Course Assignment Resources

Read the course assignment instructions carefully. If you should not understand anything, please contact one of the teachers. **You are not allowed to discuss the course assignment in any class channels**. The course assignment grade is pass / not pass

IMPORTANT!: Complete the Course Assignment by following the instructions given below.

Your GIT repository in this Assignment needs to be private.

Make sure to give 'noroff-bed1' access to your repository in GitHub.

! ***IMPORTANT: Access to your repository cannot be given AFTER the deadline. If no access is given, this will result in an immediate Not Passed grade***

-SCENARIO-

You have been provided with a web application for Animal adoptions, which uses static data – the data is current stored in JSON format.
The web application is **currently not fully functional.**
The **Front-end of this web application is already set up**. You may update anything in the Front-end, if it is required to make your back-end functional.

After signing into the web application, users can adopt one or more animals from a provided list of animals.
The application must be changed to accommodate MySQL as the database to store animals', logged-in users' and adoption data.

Users will be able to navigate through the application using a navbar, as provided in the web application.
There are five tabs in this navbar, linking to the following pages:

- Home page – landing page.
- Animals page – lists all animals in the database, regardless of adoption status. (Only logged-in users with the 'member' role can adopt animals).
- Species page – only accessible by the 'admin' user role, species information can be updated here.
- Temperament page – only accessible by the 'admin' user role, temperament information can be updated here.
- Sign in page – from there, users will be able to sign in, and sign up.

Your tasks in this course assignment are as follows:

- Change the back-end for this web application to **use a MySQL database.**
- **Create** the relevant **tables**, **columns** and **data types** with the relevant **relationships** between the tables, in the **3rd normal form**, by using the provided data from the table on the Animals page.
- The application needs **authentication** with the username and password in a 'users' table.
- Use Sequelize to handle the **connection** between the client and the database.
- Use Sequelize to create the database structure ONLY, **no initial row data should be added with sequelize.**
- All queries (CRUD) need to be executed with Sequelize, except creating/adding the initial data in the database (Unless otherwise specified).

-SETUP WORKSPACE-

Code:

Download the pre-created web-app zip **(DAB CA.zip)** file from this Course Assignment's "Resources" folder (below) in Moodle.

Install the necessary dependencies for the web application to run correctly if any are needed.

Readme file

In the readme file of your GIT hub project (Use the README file provided in the .zip folder), include the following:

- A detailed description of how to use the application (installation instructions, how to run the application, etc)
- Information on the environment variables that are needed
- Any additional external libraries/packages that were used

- Which NodeJS version is being used

-DATABASE CREATION-

Save the following data creation SQL script to your README file, under the heading “DATABASE”.

Using SQL only:

Create the ‘adoptiondb’ database without any tables or data - Tables will need to be created from within the application, using Sequelize (see instructions below).

-TABLE CREATION-

Using Sequelize only:

Create all the necessary tables in **the 3rd normal form** (Including all relationships) from within the application: (Remember – the application should keep track of adoptions)

- One animal can have many temperaments. Many animals can have the same temperaments.
- An animal can only be one species and one size.
- A user can only have one role – either ‘**admin**’ or **‘member’.**
- Each animal can **only be adopted once**.
- A single logged-in user of the ‘**member**’ role can **adopt many animals**.

-DATABASE POPULATION-

Using SQL only:

Create SQL scripts to **insert all the initial data** (found in the table below) into the necessary tables. Make sure the data is in 3rd normal form.

Save these scripts to your README file, under the heading “DATAINSERTS”.

There should be a single script per table INSERT.

Animals table						
Id	Name	Species	Birthday	Temperament	Size	
1	Coco	Dwarf Hamster	2020-02-12	calm, scared	small	
2	Ted	Tedy bear hamster	2021-02-12	calm, scared	small	
3	Coco	Jack-Russel	2020-02-12	energetic	medium	
4	Everrest	Budgy	2019-02-12	calm, happy	small	
5	Rocko	Tortouse	2020-02-12	calm, lazy	medium	
6	Goldy	Gold Fish	2023-02-12	calm	small	
7	Lizzy	Lizzard	2020-02-12	calm,lazy	medium	
8	Goga	Bearder Dragon	2018-02-12	calm, lazy, scared	large	
9	Tweet Tweet	Parrot	2020-02-12	calm, happy	large	
10	Toothless	Corn snake	2017-02-12	scared	medium	
11	Sophie	Dwarf Hamster	2020-02-12	calm, scared	small	
12	Teddy	Teddy bear hamster	2021-02-12	calm, scared	small	
13	Roger	Parrot	2020-02-18	calm, happy	large	

Insert an **admin** user with a **SQL script**, into the correct table.

