SCHOOL OF COMPUTING

DIPLOMA IN INFORMATION TECHNOLOGY
DIPLOMA IN APPLIED AI AND ANALYTICS

ST0503 BACKEND WEB DEVELOPMENT

2023/2024 SEMESTER 2
PRACTICAL ASSIGNMENT

Objective of Assignment

To allow students to practice what they have learnt in the module by creating a task tracker to gamify the sustainability movement. The goal is to encourage users to complete environmentally-friendly tasks and earn rewards while tracking their progress.

Instructions and Guidelines:

1. The assignment should be done individually and will account for **30%** of your final grade.

2. The assignment should be submitted by **Tuesday, 2 Jan, 2024 08:00 am**.

3. You are **REQUIRED** to do the following for your submission

    - Use the **Github Classroom repository** provided to commit all your progress and changes. It will be reviewed for marking.
    - Complete the CA1 Individual Report. The form is available in BrightSpace under Assignments > Forms > CA1 Individual Report.
    - Prepare the **Declaration of Academic Integrity (SOC) form**. Your Assignment will **NOT** be marked if the form is not submitted. The form is available in BrightSpace under Assignments > Forms > Declaration of Academic Integrity (SOC) form.

- You are also required to **zip up your source code** along and submit it along with your files to Brightspace. You are to remove the **node_module** folder when zipping the files.
- Below is example of what your submission should contain

```
├─ BED-CA1-PYYYYYY-CLASS.zip
├─ Declaration of Academic Integrity (SOC).docx
└─ CA1 Individual Report.docx
```

4. You are required to develop a Backend Server in NodeJS with MySQL Database.

5. The interview will be conducted during the practical lessons on week 12. You are expected to explain the program logic and modify the program during the interview. If you are absent from the interview, **you will be awarded zero mark for the assignment.**

6. **No marks will be awarded**, if the work is copied or you have allowed others to copy your work.

7. **50%** of the marks will be deducted for assignments that are received within **ONE (1) calendar** day after the submission deadline. No marks will be given thereafter.

Warning: Plagiarism means passing off as one's own the ideas, works, writings, etc., which belong to another person. In accordance with this definition, you are committing plagiarism if you copy the work of another person and turning it in as your own, even if you would have the permission of that person.

Plagiarism is a serious offence, and if you are found to have **committed, aided, and/or abetted** the offence of plagiarism, disciplinary action will be taken against you. If you are guilty of plagiarism, you may **fail all modules** in the semester, or even be **liable for expulsion**.

