

Museflow v4.0 플랫폼 구축 설계서

AI 기반 박물관 워크플로우 관리 시스템

목차

1. 프로젝트 개요
 2. 시스템 아키텍처
 3. 기술 스택
 4. 데이터베이스 설계
 5. 사용자 인터페이스 설계
 6. 핵심 기능 명세
 7. 캔버스 엔진 설계
 8. 노드 시스템 설계
 9. 인증 및 보안
 10. 배포 및 운영
 11. API 명세
 12. 성능 최적화
 13. 향후 개선 계획
-

1. 프로젝트 개요

1.1 프로젝트 배경

Museflow는 박물관 운영의 복잡한 워크플로우를 시각화하고 자동화하는 AI 기반 플랫폼입니다. 전시 기획, 교육 프로그램, 아카이브 관리 등 박물관의 다양한 업무를 하나의 통합된 플랫폼에서 관리할 수 있습니다.

1.2 주요 목표

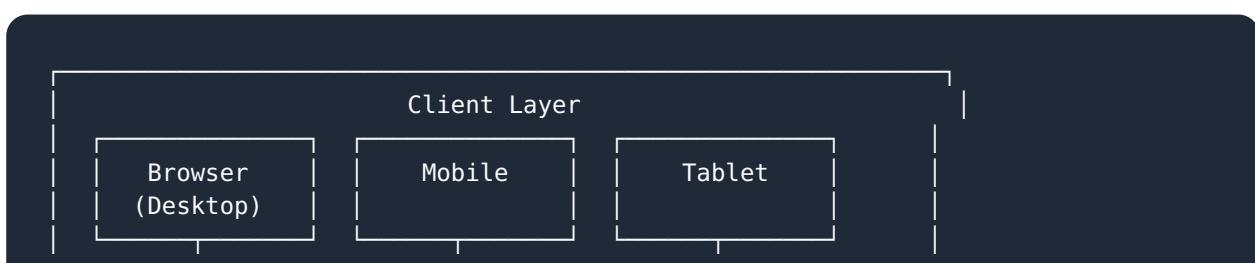
- 워크플로우 시각화: Figma 스타일의 무한 캔버스에서 업무 흐름을 시각적으로 설계
- AI 자동화:** 반복적인 업무를 AI가 자동으로 처리
- 협업 강화: 실시간 협업 및 프로젝트 관리
- 모듈형 설계: 6개 핵심 모듈(전시, 교육, 아카이브, 출판, 연구, 행정)
- 엣지 컴퓨팅: Cloudflare Workers 기반 초고속 글로벌 배포

1.3 핵심 가치

- 직관성: 누구나 쉽게 사용할 수 있는 드래그 앤 드롭 인터페이스
- 확장성: 모듈형 아키텍처로 무한 확장 가능
- 성능: 엣지 컴퓨팅으로 전 세계 어디서나 빠른 속도
- 혁신성: AI 기반 자동화로 업무 효율 극대화

2. 시스템 아키텍처

2.1 전체 아키텍처





2.2 프론트엔드 아키텍처

```

public/
└── static/
    └── css/
        └── design-system.css      # 디자인 시스템 (CSS Variables)
    └── js/
        └── core/
            ├── app.js           # 앱 초기화 및 라우팅
            ├── router.js         # SPA 라우터
            ├── auth.js           # 인증 시스템
            └── canvas-engine.js   # 캔버스 렌더링 엔진
        └── components/
    
```

	└── toast.js	# 토스트 알림
	└── node.js	# 노드 컴포넌트
	└── connection.js	# 연결선 컴포넌트
└── pages/		
└── landing.js	# 랜딩 페이지	
└── login.js	# 로그인 페이지	
└── signup.js	# 회원가입 페이지	
└── features.js	# 기능 소개	
└── content-pages.js	# 모듈/가격/정보 페이지	
└── project-manager.js	# 프로젝트 관리	
└── canvas.js	# 워크플로우 캔버스	
└── canvas-events.js	# 캔버스 이벤트 핸들러	
└── data/		
└── test-data.js	# 테스트 데이터 생성기	
└── logo-icon.png	# 네온 M 아이콘 (59KB)	
└── logo-full.png	# MuseFlow.life 로고 (40KB)	

2.3 백엔드 아키텍처

```

src/
└── index.tsx          # Hono 서버 엔트리포인트

배포 구조:
dist/
└── _worker.js          # Cloudflare Worker 번들
└── _routes.json        # 라우팅 설정
└── static/              # 정적 파일 (CDN)

```

3. 기술 스택

3.1 프론트엔드 기술

기술	버전	용도
JavaScript (Vanilla)	ES6+	핵심 로직 (React/Vue 대신 순수 JS 사용)
HTML5 Canvas	-	워크플로우 시각화
CSS Variables	-	디자인 시스템 구축

기술	버전	용도
LocalStorage API	-	클라이언트 사이드 데이터 저장
Fetch API	-	HTTP 통신

프레임워크 미사용 이유:

- 번들 크기 최소화 (10MB Cloudflare 제한)
- 초고속 로딩 속도
- 완전한 제어권 확보
- 엣지 환경 최적화

3.2 백엔드 기술

기술	버전	용도
Hono	^4.0.0	경량 웹 프레임워크
TypeScript	^5.0.0	타입 안전성
Vite	^5.0.0	빌드 도구
Wrangler	^3.78.0	Cloudflare 배포 CLI

3.3 배포 및 인프라

서비스	용도
Cloudflare Pages	정적 호스팅 + CDN
Cloudflare Workers	엣지 컴퓨팅
Cloudflare D1	(향후) SQLite 데이터베이스
Cloudflare KV	(향후) 키-밸류 저장소
Cloudflare R2	(향후) 파일 저장소

3.4 개발 도구

도구	용도
PM2	프로세스 관리
Git	버전 관리
GitHub	코드 저장소
VSCode	개발 환경

4. 데이터베이스 설계

4.1 LocalStorage 데이터 구조

4.1.1 Users Collection

```
// Key: 'museflow_users'
[
  {
    id: 1000001,                                // Unique user ID
    email: "admin@museflow.com",                  // Email (unique)
    password: "admin123",                         // Plain text (개발용, 실제는 해싱 필요)
    name: "김관리",                             // Display name
    role: "admin",                               // User role
    createdAt: "2024-01-01T00:00:00.000Z"          // ISO timestamp
  },
  // ... more users
]
```

4.1.2 Current User Session

```
// Key: 'museflow_current_user'
{
  id: 1000001,
  email: "admin@museflow.com",
```

```

        name: "김관리"
    }

```

4.1.3 Projects Collection (Per User)

```

// Key: 'museflow_projects_{userId}'
[
{
    id: 1732051200000,                      // Project ID (timestamp)
    title: "2024 특별 전시회 기획",           // Project title
    description: "현대미술 특별전시회...", // Description
    module: "exhibition",                   // Module type
    status: "in-progress",                 // Status
    createdAt: "2024-01-15T10:30:00Z",     // Created timestamp
    updatedAt: "2024-11-20T01:00:00Z",     // Last updated
    progress: 65,                          // Progress (0-100)
    tags: ["urgent", "funded"]            // Tags array
},
// ... more projects
]

```

4.1.4 Canvas Data (Per Project)

```

// Key: 'museflow_canvas_{userId}_{projectId}'
{
    nodes: [
        {
            id: "node_1732051200000_0",          // Unique node ID
            type: "Artwork",                    // Node type
            module: "exhibition",             // Module
            x: 100,                           // X position
            y: 100,                           // Y position
            width: 280,                       // Width
            height: 180,                      // Height
            status: "in-progress",           // Status
            progress: 50,                     // Progress
            inputs: [
                {
                    id: "input-0",
                    label: "Input",
                    connected: true
                }
            ],
            outputs: [                         // Output ports
                {
                    id: "output-0",
                    label: "Output",
                    connected: true
                }
            ]
        }
    ]
}

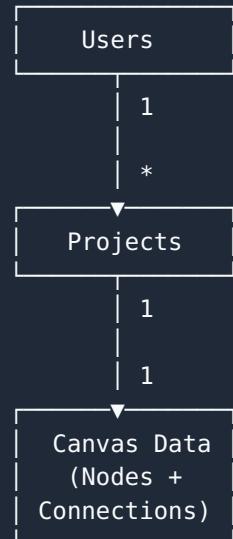
```

```

},
// ... more nodes
],
connections: [
{
  id: "conn_1732051200000_0",           // Connection ID
  sourceNodeId: "node_..._0",            // Source node
  sourceOutputIndex: 0,                 // Source port
  targetNodeId: "node_..._1",            // Target node
  targetInputIndex: 0,                  // Target port
  type: "sequential"                  // Connection type
},
// ... more connections
],
viewport: {
  x: 0,                                // Pan X
  y: 0,                                // Pan Y
  zoom: 1                               // Zoom level
},
lastSaved: "2024-11-20T01:30:00Z"      // Last save timestamp
}

