# **Innovata: Complete Project Documentation**

**Project Overview**

**Innovata** is a high-performance React web application designed for the Department of Information Science & Engineering at MIT Mysore. It serves as a comprehensive digital platform for showcasing student projects, announcements, academic guidelines, and department achievements.

**Technology Stack**

**Frontend Framework**

* **React 18.2.0** - Modern UI framework with hooks
* **React Router 6.30.0** - Client-side routing and navigation
* **React Query 4.36.1** - Server state management and data fetching
* **Vite 6.3.5** - Build tool and development server

**Data Processing**

* **PapaParse 5.5.2** - CSV parsing for Google Sheets integration
* **React Icons 5.5.0** - Icon library for UI components

**Development Tools**

* **ESLint 9.25.0** - Code linting and quality
* **rollup-plugin-visualizer** - Bundle size analysis

**Features**

**Core Functionality**

* **Dynamic Content Management** - Real-time updates via Google Sheets
* **Project Showcase** - Student project display with filtering and search
* **Announcements System** - Department news and updates
* **Awards Display** - Student achievements and recognition
* **Academic Resources** - Guidelines and template downloads

**Technical Features**

* **Responsive Design** - Mobile-first approach for all devices
* **Performance Optimized** - 100/100 Lighthouse Performance score
* **Accessibility Compliant** - 98/100 Accessibility rating
* **SEO Optimized** - Perfect SEO scores and meta tags

**Installation & Setup**

**Prerequisites**

* Node.js 18+
* npm/yarn package manager
* Google Sheets for content management

**Installation Steps**

# Clone repository  
https://github.com/R-Pradhyumna/ise-prjct.git  
cd innovate

# Install dependencies  
npm install  
  
# Set environment variables  
echo "VITE\_SHEET\_URL=your-google-sheets-csv-url" > .env  
  
# Start development server  
npm run dev  
  
# Build for production  
npm run build

**Google Sheets Configuration**

Create a Google Sheet with these columns:

* **Announcements**: Announcement Title, Announcement Date, Announcement Summary, Innovata Announcements
* **Projects**: Team No, Project Info, Project Abstract, Scheme, resource link columns
* **Prizes**: Prize Category, Prize Name, Winner Name(s), Year, Prizes
* **Formats**: Phase-1, Phase-2 (containing newline-separated URLs)

**Architecture**

**Project Structure**

src/  
├── components/  
│ ├── home/Home.jsx  
│ ├── announcements/Announcements.jsx  
│ ├── project/Project.jsx  
│ ├── prizes/Prizes.jsx  
│ └── formats/Formats.jsx  
├── pages/  
│ ├── AppLayout.jsx  
│ ├── Header.jsx  
│ └── Footer.jsx  
├── App.jsx  
└── main.jsx

**Data Flow Architecture**

* **Google Sheets** → CSV Export → **PapaParse** → **React Query** → **Components**
* Automatic caching and background updates
* Error handling with retry mechanisms
* Loading states and user feedback

**Component Documentation**

**App.jsx - Application Root**

import { QueryClient, QueryClientProvider } from '@tanstack/react-query';  
  
const queryClient = new QueryClient({  
 defaultOptions: {  
 queries: {  
 staleTime: 5 \* 60 \* 1000, // 5 minutes  
 refetchInterval: 5 \* 60 \* 1000, // Auto-refresh  
 refetchOnWindowFocus: true,  
 },  
 },  
});

**Data Fetching Pattern**

// Promise-based fetcher for React Query  
const fetchData = (csvUrl) => {  
 return new Promise((resolve, reject) => {  
 Papa.parse(csvUrl, {  
 download: true,  
 header: true,  
 skipEmptyLines: true,  
 complete: (results) => {  
 if (results.errors.length > 0) {  
 reject(new Error(`Parsing failed: ${results.errors[^0].message}`));  
 return;  
 }  
 resolve(results.data.filter(row => /\* filtering logic \*/));  
 },  
 error: (err) => reject(new Error(`Fetch failed: ${err.message}`)),  
 });  
 });  
};  
  
// Component implementation  
function Component() {  
 const { data, isLoading, error, refetch, isFetching } = useQuery({  
 queryKey: ['data', sheetUrl],  
 queryFn: () => fetchData(sheetUrl),  
 enabled: !!sheetUrl,  
 });  
  
