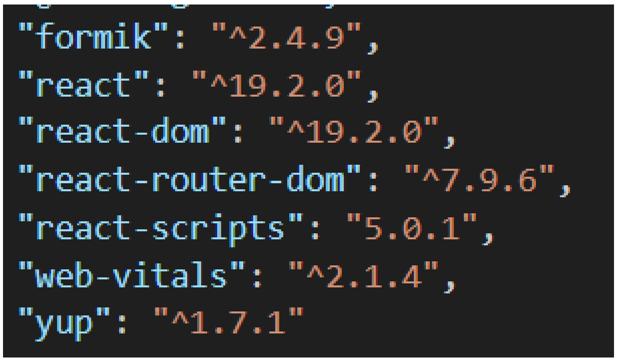
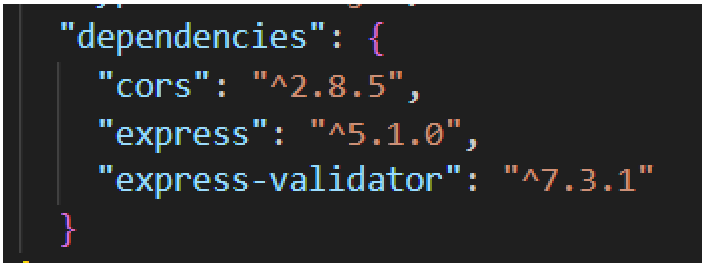
Mock Assessment 2

Installed dependencies



```
"formik": "^2.4.9",
"react": "^19.2.0",
"react-dom": "^19.2.0",
"react-router-dom": "^7.9.6",
"react-scripts": "5.0.1",
"web-vitals": "^2.1.4",
"yup": "^1.7.1"
```



```
  "dependencies": {
    "cors": "^2.8.5",
    "express": "^5.1.0",
    "express-validator": "^7.3.1"
  }
```

Q1. React Basics (JSX, Components, Props)
User Story
As a Product Page Developer, I need a reusable ProductCard component that receives a product name, price, and discount as props so that the final discounted amount can be displayed on the ShopNow listing page.
Acceptance Criteria

- **Functional component**
- **Props: title, price, discount**
- **Final price = price − discount**
- **Parent component must pass props**
- **JSX must display all fields clearly**

App.js

```
import react from 'react';
import ProductCard from '../components/ProductCard';


function App() {
  const products = [
    { title: 'Product 1', price: 100, discount: 10 },
    { title: 'Product 2', price: 200, discount: 20 },
    { title: 'Product 3', price: 300, discount: 30 },
  ];
  return (
    <div className="App">
      <h1>Product List</h1>
      {products.map((product, index) => (
        <ProductCard
          key={index}
          title={product.title}
          price={product.price}
          discount={product.discount}
        />
      ))}
    </div>
  );
}
export default App;
```

components/ProductCard.js

```
import React from "react";
import PropTypes from "prop-types";
import "./ProductCard.css";

const ProductCard = ({ title, price, discount }) => {
  const numericPrice = Number(price) || 0;
  const numericDiscount = Number(discount) || 0;
  const finalPrice = numericPrice - numericDiscount;

  return (
    <article className="product-card">
      <h3 className="product-card__title">{title}</h3>

      <div className="product-card__prices">
        <div className="product-card__price">
          <span className="label">Price</span>
          <span className="value">₹{numericPrice.toFixed(2)}</span>
        </div>

        <div className="product-card__discount">
          <span className="label">Discount</span>
          <span className="value">₹{numericDiscount.toFixed(2)}</span>
        </div>
      </div>

      <div className="product-card__final">
        <span className="label">Final Price</span>
        <span className="value">₹{finalPrice.toFixed(2)}</span>
      </div>
    </article>
  );
};

ProductCard.propTypes = {
  title: PropTypes.string.isRequired,
  price: PropTypes.oneOfType([PropTypes.string,
PropTypes.number]).isRequired,
  discount: PropTypes.oneOfType([PropTypes.string,
PropTypes.number]).isRequired,
};

export default ProductCard;
```

components/ProductCard.css

```
.product-card {
  border: 1px solid #e0e0e0;
  border-radius: 10px;
  padding: 16px;
  width: 260px;
  box-shadow: 0 6px 18px rgba(0,0,0,0.04);
  display: flex;
  flex-direction: column;
  gap: 12px;
  background: #ffffff;
}

.product-card__title {
  font-size: 1.05rem;
  margin: 0;
  color: #111827;
}

.product-card__prices {
  display: flex;
  justify-content: space-between;
  gap: 12px;
  align-items: center;
```

```css
}

.product-card__price,
.product-card__discount {
  display: flex;
  flex-direction: column;
  font-size: 0.9rem;
}

.label {
  font-size: 0.75rem;
  color: #6b7280;
}

.value {
  font-weight: 600;
  font-size: 1rem;
  color: #111827;
}

.product-card__final {
  padding-top: 8px;
  border-top: 1px dashed #e5e7eb;
  display: flex;
  justify-content: space-between;
  align-items: center;
  font-size: 1rem;
}
```

pages/ShopNow.js

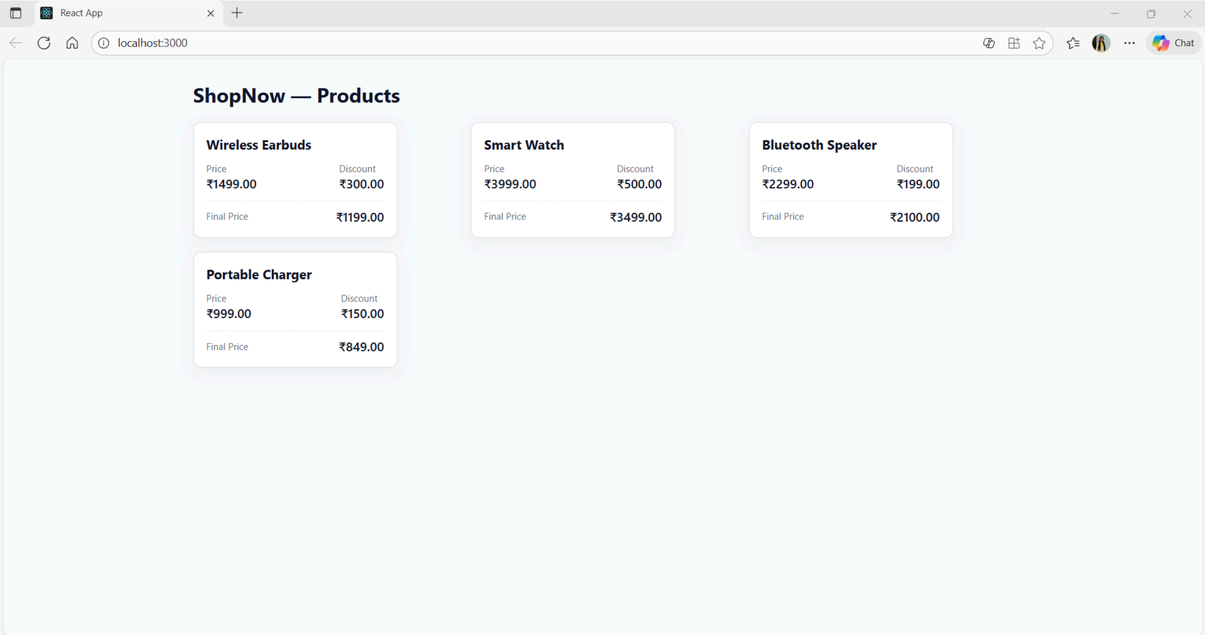
```javascript
import React from "react";
import ProductCard from "../components/ProductCard";
import "./ShopNow.css";

const PRODUCTS = [
  { id: 1, title: "Wireless Earbuds", price: 1499, discount: 300 },
  { id: 2, title: "Smart Watch", price: 3999, discount: 500 },
  { id: 3, title: "Bluetooth Speaker", price: 2299, discount: 199 },
  { id: 4, title: "Portable Charger", price: 999, discount: 150 }
];
```

```
const ShopNow = () => {
  return (
    <main className="shop-container">
      <h1 className="shop-title">ShopNow — Products</h1>

      <section className="products-grid" aria-label="products list">
        {PRODUCTS.map((p) => (
          <ProductCard key={p.id} title={p.title} price={p.price}
discount={p.discount} />
        ))}
      </section>
    </main>
  );
};

export default ShopNow;
```

pages/ShopNow.css

```
.shop-container {
  padding: 28px;
  max-width: 1100px;
  margin: 0 auto;
}

.shop-title {
  margin: 0 0 18px 0;
  font-size: 1.6rem;
  color: #0f172a;
}

.products-grid {
  display: grid;
  grid-template-columns: repeat(auto-fill, minmax(260px, 1fr));
  gap: 18px;
}
```

Screenshots:



ShopNow — Products

Wireless Earbuds

Price ₹1499.00 Discount ₹300.00

Final Price ₹1199.00

Smart Watch

Price ₹3999.00 Discount ₹500.00

Final Price ₹3499.00

Bluetooth Speaker

Price ₹2299.00 Discount ₹199.00

Final Price ₹2100.00

Portable Charger

Price ₹999.00 Discount ₹150.00

Final Price ₹849.00

Q2. React State + Controlled and Uncontrolled Components

User Story

As a Login Experience Engineer, I want the login form to capture the username through a controlled input while keeping the password field uncontrolled using useRef, so that the system can demonstrate both controlled and uncontrolled approaches.

