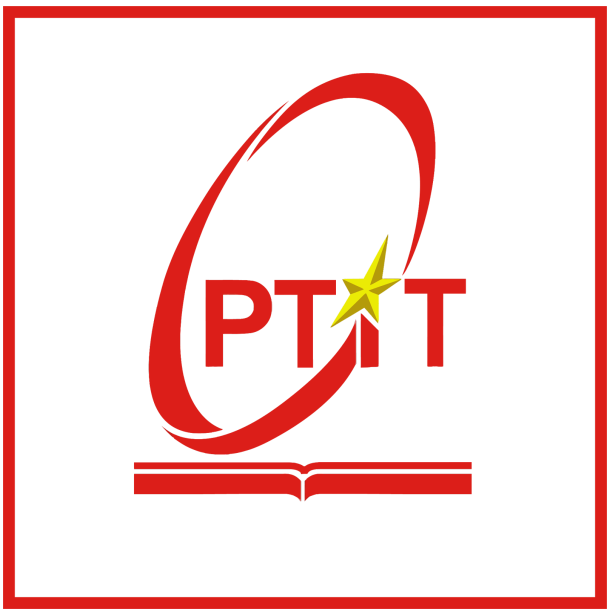
BỘ THÔNG TIN VÀ TRUYỀN THÔNG
HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG



BÁO CÁO THỰC TẬP CƠ SỞ TUẦN 7

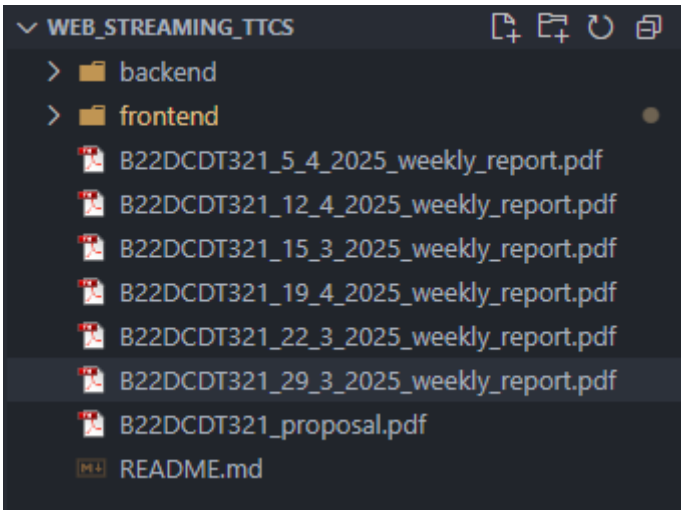
TRIỂN KHAI
CẤU TRÚC THƯ MỤC

Giảng viên hướng dẫn: TS. Kim Ngọc Bách

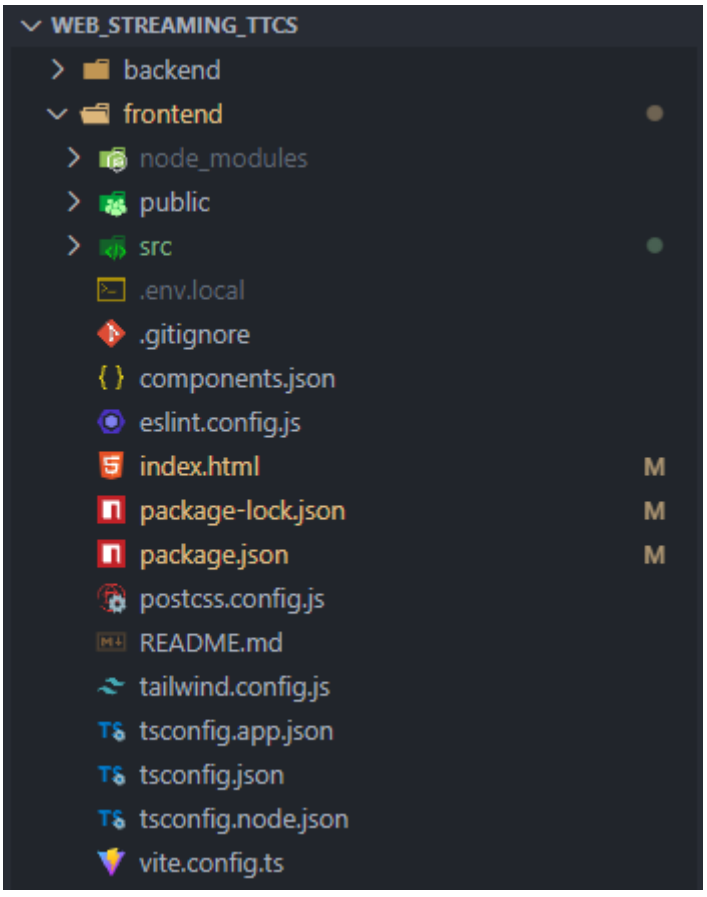
Sinh viên thực hiện:

- Nguyễn Quang Trung - B22DCDT321

- Tạo 2 thư mục backend và frontend



- Cài đặt Vite với framework React và variant TypeScript



- Cài đặt Tailwind css với Vite (phiên bản 3.4.13)

```
> npm install -D tailwindcss@3.4.13 postcss autoprefixer
> npx tailwindcss init -p
```

- Cài đặt shadcn.ui

3 Edit tsconfig.json file

The current version of Vite splits TypeScript configuration into three files, two of which need to be edited. Add the `baseUrl` and `paths` properties to the `compilerOptions` section of the `tsconfig.json` and `tsconfig.app.json` files:

```
{
  "files": [],
  "references": [
    {
      "path": "./tsconfig.app.json"
    },
    {
      "path": "./tsconfig.node.json"
    }
  ],
  "compilerOptions": {
    "baseUrl": ".",
    "paths": {
      "@/*": ["./src/*"]
    }
  }
}
```

4 Edit tsconfig.app.json file

Add the following code to the `tsconfig.app.json` file to resolve paths, for your IDE:

```
{
  "compilerOptions": {
    // ...
    "baseUrl": ".",
    "paths": {
      "@/*": [
        "./src/*"
      ]
    }
    // ...
  }
}
```

5 Update vite.config.ts

Add the following code to the vite.config.ts so your app can resolve paths without error:

```
pnpm npm yarn bun
npm install -D @types/node
```

vite.config.ts

```ts
import path from "path"
import tailwindcss from "@tailwindcss/vite"
import react from "@vitejs/plugin-react"
import { defineConfig } from "vite"

// https://vite.dev/config/
export default defineConfig({
  plugins: [react(), tailwindcss()],
  resolve: {
    alias: {
      "@": path.resolve(__dirname, "./src"),
    },
  },
})
```

6 Run the CLI

Run the `shadcn` init command to setup your project:

```
pnpm npm yarn bun
npx shadcn@latest init
```

You will be asked a few questions to configure `components.json`.

```
Which color would you like to use as base color? › Neutral
```

7 Add Components

You can now start adding components to your project.

```
pnpm npm yarn bun

npx shadcn@latest add button
```

The command above will add the `Button` component to your project. You can then import it like this:

src/App.tsx

```tsx
import { Button } from "@/components/ui/button"

function App() {
  return (
    <div className="flex flex-col items-center justify-center min-h
      <Button>Click me</Button>
    </div>
  )
}

export default App
```

- Cài đặt clerk

2 Install `@clerk/clerk-react`

The Clerk React SDK gives you access to prebuilt components, hooks, and helpers to make user authentication easier.

Run the following command to install the SDK:

```
npm yarn pnpm

terminal

1 npm install @clerk/clerk-react
```

3 Set your Clerk API keys

Add your Clerk Publishable Key to your `.env` file. It can always be retrieved from the **API keys** page in the Clerk Dashboard.

