Portfolio Application. Report

1. Setup Instructions

2. API

3. Design Rationale

4. 2FA Setup

5. Password Information

6. Files Review

Setup Instructions

Requirements:

- Node.js

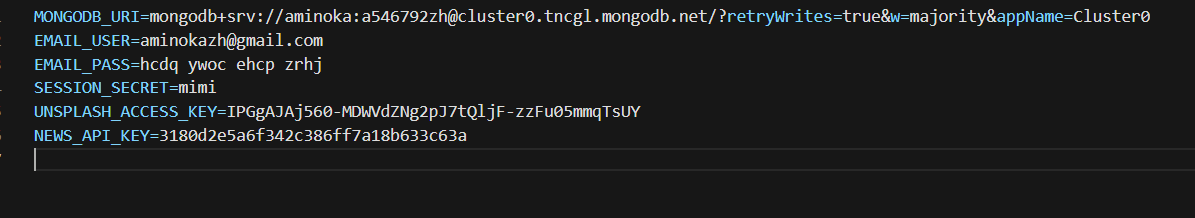
- MongoDB

- A gmail service (to send 2FA tokens)

Steps:

1. Install dependencies

2. Create a `.env` file and add these variables:



3. Start the app:

node app.js

This will run the server at `http://localhost:3000`.

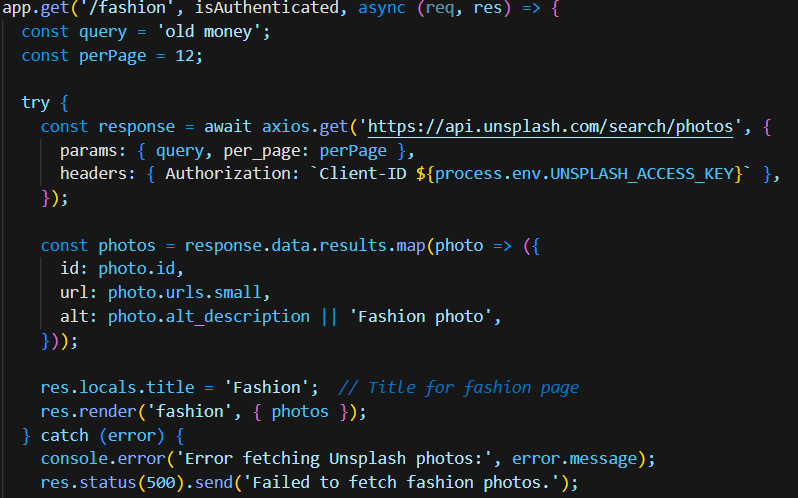
API Details

The app uses two external APIs:

1. Unsplash API

Fetches images.

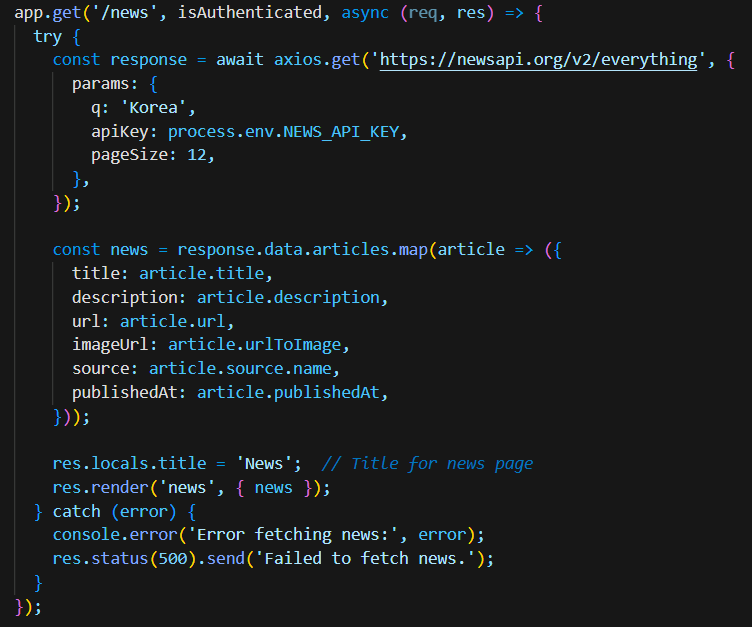
Response: A set of images



2. News API

Fetches news about Korea.

Response: last news of Korea.



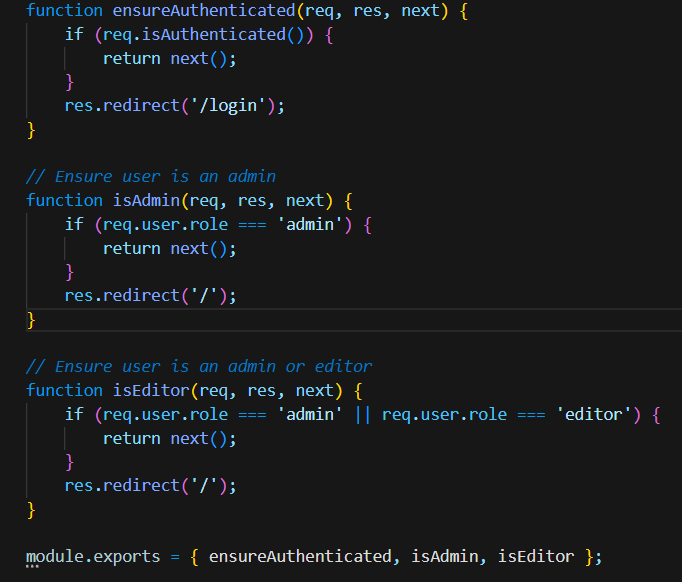
Design Rationale

The app uses Express.js with user authentication and role-based access control:

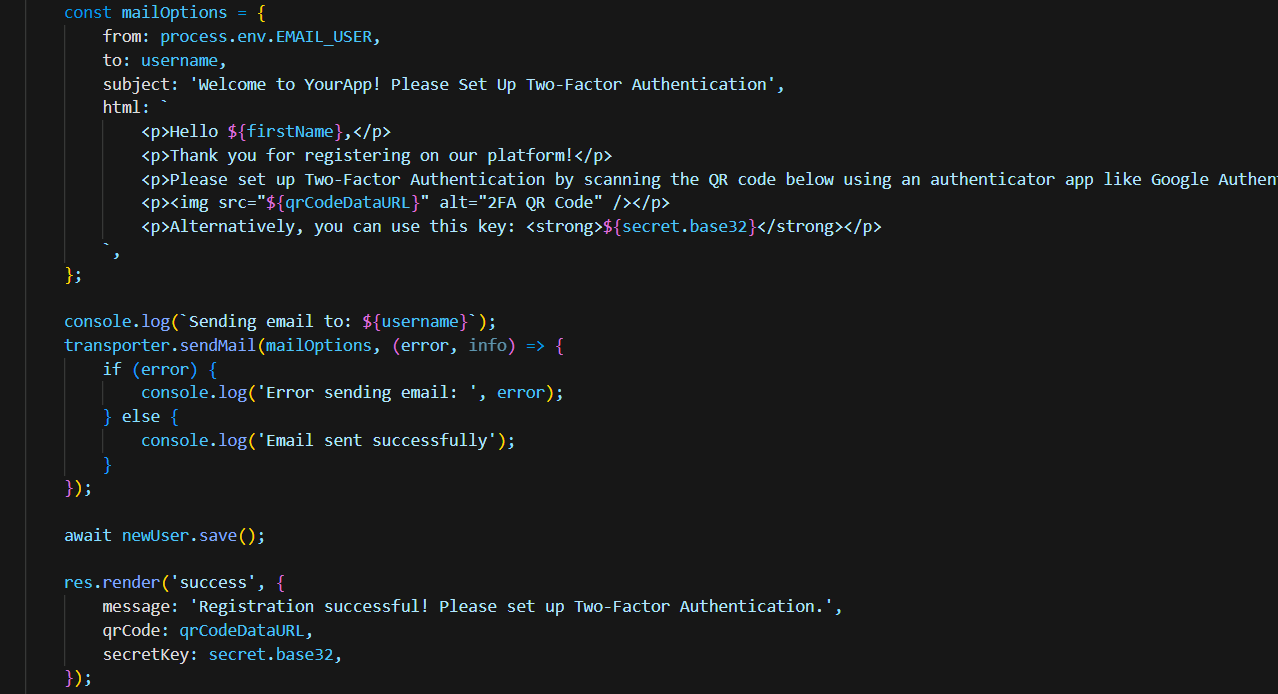
User Roles (in our imagination):

admin: create, edit, delete portfolio items.

editor: Can create their own portfolio.



2FA: After registration, a qr code is sent to the user's gmail. They need to scan it and enter code to verify their account.

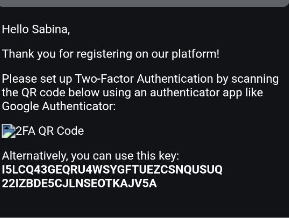


MongoDB stores user and portfolio data.



Session Management is handled by `express-session`.

2FA Setup



How 2FA works:

1. After registration, a qr is sent to the user's gmail.

2. The user enters the code from auth app to complete registration.

3. After successful verification, the user is redirected to the login page.

Password Information

Passwords are hashed using `bcryptjs`.

During registration, the password is securely stored after hashing.

During login, the provided password is checked against the hashed password.

Files Review

1. Root Files

`app.js`:

The main entry point of the application. Handles server setup, routing, and core application logic.

`.env`:

Configuration file for sensitive information like API keys and database connection strings.

`package.json` & `package-lock.json`:

Define the project's dependencies, metadata, and scripts. Lock file ensures consistent dependency versions.

`README.md/report.txt`:

Contains project documentation, including setup instructions and API details.

2. middlewares Directory

`auth.js`:

Middleware for role-based access control, verifying user roles like admin or editor.

3. models Directory

`User.js`:

Defines the database schema for user information, including fields like email, password, and roles.

`PortfolioItem.js`:

Specifies the database schema for portfolio items, such as descriptions and associated media.

4. config/utils Directory

`mailer.js`:

Handles email functionality, such as sending verification tokens for two-factor authentication.

5. public Directory

`styles.css`:

Contains CSS styles for the web application.

6. routes Directory

`auth.js`:

Defines routes related to user authentication processes, such as registration, login, and verification.

`portfolio.js`:

Handles routes related to portfolio management, such as creating, updating, and deleting portfolio items.

7. views Directory

This directory contains EJS templates for rendering dynamic web pages.

`admin/Dashboard.ejs`:

Dashboard interface for administrators to manage portfolio items.

`fashion.ejs`:

Displays images fetched from the unsplash API.

`login.ejs`:

Template for the user login page.

`.index.ejs`:

Renders the user's portfolio page.

`register.ejs`:

Template for the user registration page.

`news.ejs`:

Displays news fetched from the news API.

`success.ejs`:

Page for entering verification during two-factor authentication.