

Last indexed: 11 August 2025 (172577)

Overview

Repository Structure

Getting Started

Project 1: Product Management (Proy1)

Database & Configuration

Product Management System

Development Environment

Project 2: Course Platform (Proy2_Cursos)

Data Models & Database

Authentication & Middleware

Development & Testing

Project 3: Food Delivery Platform (Proy3_Pedidos)

Architecture & Dependencies

Database Models & Schema

Environment & Configuration

Development Environment

VS Code & Debugging

Dependency Management

Authentication & Middleware

> Relevant source files

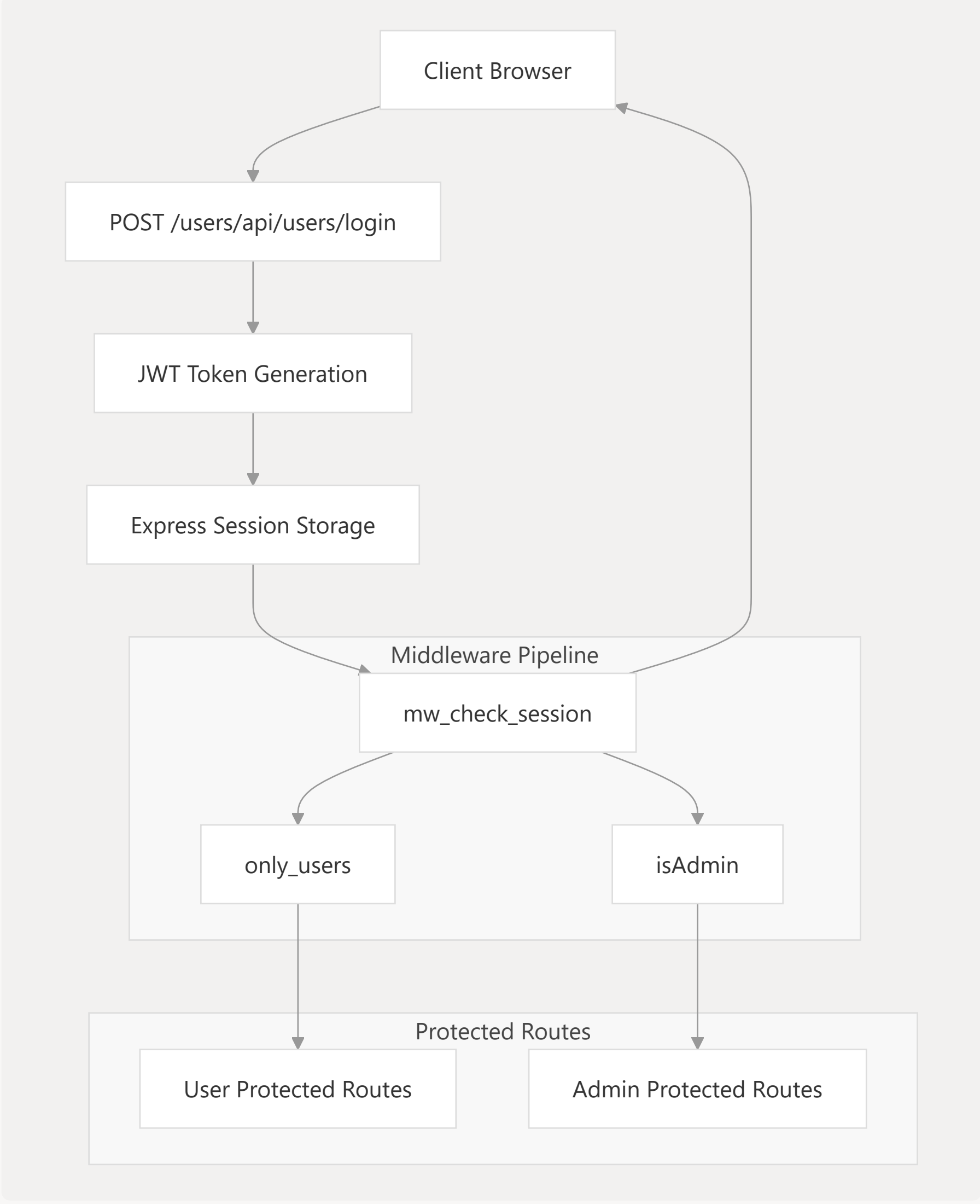
This document covers the authentication and middleware system implemented in Project 2 (Course Platform). The system provides JWT-based authentication, session management, and role-based authorization middleware that protects administrative routes and user-specific functionality.


