# Scientific Project Integration Specification

## Purpose

This document defines the standardized approach for integrating scientific reports and data visualization projects into the React portfolio, ensuring seamless user experience and avoiding integration challenges.

## Project Architecture Requirements

### 1. Directory Structure Standard

```
public/projects/{project-id}/
├── index.html                    # Entry point (built for embedding)
├── data/                         # All data files (.json, .csv, etc.)
├── static/                       # Generated assets (JS, CSS, images)
├── assets/                       # Source images, documents
└── manifest.json                 # Project metadata
```

### 2. Project Manifest Schema

```json
{
  "projectId": "hearing-loss-diabetes",
  "title": "Hearing Loss & Gestational Diabetes Study",
  "description": "Interactive research report analyzing maternal GDM and congenital hearing loss",
  "version": "1.0.0",
  "authors": ["Dr. Example", "Research Team"],
  "type": "scientific-report",
  "entryPoint": "index.html",
  "dataEndpoints": [
    "/data/summary_statistics.json",
    "/data/demographics.json",
    "/data/forest_plot.json"
  ],
  "tags": ["research", "data-viz", "react", "d3js"],
  "technologies": ["React", "D3.js", "TypeScript"],
  "buildSystem": "vite",
  "integration": {
    "routePath": "/projects/hearing-loss-diabetes",
    "embedType": "iframe-with-navigation",
    "dataBaseUrl": "./data",
    "requiresAuth": false,
    "responsive": true
  },
```

```
  "portfolio": {
    "organelleColor": "rgba(59,130,246,0.1)",
    "organelleIcon": "stats",
    "workSectionTitle": "Hearing Loss & Gestational Diabetes Study",
    "workSectionDescription": "Interactive research report analyzing the relationship between materna
  }
}
```

## Data Management Standards

### 3. Data Provider Pattern

```
// Standardized data context for all projects
interface ProjectDataContext {
  projectId: string;
  baseUrl: string;
  dataCache: Map<string, any>;
  loadData: (endpoint: string) => Promise<any>;
  clearCache: () => void;
}

// Implementation requirement
const ProjectDataProvider = ({ children, projectId }) => {
  const baseUrl = `./data`; // Always relative to project root
  // ... implementation
};
```

### 4. Data File Standards

```
// Required data validation interfaces
interface ProjectMetrics {
  total_sample_size: number;
  primary_outcome_cases: number;
  primary_outcome_rate: number;
  study_period: {
    start_date: string;
    end_date: string;
  };
}

interface VisualizationData {
  id: string;
  type: 'forest-plot' | 'demographics' | 'comparison' | 'timeline';
  title: string;
```

```
  data: any[];
  config: {
    width?: number;
    height?: number;
    colorScheme: 'primary' | 'secondary' | 'accent';
    interactive: boolean;
  };
}
```

## Design System Integration

### 5. Scientific Visualization Standards

```
/* Required CSS custom properties for consistency */
:root {
  /* Scientific Color Palette */
  --color-scientific-primary: #3b82f6;      /* Statistical significance */
  --color-scientific-secondary: #64748b;    /* Supporting data */
  --color-scientific-accent: #06b6d4;       /* Categories/demographics */
  --color-scientific-success: #10b981;      /* Positive outcomes */
  --color-scientific-warning: #f59e0b;      /* Caution/attention */
  --color-scientific-danger: #ef4444;       /* Risk factors */

  /* Typography Scale */
  --font-size-hero: clamp(2.5rem, 5vw, 4rem);
  --font-size-h1: clamp(1.875rem, 4vw, 2.5rem);
  --font-size-h2: clamp(1.5rem, 3vw, 2rem);
  --font-size-body: 1rem;
  --font-size-caption: 0.875rem;

  /* Spacing System */
  --space-1: 0.25rem;
  --space-2: 0.5rem;
  --space-3: 0.75rem;
  --space-4: 1rem;
  --space-5: 1.25rem;
  --space-6: 1.5rem;
  --space-8: 2rem;
  --space-10: 2.5rem;

  /* Component Standards */
  --card-radius: 12px;
  --card-shadow: 0 4px 15px rgba(0,0,0,0.1);
  --card-padding: var(--space-6);
```

```
  /* Animation Standards */
  --transition-fast: 0.15s ease-out;
  --transition-medium: 0.3s ease-out;
  --transition-slow: 0.5s ease-out;
}
```

### 6. Component Library Standards

```
// Required reusable component interfaces
interface ScientificCard {
  title: string;
  value: string | number;
  description?: string;
  type: 'primary' | 'secondary' | 'accent';
  interactive?: boolean;
  loading?: boolean;
}

interface DataVisualization {
  data: any[];
  width?: number;
  height?: number;
  type: 'forest-plot' | 'bar-chart' | 'donut-chart' | 'line-chart';
  colorScheme: 'primary' | 'secondary' | 'accent';
  interactive: boolean;
  tooltip?: boolean;
  legend?: boolean;
}

interface StatisticalTable {
  data: any[];
  columns: TableColumn[];
  sortable?: boolean;
  filterable?: boolean;
  pagination?: boolean;
  precision?: number;
}
```

