```markdown

# ProjectAnalyzer

## Overview

### Introduction

ProjectAnalyzer is a comprehensive web-based tool designed to analyze and visualize project data. It provides developers and project managers with insights into project structure, dependencies, and performance metrics. The tool is built using modern web technologies and is open-source, allowing for customization and extension.

### Purpose and Features

The primary purpose of ProjectAnalyzer is to simplify the process of understanding complex project structures and dependencies. Key features include:

- **Project Structure Visualization**: A graphical representation of the project directory and files.
- **Dependency Analysis**: Detailed breakdown of project dependencies and their versions.
- **Performance Metrics**: Insights into project performance, including build times and resource usage.
- **Interactive Dashboard**: A user-friendly interface for navigating and analyzing project data.

### Intended Users and Use Cases

ProjectAnalyzer is intended for developers, project managers, and anyone involved in software development who needs to understand and optimize project structures. It is particularly useful for:

- Analyzing large-scale projects with complex dependencies.
- Monitoring project performance during development.
- Teaching and demonstrating project structures in educational settings.

## Installation

### Required Software

- Node.js (v16.0.0 or higher)
- Python (for certain dependencies)
- Modern web browser (Chrome, Firefox, Safari)

### Step-by-Step Installation

1. **Clone the Repository**: `bash git clone https://github.com/yourusername/ProjectAnalyzer.git cd ProjectAnalyzer`
2. **Install Dependencies**: `bash npm install`
3. **Install Python Requirements** (if applicable): `bash pip install -r requirements.txt`
4. **Run the Development Server**: `bash npm start`

   The application should be accessible at `http://localhost:3000`.

## Code Structure

### Directory Structure

```
. â”œâ”€â”€ .gitignore â”œâ”€â”€ README.md â”œâ”€â”€ eslint.config.js â”œâ”€â”€ index.html â”œâ”€â”€ package-lock.json â”œâ”€â”€ package.json â”œâ”€â”€ public/ â”‚ â”œâ”€â”€ index.html â”‚ â”‚â”€â”€ ... â”œâ”€â”€ src/ â”‚ â”œâ”€â”€ components/ â”‚ â”‚ â”œâ”€â”€ ProjectTree.js â”‚ â”‚ â”œâ”€â”€ DependencyGraph.js â”‚ â”‚ â”‚â”€â”€ ... â”‚ â”œâ”€â”€ assets/ â”‚ â”‚ â”œâ”€â”€ styles.css â”‚ â”‚ â”‚â”€â”€ ... â”‚ â”œâ”€â”€ App.js â”‚ â”‚â”€â”€ index.js â”‚â”€â”€ vite.config.js
```

### Key Files and Their Roles

| File/Directory | Description | |------------------------|----------------------------------------------| | `index.html` | Entry point for the application. | | `package.json` | Configuration file for npm dependencies. | | `src/App.js` | Main application component. | | `src/components/` |
```

Directory containing React components. | | `src/assets/` | Directory for styles, images, and other assets. | | `vite.config.js` | Configuration file for Vite build tool. |

Features & Functionality

Main Features

1. **Project Structure Visualization**:
   - The tool provides a visual representation of the project directory using a tree structure.
   - Users can expand and collapse directories to view nested files.
2. **Dependency Analysis**:
   - Displays a graph of project dependencies.
   - Shows version information and compatibility status.
3. **Performance Metrics**:
   - Tracks and displays build times, memory usage, and other performance metrics.
   - Provides recommendations for optimization.

Code Snippets

Example of Project Tree Component

```javascript // src/components/ProjectTree.js import React from 'react'; import { FiFolder, FiFile } from 'lucide-react';

const ProjectTree = ({ nodes }) => { return (

{nodes.map((node, index) => (
) : ( )} {node.name} {node.children && ( )}
))}
); };

export default ProjectTree; ```

# API Documentation

## Backend API Endpoints

1. **GET /api/projects**
   - **Description**: Retrieves a list of all projects.
   - **Response**: `json [ { "id": 1, "name": "Project A", "structure": {...}, "dependencies": {...} }, ... ]`
2. **GET /api/projects/:id**
   - **Description**: Retrieves details of a specific project.
   - **Response**: `json { "id": 1, "name": "Project A", "structure": {...}, "dependencies": {...}, "metrics": {...} }`

# Configuration

## Project Settings

- **Environment Variables**: Configure the application by creating a `.env` file in the root directory. `env REACT_APP_API_URL=http://localhost:3000/api`

# Dependencies

| Dependency | Version | |----------------|--------------| | axios | ^1.2.2 | | lucide-react | ^2.0.8 | | react | ^18.2.0 | | react-dom | ^18.2.0 | | react-markdown | ^7.1.3 | | react-router-dom| ^6.2.1 |

# Usage
```

Running the Application

1. **Development Mode**: `bash npm start`
2. **Production Build**: `bash npm run build`
3. **Accessing the Dashboard**:
    - Open `http://localhost:3000` in your browser.
    - Navigate to the dashboard to analyze your projects.

Testing

Running Tests

1. **Unit Tests**: `bash npm test`
2. **Integration Tests**: `bash npm run test:integration`

Deployment

Deployment Guide

1. **Using Vercel**:
    - Install Vercel CLI: `bash npm install -g vercel`
    - Deploy the application: `bash vercel`
2. **Using Netlify**:
    - Log in to your Netlify account.
    - Connect your GitHub repository.
    - Configure the build settings and deploy.
3. **Custom Deployment**:
    - Build the application for production: `bash npm run build`
    - Serve the `dist` directory using your preferred web server.

This documentation provides a comprehensive guide to understanding, installing, configuring, and deploying ProjectAnalyzer. For further assistance, please refer to the project's GitHub repository or contact the maintainers. ```