```

4.2 데이터 관계도



4.3 데이터 저장 전략

자동 저장 (Auto-save)

- 간격: 10초마다 자동 저장
- 트리거: 노드 이동, 연결 생성, 상태 변경

- 방식: `localStorage.setItem()` 호출

수동 저장

- 단축키: `Ctrl/Cmd + S`
- 버튼: 상단 툴바의 "Save" 버튼
- 알림: Toast 메시지로 저장 완료 알림

5. 사용자 인터페이스 설계

5.1 디자인 시스템

5.1.1 Color Palette

```
:root {
  /* Primary Colors */
  --primary-50: #f5f3ff;
  --primary-100: #ede9fe;
  --primary-200: #ddd6fe;
  --primary-300: #c4b5fd;
  --primary-400: #a78bfa;
  --primary-500: #8b5cf6;      /* Main brand color */
  --primary-600: #7c3aed;
  --primary-700: #6d28d9;
  --primary-800: #5b21b6;
  --primary-900: #4c1d95;

  /* Grayscale */
  --gray-50: #f9fafb;
  --gray-100: #f3f4f6;
  --gray-200: #e5e7eb;
  --gray-300: #d1d5db;
  --gray-400: #9ca3af;
  --gray-500: #6b7280;
  --gray-600: #4b5563;
  --gray-700: #374151;
  --gray-800: #1f2937;
  --gray-900: #111827;

  /* Semantic Colors */
  --success-500: #10b981;
  --warning-500: #f59e0b;
  --error-500: #ef4444;
```

```
--info-500: #3b82f6;  
}
```

5.1.2 Typography

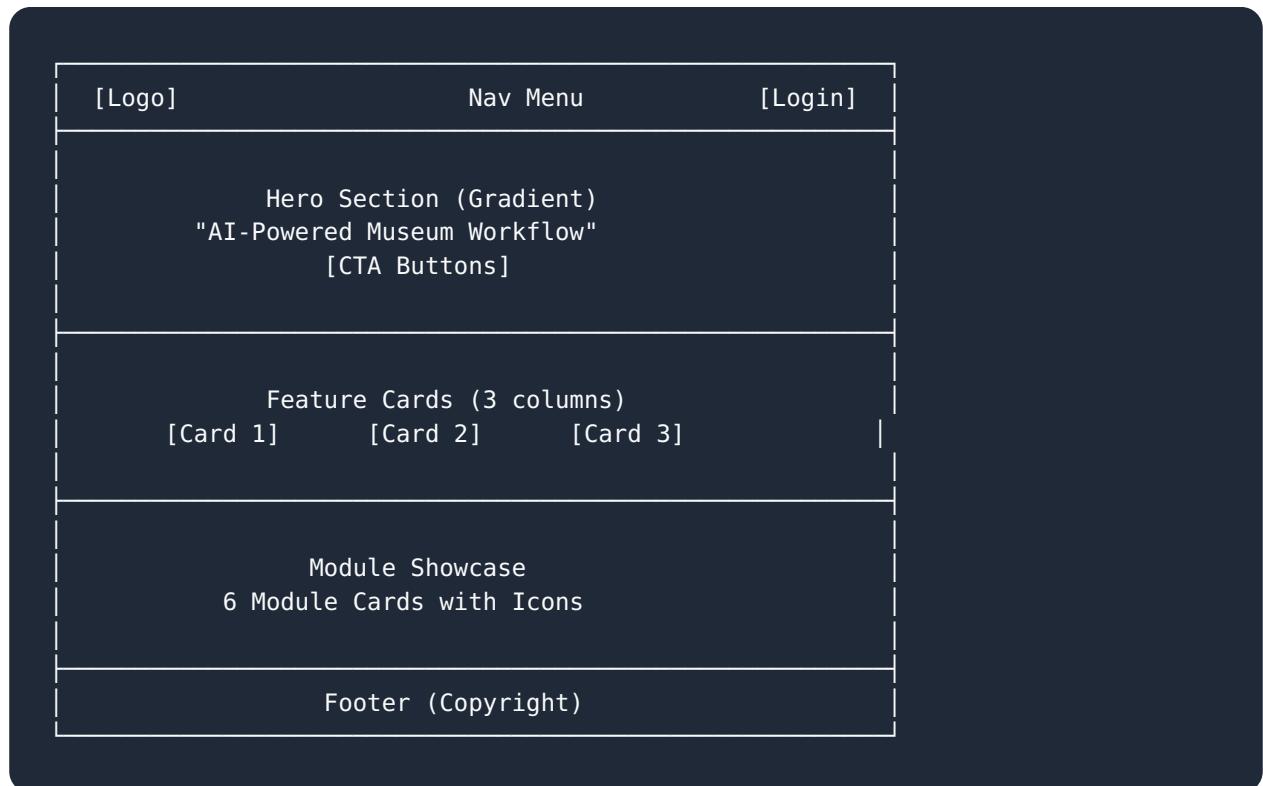
```
:root {  
  /* Font Family */  
  --font-sans: 'Inter', -apple-system, BlinkMacSystemFont, sans-serif;  
  
  /* Font Sizes */  
  --text-xs: 0.75rem; /* 12px */  
  --text-sm: 0.875rem; /* 14px */  
  --text-base: 1rem; /* 16px */  
  --text-lg: 1.125rem; /* 18px */  
  --text-xl: 1.25rem; /* 20px */  
  --text-2xl: 1.5rem; /* 24px */  
  --text-3xl: 1.875rem; /* 30px */  
  --text-4xl: 2.25rem; /* 36px */  
  --text-5xl: 3rem; /* 48px */  
  
  /* Font Weights */  
  --font-light: 300;  
  --font-normal: 400;  
  --font-medium: 500;  
  --font-semibold: 600;  
  --font-bold: 700;  
  --font-extrabold: 800;  
}
```

5.1.3 Spacing

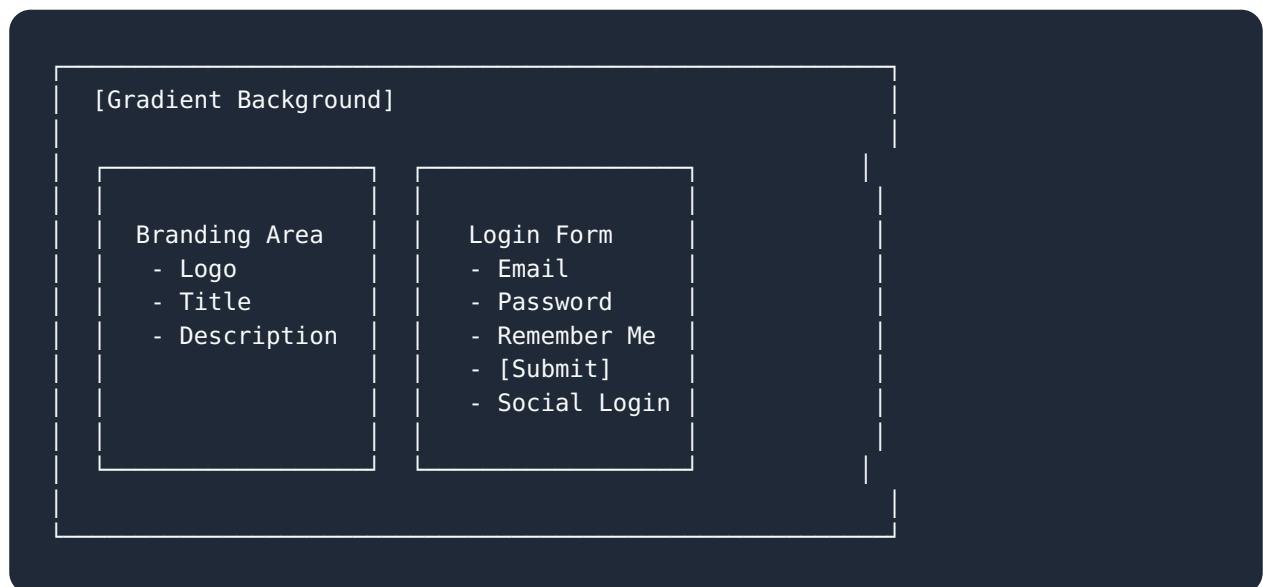
```
:root {  
  --spacing-1: 0.25rem; /* 4px */  
  --spacing-2: 0.5rem; /* 8px */  
  --spacing-3: 0.75rem; /* 12px */  
  --spacing-4: 1rem; /* 16px */  
  --spacing-5: 1.25rem; /* 20px */  
  --spacing-6: 1.5rem; /* 24px */  
  --spacing-8: 2rem; /* 32px */  
  --spacing-10: 2.5rem; /* 40px */  
  --spacing-12: 3rem; /* 48px */  
  --spacing-16: 4rem; /* 64px */  
}
```

