

Interactive Tutorial Component

Homepage:

Interactive Tutorials

Create Tutorial

Welcome to Interactive Tutorials

Create and manage interactive tutorials with videos, steps, and quizzes

Create Your First Tutorial

Available Tutorials (2)

DSA

A complete introduction to Data Structures and Algorithms, covering core topics essential for problem solving and coding interviews

2 modules

Edit Delete

Start Learning

OS

Covers the role, purpose, and types of operating systems. Introduces concepts like system calls, kernel vs. user mode, and basic OS services.

3 modules

Edit Delete

Start Learning

Add tutorial:

Interactive Tutorials

Create Tutorial

Add New Tutorial

☒ Advanced Mode (Multiple Modules)

Title

Description

Video URL

Module 1

Module Title

Content

Questions

+ Add Question

Question 1

Question Text

Options

+ Add Option

Option 1

Option 2

Correct Answer

Select correct answer

Add Module

Cancel

Create Tutorial


Courses:

[Interactive Tutorials](#)[Create Tutorial](#)

[← Back to Tutorials](#)

OS

Covers the role, purpose, and types of operating systems. Introduces concepts like system calls, kernel vs. user mode, and basic OS services.



Operating System Full Course | Operating System Tutorials for Beginners

Watch on [YouTube](#)

Progress

Module 1/3

Previous

Next

Module 1: Introduction to Operating Systems

Covers the role, purpose, and types of operating systems. Introduces concepts like system calls, kernel vs. user mode, and basic OS services.

Question 1: What is the primary role of an Operating System?

Previous


Next

[Interactive Tutorials](#)[Create Tutorial](#)

[← Back to Tutorials](#)

OS

Covers the role, purpose, and types of operating systems. Introduces concepts like system calls, kernel vs. user mode, and basic OS services.



Operating System Full Course | Operating System Tutorials for Beginners

Watch on [YouTube](#)

Progress

Module 2/3

Previous

Next

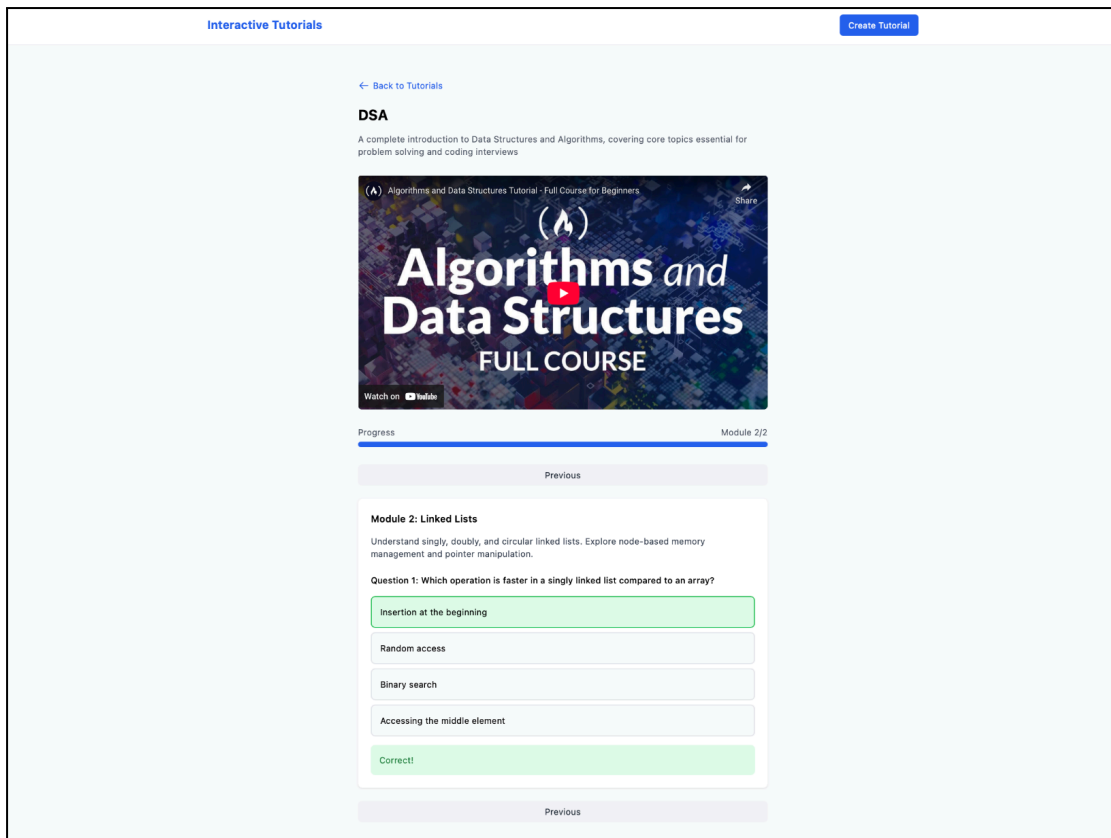
Module 2: Process Management

Focuses on how the OS handles processes including creation, scheduling, termination, and inter-process communication.

