

STEP 32: Cinematic Demo Engine - Complete Documentation

Overview

The Cinematic Demo Engine is a comprehensive system for showcasing FYNDR's capabilities through automated, narrated demonstrations. It provides both buyer and supplier perspectives, with support for cinematic (automated) and guided (manual) demo modes.

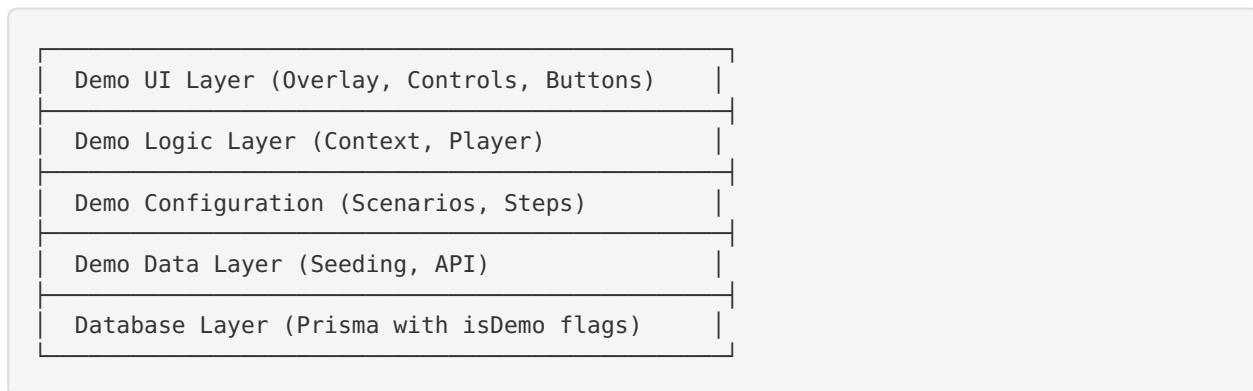
Table of Contents

1. [Architecture](#)
2. [Components](#)
3. [Demo Data](#)
4. [Demo Scenarios](#)
5. [Usage Guide](#)
6. [API Reference](#)
7. [Customization](#)
8. [Troubleshooting](#)

Architecture

High-Level Design

The demo system consists of five main layers:



Key Design Principles

1. **Non-Intrusive:** Demo data is isolated using `isDemo` flags
2. **Reusable:** Components can be used across buyer/supplier portals
3. **Extensible:** Easy to add new scenarios and steps
4. **Cinematic:** Automated narration with smooth transitions
5. **Interactive:** Guided mode for hands-on exploration

Components

1. DemoProvider (app/components/demo/demo-context.tsx)

Purpose: Manages global demo state and provides context to all child components.

Key Features:

- Tracks demo mode (active/inactive)
- Manages current step and scenario
- Handles play/pause/stop controls
- Auto-advances steps in cinematic mode

API:

```
interface DemoContextType {
  isDemoMode: boolean;
  isPlaying: boolean;
  scenarioId: string | null;
  currentStepIndex: number;
  currentStep: DemoStep | null;
  mode: "cinematic" | "guided" | null;

  startDemo: (scenarioId: string, mode: "cinematic" | "guided") => void;
  stopDemo: () => void;
  nextStep: () => void;
  prevStep: () => void;
  pauseDemo: () => void;
  resumeDemo: () => void;
  jumpToStep: (stepIndex: number) => void;
}
```

Usage:

```
import { useDemoContext } from '@/app/components/demo/demo-context';

function MyComponent() {
  const { isDemoMode, currentStep, nextStep } = useDemoContext();
  // ...
}
```

2. DemoPlayer (app/components/demo/demo-player.tsx)

Purpose: Executes demo steps (navigation, highlighting, clicks, typing).

Key Features:

- Navigates between routes
- Highlights UI elements
- Simulates user interactions
- Scrolls elements into view

Supported Actions:

- `navigate` : Changes route
- `highlight` : Adds visual highlight to element
- `scrollIntoView` : Scrolls element to center

- `click` : Simulates click on element
 - `type` : Simulates text input
-

3. DemoOverlay (`app/components/demo/demo-overlay.tsx`)

Purpose: Displays visual overlay with controls and narration.

