CODE REVIEW EVALUATION FORM

JavaScript & Express.js | Undergraduate Programming Course

1. SUBMISSION INFORMATION

Course:	ICS 385 Web Development and Administration	**Section:**	N/A
Instructor:	Dr. Debasis Bhattacharya	**Semester:**	Spring
Student Name:	N/A	**Student ID:**	-------
Project Title:	Hawaii Tourism LOS Calculator	**Date:**	02/22/2026
Reviewer:	Lilith Gannone	**Review Type:**	Peer / Instructor

2. CODE SUBMISSION DETAILS

Repository URL:	https://github.com/debasisb/hawaii-tourism-los-calculator		
Branch:	main	**Commit Hash:**	
Files Reviewed:	all	**Lines of Code:**	

3. CODE OVERVIEW & PURPOSE

Briefly describe the purpose of the submitted code, its main functionality, the Express.js routes implemented, and any middleware or external packages used.

Summary:

Description and Main Functionality: This program is a Hawai'i tourism length of stay (LOS) calculator. It allows users to select a user category (and optionally a location) and outputs a summary of LOS statistics based on tourism data from the Hawai'i Department of Business, Economic Development & Tourism. It can be run on a server or as a standalone application.
Express.js Routes: GET (GET /api/categories, GET /api/data, etc) and app.use.
Middleware: cors(), express.json(), express.urlencoded, and express.static.
External Packages: Express, MongoDB, COS, dotenv, chart rendering, Papa Parse, and csv-parser.

4. EVALUATION CRITERIA

Understood with assistance via Codex comments and discussion.

Rate each criterion on the scale provided. Use the descriptors as guidance. A score of 4 = Excellent, 3 = Proficient, 2 = Developing, 1 = Beginning, 0 = Not Attempted.

Criterion	Description	Score (0–4)	Weight
Code Correctness & Functionality	Application runs without errors; all Express routes return expected responses; edge cases handled.	4	20%

Criterion	Description	Score (0–4)	Weight
Code Structure & Organization	Logical file/folder structure (e.g., routes/, controllers/, models/); separation of concerns; modular design.	4	15%
Naming Conventions & Readability	Variables, functions, and routes use clear, descriptive names following camelCase conventions; consistent formatting.	4	10%
Express.js Best Practices	Proper use of Router, middleware chaining, error-handling middleware, appropriate HTTP methods and status codes.	3	15%
Error Handling & Validation	Input validation present; try/catch or .catch() used; meaningful error messages returned to client.	2	10%
Comments & Documentation	Inline comments explain non-obvious logic; README or header comments describe setup, dependencies, and usage.	2	10%
Security Considerations	No hardcoded secrets; use of environment variables; input sanitization; helmet or CORS configured if applicable.	2	10%
Testing & Reliability	At least basic test cases provided (e.g., using Jest or Supertest); tests cover primary routes and edge cases.	2	10%

Total Weighted Score:	_____ / 4.00	**Percentage:**	_____ %

5. DETAILED FINDINGS — CODE-LEVEL OBSERVATIONS

Document specific issues, bugs, or noteworthy patterns found during the review. Reference file names and line numbers where applicable.

#	File / Line	Severity	Category	Description / Observation
1	public: index.html/102	High / Med / Low	specific issue	external CDN script is loaded without integrity attributes- can cause security issue for users
2	standalone: index.html/396	High / Med / Low	specific issue	external CDN script is loaded without integrity attributes- can cause security issue for users
3	routes: api.js/120	High / Med / Low	specific issue	raw error messages are returned to users.
4	routes: api.js/125	High / Med / Low	specific issue	no permission filtering- endpoint exposes raw dataset records
5	importData.js/28	High / Med / Low	specific issue	data-loss risk. All records are deleted prior to validating that import was successful.
6	server.js/28	High / Med / Low	specific issue	the app exits immediately if MongoDB is unavailable.
7	tourismdata.js/33	High / Med / Low	specific issue	nothing in place preventing duplicate group indicator datasets.
8	public: app.js/118	High / Med / Low	specific issue	there is no request timeout or retry behavior. No messages for users and they may be stuck waiting.