5.2 주요 페이지 레이아웃

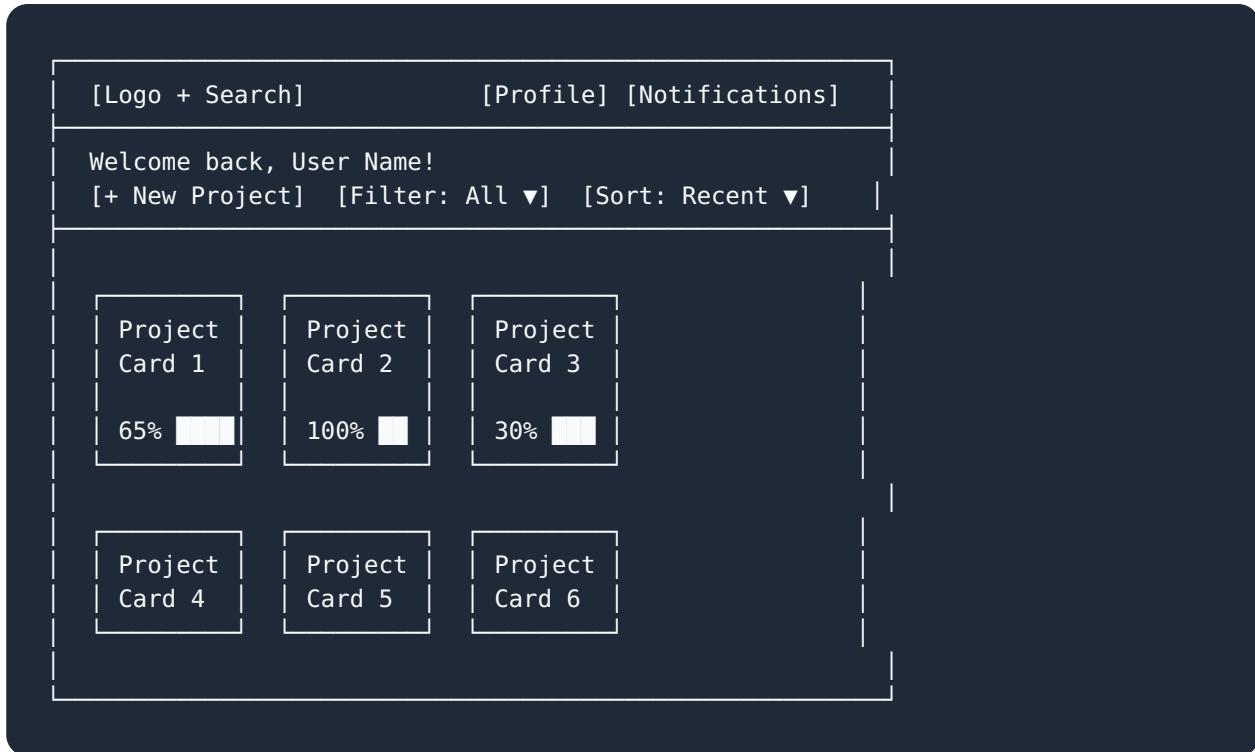
5.2.1 Landing Page



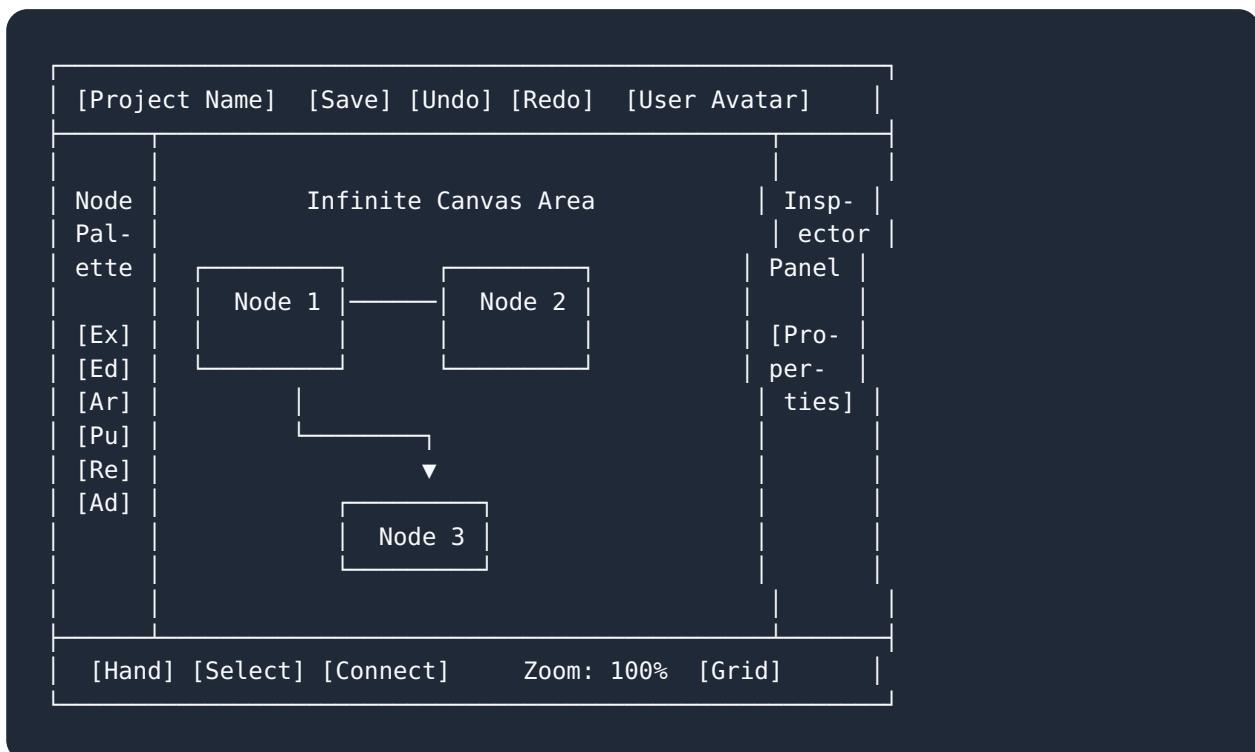
5.2.2 Login/Signup Page



5.2.3 Project Manager Dashboard



5.2.4 Workflow Canvas



5.3 반응형 디자인

Breakpoints

```
/* Mobile First Approach */
--breakpoint-sm: 640px;      /* Mobile landscape */
--breakpoint-md: 768px;      /* Tablet portrait */
--breakpoint-lg: 1024px;     /* Tablet landscape */
--breakpoint-xl: 1280px;     /* Desktop */
--breakpoint-2xl: 1536px;    /* Large desktop */
```

반응형 전략

- **Mobile (< 768px)**: 세로 스택 레이아웃, 햄버거 메뉴
- **Tablet (768-1200px)**: 2열 그리드, 사이드바 축소
- **Desktop (> 1200px)**: 3열 그리드, 전체 기능 표시

