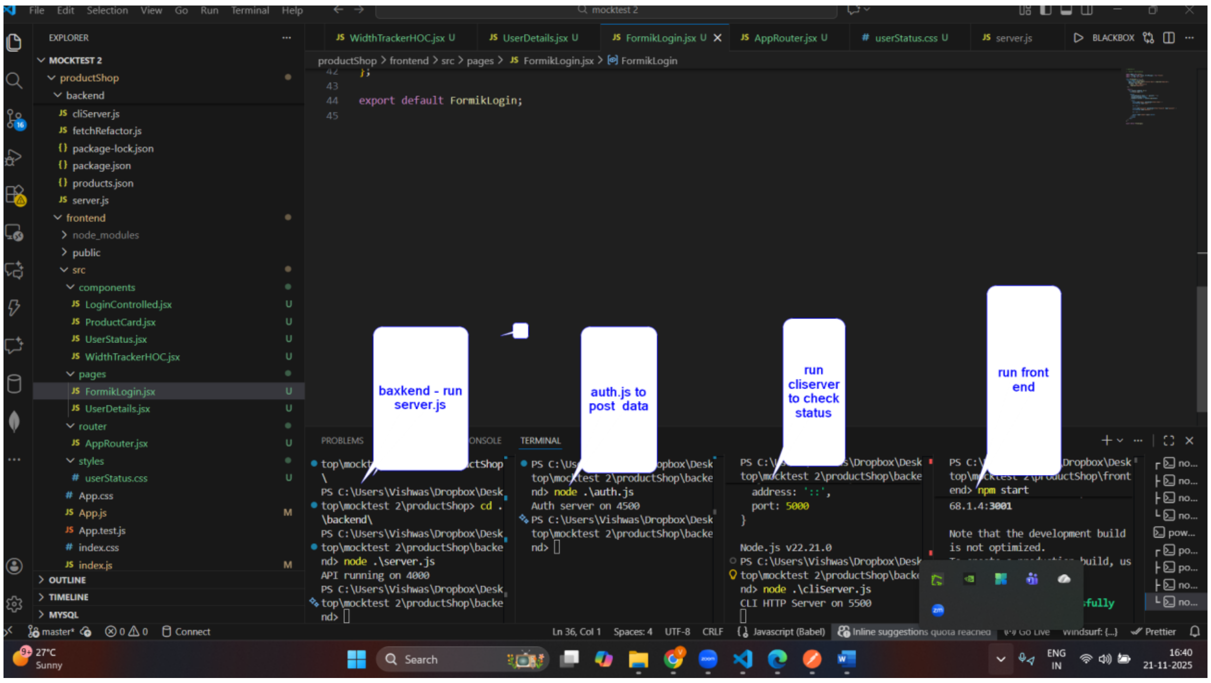
Preptest – 2

Dependencies for backend-npm init -y , npm install express express-validator jsonwebtoken

Run node – server.js , server.cliserver.js(check status) , auth.js(for post method)

Dependencies for frontend - npm install react-router-dom formik yup prop-types

Run – npm start



```
42  };
43
44  export default FormikLogin;
45
```

```
top\mockt              ctShop
\
PS C:\Users\Vishwas\Dropbox\Desk
top\mocktest 2\productShop> cd .
\backend\
PS C:\Users\Vishwas\Dropbox\Desk
top\mocktest 2\productShop\backe
nd> node .\server.js
API running on 4000
PS C:\Users\Vishwas\Dropbox\Desk
top\mocktest 2\productShop\backe
nd>
```

```
PS C:\Us              opbox\Desk
top\mocktest 2\productShop\backe
nd> node .\auth.js
Auth server on 4500
PS C:\Users\Vishwas\Dropbox\Desk
top\mocktest 2\productShop\backe
nd>
```

```
PS C:\              s\Dropbox\Desk
top\mocktest 2\productShop\backe
  address: '::',
  port: 5000
}

Node.js v22.21.0
PS C:\Users\Vishwas\Dropbox\Desk
top\mocktest 2\productShop\backe
nd> node .\cliServer.js
CLI HTTP Server on 5500
```

```
PS C:\              Dropbox\Desk
top\m               uctShop\front
end> npm start
68.1.4:3001

Note that the development build
is not optimized.
                         build, us
                         fully
```

Q1. React Basics (JSX, Components, Props)

// question 1

//

// React Basics (JSX, Components, Props)

import React from "react";

```
// A simple functional component receiving props
const ProductCard = ({ title, price, discount }) => {
  const finalPrice = price - discount;

  return (
    <div style={{ border: "1px solid #ccc", padding: 16, margin: 10 }}>
      <h3>{title}</h3>
      <p>Original Price: ₹{price}</p>
      <p>Discount: ₹{discount}</p>
      <h4>Final Price: ₹{finalPrice}</h4>
    </div>
  );
};

export default ProductCard;
```

