

Q-1 React Basics (JSX, Components, Props)

Product Card

ProductCard.jsx

```
frontend > src > components > ProductCard.jsx > ...
1 import React from 'react'
2 import PropTypes from 'prop-types'
3 export default function ProductCard({ title, price, discount }){
4   const finalPrice = Number(price) - Number(discount);
5   return (
6     <div style={{ border: '1px solid #ddd', padding: 12, width: 260 }}>
7       <h3>{title}</h3>
8       <p>Price: ${price}</p>
9       <p>Discount: ${discount}</p>
10      <p><strong>Final Price: ${finalPrice}</strong></p>
11      <button>Shop Now</button>
12    </div>
13  );
14}
15 ProductCard.propTypes = {
16   title: PropTypes.string.isRequired,
17   price: PropTypes.oneOfType([PropTypes.string, PropTypes.number]).isRequired,
18   discount: PropTypes.oneOfType([PropTypes.string, PropTypes.number]).isRequired
19};
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

node - frontend + ×

Note that the development build is not optimized.
To create a production build, use `npm run build`.

webpack compiled successfully



[Home](#)

Sneakers

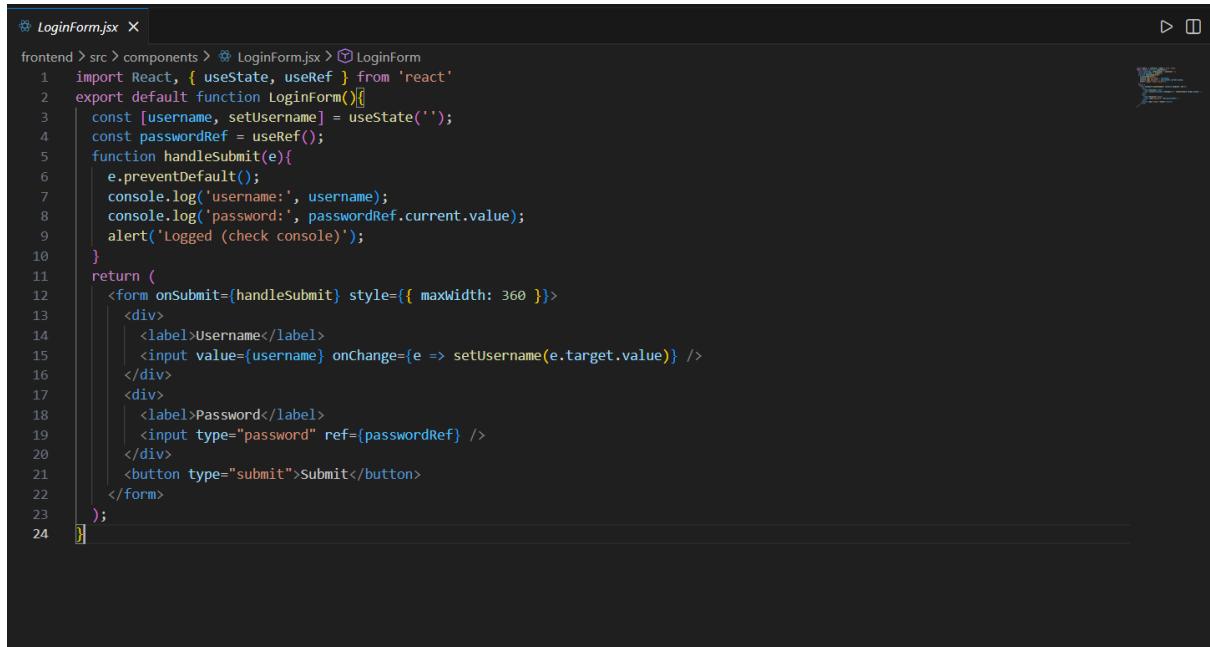
Price: ₹120

Discount: ₹20

Final Price: ₹100

[Shop Now](#)

Q-2 React State + Controlled and Uncontrolled Components