Question 1: Which of the following is NOT a valid process state?

Previous

Next



Source Code:

Docker-compose.yml:

services:

backend:

build: ./backend

ports:

- "5000:5000"

volumes:

- ./backend:/app

- ./backend/data:/app/data

environment:

- NODE_ENV=development

networks:

- appnet

restart: unless-stopped

healthcheck:

test: ["CMD", "curl", "-f", "http://localhost:5000/health"]

interval: 30s

timeout: 10s

retries: 3

dns:

- 8.8.8.8

- 8.8.4.4

frontend:

build: ./frontend

ports:

- "5173:80"

volumes:

- ./frontend:/app

- /app/node_modules

environment:

- VITE_API_URL=http://backend:5000

networks:

- appnet

restart: unless-stopped

depends_on:

- backend

dns:

- 8.8.8.8

- 8.8.4.4

networks:

appnet:

driver: bridge

ipam:

driver: default

config:

- subnet: 172.20.0.0/16

frontend/index.html:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8" />

<link rel="icon" type="image/svg+xml" href="/vite.svg" />

<meta name="viewport" content="width=device-width, initial-scale=1.0" />

<title>Interactive Tutorials</title>

</head>

<body>

<div id="root"></div>

<script type="module" src="/src/main.jsx"></script>

</body>

</html>

tailwind.config.js:

```
export default {
  content: [
    './index.html',
    './src/**/*. {js,ts,jsx,tsx}',
  ],
  theme: {
    extend: {},
  },
  plugins: [
    require('@tailwindcss/aspect-ratio'),
  ],
}
```

frontend/public/index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <title>Interactive Tutorials</title>
  </head>
  <body>
    <div id="root"></div>
  </body>
</html>
```

Interactive_Tutorials_microservice/frontend/vite.config.js:

```
import { defineConfig } from 'vite'
import react from '@vitejs/plugin-react'
export default defineConfig({
  plugins: [react()],
  server: {
    host: '0.0.0.0',
    port: 5173,
    proxy: {
      '/api': {
        target: 'http://localhost:5000',
        changeOrigin: true,
      },
    },
  },
  build: {
    target: 'esnext',
    rollupOptions: {
```

```
output: {  
  manualChunks: undefined  
}  
}  
}  
})
```

Interactive_Tutorials_microservice/frontend/postcss.config.js

```
export default {  
  plugins: {  
    tailwindcss: {},  
    autoprefixer: {},  
  },  
}
```

Interactive_Tutorials_microservice/frontend/src/index.js

```
import React from 'react';  
import ReactDOM from 'react-dom/client';  
import App from './App';  
import './index.css';  
const root = ReactDOM.createRoot(document.getElementById('root'));  
root.render(<App />);
```

Interactive_Tutorials_microservice/frontend/src/index.css

```
@tailwind base;  
@tailwind components;  
@tailwind utilities;
```

Interactive_Tutorials_microservice/frontend/src/services/api.js