Key Features:

- Top control bar with progress
- Bottom narration box
- Play/pause/stop buttons
- Step navigation (guided mode)
- Role badges (Buyer/Supplier)

Visual Elements:

- Semi-transparent backdrop
 - Pulsing highlight animation
 - Gradient control bars
 - Progress indicator
-

4. DemoButton (`app/components/demo/demo-button.tsx`)

Purpose: Entry point for starting demos from UI.

Key Features:

- Mode selection modal (cinematic/guided)
- Variant support (buyer/supplier)
- Option 3 modal link
- Auto-hides during demo

Props:

```
interface DemoButtonProps {
  variant?: "buyer" | "supplier";
}
```

5. Option3Modal (`app/components/demo/option3-modal.tsx`)

Purpose: Displays baseline vs. upgrade features comparison.

Key Features:

- Two-column layout
- Visual differentiation
- Feature lists with icons
- Call-to-action

Sections:

- Baseline Features (gray)
- AI-Powered Upgrade Features (blue/purple gradient)

6. DemoInitializer (`app/components/demo/demo-initializer.tsx`)

Purpose: Handles URL query parameters for auto-starting demos.

Supported Query Params:

- `?demo=true` - Auto-start demo
- `?scenario=fyndr_full_flow` - Select scenario
- `?mode=cinematic` - Select mode

Example URLs:

```
/dashboard?demo=true&mode=cinematic
/supplier?demo=true&scenario=supplier_only_flow&mode=guided
```

Demo Data

Database Schema Changes

Added `isDemo Boolean @default(false)` to:

- User
- Company
- RFP
- SupplierContact
- SupplierResponse
- SupplierQuestion
- SupplierBroadcastMessage
- ActivityLog

Demo Scenario Data

Buyer User:

- Email: `diane.demo@cloudstack.com`
- Password: `demo123`
- Role: Buyer

Company:

- Name: CloudStack Networks (Demo)
- Description: Leading cloud services provider

Primary RFP:

- Title: Unified Communications & Contact Center RFP – 2025
- Budget: \$500,000
- Stage: PRICING_LEGAL REVIEW
- Priority: HIGH
- 4 competing suppliers

Suppliers:

1. Acme Connect Solutions (Score: 8.5)
2. Northwind Voice Systems (Score: 7.8)

3. Contoso Cloud Communications (Score: 9.1)

4. Fabrikam Unified Solutions (Score: 7.2)

Data Seeding

Functions (`lib/demo/scenario.ts`):

- `createDemoScenarioData()` : Creates fresh demo data
 - `getOrCreateDemoScenario()` : Returns existing or creates new
 - `resetDemoScenario()` : Clears and recreates demo data
-

Demo Scenarios

fyndr_full_flow

Description: Complete end-to-end demonstration from buyer and supplier perspectives.

Duration: ~75 seconds (cinematic mode)

Steps:

1. Buyer Dashboard Introduction
2. Pipeline Widget Highlight
3. Navigate to RFP List
4. Primary RFP Highlight
5. RFP Detail View
6. Timeline Section
7. Supplier Contacts
8. Supplier Responses
9. Switch to Supplier Portal
10. Supplier Dashboard
11. Priority Actions
12. RFP Detail (Supplier View)
13. Submission Progress
14. Quick Actions
15. Return to Buyer Dashboard
16. Final Overview
17. Demo Complete

supplier_only_flow

Description: Focused tour of supplier-facing features.