```
LoginForm.jsx
frontend > src > components > LoginForm.jsx > LoginForm
1 import React, { useState, useRef } from 'react'
2 export default function LoginForm(){
3   const [username, setUsername] = useState('');
4   const passwordRef = useRef();
5   function handleSubmit(e){
6     e.preventDefault();
7     console.log('username:', username);
8     console.log('password:', passwordRef.current.value);
9     alert('Logged (check console)');
10  }
11  return (
12    <form onSubmit={handleSubmit} style={{ maxWidth: 360 }}>
13      <div>
14        <label>Username</label>
15        <input value={username} onChange={e => setUsername(e.target.value)} />
16      </div>
17      <div>
18        <label>Password</label>
19        <input type="password" ref={passwordRef} />
20      </div>
21      <button type="submit">Submit</button>
22    </form>
23  );
24}
```



ShopNow

[Home](#) | [Login](#)

Username

Password

Q-3 React Class Component, Lifecycle, PropTypes, Styling

The screenshot shows a code editor window with the file `UserStatus.jsx` open. The code defines a React class component `UserStatus` with lifecycle methods `componentDidMount` and `componentWillUnmount`, and a `render` method. It uses `PropTypes` for prop validation. The terminal below shows a successful webpack compilation.

```
1 import React from 'react'
2 import PropTypes from 'prop-types'
3 export default class UserStatus extends React.Component{
4   constructor(props){
5     super(props);
6     this.state={message:'Fetching user status...'}
7   }
8   componentDidMount(){
9     this.timer=setTimeout(()=>{this.setState({message:'Active User'})},2000);
10 }
11 componentWillUnmount(){ clearTimeout(this.timer); }
12 render(){
13   return <div style={{padding:8, background:'#f6f6f6'}}>{this.state.message}</div>;
14 }
15 }
16 UserStatus.propTypes={ userId:PropTypes.number.isRequired };
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Note that the development build is not optimized.
To create a production build, use `npm run build`.

webpack compiled successfully



ShopNow

[Home](#) | [Login](#)

Sneakers

Price: ₹120

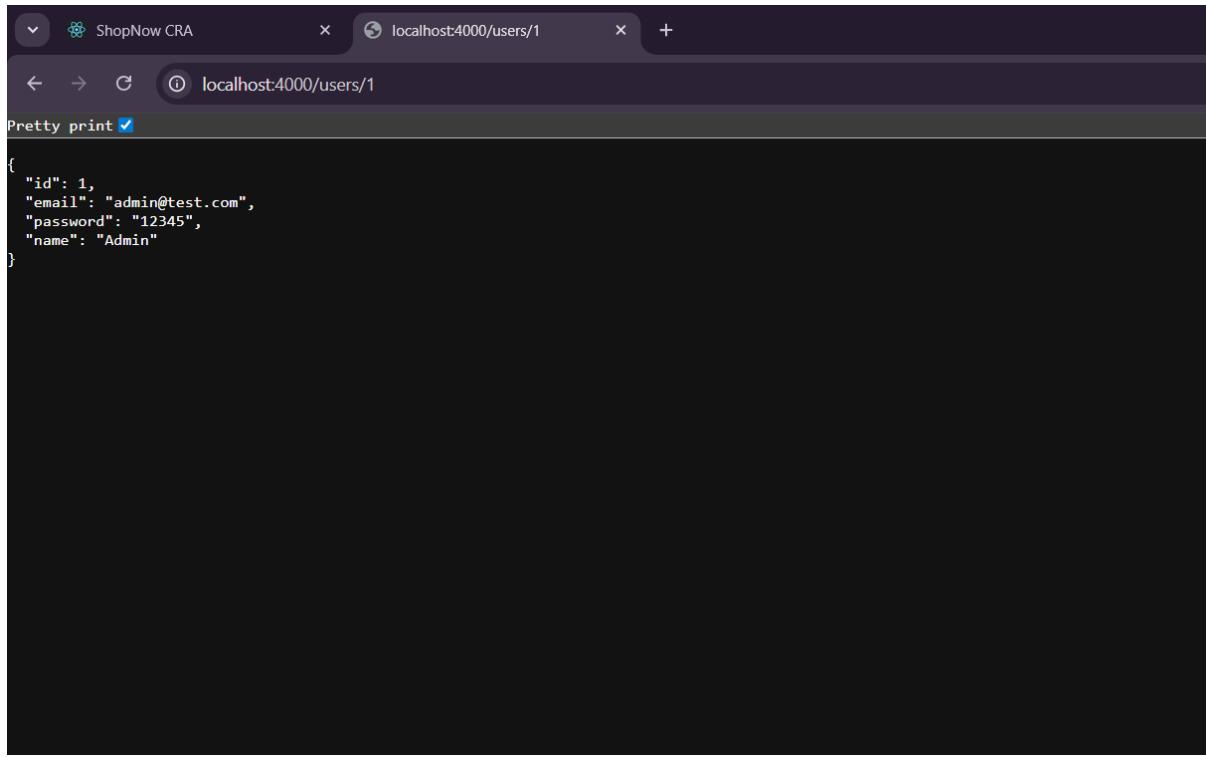
Discount: ₹20

Final Price: ₹100

[Shop Now](#)

