CODE REVIEW EVALUATION FORM

JavaScript & Express.js | Undergraduate Programming Course

1. SUBMISSION INFORMATION

Course:	**ICS 385**	**Section:**	
Instructor:	**Debasis Bhattacharya**	**Semester:**	
Student Name:	**Coralia Montalvo**	**Student ID:**	
Project Title:	**Secrets Project - Security Code Review**	**Date:**	**2/15/2026**
Reviewer:		**Review Type:**	Peer / Instructor

2. CODE SUBMISSION DETAILS

Repository URL:	**https://github.com/coraliam-dev/ICS385-coraliam-dev-bio.git**		
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: The submitted code is a web server called Secrets designed to protect this secret behind a password. When a user submits the password "ILoveProgramming" they are granted access to a secret page. If the password is wrong, they are redirected to the login page. This shows basic password checking and routing in a Node.js/Express web application. Body parsers are used as middleware to parse form data from POST requests.

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.		20%

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.

1	index.js Line 21	**High** / Med / Low	Security	The secret is stored as a plaintext in the source code
2	index.js Line 21	High / Med / Low	Security	The Password “ILoveProgramming” is hardcoded in the POST route.
3		**High** / Med / Low	Transport Security	Without HTTPS all data can be intercepted by attackers on the network.
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)	
Server Setup	Entry point file is clearly identified (e.g., app.js or server.js)	Y
Routing	Routes are organized using express.Router()	N
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	

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?

: The application is successful opening up a web server.

It uses middleware to capture user input and handles routing between the login and secret pages.

Code is pretty easy to read.

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

: Store passwords more securely. Use a secret setting so others cannot see.

Make sure the website used HTTPS so no one can listen in and hack/steal.

Show a message if the password was incorrect.

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:		**Numeric Score:**	_____ / 100

9. REQUIRED REVISIONS & ACTION ITEMS

List any mandatory changes the student must complete before resubmission.

1	Remove Hardcoded Credentials	**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:	**Coralia Montalvo**	**Date:**	**02/15/2026**
Student Signature:		**Date:**	
Instructor Signature:		**Date:**	