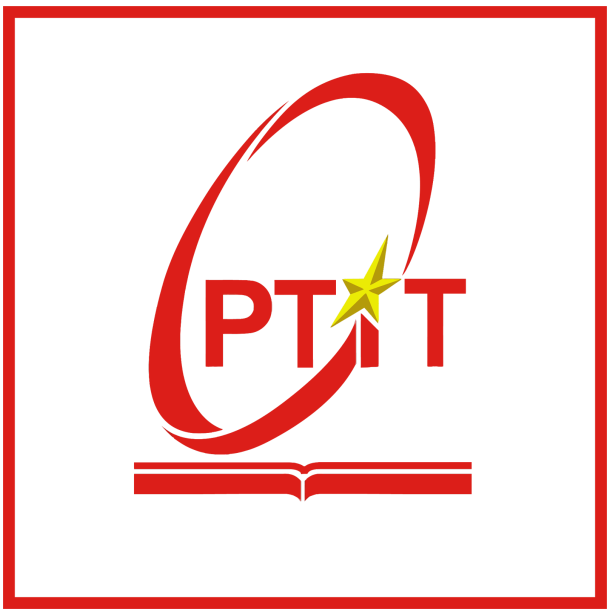
BỘ THÔNG TIN VÀ TRUYỀN THÔNG
HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG



BÁO CÁO THỰC TẬP CƠ SỞ TUẦN 9

TRIỂN KHAI
UX/UI VÀ FRONT END, BACK END
(TIẾP)

Giảng viên hướng dẫn: TS. Kim Ngọc Bách

Sinh viên thực hiện:

- Nguyễn Quang Trung - B22DCDT321

MỤC LỤC

Main Layout

MainLayout là một layout chính của ứng dụng phát nhạc, được thiết kế với 3 phần chính:

Sidebar Trái (LeftSidebar):

- Hiển thị menu điều hướng chính
- Chứa các liên kết đến các trang như Home, Search, Messages
- Hiển thị danh sách playlist của người dùng
- Có thể điều chỉnh kích thước (từ 10% đến 30% chiều rộng)

Nội Dung Chính (Main Content):

- Khu vực hiển thị nội dung chính của ứng dụng
- Sử dụng React Router để chuyển đổi giữa các trang
- Chiếm 60% chiều rộng trên desktop và 80% trên mobile
- Hiển thị các trang như:
  - Trang chủ với các bài hát nổi bật
  - Trang tìm kiếm
  - Trang chi tiết bài hát
  - Trang album

Sidebar Phải (FriendsActivity):

- Hiển thị hoạt động của bạn bè
- Chỉ xuất hiện trên desktop (không hiển thị trên mobile)
- Có thể thu gọn hoàn toàn
- Chiếm 20% chiều rộng mặc định

Các tính năng đặc biệt:

Responsive Design:

- Tự động điều chỉnh layout dựa trên kích thước màn hình
- Ẩn sidebar phải trên thiết bị di động
- Tối ưu hóa không gian hiển thị cho từng thiết bị

Resizable Panels:

- Cho phép người dùng điều chỉnh kích thước các panel
- Có thanh kéo (ResizableHandle) giữa các panel
- Giới hạn kích thước tối thiểu và tối đa cho mỗi panel

Audio Player:

- Thanh phát nhạc luôn hiển thị
- Điều khiển phát nhạc ở dưới cùng (PlaybackControls)