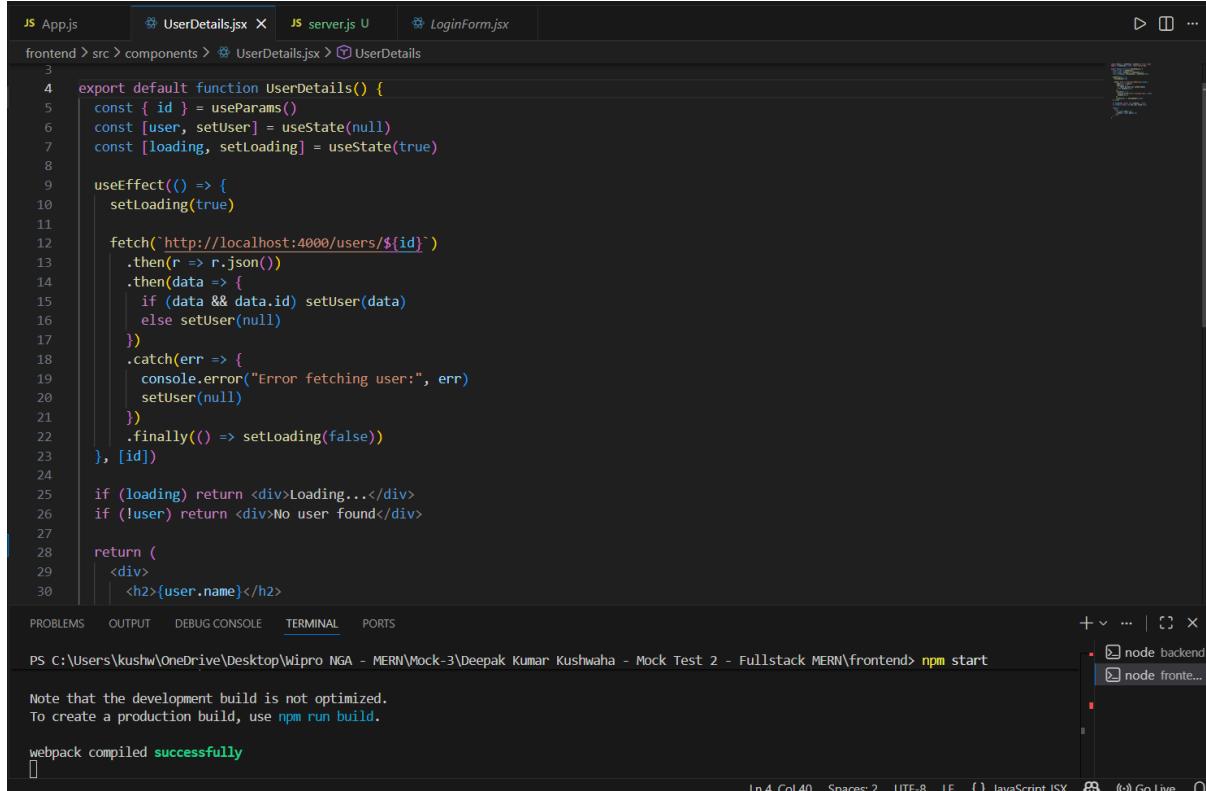
Active User

Q-4 React Router + API Integration



A screenshot of a web browser window titled "ShopNow CRA". The address bar shows "localhost:4000/users/1". The content area displays a JSON object with "Pretty print" checked:

```
{ "id": 1, "email": "admin@test.com", "password": "12345", "name": "Admin" }
```



A screenshot of a code editor (VS Code) showing the file "UserDetails.js". The code defines a functional component "UserDetails" that fetches user details from the API based on the id passed in the URL parameters.

```
3
4  export default function UserDetails() {
5    const { id } = useParams()
6    const [user, setUser] = useState(null)
7    const [loading, setLoading] = useState(true)
8
9    useEffect(() => {
10      setLoading(true)
11
12      fetch(`http://localhost:4000/users/${id}`)
13        .then(r => r.json())
14        .then(data => {
15          if (data && data.id) setUser(data)
16          else setUser(null)
17        })
18        .catch(err => {
19          console.error("Error fetching user:", err)
20          setUser(null)
21        })
22        .finally(() => setLoading(false))
23    }, [id])
24
25    if (loading) return <div>Loading...</div>
26    if (!user) return <div>No user found</div>
27
28    return (
29      <div>
30        <h2>{user.name}</h2>
```

The terminal at the bottom shows the command "npm start" being run, and the output indicates that webpack compiled successfully.