Acceptance Criteria

- **Username: controlled component**
- **Password: uncontrolled using ref**
- **On submit: log both values**
- **Clear and structured JSX form**

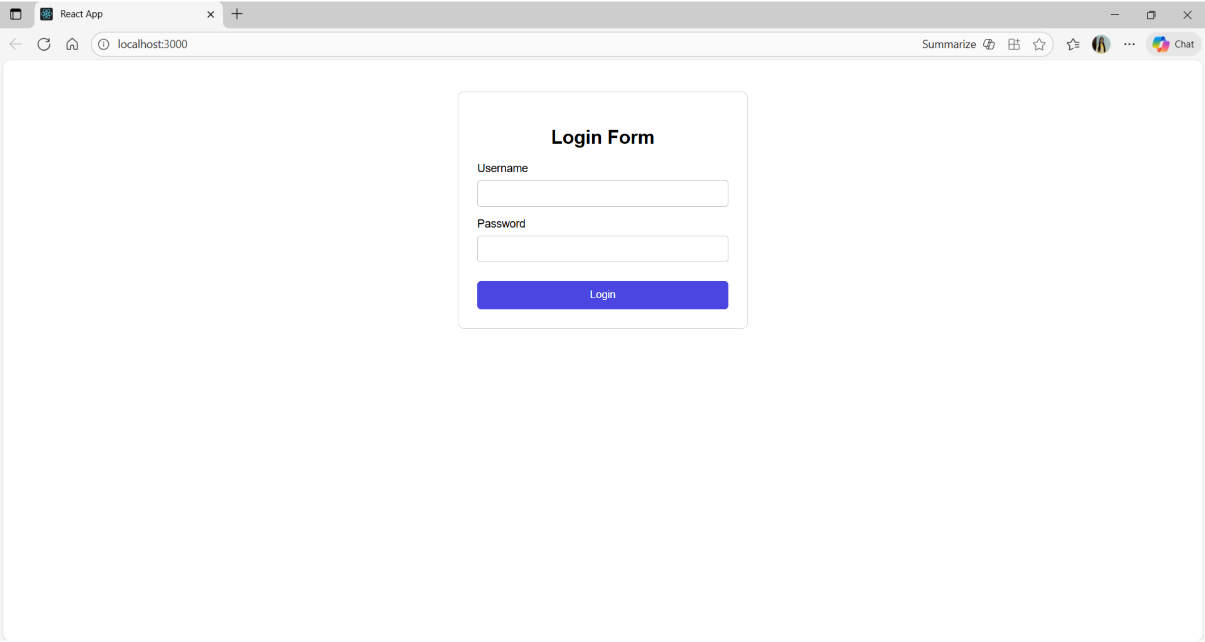
pages/LoginForm.js

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

const LoginForm = () => {
  const [username, setUsername] = useState("");
  const passwordRef = useRef(null);

  const handleSubmit = (event) => {
    event.preventDefault();

    const passwordValue = passwordRef.current.value;

    console.log("Username:", username);
    console.log("Password:", passwordValue);
  };

  return (
    <form onSubmit={handleSubmit} style={styles.form}>
      <h2 style={styles.title}>Login Form</h2>

      {/* Controlled Input */}
      <div style={styles.field}>
        <label style={styles.label}>Username</label>
        <input
          type="text"
          value={username}
          onChange={(e) => setUsername(e.target.value)}
          style={styles.input}
        />
      </div>

      {/* Uncontrolled Input */}
      <div style={styles.field}>
        <label style={styles.label}>Password</label>
        <input
          type="password"
          ref={passwordRef}
          style={styles.input}
        />
      </div>

      <button type="submit" style={styles.button}>
        Login
      </button>
    </form>
  );
};

const styles = {
  form: {
    width: "320px",
    margin: "40px auto",
    padding: "24px",
    border: "1px solid #ddd",
    borderRadius: "8px",
    fontFamily: "Arial",
  },
  title: {
    marginBottom: "18px",
    textAlign: "center",
  },
  field: {
    marginBottom: "14px",
    display: "flex",
    flexDirection: "column",
  },
  label: {
    marginBottom: "6px",
    fontSize: "14px",
  },
  input: {
    padding: "8px",
    borderRadius: "4px",
    border: "1px solid #ccc",
  },
  button: {
    marginTop: "10px",
    padding: "10px",
    width: "100%",
    background: "#4F46E5",
    color: "#fff",
    border: "none",
    borderRadius: "5px",
    cursor: "pointer",
  },
};

export default LoginForm;
```

App.js

```
import React from "react";
import LoginForm from "./pages/LoginForm";

function App() {
  return (
    <div>
      <LoginForm />
    </div>
  );
}

export default App;
```

Screenshots:
Login Page & Data in console

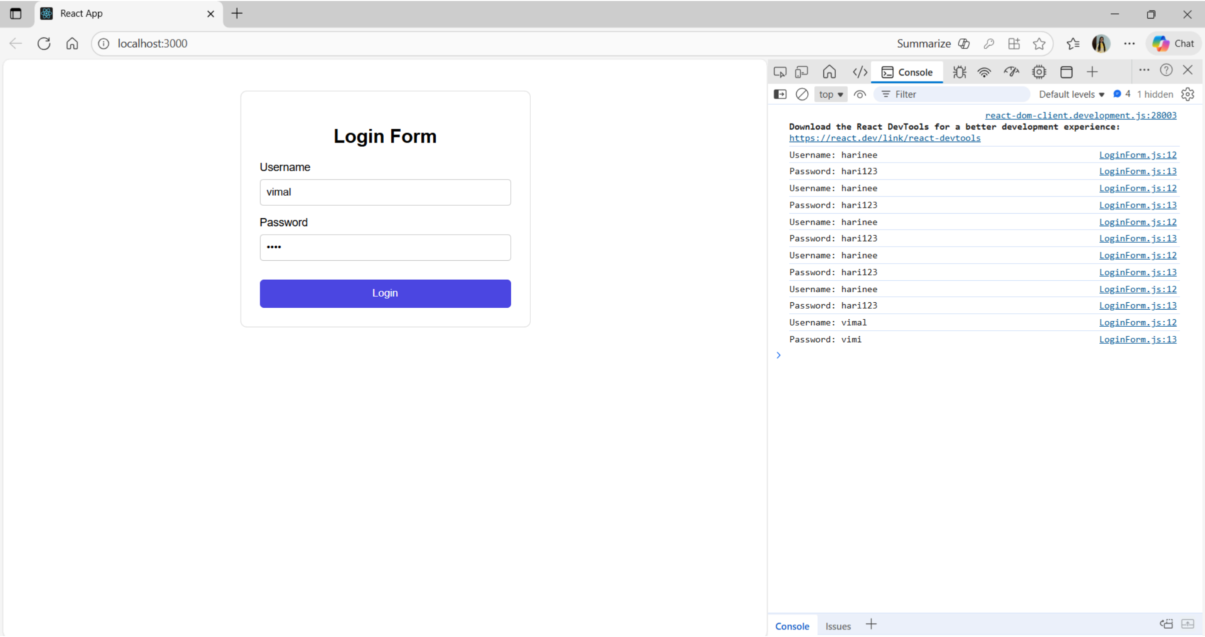


Login Form

Username

Password

Login



Login Form

Username: vimal

Password: ••••

Login