Q2. React State + Controlled and Uncontrolled Components

```
// question 2
//
// React State + Controlled and Uncontrolled Components

import React, { useRef, useState } from "react";
```

```
const LoginControlled = () => {
  const [username, setUsername] = useState(""); // controlled
  const passwordRef = useRef(); // uncontrolled

  const handleSubmit = (e) => {
    e.preventDefault();
    console.log("Username:", username);
    console.log("Password:", passwordRef.current.value);
  };

  return (
    <form onSubmit={handleSubmit} style={{ margin: 20 }}>
      <h3>Login Form</h3>

      <input
        type="text"
        placeholder="Enter Username"
        value={username}
        onChange={(e) => setUsername(e.target.value)}
      />

      <br /><br />
      <input type="password" placeholder="Enter Password"
ref={passwordRef} />

      <br /><br />

      <button type="submit">Login</button>
    </form>
  );
};

export default LoginControlled;
```

Q3. React Class Component, Lifecycle, PropTypes, Styling

```
// question 3

//React Class Component, Lifecycle, PropTypes, Styling

import React, { Component } from "react";
import PropTypes from "prop-types";
import "../styles/userStatus.css";

class UserStatus extends Component {
  constructor(props) {
    super(props);
    this.state = {
      status: "Fetching user status.",
    };
  }

  componentDidMount() {
    setTimeout(() => {
      this.setState({ status: "Active User" });
    }, 2000);
  }

  render() {
    return (
      <div className="userBox">
        <p>User ID: {this.props.userId}</p>
        <h3>Status: {this.state.status}</h3>
      </div>
    );
  }
}

UserStatus.propTypes = {
  userId: PropTypes.number.isRequired,
};

export default UserStatus;
```

Q4. React Router + API Integration

```
// question 4

// React Router + API Integration
import React, { useEffect, useState } from "react";
import { useParams } from "react-router-dom";

const UserDetails = () => {
  const { id } = useParams();
  const [user, setUser] = useState(null);

  useEffect(() => {
    fetch(`http://localhost:4000/users/${id}`)
      .then((res) => res.json())
      .then((data) => setUser(data));
  }, [id]);

  if (!user) return <h3>Loading..</h3>;

  return (
    <div style={{ padding: 20 }}>
      <h2>User Details</h2>
      <p>ID: {user.id}</p>
      <p>Name: {user.name}</p>
    </div>
  );
};

export default UserDetails;
```

Q5. Reusability Using HOC or Render Props

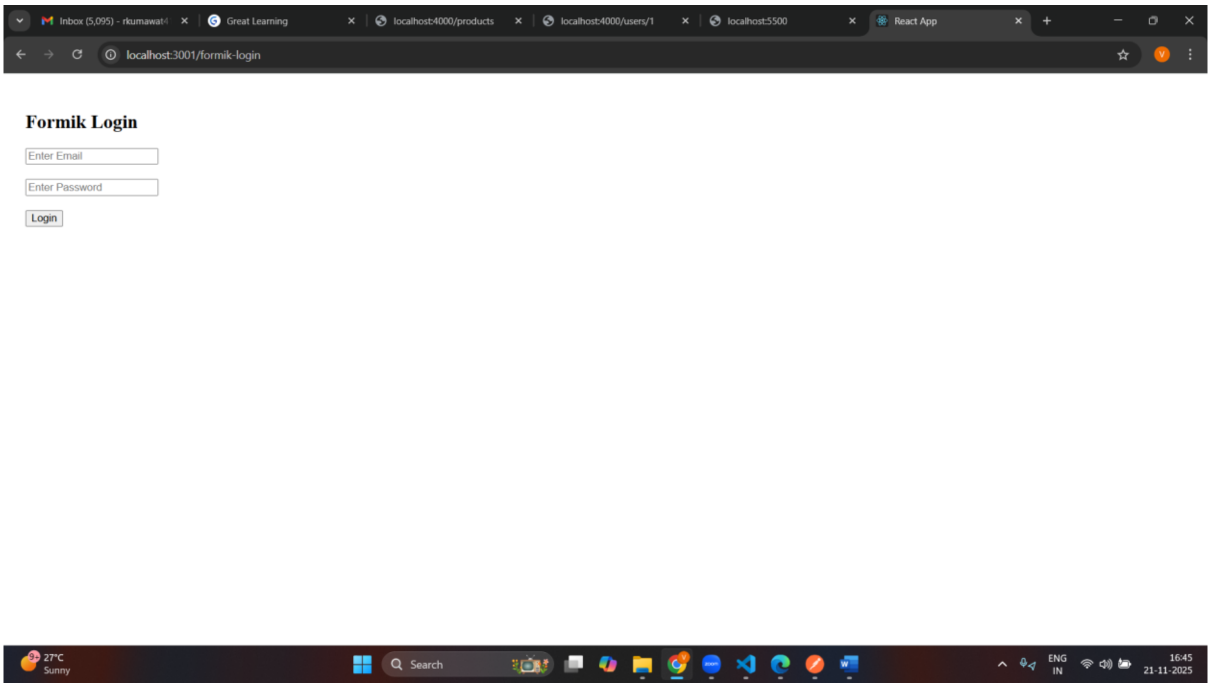
```
// question 5

// Reusability Using HOC or Render Props
```

```
import React, { useEffect, useState } from "react";

// Higher Order Component
const withWindowWidth = (WrappedComponent) => {
  return () => {
    const [width, setWidth] = useState(window.innerWidth);

    useEffect(() => {
      const handleResize = () => setWidth(window.innerWidth);

      window.addEventListener("resize", handleResize);
      return () => window.removeEventListener("resize",
handleResize);
    }, []);

    return <WrappedComponent windowWidth={width} />;
  };
};

export default withWindowWidth;
```