6. 핵심 기능 명세

6.1 인증 시스템

6.1.1 회원가입

기능: 새 사용자 계정 생성

입력 필드:

- Full Name (필수, 2-50자)
- Email Address (필수, 이메일 형식)
- Password (필수, 8자 이상)
- Confirm Password (필수, 일치 확인)
- Terms Acceptance (필수, 체크박스)

검증 규칙:

```
{
  name: {
    required: true,
```

```

        minLength: 2,
        maxLength: 50
    },
    email: {
        required: true,
        pattern: /^[^\\s@]+@[^\\s@]+\.[^\\s@]+$/,
        unique: true // DB에서 중복 확인
    },
    password: {
        required: true,
        minLength: 8,
        strength: {
            lowercase: true,
            uppercase: true,
            number: true,
            special: true // Optional
        }
    }
}

```

처리 흐름:

1. 클라이언트 사이드 검증
2. 이메일 중복 확인
3. 사용자 생성 (ID는 timestamp)
4. LocalStorage에 저장
5. 자동 로그인 처리
6. Project Manager로 리다이렉트

비밀번호 강도 표시:

- Weak (빨강): 8자 미만 또는 단순
- Fair (주황): 8자 이상, 소문자만
- Good (초록): 대소문자 + 숫자
- Strong (진한 초록): 대소문자 + 숫자 + 특수문자

6.1.2 로그인

기능: 기존 사용자 인증

입력 필드:

- Email Address
- Password
- Remember Me (선택)

처리 흐름:

```

async function login(email, password) {
    // 1. 사용자 조회

```

```

const users = getAllUsers();
const user = users.find(u =>
  u.email.toLowerCase() === email.toLowerCase() &&
  u.password === password
);

// 2. 인증 실패
if (!user) {
  return {
    success: false,
    error: '이메일 또는 비밀번호가 올바르지 않습니다.'
  };
}

// 3. 세션 저장
localStorage.setItem('museflow_current_user', JSON.stringify({
  id: user.id,
  email: user.email,
  name: user.name
}));

// 4. Remember Me 처리
if (rememberMe) {
  localStorage.setItem('museflow_remember', 'true');
}

// 5. 성공 응답
return {
  success: true,
  user: { id, email, name }
};
}

```

보안 고려사항:

- 비밀번호는 실제 운영환경에서 bcrypt/argon2로 해싱 필요
- HTTPS 통신 필수
- Rate limiting 구현 (로그인 시도 제한)
- Session timeout 구현

6.1.3 로그아웃

기능: 사용자 세션 종료

처리:

```

function logout() {
  localStorage.removeItem('museflow_current_user');
  localStorage.removeItem('museflow_remember');
  navigate('/login');
}

```

6.2 프로젝트 관리

6.2.1 프로젝트 생성

입력 필드:

- Project Title (필수)
- Description (선택)
- Module (필수, 드롭다운)
- Tags (선택, 다중 선택)

프로젝트 구조:

```
{
  id: Date.now(),           // Unique ID
  title: "전시회 기획",
  description: "2024년 특별 전시...",
  module: "exhibition",    // 모듈 타입
  status: "pending",         // 초기 상태
  progress: 0,              // 초기 진행률
  createdAt: new Date().toISOString(),
  updatedAt: new Date().toISOString(),
  tags: ["urgent", "funded"]
}
```

6.2.2 프로젝트 필터링

필터 옵션:

- **Status:** All, Pending, In Progress, Completed
- **Module:** All, Exhibition, Education, Archive, Publication, Research, Administration
- **Tags:** 동적 태그 목록

정렬 옵션:

- Recent (최신순)
- Oldest (오래된순)
- Name A-Z (이름 오름차순)
- Name Z-A (이름 내림차순)
- Progress (진행률순)

6.2.3 프로젝트 검색

기능: 실시간 검색

검색 대상:

- 프로젝트 제목
- 설명
- 태그

구현:

```
function searchProjects(query) {
  const lowerQuery = query.toLowerCase();
  return projects.filter(p =>
    p.title.toLowerCase().includes(lowerQuery) ||
    p.description.toLowerCase().includes(lowerQuery) ||
    p.tags.some(tag => tag.toLowerCase().includes(lowerQuery))
  );
}
```

6.3 워크플로우 캔버스

6.3.1 무한 캔버스 시스템

기능: 무제한 작업 공간

특징:

- 팬(**Pan**): 마우스 드래그로 이동
- 줌(**Zoom**): 마우스 훨 또는 핀치
- 그리드: 20px 기본 그리드, 100px 메인 그리드
- 미니맵: (향후) 전체 캔버스 미리보기

좌표 시스템:

```
// Screen Space → World Space 변환
function screenToWorld(screenX, screenY) {
  return {
    x: (screenX - viewport.x) / viewport.zoom,
    y: (screenY - viewport.y) / viewport.zoom
  };
}

// World Space → Screen Space 변환
function worldToScreen(worldX, worldY) {
  return {
    x: worldX * viewport.zoom + viewport.x,
    y: worldY * viewport.zoom + viewport.y
  };
}
```

줌 레벨:

- Minimum: 25% (0.25x)
- Maximum: 400% (4.0x)
- Default: 100% (1.0x)
- Step: 10% (1.1x factor)

6.3.2 노드 팔레트

구성: 6개 모듈 × 14-15개 노드 = 88+ 노드

표시 방식:

- 모듈별 아코디언
- 드래그 앤 드롭 인터페이스
- 검색 필터

노드 추가 플로우:

1. 팔레트에서 노드 클릭 또는 드래그
2. 캔버스에 드롭 (마우스 위치)
3. 자동 ID 생성
4. 초기 상태 설정
5. 캔버스에 렌더링

6.3.3 노드 연결

연결 타입:

- **Sequential**: 순차적 워크플로우
- **Dependency**: 의존성 관계
- **Reference**: 참조 관계
- **Dataflow**: 데이터 전달

연결 생성:

```
function createConnection(sourceNode, sourcePort, targetNode, targetPort) {
  const connection = {
    id: `conn_${Date.now()}_${Math.random()}`,
    sourceNodeId: sourceNode.id,
    sourceOutputIndex: sourcePort,
    targetNodeId: targetNode.id,
    targetInputIndex: targetPort,
    type: 'sequential'
  };

  // Bezier curve 계산
  const curve = calculateBezierCurve(
    sourceNode.outputs[sourcePort].position,
    targetNode.inputs[targetPort].position
  );
}
```

```

);
connection.curve = curve;
return connection;
}

```

Bezier Curve 렌더링:

```

function drawConnection(ctx, connection) {
  const start = getPortPosition(connection.source);
  const end = getPortPosition(connection.target);

  // Control points
  const distance = Math.abs(end.x - start.x);
  const controlOffset = Math.min(distance * 0.5, 150);

  const cp1 = { x: start.x + controlOffset, y: start.y };
  const cp2 = { x: end.x - controlOffset, y: end.y };

  // Draw curve
  ctx.beginPath();
  ctx.moveTo(start.x, start.y);
  ctx.bezierCurveTo(cp1.x, cp1.y, cp2.x, cp2.y, end.x, end.y);
  ctx.strokeStyle = getConnectionColor(connection.type);
  ctx.lineWidth = 3;
  ctx.stroke();
}

```

6.3.4 노드 인스펙터

표시 정보:

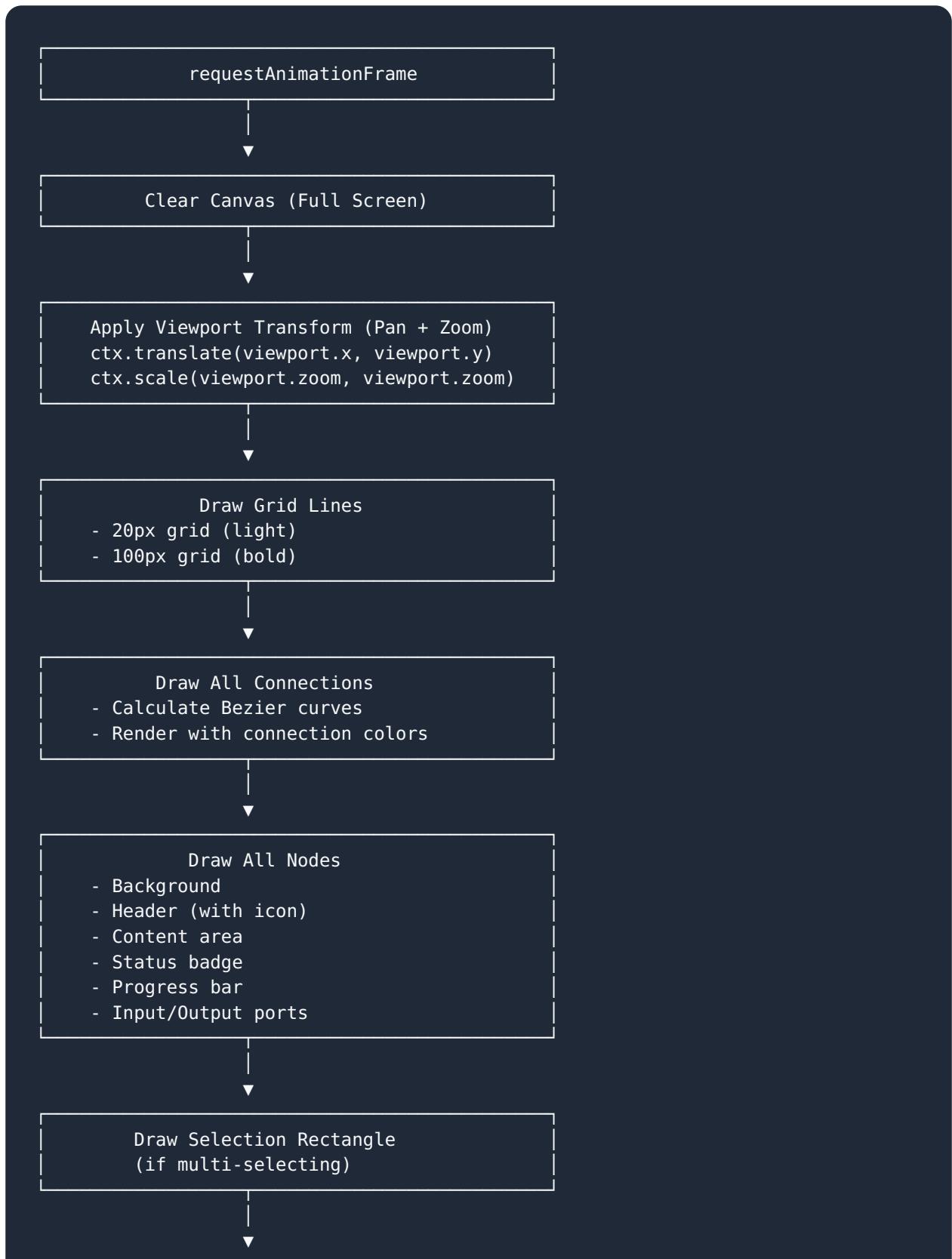
- 노드 타입 및 이름
- 현재 상태 (Pending/In Progress/Completed/Error)
- 진행률 슬라이더 (0-100%)
- 설명 필드 (편집 가능)
- 입출력 포트 목록
- 생성/수정 시간

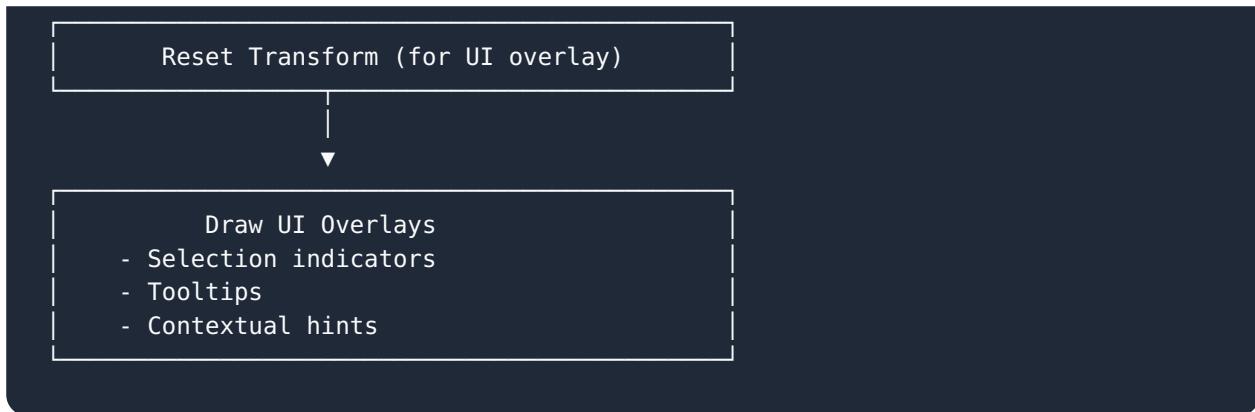
편집 기능:

- 노드 이름 변경
- 상태 업데이트
- 진행률 조정
- 설명 추가/수정

7. 캔버스 엔진 설계

7.1 렌더링 파이프라인





7.2 이벤트 처리 시스템

7.2.1 마우스 이벤트

```

const CanvasEvents = {
  // Mouse down
  handleMouseDown(e) {
    const worldPos = screenToWorld(e.clientX, e.clientY);

    // Check if clicking on node
    const clickedNode = findNodeAt(worldPos.x, worldPos.y);
    if (clickedNode) {
      startDraggingNode(clickedNode);
      return;
    }

    // Check if clicking on port (for connections)
    const clickedPort = findPortAt(worldPos.x, worldPos.y);
    if (clickedPort) {
      startConnection(clickedPort);
      return;
    }

    // Otherwise, start panning
    startPanning(e.clientX, e.clientY);
  },

  // Mouse move
  handleMouseMove(e) {
    if (isDraggingNode) {
      updateNodePosition(e.clientX, e.clientY);
    } else if (isConnecting) {
      updateConnectionPreview(e.clientX, e.clientY);
    } else if (isPanning) {
      updatePan(e.clientX, e.clientY);
    }
  },

  // Mouse up
  handleMouseUp(e) {
  }
};
  
```

```

if (isDraggingNode) {
    finishDraggingNode();
} else if (isConnecting) {
    finishConnection(e.clientX, e.clientY);
} else if (isPanning) {
    finishPanning();
}
},

// Mouse wheel (zoom)
handleWheel(e) {
    e.preventDefault();
    const delta = e.deltaY;
    zoomAt(e.clientX, e.clientY, delta);
}
};

```

7.2.2 키보드 이벤트

```

const KeyboardShortcuts = {
    'Delete': () => deleteSelectedNodes(),
    'Ctrl+C': () => copySelectedNodes(),
    'Ctrl+V': () => pasteNodes(),
    'Ctrl+Z': () => undo(),
    'Ctrl+Y': () => redo(),
    'Ctrl+S': () => saveCanvas(),
    'Ctrl+A': () => selectAllNodes(),
    'Escape': () => deselectAll(),
    'Space': () => toggleHandTool(),
    '+': () => zoomIn(),
    '-': () => zoomOut(),
    '0': () => resetZoom()
};

```

7.3 성능 최적화

7.3.1 렌더링 최적화

```

const OptimizationStrategies = {
    // Culling: 화면 밖 노드는 렌더링 안함
    culling: {
        isNodeVisible(node) {
            const screenPos = worldToScreen(node.x, node.y);
            return (
                screenPos.x + node.width > 0 &&
                screenPos.x < canvas.width &&
                screenPos.y + node.height > 0 &&
                screenPos.y < canvas.height
            );
        }
    }
};

```

```

        );
    }
    },
};

// Dirty Rectangle: 변경된 영역만 다시 그리기
dirtyRect: {
    invalidate(x, y, width, height) {
        dirtyRegions.push({ x, y, width, height });
        requestRender();
    }
},
};

// Level of Detail: 줌 레벨에 따라 디테일 조정
lod: {
    getDetailLevel(zoom) {
        if (zoom < 0.5) return 'low';
        if (zoom < 1.5) return 'medium';
        return 'high';
    }
}
};