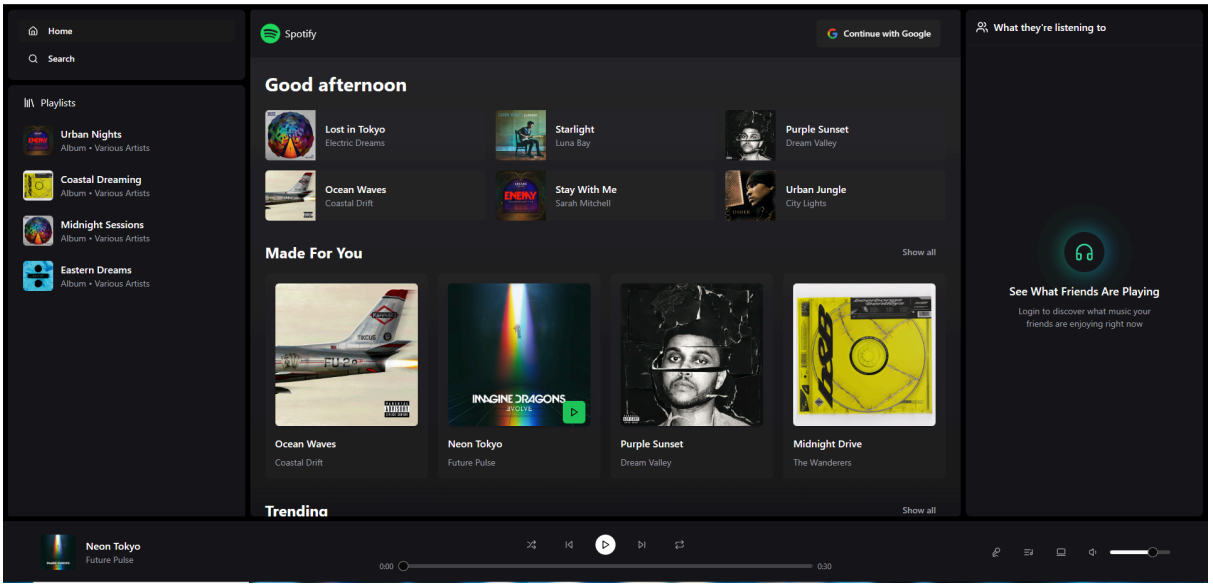
Giao diện:

- Thiết kế tối giản với nền đen
- Chữ màu trắng để tương phản
- Có padding và spacing hợp lý
- Sử dụng flexbox để căn chỉnh các phần tử

Tính năng tương tác:

- Có thể kéo thả để điều chỉnh kích thước
- Chuyển đổi mượt mà giữa các trang
- Hiệu ứng hover và transition mượt mà

Đây là một layout hiện đại, linh hoạt và thân thiện với người dùng, phù hợp cho một ứng dụng phát nhạc chuyên nghiệp.



Giải thích chi tiết MainLayout

1. Container Chính

```
<div className='h-screen bg-black text-white flex
flex-col'>
```

- h-screen: Chiếm toàn bộ chiều cao màn hình

- bg-black: Nền màu đen

- text-white: Chữ màu trắng

- flex flex-col: Sắp xếp các phần tử theo chiều dọc

2. ResizablePanelGroup

```
<ResizablePanelGroup direction='horizontal'
className='flex-1 flex h-full overflow-hidden p-2'>
```

- direction='horizontal': Các panel được sắp xếp theo chiều ngang

- flex-1: Chiếm toàn bộ không gian còn lại

- overflow-hidden: Ẩn phần nội dung tràn

- p-2: Padding 2 đơn vị (8px)

3. AudioPlayer

```
<AudioPlayer />
```

- Component phát nhạc, hiển thị ở đầu layout

4. LeftSidebar

```
<ResizablePanel defaultSize={20} minSize={isMobile ? 0 :
10} maxSize={30}>
   <LeftSidebar />
 </ResizablePanel>
```

- defaultSize={20}: Chiếm 20% chiều rộng mặc định

- minSize={isMobile ? 0 : 10}:

- Trên mobile: có thể thu nhỏ hoàn toàn (0%)

- Trên desktop: tối thiểu 10%

- maxSize={30}: Có thể mở rộng tối đa 30%

5. ResizableHandle

```
<ResizableHandle className='w-2 bg-black rounded-lg
transition-colors' />
```

- w-2: Chiều rộng 2 đơn vị (8px)

- bg-black: Màu nền đen

- rounded-lg: Bo góc

- transition-colors: Hiệu ứng chuyển màu mượt mà

- Dùng để kéo thả điều chỉnh kích thước giữa các panel

6. Main Content

```
<ResizablePanel defaultSize={isMobile ? 80 : 60}>
   <Outlet />
 </ResizablePanel>
```

- defaultSize={isMobile ? 80 : 60}:

- Trên mobile: chiếm 80% chiều rộng

- Trên desktop: chiếm 60% chiều rộng

- <Outlet />: Render nội dung của route hiện tại

7. RightSidebar (FriendsActivity)

```
{!isMobile && (
   <>
   <ResizableHandle className='w-2 bg-black rounded-lg
transition-colors' />
   <ResizablePanel defaultSize={20} minSize={0}
maxSize={25} collapsedSize={0}>
        <FriendsActivity />
   </ResizablePanel>
```

```
    </>
  )}
```

- Chỉ hiển thị khi không phải mobile (!isMobile)
- defaultSize={20}: Chiếm 20% chiều rộng mặc định
- minSize={0}: Có thể thu nhỏ hoàn toàn
- maxSize={25}: Có thể mở rộng tối đa 25%
- collapsedSize={0}: Khi thu gọn sẽ có kích thước 0

8. PlaybackControls

```
<PlaybackControls />
```

- Thanh điều khiển phát nhạc ở dưới cùng
- Hiển thị các nút điều khiển như play/pause, next/previous, volume

→ Khi mở ứng dụng:

- Layout sẽ tự động chia không gian theo tỷ lệ mặc định
- Trên desktop: 20% (left) - 60% (main) - 20% (right)
- Trên mobile: 0% (left) - 80% (main) - 0% (right)

→ Khi resize màn hình:

- Tự động chuyển đổi giữa layout mobile và desktop
- Ẩn/hiện sidebar phải
- Điều chỉnh kích thước các panel

→ Khi kéo thả:

- Người dùng có thể điều chỉnh kích thước các panel
- Có giới hạn min/max để đảm bảo trải nghiệm tốt
- Các panel khác sẽ tự động điều chỉnh để phù hợp

Backend

Các routers:

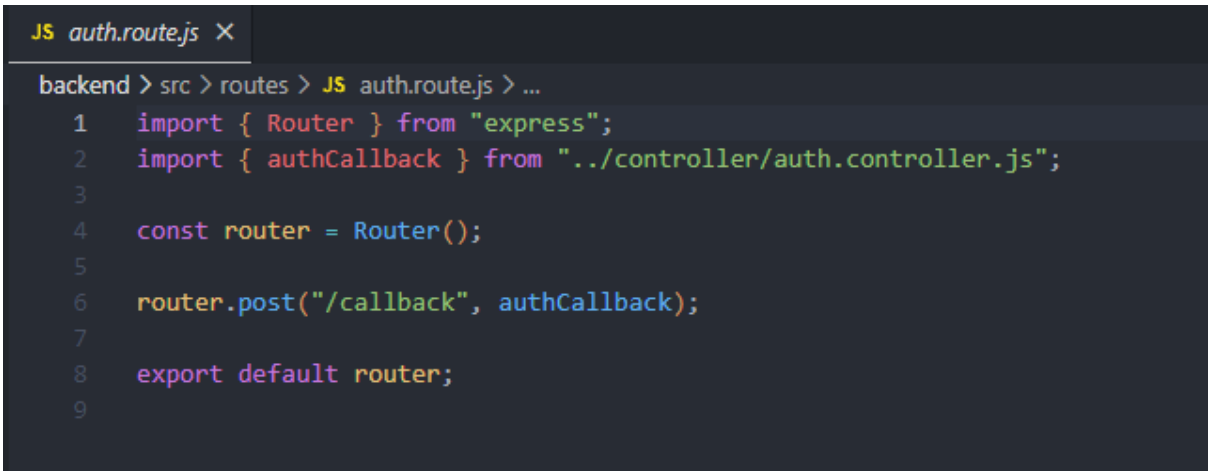
- Route admin

```
JS admin.route.js ×
backend > src > routes > JS admin.route.js > ...
1   import { Router } from "express";
2   import { createSong, deleteSong, createAlbum, deleteAlbum, checkAdmin } from "../controller/admin.controller.js";
3   import { protectRoute, requireAdmin } from "../middleware/auth.middleware.js";
4
5   const router = Router();
6
7   // Middleware to protect routes and require admin access
8   router.use(protectRoute, requireAdmin);
9
10  router.get("/check", checkAdmin);
11
12  router.post("/songs", createSong);
13  router.delete("/songs/:id", deleteSong);
14
15  router.post("/albums", createAlbum);
16  router.delete("/albums/:id", deleteAlbum);
17
18
19  export default router
```