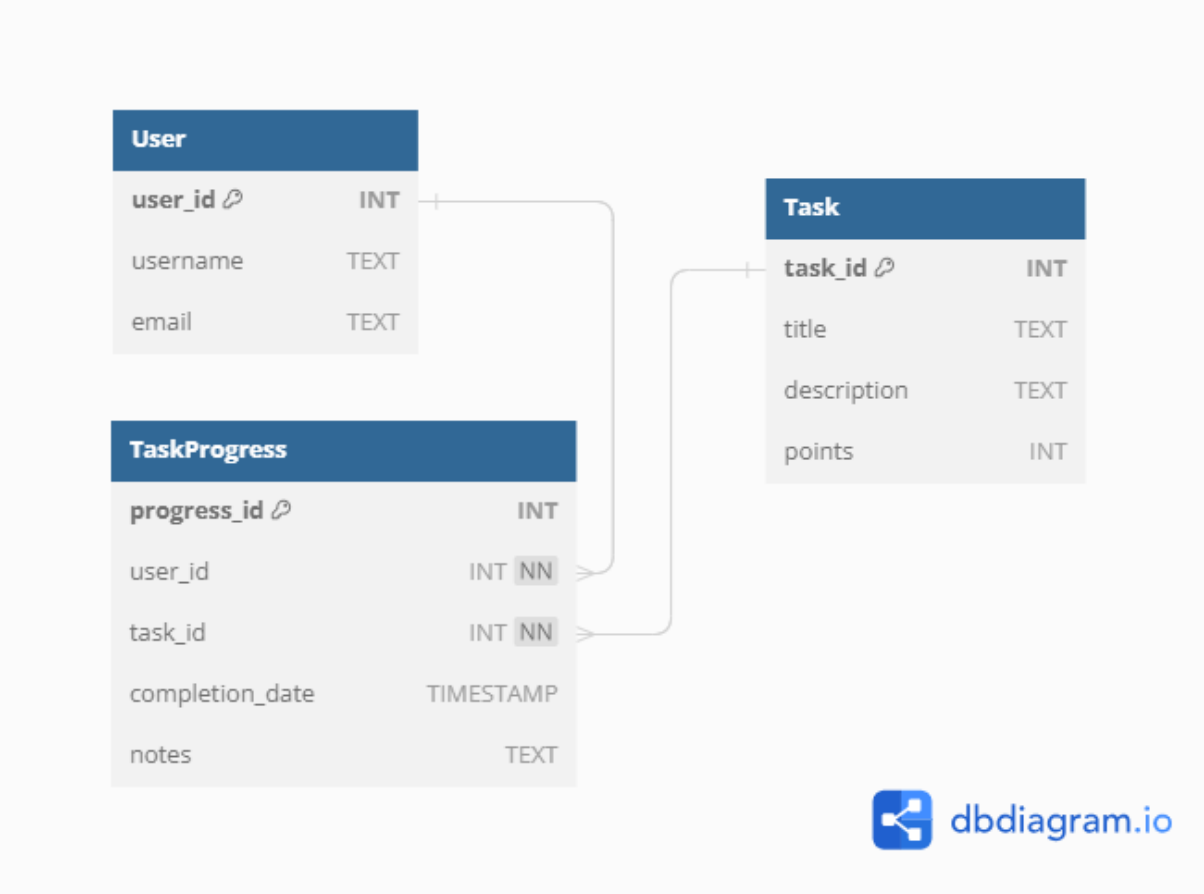
Task Tracker for Gamifying Sustainability Movement

Section A: Project Specification

In this project, we will be working on creating a task tracker to gamify the sustainability movement. The goal is to encourage users to complete environmentally-friendly tasks and earn rewards while tracking their progress. Let's dive into the detailed specifications for the initial database setup and the endpoints requirements.

ERD (Entity-Relationship Diagram):

The initial ERD consists of three tables - User, Task, and TaskProgress.



MySQL Tables:

```
CREATE TABLE `User` (
  `user_id` INT PRIMARY KEY AUTO_INCREMENT,
  `username` TEXT,
  `email` TEXT
);

CREATE TABLE `Task` (
  `task_id` INT PRIMARY KEY AUTO_INCREMENT,
  `title` TEXT,
  `description` TEXT,
  `points` INT
);

CREATE TABLE `TaskProgress` (
  `progress_id` INT PRIMARY KEY AUTO_INCREMENT,
  `user_id` INT NOT NULL,
  `task_id` INT NOT NULL,
  `completion_date` TIMESTAMP,
  `notes` TEXT
);
```

Here's the initial data in table format to insert into the Task table:

task_id	title	description	points
1	Plant a Tree	Plant a tree in your neighbourhood or a designated green area.	50
2	Use Public Transportation	Use public transportation or carpool instead of driving alone.	30
3	Reduce Plastic Usage	Commit to using reusable bags and containers.	40
4	Energy Conservation	Turn off lights and appliances when not in use.	25
5	Composting	Start composting kitchen scraps to create natural fertilizer.	35

Endpoints Requirements for Section A:

1. ## POST /users

   Create a new user by providing their username and email in the request body. Upon successful creation, the response should include the newly generated user_id.

   Example Request Body:

```
{
 "username": "greenUser123",
 "email": "user123@example.com"
}
```

   Example Response:

```
{
 "user_id": 1,
 "username": "greenUser123",
 "email": "user123@example.com"
}
```

   Status Code: 201 Created

   Error Handling:

   - If the provided email is already associated with another user, return 409 Conflict.
   - If the request body is missing username or email, return 400 Bad Request.