```

7.3.2 메모리 관리

```

// Object pooling for frequently created objects
const ObjectPool = {
    connections: [],
    nodes: [],

    getConnection() {
        return this.connections.pop() || new Connection();
    },

    releaseConnection(conn) {
        conn.reset();
        this.connections.push(conn);
    }
};

```

8. 노드 시스템 설계

8.1 노드 타입 정의

8.1.1 전시(Exhibition) 모듈 (15개 노드)

```
const ExhibitionNodes = [
  {
    type: 'Artwork',
    icon: ' ',
    description: '작품 정보 및 상세 데이터',
    inputs: ['concept', 'artist info'],
    outputs: ['display', 'label', 'catalog'],
    aiCapabilities: ['작품 분석', '작가 연구', '전시 배치 제안']
  },
  {
    type: 'Timeline',
    icon: ' ',
    description: '전시 일정 및 마일스톤',
    inputs: ['start date', 'end date'],
    outputs: ['schedule', 'milestones', 'reminders'],
    aiCapabilities: ['일정 최적화', '리스크 예측', '알림 자동화']
  },
  {
    type: 'Gallery',
    icon: ' ',
    description: '전시 공간 정보',
    inputs: ['space info', 'layout'],
    outputs: ['floor plan', 'capacity', 'requirements'],
    aiCapabilities: ['공간 배치 최적화', '동선 분석', '3D 시뮬레이션']
  },
  {
    type: 'Catalog',
    icon: ' ',
    description: '전시 도록 제작',
    inputs: ['artwork data', 'essays'],
    outputs: ['catalog draft', 'images', 'layout'],
    aiCapabilities: ['자동 편집', '번역', '레이아웃 제안']
  },
  {
    type: 'Budget',
    icon: ' ',
    description: '예산 계획 및 관리',
    inputs: ['estimates', 'expenses'],
    outputs: ['budget report', 'forecasts', 'alerts'],
    aiCapabilities: ['비용 예측', '예산 최적화', '재무 리포트']
  },
  // ... 10 more nodes
];
```

8.1.2 교육(Education) 모듈 (14개 노드)

```
const EducationNodes = [
  {
    type: 'Curriculum',
    icon: ' ',
    description: '교육 커리큘럼 설계',
    inputs: ['objectives', 'audience'],
    outputs: ['lesson plan', 'materials', 'assessments'],
    aiCapabilities: ['커리큘럼 생성', '난이도 조정', '개인화 학습']
  },
  {
    type: 'Workshop',
    icon: ' ',
    description: '워크숍 프로그램',
    inputs: ['topic', 'duration', 'materials'],
    outputs: ['schedule', 'guide', 'feedback form'],
    aiCapabilities: ['활동 추천', '자료 생성', '참여도 분석']
  },
  // ... 12 more nodes
];
```

8.1.3 아카이브(Archive) 모듈 (15개 노드)

```
const ArchiveNodes = [
  {
    type: 'Digitization',
    icon: ' ',
    description: '디지털화 작업',
    inputs: ['physical items', 'scan settings'],
    outputs: ['digital files', 'metadata', 'quality report'],
    aiCapabilities: ['자동 메타데이터', '이미지 보정', '품질 검사']
  },
  {
    type: 'Metadata',
    icon: ' ',
    description: '메타데이터 관리',
    inputs: ['item info', 'standards'],
    outputs: ['structured data', 'tags', 'categories'],
    aiCapabilities: ['자동 태깅', '분류', '관계 추출']
  },
  // ... 13 more nodes
];
```

8.1.4 출판(Publication) 모듈 (14개 노드)

```
const PublicationNodes = [
  {
```

```

type: 'Manuscript',
icon: ' ',
description: '원고 작성 및 편집',
inputs: ['draft', 'guidelines'],
outputs: ['edited text', 'comments', 'versions'],
aiCapabilities: ['문법 검사', '스타일 제안', '번역']
},
{
  type: 'Design',
  icon: ' ',
  description: '출판물 디자인',
  inputs: ['content', 'branding'],
  outputs: ['layout', 'mockups', 'print files'],
  aiCapabilities: ['레이아웃 생성', '색상 제안', '폰트 매칭']
},
// ... 12 more nodes
];

```

8.1.5 연구(Research) 모듈 (15개 노드)

```

const ResearchNodes = [
{
  type: 'Hypothesis',
  icon: ' ',
  description: '연구 가설 설정',
  inputs: ['background', 'objectives'],
  outputs: ['hypothesis', 'methodology', 'timeline'],
  aiCapabilities: ['가설 검증', '방법론 제안', '선행연구 분석']
},
{
  type: 'Data Collection',
  icon: ' ',
  description: '데이터 수집 및 정리',
  inputs: ['sources', 'methods'],
  outputs: ['raw data', 'cleaned data', 'documentation'],
  aiCapabilities: ['데이터 정제', '이상치 탐지', '자동 문서화']
},
// ... 13 more nodes
];

```

8.1.6 행정(Administration) 모듈 (15개 노드)

```

const AdministrationNodes = [
{
  type: 'Planning',
  icon: ' ',
  description: '기획 및 계획 수립',
  inputs: ['goals', 'resources'],
  outputs: ['plan', 'timeline', 'kpis'],
  aiCapabilities: ['목표 설정', '일정 최적화', 'KPI 추적']
}
];

```

```

},
{
  type: 'HR',
  icon: ' ',
  description: '인사 관리',
  inputs: ['staff info', 'roles'],
  outputs: ['assignments', 'schedules', 'evaluations'],
  aiCapabilities: ['업무 배분', '스케줄링', '성과 분석']
},
// ... 13 more nodes
];

```

8.2 노드 클래스 구조

```

class Node {
  constructor(type, module, x = 0, y = 0) {
    // Identity
    this.id = `node_${Date.now()}_${Math.random().toString(36).substr(2, 9)}`;
    this.type = type;
    this.module = module;

    // Position & Size
    this.x = x;
    this.y = y;
    this.width = 280;
    this.height = 180;

    // State
    this.status = 'pending'; // pending | in-progress | completed | error
    this.progress = 0; // 0-100
    this.description = '';

    // Connections
    this.inputs = this.generateInputs();
    this.outputs = this.generateOutputs();

    // Metadata
    this.createdAt = new Date().toISOString();
    this.updatedAt = new Date().toISOString();
  }

  // Generate input ports based on node type
  generateInputs() {
    const nodeConfig = NODE_TYPES[this.module].find(n => n.type === this.type);
    return nodeConfig.inputs.map((label, index) => ({
      id: `input-${index}`,
      label: label,
      connected: false,
      position: { x: 0, y: 40 + index * 20 } // Relative position
    }));
  }

  // Generate output ports
}

