Milestone Two Enhancement Narrative

# Overview of Enhancements

This narrative describes the changes made to improve the software design and engineering aspects of the Travlr project. The enhancements address modularity, centralized configuration and error handling, refactoring outdated practices, and improving controller logic structure.

# Enhancement Summary with Screenshot References

* ✅ Refactored `app.js` to use `const` instead of `var`, added centralized error handler middleware, and imported configuration from a new `config.js` file.
* ✅ Created `middleware/errorHandler.js` for standardized JSON and HBS error responses.
* ✅ Introduced `config.js` to centralize API base URLs, request options, and environment variables.
* ✅ Updated `routes/index.js` to include logging middleware and proper scoping.
* ✅ Refactored `controllers/travel.js` to use `async/await`, fetch API config from `config.js`, and improve error safety and logic clarity.

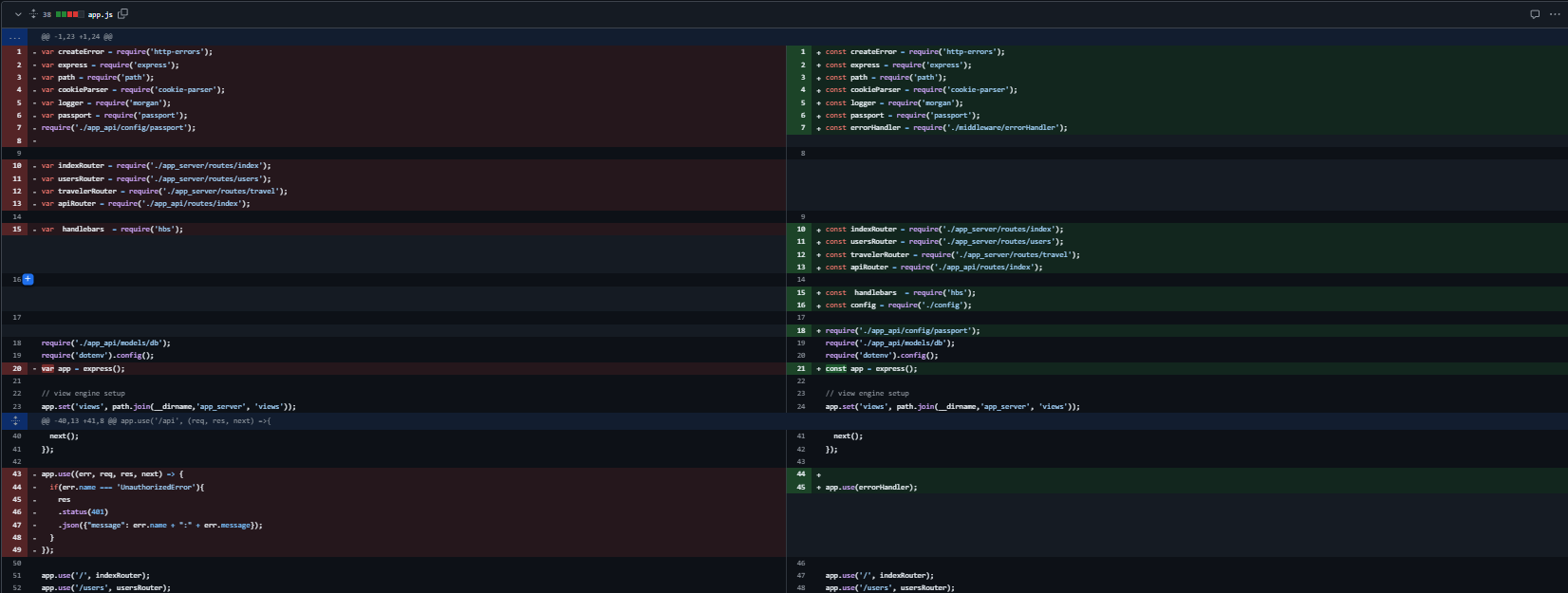
# Rationale

These changes bring the application into alignment with modern JavaScript and Node.js practices. Centralized configuration improves security and maintainability. Middleware-based error handling creates a more consistent experience for both API consumers and UI users. Transitioning to `const`/`let` avoids scoping issues and modernizes the codebase.