Duration: ~11 seconds (cinematic mode)

Steps:

1. Supplier Portal Introduction
 2. Dashboard Overview
 3. Explore Prompt
-

Usage Guide

For End Users

Starting a Demo

1. From Buyer Dashboard:

- Click “Watch Demo” button in top header
- Select “Cinematic Mode” or “Guided Mode”
- Sit back and watch (cinematic) or click through steps (guided)

2. From Supplier Portal:

- Click “Watch Demo” button in sidebar or top bar
- Follow same mode selection process

3. Via URL:

- Navigate to `/dashboard?demo=true&mode=cinematic`
- Demo will auto-start after page load

Cinematic Mode

- **Features:**

- Fully automated
- Narrated with text overlays
- Timed step progression
- Pause/resume controls

- **Controls:**

- Pause: Click pause button in top bar
- Resume: Click play button
- Stop: Click stop (square) button

Guided Mode

- **Features:**

- Manual step navigation
- Stay on each step as long as needed
- Full control over progression

- **Controls:**

- Next: Click right arrow or → key
- Previous: Click left arrow or ← key
- Stop: Click stop (square) button

For Administrators

Seeding Demo Data

Manual Seeding:

```
# Via API (requires buyer authentication)
curl -X POST http://localhost:3000/api/demo/seed
```

Reset Demo Data:

```
curl -X POST "http://localhost:3000/api/demo/seed?reset=true"
```

Check Status:

```
curl http://localhost:3000/api/demo/status
```

API Reference

POST /api/demo/seed

Seeds or resets demo data.

Authentication: Required (Buyer role)

Query Parameters:

- reset (optional): If "true", deletes existing demo data first

Response:

```
{
  "success": true,
  "data": {
    "buyerUserId": "uuid",
    "buyerOrgId": "uuid",
    "primaryRfpId": "uuid",
    "secondaryRfpIds": ["uuid1", "uuid2"],
    "supplierContactIds": ["uuid1", "uuid2", "uuid3", "uuid4"],
    "scenarioMetadata": {
      "createdAt": "2024-01-01T00:00:00.000Z",
      "version": "1.0.0"
    }
  }
}
```

GET /api/demo/status

Returns current status of demo data.

Authentication: Required (Buyer role)

Response:

```
{
  "exists": true,
  "demoRfpCount": 3,
  "demoSupplierCount": 4,
  "lastSeeded": "2024-01-01T00:00:00.000Z"
}
```

Customization

Adding New Demo Scenarios

1. Define Scenario in `lib/demo/demo-scenarios.ts`:

```
export const DEMO_SCENARIOS: Record<string, DemoScenarioConfig> = {
  my_new_scenario: {
    id: "my_new_scenario",
    name: "My Custom Demo",
    description: "A custom demonstration",
    steps: [
      {
        id: "step1",
        timeOffsetMs: 0,
        route: "/dashboard",
        action: "navigate",
        text: "Welcome to my custom demo!",
        role: "buyer",
        duration: 3000
      },
      // ... more steps
    ]
  }
};
```

1. Update `DemoButton` (optional) to include new scenario in selection.

Adding New Demo Steps

Step Properties:

```
interface DemoStep {
  id: string; // Unique identifier
  timeOffsetMs: number; // Start time offset (ms)
  targetSelector?: string; // CSS selector for target element
  action?: "highlight" | "click" | "scrollIntoView" | "type" | "navigate";
  text?: string; // Narration text
  role?: "buyer" | "supplier"; // Current role context
  route?: string; // Route for navigation
  duration?: number; // Display duration (ms)
}
```

Example - Highlight a Widget:

```
{
  id: "highlight_pipeline",
  timeOffsetMs: 4000,
  targetSelector: "[data-demo-widget='pipeline']",
  action: "highlight",
  text: "The Pipeline Widget shows all active RFPs.",
  role: "buyer",
  duration: 5000
}
```

Styling Demo Elements

Add `data-demo-*` attributes to components:

```
<div data-demo-widget="pipeline" className="...>
  /* Pipeline Widget Content */
</div>
```

Then target in demo scenarios:

```
targetSelector: "[data-demo-widget='pipeline']"
```

Troubleshooting

Common Issues

Demo Data Not Seeding

Problem: API returns error when seeding demo data.

Solutions:

1. Check database connection
2. Ensure migrations are up-to-date: `npx prisma migrate dev`
3. Check server logs for specific errors
4. Verify buyer authentication

Demo Not Auto-Starting from URL

Problem: URL parameter doesn't start demo.