```

```
generateOutputs() {
  const nodeConfig = NODE_TYPES[this.module].find(n => n.type === this.type);
  return nodeConfig.outputs.map((label, index) => ({
    id: `output-${index}`,
    label: label,
    connected: false,
    position: { x: this.width, y: 40 + index * 20 }
  }));
}

// Render node on canvas
render(ctx, zoom) {
  // Background
  ctx.fillStyle = this.getBackgroundColor();
  ctx.fillRect(this.x, this.y, this.width, this.height);

  // Border
  ctx.strokeStyle = this.getBorderColor();
  ctx.lineWidth = 2;
  ctx.strokeRect(this.x, this.y, this.width, this.height);

  // Header
  this.renderHeader(ctx);

  // Content
  this.renderContent(ctx);

  // Ports
  this.renderPorts(ctx);

  // Status Badge
  this.renderStatusBadge(ctx);

  // Progress Bar
  this.renderProgressBar(ctx);
}

renderHeader(ctx) {
  // Header background
  ctx.fillStyle = this.getModuleColor();
  ctx.fillRect(this.x, this.y, this.width, 40);

  // Icon
  ctx.font = '24px Arial';
  ctx.fillText(this.getIcon(), this.x + 12, this.y + 28);

  // Title
  ctx.fillStyle = '#fff';
  ctx.font = 'bold 14px Inter';
  ctx.fillText(this.type, this.x + 48, this.y + 25);
}

renderContent(ctx) {
  ctx.fillStyle = '#374151';
  ctx.font = '12px Inter';

  // Description (truncated)
}
```

```

const desc = this.description || 'No description';
const truncated = desc.length > 30 ? desc.substr(0, 27) + '...' : desc;
ctx.fillText(truncated, this.x + 12, this.y + 60);
}

renderPorts(ctx) {
  // Input ports
  this.inputs.forEach((input, i) => {
    const y = this.y + 80 + i * 25;
    ctx.fillStyle = input.connected ? '#10b981' : '#9ca3af';
    ctx.beginPath();
    ctx.arc(this.x, y, 6, 0, Math.PI * 2);
    ctx.fill();

    // Label
    ctx.fillStyle = '#6b7280';
    ctx.font = '11px Inter';
    ctx.fillText(input.label, this.x + 12, y + 4);
  });
}

// Output ports
this.outputs.forEach((output, i) => {
  const y = this.y + 80 + i * 25;
  ctx.fillStyle = output.connected ? '#10b981' : '#9ca3af';
  ctx.beginPath();
  ctx.arc(this.x + this.width, y, 6, 0, Math.PI * 2);
  ctx.fill();

  // Label
  ctx.fillStyle = '#6b7280';
  ctx.font = '11px Inter';
  ctx.textAlign = 'right';
  ctx.fillText(output.label, this.x + this.width - 12, y + 4);
  ctx.textAlign = 'left';
});
}

renderStatusBadge(ctx) {
  const badges = {
    'pending': { text: 'Pending', color: '#f59e0b' },
    'in-progress': { text: 'In Progress', color: '#3b82f6' },
    'completed': { text: 'Completed', color: '#10b981' },
    'error': { text: 'Error', color: '#ef4444' }
  };

  const badge = badges[this.status];
  const badgeX = this.x + this.width - 80;
  const badgeY = this.y + 8;

  ctx.fillStyle = badge.color;
  ctx.fillRect(badgeX, badgeY, 70, 20);

  ctx.fillStyle = '#fff';
  ctx.font = '10px Inter';
  ctx.textAlign = 'center';
  ctx.fillText(badge.text, badgeX + 35, badgeY + 14);
  ctx.textAlign = 'left';
}

```

```

}

renderProgressBar(ctx) {
  const barY = this.y + this.height - 12;
  const barWidth = this.width - 24;

  // Background
  ctx.fillStyle = '#e5e7eb';
  ctx.fillRect(this.x + 12, barY, barWidth, 6);

  // Progress
  const progressWidth = (barWidth * this.progress) / 100;
  ctx.fillStyle = '#8b5cf6';
  ctx.fillRect(this.x + 12, barY, progressWidth, 6);
}

getModuleColor() {
  const colors = {
    exhibition: '#8b5cf6',
    education: '#3b82f6',
    archive: '#10b981',
    publication: '#f59e0b',
    research: '#ef4444',
    administration: '#6b7280'
  };
  return colors[this.module] || '#8b5cf6';
}

// Serialize node to JSON
 toJSON() {
  return {
    id: this.id,
    type: this.type,
    module: this.module,
    x: this.x,
    y: this.y,
    width: this.width,
    height: this.height,
    status: this.status,
    progress: this.progress,
    description: this.description,
    inputs: this.inputs,
    outputs: this.outputs,
    createdAt: this.createdAt,
    updatedAt: this.updatedAt
  };
}

// Deserialize from JSON
static fromJSON(data) {
  const node = new Node(data.type, data.module, data.x, data.y);
  Object.assign(node, data);
  return node;
}
}

```

9. 인증 및 보안

9.1 현재 구현 (LocalStorage)

장점

- 빠른 개발 및 테스트
- 서버 불필요
- 완전한 오프라인 작동

단점

- 보안 취약 (비밀번호 평문 저장)
- 다중 기기 동기화 불가
- 브라우저 제한 (5MB)

9.2 향후 보안 강화 방안

9.2.1 비밀번호 해싱

```
// bcrypt 또는 argon2 사용
import bcrypt from 'bcryptjs';

async function hashPassword(password) {
  const salt = await bcrypt.genSalt(10);
  return await bcrypt.hash(password, salt);
}

async function verifyPassword(password, hash) {
  return await bcrypt.compare(password, hash);
}
```

9.2.2 JWT 토큰 인증

```
// 서버 사이드에서 JWT 생성
function generateToken(user) {
  return jwt.sign(
    { id: user.id, email: user.email },
```

```

    process.env.JWT_SECRET,
    { expiresIn: '7d' }
  );
}

// 클라이언트에서 토큰 저장 및 사용
localStorage.setItem('auth_token', token);

// API 요청시 헤더에 포함
fetch('/api/projects', {
  headers: {
    'Authorization': `Bearer ${token}`
  }
});

```

9.2.3 OAuth 2.0 소셜 로그인

```

// Google OAuth 예시
async function googleLogin() {
  const response = await fetch('/auth/google/callback', {
    method: 'POST',
    body: JSON.stringify({ code: authCode })
  });

  const { token, user } = await response.json();
  localStorage.setItem('auth_token', token);
}

```

9.2.4 HTTPS 및 CORS 설정

```

// Cloudflare Worker에서 CORS 설정
app.use('/api/*', cors({
  origin: ['https://museflow.pages.dev'],
  credentials: true
}));

```

10. 배포 및 운영

10.1 개발 환경

10.1.1 로컬 개발 서버

```
# Vite 개발 서버 (Hot Module Replacement)
npm run dev

# Wrangler 로컬 서버 (Cloudflare Workers 환경 시뮬레이션)
npm run dev: sandbox
```

10.1.2 PM2 프로세스 관리

```
// ecosystem.config.cjs
module.exports = {
  apps: [
    {
      name: 'museflow-v4',
      script: 'npx',
      args: 'wrangler pages dev dist --ip 0.0.0.0 --port 3001',
      env: {
        NODE_ENV: 'development',
        PORT: 3001
      },
      watch: false,
      instances: 1,
      exec_mode: 'fork'
    }
  ];
};
```

10.2 빌드 프로세스

```
# 1. TypeScript 컴파일 + Vite 번들링
npm run build

# 결과물:
# dist/
#   └── _worker.js      (30KB - Hono 서버 번들)
#   └── _routes.json    (라우팅 설정)
#     └── static/        (CSS, JS, 이미지)
```

10.3 Cloudflare Pages 배포

10.3.1 초기 설정

```
# 1. Cloudflare API 키 설정
setup.cloudflare_api_key

# 2. 프로젝트 생성
npx wrangler pages project create museflow-v4 \
--production-branch main \
--compatibility-date 2024-01-01

# 3. 배포
npm run deploy
```

10.3.2 배포 워크플로우

```
# 자동화 스크립트
#!/bin/bash

# 1. 빌드
npm run build

# 2. 배포
npx wrangler pages deploy dist --project-name museflow-v4

# 3. 환경 변수 설정 (필요시)
npx wrangler pages secret put API_KEY --project-name museflow-v4

# 4. 배포 확인
curl https://museflow-v4.pages.dev
```

10.3.3 CI/CD (GitHub Actions)

```
# .github/workflows/deploy.yml
name: Deploy to Cloudflare Pages

on:
  push:
    branches: [main]

jobs:
  deploy:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
```

```

- name: Setup Node.js
  uses: actions/setup-node@v3
  with:
    node-version: '18'

- name: Install dependencies
  run: npm ci

- name: Build
  run: npm run build

- name: Deploy to Cloudflare Pages
  uses: cloudflare/pages-action@v1
  with:
    apiToken: ${{ secrets.CLOUDFLARE_API_TOKEN }}
    accountId: ${{ secrets.CLOUDFLARE_ACCOUNT_ID }}
    projectName: museflow-v4
    directory: dist
  
```

10.4 모니터링 및 로깅

10.4.1 Cloudflare Analytics

- 페이지뷰
- 방문자 수
- 응답 시간
- 에러율

10.4.2 커스텀 로깅

```

// 서버 사이드 로깅
app.use('*', async (c, next) => {
  const start = Date.now();
  await next();
  const duration = Date.now() - start;

  console.log({
    method: c.req.method,
    path: c.req.path,
    duration: `${duration}ms`,
    status: c.res.status
  });
});

// 클라이언트 사이드 에러 추적
window.addEventListener('error', (event) => {
  console.error('Global error:', {
    message: event.message,
  });
});
  
