# **YouTube Video Downloader Project Documentation**

This document provides a comprehensive guide to setting up, running, and understanding your YouTube Video Downloader application.

## **1. Project Overview**

This project consists of a simple web application designed to download YouTube videos by URL. It features a clean, modern frontend built with React and Tailwind CSS, and a robust backend powered by Node.js and Express.js, leveraging the yt-dlp command-line tool for video processing.

**Disclaimer:** This tool is provided for educational and personal use only. Downloading copyrighted material from YouTube without explicit permission from the copyright holder may violate YouTube's Terms of Service and copyright laws. Users are solely responsible for ensuring their use complies with all applicable laws and terms.

## **2. Features**

* **User-Friendly Interface:** A visually appealing and intuitive frontend for pasting YouTube video URLs.
* **Real-time Feedback:** Provides status updates (processing, success, error) to the user.
* **Backend Processing:** Handles the actual video extraction and download logic.
* **Direct Download:** Initiates a direct file download in the browser once the video is processed.
* **Responsive Design:** Adapts to various screen sizes (mobile, tablet, desktop).

## **3. Technologies Used**

### **Frontend**

* **React:** A JavaScript library for building user interfaces.
* **Tailwind CSS:** A utility-first CSS framework for rapid UI development.
* **HTML/CSS/JavaScript:** Standard web technologies.

### **Backend**

* **Node.js:** A JavaScript runtime environment.
* **Express.js:** A fast, unopinionated, minimalist web framework for Node.js.
* **youtube-dl-exec:** An npm package that acts as a wrapper for the yt-dlp command-line tool.
* **yt-dlp (Command-Line Tool):** The core video downloading program, known for its active maintenance and ability to keep up with YouTube's changes.
* **cors:** Node.js middleware for enabling Cross-Origin Resource Sharing.
* **child\_process (Node.js built-in):** Used to spawn and control the yt-dlp command-line process.

## **4. Prerequisites**

Before you begin, ensure you have the following installed on your system:

1. **Node.js and npm (or Yarn):**
   * Download and install from [nodejs.org](https://nodejs.org/). npm (Node Package Manager) is included with Node.js.
   * Verify installation by running:  
     node -v  
     npm -v
2. **yt-dlp (Command-Line Tool):**
   * This is crucial for the backend to function.
   * **Download yt-dlp.exe:** Go to <https://github.com/yt-dlp/yt-dlp/releases> and download the yt-dlp.exe file (for Windows).
   * **Place it in your System PATH:**
     + Create a dedicated folder (e.g., C:\Tools\yt-dlp).
     + Move yt-dlp.exe into this folder.
     + Add this folder's path to your Windows System Environment Variables Path.
       - Search for "Environment Variables" -> "Edit the system environment variables" -> "Environment Variables..." button -> Under "System variables", select "Path" -> "Edit..." -> "New" -> Add the path (e.g., C:\Tools\yt-dlp). Click OK on all windows.
     + **Crucially, close and reopen all Command Prompt windows** after modifying the PATH.
   * **Verify installation:** Open a **new** Command Prompt and run:  
     yt-dlp --version  
       
     It should display the version number. If it says 'yt-dlp' is not recognized, your PATH setup is incorrect.

## **5. Project Structure**

Your project should be organized into two main, separate directories:

Tube Grabr/  
├── youtube-downloader-frontend/  
│ ├── public/  
│ │ └── index.html  
│ ├── src/  
│ │ ├── App.js  
│ │ ├── index.css  
│ │ └── index.js  
│ ├── package.json  
│ ├── package-lock.json  
│ ├── tailwind.config.js  
│ └── postcss.config.js  
└── youtube-downloader-backend/  
 ├── node\_modules/  
 ├── package.json  
 ├── package-lock.json  
 └── server.js

## **6. Setup Instructions**

Follow these steps carefully to set up both your frontend and backend.