```
                                    react-dom-client.development.js:28003
Download the React DevTools for a better development experience:
https://react.dev/link/react-devtools
Username: harinee                                      LoginForm.js:12
Password: hari123                                      LoginForm.js:13
Username: harinee                                      LoginForm.js:12
Password: hari123                                      LoginForm.js:13
Username: harinee                                      LoginForm.js:12
Password: hari123                                      LoginForm.js:13
Username: harinee                                      LoginForm.js:12
Password: hari123                                      LoginForm.js:13
Username: harinee                                      LoginForm.js:12
Password: hari123                                      LoginForm.js:13
Username: vimal                                        LoginForm.js:12
Password: vimi                                         LoginForm.js:13
>
```

Q3. React Class Component, Lifecycle, PropTypes, Styling

User Story

As an Account Status Developer, I need a UserStatus class-based component that initially displays “Fetching user status…” and after two seconds shows “Active User,” ensuring that the component accepts a required numeric userId prop with styling applied.

Acceptance Criteria

- **Class component**
- **Lifecycle method updates message after 2 seconds**
- **Uses PropTypes for validation**
- **Includes basic CSS or inline styles**

pages/UserStatus.js

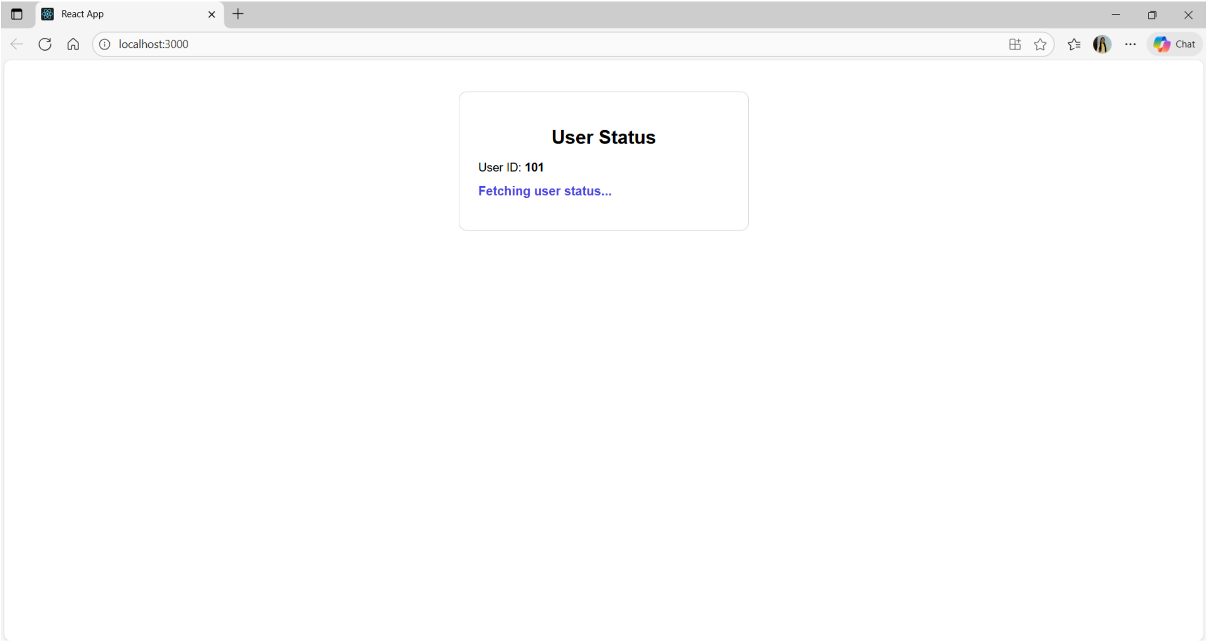
```
import React, { Component } from "react";
import PropTypes from "prop-types";

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

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

  render() {
    const { userId } = this.props;

    return (
      <div style={styles.container}>
        <h2 style={styles.heading}>User Status</h2>

        <p style={styles.info}>
          User ID: <span style={styles.bold}>{userId}</span>
        </p>

        <p style={styles.status}>{this.state.status}</p>
      </div>
    );
  }
}

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

const styles = {
  container: {
    margin: "40px auto",
    padding: "24px",
    width: "320px",
    borderRadius: "10px",
    border: "1px solid #ddd",
    fontFamily: "Arial",
  },
  heading: {
    marginBottom: "16px",
    textAlign: "center",
  },
  info: {
    marginBottom: "12px",
    fontSize: "15px",
  },
  bold: {
    fontWeight: "600",
  },
  status: {
    marginTop: "10px",
    fontSize: "16px",
    color: "#4F46E5",
    fontWeight: "bold",
  },
};

export default UserStatus;
```

App.js

```
import React from "react";
import UserStatus from "./pages/UserStatus";
function App() {
  return <UserStatus userId={101} />;
}
export default App;
```

Screenshots:

Fetching User status

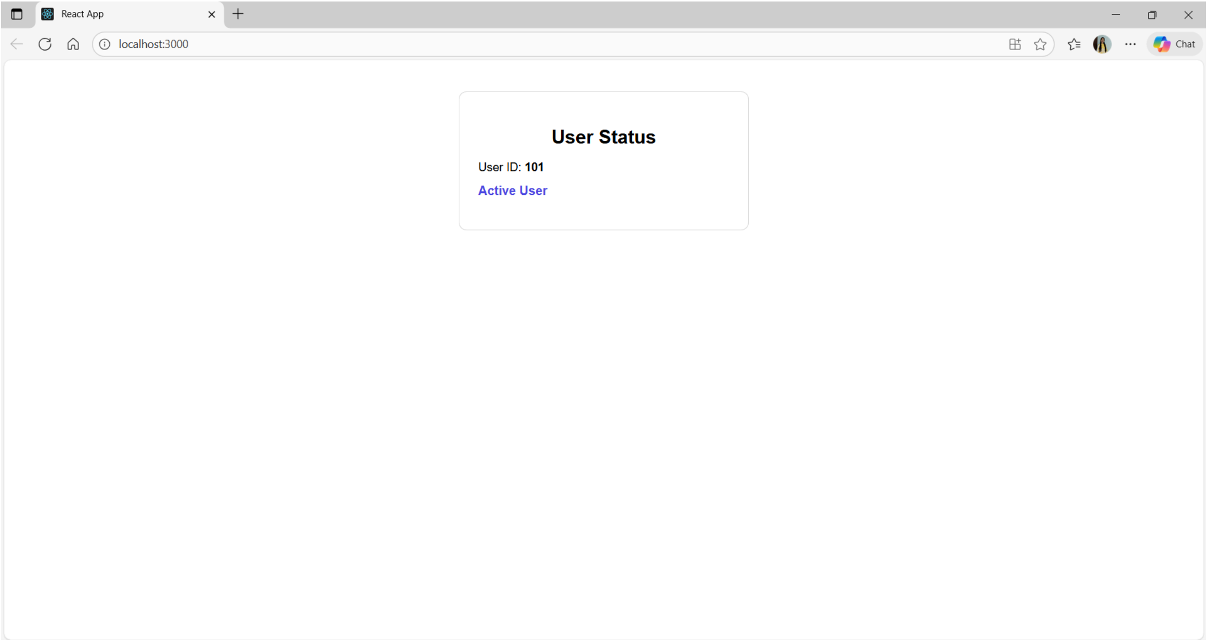


User Status

User ID: **101**

Fetching user status...

After 2 Second:



User Status

User ID: **101**

Active User

Q4. React Router + API Integration
User Story
As a User Details Page Developer, I need to configure routing so that /users/:id loads the UserDetails component which fetches user data from http://localhost:4000/users/:id, allowing individual user pages in the ShopNow admin panel.
Acceptance Criteria

- **React Router setup**
- **Parameter-based route**
- **Fetching user data using the id param**
- **Clean component structure**

pages/UserDetails.js

```
import { useParams } from "react-router-dom";
import { useEffect, useState } from "react";

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 <p>Loading...</p>;

  return (
    <div style={{ marginTop: 20 }}>
      <h3>User Details</h3>
      <p>ID: {user.id}</p>
      <p>Name: {user.name}</p>
      <p>Email: {user.email}</p>
    </div>
  );
};