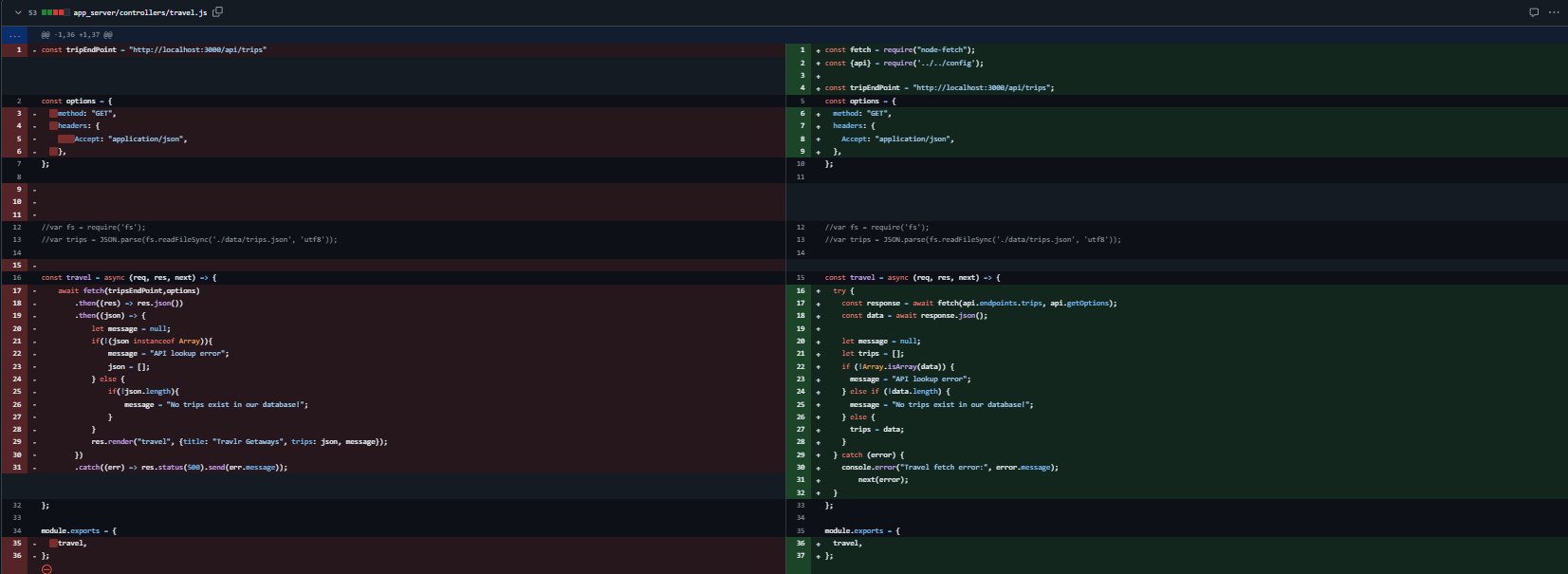
# Before and After Highlights

The following screenshots reflect key code changes:

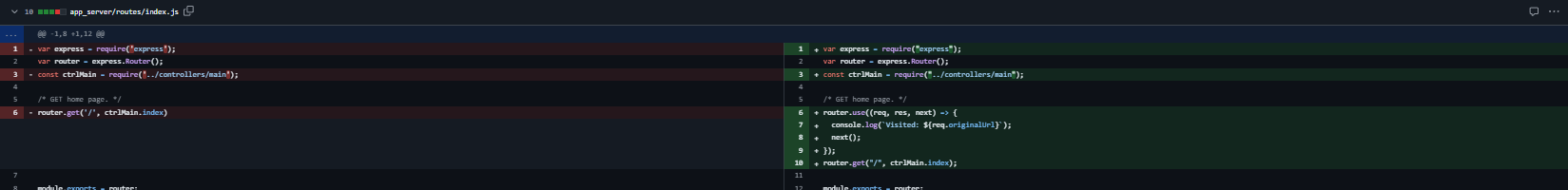
• Error handler middleware added in `middleware/errorHandler.js`