Q6. Formik + Yup Validation

```
// question 6

//  Formik + Yup Validation

import React from "react";
import { Formik, Form, Field, ErrorMessage } from "formik";
import * as Yup from "yup";

const FormikLogin = () => {
  const schema = Yup.object({
    email: Yup.string().email("Invalid
Email").required("Required"),
    password: Yup.string()
      .min(6, "Min 6 characters")
      .required("Required"),
  });

  return (
    <div style={{ padding: 20 }}>
      <h2>Formik Login</h2>
      <Formik
        initialValues={{ email: "", password: "" }}
        validationSchema={schema}
        onSubmit={(values) => console.log(values)}
      >
        <Form>
          <Field name="email" placeholder="Enter Email" />
          <ErrorMessage name="email" />

          <br /><br />

          <Field name="password" placeholder="Enter Password"
type="password" />
          <ErrorMessage name="password" />

          <br /><br />

          <button type="submit">Login</button>
        </Form>
      </Formik>
    </div>
```

```
  );
};

export default FormikLogin;
```



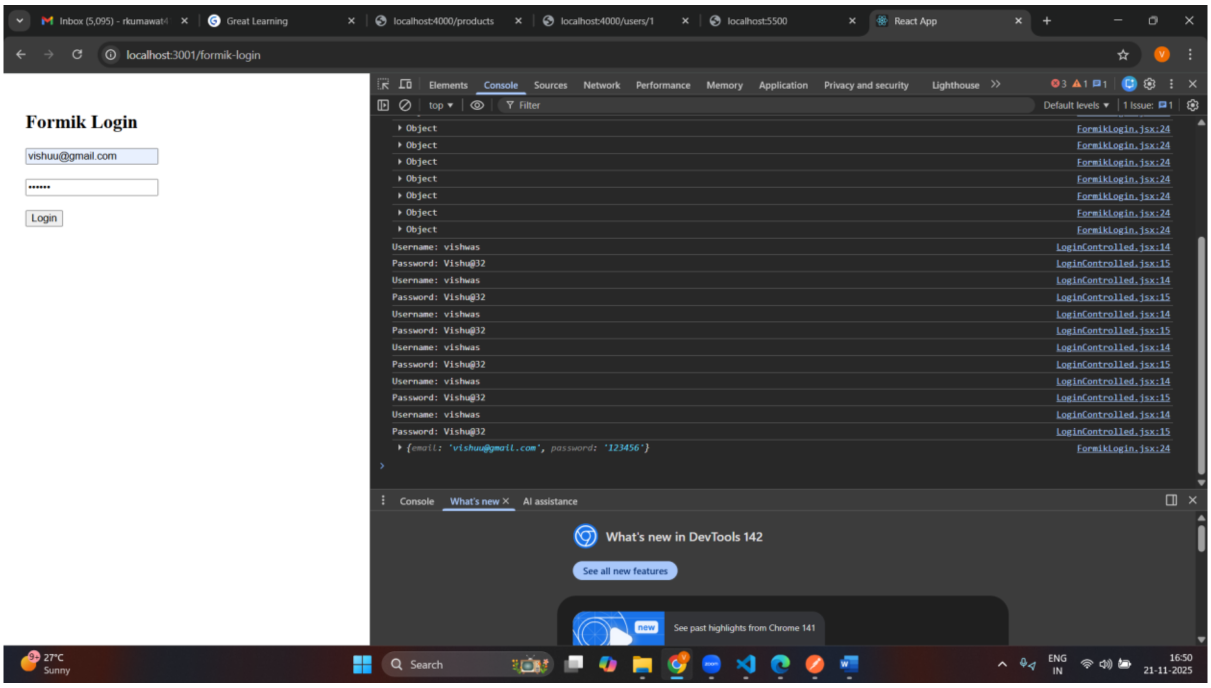
localhost:3001/formik-login

Formik Login

Enter Email

Enter Password

Login



localhost:3001/formik-login

Formik Login

vishuu@gmail.com

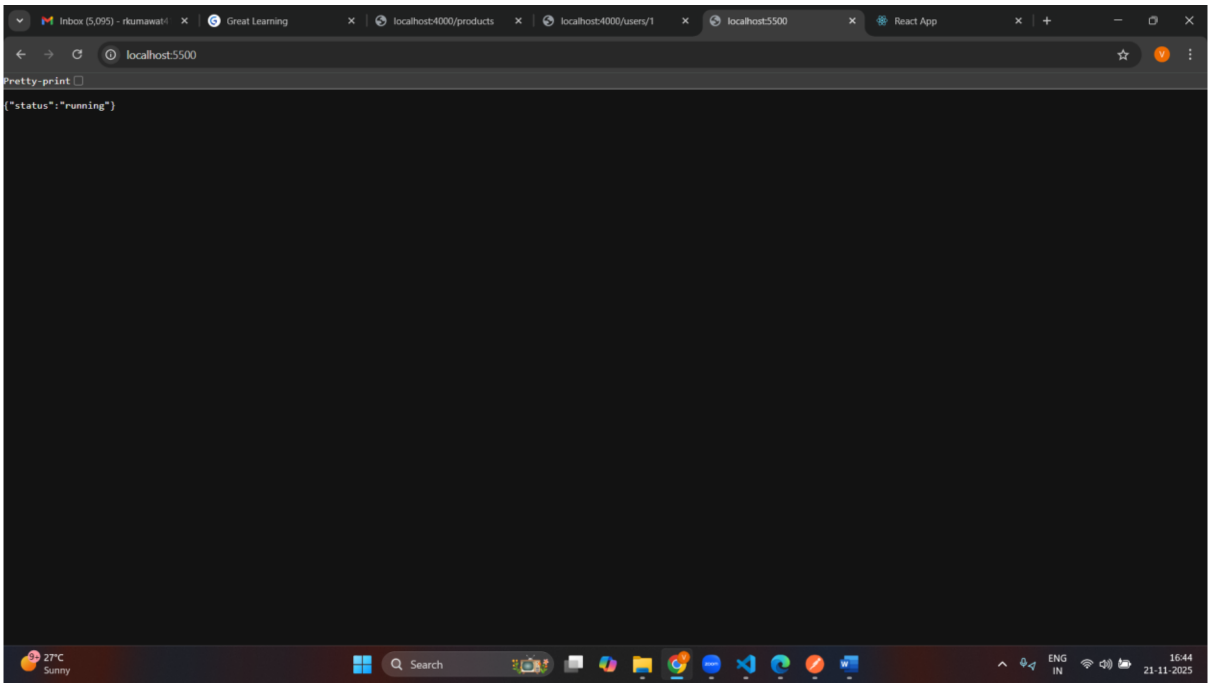
••••••

Login

```
▸ Object                                          FormikLogin.jsx:24
▸ Object                                          FormikLogin.jsx:24
▸ Object                                          FormikLogin.jsx:24
▸ Object                                          FormikLogin.jsx:24
▸ Object                                          FormikLogin.jsx:24
▸ Object                                          FormikLogin.jsx:24