 // Render with loading/error states  
}

**Performance Optimizations**

// Memoized components  
const ItemComponent = memo(({ data }) => {  
 // Only re-renders when data changes  
});  
  
// Memoized calculations  
const filteredData = useMemo(() =>   
 data.filter(item => item.category === selectedCategory),  
 [data, selectedCategory]  
);  
  
// Stable callbacks  
const handleClick = useCallback((id) => {  
 // Handler logic  
}, [dependency]);

**Configuration**

**Vite Configuration (vite.config.js)**

import { defineConfig } from 'vite';  
import react from '@vitejs/plugin-react';  
import { visualizer } from 'rollup-plugin-visualizer';  
  
export default defineConfig({  
 plugins: [  
 react(),  
 visualizer({ open: true, gzipSize: true })  
 ],  
 build: {  
 rollupOptions: {  
 output: {  
 manualChunks: {  
 'react-vendor': ['react', 'react-dom', 'react-router-dom'],  
 'query-vendor': ['@tanstack/react-query'],  
 'csv-parser': ['papaparse'],  
 },  
 },  
 },  
 },  
});

**Environment Variables**

VITE\_SHEET\_URL=https://docs.google.com/spreadsheets/d/SHEET\_ID/export?format=csv

**Deployment**

**Netlify Configuration (netlify.toml)**

[[redirects]]  
 from = "/\*"  
 to = "/index.html"  
 status = 200  
  
[build]  
 publish = "dist"  
 command = "npm run build"  
  
[build.environment]  
 NODE\_VERSION = "18"

**Build Performance**

* Build time: ~2.19s
* Total bundle size: ~280KB (gzipped)
* Code splitting: ✅ Route-based chunks
* Tree shaking: ✅ Unused code eliminated

**Performance Metrics**

**Lighthouse Scores (Production)**

* **Performance**: 100/100
* **Accessibility**: 98/100
* **Best Practices**: 100/100
* **SEO**: 100/100

**Core Web Vitals**

* **First Contentful Paint**: <1.5s
* **Largest Contentful Paint**: <2.5s
* **Cumulative Layout Shift**: <0.1
* **First Input Delay**: <100ms

**Bundle Analysis**

react-vendor.js 164.61 kB │ gzip: 53.83 kB  
query-vendor.js 38.63 kB │ gzip: 11.87 kB   
csv-parser.js 19.40 kB │ gzip: 7.18 kB  
components/\*.js ~40 kB │ gzip: ~15 kB

**Content Management**

**Google Sheets Integration**

* Real-time content updates without code deployment
* Collaborative editing for multiple administrators
* Version control through Google Sheets history
* Automatic data validation and error handling

**Content Update Workflow**

1. Update content in Google Sheets
2. Changes reflect immediately in the application
3. React Query handles cache invalidation
4. Background updates maintain user experience

**Security & Best Practices**

**Security Measures**

* Content Security Policy (CSP) implementation
* Input sanitization for dynamic content
* HTTPS enforcement in production
* Secure Google Sheets access (view-only public links)

**Accessibility Features**

* WCAG AA compliance
* Semantic HTML structure
* ARIA labels and roles
* Keyboard navigation support
* Screen reader compatibility

**Code Quality**

* ESLint configuration for consistent code style
* Component-level error boundaries
* TypeScript-ready architecture
* Modern React patterns and hooks

**Maintenance & Updates**

**Content Updates**

* Non-technical users can update content via Google Sheets
* No code deployment required for content changes
* Automatic data refresh every 5 minutes
* Manual refresh capability for immediate updates

**Technical Maintenance**

* Dependency updates through npm
* Bundle size monitoring with visualizer
* Performance monitoring via Lighthouse
* Error tracking and user feedback

**Project Statistics**

|  |  |
| --- | --- |
| Metric | Value |
| **Components** | 25+ React components |
| **Bundle Size** | ~280KB (gzipped) |
| **Performance** | 100/100 Lighthouse |
| **Build Time** | 2.19s |
| **Dependencies** | 12 production packages |
| **Test Coverage** | Comprehensive error handling |