CODE REVIEW EVALUATION FORM

JavaScript & Express.js | Undergraduate Programming Course

1. SUBMISSION INFORMATION

Course: ICS 385	**Section: 24**
Instructor: Debasis Bhattacharya	**Semester: Spring 2026**

Student Name: Noah Munz	**Student ID: 23760630**
Project Title: 3.5 Secrets Project	**Date: 2/14/2026**
Reviewer: Self / Instructor	**Review Type:** Peer / Instructor

2. CODE SUBMISSION DETAILS

Repository URL: https://github.com/Noahmunz21/ics385spring2026/tree/main/wk5/3.5%20Secrets%20Project	
Branch: 'Main'	**Commit Hash: Commit 0bd5d5a (Shortened Version)**
Files Reviewed:index.js, solution.js, package.json, index.html, secret.html, and package-lock.json.	**Lines of Code: 110 Total excluding package-lock.json; 752 if it was included.**

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 specific project was a minimally detailed Express.js web server page that stores and protects a secrets page behind a password. The code uses two different but specific routes; one being a GET / that servers as a login form (index.html), and the POST / check validates the submitted and identified password using custom checkpassword middleware. This 'middleware' compares the form and its input against a hardcoded string/identifier. It the string is correct and match up a global boolean (true/false - userISauthorized) is set to true and the secret.html file is served accordingly; if not, the enduser wil be redirected to the original login page prompting account credentials. The two dependencies are Express 4 and body-parser (I think). The HTML files included serve directly from the public folder using the _dirname path resolution.

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.	2	15%
Naming Conventions & Readability	Variables, functions, and routes use clear, descriptive names following camelCase conventions; consistent formatting.	3	10%
Express.js Best Practices	Proper use of Router, middleware chaining, error-handling middleware, appropriate HTTP methods and status codes.	3	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.	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:	2.15 / 4.00	**Percentage:**	53.75 %

#	File / Line	Severity	Category	Description / Observation
1	**solution.js / 7**	High / Med / Low = **HIGH**	**Security**	**The password was literally hardcoded in plaintext. Anyone with source access can read the 'secret'**
2	**solution.js / 11**	High / Med / Low= **HIGH**	**Security**	**The global userisauthorized flag; where one login unlocks for all users.**
3	**solution.js / 3**	High / Med / Low = **MED**	**Express Practice**	**Port hardcoded to 3000; use process.env.PORT 3000 for more flexibility.**
4	**solution.js / 26**	High / Med / Low = **MED**	**Error Handling**	**Failed login attempt returns the error code HTTP 200, but it should be 401; unauthorized.**
5	**All the files**	High / Med / Low = **MED**	**Documentation**	**No comments, README file, or input/setup script or instructions/guide.**
6	**All the files**	High / Med / Low = **MED**	**Testing**	**There were no test cases prior. The testing was done manually.**
7	**solution.js**	High / Med / Low = **LOW**	**Structure / Syntax**	**All the login was in one file, it should be diversified evenly for more flexibility and increased security.**
8	**solution.js**	High / Med / Low = **LOW**	**Express Practice**	**There was no error-handling middleware used; these errors provide guidance to diagnose issue.**

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	**Y**
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/A**
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?

:

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

:

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:	**Noah Munz**	**Date:**	**2/14/2026**
Instructor Signature:		**Date:**	