For information about the database models that support this authentication system, see [Data Models & Database](#). For broader project architecture details, see [Project 2: Course Platform](#).


Authentication Architecture Overview

The authentication system in Proy2_Cursos implements a hybrid approach combining JWT tokens with Express sessions to provide secure access control and user state management.

Authentication Flow Architecture



Sources:  Proy2_Cursos/config/middleware.js 1-89

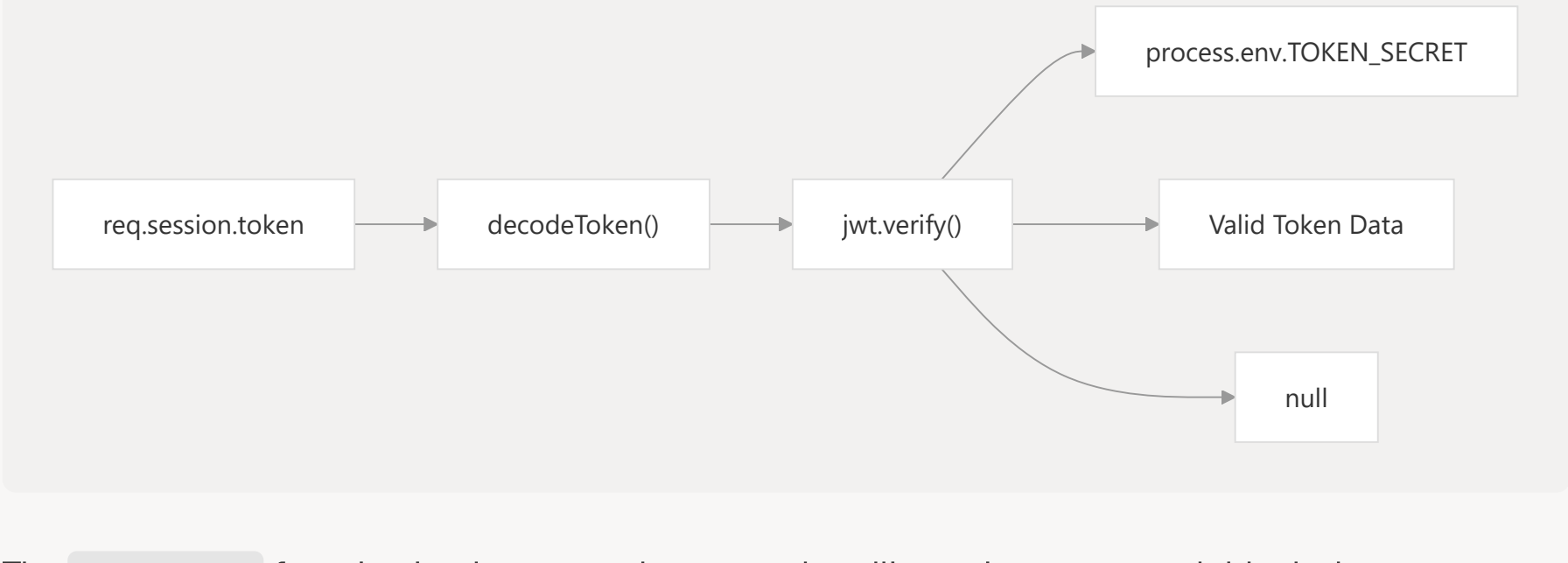
 Proy2_Cursos/__tests__/app.test.js 32-80

JWT Token Management

The system uses JSON Web Tokens for authentication with the `jsonwebtoken` library. Token operations are centralized in the `decodeToken` function.