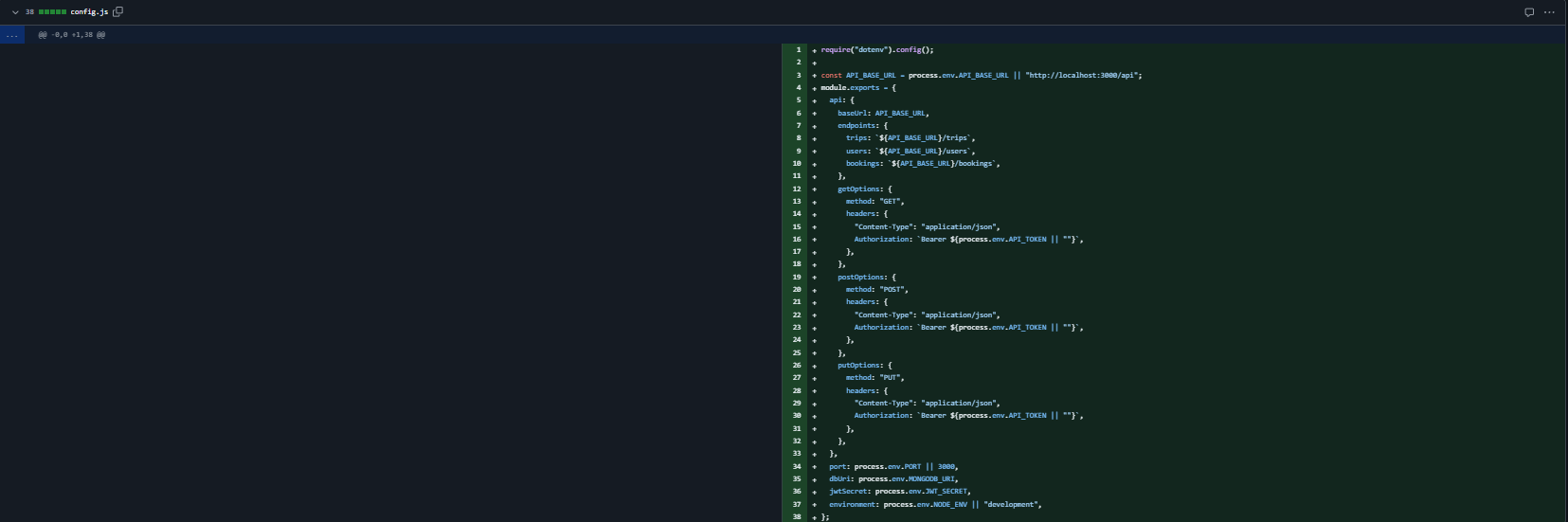
• New centralized configuration in `config.js`



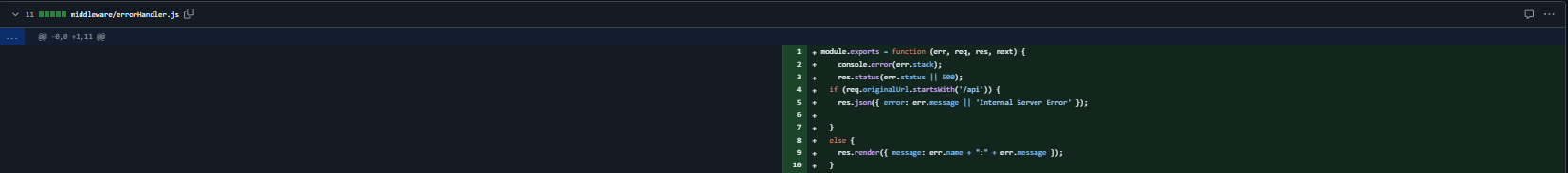
• Route cleanup and logging middleware in `app\_server/routes/index.js`



• Major controller refactor for `travel.js`



• Updated `app.js` with modular imports and cleaner structure



# Skills Demonstrated and Outcomes Met

These enhancements support the following course outcomes:

* ✔ Design, develop, and deliver professional-quality software that is modular and maintainable.
* ✔ Implement secure and efficient practices such as scoped variables, configuration management, and error abstraction.
* ✔ Apply clean coding and middleware principles in a full-stack application context.

# Reflection

Working through these changes provided valuable insight into separating concerns within a Node.js application. The act of abstracting error handling and configuration improved the testability and readability of the application. Additionally, updating the controller logic reinforced my understanding of async patterns and centralized logic flow.