🔧 Test Workflow Generator - 5-Minute Code Overview

> 🚀 A modern drag-and-drop test workflow builder with beautiful UI and seamless user experience

📖 Project Introduction (30 seconds)

This is a modern **drag-and-drop test workflow builder** built with Next.js 15, React 19, and TypeScript. The application allows users to create custom test sequences by dragging functions from a sidebar into a workflow canvas, then execute them with visual feedback.

🏗️ Architecture Overview (1 minute)

💻 Tech Stack

- 🌐 **Frontend**: Next.js 15 with App Router
- 🎨 **Styling**: Tailwind CSS v4 with custom gradients and animations
- 📝 **Language**: TypeScript for type safety
- ⚛️ **UI Framework**: React 19 with modern hooks

📁 Project Structure

```
src/app/
├── 📄 page.tsx          # Main workflow interface component
├── 🎯 layout.tsx        # Root layout with fonts
└── 🎨 globals.css       # Global styles and Tailwind
```

⚙️ Core Components Breakdown (2 minutes)

1. 🧠 State Management (`page.tsx:20-23`)

```
// 📋 Main workflow state - stores array of dropped functions
const [workflowItems, setWorkflowItems] = useState<WorkflowItem[]>([]);

// ⏳ Execution state - prevents multiple simultaneous runs
const [isRunning, setIsRunning] = useState(false);

// 🖱️ Drag tracking - identifies which function is being dragged
const [draggedFunction, setDraggedFunction] = useState<string>('');

// 📍 Drop position indicator - shows where item will be inserted
const [dragOverIndex, setDragOverIndex] = useState<number | null>(null);
```

2. 🖱️ Drag-and-Drop System

- 📋 **Functions Panel**: 5 predefined functions (Function1-5) with draggable cards

- 🎯 **Drop Handlers**: Support for both canvas drops and precise insertion between items
- ✨ **Visual Feedback**: Real-time drag indicators and hover effects

3. 🔄 Workflow Execution (`page.tsx:97-111`)

```
const runWorkflow = async () => {
  // 🚫 Prevent execution if workflow is empty
  if (workflowItems.length === 0) {
    alert('Please add functions to your workflow before running.');
    return;
  }

  // 🔒 Set running state to disable UI interactions
  setIsRunning(true);

  // 🔄 Execute each function in sequence
  for (let i = 0; i < workflowItems.length; i++) {
    const currentItem = workflowItems[i];

    // ⏱️ Simulate function execution time (1 second delay)
    await new Promise(resolve => setTimeout(resolve, 1000));

    // 📢 Show execution feedback with specific function name
    alert(`Test ${i + 1}: ${currentItem.name} has been executed`);
  }

  // ✅ Reset running state to re-enable UI
  setIsRunning(false);
};
```

🎨 UI/UX Features (1 minute)

✨ Advanced Styling

- 🪟 **Glass-morphism Design**: Backdrop blur with semi-transparent panels
- 🌈 **Gradient Animations**: Dynamic color transitions and hover effects
- 🎭 **Micro-interactions**: Scale transforms, rotation effects, and loading spinners

📱 Responsive Layout

- 📚 **Left Panel**: Fixed-width function library (320px)
- 🖼️ **Right Panel**: Flexible workflow canvas with overflow handling
- 📲 **Mobile-first**: Tailwind responsive classes throughout

🎯 User Experience

- 👆 **Drag Indicators**: Visual feedback showing where items will be inserted
- ⏳ **Loading States**: Animated spinner during workflow execution
- 🚫 **Error Handling**: Validation for empty workflows

💎 Code Quality & Best Practices (30 seconds)

📝 TypeScript Integration

- 🏷️ **Interface Definitions**: Strong typing for `WorkflowItem` structure
- 🛡️ **Type Safety**: Proper event typing for drag handlers
- ⚛️ **Modern React**: Using latest hooks and functional components

🔧 Development Setup

- 📏 **ESLint**: Next.js recommended configuration
- 🔥 **Hot Reload**: Instant development feedback
- ⚡ **Build Optimization**: Next.js automatic code splitting

🚀 Quick Start

```
npm run dev
# Open http://localhost:3000
```

🎬 **Demo Flow**: Drag functions → Build workflow → Click "Run Workflow" → See execution alerts

🌐 Live Demo

🚀 **Live Site**: test-workflow-nextjs.vercel.app

🚀 Deployment Process

This project was deployed to Vercel using the command line interface:

📋 **Deployment Steps Used:**

1. 📦 **Install Vercel CLI**:

   ```
   npm install -g vercel
   ```

2. 🔐 **Login to Vercel**:

   ```
   vercel login
   ```

3. 🚀 **Deploy from Project Directory**:

   ```
   cd test-workflow-nextjs
   vercel --prod
   ```

4. ⚙️ **Vercel Configuration**:

   - 🔧 Framework: Next.js (auto-detected)
   - 🏗️ Build Command: `npm run build`
   - 💻 Development Command: `npm run dev`
   - 📦 Install Command: `npm install`
   - 📁 Output Directory: `.next`

5. 🌐 **Custom Domain Setup**:

   ```
   vercel domains add test-workflow-nextjs.vercel.app
   vercel alias test-workflow-nextjs.vercel.app
   ```

🎯 **Features Enabled:**

- ⚡ Automatic deployments on git push
- 🔄 Preview deployments for pull requests
- 📊 Build optimization and edge caching
- 🌐 Global CDN distribution

🎯 Key Features Summary

Feature	Technology	Description
🖱️ **Drag & Drop**	HTML5 DnD API	Intuitive function placement
🎨 **Modern UI**	Tailwind CSS v4	Glass-morphism design
⚡ **Performance**	Next.js 15	Optimized builds & SSG
📝 **Type Safety**	TypeScript	Full type coverage
🔄 **Real-time**	React 19	Instant visual feedback

🤝 Contributing

Feel free to contribute to this project! Whether it's:

- 🐛 **Bug fixes**
- ✨ **New features**
- 📚 **Documentation improvements**
- 🎨 **UI/UX enhancements**

📄 License

This project is open source and available under the MIT License.

💡 **This codebase demonstrates modern React patterns, TypeScript best practices, and advanced CSS techniques in a practical, user-friendly application.**

> 🌟 **Star this repo** if you found it helpful! | 🐦 **Share** with your network | 📝 **Fork** and make it your own