Solutions:

1. Verify URL format: `?demo=true&mode=cinematic`
2. Check browser console for errors
3. Ensure DemoInitializer is wrapped in Suspense
4. Clear browser cache

Elements Not Highlighting

Problem: Demo step doesn't highlight target element.

Solutions:

1. Verify `targetSelector` is correct
2. Check if element exists in DOM when step executes
3. Add delay before action if element loads async
4. Use browser DevTools to inspect element selector

Demo Steps Executing Too Fast/Slow

Problem: Timing feels off in cinematic mode.

Solutions:

1. Adjust `duration` property for each step
2. Modify `timeOffsetMs` for better pacing
3. Add intermediate steps for smoother transitions

Debug Mode

Enable debug logging:

```
// In demo-player.tsx, add console.logs:  
console.log('Executing step:', currentStep.id, currentStep);  
  
// In demo-context.tsx:  
console.log('Step changed:', currentIndex, currentStep);
```

Best Practices

Creating Effective Demos

1. **Start with Context:** Begin each demo with an introductory step
2. **Show, Don't Tell:** Highlight specific UI elements
3. **Pace Appropriately:** 3-5 seconds per step is typical
4. **Use Role Badges:** Make it clear whether showing buyer or supplier view
5. **End Gracefully:** Include a conclusion step

Writing Narration Text

1. **Be Concise:** 1-2 sentences per step
2. **Action-Oriented:** “Let’s view...”, “Now we’ll...”
3. **Highlight Value:** Explain why feature matters
4. **Avoid Jargon:** Use clear, accessible language

Managing Demo Data

1. **Isolate Demo Data:** Always use `isDemo: true` flag
2. **Regular Resets:** Reset demo data weekly or after major changes
3. **Monitor Performance:** Demo data shouldn’t affect production queries
4. **Backup Strategy:** Document demo data creation process

Option 3: Baseline vs. Upgrade

Baseline Features (Included)

- ✓ Basic RFP creation and management
- ✓ Supplier invitation system
- ✓ Simple response collection
- ✓ Manual evaluation process
- ✓ Basic timeline tracking
- ✓ Email notifications
- ✓ Document upload/download

AI-Powered Upgrade Features (NEW)

- ⚡ AI-Powered Opportunity Scoring
- ⚡ Automated RFP Stage Management with SLA tracking
- ⚡ Smart Supplier Performance Scorecards
- ⚡ AI Response Extraction & Analysis
- ⚡ Intelligent Comparison Engine

- ⚡ Supplier Readiness AI Agent
 - ⚡ Q&A Dialogue System with broadcast messaging
 - ⚡ Comprehensive Activity Logging & Audit Trail
 - ⚡ Advanced Notification Engine with preferences
 - ⚡ Interactive Demo Mode (this feature!)
-

Appendix

File Structure

```

nextjs_space/
├── app/
│   └── components/
│       └── demo/
│           ├── demo-context.tsx
│           ├── demo-player.tsx
│           ├── demo-overlay.tsx
│           ├── demo-button.tsx
│           ├── demo-initializer.tsx
│           └── option3-modal.tsx
└── api/
    └── demo/
        ├── seed/route.ts
        └── status/route.ts
└── dashboard/
    └── dashboard-layout.tsx (integrated)
└── supplier/
    └── supplier-layout.tsx (integrated)
└── providers.tsx (integrated)
lib/
└── demo/
    ├── scenario.ts
    └── demo-scenarios.ts
└── prisma/
    └── schema.prisma (updated)
docs/
└── STEP_32_CINEMATIC_DEMO_ENGINE.md

```

Dependencies

Required:

- Next.js 14+
- React 18+
- Prisma 5+
- NextAuth.js
- Lucide React (icons)

Optional:

- TypeScript (recommended)

Version History

- **v1.0.0** (2024-01): Initial release
- Cinematic and guided modes
- Buyer and supplier demos

- Demo data seeding
 - Option 3 modal
-

Support & Feedback

For issues, enhancements, or questions:

1. Check this documentation first
 2. Review troubleshooting section
 3. Check demo scenario configurations
 4. Review demo data seeding logs
-

License

This demo engine is part of the FYNDR RFP Management System and follows the same license terms.

End of Documentation