Function	Purpose	Return Value
<code>decodeToken</code>	Verifies and decodes JWT tokens	Decoded token data or <code>null</code>

Token Verification Process



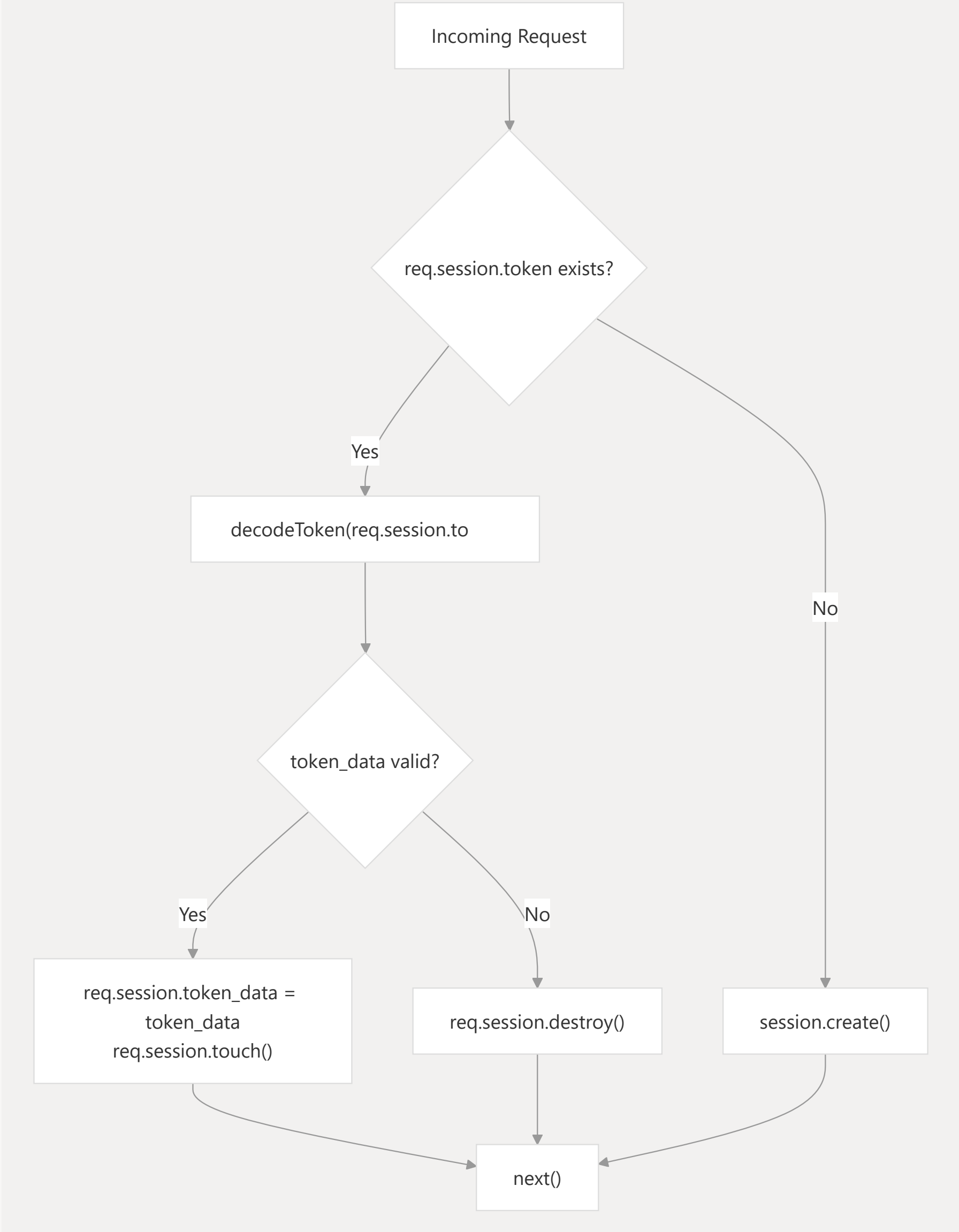
The `decodeToken` function implements robust error handling using a try-catch block that returns `null` for invalid tokens rather than throwing exceptions.

Sources:  Proy2_Cursos/config/middleware.js 8-15

Session Management Middleware


The `mw_check_session` middleware executes on every request to validate tokens and maintain session state. This middleware ensures consistent authentication state across the application.

