CODE REVIEW EVALUATION FORM

JavaScript & Express.js | Undergraduate Programming Course

1. SUBMISSION INFORMATION

Course:	ICS 385	**Section:**	Week 5
Instructor:	Debasis Bhattacharya	**Semester:**	Spring 2026
Student Name:	Bryceson Gaoiran	**Student ID:**	28810114
Project Title:	Week 5d - Code Review of Secrets	**Date:**	2/10/2026
Reviewer:	Bryceson Gaoiran	**Review Type:**	Peer / Instructor

2. CODE SUBMISSION DETAILS

Repository URL:	https://github.com/Bryceson-ai/ics385spring2026/blob/main/week5/3.5%20Secrets%20Project/Bryceson_Code_Review_JS_Express.pdf		
Branch:	Main	**Commit Hash:**	
Files Reviewed:	Secrets	**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.js server that uses a body parser middleware to check the submitted password against "ILoveProgramming" on the POST /check route and outputs either the secrets page or the login page based on the authentication.

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.	1	15%
Naming Conventions & Readability	Variables, functions, and routes use clear, descriptive names following camelCase conventions; consistent formatting.	2	10%
Express.js Best Practices	Proper use of Router, middleware chaining, error-handling middleware, appropriate HTTP methods and status codes.	2	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.	1	10%
Security Considerations	No hardcoded secrets; use of environment variables; input sanitization; helmet or CORS configured if applicable.	0	10%
Testing & Reliability	At least basic test cases provided (e.g., using Jest or Supertest); tests cover primary routes and edge cases.	0	10%

Total Weighted Score:	1.36 / 4.00	**Percentage:**	34% %

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, line 14	High / Med / Low	Security	Hardcoded password "ILoveProgramming" is in plain text when it should be using environment variables and bcrypt hashing
2	solution.js, line 11	High / Med / Low	Security	Global userIsAuthorised flag persists across all users; one user authenticating affects all sessions
3	solution.js, line 1-40	High / Med / Low	Structure	All code in single file; no separation of concerns; routes, middleware, and logic mixed together
4	solution.js, line 16	High / Med / Low	Error Handling	No validation on req.body["password"]; could be undefined/null, causing silent failures
5	solution.js, line 23-29	High / Med / Low	Express Best Practices	POST route doesn't set appropriate HTTP status codes (200 vs 401); doesn't use proper authentication patterns (sessions/JWT)
6	solution.js, line 12	High / Med / Low	Documentation	No comments explaining middleware flow, authentication logic, or file serving logic
7	solution.js, line 39	High / Med / Low	Documentation	No README explaining how to run project, set password, or configure port
8	solution.js, line 8	High / Med / Low	Best practice	Hard-coded port 3000 should be configurable via environment variable for flexibility across environments

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
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)	Y
Routing	Route parameters and query strings used correctly	N/A
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)	N/A
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/A

Category	Checklist Item	Y / N / N/A
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 submission demonstrates the fundamental Express.js concepts well and has a working server that serves HTML files and handles user input. Another thing that the submission does well is that the middleware integration with body-parser is implemented correctly, and the application runs without crashes.

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

:

The student should focus on prioritizing security with the use of environment variables for sensitive data. They should also refactor the code into modular files and add error handling with an informative message to the users when authentication fails.

Suggested Learning Resources

:

https://www.npmjs.com/package/bcryptjs for password hashing

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: D		**Numeric Score:**	65 / 100

9. REQUIRED REVISIONS & ACTION ITEMS

List any mandatory changes the student must complete before resubmission.

#	Action Item	Priority	Due Date
1	Implement environment variables with a .env file to store the password and port	High / Med / Low	3/1/2026
2	Add password hashing using bcryptjs	High / Med / Low	3/1/2026
3	Add input validation and error-handling middleware with meaningful error messages to user	High / Med / Low	3/1/2026
4	Add inline comments documenting authentication flow, middleware, and key logic	High / Med / Low	3/1/2026

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:	Bryceson Gaoiran	**Date:**	2/11/2026
Student Signature:		**Date:**	
Instructor Signature:		**Date:**	