▸ Object                                          FormikLogin.jsx:24
Username: vishwas                                 LoginControlled.jsx:14
Password: Vishu@32                                LoginControlled.jsx:15
Username: vishwas                                 LoginControlled.jsx:14
Password: Vishu@32                                LoginControlled.jsx:15
Username: vishwas                                 LoginControlled.jsx:14
Password: Vishu@32                                LoginControlled.jsx:15
Username: vishwas                                 LoginControlled.jsx:14
Password: Vishu@32                                LoginControlled.jsx:15
Username: vishwas                                 LoginControlled.jsx:14
Password: Vishu@32                                LoginControlled.jsx:15
Username: vishwas                                 LoginControlled.jsx:14
Password: Vishu@32                                LoginControlled.jsx:15
▸ {email: 'vishuu@gmail.com', password: '123456'} FormikLogin.jsx:24
>
```

Q7. Node.js Core Modules

```
// question 7

// Node.js Core Modules

const fs = require("fs");
const path = require("path");
const http = require("http");

// folder-safe path
const logPath = path.join(__dirname, "logs");

// create folder if not exists
if (!fs.existsSync(logPath)) {
  fs.mkdirSync(logPath);
}

// write log
fs.writeFileSync(path.join(logPath, "app.log"), "App started");

// simple HTTP server
const server = http.createServer((req, res) => {
  res.writeHead(200, { "Content-Type": "application/json" });
  res.end(JSON.stringify({ status: "running" }));
});

server.listen(5000, () => console.log("CLI HTTP Server on 5000"));
```



```
{"status":"running"}
```

Q8. Asynchronous JavaScript (Callbacks → Promise → Async/Await)

