CODE REVIEW EVALUATION FORM

JavaScript & Express.js | Undergraduate Programming Course

1. SUBMISSION INFORMATION

Course:	**ICS 385**	**Section:**	
Instructor:	**Dr. Bhattacharya**	**Semester:**	**Spring 2026**
Student Name:	**Todd Yoshioka**	**Student ID:**	
Project Title:	**Secrets javascript**	**Date:**	**2/14/2026**
Reviewer:	**Todd Yoshioka**	**Review Type:**	Peer / Instructor

2. CODE SUBMISSION DETAILS

Repository URL:	**https://github.com/toddhy/ics385spring2026/blob/main/week5/3.5%20Secrets%20Project/solution.js**		
Branch:	**main**	**Commit Hash:**	
Files Reviewed:	**solution.js**	**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: The code is an Express server that routes user to different HTML pages based on whether they submit a correct password in the box or not. Routes are a GET to index.html, and POST to either secret.html or index.html. Middleware used is bodyparser and a custom function called passwordCheck, which checks user input against a hardcoded password.

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	Application runs without errors; all Express routes return expected responses; edge cases handled.	4	20%

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.	4	15%
Error Handling & Validation	Input validation present; try/catch or .catch() used; meaningful error messages returned to client.	1	10%
Comments & Documentation	Inline comments explain non-obvious logic; README or header comments describe setup, dependencies, and usage.	1	10%
Security Considerations	No hardcoded secrets; use of environment variables; input sanitization; helmet or CORS configured if applicable.	1	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:	2.8 / 4.00	**Percentage:**	70 %

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	Hardcoded password	High / Med / Low	High	Line 16 of solution.js.

2	No documentation	High / Med / Low	Low	
3	No error handling	High / Med / Low	Low	
4		High / Med / Low		
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)	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)	y
Routing	Route parameters and query strings used correctly	y
Middleware	Body-parser or express.json() configured for request parsing	y
Middleware	Custom middleware is reusable and well-documented	n
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	n
Dependencies	package.json lists all dependencies; no unused packages	y
Dependencies	node_modules excluded via .gitignore	n

Security	Environment variables managed via .env / dotenv	n

Security	No sensitive data committed to version control	n

7. QUALITATIVE FEEDBACK

Strengths — What does this submission do well?

: It works as intended. Code is simple and easy to follow. Only essential modules loaded.

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

: Main thing is hardcoded password needs to be removed. Could have some explanatory comments and error handling, although program is so simple that maybe not necessary.

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:	**B**	**Numeric Score:**	89 / 100

9. REQUIRED REVISIONS & ACTION ITEMS

List any mandatory changes the student must complete before resubmission.

1	Remove hardcoded password by implementing an alternative.	High / Med / Low	High
2	Explanatory comments and error handling.	High / Med / Low	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:	**Todd Yoshioka**	**Date:**	**2/14/26**
Student Signature:		**Date:**	
Instructor Signature:		**Date:**	