export default UserDetails;
```

App.js

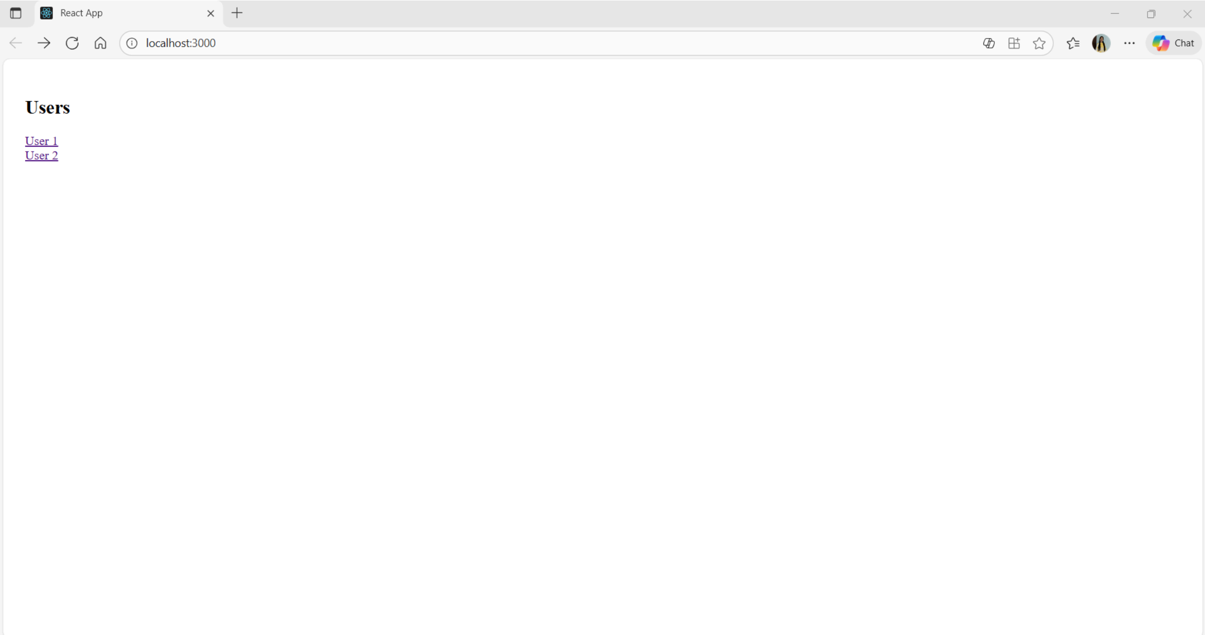
```
import { Routes, Route, Link } from "react-router-dom";
import UserDetails from "./pages/UserDetails";
```

```
function App() {
  return (
    <div style={{ padding: 20 }}>
      <h2>Users</h2>
      <Link to="/users/1">User 1</Link> <br />
      <Link to="/users/2">User 2</Link>

      <Routes>
        <Route path="/users/:id" element={<UserDetails />} />
      </Routes>
    </div>
  );
}

export default App;
```

backend/server.js

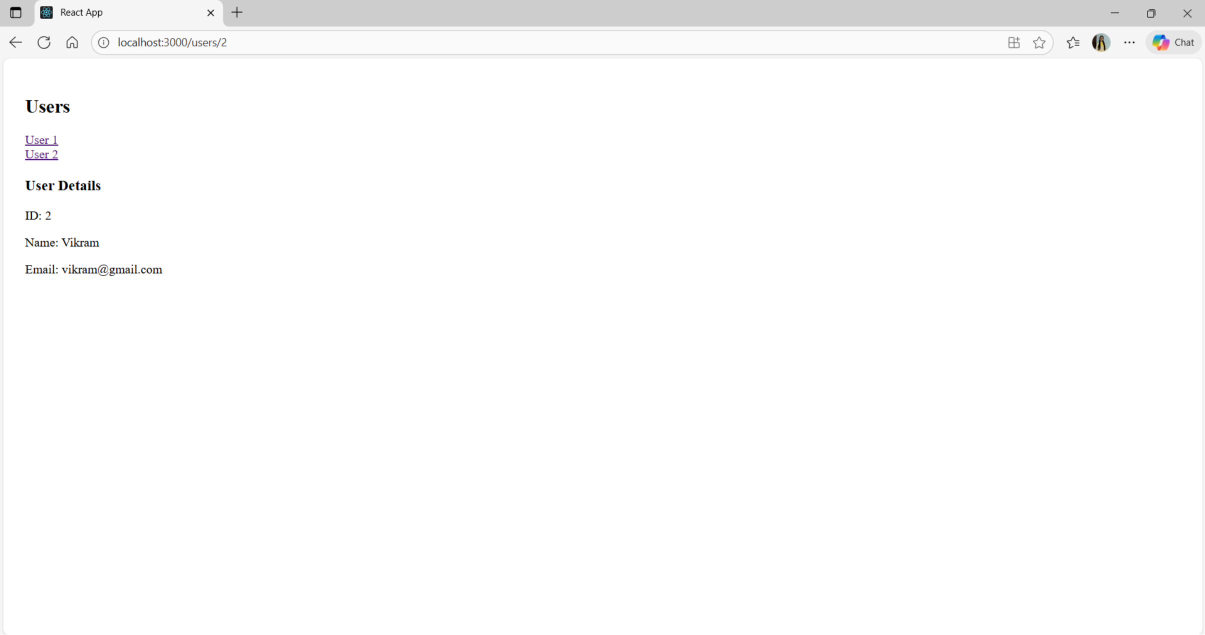
```
const express = require("express");
const cors = require("cors");
const app = express();

app.use(cors());

const USERS = [
  { id: 1, name: "Asha", email: "asha@gmail.com" },
  { id: 2, name: "Vikram", email: "vikram@gmail.com" }
];

app.get("/users/:id", (req, res) => {
  const id = Number(req.params.id);
  const user = USERS.find((u) => u.id === id);
  if (!user) return res.status(404).json({ error: "User not found" });
  res.json(user);
});

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

Screenshots:



Users

User 1
User 2



Users

User 1
User 2

User Details

ID: 2

Name: Vikram

Email: vikram@gmail.com

Q5. Reusability Using HOC or Render Props

User Story

As a Frontend Performance Engineer, I want a reusable utility (HOC or Render Props) that tracks the window width so that different components can display responsive UI behavior based on the same shared logic.

Acceptance Criteria

- **Reusable logic for windowWidth**
- **Implemented using Higher-Order Component or Render Props**
- **Demonstrate usage in at least one component**

hoc/withWindowWidth.js

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

const withWindowWidth = (WrappedComponent) => {
  return function () {
    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;
```

pages/ShowWidth.js

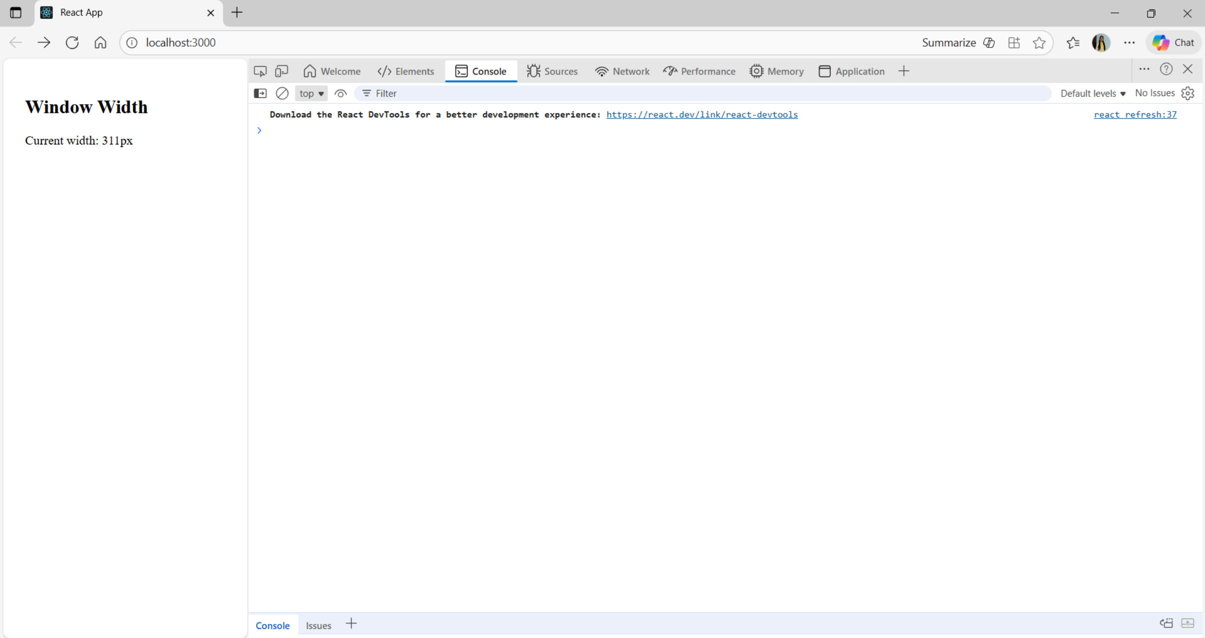
```
import React from "react";

const ShowWidth = ({ windowWidth }) => {
  return (
    <div style={{ padding: 20 }}>
      <h2>Window Width</h2>
      <p>Current width: {windowWidth}px</p>
    </div>
  );
};

export default ShowWidth;
```

App.js

```
import ShowWidth from "./pages/ShowWidth";
import withWindowWidth from "./hoc/withWindowWidth";

const WidthTracker = withWindowWidth(ShowWidth);
```

```
function App() {
  return (
    <div>
      <WidthTracker />
    </div>
  );
}