2. GET /users

Retrieve a list of all users with their respective user_id, username, and email.

Example Response:

```
[
  {
    "user_id": 1,
    "username": "greenUser123",
    "email": "user123@example.com"
  },
  {
    "user_id": 2,
    "username": "ecoWarrior",
    "email": "warrior@example.com"
  }
]
```

Status Code: 200 OK

3. **GET /users/{user_id}**

   Retrieve details of a specific user by providing their user_id. The response should include username, email, and total points earned by the user.

   Example Response:

```
{
  "user_id": 1,
  "username": "greenUser123",
  "email": "user123@example.com",
  "total_points": 250
}
```

   Status Code: 200 OK

   Error Handling:

   - If the requested user_id does not exist, return 404 Not Found.

4. PUT /users/{user_id}

Update user details by providing user_id in the URL and updating username or email in the request body.

Example Request Body:

```
{
 "username": "sustainableUser",
 "email": "user123_updated@example.com"
}
```

Example Response:

```
{
 "user_id": 1,
 "username": "sustainableUser",
 "email": "user123_updated@example.com"
}
```

Status Code: 200 OK

Error Handling:

- If the requested user_id does not exist, return 404 Not Found.
- If the provided username or email is already associated with another user, return 409 Conflict.

5. **DELETE /users/{user_id}**

Delete a user by providing their user_id. The user's associated task progress and rewards, if any, should also be deleted.

No Response Body

Status Code: 204 No Content

Error Handling:

- If the requested user_id does not exist, return 404 Not Found.

6. **POST /tasks**

Create a new task by providing title, description, and points in the request body. Upon successful creation, the response should include the newly generated task_id.

Example Request Body:

```
{
 "title": "No Plastic Bottles",
 "description": "Avoid purchasing bottled water and use a reusable water bottle instead.",
 "points": 40
}
```

Example Response:

```
{
 "task_id": 6,
 "title": "No Plastic Bottles",
 "description": "Avoid purchasing bottled water and use a reusable water bottle instead.",
 "points": 40
}
```

Status Code: 201 Created

Error Handling:

- If the request body is missing title or description or points, return 400 Bad Request.

7. GET /tasks

Retrieve a list of all tasks with their respective task_id, title, description, and points.

Example Response:

```
[
  {
    "task_id": 1,
    "title": "Plant a Tree",
    "description": "Plant a tree in your neighborhood or a designated green area.",
    "points": 50
  },
  {
    "task_id": 2,
    "title": "Use Public Transportation",
    "description": "Use public transportation or carpool instead of driving alone.",
    "points": 30
  }
]
```

Status Code: 200 OK

8. ## GET /tasks/{task_id}

Retrieve details of a specific task by providing its task_id. The response should include title, description, and points.

Example Response:

```
{
  "task_id": 1,
  "title": "Plant a Tree",
  "description": "Plant a tree in your neighborhood or a designated green area.",
  "points": 50
}
```

Status Code: 200 OK

Error Handling:

- If the requested task_id does not exist, return 404 Not Found.

9. **PUT /tasks/{task_id}**

Update task details by providing task_id in the URL and updating title, description, or points in the request body.

Example Request Body:

```
{
 "title": "Plant Two Trees",
 "description": "Plant two trees in your neighborhood or a designated green area.",
 "points": 60
}
```

Example Response:

```
{
 "task_id": 1,
 "title": "Plant Two Trees",
 "description": "Plant two trees in your neighborhood or a designated green area.",
 "points": 60
}
```

Status Code: 200 OK

Error Handling:

- If the requested task_id does not exist, return 404 Not Found.
- If the request body is missing title or description or points, return 400 Bad Request.

10. DELETE /tasks/{task_id}

Delete a task by providing its task_id. The task's associated task progress, if any, should also be deleted.

No Response Body

Status Code: 204 No Content

Error Handling:

- If the requested task_id does not exist, return 404 Not Found.