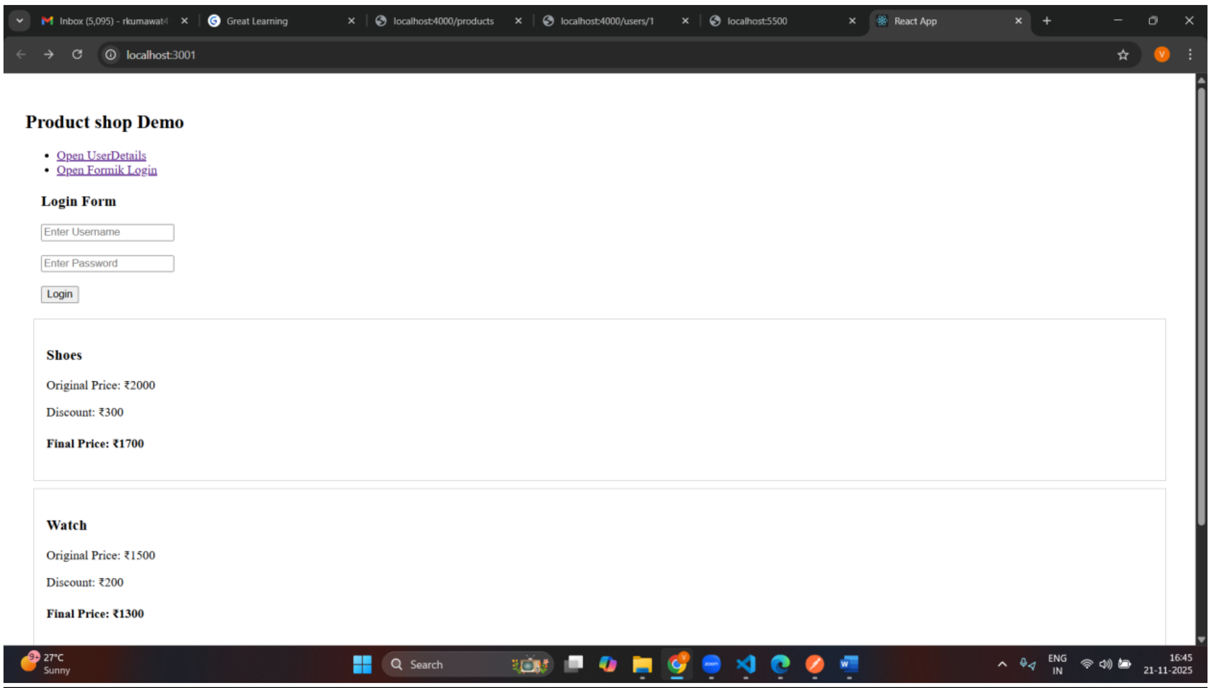
```
// question 8

// Asynchronous JavaScript (Callbacks → Promise → Async/Await)
// OLD CALLBACK VERSION
function fetchDataCallback(cb) {
  setTimeout(() => {
    cb(null, { id: 1, name: "Node.js" });
  }, 1000);
}

// PROMISE VERSION
function fetchDataPromise() {
  return new Promise((resolve) => {
    setTimeout(() => {
      resolve({ id: 1, name: "Node.js" });
    }, 1000);
  });
}

// ASYNC AWAIT VERSION
async function fetchDataAsync() {
  const data = await fetchDataPromise();
  console.log("Async/Await:", data);
}

// Outputs
fetchDataCallback((_, data) => console.log("Callback:", data));
fetchDataPromise().then((d) => console.log("Promise:", d));
fetchDataAsync();
```

Q9. Express Routing + Middleware + Validation

```
  // quetion 9
  // Express Routing + Middleware + Validation



const express = require("express");
const { body, validationResult } = require("express-validator");

const app = express();
app.use(express.json());

// global middleware
app.use((req, res, next) => {
  console.log(`${req.method} ${req.url}`);
  next();
});
```

```
// GET
app.get("/products", (req, res) => {
  res.json([
    { id: 1, name: "Shoes", price: 2000 },
    { id: 2, name: "Watch", price: 1500 },
  ]);
});

// POST with validation
app.post(
  "/products",
  [
    body("name").notEmpty(),
    body("price").isNumeric(),
  ],
  (req, res) => {
    const errors = validationResult(req);
    if (!errors.isEmpty()) return res.status(400).json(errors);

    res.json({ message: "Product added", data: req.body });
  }
);
```

```
// ADD this in server.js below products routes
app.get("/users/:id", (req, res) => {
  const { id } = req.params;

  // Demo hardcoded data
  const user = {
    id,
    name: `User ${id}`,
    email: `user${id}@test.com`,
  };

  res.json(user);
});

app.listen(4000, () => console.log("API running on 4000"));
```



Inbox (5,095) - rkumawat4 | Great Learning | localhost:4000/products | localhost:4000/users/1 | localhost:5500 | React App