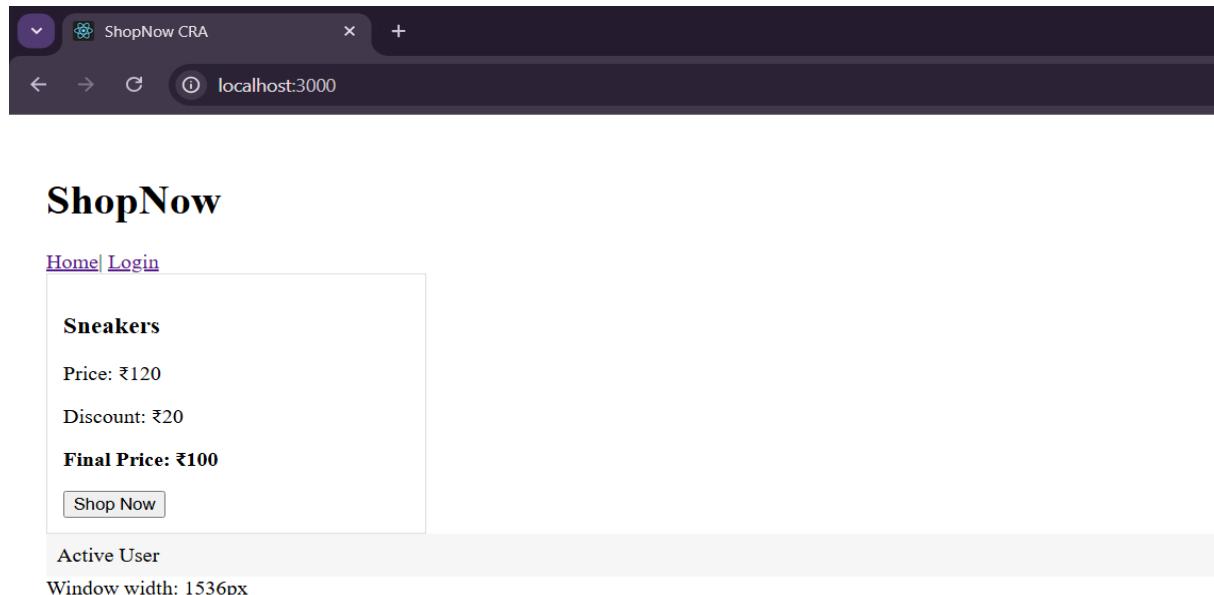
ShopNow

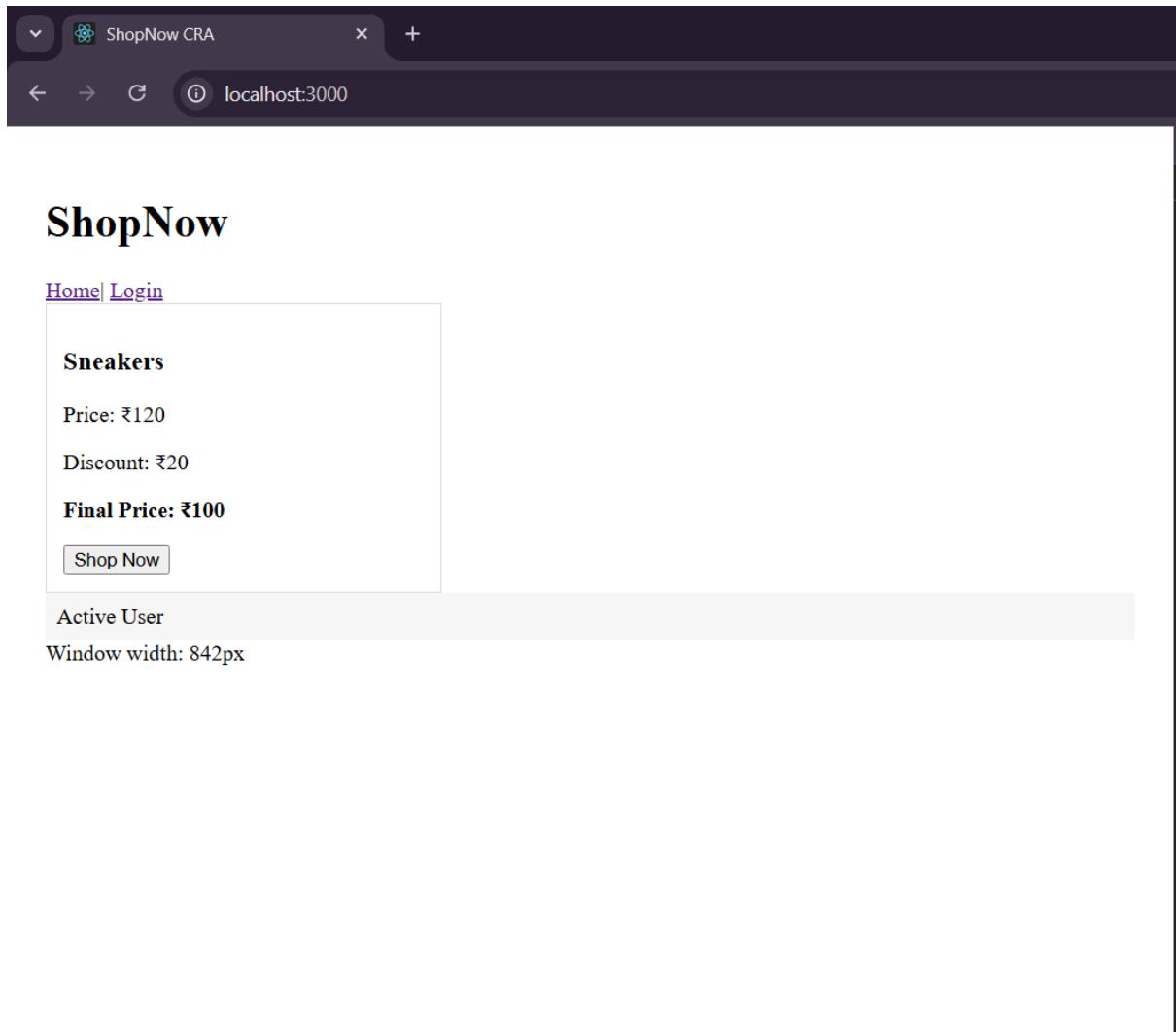
[Home](#) | [Login](#)

Admin

Email: admin@test.com

Q-5 Reusability Using HOC or Render Props





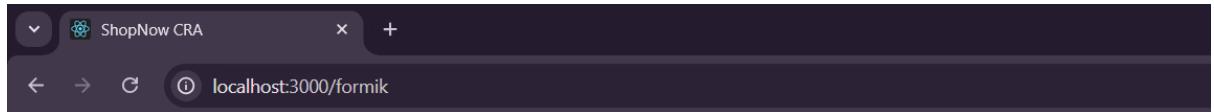
Q-6 Formik + Yup Validation

The screenshot shows the VS Code interface with the following details:

- Editor:** The code editor has tabs for `App.js`, `FormikLogin.jsx` (which is the active tab), and `UserDetails.jsx`. The `FormikLogin.jsx` tab contains the following code:

```
1 import React from 'react'
2 import { Formik, Field, Form, ErrorMessage } from 'formik'
3 import * as Yup from 'yup'
4 const schema=Yup.object().shape({
5   email:Yup.string().email().required(),
6   password:Yup.string().min(6).required()
7 });
8 export default function FormikLogin(){
9   return (
10     <Formik initialValues={{email:'',password:''}} validationSchema={schema}
11       onSubmit={v=>(console.log(v);alert('Submitted'))}>
12       <Form>
13         <div><label>Email</label><Field name="email" /><ErrorMessage name="email" /></div>
14         <div><label>Password</label><Field name="password" type="password" /><ErrorMessage name="password" /></div>
15         <button type="submit">Login</button>
16       </Form>
17     </Formik>
18   );
19 }
```

- Terminal:** The terminal shows the command `npm start` being run in the directory `C:\Users\kushw\OneDrive\Desktop\Wipro NGA - MERN\Mock-3\Deepak Kumar Kushwaha - Mock Test 2 - Fullstack MERN\frontend`. The output indicates that the development build is not optimized and suggests using `npm run build`. It also shows that webpack compiled successfully.
- Bottom Status Bar:** The status bar shows the node backend and node frontend processes running.