11. **POST /task_progress**

    Create task progress for a user (marking a task as completed) by providing user_id, task_id, completion_date, and optional notes in the request body.

    Example Request Body:

```
{
 "user_id": 1,
 "task_id": 1,
 "completion_date": "2023-07-30",
 "notes": "Planted a tree in the park near my house."
}
```

    Example Response:

```
{
 "progress_id": 1,
 "user_id": 1,
 "task_id": 1,
 "completion_date": "2023-07-30",
 "notes": "Planted a tree in the park near my house."
}
```

    Status Code: 201 Created

    Error Handling:

    - If the requested user_id or task_id does not exist, return 404 Not Found.
    - If the request body is missing completion_date, return 400 Bad Request.

12. ## GET /task_progress/{progress_id}

Retrieve details of a specific task progress by providing its progress_id. The response should include user_id, task_id, completion_date, and notes.

Example Response:

```
{
  "progress_id": 1,
  "user_id": 1,
  "task_id": 1,
  "completion_date": "2023-07-30",
  "notes": "Planted a tree in the park near my house."
}
```

Status Code: 200 OK

Error Handling:

- If the requested progress_id does not exist, return 404 Not Found.

13. ## PUT /task_progress/{progress_id}

Update task progress details by providing progress_id in the URL notes in the request body.

Example Request Body:

```
{
 "notes": "Planted two trees this time!"
}
```

Example Response:

```
{
 "progress_id": 1,
 "user_id": 1,
 "task_id": 1,
 "completion_date": "2023-07-30",
 "notes": "Planted two trees this time!"
}
```

Status Code: 200 OK

Error Handling:

- If the requested progress_id does not exist, return 404 Not Found.
- If the request body is missing notes, return 400 Bad Request.

14. DELETE/task_progress/{progress_id}

Delete task progress by providing its progress_id. The task progress should be removed from the database.

No Response Body

Status Code: 204 No Content

Error Handling:

- If the requested progress_id does not exist, return 404 Not Found.

Section B: Design Your Own Themed Tables and Endpoints

In this section, we invite you to explore your creativity and design your own themed tables and endpoints for the gamified sustainability task tracker. Embrace the RPG genre, digital pets, or any other exciting theme that sparks your imagination. Feel free to draw inspiration from popular games, fantasy worlds, or unique concepts to shape your project's narrative.

Theme Ideas:

1. **Fantasy Adventures:** Create tables and endpoints to manage quests, magical items, and character progression. Unveil a world of enchanted creatures, mythical artifacts, and epic quests.
2. **Pet Guardians:** Design tables and endpoints to care for digital pets, including feeding, grooming, and training. Players can embark on quests to unlock new pet breeds and abilities.
3. **Eco-Warriors:** Create a world where players form teams of eco-warriors to complete sustainability challenges, earn rewards, and save the environment.
4. **Cosmic Explorers:** Design a space-themed RPG where players venture into the cosmos to collect rare resources, discover new planets, and establish colonies.
5. **Magical Academia:** Develop tables and endpoints for a magical academy, where students attend classes, learn spells, and participate in magical competitions.

Endpoint Ideas:

1. Create endpoints for accepting and completing quests, with various rewards and consequences based on the choices made.
2. Design endpoints to manage the inventory of magical items, pets, or space exploration gear.
3. Implement endpoints for battling against fantasy creatures, eco-threats, or space challenges.
4. Craft endpoints for character customization, allowing players to choose appearances, powers, and abilities.

5. Create social endpoints for forming alliances, trading items, or participating in events with other players.

Remember, the goal of this section is to have fun and unleash your creativity while building a captivating RPG-themed gamified sustainability task tracker. Your design choices should align with the theme and narrative you choose. Feel free to add elements that make your project unique and engaging.

Assessment Requirements

We will be evaluating different aspects of your gamified sustainability task tracker as described in **Section A and B**. These aspects, or competencies, are critical to the overall success and effectiveness of the application. Please take this as an opportunity to showcase your skills and creativity in designing and implementing a gamified platform that promotes sustainability and eco-friendly behaviours.

