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 taken from test books, articles, films, plays, handouts, the internet, or any other source, whether published or unpublished.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".

རྒྱལ་ཡུལ་འཛིན་གཏུག་ལག་སློབ་སྡེ།།

College of Science and Technology
Rinchending: Bhutan

Practical Assignment 2 - Building a RESTful API with 3rd Party Libraries

Introduction

In this practical assignment, you will demonstrate your skills in building a robust and scalable RESTful API. You will be tasked with designing and implementing a complete API solution using a modern web framework i.e Hono. The assignment will assess your ability to organize routes and controllers effectively, ensuring a clean and maintainable codebase.

A key aspect of the assignment is integrating your API with a database using an Object-Relational Mapping (ORM) tool like Prisma. You will be required to implement full CRUD (Create, Read, Update, Delete) operations, allowing clients to interact with the data stored in your database.

Furthermore, you will need to incorporate essential features such as authentication (token or session-based), rate-limiting, and pagination. These features are crucial for ensuring the security, performance, and usability of your API.

Throughout the assignment, you will be expected to follow best practices in API design, code organization, and documentation. You should aim to create a well-structured, efficient, and secure API .

By completing this assignment, you will gain valuable hands-on experience in building APIs, working with databases, and implementing essential features for a production-ready application. This practical experience will not only reinforce your understanding of the concepts but also prepare you for real-world scenarios you may encounter as a developer.

So, let's get started! Review the detailed requirements and guidelines provided, and feel free to reach out to your instructors or teaching assistants if you have any questions or need further assistance.

Requirements

1. Implement a RESTful API using a modern web framework - Hono. The framework should handle HTTP requests, routing, and middleware functions.

   You are strictly required to use Hono ONLY. Any other framework used will result in immediate failure.

2. Integrate your API with a database using an ORM (Object-Relational Mapping) tool like Prisma. The database should store data for your API's resources.

   You are strictly required to use Prisma OR official drivers of your chosen database ONLY. Any other libraries used will result in immediate failure.

3. Implement complete CRUD (Create, Read, Update, Delete) operations for the pokemon resource in your API. This should include endpoints for creating, retrieving, updating, and deleting data.
4. Implement authentication mechanisms for your API, either token-based (e.g., JSON Web Tokens) or session-based. This should restrict access to certain endpoints or operations based on user authentication.

   You are strictly required to implement the following functionality.**

   A user should be able to store data on Pokemons that he/she has caught and view what he has caught. This resource should be protected by authentication.

5. Implement rate-limiting mechanisms to prevent abuse or overload of your API. This could involve limiting the number of requests per IP address or per authenticated user within a specific time frame.
6. Implement pagination for endpoints that return your datasets. This will allow clients to retrieve data in smaller chunks, improving performance and usability.
7. Implement proper error handling and response mechanisms for your API. This should include appropriate HTTP status codes and error messages for different scenarios.
8. Document your API's endpoints, request/response formats, authentication mechanisms, and any other relevant information in a README file.

Expected Outcomes

1. A functional RESTful API built using a modern web framework, with clearly defined routes and controllers.
2. Successful integration of the API with a database using an ORM tool, allowing data storage and retrieval.
3. Implementation of CRUD operations for resources, enabling clients to create, read, update, and delete data.
4. Secure authentication mechanisms in place, either token-based or session-based, to restrict access to certain endpoints or operations.
5. Rate-limiting mechanisms are implemented to prevent API abuse or overload, ensuring consistent performance and availability.
6. Pagination support for endpoints that return large datasets, improving usability and performance for clients.
7. Proper error handling and response mechanisms in place, providing clear and informative error messages and HTTP status codes.
8. Comprehensive documentation for the API, including endpoint descriptions, request/response formats, authentication mechanisms, and any other relevant information.
9. Successful cloning of the PokeAPI functionality for the specified resources (Pokemon: https://pokeapi.co/docs/v2#pokemon-section), with identical endpoints, request/response formats, and data structures. Inclusive of the tracking of pokemon caught and saved by a user.

Submission

Submit the following to the sheet provided here https://docs.google.com/spreadsheets/d/1Zc4jgR8plMgKUfQPLtRKC7hRO3M82IhtPQtMfAIMhAE/edit#gid=0:

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_PA2" (eg: 02190108_WEB102_PA2)
b. **Submission Portal**: Github
c. **Deployment to Cloud:** Render- "xxxxx.onrender.com" OR any other cloud provider.

Date of Submission:

9th June, 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%

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