6. EXPRESS.JS & JAVASCRIPT CHECKLIST

Understood via review, Codex comments, and Codex chat.

Check each item that applies to the submitted code. Mark Y (Yes), N (No), or N/A.

Category	Checklist Item	Y / N / N/A
Server Setup	Server listens on a configurable port (e.g., process.env.PORT)	Y
Server Setup	Entry point file is clearly identified (e.g., app.js or server.js)	Y
Routing	Routes are organized using express.Router()	Y
Routing	RESTful conventions followed (GET, POST, PUT/PATCH, DELETE)	N
Routing	Route parameters and query strings used correctly	N/A
Middleware	Body-parser or express.json() configured for request parsing	Y
Middleware	Custom middleware is reusable and well-documented	N/A
Middleware	Error-handling middleware defined with (err, req, res, next) signature	N
Async/Await	Promises and async/await used correctly (no unhandled rejections)	Y
Async/Await	Callback patterns avoided in favor of modern async patterns	Y/N
Dependencies	package.json lists all dependencies; no unused packages	Y
Dependencies	node_modules excluded via .gitignore	Y

Category	Checklist Item	Y / N / N/A
Security	Environment variables managed via .env / dotenv	Y
Security	No sensitive data committed to version control	N

7. QUALITATIVE FEEDBACK

Strengths — What does this submission do well?

:

The current program runs well. It successfully completes its main functionalities. It is well organized and modern async patterns are primarily followed. This specifically will help with blocking prevention and error handling. Readme.md file helps users run the standalone version of the program. Users can run this program without needing to install additional dependencies. Detailed summaries available in .md files. Data visualizations available and functional. GET and POST routes successfully implemented. Code such as "hideLoading", "showLoading", etc. support error handling. The server handles shutdown via SIGINT. Data flow is clean and well-structured. Form data is checked and handled well on the server and in the browser.

Areas for Improvement — What should the student focus on next?

:

Per AI analysis of the code (via comments made to code highlighting specific issues), top priorities should be to secure client-facing error messaging, add validation measures for external CDN scripts, address the data loss risk during imports, improve MongoDB availability issues, prevent duplicate data error potentiality, and implement request timeout or retry behavior for API calls. UI could be altered to increase accessibility. Audio support cues could be added.

Suggested Learning Resources

:

To improve this code, one could run an AI-analysis of the code to understand potential issues/ security risks. One could also refer to recourses like W3Schools, Udemy, and Coursera to understand the basics of UX design. W3Schools and Udemy can help to explain Express and Node best practices, syntax, and basics. Coursera can help one to understand the importance of accessibility.

8. OVERALL ASSESSMENT

Grade	Range	Description
A / Excellent	90–100%	Code is well-structured, fully functional, secure, and demonstrates mastery of Express.js concepts.
B / Proficient	80–89%	Code works correctly with minor issues; good organization and documentation; some improvements possible.
C / Developing	70–79%	Code runs but has notable gaps in structure, error handling, or best practices; needs revision.
D / Beginning	60–69%	Significant issues with functionality, structure, or documentation; substantial rework required.
F / Incomplete	Below 60%	Code does not compile/run or is largely incomplete; fundamental concepts not demonstrated.

Final Grade Assigned:		**Numeric Score:**	_____ / 100

9. REQUIRED REVISIONS & ACTION ITEMS

List any mandatory changes the student must complete before resubmission.

#	Action Item	Priority	Due Date
1	Secure client-facing error messaging	High / Med / Low	
2	Add validation measures for external CDN scripts	High / Med / Low	
3	Address the data loss risk during imports	High / Med / Low	
4	Prevent duplicate data error potentiality	High / Med / Low	

10. ACADEMIC INTEGRITY ACKNOWLEDGMENT

By signing below, the reviewer confirms that this evaluation was conducted fairly and objectively. The student acknowledges receipt of this feedback and understands the revisions required.

Reviewer Signature:	Lily Gannone	**Date:**	02/22/2025
Student Signature:		**Date:**	
Instructor Signature:		**Date:**	