export default App;
```

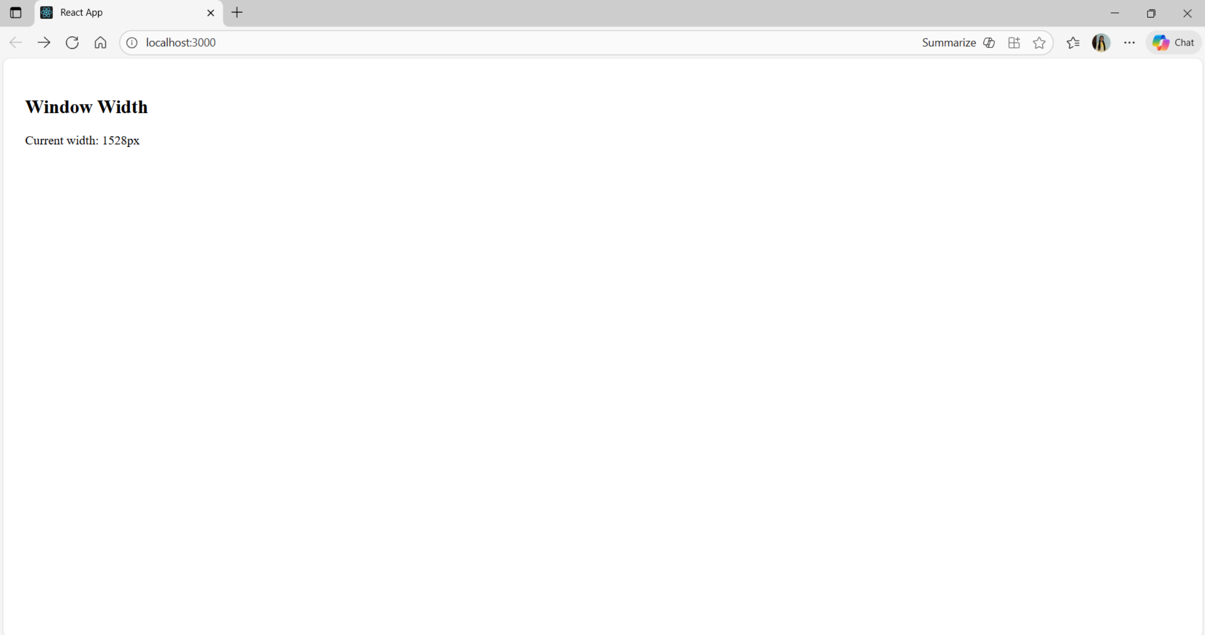
Screenshots:



Window Width

Current width: 311px

```
Download the React DevTools for a better development experience: https://react.dev/link/react-devtools        react_refresh:37
```



Window Width

Current width: 1528px

Q6. Formik + Yup Validation
User Story
As an Authentication Feature Developer, I must build a login form using Formik and Yup where the email field is validated for proper format and the password field ensures a minimum of six characters to support basic input validation in ShopNow.
Acceptance Criteria

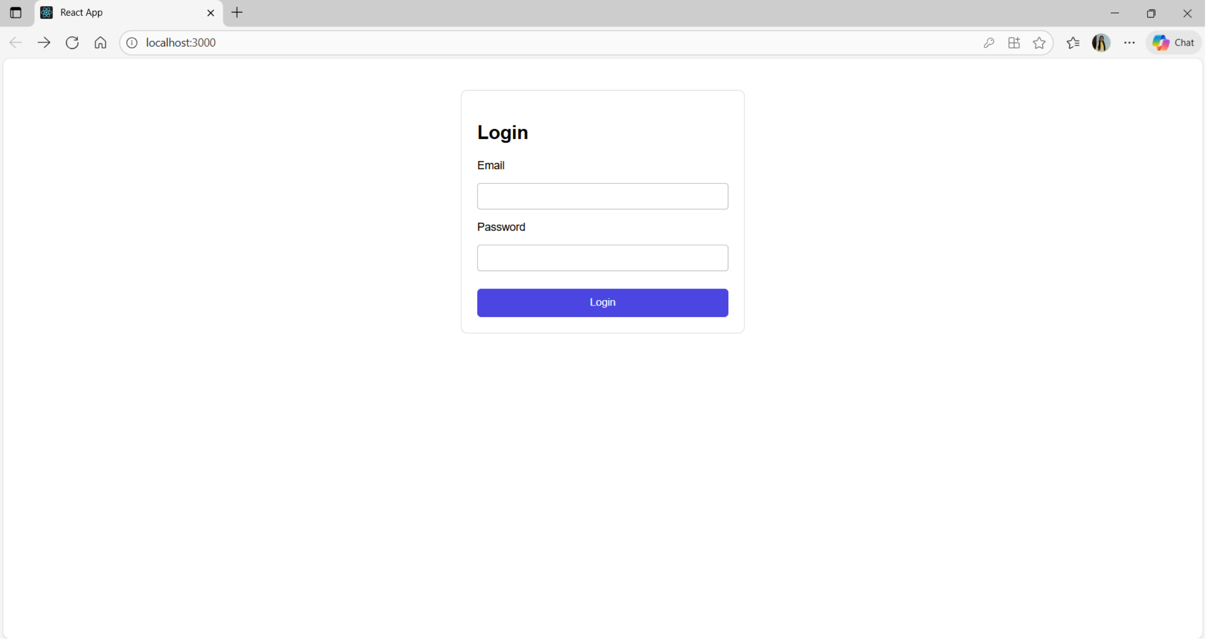
- **Formik form**
- **Yup validation schema**
- **Submit handler**
- **Clean UI layout**

pages/LoginFormik.js

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

const LoginFormik = () => {
  const initialValues = {
    email: "",
    password: ""
  };

  const validationSchema = Yup.object({
    email: Yup.string()
      .email("Invalid email format")
      .required("Email is required"),
    password: Yup.string()
      .min(6, "Minimum 6 characters")
      .required("Password is required")
  });

  const handleSubmit = (values) => {
    console.log("Form Data:", values);
  };

  return (
    <div style={styles.container}>
      <h2>Login</h2>

      <Formik
        initialValues={initialValues}
        validationSchema={validationSchema}
        onSubmit={handleSubmit}
      >
        <Form style={styles.form}>
          <label style={styles.label}>Email</label>
          <Field name="email" type="text" style={styles.input} />
          <ErrorMessage name="email" component="div"
style={styles.error} />

          <label style={styles.label}>Password</label>
          <Field name="password" type="password" style={styles.input}
/>
          <ErrorMessage name="password" component="div"
style={styles.error} />

          <button type="submit" style={styles.btn}>Login</button>
        </Form>
      </Formik>
    </div>
  );
};

const styles = {
  container: {
    width: "320px",
    margin: "40px auto",
    padding: "20px",
    border: "1px solid #ddd",
    borderRadius: "8px",
    fontFamily: "Arial"
  },
  form: {
    display: "flex",
    flexDirection: "column",
    gap: "14px"
  },
  label: {
    fontSize: "14px"
  },
  input: {
    padding: "8px",
    borderRadius: "4px",
    border: "1px solid #bbb"
  },
  btn: {
    padding: "10px",
    marginTop: "8px",
    background: "#4F46E5",
    color: "#fff",
    border: "none",
    borderRadius: "5px",
    cursor: "pointer"
  },
  error: {
    color: "crimson",
    fontSize: "13px"
  }
};

export default LoginFormik;
```