## Build System Requirements

### 7. Vite Configuration Standard

```
// Required vite.config.js for projects
import { defineConfig } from 'vite';
```

```
import react from '@vitejs/plugin-react';

export default defineConfig({
  plugins: [react()],
  base: './',  // CRITICAL: Always use relative paths
  build: {
    outDir: 'build',
    assetsDir: 'static',
    sourcemap: true,
    rollupOptions: {
      output: {
        manualChunks: {
          'd3': ['d3'],
          'react': ['react', 'react-dom']
        }
      }
    }
  },
  define: {
    'process.env.REACT_APP_DATA_BASE_URL': JSON.stringify('./data')
  }
});
```

### 8. Package.json Requirements

```
{
  "name": "{project-id}",
  "scripts": {
    "dev": "vite",
    "build": "vite build",
    "build:integration": "VITE_BASE_URL='./' vite build",
    "preview": "vite preview"
  },
  "dependencies": {
    "react": "^19.0.0",
    "react-dom": "^19.0.0",
    "d3": "^7.9.0"
  },
  "devDependencies": {
    "vite": "^6.2.0",
    "@vitejs/plugin-react": "^4.3.4",
    "typescript": "^5.0.0"
  }
}
```

## Portfolio Integration

### 9. Route Integration Pattern

```
// Portfolio App.jsx integration
const projectRoutes = projects.map(project => (
  <Route
    key={project.projectId}
    path={project.integration.routePath}
    element={
      <Layout>
        <ProjectViewer
          projectId={project.projectId}
          embedType={project.integration.embedType}
        />
      </Layout>
    }
  />
));
```

### 10. Project Viewer Component

```
interface ProjectViewerProps {
  projectId: string;
  embedType: 'iframe-with-navigation' | 'direct-embed' | 'redirect';
}

const ProjectViewer = ({ projectId, embedType }) => {
  const navigate = useNavigate();

  if (embedType === 'iframe-with-navigation') {
    return (
      <div className="project-viewer">
        <ProjectNavigation
          onBack={() => navigate('/work')}
          title={project.title}
        />
        <ProjectIframe
          src={`/projects/${projectId}/index.html`}
          title={project.title}
        />
      </div>
    );
  }
```

```
  // Handle other embed types...
};
```

### 11. Work Section Integration

```
// Auto-generation from project manifests
const generateWorkItems = (projects: ProjectManifest[]) => {
  return projects.map(project => ({
    title: project.portfolio.workSectionTitle,
    description: project.portfolio.workSectionDescription,
    tags: project.tags,
    technologies: project.technologies.join(', '),
    role: project.authors[0],
    link: project.integration.routePath,
    icon: project.portfolio.organelleIcon,
    bg: project.portfolio.organelleColor
  }));
};
```

## Testing & Validation Standards

### 12. Required Test Coverage

```
describe('Scientific Project Integration', () => {
  test('Data loading from standardized endpoints', async () => {
    // Test data fetch from ./data/ path
  });

  test('Responsive design across devices', () => {
    // Test viewport compatibility
  });

  test('Accessibility compliance (WCAG 2.1)', () => {
    // Test screen reader compatibility
  });

  test('Performance benchmarks', () => {
    // Test load times < 3s
    // Test bundle size < 500KB
  });

  test('Cross-browser compatibility', () => {
    // Test Chrome, Firefox, Safari, Edge
```

```
  });
});
```

### 13. Quality Gates

- ☐ Data validation passes
- ☐ All visualizations render correctly
- ☐ Responsive design works on mobile/tablet/desktop
- ☐ Navigation integration functional
- ☐ Performance metrics meet standards
- ☐ Accessibility audit passes
- ☐ Cross-browser testing complete

## Deployment Process

### 14. Build Pipeline

```
# Standard build process for all projects
npm run build:integration
npm run test:integration
npm run validate:manifest
npm run deploy:portfolio
```

### 15. Integration Checklist

- ☐ Project manifest valid
- ☐ Data files in ./data/ directory
- ☐ Build uses relative paths
- ☐ Navigation integration works
- ☐ Performance benchmarks met
- ☐ Accessibility compliance verified
- ☐ Cross-browser testing passed
- ☐ Portfolio organelle updated
- ☐ Route configuration added

## Troubleshooting Guide

### Common Issues & Solutions

1. **Data Loading Failures**
   - Verify data files in ./data/ directory
   - Check REACT_APP_DATA_BASE_URL="./data"