localhost:3001

Product shop Demo

- Open UserDetails
- Open Formik Login

Login Form

Enter Username

Enter Password

Login

Shoes

Original Price: ₹2000

Discount: ₹300

Final Price: ₹1700

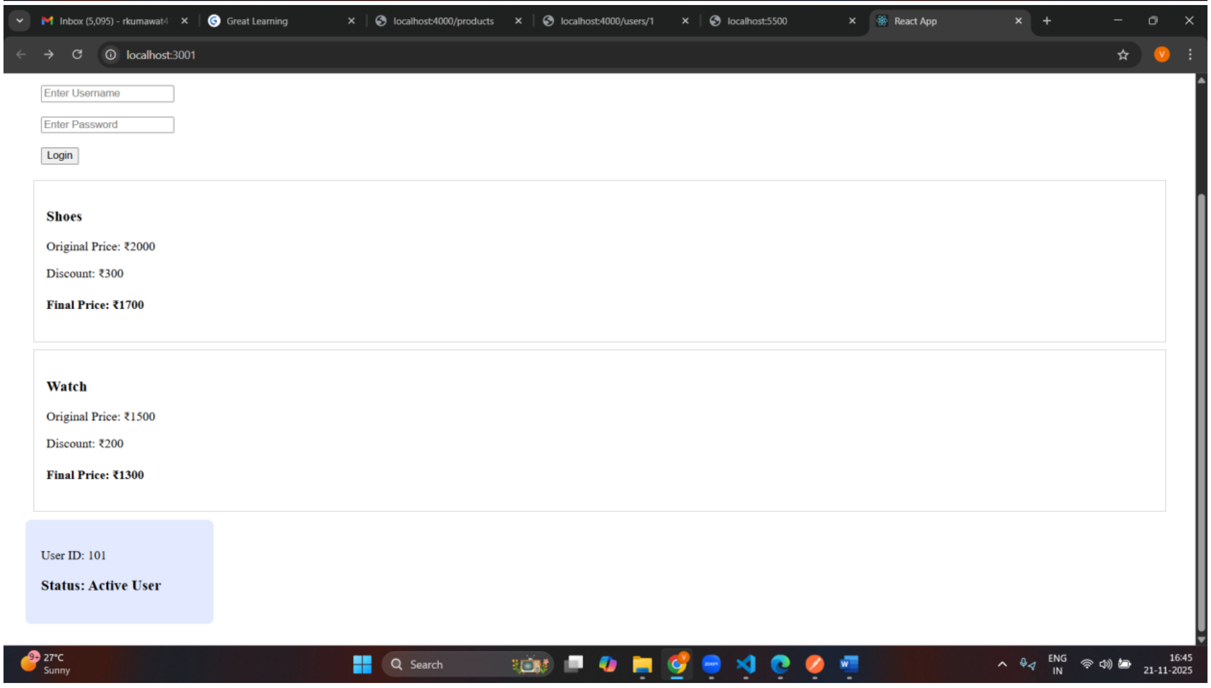
Watch

Original Price: ₹1500

Discount: ₹200

Final Price: ₹1300

27°C Sunny | Search | ENG IN | 16:45 21-11-2025



Inbox (5,095) - rkumawat4 | Great Learning | localhost:4000/products | localhost:4000/users/1 | localhost:5500 | React App

