CODE REVIEW EVALUATION FORM

JavaScript & Express.js | Undergraduate Programming Course

1. SUBMISSION INFORMATION

Course:	ICS 385	**Section:**	
Instructor:		**Semester:**	Spring 2026
Student Name:	Lexter Cortez	**Student ID:**	
Project Title:	Week 5 Secret Project	**Date:**	2/15/2026
Reviewer:	Lexter Cortez	**Review Type:**	Peer / Instructor

2. CODE SUBMISSION DETAILS

Repository URL:	https://github.com/cortezlb-coder/ics385spring2026/tree/main/week5		
Branch:		**Commit Hash:**	
Files Reviewed:		**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:

This code is a site that has a log in screen and password protected content behind it.
The express.js framework creates the web server and handles the requests.
The GET route "/" points to index.html while the POST route "/check" if the password is correct goes to secret.html
The middleware that checks that password is passworkCheck.

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.		20%

Criterion	Description	Score (0–4)	Weight
Code Structure & Organization	Logical file/folder structure (e.g., routes/, controllers/, models/); separation of concerns; modular design.		15%
Naming Conventions & Readability	Variables, functions, and routes use clear, descriptive names following camelCase conventions; consistent formatting.		10%
Express.js Best Practices	Proper use of Router, middleware chaining, error-handling middleware, appropriate HTTP methods and status codes.		15%
Error Handling & Validation	Input validation present; try/catch or .catch() used; meaningful error messages returned to client.		10%
Comments & Documentation	Inline comments explain non-obvious logic; README or header comments describe setup, dependencies, and usage.		10%
Security Considerations	No hardcoded secrets; use of environment variables; input sanitization; helmet or CORS configured if applicable.		10%
Testing & Reliability	At least basic test cases provided (e.g., using Jest or Supertest); tests cover primary routes and edge cases.		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	solution.js/ 16	High / Med / Low high		password in plaintext
2	solution.js/ 17	High / Med / Low		allows everyone to login after one correct
3		High / Med / 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.

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()	
Routing	RESTful conventions followed (GET, POST, PUT/PATCH, DELETE)	
Routing	Route parameters and query strings used correctly	
Middleware	Body-parser or express.json() configured for request parsing	
Middleware	Custom middleware is reusable and well-documented	
Middleware	Error-handling middleware defined with (err, req, res, next) signature	
Async/Await	Promises and async/await used correctly (no unhandled rejections)	
Async/Await	Callback patterns avoided in favor of modern async patterns	
Dependencies	package.json lists all dependencies; no unused packages	
Dependencies	node_modules excluded via .gitignore	

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

7. QUALITATIVE FEEDBACK

Strengths — What does this submission do well?

:

Easy to read and straight forward

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

:

Securing the site, Another one is that the source code has the password in plaintext.
I would remedy this requiring a user and password alongside giving each user a session rather than one correct password enabling all to enter. I would also add ways to prevent a brute force by adding timeouts and the more failed attempts the longer the timeout is.

Suggested Learning Resources

:

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		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:		**Date:**	
Student Signature:		**Date:**	
Instructor Signature:		**Date:**	