### **6.1. Frontend Setup (React App)**

1. **Delete Existing Frontend Folder (if any):**
   * Navigate to C:\Users\91902\Desktop\Tube Grabr\.
   * **Completely delete** the youtube-downloader-frontend folder if it exists.
2. **Create New Frontend Folder:**
   * In C:\Users\91902\Desktop\Tube Grabr\, create a **new, empty folder** named youtube-downloader-frontend.
3. **Create Subfolders:**
   * Inside youtube-downloader-frontend, create two new subfolders: public and src.
4. **Create and Populate Files:**
   * Open VS Code and open the youtube-downloader-frontend folder.
   * Create the following files and paste their exact content (from our previous conversations, or copy them from the provided code block below for convenience):

**youtube-downloader-frontend/package.json**:{  
 "name": "youtube-downloader-frontend",  
 "version": "0.1.0",  
 "private": true,  
 "dependencies": {  
 "react": "^18.2.0",  
 "react-dom": "^18.2.0",  
 "react-scripts": "5.0.1"  
 },  
 "scripts": {  
 "start": "react-scripts start",  
 "build": "react-scripts build",  
 "test": "react-scripts test",  
 "eject": "react-scripts eject"  
 },  
 "eslintConfig": {  
 "extends": [  
 "react-app",  
 "react-app/jest"  
 ]  
 },  
 "browserslist": {  
 "production": [  
 ">0.2%",  
 "not dead",  
 "not op\_mini all"  
 ],  
 "development": [  
 "last 1 chrome version",  
 "last 1 firefox version",  
 "last 1 safari version"  
 ]  
 }  
}  
**youtube-downloader-frontend/public/index.html**:<!DOCTYPE html>  
<html lang="en">  
 <head>  
 <meta charset="utf-8" />  
 <link rel="icon" href="%PUBLIC\_URL%/favicon.ico" />  
 <meta name="viewport" content="width=device-width, initial-scale=1" />  
 <meta name="theme-color" content="#000000" />  
 <meta  
 name="description"  
 content="YouTube Video Downloader Frontend"  
 />  
 <link rel="apple-touch-icon" href="%PUBLIC\_URL%/logo192.png" />  
 <link rel="manifest" href="%PUBLIC\_URL%/manifest.json" />  
 <title>YouTube Downloader</title>  
 <!-- Google Fonts - Inter -->  
 <link href="https://fonts.googleapis.com/css2?family=Inter:wght@400;600;700;800;900&display=swap" rel="stylesheet">  
 </head>  
 <body>  
 <noscript>You need to enable JavaScript to run this app.</noscript>  
 <div id="root"></div>  
 </body>  
</html>  
**youtube-downloader-frontend/src/index.js**:import React from 'react';  
import ReactDOM from 'react-dom/client';  
import './index.css'; // This imports our Tailwind CSS  
import App from './App';  
  
const root = ReactDOM.createRoot(document.getElementById('root'));  
root.render(  
 <React.StrictMode>  
 <App />  
 </React.StrictMode>  
);  
**youtube-downloader-frontend/src/index.css**:@tailwind base;  
@tailwind components;  
@tailwind utilities;  
  
/\* Ensure the body uses the Inter font and takes full height \*/  
body {  
 margin: 0;  
 -webkit-font-smoothing: antialiased;  
 -moz-osx-font-smoothing: grayscale;  
}  
  
code {  
 font-family: source-code-pro, Menlo, Monaco, Consolas, 'Courier New',  
 monospace;  
}  
**youtube-downloader-frontend/tailwind.config.js**:/\*\* @type {import('tailwindcss').Config} \*/  
module.exports = {  
 content: [  
 "./src/\*\*/\*.{js,jsx,ts,tsx}",  
 "./public/index.html",  
 ],  
 theme: {  
 extend: {  
 fontFamily: {  
 inter: ['Inter', 'sans-serif'], // Define 'Inter' font family  
 },  
 },  
 },  
 plugins: [],  
}  
**youtube-downloader-frontend/postcss.config.js**:module.exports = {  
 plugins: {  
 tailwindcss: {},  
 autoprefixer: {},  
 },  
}  
**youtube-downloader-frontend/src/App.js**:import React, { useState } from 'react';  
  
// Mock SVG Icons for demonstration  
const CheckCircle = ({ className }) => (  
 <svg xmlns="http://www.w3.org/2000/svg" width="24" height="24" viewBox="0 0 24 24" fill="none" stroke="currentColor" strokeWidth="2" strokeLinecap="round" strokeLinejoin="round" className={className}>  
 <path d="M22 11.08V12a10 10 0 1 1-5.93-9.14" />  
 <polyline points="22 4 12 14.01 9 11.01" />  
 </svg>  
);  
  