- Route album

```
JS album.route.js ×
backend > src > routes > JS album.route.js > ...
1   import { Router } from "express";
2   import { getAlbumById, getAllAlbums } from "../controller/album.controller.js";
3
4   const router = Router();
5
6   router.get("/", getAllAlbums);
7   router.get("/:albumId", getAlbumById);
8
9   export default router;
10
```

- Route auth



```
JS auth.route.js ×
backend > src > routes > JS auth.route.js > ...
1   import { Router } from "express";
2   import { authCallback } from "../controller/auth.controller.js";
3
4   const router = Router();
5
6   router.post("/callback", authCallback);
7
8   export default router;
9
```

- Route song

```
backend > src > routes > JS song.route.js > ...
import { Router } from "express";
import { getAllSongs, getFeaturedSongs, getMadeForYouSongs, getTrendingSongs, getSongsByTitle } from "../controller/song.controller.js";
import { protectRoute, requireAdmin } from "../middleware/auth.middleware.js";

const router = Router();

router.get("/", protectRoute, requireAdmin, getAllSongs);
router.get("/featured", getFeaturedSongs);
router.get("/made-for-you", getMadeForYouSongs);
router.get("/trending", getTrendingSongs);
router.get("/search", getSongsByTitle);

export default router;
```

- Route stat

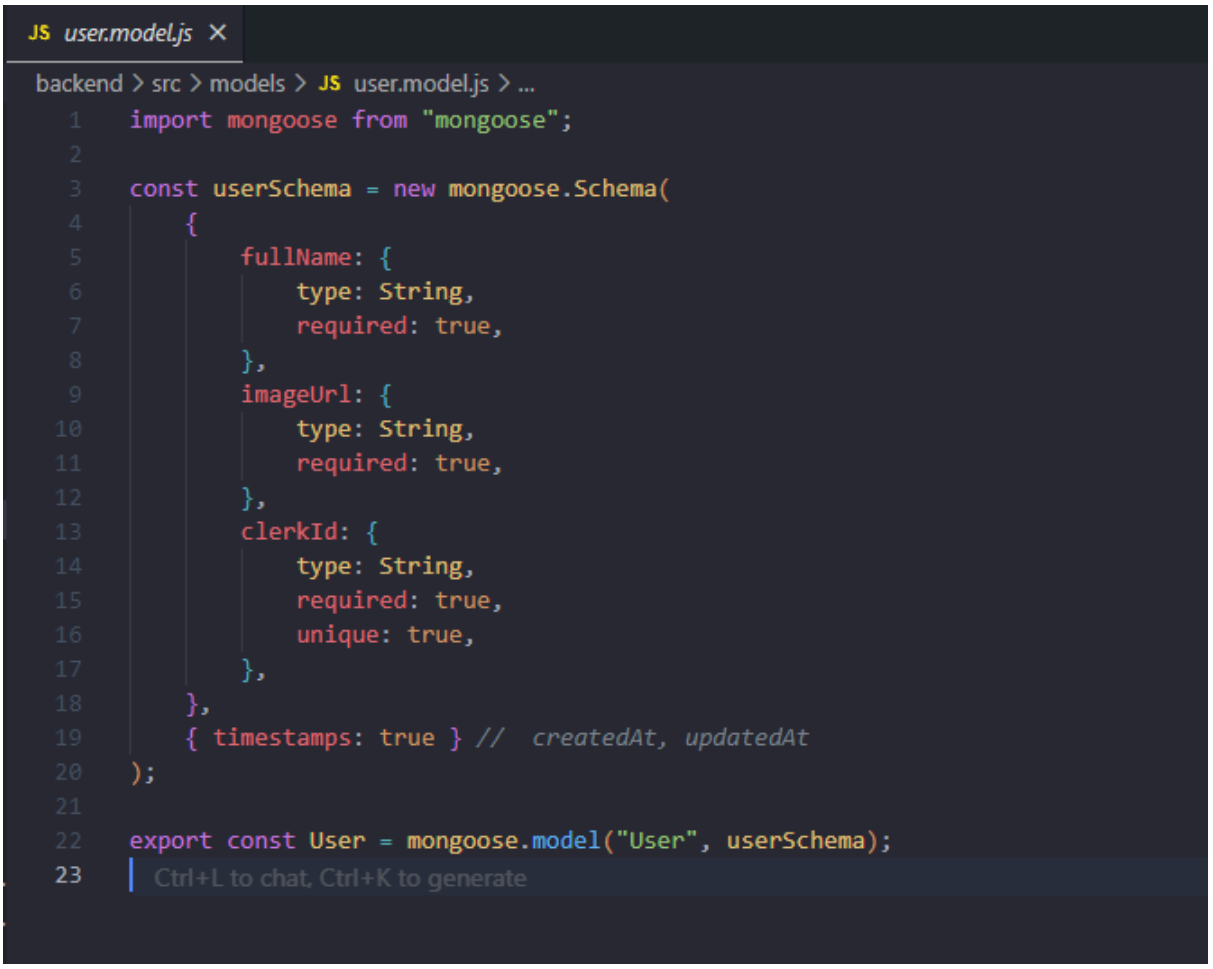
```
backend > src > routes > JS stat.route.js > ...
import { Router } from "express";
import { protectRoute, requireAdmin } from "../middleware/auth.middleware.js";
import { getStats } from "../controller/stat.controller.js";

const router = Router();

router.get("/", protectRoute, requireAdmin, getStats);

export default router;
```