ShopNow

[Home](#) | [Login](#) | [Formik](#)
Email email must be a valid email
Password password is a required field

Q-7 Node.js Core Modules

The screenshot shows the Visual Studio Code interface. The code editor tab bar includes 'App.js', 'cli.js', 'UserDetails.jsx', and 'server.js'. The 'cli.js' tab is active, displaying the following code:

```
backend > cli.js > ...
1 const fs=require('fs')
2 const path=require('path')
3 const http=require('http')
4
5 const logDir=path.join(__dirname,'logs')
6 if(!fs.existsSync(logDir)) fs.mkdirSync(logDir)
7 const file=path.join(logDir,'app.log')
8 fs.appendFileSync(file,'App started\n')
9
10 http.createServer((req,res)=>{
11   res.setHeader('Content-Type','application/json')
12   res.end(JSON.stringify({status:'running'}))
13 }).listen(5000,()=>console.log("CLI server 5000"))
14
```

The terminal tab bar includes 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL', and 'PORTS'. The 'TERMINAL' tab is active, showing the command line output:

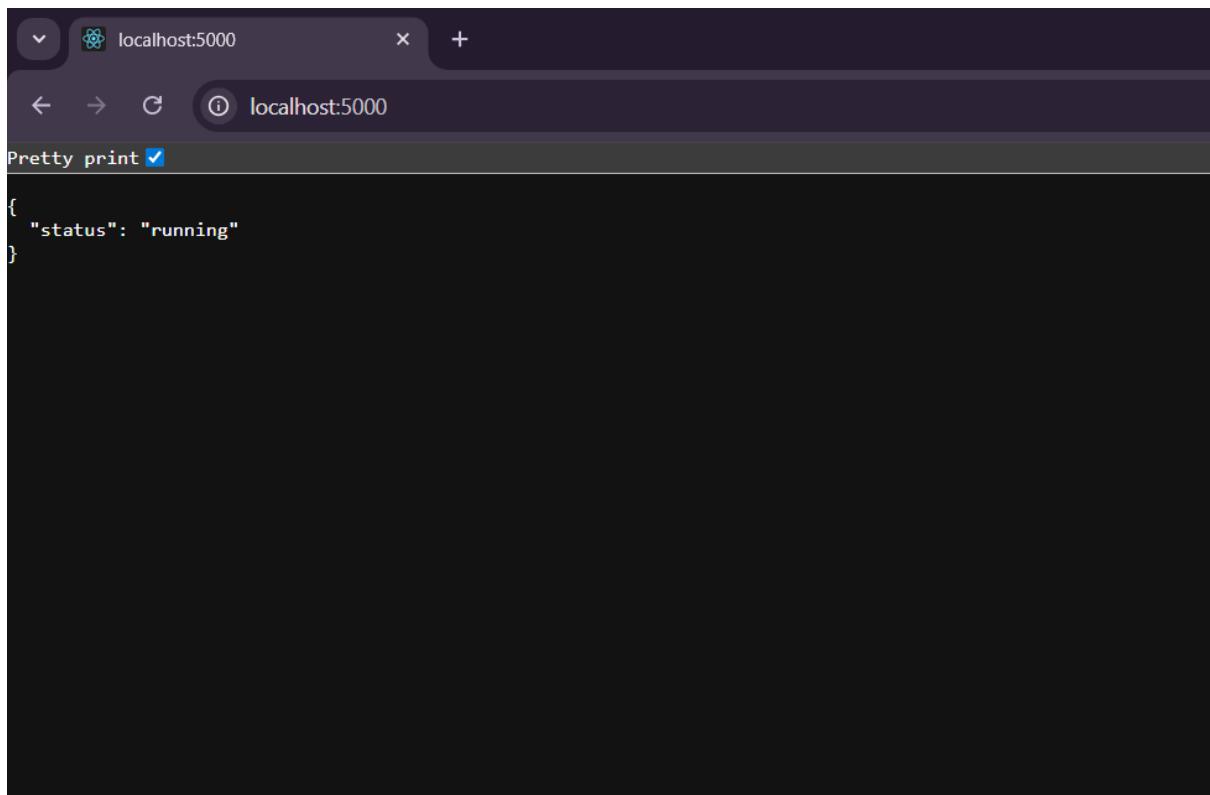
```
To address all issues, run:
GET /users/1
GET /users/1
GET /users/1
PS C:\Users\kushw\OneDrive\Desktop\Wipro NGA - MERN\Mock-3\Deepak Kumar Kushwaha - Mock Test 2 - Fullstack MERN\backend> node cli.js
CLI server 5000
```

