CODE REVIEW EVALUATION FORM

JavaScript & Express.js | Undergraduate Programming Course

1. SUBMISSION INFORMATION

Course: ICS 385	**Section: N/A**
Instructor: Debasis Bhattacharya	**Semester: Spring 2026**
Student Name: Noah Munz	**Student ID: 23760630**
Project Title: Hawaii Tourism LOS Calculator Code	**Date: 2/21/2026**
Reviewer: Self / Instructor	**Review Type:** Peer / Instructor

2. CODE SUBMISSION DETAILS

Repository URL:	
Branch: Main	**Commit Hash:**
Files Reviewed: index.html, app.js, data.csv, readme.md	**Lines of Code: 370 (app.js) + HTML/CSV**

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:

Browser-only Hawaii Tourism LOS calculator. Fetches/parses local CSV (1999-2025) via PapaParse, computes avg/min/max by visitor category and island location, renders Chart.js trend line.
No backend — all logic runs client-side in app.js. External CDN deps: Chart.js 4.4.0, PapaParse 5.4.1.
No Node.js required.

4. EVALUATION CRITERIA

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.	**3**	20%

Criterion	Description	Score (0–4)	Weight
Code Structure & Organization	Logical file/folder structure (e.g., routes/, controllers/, models/); separation of concerns; modular design.	**3**	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.	**N/A**	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.	**1**	10%

Total Weighted Score:	2.65 / 4.00	**Percentage:**	66 %

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	**app.js**	High / Med / Low **= High**	**Bug**	**processData(): !isNaN(key) accepts empty strings as year columns — replace with /^\d{4}$/ regex**
2	**app.js**	High / Med / Low **= High**	**Bug**	**processData(): row[key].trim() throws TypeError if row[key] is undefined — add null guard**
3	**app.js**	High / Med / Low **Medium**	**Security**	**displayResults()/showError(): use .textContent not innerHTML — prevents XSS from data strings**
4	**app.js**	High / Med / Low **Medium**	**Resilien ce**	**loadCSVData(): no isLoading guard — concurrent calls can corrupt the in-memory tourismData array**
5	**app.js**	High / Med / Low **Medium**	**Bug**	**createChart(): no null-guard before canvas.getContext() — TypeError if #losChart missing from DOM**
6	**app.js**	High / Med / Low **= Low**	**Maintain ability**	**Magic strings/patterns not in named constants — SKIP_KEYWORDS and year check buried in function bodies**
7	**app.js**	High / Med / Low **= Low**	**Docume ntation**	**No JSDoc or inline comments — function intent and algorithm logic entirely undocumented in originai**
8	**data.csv**	High / Med / Low **= Low**	**Data Currenc y**	**CSV only covers 1999-2021; DBEDT warehouse extends to 2025 — updated data.csv with 2022-2025 columns**

6. EXPRESS.JS & JAVASCRIPT CHECKLIST

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

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

7. QUALITATIVE FEEDBACK

Strengths — What does this submission do well?

: Clean separation of concerns across loadCSVData, processData, calculateLOS, and displayResults.
Chart.js destroy-before-render prevents canvas memory leaks.
PapaParse is the right choice for in-browser CSV parsing.
SKIP_KEYWORDS footer filter is elegant and easy to extend.
Responsive CSS with grid and media queries is polished.
No-install standalone approach is well-suited to the use case.

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

:1. Fix !isNaN(key): replace with /^\d{4}$/ regex to prevent silent data corruption.
2. Null-guard before row[key].trim() — prevents TypeErrors on ragged CSV rows.
3. Standardize all DOM writes to .textContent to eliminate XSS risk.
4. Add isLoading flag to prevent duplicate fetch calls.
5. Add JSDoc + inline comments throughout app.js for maintainability.

Suggested Learning Resources

:MDN: Element.textContent vs innerHTML (XSS implications).
MDN: isNaN() quirks and Number.isNaN().
PapaParse docs: skipEmptyLines option.
Chart.js docs: chart.destroy() lifecycle management.
OWASP DOM-based XSS Prevention Cheat Sheet.
JSDoc reference: jsdoc.app.

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: B / Proficient		**Numeric Score:**	83 / 100

9. REQUIRED REVISIONS & ACTION ITEMS

List any mandatory changes the student must complete before resubmission.

#	Action Item	Priority	Due Date
1		High / Med / Low	
2		High / Med / Low	
3		High / Med / Low	
4		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:	Noah Munz	**Date:**	2/21/2026
Student Signature:	Noah Munz	**Date:**	2/21/2026
Instructor Signature:		**Date:**	