App.js

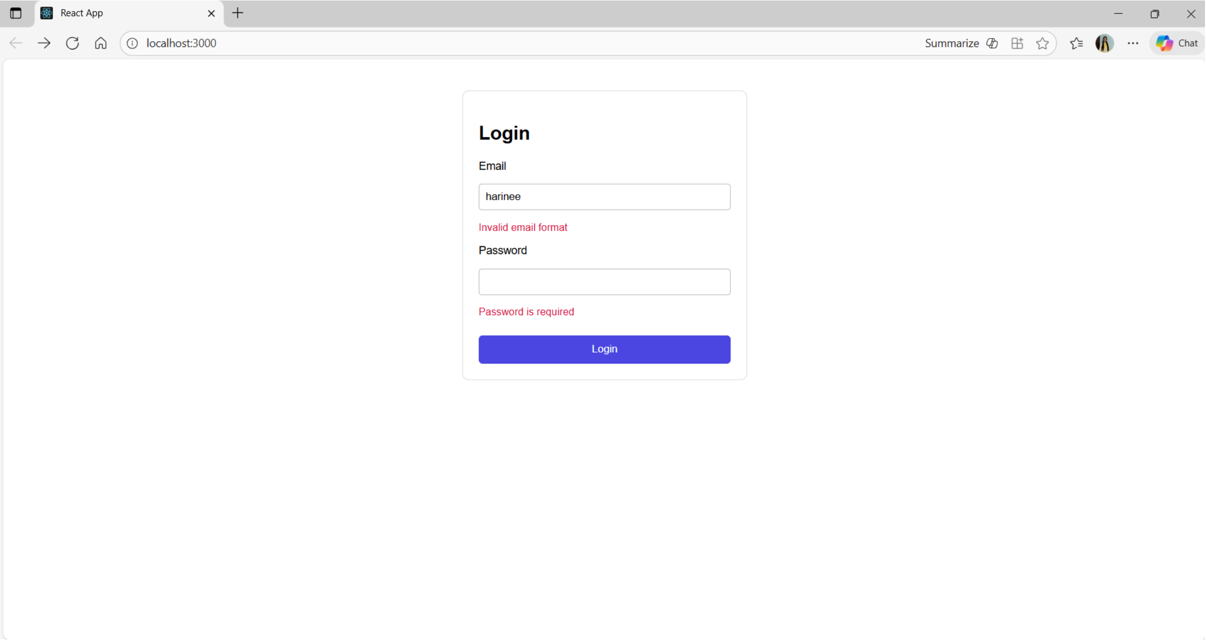
```
import LoginFormik from "./pages/LoginFormik";

function App() {
  return <LoginFormik />;
}

export default App;
```



Login

Email

Password



Login

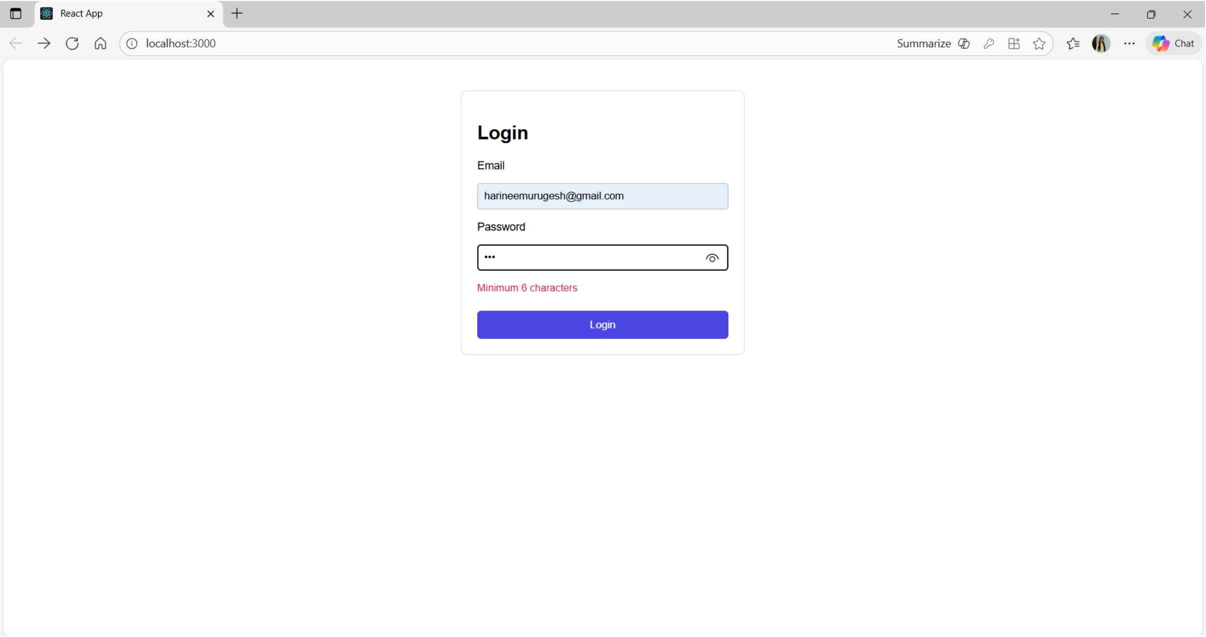
Email

harinee

Invalid email format

Password

Password is required



Login

Email

harineemurugesh@gmail.com

Password

•••

Minimum 6 characters

Q7. Node.js Core Modules

User Story

As a Backend Logging Engineer, I need a Node CLI script that writes “App started” into logs/app.log and starts a simple JSON HTTP server returning { "status": "running" }, using core modules so that the system is operational without external dependencies.

Acceptance Criteria

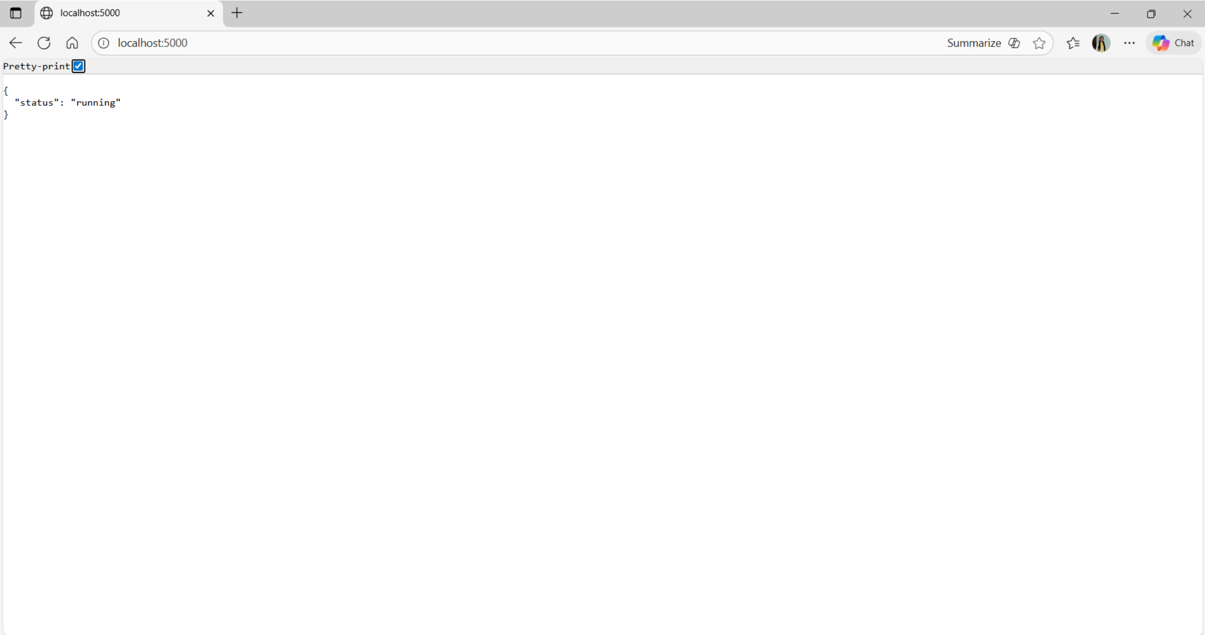
- **Use fs to write log**
- **Use path to create directory-safe file path**
- **Create HTTP server**

server.js

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

const logsDir = path.join(__dirname, "logs");

if (!fs.existsSync(logsDir)) {
  fs.mkdirSync(logsDir);
}

const logFile = path.join(logsDir, "app.log");
fs.writeFileSync(logFile, "App started", { flag: "w" });

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

// Start server
server.listen(5000, () => {
  console.log("Server running on http://localhost:5000");
});
```

Screenshots:

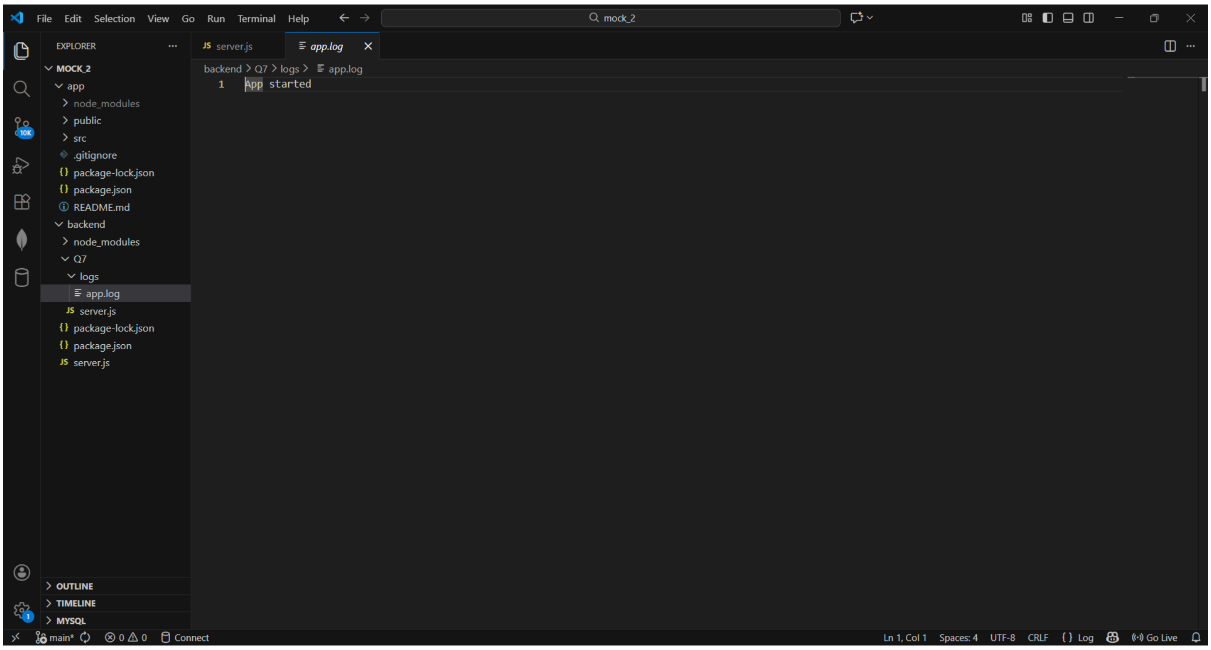
Browser Output:



Pretty-print ☑

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

It creates a app.log