const LinkIcon = ({ className }) => (  
 <svg xmlns="http://www.w3.org/2000/svg" width="24" height="24" viewBox="0 0 24 24" fill="none" stroke="currentColor" strokeWidth="2" strokeLinecap="round" strokeLinejoin="round" className={className}>  
 <path d="M10 13a5 5 0 0 0 7.54.54l3-3a5 5 0 0 0-7.07-7.07L13 7.07a1 1 0 0 1 1.41 1.41L11.41 10a3 3 0 0 1-4.24-4.24l1.41-1.41a1 1 0 0 1 1.41 1.41z" />  
 <path d="M14 11a5 5 0 0 0-7.54-.54l-3 3a5 5 0 0 0 7.07 7.07L11 16.93a1 1 0 0 1-1.41-1.41L12.59 14a3 3 0 0 1 4.24 4.24l-1.41 1.41a1 1 0 0 1-1.41-1.41z" />  
 </svg>  
);  
  
const Loader2 = ({ className }) => <svg xmlns="http://www.w3.org/2000/svg" width="24" height="24" viewBox="0 0 24 24" fill="none" stroke="currentColor" strokeWidth="2" strokeLinecap="round" strokeLinejoin="round" className={className}><path d="M21 12a9 9 0 1 1-6.219-8.56"/></svg>;  
  
  
function App() {  
 const [videoUrl, setVideoUrl] = useState('');  
 const [downloadStatus, setDownloadStatus] = useState(''); // 'idle', 'loading', 'success', 'error'  
 const [errorMessage, setErrorMessage] = useState('');  
  
 const handleDownload = async () => {  
 if (!videoUrl) {  
 setErrorMessage('Please enter a YouTube video URL.');  
 setDownloadStatus('error');  
 return;  
 }  
 setErrorMessage('');  
 setDownloadStatus('loading');  
  
 // This is the key change: Directly open a new window/tab to the backend's GET endpoint.  
 // The backend's /download endpoint will now accept the URL as a query parameter.  
 const downloadLink = `http://localhost:5000/download?url=${encodeURIComponent(videoUrl)}`;  
 window.open(downloadLink, '\_blank'); // Open in new tab to trigger download  
  
 // Simulate success/error for frontend UI, as the actual download is handled by the browser  
 // after window.open. We don't get direct feedback from window.open.  
 setTimeout(() => {  
 const isValidUrl = videoUrl.includes('youtube.com/watch?v=') || videoUrl.includes('youtu.be/');  
 if (isValidUrl) {  
 setDownloadStatus('success');  
 setErrorMessage(''); // Clear any previous error messages  
 } else {  
 setErrorMessage('Invalid YouTube URL. Please check the format.');  
 setDownloadStatus('error');  
 }  
 }, 500); // Short delay for UI feedback  
 };  
  
 const handlePaste = async () => {  
 try {  
 const text = await navigator.clipboard.readText();  
 setVideoUrl(text);  
 setDownloadStatus('idle'); // Reset status on paste  
 setErrorMessage('');  
 } catch (err) {  
 setErrorMessage('Failed to read from clipboard. Please paste manually.');  
 setDownloadStatus('error');  
 }  
 };  
  
 return (  
 <div className="min-h-screen relative overflow-hidden flex items-center justify-center p-4 font-inter bg-gray-900">  
 {/\* Background Gradient with Animation \*/}  
 <style>  
 {`  
 @keyframes backgroundAnimation {  
 0% { background-position: 0% 50%; }  
 50% { background-position: 100% 50%; }  
 100% { background-position: 0% 50%; }  
 }  
  
 .animated-gradient-background {  
 background: linear-gradient(270deg, #6a11cb, #2575fc, #6a11cb);  
 background-size: 600% 600%;  
 animation: backgroundAnimation 20s ease infinite;  
 position: absolute;  
 top: 0;  
 left: 0;  
 width: 100%;  
 height: 100%;  
 z-index: -1;  
 }  
  
 /\* Add a subtle overlay for more depth \*/  
 .animated-gradient-background::before {  
 content: '';  
 position: absolute;  
 top: 0;  
 left: 0;  
 width: 100%;  
 height: 100%;  
 background: radial-gradient(circle at center, rgba(255,255,255,0.1) 0%, rgba(255,255,255,0) 70%);  
 opacity: 0.8;  
 pointer-events: none;  
 }  
  