The screenshot shows the Visual Studio Code interface with a different view. The left sidebar is the Explorer tab, showing the project structure with files like 'App.js', 'cli.js', 'UserDetails.jsx', 'server.js', 'index.js', 'package-lock.json', and 'package.json'. The right side shows the code editor and terminal. The code editor tab bar includes 'App.js', 'app.log', 'UserDetails.jsx', and 'server.js'. The 'app.log' tab is active, displaying the logs from the application:

```
backend > logs > app.log
1 App started
2
```

The terminal tab bar includes 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL', and 'PORTS'. The 'TERMINAL' tab is active, showing the command line output:

```
To address all issues, run:
GET /users/1
GET /users/1
GET /users/1
PS C:\Users\kushw\OneDrive\Desktop\Wipro NGA - MERN\Mock-3\Deepak Kumar Kushwaha - Mock Test 2 - Fullstack MERN\backend> node cli.js
CLI server 5000
```



A screenshot of a web browser window displaying a JSON response. The address bar shows "localhost:5000". Below the address bar, there is a toolbar with icons for back, forward, and search. A dropdown menu is open, showing "Pretty print" with a checked checkbox. The main content area displays the following JSON object:

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

Q-8 Asynchronous JavaScript (Callbacks → Promise → Async/Await)

The screenshot shows a code editor interface with several tabs at the top: App.js, async_examples.js (which is the active tab), UserDetails.jsx, and server.js. The code in the active tab, async_examples.js, demonstrates three different ways to handle asynchronous operations:

```
backend > JS async_examples.js > fetchDataCallback
1 function fetchDataCallback(cb){
2   setTimeout(()=>cb(null, 'callback-data'),200)
3 }
4 function fetchDataPromise(){
5   return new Promise(r=>setTimeout(()=>r('promise-data'),200))
6 }
7 async function fetchDataAsync(){
8   return await fetchDataPromise()
9 }
10 fetchDataCallback((e,d)=>console.log('cb',d))
11 fetchDataPromise().then(d=>console.log('promise',d))
12 :(async()=>console.log('async',await fetchDataAsync()))()
```

Below the code editor is a terminal window showing the execution of the script:

```
To address all issues, run:
GET /users/1
GET /users/1
GET /users/1
PS C:\Users\kushw\OneDrive\Desktop\Wipro NGA - MERN\Mock-3\Deepak Kumar Kushwaha - Mock Test 2 - Fullstack MERN\backend> node cli.js
CLI server 5000
PS C:\Users\kushw\OneDrive\Desktop\Wipro NGA - MERN\Mock-3\Deepak Kumar Kushwaha - Mock Test 2 - Fullstack MERN\backend> node async_examples.js
cb callback-data
promise promise-data
async promise-data
PS C:\Users\kushw\OneDrive\Desktop\Wipro NGA - MERN\Mock-3\Deepak Kumar Kushwaha - Mock Test 2 - Fullstack MERN\backend>
```

The terminal also indicates that there are no issues to address.

Q-9 Express Routing + Middleware + Validation

The screenshot shows a code editor with several tabs: App.js, products.json, UserDetails.jsx, and server.js. The server.js tab is active, displaying the following code:

```

backend > JS server.js > app.use callback
  6
  7  const cors = require('cors');
  8
  9  const app = express();
 10  app.use(cors());
 11  app.use(bodyParser.json());
 12
 13  let products = [ { id:1, name:'Sneakers', price:120}, {id:2,name:'Boots',price:150} ];
 14
 15  app.get('/products',(req,res)=>res.json(products));
 16
 17  app.post('/products',[body('name').isString().notEmpty(),
 18                      body('price').isNumeric()],
 19                      ,(req,res)=>{
 20                      const errors = validationResult(req);
 21                      if(!errors.isEmpty()) return res.status(400).json({errors:errors.array()});
 22                      const p={id:products.length+1, ...req.body};
 23                      products.push(p);
 24                      res.status(201).json(p)
 25                  })
 26  })
  