Users Table				
id	fullName	username	password	role
1	System admin	Admin	admin1234	admin

2	User	User	user1234	member
3	User2	User	User1234	member

-DATABASE ACCESS-

Save the following user-access SQL script to your README file, under the heading "DATABASEACCESS".

Using SQL only:

- Create a new "dabcaowner" login for the database, with the password "dabca1234" with the "database owner" rights and permissions

-AUTHENTICATION-

The 'users' table should have the username and password for users that have signed-up for this service.

Make sure to:

- Implement the sign-up feature in the application / back-end
- PassportJS must be implemented to authenticate logged-in users, from the 'users' table. (Hashing of passwords is not required)

-ANIMAL ADOPTION-

All these routes must be created in the **animals.js** route file.
POST handlers should be used for data creation, updates and additions, while GET handlers should be used to display a view.
Creation, deletion and update operations must be authenticated.

Non-logged-in users (guest users) should not be able to adopt any animal.

- The "Animals" page should show all the animal records in the table.
- There must be an indication in the table, of which animals have been adopted (Such as the 'Adopted' column).
- Only animals that have not already been adopted, can be adopted.
- Remember, only logged-in users of the **'member'** role can adopt animals.
- Only the logged-in user with the **'admin'** role can cancel any adoption.
- When an adoption has been cancelled, the relevant adoption record should be deleted from the database – **do NOT delete the animal or the user**.
- An animal's age should be **calculated** using the Birthday field – **do NOT store this age field in the database.**

-SPECIES-

All these routes must be created in the **species.js** route file. All the endpoints need to be POST handlers, NOT GET. CRUD operations must be authenticated. Only users of the **'admin'** role should be able to modify/add species data.

- A user with the **'admin'** role can add a new species or update existing species' names.
- A species cannot be deleted if there are any dependencies on that record in the database (e.g.: "Parrot" species cannot be deleted if there is an animal of species, "Parrot").

-TEMPERAMENT-

All these routes must be created in the **temperament.js** route file. All the endpoints need to be POST handlers, NOT GET. CRUD operations must be authenticated. Only users of the **'admin'** role should be able to modify/add temperament data.

- A user of the **'admin'** role can add a new temperament or update existing temperaments.
- A temperament cannot be deleted if there are any dependencies on that record in the database (e.g.: "Calm" species cannot be deleted if there is an animal with the temperament, "Calm").

-DATABASE QUERIES-

Save the following user-access **SQL scripts** to your README file, under the heading "DATABASQUERIES".

Using SQL only, write the following individual queries:

1. Return the most popular animal name.
2. Return a list of animals that have been adopted, and the name of the user that adopted them.
3. Return a list of all the animals, sorted by age from youngest to oldest.
4. Return all the animals born between 31 December 2017 and 31 December 2020.
5. Return the number of animals per size (return each size and the number).

6. CREATE a trigger to implement the following feature - Whenever a new animal of Species type “Lizard” is added to the database, the last created user will automatically adopt that animal.

-FINALLY - SUBMISSION-

1. Your web application **must be committed to your GitHub repository** BEFORE the course assignment deadline.

2. The repository name should be “**FName_LName_DAB_CA_ClassXXYY” (e.g. FJ_BOTHA_DAB_CA_JAN23FT)**
   (Replace 'Class' with your class, e.g. 'Aug', 'Oct', etc)
   (Replace 'XX' with your class year e.g. 22, 23)
   (Replace 'YY' with either FT for Fulltime, or PT for Parttime)

3. The **link to your repository** needs to be submitted on Moodle, in a .**txt file** named the following: “**FName_LName_DAB_CA_ClassXXYY.txt**”
   (Replace 'Class' with your class, e.g. 'Aug', 'Oct', etc)
   (Replace 'XX' with your class year e.g. 22, 23)
   (Replace 'YY' with either FT for Fulltime, or PT for Parttime)

   EXAMPLE: FJ_BOTHA_DAB_CA_JAN23FT.txt

- CHECKLIST-

Grading Checklist	
Database Creation	
	SQL Script correct
Table creation	
	All the relevant tables have been created
	All the relationships are created and correct
Database population	
	All the relevant table insert scripts are provided, and are correct
Database Access	
	SQL Script to create the user
Authentication	
	Sign up feature implemented
	PassportJS used for authentication
Animal Adoption	
	All records are displayed on the Animals view
	Adoption status shown
	Animal age is calculated and displayed - not stored
	Only logged in members can adopt an animal
	Only logged in Admin users can cancel an adoption correctly
Species	
	All the required routes have been created
	The delete operation functions correctly
Temperament	
	All the required routes have been created
	The delete operation functions correctly
Database queries	
	Query 1 - 6 working correctly

DAB CA.zip

Last ned mappe

Previous Activity

Fortsett til...

Next Activity