Session Validation Flow



The middleware performs these operations:

- Token validation and decoding
- Session destruction for invalid tokens
- Session refresh via `req.session.touch()`
- Automatic session creation for new requests

Sources:  Proy2_Cursos/config/middleware.js 37-58

Authorization Middleware Components

The system provides two levels of authorization middleware that protect different route categories.

User-Level Authorization

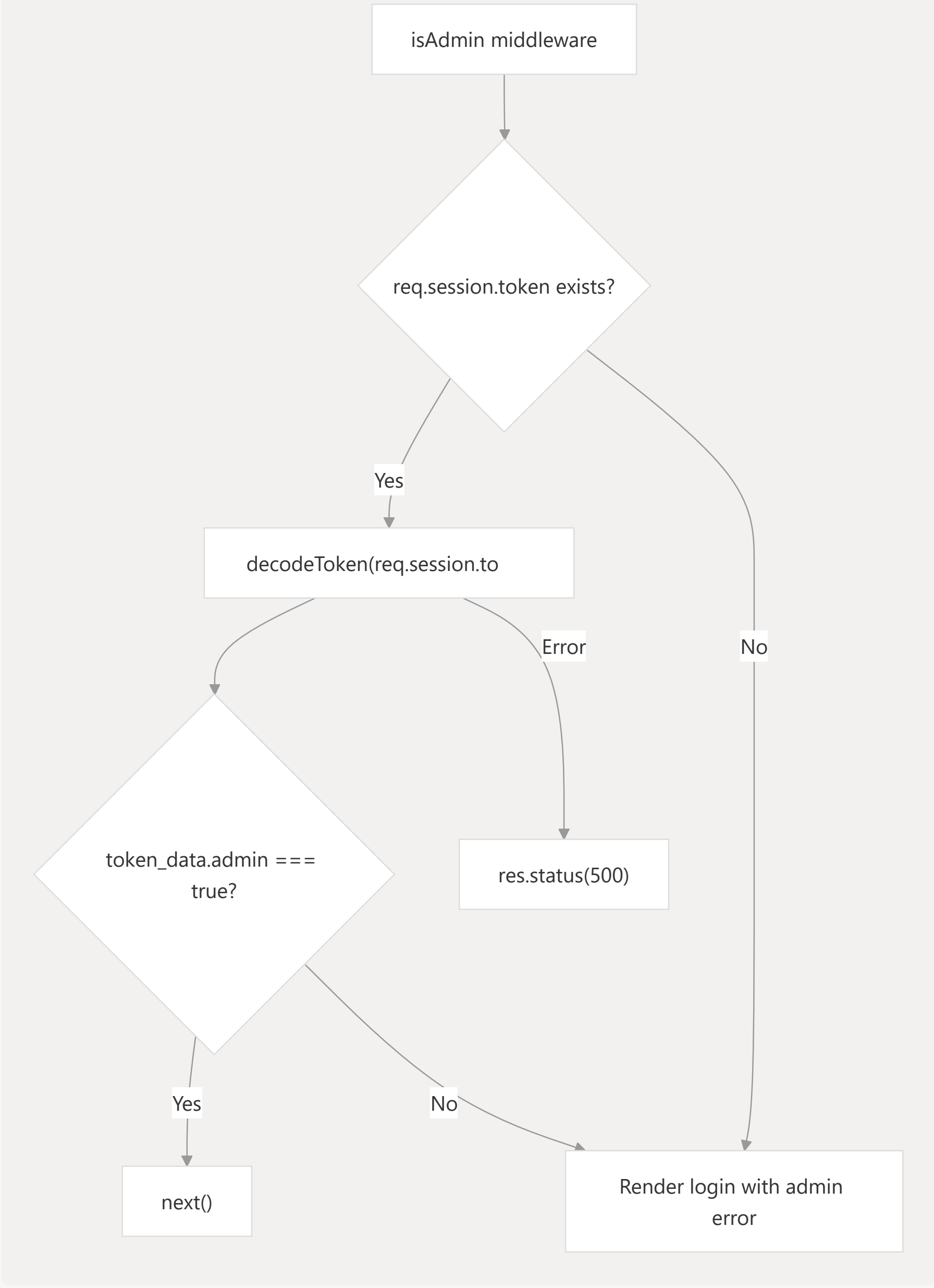
The `only_users` middleware restricts access to authenticated users only.


