CODE REVIEW EVALUATION FORM

JavaScript & Express.js | Undergraduate Programming Course

1. SUBMISSION INFORMATION

Course:	**ICS 385**	**Section:**	**1**
Instructor:	**Debasis Bhattacharya**	**Semester:**	**Spring 2026**
Student Name:	**Jonas Marcial**	**Student ID:**	**26502437**
Project Title:	**Assignment 6c - Hawaii Tourism LOS**	**Date:**	**2/20/2026**
Reviewer:	**Jonas Marcial**	**Review Type:**	Peer / Instructor

2. CODE SUBMISSION DETAILS

Repository URL:	**https://github.com/jmarcial2001/ics385spring2026.git**		
Branch:	**Main**	**Commit Hash:**	
Files Reviewed:	**standalone/app.js standalone/data.csv standalone/index.html standalone/styles.css README.md**	**Lines of Code:**	**250-300**

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:

This project is a standalone Hawaii Tourism Length of Stay (LOS) Calculator that loads tourism data from a CSV file and allows users to calculate average length of stay by visitor group and location. The application runs entirely in the browser using vanilla JavaScript and the PapaParse library to parse CSV data. It processes tourism data from 1999 through 2024 and dynamically calculates statistics and displays results in the UI. The app does not require a backend server or database in the standalone version.

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.

Code Correctness & Functionality	The application runs without errors in the browser using Live Server. The CSV file loads correctly and displays 117 records in the console. Calculations and dropdown filtering work as expected.	4	20%

Code Structure & Organization	The project is clearly organized with separate folders for standalone and server-based versions. In the standalone version, all JavaScript logic is in one file (app.js), which works but could be further modularized for better scalability.	3	15%
Naming Conventions & Readability	Variable and function names are clear and descriptive (e.g., loadCSVData, processData, tourismData). Code is consistently formatted and easy to follow.	4	10%
Express.js Best Practices		0	15%
Error Handling & Validation	Basic error handling is implemented using try/catch. During review, response validation (response.ok) was added before parsing the CSV file to prevent processing failed fetch responses. Minor input validation improvements were also made by trimming CSV fields.	3	10%
Comments & Documentation	Code includes helpful comments explaining major sections. However, the README documentation is outdated and still references data coverage through 2021, while the updated CSV now includes data through 2024.	3	10%
Security Considerations	The standalone version does not contain sensitive credentials or hardcoded secrets. Since it runs entirely client-side, security risks are limited. Basic input trimming was added to improve data integrity.	3	10%
Testing & Reliability	There are no automated tests included. Functionality was manually tested in the browser.	2	10%

Total Weighted Score:	3.2 / 4.00	**Percentage:**	100%

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.

1	standalone/app.js	High / **Med** / Low	Error Handling	Added response.ok validation before parsing CSV
2	standalone/app.js	High / Med / **Low**	Data Integrity	Added sorting of yearly data to ensure chronological order
3	standalone/app.js	High / Med / **Low**	Input Validation	Trimmed CSV values to prevent whitespace inconsistencies
4	standalone/data.csv	High / **Med** / Low	Data Accuracy	Updated data through 2024 to match DBEDT warehouse
5		High / Med / Low		
6		High / Med / Low		
7		High / Med / Low		
8		High / Med / Low		

6. EXPRESS.JS & JAVASCRIPT CHECKLIST

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

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?

: The application runs smoothly and processes time-series tourism data effectively. The UI is simple and easy to use. CSV parsing with PapaParse is implemented correctly. After updating the dataset, the application continued to function properly, indicating solid data handling logic.

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

The JavaScript logic could be modularized into separate files for better maintainability. Automated testing could improve reliability. The README should be updated to reflect the current data range (1999–2024) and clarify the difference between the standalone and Express versions.

Suggested Learning Resources

:

8. OVERALL ASSESSMENT

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:	**92%**	**Numeric Score:**	92 / 100

9. REQUIRED REVISIONS & ACTION ITEMS

List any mandatory changes the student must complete before resubmission.

1	Modularize app.js into separate data-processing and UI modules for better maintainability.	High / Med / **Low**	
2	Add basic automated tests for calculation logic.	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:	**Jonas**	**Date:**	**2/20/2026**
Student Signature:	**Jonas Marcial**	**Date:**	**2/20/2026**
Instructor Signature:		**Date:**	