.env — Spotify Clone

```
1  VITE_CLERK_PUBLISHABLE_KEY=pk_test_ZGlzY3JldGUtb3JjYSO2NC5jbGVyay5hY2NvdW5Ocy
```

4 Import the Clerk Publishable Key

In your `main.tsx` file, import your Clerk Publishable Key. You can add an `if` statement to check that it is imported and that it exists. This will prevent running the app without the Publishable Key, and will also prevent TypeScript errors.

src/main.tsx

```tsx
import React from 'react'
import ReactDOM from 'react-dom/client'
import App from './App.tsx'
import './index.css'

// Import your Publishable Key
const PUBLISHABLE_KEY = import.meta.env.VITE_CLERK_PUBLISHABLE_KEY

if (!PUBLISHABLE_KEY) {
  throw new Error('Add your Clerk Publishable Key to the .env file')
}

ReactDOM.createRoot(document.getElementById('root')!).render(
  <React.StrictMode>
    <App />
  </React.StrictMode>,
)
```

5 Add `<ClerkProvider>` to your app

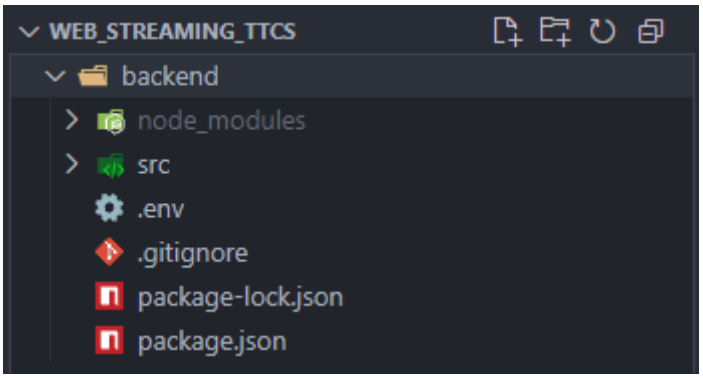
The `<ClerkProvider>` component provides session and user context to Clerk's hooks and components. It's recommended to wrap your entire app at the entry point with `<ClerkProvider>` to make authentication globally accessible. See the reference docs for other configuration options.

Pass your Publishable Key as a prop to the component.

