Disclaimer: Code understood via AI comments generated by Codex on VSCode.

CODE REVIEW EVALUATION FORM

JavaScript & Express.js | Undergraduate Programming Course

1. SUBMISSION INFORMATION

Course:	ICS 385 Web Development and Administration	**Section:**	N/A
Instructor:	Dr. Debasis Bhattacharya	**Semester:**	Spring
Student Name:	N/A	**Student ID:**	N/A
Project Title:	5d - Secrets	**Date:**	02/15/2026
Reviewer:	Lilith Gannone	**Review Type:**	Peer / Instructor

2. CODE SUBMISSION DETAILS

Repository URL:	https://github.com/debasisb/ics385spring2026/tree/main/week5/3.5%20Secrets%20Project		
Branch:	main	**Commit Hash:**	
Files Reviewed:	index.js, package.json, solution.js, secret.html, index.html	**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:

Purpose: The purpose of the secrets code is to output the information in the secret.html file to users who correctly input the hardcoded password to the HTML form. If the password is input correctly, the public/secret.html is sent. If not, the index page is sent again. A basic Express server is demonstrated in the code.
Functionality and Express.js Routes: The main functionalities are demonstrated in GET and POST (Express routes). GET renders the initial page. POST decides if the secret.html content should be sent or if the original page will be sent again based on if user submission meets the "function passwordCheck(req, res, next)" criteria.
Middleware: "app.use(bodyParser.urlencoded({ extended: true }));" is used to allow reading of "req.body["password"]". "passwordCheck(req, res, next)" is also a form of middleware. "app.use(passwordCheck);" applies the middleware to the POST / check.
External Packages: Express and body-parser as seen in "import express from "express";" and "import bodyParser from "body-parser";". The other two import code lines are node built ins.

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.	2	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.	1	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.	0	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.	2	10%

Total Weighted Score:	1.45 / 4.00	**Percentage:**	36 %

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	specific issue	Hardcoded password visible as plaintext is source code.
2	solution.js/10	High / Med / Low	specific issue	"userIsAuthorized" is not unique to the user. Authorization shared across all users until server restarts.
3	full project	High / Med / Low	noteworthy pattern	No comments added to code. No readme.md file.
4	solution.js/8	High / Med / Low	specific issue	Server does not listen on a configurable port. "const port = 3000" is hardcoded.
5	full project	High / Med / Low	noteworthy pattern	No error handling.
6	solution.js/31	High / Med / Low	specific issue	No real handling for unauthorized user status. Login page is returned when password is input incorrectly.
7	index.htm/11	High / Med / Low	specific issue	Password is visible on screen. Should be type="password".
8	full project	High / Med / Low	noteworthy pattern	Unstyled HTML. No error messages or authorized/ unauthorized messages. No accessibility enhancements. Overall visual formatting needs improvement.

6. EXPRESS.JS & JAVASCRIPT CHECKLIST

Understood complex categories by adding comments to VS code identifying specifically where these items would be present/ what is not applicable.

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)	N
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/A
Dependencies	package.json lists all dependencies; no unused packages	N
Dependencies	node_modules excluded via .gitignore	N

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 code is functional. It runs and demonstrates a basic Express server. It stands as a good resource for beginner coders first learning about middleware. Its authentication approach is very insecure, but the code itself is functional. The code is also clear and understandable for beginner coders.

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

:

Security should be the number one priority as far as improvements go. The main issues are that the hardcoded password is visible in the source code, the authorization can be shared across multiple users across the server, there is no handling for unauthorized inputs, and the typed password is visible on the screen. Error handling, visual and accessibility improvements, and the hardcoded server port could also be improved. Comments should also be added to the code. A readme.md file should be added, as well.

Suggested Learning Resources

:

To improve this code, the developer could look into W3Schools, Udemy, and Coursera. W3Schools and Udemy can help to explain Express and Node best practices, syntax, and basics. Coursera can help users to better understand the importance of accessibility and creating user-centered user interfaces. The developer should also look into authentication and security best practices.

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:	65%	**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	Address finding 1: Hardcoded password visible as plaintext is source code. Need to remove hardcoded password from code.	High / Med / Low	
2	Address finding 2: Ensure authorization is user specific.	High / Med / Low	
3	Address finding 6: Give user feedback if password is input incorrectly. Possibly lock out users for invalid input.	High / Med / Low	
4	Address finding 7: Password is visible on screen. Should be type="password" not type="text".	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:	Lilith Gannone	**Date:**	02/15/2026
Student Signature:	N/A	**Date:**	
Instructor Signature:	N/A	**Date:**	