```

Below the code editor is a terminal window showing the following logs:

- PS C:\Users\kushw\OneDrive\Desktop\Wipro NGA - MERN\Mock-3\Deepak Kumar Kushwaha - Mock Test 2 - Fullstack MERN\backend> node cli.js
 CLI server 5000
- PS C:\Users\kushw\OneDrive\Desktop\Wipro NGA - MERN\Mock-3\Deepak Kumar Kushwaha - Mock Test 2 - Fullstack MERN\backend> node async_examples.js
 cb callback-data
 promise promise-data
 async promise-data
- PS C:\Users\kushw\OneDrive\Desktop\Wipro NGA - MERN\Mock-3\Deepak Kumar Kushwaha - Mock Test 2 - Fullstack MERN\backend> no
 node server.js
 Backend running 4000
- PS C:\Users\kushw\OneDrive\Desktop\Wipro NGA - MERN\Mock-3\Deepak Kumar Kushwaha - Mock Test 2 - Fullstack MERN\backend> node server.js
 Backend running 4000
 GET /
 GET /products

The screenshot shows a Postman request for `http://localhost:4000/products`. The request method is `GET`, and the URL is `http://localhost:4000/products`. The response status is `200 OK`, time `20 ms`, size `343 B`.

Body

```

1  [
2    {
3      "id": 1,
4      "name": "Sneakers",
5      "price": 120
6    },
7    {
8      "id": 2,
9      "name": "Boots",
10     "price": 150
11   }
12 ]
  
```

HTTP <http://localhost:4000/products>

POST http://localhost:4000/products Send

Params Authorization Headers (8) **Body** Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary JSON Cookies Beautify

```
1 {
2   "name": "Shirt",
3   "price": 999
4 }
```

Body Cookies Headers (8) Test Results Status: 201 Created Time: 41 ms Size: 307 B Save Response

Pretty Raw Preview Visualize JSON Search

```
1 {
2   "id": 3,
3   "name": "Shirt",
4   "price": 999
5 }
```

```
Backend running 4000
PS C:\Users\kushw\OneDrive\Desktop\Wipro NGA - MERN\Mock-3\Deepak Kumar Kushwaha - Mock Test 2 - Fullstack MERN\backend> node server.js
Backend running 4000
GET /
GET /products
GET /products
POST /products
```

Q-10 REST API + JWT Authentication

The screenshot shows two requests in the Postman interface:

POST http://localhost:4000/login

Body (JSON):

```
1 {
2     "email": "admin@test.com",
3     "password": "12345"
4 }
```

Response (Pretty JSON):

```
1 {
2     "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJsb2dpbiIsInN1YiI6ImFkb2JlIiwiaWF0IjoxNjAxMjEwOTkxLCJleHAiOjE2MjAxMjEwOTkxfQ.3nLNHVXpG_bC3DnogqJDNmfmLa1DG-q_WlyeUujtw"
3 }
```

GET http://localhost:4000/dashboard

Headers (7):

- Host: <calculated when request is sent>
- User-Agent: PostmanRuntime/7.49.1
- Accept: */*
- Accept-Encoding: gzip, deflate, br
- Connection: keep-alive
- Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJsb2dpbiIsInN1YiI6ImFkb2JlIiwiaWF0IjoxNjAxMjEwOTkxLCJleHAiOjE2MjAxMjEwOTkxfQ.3nLNHVXpG_bC3DnogqJDNmfmLa1DG-q_WlyeUujtw

Response (Pretty JSON):

```
1 {
2     "msg": "Welcome user 1"
3 }
```

HTTP <http://localhost:4000/dashboard> Save

GET http://localhost:4000/dashboard Send

Params Authorization Headers (7) Body Pre-request Script Tests Settings Cookies

Key	Value
Host	<calculated when request is sent>
User-Agent	PostmanRuntime/7.49.1
Accept	*
Accept-Encoding	gzip, deflate, br
Connection	keep-alive
Authorization	Bearer

Body Cookies Headers (8) Test Results Status: 401 Unauthorized Time: 7 ms Size: 302 B Save Response

Pretty Raw Preview Visualize JSON

```
1 "error": "Invalid token"
```