Coding Challenges: Authentication and Authorization in Express

1. Implement User Registration

Task:
Write an Express route handler for `/api/register` that:

- Accepts `email`, `name`, `password`, and `user_type_id` in the request body.
- Hashes the password using bcrypt.
- Saves the new user to the database.
- Returns a JWT token on successful registration.

---

2. Implement User Login

Task:
Write an Express route handler for `/api/login` that:

- Accepts `email` and `password` in the request body.
- Validates the credentials by comparing the provided password with the hashed password in the database.
- Returns a JWT token if authentication is successful; otherwise, returns an error.

---

3. Create JWT Verification Middleware

Task:
Write Express middleware called `verifyUserToken` that:

- Checks for a JWT token in the `Authorization` header.
- Verifies the token using a secret key.
- Attaches the decoded user info to `req.user`.
- Returns an error if the token is missing or invalid.

---

4. Create Role-Based Authorization Middleware

Task:
Write two middleware functions:

- `IsUser`: Allows access only if `req.user.user_type_id === 0`.
- `IsAdmin`: Allows access only if `req.user.user_type_id === 1`.
- Return an error if the user does not have the correct role.

Coding Challenges: DTOs & class-transformer in Express.js

1. Create a DTO Class

Task:
Write a TypeScript class `BookDTO` with fields: `title`, `author`, `publishedYear`, `isbn`, and `internalNotes`.

- Use @Expose for all fields except `internalNotes`, which should use @Exclude.

---

2. Convert a Plain Object to a DTO

Task:
Given a plain object representing a book, use `plainToClass` to convert it to a `BookDTO` instance, ensuring only exposed fields are included.

---

3. Handle Incoming Request Data

Task:
Write an Express POST route `/books` that:

- Accepts a book object in the request body.
- Converts it to a `BookDTO` using class-transformer.
- Logs the DTO to the console.

---

4. Prepare a DTO for API Response

Task:
Write an Express GET route `/books/:id` that:

- Fetches a book object (mock or from a database).
- Converts it to a `BookDTO` and sends only the exposed fields as the response.

Coding Challenges: Error Handling in Express.js

1. Add a Custom Error Handler

Task:
Write and register a custom error-handling middleware in your Express app that logs the error and sends a JSON response with the error message.

---

2. Synchronous Error Route

Task:
Create a route `/sync-error` that throws a synchronous error.

- Confirm that your error handler catches and responds to it.

---

3. Asynchronous Error Route

Task:

Create a route `/async-error` that triggers an error inside an async function.

- Use try/catch and next(err) to pass the error to your error handler.

---

4. Manual Error Forwarding

Task:

Create a route `/manual-error` that creates an error object and passes it to next(err).

---

5. Hide Stack Trace in Production

Task:

Modify your error handler so that the stack trace is only included in the response if NODE_ENV is not ‘production’.