Below are the key competencies that will be assessed:

1. **Architecture** (10 Marks)
   This competency assesses the organization of your codebase and the folder structure of your project. We will be looking for a clear and well-defined architectural design that enables easy maintenance, scalability, and expansion.
2. **Dependency Management** (5 Marks)
   In this aspect, we will evaluate how you manage the package.json, external dependencies and libraries used in your project. Proper dependency management ensures stability, security, and efficient updates.
3. **API Design** (10 Marks)
   API design is crucial for creating a user-friendly and intuitive interface for your task tracker. We'll be examining how well you adhere to RESTful conventions and provide consistent and well-documented endpoints.
4. **Middleware Usage** (10 Marks)
   This competency focuses on the use of middleware to streamline request processing. We will assess how effectively you leverage middleware functions in your application.
5. **Database Design** (5 Marks)
   Database design plays a key role in data management and efficiency. We will evaluate your schema design, normalization, and relationship definitions for effective data storage and retrieval.
6. **SQL Queries** (10 Marks)
   This aspect involves assessing your SQL query skills, including query optimization, indexing, and efficient data manipulation to ensure smooth and responsive interactions with the database.

7. **Functionality** (10 Marks)

   Functionality refers to how well your task tracker meets the specified requirements and fulfils its intended purpose. We will be looking for a reliable and well-implemented set of core features.

8. **Code Quality** (15 Marks)

   Code quality is essential for maintainability and readability. We will assess the overall cleanliness, organization, and adherence to coding best practices in your codebase. You may want to read up on coding best practices (example: Naming Conventions, Design Patterns)

9. **Modularity** (10 Marks)

   Modularity involves breaking down your code into logical and reusable modules. We will evaluate how well you've organized your project to promote code reusability and maintainability.

10. **Error Handling** (10 Marks)

    Error handling is critical for providing a seamless user experience. We will examine how well you manage errors, provide informative feedback, and handle exceptional situations gracefully.

11. **Documentation** (5 Marks)

    Documentation is key to helping others understand and work with your code. We will assess the clarity and comprehensiveness of your code comments and external documentation.

Please approach each competency with creativity and attention to detail. The focus is on understanding your approach, decision-making, and implementation choices throughout the project.

Version Control Requirement

In this project, we will be using Git for version control and GitHub Classroom for managing code submissions. Version control is a crucial aspect of software development that allows you to track changes, collaborate effectively, and maintain a clean and organized codebase. To ensure smooth progress and efficient evaluation, it is mandatory for all students to commit their code regularly using Git.

Version Control with Git:

1. **Commit Regularly:** It is essential to commit your code regularly (**minimally once a week**) as you make progress on your project. Regular commits demonstrate a well-organized development process, help track changes, and make it easier to revert to previous states if needed.
2. **Commit Rationale:** For each commit, provide a clear and concise rationale in the commit message. A good commit message explains the changes made in the commit and why those changes were made. This helps reviewers understand your thought process and the purpose of each commit.

GitHub Classroom Submissions:

1. **Use GitHub Classroom:** All code must be committed through GitHub Classroom. Follow the provided repository link to create your personal repository and commit your code into it.
2. **Submission README:** Include a detailed README file in your repository. The README should contain clear instructions on how to run your application, any prerequisites or dependencies, and any additional details that will assist the reviewers in evaluating your project.

Penalties for Non-Compliance:

Failure to adhere to the version control requirements and submission guidelines will result in penalties that could impact your final evaluation. Penalties may include:

- **Disorganized Commits:** Poorly organized commits without clear rationale may lead to a deduction of points in the code quality assessment.

- **Incomplete Submission:** If your submission is incomplete or missing critical components, it may be deemed ineligible for evaluation, resulting in a zero score for the affected parts.

Note: The goal of these requirements is to promote good development practices and ensure a fair and transparent evaluation process. By adhering to the version control guidelines and submission protocols, you'll have a well-documented, organized, and properly managed project.

-- End of Assignment Brief --