- Validate data file formats

2. **Asset Loading Issues**
   - Ensure base: ‘./’ in vite.config.js
   - Use relative paths for all assets
   - Check static asset directory structure
3. **Navigation Integration**
   - Implement ProjectNavigation component
   - Test back navigation to portfolio
   - Verify route configuration
4. **Performance Issues**
   - Implement code splitting
   - Optimize bundle size
   - Use lazy loading for large datasets

## Agent Implementation Requirements

This specification serves as the foundation for agent-generated projects. Each agent must:

1. **Follow this specification exactly** to ensure integration compatibility
2. **Validate against quality gates** before delivery
3. **Generate standardized project structure** automatically
4. **Test integration** with the portfolio system
5. **Document deviations** and get approval from Coordinator and Scientist agents

### Agent Document-Based Coordination Integration

**Coordinator Agent Responsibilities:**

```
Document Management:
  - Create project status document with integration specifications
  - Monitor agent progress against integration requirements
  - Validate final deliverable meets portfolio compatibility
  - Track integration testing results in real-time

Integration Checkpoints:
  - handoff_queue.push({
      agent: "architect",
      deliverable: "integration_architecture",
      deadline: "{{NOW + 1 day}}",
```

```
      context: {
        spec_version: "v1.2",
        portfolio_requirements: SCIENTIFIC_PROJECT_INTEGRATION_SPEC,
        integration_type: "iframe-with-navigation"
      }
    })

Quality Gates:
  integration_compatibility:
    current_score: 0-100
    requirements:
      manifest_valid: true
      data_paths_correct: true
      navigation_working: true
      performance_benchmarks_met: true
```

**Agent Integration Responsibilities: Scientist Agent:**

```
Document Updates:
  agent_statuses.scientist:
    integration_validation:
      data_quality_approved: boolean
      methodology_documented: boolean
      ethical_compliance_verified: boolean
      scientific_accuracy_validated: boolean
```

**Architect Agent:**

```
Document Updates:
  agent_statuses.architect:
    integration_architecture:
      vite_config_compliant: boolean
      data_provider_implemented: boolean
      component_structure_standardized: boolean
      performance_optimized: boolean

  project_artifacts:
    architecture:
      - vite.config.js
      - data-provider-pattern.ts
      - component-hierarchy.json
```

**Designer Agent:**

```
Document Updates:
  agent_statuses.designer:
    design_integration:
      css_variables_implemented: boolean
      component_library_used: boolean
      accessibility_compliant: boolean
      responsive_design_tested: boolean

  project_artifacts:
    design_assets:
      - design-system.css
      - component-specifications.json
      - accessibility-audit.json
```

**Developer Agent:**

```
Document Updates:
  agent_statuses.developer:
    code_integration:
      spec_compliance_validated: boolean
      data_loading_tested: boolean
      build_process_verified: boolean
      portfolio_compatibility_tested: boolean

  project_artifacts:
    generated_files:
      - package.json
      - manifest.json
      - src/components/**
      - public/data/**
```

**Validator Agent:**

```
Document Updates:
  agent_statuses.validator:
    integration_testing:
      data_loading_tests_passed: boolean
      responsive_design_verified: boolean
      accessibility_compliance_confirmed: boolean
      performance_benchmarks_met: boolean
      cross_browser_compatibility_verified: boolean
      portfolio_navigation_working: boolean

  quality_metrics:
    integration_compatibility: 0-100
```

```
  test_results:
    - test_name: "Data Loading Integration"
      status: "passed" | "failed"
      details: "All endpoints load correctly from ./data/"
    - test_name: "Portfolio Navigation"
      status: "passed" | "failed"
      details: "Back navigation works correctly"
```

**Integration Handoff Pattern:**

```
# Example: Developer → Validator Integration Handoff
Completion Trigger:
  agent_statuses.developer:
    code_completion: 100
    spec_compliance_validated: true
    build_process_verified: true

Coordinator Action:
  handoff_queue.push({
    agent: "validator",
    deliverable: "integration_validation",
    deadline: "{{NOW + 1 day}}",
    dependencies: ["code_implementation"],
    context: {
      integration_spec: SCIENTIFIC_PROJECT_INTEGRATION_SPEC,
      test_checklist: [
        "data_loading_from_relative_paths",
        "portfolio_navigation_integration",
        "responsive_design_validation",
        "accessibility_compliance_check",
        "performance_benchmark_validation"
      ],
      quality_gates: {
        min_integration_compatibility: 100,
        max_load_time: 3000,
        max_bundle_size: 512000
      }
    }
  })

Final Integration Approval:
  project_progress:
    overall_status: "completed"
    integration_verified: true
```

```yaml
  quality_metrics:
    integration_compatibility: 100
  blockers_and_issues: [] # All integration issues resolved
```

---

*This specification prevents integration issues and ensures consistent, high-quality scientific reports that seamlessly integrate with the portfolio system.*