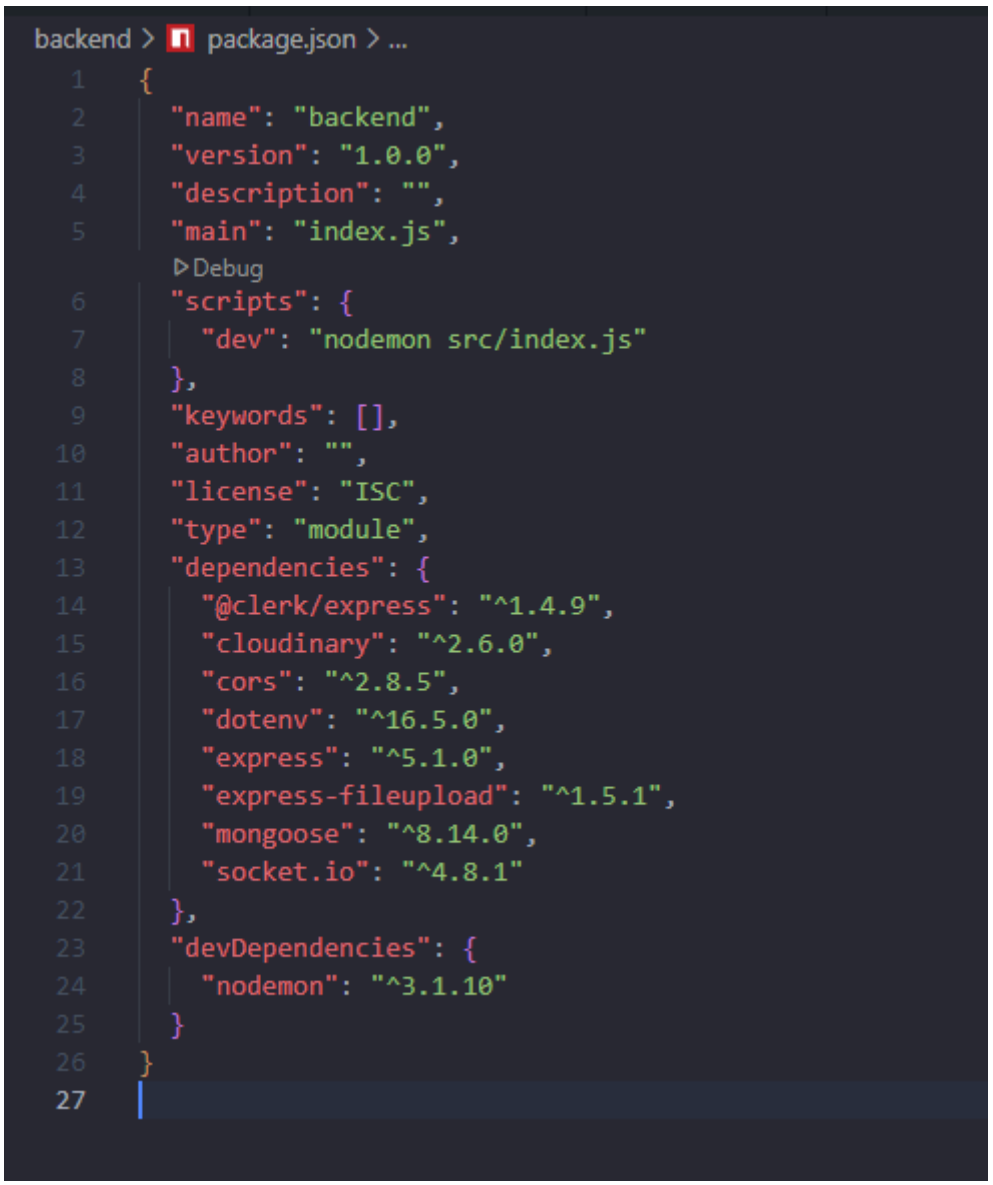
src/main.tsx

```tsx
import React from 'react'
import ReactDOM from 'react-dom/client'
import App from './App.tsx'
import './index.css'
import { ClerkProvider } from '@clerk/clerk-react'

// Import your Publishable Key
const PUBLISHABLE_KEY = import.meta.env.VITE_CLERK_PUBLISHABLE_KEY

if (!PUBLISHABLE_KEY) {
  throw new Error('Add your Clerk Publishable Key to the .env file')
}

ReactDOM.createRoot(document.getElementById('root')!).render(
  <React.StrictMode>
    <ClerkProvider publishableKey={PUBLISHABLE_KEY} afterSignOutUrl="/">
      <App />
    </ClerkProvider>
  </React.StrictMode>,
)
```