```
App started
```

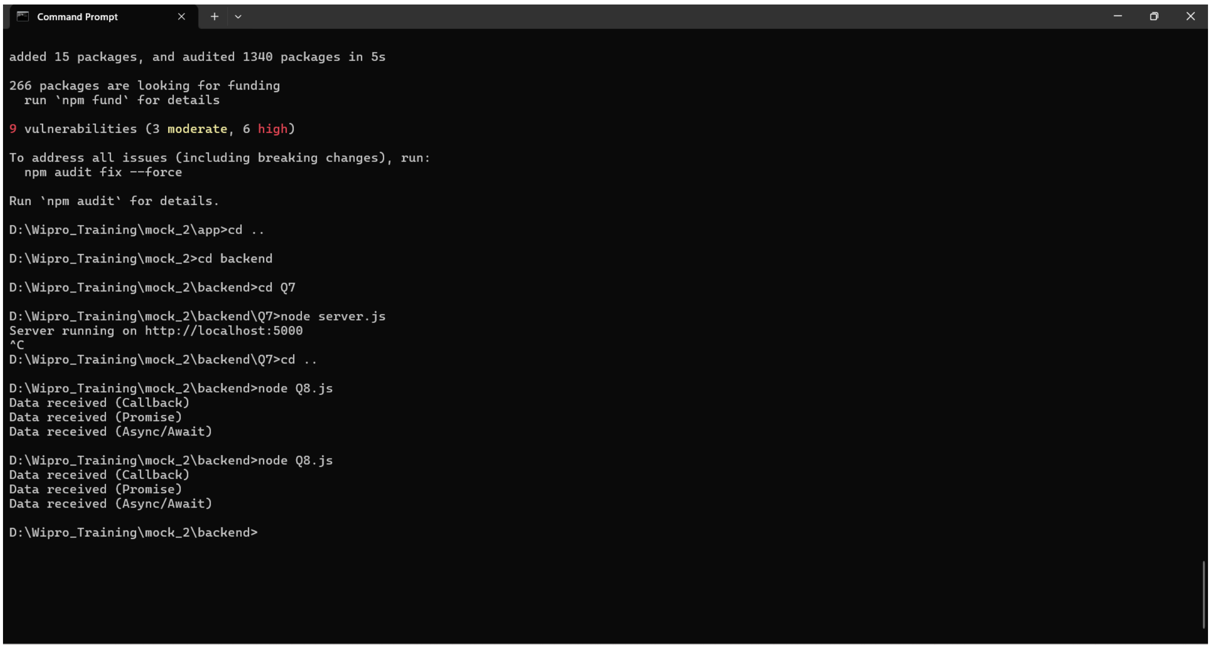
Q8. Asynchronous JavaScript (Callbacks → Promise → Async/Await)
User Story
As a Modernization Engineer, I must refactor the existing callback-based fetchData function into Promise-based and Async/Await-based versions so that the team follows updated asynchronous patterns.
Acceptance Criteria

- **Correct Promise version**
- **Correct Async/Await version**
- **Correct output message**
- **Maintain same functionality**

Q8.js

```
// 1. Callback version
function fetchDataCallback(callback) {
  setTimeout(() => {
    callback("Data received (Callback)");
  }, 1000);
}

fetchDataCallback((result) => {
  console.log(result);
});

// 2. Promise version
function fetchDataPromise() {
  return new Promise((resolve) => {
    setTimeout(() => {
      resolve("Data received (Promise)");
    }, 1000);
  });
}

fetchDataPromise().then((result) => {
  console.log(result);
});

// 3. Async/Await version
function fetchDataAsync() {
  return new Promise((resolve) => {
    setTimeout(() => {
      resolve("Data received (Async/Await)");
    }, 1000);
  });
}

async function getData() {
  const result = await fetchDataAsync();
  console.log(result);
}

getData();
```

Command Prompt Output:



```
added 15 packages, and audited 1340 packages in 5s

266 packages are looking for funding
  run `npm fund` for details

9 vulnerabilities (3 moderate, 6 high)

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.

D:\Wipro_Training\mock_2\app>cd ..

D:\Wipro_Training\mock_2>cd backend

D:\Wipro_Training\mock_2\backend>cd Q7

D:\Wipro_Training\mock_2\backend\Q7>node server.js
Server running on http://localhost:5000
^C
D:\Wipro_Training\mock_2\backend\Q7>cd ..

D:\Wipro_Training\mock_2\backend>node Q8.js
Data received (Callback)
Data received (Promise)
Data received (Async/Await)

D:\Wipro_Training\mock_2\backend>node Q8.js
Data received (Callback)
Data received (Promise)
Data received (Async/Await)

D:\Wipro_Training\mock_2\backend>
```

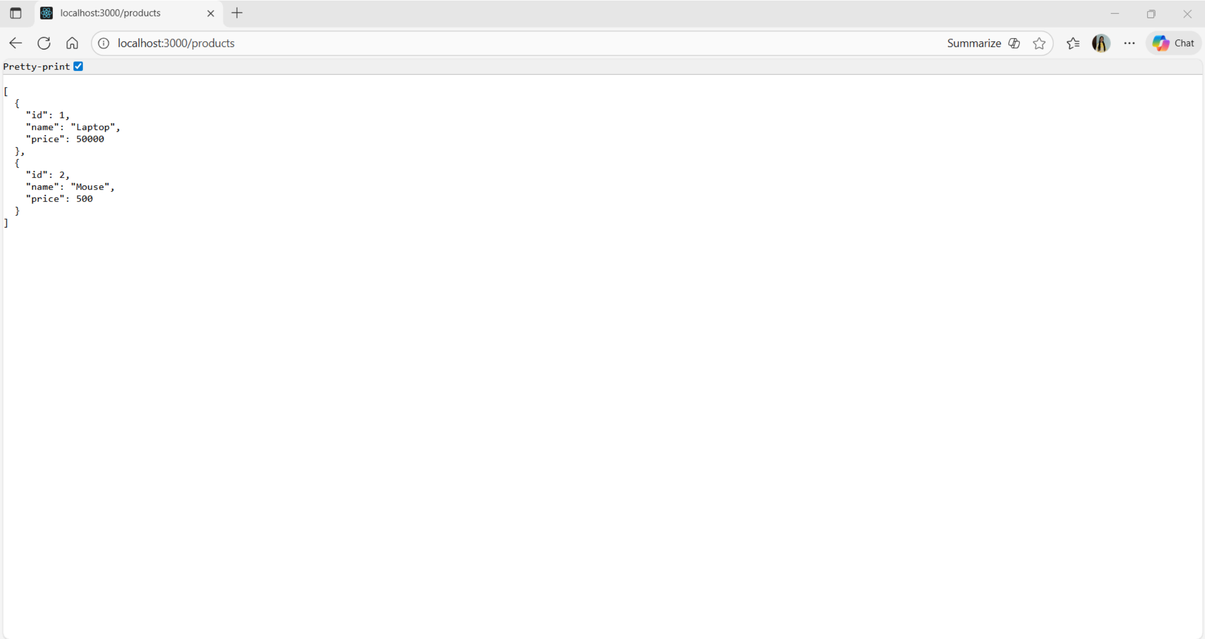
Q9. Express Routing + Middleware + Validation
User Story
As a Product API Developer, I must build an Express server where /products returns a list of products, /products POST validates name and price using express-validator, and a global middleware logs request method and URL so that the API follows basic standards.
Acceptance Criteria

- **GET and POST endpoints**
- **express-validator rules**
- **Middleware logs method + URL**
- **Return JSON responses**

Q9/server.js