```

```
        source: event.filename,
        line: event.lineno,
        column: event.colno
    });
});
```

11. API 명세

11.1 인증 API (LocalStorage 기반)

회원가입

```
POST /api/auth/register (클라이언트 사이드)
```

Request:

```
{
  name: string,
  email: string,
  password: string
}
```

Response:

```
{
  success: boolean,
  user?: {
    id: number,
    email: string,
    name: string
  },
  error?: string
}
```

로그인

```
POST /api/auth/login (클라이언트 사이드)
```

Request:

```
{
  email: string,
  password: string
}
```

```
Response:
{
  success: boolean,
  user?: {
    id: number,
    email: string,
    name: string
  },
  error?: string
}
```

11.2 프로젝트 API

프로젝트 목록 조회

GET /api/projects (클라이언트 사이드)

```
Response:
{
  projects: [
    {
      id: number,
      title: string,
      description: string,
      module: string,
      status: string,
      progress: number,
      createdAt: string,
      updatedAt: string,
      tags: string[]
    }
  ]
}
```

프로젝트 생성

POST /api/projects (클라이언트 사이드)

```
Request:
{
  title: string,
  description: string,
  module: string,
  tags: string[]
}
```

```
Response:
{
  success: boolean,
```

```
project?: { ... }
}
```

11.3 캔버스 API

캔버스 데이터 저장

POST /api/canvas/save (클라이언트 사이드)

Request:

```
{
  projectId: number,
  nodes: Node[],
  connections: Connection[],
  viewport: { x, y, zoom }
}
```

Response:

```
{
  success: boolean,
  lastSaved: string
}
```

12. 성능 최적화

12.1 번들 크기 최적화

현재 번들 크기

```
dist/_worker.js: 30.01 KB (압축 전)
- Hono 프레임워크: ~15KB
- 애플리케이션 코드: ~15KB
```

최적화 전략

1. **Tree Shaking**: 사용하지 않는 코드 제거
2. **Code Splitting**: 페이지별 lazy loading

3. **Minification:** 코드 압축 및 난독화

12.2 렌더링 성능

목표 성능 지표

- **60 FPS:** 캔버스 렌더링
- < **100ms:** 노드 드래그 응답 시간
- < **50ms:** 줌/팬 응답 시간

최적화 기법

1. **RequestAnimationFrame:** 브라우저 리프레시와 동기화
2. **Viewport Culling:** 보이는 영역만 렌더링
3. **Object Pooling:** 객체 재사용

12.3 메모리 관리

메모리 사용량 목표

- < **50MB:** 일반 프로젝트
- < **100MB:** 대형 프로젝트 (100+ 노드)

메모리 누수 방지

```
// 이벤트 리스너 정리
function cleanup() {
  canvas.removeEventListener('mousedown', handleMouseDown);
  canvas.removeEventListener('mousemove', handleMouseMove);
  canvas.removeEventListener('mouseup', handleMouseUp);
  // ... more cleanup
}
```

13. 향후 개선 계획

13.1 Phase 2 - 클라우드 데이터베이스

Cloudflare D1 통합

```
-- Users 테이블
CREATE TABLE users (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    email TEXT UNIQUE NOT NULL,
    password_hash TEXT NOT NULL,
    name TEXT NOT NULL,
    created_at DATETIME DEFAULT CURRENT_TIMESTAMP
);

-- Projects 테이블
CREATE TABLE projects (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    user_id INTEGER NOT NULL,
    title TEXT NOT NULL,
    description TEXT,
    module TEXT NOT NULL,
    status TEXT NOT NULL,
    progress INTEGER DEFAULT 0,
    created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
    updated_at DATETIME DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (user_id) REFERENCES users(id)
);

-- Canvas 테이블
CREATE TABLE canvas_data (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    project_id INTEGER NOT NULL,
    nodes_json TEXT NOT NULL,
    connections_json TEXT NOT NULL,
    viewport_json TEXT NOT NULL,
    updated_at DATETIME DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (project_id) REFERENCES projects(id)
);
```

13.2 Phase 3 - 실시간 협업

WebSocket 기반 실시간 동기화

```
// Cloudflare Durable Objects 활용
class CollaborationRoom {
```

```

constructor(state) {
    this.state = state;
    this.sessions = [];
}

async handleWebSocket(websocket) {
    websocket.accept();
    this.sessions.push(websocket);

    websocket.addEventListener('message', (msg) => {
        // 다른 사용자들에게 브로드캐스트
        this.broadcast(msg.data, websocket);
    });
}

broadcast(message, sender) {
    this.sessions.forEach(session => {
        if (session !== sender) {
            session.send(message);
        }
    });
}
}

```

13.3 Phase 4 - AI 기능 강화

OpenAI/Anthropic API 통합

```

// AI 노드 자동 생성
async function generateWorkflow(description) {
    const response = await fetch('https://api.openai.com/v1/chat/completions', {
        method: 'POST',
        headers: {
            'Authorization': `Bearer ${OPENAI_API_KEY}`,
            'Content-Type': 'application/json'
        },
        body: JSON.stringify({
            model: 'gpt-4',
            messages: [{{
                role: 'user',
                content: `Generate a museum workflow for: ${description}`
            }}]
        })
    });

    const { nodes, connections } = await response.json();
    return { nodes, connections };
}

```

13.4 Phase 5 - 모바일 앱

React Native / Flutter

- iOS/Android 네이티브 앱
- 오프라인 모드
- 푸시 알림
- 카메라 통합 (QR 코드 스캔 등)

부록

A. 테스트 사용자 계정

Email	Password	Role	Name
admin@museflow.com	admin123	Admin	김관리
curator@museflow.com	curator123	Curator	박큐레이터
researcher@museflow.com	research123	Researcher	이연구원
educator@museflow.com	edu123	Educator	최교육
test@museflow.com	test1234	User	테스트 사용자

B. 개발 환경 설정

```
# 1. 프로젝트 클론
git clone https://github.com/username/museflow-v4.git
cd museflow-v4

# 2. 의존성 설치
npm install

# 3. 개발 서버 시작
npm run dev
```

```
# 4. 빌드  
npm run build  
  
# 5. 프로덕션 배포  
npm run deploy
```

C. 디렉토리 구조 전체

```
museflow-v4/  
├── public/  
│   └── static/  
│       ├── css/  
│       │   └── design-system.css  
│       └── js/  
│           ├── core/  
│           │   ├── app.js  
│           │   ├── router.js  
│           │   ├── auth.js  
│           │   └── canvas-engine.js  
│           ├── components/  
│           │   ├── toast.js  
│           │   ├── node.js  
│           │   └── connection.js  
│           └── pages/  
│               ├── landing.js  
│               ├── login.js  
│               ├── signup.js  
│               ├── features.js  
│               ├── content-pages.js  
│               ├── project-manager.js  
│               ├── canvas.js  
│               └── canvas-events.js  
│           └── data/  
│               └── test-data.js  
└── logo-icon.png  
└── logo-full.png  
└── logo.svg  
├── src/  
│   └── index.tsx  
└── dist/  
    ├── _worker.js  
    ├── _routes.json  
    └── static/  
└── .gitignore  
└── ecosystem.config.cjs  
└── package.json  
└── tsconfig.json  
└── vite.config.ts  
└── wrangler.jsonc  
└── README.md
```

D. 주요 npm 스크립트

```
{  
  "scripts": {  
    "dev": "vite",  
    "dev:sandbox": "wrangler pages dev dist --ip 0.0.0.0 --port 3001",  
    "build": "vite build",  
    "preview": "wrangler pages dev dist",  
    "deploy": "npm run build && wrangler pages deploy dist --project-name museflow-v4",  
    "clean-port": "fuser -k 3001/tcp 2>/dev/null || true",  
    "test": "curl http://localhost:3001"  
  }  
}
```

문서 정보

- 버전: 4.0
- 작성일: 2024년 11월 20일
- 작성자: 남현우 교수
- 최종 수정: 2024년 11월 20일
- 상태: 개발 진행 중
- 프로젝트 위치: </home/user/museflow-v4>
- 서비스 URL: <https://3001-i71nxbnvqsj65b78m7n0-2e1b9533.sandbox.novita.ai>