```
import axios from 'axios';  
const API_URL = import.meta.env.VITE_API_URL;  
const api = axios.create({  
  baseURL: API_URL,  
  headers: {  
    'Content-Type': 'application/json',  
  },  
});  
export const getTutorials = () => api.get('/tutorials');  
export const getTutorial = (id) => api.get(`/tutorials/${id}`);  
export const createTutorial = (tutorial) => api.post('/tutorials', tutorial);  
export const updateTutorial = (id, tutorial) => api.put(`/tutorials/${id}`, tutorial);  
export const deleteTutorial = (id) => api.delete(`/tutorials/${id}`);
```

App.jsx:

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

import { BrowserRouter as Router, Routes, Route, Link } from 'react-router-dom';

import TutorialPage from './components/TutorialPage';

import AddTutorial from './components/AddTutorial';

import EditTutorial from './components/EditTutorial';

const App = () => {

  const [tutorials, setTutorials] = useState([]);

  const [loading, setLoading] = useState(true);

  const [error, setError] = useState(null);

  const fetchTutorials = async () => {

    try {

      const response = await fetch(`${import.meta.env.VITE_API_URL}/api/tutorials`);

      if (!response.ok) {

        throw new Error('Failed to fetch tutorials');

      }

      const data = await response.json();

      setTutorials(data);

    } catch (err) {

      setError(err.message);

    } finally {

      setLoading(false);

    }

  };

  const handleDelete = async (id) => {

    if (window.confirm('Are you sure you want to delete this tutorial?')) {
```

```
try {  
  
  const response = await fetch(`${import.meta.env.VITE_API_URL}/api/tutorials/${id}`, {  
  
    method: 'DELETE',  
  
  });  
  
  if (!response.ok) {  
  
    throw new Error('Failed to delete tutorial');  
  
  }  
  
  fetchTutorials();  
  
} catch (err) {  
  
  setError(err.message);  
  
}  
  
}  
  
};  
  
useEffect(() => {  
  
  fetchTutorials();  
  
}, []);  
  
const getTutorialProgress = (tutorial) => {  
  
  if (tutorial.modules && tutorial.modules.length > 0) {  
  
    return `${tutorial.modules.length} module${tutorial.modules.length !== 1 ? 's' : ''}`;  
  
  } else if (tutorial.steps && tutorial.steps.length > 0) {  
  
    return `${tutorial.steps.length} step${tutorial.steps.length !== 1 ? 's' : ''}`;  
  
  }  
  
  return 'No content';  
  
};  
  
const HomePage = () => {  
  
  if (loading) {
```



```
return (  
  
<div className="flex items-center justify-center min-h-screen">  
  
<div className="animate-spin rounded-full h-12 w-12 border-t-2 border-b-2  
border-blue-500"></div>  
  
</div>  
  
>;  
  
>  
  
if (error) {  
  
return (  
  
<div className="flex items-center justify-center min-h-screen">  
  
<div className="text-red-600 text-xl">{error}</div>  
  
</div>  
  
>;  
  
>  
  
return (  
  
<div>  
  
<div className="bg-gradient-to-r from-blue-500 to-purple-600 rounded-lg p-8 mb-8 text-white">  
  
<h1 className="text-4xl font-bold mb-4">Welcome to Interactive Tutorials</h1>  
  
<p className="text-xl mb-6">  
  
Create and manage interactive tutorials with videos, steps, and quizzes  
  
</p>  
  
<Link  
  
to="/add"  
  
className="inline-block px-6 py-3 bg-white text-blue-600 rounded-md hover:bg-gray-100"  
  
>  
  
Create Your First Tutorial  
  
</Link>
```

```
</div>

<div className="mb-8">

<h2 className="text-2xl font-bold mb-4">

Available Tutorials ( {tutorials.length} )

</h2>

{tutorials.length === 0 ? (

<div className="text-center py-8 text-gray-500">

No tutorials available. Create your first tutorial!

</div>

) : (

<div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3 gap-6">

{tutorials.map((tutorial) => (

<div

key={tutorial.id}

className="bg-white rounded-lg shadow-md overflow-hidden hover:shadow-lg transition-shadow"

>

<div className="p-6">

<h3 className="text-xl font-semibold mb-2">{tutorial.title}</h3>

<p className="text-gray-600 mb-4">{tutorial.description}</p>

<div className="flex items-center justify-between">

<span className="text-sm text-gray-500">

{getTutorialProgress(tutorial)}

</span>

<div className="flex space-x-2">

<Link

to={`'/edit/${tutorial.id}`} `}
```

```
className="text-blue-600 hover:text-blue-800"
```

```
>
```

```
Edit
```

```
</Link>
```

```
<button
```

```
onClick={() => handleDelete(tutorial.id)}
```

```
className="text-red-600 hover:text-red-800"
```

```
>
```

```
Delete
```

```
</button>
```

```
</div>
```

```
</div>
```

```
</div>
```

```
<Link
```

```
to={` /tutorial/${tutorial.id}`}
```

```
className="block w-full py-3 text-center bg-gray-100 hover:bg-gray-200 text-gray-800  
font-medium"
```

```
>
```

```
Start Learning
```

```
</Link>
```

```
</div>
```

```
))}
```

```
</div>
```

```
))}
```

```
</div>
```

```
</div>
```

```
);
```

```
};
```

```
return (
```

```
<Router>
```

```
<div className="min-h-screen bg-gray-50">
```

```
<nav className="bg-white shadow-sm">
```

```
<div className="container mx-auto px-4 py-4">
```

```
<div className="flex justify-between items-center">
```

```
<Link to="/" className="text-2xl font-bold text-blue-600">
```

Interactive Tutorials

```
</Link>
```

```
<Link
```

```
to="/add"
```

```
className="px-4 py-2 bg-blue-600 text-white rounded-md hover:bg-blue-700"
```

```
>
```

Create Tutorial

```
</Link>
```

```
</div>
```

```
</div>
```

```
</nav>
```

```
<main className="container mx-auto px-4 py-8">
```

```
<Routes>
```

```
<Route path="/" element={<HomePage />} />
```

```
<Route path="/add" element={<AddTutorial onTutorialAdded={fetchTutorials} />} />
```

```
<Route path="/edit/:id" element={<EditTutorial onTutorialUpdated={fetchTutorials} />} />
```

```
<Route path="/tutorial/:id" element={<TutorialPage />} />
```

```
</Routes>
```

```
</main>

</div>

</Router>

);

};

export default App;
```

Main.jsx:

```
import React from 'react';

import ReactDOM from 'react-dom/client';

import App from './App';

import './index.css'; // Optional: only if you're using Tailwind or custom styles

const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(

  <React.StrictMode>

  <App />

  </React.StrictMode>

);
```