- Route user

```
backend > src > routes > JS user.route.js > ...
import { Router } from "express";
import { protectRoute } from "../middleware/auth.middleware.js";
import { getAllUsers, getMessages } from "../controller/user.controller.js";
const router = Router();

router.get("/", protectRoute, getAllUsers);
router.get("/messages/:userId", protectRoute, getMessages);

export default router;
```

Các models:

- Model user:



- Model message:

- Model song:

song.model.js

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

```
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:

album.model.js

backend > src > models > 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);

```

Middlewares:

- Auth middleware để kiểm tra token và role của client:

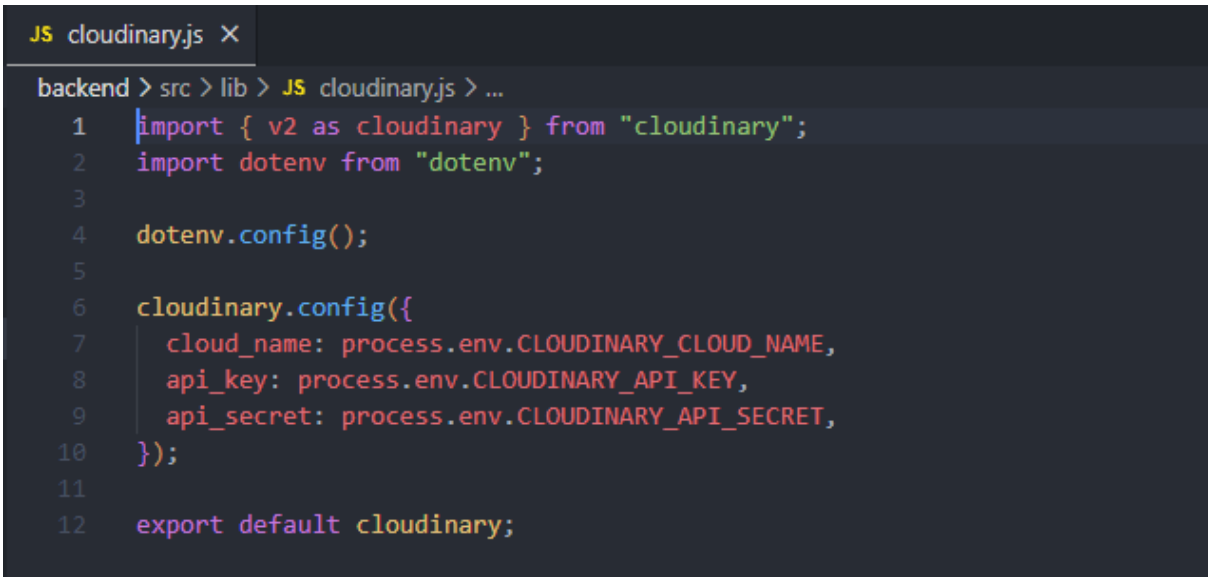
auth.middleware.js

backend > src > middleware > auth.middleware.js > ...

```
import { clerkClient } from "@clerk/express";

export const protectRoute = async (req, res, next) => {
    if (!req.auth.userId) {
        return res.status(401).json({ message: "Unauthorized - you must be logged in" });
    }
    next();
};

export const requireAdmin = async (req, res, next) => {
    try {
        const currentUser = await clerkClient.users.getUser(req.auth.userId);