 /\* Central gradient effect \*/  
 .central-gradient-effect {  
 background: linear-gradient(90deg, #FF0000, #FF7F00, #FFFF00, #00FF00, #0000FF, #4B0082, #9400D3);  
 filter: blur(40px) opacity(0.6); /\* Adjust blur and opacity for desired 'noisy' effect \*/  
 border-radius: 9999px; /\* Fully rounded \*/  
 width: 100%;  
 height: 100%;  
 position: absolute;  
 top: 0;  
 left: 0;  
 }  
 `}  
 </style>  
 <div className="animated-gradient-background"></div>  
  
 <div className="relative z-10 w-full max-w-2xl text-center">  
 {/\* Main Heading \*/}  
 <h1 className="text-5xl md:text-6xl lg:text-7xl font-extrabold mb-8 leading-tight" style={{  
 background: 'linear-gradient(90deg, #FF0000, #FF7F00, #FFFF00, #00FF00, #0000FF, #4B0082, #9400D3)',  
 WebkitBackgroundClip: 'text',  
 WebkitTextFillColor: 'transparent',  
 }}>  
 Download Youtube Video by Link!  
 </h1>  
  
 {/\* Features Section \*/}  
 <div className="flex justify-center items-center space-x-6 mb-12 text-white text-lg md:text-xl">  
 <div className="flex items-center">  
 <CheckCircle className="w-6 h-6 text-green-400 mr-2" />  
 <span>Unlimited Downloads</span>  
 </div>  
 <div className="flex items-center">  
 <CheckCircle className="w-6 h-6 text-green-400 mr-2" />  
 <span>No Watermark</span>  
 </div>  
 </div>  
  
 {/\* Input and Buttons Section \*/}  
 <div className="relative flex flex-col md:flex-row items-center justify-center gap-4 px-4 py-4 rounded-full">  
 {/\* Central gradient effect behind the input and button \*/}  
 <div className="central-gradient-effect"></div>  
  
 <div className="relative flex-grow w-full md:w-auto z-20"> {/\* z-20 to keep content above gradient \*/}  
 <LinkIcon className="absolute left-4 top-1/2 -translate-y-1/2 text-gray-400" />  
 <input  
 type="url"  
 className="w-full pl-12 pr-4 py-4 rounded-full text-white text-lg placeholder-gray-400 focus:outline-none focus:ring-2 focus:ring-blue-500 border border-transparent transition-all duration-300  
 bg-gray-800 bg-opacity-70" /\* Adjusted background for input to see gradient through \*/  
 placeholder="Paste your video link here..."  
 value={videoUrl}  
 onChange={(e) => setVideoUrl(e.target.value)}  
 onFocus={() => {setDownloadStatus('idle'); setErrorMessage('')}}  
 />  
 </div>  
 <button  
 onClick={handleDownload}  
 className="w-full md:w-auto px-8 py-4 rounded-full font-bold text-lg text-white shadow-lg transition-all duration-300 transform hover:scale-105  
 bg-gradient-to-r from-purple-600 to-blue-500 hover:from-purple-700 hover:to-blue-600 flex items-center justify-center z-20" /\* z-20 to keep content above gradient \*/  
 disabled={downloadStatus === 'loading'}  
 >  
 {downloadStatus === 'loading' ? (  
 <>  
 <Loader2 className="animate-spin mr-3" />  
 Processing...  
 </>  
 ) : (  
 'Download'  
 )}  
 </button>  
 </div>  
  
 {/\* Paste Button \*/}  
 <div className="mt-4">  
 <button  
 onClick={handlePaste}  
 className="px-6 py-2 rounded-full bg-gray-700 text-white text-md font-medium hover:bg-gray-600 transition-colors duration-300"  
 >  
 Paste  
 </button>  
 </div>  
  
 {/\* Status Messages \*/}  
 {downloadStatus === 'success' && (  
 <div className="mt-6 p-4 bg-green-500 bg-opacity-20 text-green-300 rounded-xl flex items-center justify-center text-base font-medium shadow-md mx-auto max-w-sm">  
 <CheckCircle className="w-5 h-5 mr-2 text-green-400" />  
 <p>Video link processed! Ready for download options.</p>  
 </div>  
 )}  
  
 {downloadStatus === 'error' && errorMessage && (  
 <div className="mt-6 p-4 bg-red-500 bg-opacity-20 text-red-300 rounded-xl flex items-center justify-center text-base font-medium shadow-md mx-auto max-w-sm">  
 <p>{errorMessage}</p>  
 </div>  
 )}  
 </div>  
 </div>  
 );  
}  
  
export default App;