- Setup folder backend

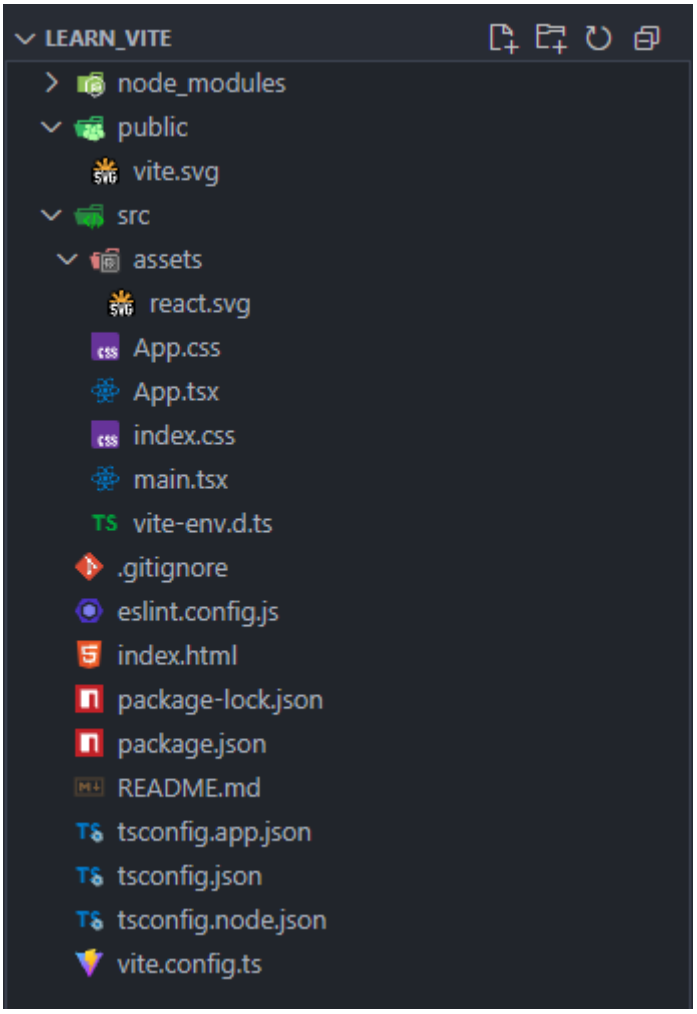


- Cài đặt Nodejs bằng lệnh npm init -y và các thư viện cần thiết



- Tạo file src/index.js làm đầu vào cho server

Cấu trúc thư mục Vite (React + TypeScript)



1. public/ → File tài nguyên tĩnh

Bên trong là file hình ảnh svg logo Vite. Các tài nguyên trong public/ sẽ được serve trực tiếp khi dự án chạy, không qua xử lý của Vite.

2. src/ → Chứa code React

Đây là thư mục chứa toàn bộ mã nguồn chính.

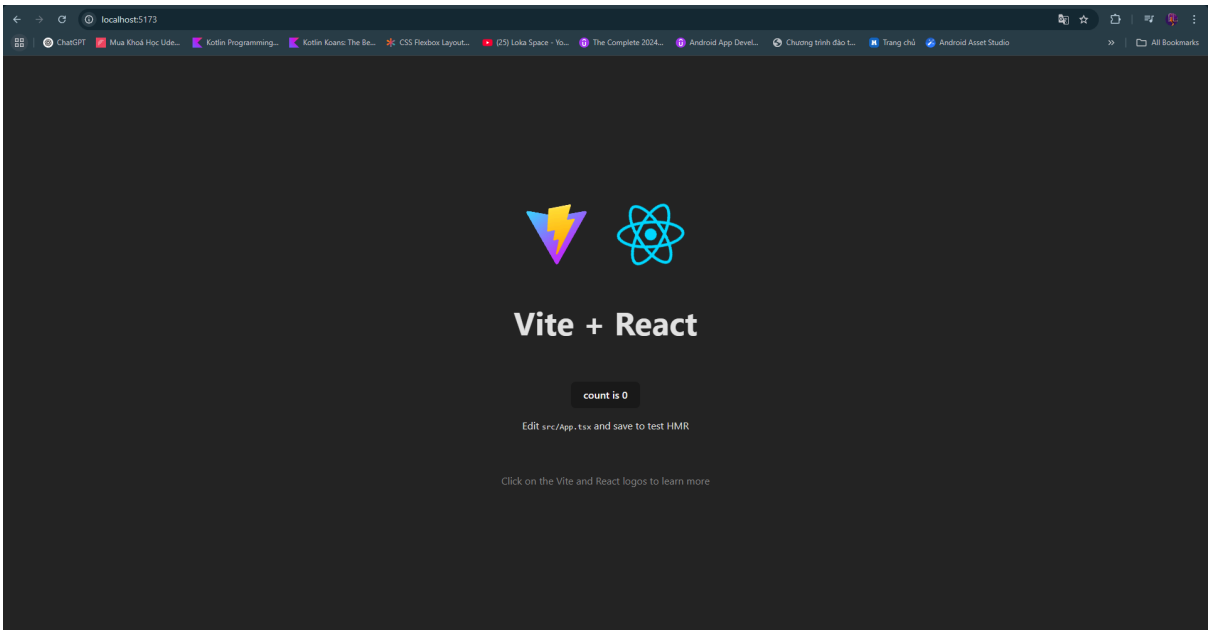
App.tsx	Thành phần chính của React (component), nơi xây dựng giao diện ban đầu.
App.css	