        const isAdmin = process.env.ADMIN_EMAIL === currentUser.primaryEmailAddress?.emailAddress;

        if (!isAdmin) {
            return res.status(403).json({ message: "Unauthorized - you must be an admin" });
        }

        next();
    } catch (error) {
        next(error);
    }
};

```

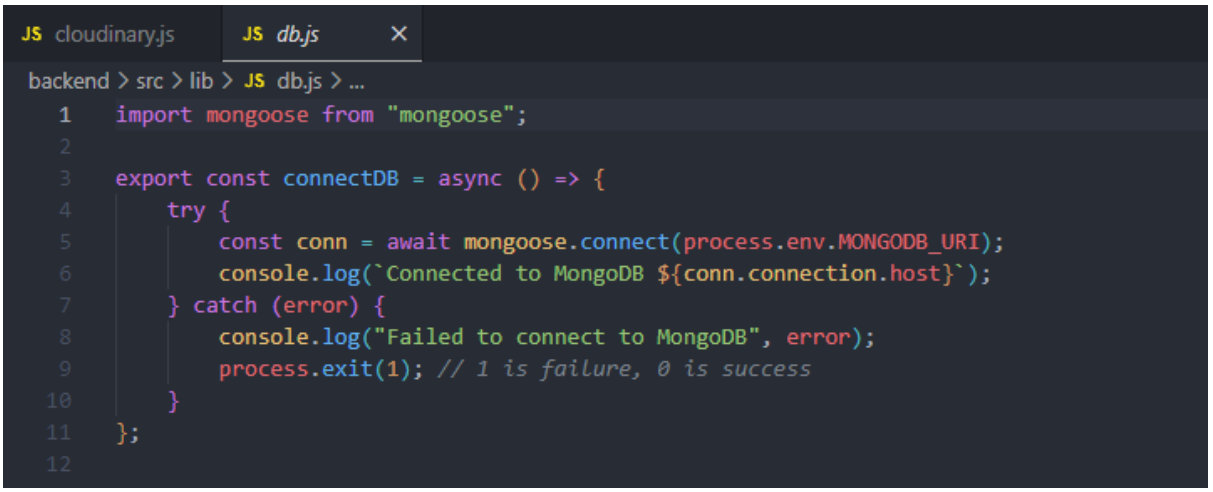
Libs:

- Cloudinary: Up ảnh và nhạc lên cloud



```
backend > src > lib > JS cloudinary.js > ...
1  import { v2 as cloudinary } from "cloudinary";
2  import dotenv from "dotenv";
3
4  dotenv.config();
5
6  cloudinary.config({
7    cloud_name: process.env.CLOUDINARY_CLOUD_NAME,
8    api_key: process.env.CLOUDINARY_API_KEY,
9    api_secret: process.env.CLOUDINARY_API_SECRET,
10 });
11
12 export default cloudinary;
```

- Mongoose: Connect tới database



```
backend > src > lib > JS db.js > ...
1  import mongoose from "mongoose";
2
3  export const connectDB = async () => {
4      try {
5          const conn = await mongoose.connect(process.env.MONGODB_URI);
6          console.log(`Connected to MongoDB ${conn.connection.host}`);
7      } catch (error) {
8          console.log("Failed to connect to MongoDB", error);
9          process.exit(1); // 1 is failure, 0 is success
10     }
11 };
12
```

Và các controllers tương ứng với các routes đã khai báo trong các routers.