```
const express = require("express");
const { body, validationResult } = require("express-validator");
const app = express();
app.use(express.json());

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


let products = [
  { id: 1, name: "Laptop", price: 50000 },
  { id: 2, name: "Mouse", price: 500 }
];


app.get("/products", (req, res) => {
  res.json(products);
});


app.post(
  "/products",
  [
    body("name").notEmpty().withMessage("Name is required"),
    body("price")
      .isFloat({ gt: 0 })
      .withMessage("Price must be a number greater than 0")
  ],
  (req, res) => {
    const errors = validationResult(req);

    if (!errors.isEmpty()) {
      return res.status(400).json({ errors: errors.array() });
    }

    const { name, price } = req.body;
    const newProduct = {
      id: products.length + 1,
      name,
      price
    };
    products.push(newProduct);
    res.status(201).json(newProduct);
  }
);

// Start server
app.listen(3000, () => {
  console.log("Server running on http://localhost:3000");
});
```

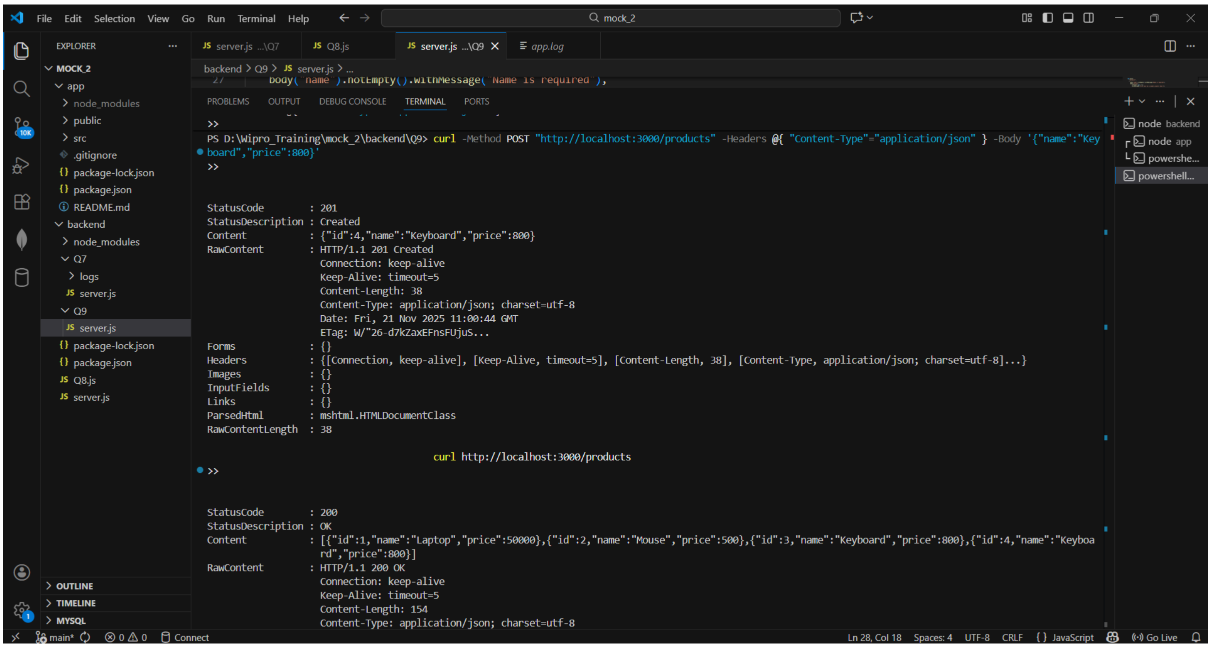
Screenshots:



```
[
  {
    "id": 1,
    "name": "Laptop",
    "price": 50000
  },
  {
    "id": 2,
    "name": "Mouse",
    "price": 500
  }
]
```

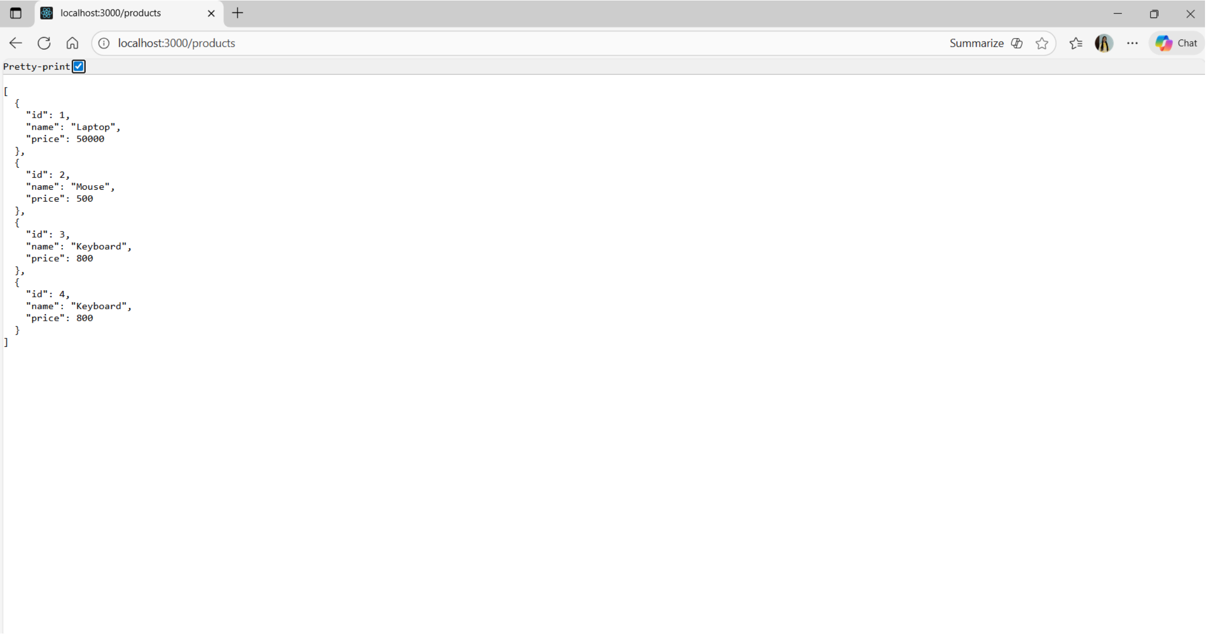
POST Request command in Cmd or Terminal:

```
curl -Method POST "http://localhost:3000/products" -Headers @{
"Content-Type"="application/json" } -Body '{"name":"Keyboard","price":800}'
```



```
27        body("name").notEmpty().withMessage("Name is required"),
```

```
>>
PS D:\Wipro_Training\mock_2\backend\Q9> curl -Method POST "http://localhost:3000/products" -Headers @{ "Content-Type"="application/json" } -Body '{"name":"Key
board","price":800}'
>>


StatusCode        : 201
StatusDescription : Created
Content           : {"id":4,"name":"Keyboard","price":800}
RawContent        : HTTP/1.1 201 Created
                    Connection: keep-alive
                    Keep-Alive: timeout=5
                    Content-Length: 38
                    Content-Type: application/json; charset=utf-8
                    Date: Fri, 21 Nov 2025 11:00:44 GMT
                    ETag: W/"26-d7kZaxEFnsFUjuS...
Forms             : {}
Headers           : {[Connection, keep-alive], [Keep-Alive, timeout=5], [Content-Length, 38], [Content-Type, application/json; charset=utf-8]...}
Images            : {}
InputFields       : {}
Links             : {}
ParsedHtml        : mshtml.HTMLDocumentClass
RawContentLength  : 38

                                curl http://localhost:3000/products
>>


StatusCode        : 200
StatusDescription : OK
Content           : [{"id":1,"name":"Laptop","price":50000},{"id":2,"name":"Mouse","price":500},{"id":3,"name":"Keyboard","price":800},{"id":4,"name":"Keyboa
                    rd","price":800}]
RawContent        : HTTP/1.1 200 OK
                    Connection: keep-alive
                    Keep-Alive: timeout=5
                    Content-Length: 154
                    Content-Type: application/json; charset=utf-8
```



localhost:3000/products

Pretty-print ☑

```
[
  {
    "id": 1,
    "name": "Laptop",
    "price": 50000
  },
  {
    "id": 2,
    "name": "Mouse",
    "price": 500
  },
  {
    "id": 3,
    "name": "Keyboard",
    "price": 800
  },
  {
    "id": 4,
    "name": "Keyboard",
    "price": 800
  }
]
```