index.css	File css toàn cục, áp dụng cho toàn bộ dự án.
main.tsx	Entry point của ứng dụng React. Nơi ReactDOM.createRoot() được gọi để render App vào index.html.
vite-env.d.ts	File để khai báo các kiểu dữ liệu đặc biệt cho môi trường Vite giúp TypeScript hiểu các file như svg, .env, …

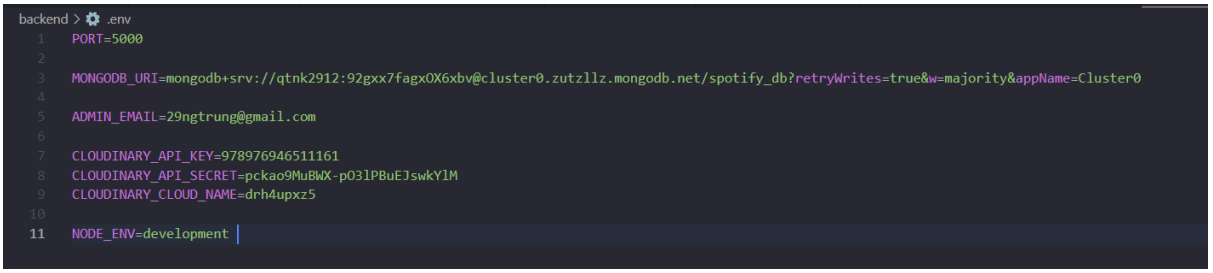
3. Các file cấu hình

Dùng để cấu hình cho Vite, React, TypeScript, ESLint, Node, Git.

- Khi chạy npm run dev



- Cài đặt và kết nối MongoDB



- Model Song (bài hát)

backend > src > models > JS song.model.js > songSchema

```
import mongoose from "mongoose";

const songSchema = new mongoose.Schema(
    {
        title: {
            type: String,
            required: true,
        },
        artist: {
            type: String,
            required: true,
        },
        imageUrl: {
            type: String,
            required: true,
        },
        audioUrl: {
            type: String,
            required: true,
        },
        duration: {
            type: Number,
            required: true,
        },
        albumId: {
            type: mongoose.Schema.Types.ObjectId,
            ref: "Album",
            required: false,
        },
    },
    { timestamps: true }
);

export const Song = mongoose.model("Song", songSchema);

```

- Model Album

backend > src > models > JS album.model.js > ...

```
import mongoose from "mongoose";

const albumSchema = new mongoose.Schema(
    {
        title: { type: String, required: true },
        artist: { type: String, required: true },
        imageUrl: { type: String, required: true },
        releaseYear: { type: Number, required: true },
        songs: [{ type: mongoose.Schema.Types.ObjectId, ref: "Song" }],
    },
    { timestamps: true }
); //  createdAt, updatedAt

export const Album = mongoose.model("Album", albumSchema);

```

- Model User

backend > src > models > JS user.model.js > ...

```
import mongoose from "mongoose";

const userSchema = new mongoose.Schema(
    {
        fullName: {
            type: String,
            required: true,
        },
        imageUrl: {
            type: String,
            required: true,
        },
        clerkId: {
            type: String,
            required: true,
            unique: true,
        },
    },
    { timestamps: true } //  createdAt, updatedAt
);

export const User = mongoose.model("User", userSchema);

```

- Model Message

backend > src > models > JS message.model.js > ...

```js
import mongoose from "mongoose";

const messageSchema = new mongoose.Schema(
    {
        senderId: { type: String, required: true }, // Clerk user ID
        receiverId: { type: String, required: true }, // Clerk user ID
        content: { type: String, required: true },
    },
    { timestamps: true }
);

export const Message = mongoose.model("Message", messageSchema);

```