localhost:3001

Enter Username

Enter Password

Login

Shoes

Original Price: ₹2000

Discount: ₹300

Final Price: ₹1700

Watch

Original Price: ₹1500

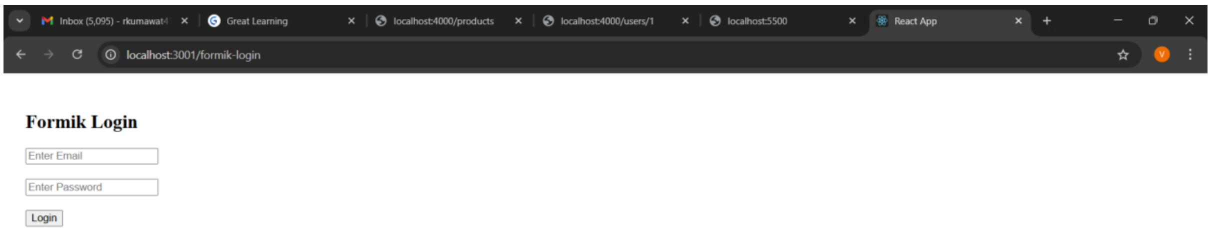
Discount: ₹200

Final Price: ₹1300

User ID: 101

Status: Active User

27°C Sunny | Search | ENG IN | 16:45 21-11-2025

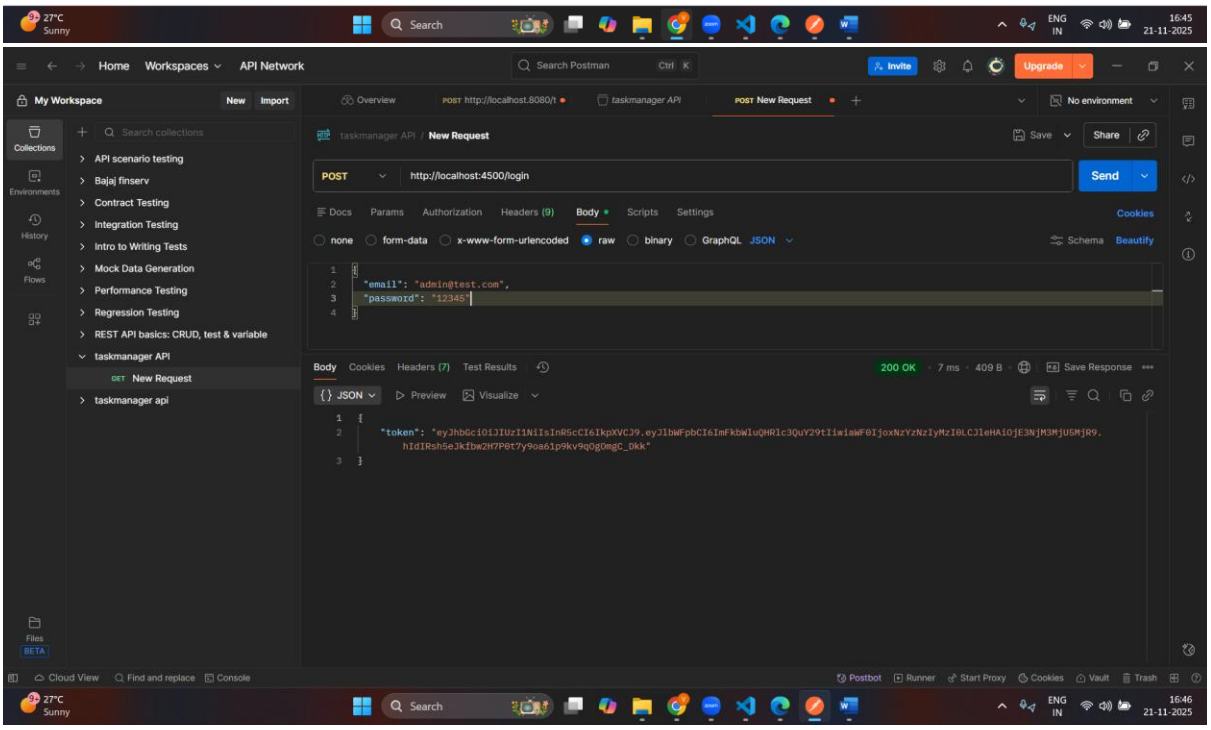


Formik Login

Enter Email

Enter Password

Login

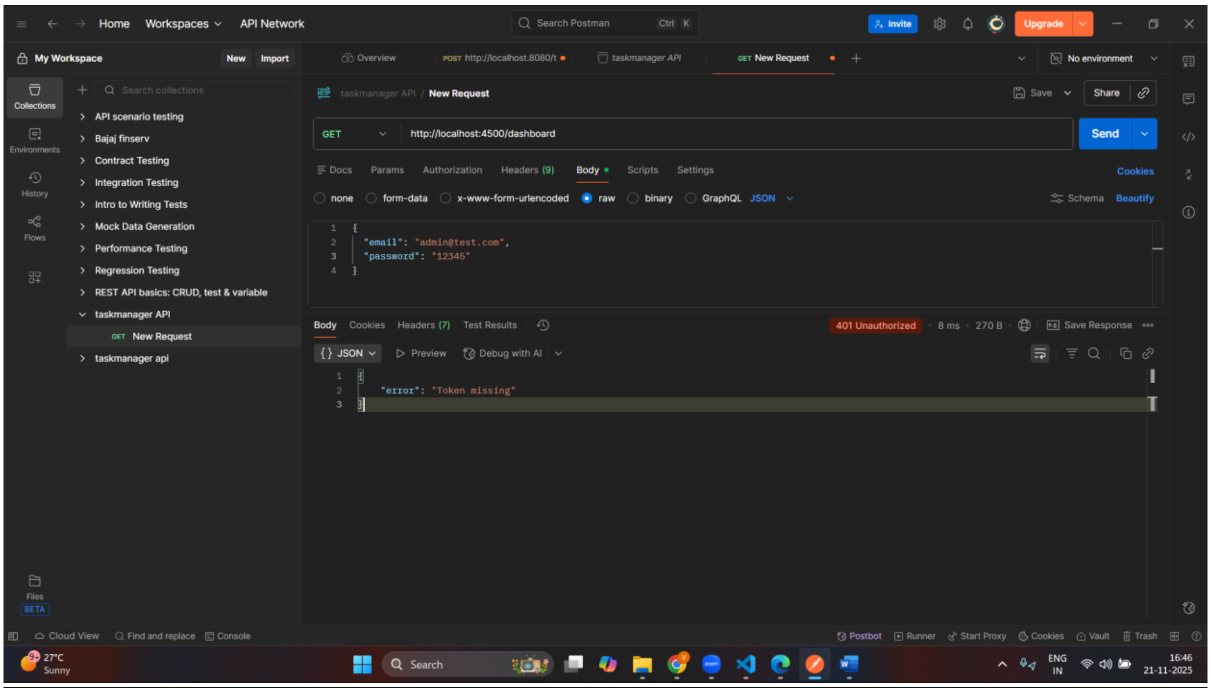


POST http://localhost:4500/login

```
{
"email": "admin@test.com",
"password": "12345"
}
```

200 OK · 7 ms · 409 B

```
{
    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJlbWFpbCI6ImFkbWluQHRlc3QuY29tIiwiaWF0IjoxNzYzNzIyMzI0LCJleHAiOjE3NjM3MjU5MjR9.
      hIdIRsh5eJkfbw2H7P0t7y9oa61p9kv9qOgOmgC_Dkk"
}
```



