Royal University of Bhutan

འབྲུག་རྒྱལ་འཛིན་གཙུག་ལག་སློབ་གྲྭ།

College of Science and Technology
Rinchending: Bhutan

College of Science and Technology

RUB Wheel of Academic Law: Academic Dishonesty

Section H2 of the Royal University of Bhutan's *Wheel of Academic Law* provides the following definition of academic dishonesty:

Academic dishonesty may be defined as any attempt by a student to gain an unfair advantage in any assessment. It may be demonstrated by one of the following:

1. **Collusion:** the representation of a piece of unauthorized group work as the work of a single candidate.
2. **Commissioning:** submitting an assignment done by another person as the student's own work.
3. **Duplication**: the inclusion in coursework of material identical or substantially similar to material which has already been submitted for any other assessment within the University.
4. **False declaration**: making a false declaration in order to receive special consideration by an Examination Board or to obtain extensions to deadlines or exemption from work.
5. **Falsification of data**: presentation of data in laboratory reports, projects, etc., based on work purported to have been carried out by the student, which has been invented, altered or copied by the student.
6. **Plagiarism**: the unacknowledged use of another's work as if it were one's own.

Examples are:

- verbatim copying of another's work without acknowledgement.
- paraphrasing of another's work by simply changing a few words or altering the order of presentation, without acknowledgement.
- ideas or intellectual data in any form presented as one's own without acknowledging the source(s).
- making significant use of unattributed digital images such as graphs, tables, photographs, etc. taken from test books, articles, films, plays, handouts, the internet, or any other source, whether published or unpublished.
- submission of a piece of work which has previously been assessed for a different award or module or at a different institution as if it were new work.
- use of any material without prior permission of copyright from the appropriate authority or owner of the materials used".

Royal University of Bhutan

འབྲུག་རྒྱལ་འཛིན་གཙུག་ལག་སློབ་སྡེ།།

College of Science and Technology
Rinchending: Bhutan

Practical Assignment 1 - Building a RESTful API with Node.js Standard Libraries

Introduction

In this assignment, you are tasked with developing a HTTP server using only the Node.js Standard Libraries. The API should support CRUD (Create, Read, Update, Delete) operations for a resource of your choice (e.g., products, users, blog posts). The primary objective is to gain hands-on experience in building a server-side application and designing a RESTful API architecture.

Requirements

Resource Selection:

- Choose a resource for your API (e.g., products, users, blog posts).
- Define the properties or fields associated with your chosen resource.

Server Setup:

- Use the `http` module from the Node.js Standard Libraries to create an HTTP server.
- Configure the server to listen on a specific port (e.g., 3000).

Data Storage:

- Use the `fs` module from the Node.js Standard Libraries to store and retrieve resource data.
- Create a JSON file to store your resource data.
- Implement CRUD operations by reading from and writing to the JSON file.

Routing and Endpoints:

- Implement the following endpoints for your chosen resource:
    - `GET /resource`: Retrieve a list of all resources.
    - `GET /resource/:id`: Retrieve a specific resource by its ID.
    - `POST /resource`: Create a new resource.
    - `PUT /resource/:id`: Update an existing resource by its ID.
    - `PATCH /resource/:id`: Partial update of an existing resource by its ID.

- `DELETE /resource/:id`: Delete a resource by its ID.

Request Handling:

- Parse incoming request bodies (for `POST` and `PUT` requests) using the appropriate Node.js Standard Libraries.
- Validate incoming data and handle errors appropriately.
- You must use `try...catch...finally` blocks to handle exceptions and ensure proper resource cleanup.

Response Handling:

- Send appropriate responses with relevant status codes (e.g., 200 OK, 201 Created, 404 Not Found) and data payload (if applicable).
- Handle edge cases and error scenarios gracefully.
- You must use `try...catch...finally` blocks to handle exceptions and ensure proper resource cleanup.

Documentation:

- Document your API endpoints, including the expected request/response formats and any specific requirements or constraints.
  - Example: https://docs.github.com/en/rest/using-the-rest-api/getting-started-with-the-rest-api?apiVersion=2022-11-28
  - https://docs.github.com/en/rest/actions/artifacts?apiVersion=2022-11-28

Expected Outcomes

By the end of this assignment, you should have a fully functional RESTful API built using only the Node.js Standard Libraries. The API should support CRUD operations for your chosen resource, handle requests and responses correctly, and include appropriate error handling and validation mechanisms. The resource data should be stored and retrieved from a JSON file using the `fs` module. Additionally, you should implement `try...catch...finally` blocks to handle exceptions and ensure proper resource cleanup during request and response handling.

You should have gained practical experience in server-side development, API design, and working with Node.js Standard Libraries. Your assignment submission should include the source code, documentation, and any additional instructions or notes relevant to running and testing your API.

Note: While this assignment restricts you to using only the Node.js Standard Libraries, in real-world scenarios, you would typically leverage external libraries and frameworks (e.g., Express.js, Hono) to streamline the development process and take advantage of established best practices and features.

Submission

Submit the following to the sheet provide here WEB102 - CAP1 SUBMISSION :

1. Your Github Repository containing your working web application including a descriptive README.md. Please note that as the assignment doesn't require a report submission, ensure that the comments successfully suffice in supporting your thought process through this assignment. You are required to define the functionality in your README.md that you application would achieve.
    a. **Github repository name format**: “stdno_WEB102_PA1” (eg: 02190108_WEB102_PA1)
    b. **Submission Portal**: Github
    c. **Deployment to Cloud:** Render- ”xxxxx.onrender.com” OR any other cloud provider.

Date of Submission:

15th May, 2024

Note:

- Plagiarism of work without referencing and understanding the work will not be tolerated.
- Late submission of assignments will cause a 5% deduction in grades every day from the date of submission.
- Any mistake in file format, name of file etc will result in a deduction of 1 mark on every mistaken submission.

Grading Criteria

This continuous assessment will be assessed based on the following criteria:

Functional Requirements	- 10.0%
Code Quality and Efficiency	- 10.0%
Documentation and Code Organization	- 5.0%

Bonus Marks: Authentication (5 MARKS)

Note: You will not be able to score higher than the maximum grade.

- Use the `crypto` module from the Node.js Standard Libraries to implement authentication for your API.
- Choose either the `hkdf` or `pbkdf2` algorithm from the `crypto` module to hash and salt passwords.
- Create two new endpoints: `/register` and `/login`.
- The `/register` endpoint should allow users to create an account by providing a username and password.
- The `/login` endpoint should authenticate users by verifying their credentials and issuing a session token or JWT (JSON Web Token).
- Implement appropriate error handling and validation for the authentication endpoints.
- Create a middleware function to process authenticated requests for protected resources (e.g., specific CRUD operations).
- Implement the middleware function to verify the provided session token or JWT and grant access to protected resources accordingly.

Penalties

Note: You will not be able to score less than zero.

- **[-50% of overall grade of assignment]** - Student is unable to justify and explain code logic of his/her implementation