CSE 341 Final Project Contributions Diogo Rangel Week06 and Week07 - Final Project

CSE 341 Final Project - Diogo Rangel Dos Santos

Project Contributions Report (Diogo Rangel Dos Santos)

This report details my contributions to the final project, covering the creation of the code structure, the implementation of key functionalities, and the critical debugging of the authentication system.

Week 06 : W06 Final Project Part 2: Last Two Collections with Crud Operations, OAuth and Testing

I. Core Development and Project Structure (Original Contribution)

I developed the largest part of the project, including:

- Project Scope: Developed the biggest part of the project, including initial setup and core features.
- Data Setup: Responsible for initial Data structures and setup
- User Management: Implemented the Users and registration setup and related files.
- Routes: Defined and structured the primary application Routes (index.js, auth.js, users.js, records.js, etc.).
- Middleware: Developed and implemented core Middleware components (e.g., authentication checks).
- Models: Created and defined the database Models (schemas) for MongoDB/Mongoose.
- Overall: Executed the project in collaboration with the team, completing all required modules.

II. Critical Bug Fixing & Authentication Implementation (Refactoring)

In addition to building the structure, I executed crucial fixes to stabilize the Google OAuth authentication flow and core server integrity:

- server.js - Critical Middleware Order: Corrected the middleware execution order for Passport and Session. The express-session middleware is now executed before passport.initialize() and passport.session(). This correction solved the "User ID: Not Logged In" error and the Passport error requiring session support.

server.js - CORS Configuration: Configured sameSite: 'none' and secure: true for cookies in the production environment (Render/HTTPS). This change was essential to enable cross-origin session cookie transmission, ensuring the OAuth flow functions correctly after deployment.

- routes/index.js - Dashboard Route: Added the base /dashboard route and the isAuthenticated middleware to routes/index.js. This resolved the "Cannot GET /dashboard" error that occurred after successful Google login redirection.

- routes/auth.js - Robust Logout: Refactored the /logout route logic to ensure Passport's req.logout() is called before req.session.destroy() and res.clearCookie(), ensuring a complete and reliable session termination.

- routes/categories.js - Route Security: Implemented CRUD routes and applied the isAuthenticated middleware to all data modification operations (POST, PUT, DELETE).

- Dependencies: Identified and resolved the runtime error (MODULE_NOT_FOUND) by requesting the installation of the missing express-session package.

W07 Final Project Part 3: Finish Project

Business Logic and Control (/controllers/*.js)

The controllers are the core execution environment for the application's business logic. All **CRUD (Create, Read, Update, Delete) handlers** were developed within this layer. A crucial element of this implementation was the embedding of stringent authorization logic directly into the controllers. This measure guarantees that all data modification and deletion operations are strictly limited to the authenticated **resource owner**, thereby enforcing data integrity and user separation.

API Routing and Structure (/routes/*.js and routes/index.js)

The Express route structure was meticulously established to define the API contract. A clear architectural distinction was enforced between endpoints providing **public read access** and those requiring **authenticated write access**, utilizing middleware for differentiation. The main router (routes/index.js) was configured to handle seamless module integration, ensuring all component routes are correctly mapped under their respective base paths.

Security and Access Control (/middleware/authenticate.js)

The **isAuthenticated middleware** was implemented to function as the application's centralized access control point. This layer is tightly integrated with the session management system, providing an efficient

mechanism to restrict access to sensitive API resources based on the user's current authentication status.

Authentication System (config/passport.js & server.js)

The foundation for user identity management was established through the integration of the **Passport.js library**. The configuration included setting up the Google OAuth 2.0 strategy and managing the user serialization/deserialization processes crucial for persistent session handling across requests.

Server Environment Configuration (server.js)

The server environment was configured to ensure production readiness. This involved setting up the **CORS policy** for secure cross-origin requests, **Express Session management**—with specific production-ready cookie attributes (sameSite: 'none' and secure: true)—and the correct initialization of both **Passport** and the persistent **MongoDB connection**.

Principle of Resource Ownership and Data Isolation

A fundamental security contribution was the implementation of resource-level authorization. Every data document (Record, Category, Comment) is linked to the user who created it via a mandatory ownerId field. All authenticated GET, PUT, and DELETE requests are pre-processed by the controllers to enforce that the requesting user's ID matches the document's ownerId. This mechanism guarantees complete **data isolation**, ensuring that users can only view, modify, or delete resources that they personally own, thereby securing privacy and preventing unauthorized data manipulation.

Conclusion of Implementation Phase

The system implementation successfully established a fully functional and secure RESTful API. The architectural decisions made during this phase prioritized modularity and security, resulting in distinct and well-defined layers for routing, business logic, and data handling. This robust foundation ensures the API is prepared for future feature expansion and client-side consumption.