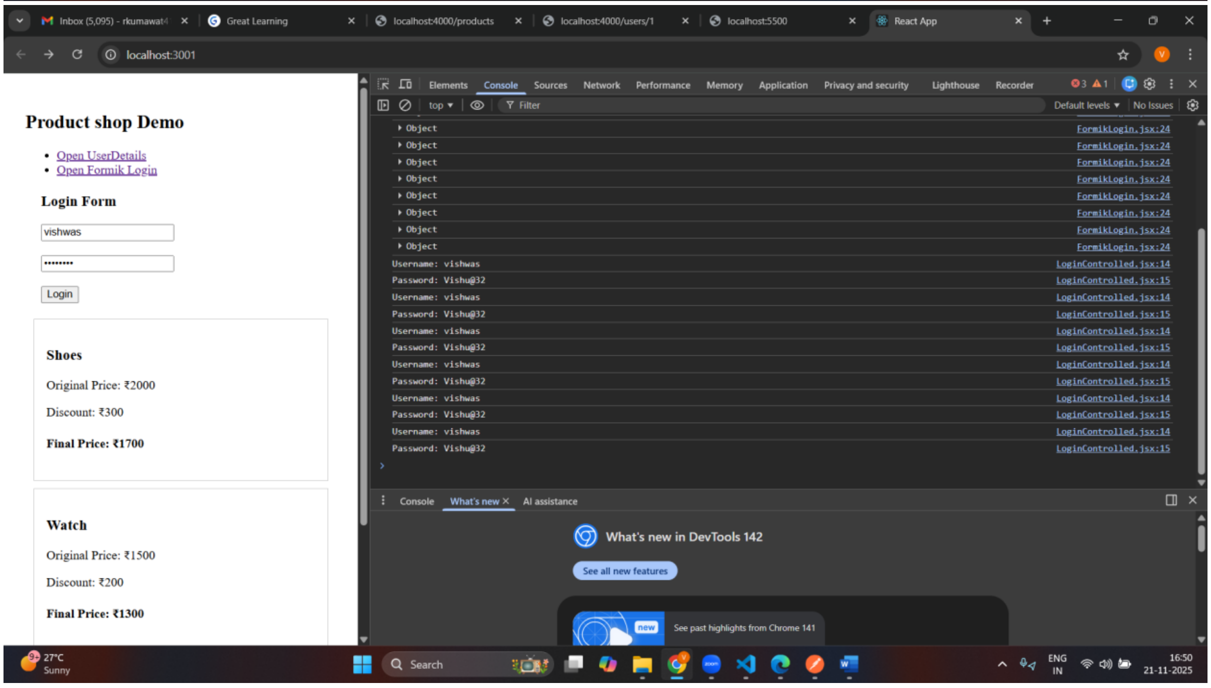
taskmanager API / New Request

GET http://localhost:4500/dashboard

```
{
  "email": "admin@test.com",
  "password": "12345"
}
```

401 Unauthorized · 8 ms · 270 B

```
{
    "error": "Token missing"
}
```



localhost:3001

Product shop Demo

- Open UserDetails
- Open Formik Login

Login Form

vishwas

Login

Shoes

Original Price: ₹2000

Discount: ₹300

Final Price: ₹1700

Watch

Original Price: ₹1500

Discount: ₹200

Final Price: ₹1300

```
▸ Object                                    FormikLogin.jsx:24
▸ Object                                    FormikLogin.jsx:24
▸ Object                                    FormikLogin.jsx:24
▸ Object                                    FormikLogin.jsx:24
▸ Object                                    FormikLogin.jsx:24
▸ Object                                    FormikLogin.jsx:24
▸ Object                                    FormikLogin.jsx:24
▸ Object                                    FormikLogin.jsx:24
Username: vishwas                           LoginControlled.jsx:14
Password: Vishu@32                          LoginControlled.jsx:15
Username: vishwas                           LoginControlled.jsx:14
Password: Vishu@32                          LoginControlled.jsx:15
Username: vishwas                           LoginControlled.jsx:14
Password: Vishu@32                          LoginControlled.jsx:15
Username: vishwas                           LoginControlled.jsx:14
Password: Vishu@32                          LoginControlled.jsx:15
Username: vishwas                           LoginControlled.jsx:14
Password: Vishu@32                          LoginControlled.jsx:15
Username: vishwas                           LoginControlled.jsx:14
Password: Vishu@32                          LoginControlled.jsx:15
```

What's new in DevTools 142

Q10. REST API + JWT Authentication

```
// question 10

// REST API + JWT Authentication


const express = require("express");
const jwt = require("jsonwebtoken");

const app = express();
app.use(express.json());

const SECRET = "MYSECRET123";

// LOGIN route
app.post("/login", (req, res) => {
  const { email, password } = req.body;

  if (email === "admin@test.com" && password === "Admin@345") {
    const token = jwt.sign({ email }, SECRET, { expiresIn: "1h" });
    return res.json({ token });
  }
  res.status(401).json({ error: "Invalid Credentials" });
});

// MIDDLEWARE
function authMiddleware(req, res, next) {
  const token = req.headers.authorization?.split(" ")[1];

  if (!token) return res.status(401).json({ error: "Token missing" });

  try {
    jwt.verify(token, SECRET);
    next();
  } catch {
    res.status(401).json({ error: "Invalid token" });
  }
}

// PROTECTED
app.get("/dashboard", authMiddleware, (req, res) => {
  res.json({ message: "Welcome to Dashboard" });
});

app.listen(4500, () => console.log("Auth server on 4500"));
```