Check	Action	Response
<code>req.session.token</code> exists	Allow access	<code>next()</code>
No token present	Deny access	Render login page with error


Admin-Level Authorization

The `isAdmin` middleware provides granular admin access control by examining the `admin` claim in decoded JWT tokens.

Admin Authorization Logic



Sources:  Proy2_Cursos/config/middleware.js 17-34


 Proy2_Cursos/config/middleware.js 63-83

User Model Integration

The authentication system integrates with the Sequelize User model that defines the database schema for user authentication.

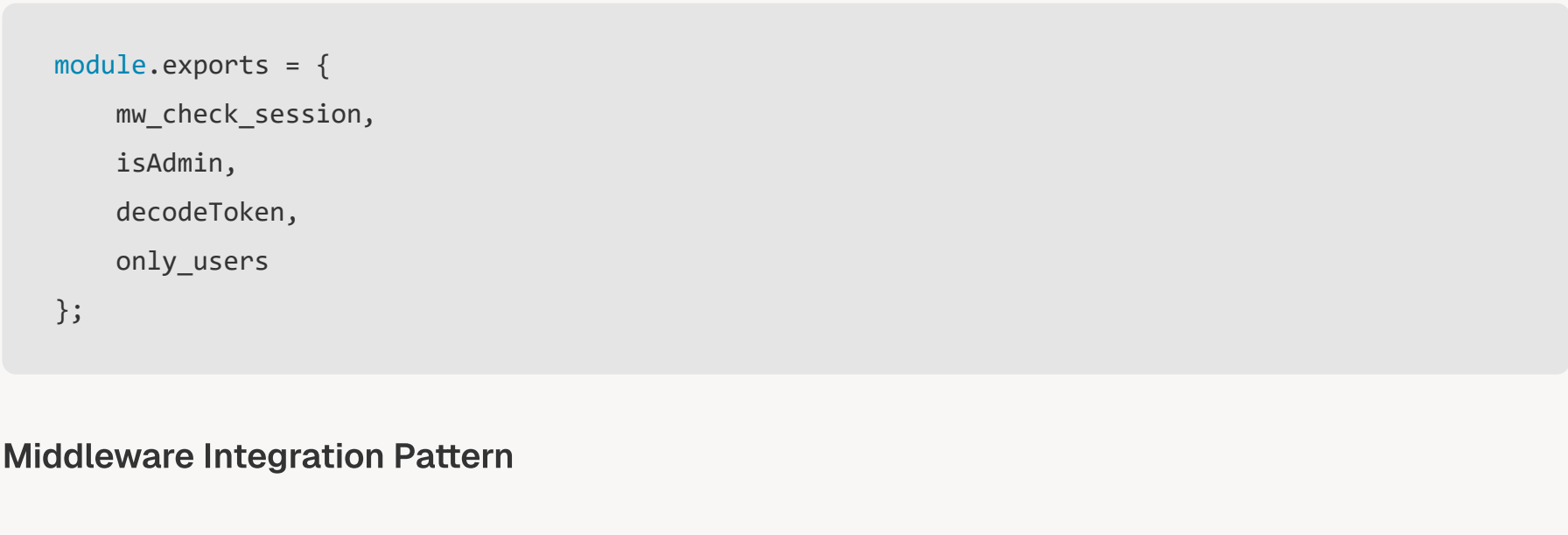
Field	Type	Constraints	Purpose
<code>name</code>	STRING	NOT NULL	Display name
<code>user</code>	STRING	NOT NULL, UNIQUE	Login username
<code>password</code>	STRING	NOT NULL	Encrypted password
<code>isAdmin</code>	BOOLEAN	NOT NULL, DEFAULT false	Admin role flag

The `isAdmin` field directly corresponds to the `admin` claim stored in JWT tokens, enabling role-based access control.

Sources:  Proy2_Cursos/models/user.js 1-28


Middleware Export and Integration

The middleware module exports four key functions for use throughout the application:



The testing suite validates this middleware integration by simulating authentication flows and verifying access control for protected admin routes.

Sources:  Proy2_Cursos/config/middleware.js 86-88

 Proy2_Cursos/__tests__/app.test.js 61-79