CIS 3368 SPRING 2024 Final Project

# Daycare Center – Kids, Teachers, Rooms

For your final project you will be building a web application following a standard 3 tier architecture:

**UI – REST API Services – Database**

This application will help users to plan and manage the a kids day care facility.
**Overview**: The user first **logs in** from a **login page.**
The user has the option to manage the facility, the classrooms, the teachers, and the children. A facility consists of several classrooms, and a classroom can be handled by one or more teachers. The children get assigned to classrooms, which each have a certain max capacity. Thus, you will need to create the following three tables:
**facility, classroom**, **teacher**, **child**.

For all tables, the id should be the primary key, and autogenerated. Pick appropriate data types for each column.

The facility table should at minimum contain the following columns:
**id, name**
The **name** allows to give the facility a specific name.

The classroom table should have the following columns:
**id, capacity, name, facility**
The **id** should be automatically generated, the **capacity** column indicates how many children can be assigned to this classroom. The **facility** column is a **foreign key** to the **id column** of the facility table, referencing which facility the classroom is located in. The **name** allows to give the classroo a specific name.

The teacher table should have the following columns:
**id, firstname, lastname, room**
The **id** should be automatically generated, the **firstname** and **lastname** are self explanatory, and the **room** column is a foreign key to the **id column** of the classroom table.

The child table should have the following columns:
**id, firstname, lastname, age, room**
The **id** should be automatically generated, the **firstname** and **lastname** are self explanatory, the **age** column indicates the child's age in years, and the **room** column is a foreign key to the **id column** of the classroom table.

**There is one important rule that must always be adhered to:** a teacher can never watch **more than 10 children at once**. That imposes an important constraint: If a classroom has a capacity of 20, but only 1 teacher is working that classroom, then even though the capacity is 20, only 10 children can be assigned to that classroom. Only if a second teacher is assigned to that room, one should be able to assign children up to the capacity of 20.

The user should be able to manage these tables on **separate pages**, where they can **view, add, update, and delete (CRUD)** the corresponding data. **Note**: When the user adds/updates a (new) **classroom**, the user should have to **pick a facility** that's already been entered and **exists** (e.g. from a dropdown, or a list). The same concept applies to entering/updating a **teacher**, where the **classroom** needs to be able to **be selected from among the available records from the classroom table.** Analogously, a child can only be assigned to a classroom that already exists.

## *Requirements*

This project will be developed in **2 Sprints**. The first sprint focuses on the Python backend development, all the logic/functionality, and the deployment of the REST API services with Flask. The second sprint focuses on the UI development in JavaScript using the EJS. Both **code submissions** will happen into the **same repository** on **GitHub Classroom**. There must be a folder **'frontend'** and a folder **'backend'** in that repo, with the corresponding code bases. **Do not forget to comment and document your references.**

Each Sprint for the project is due at **midnight** of the listed due date on the schedule.

**No late submission will be accepted.**

Part of the submission for Sprint 2 will be a 5-minute long **presentation video**, giving an overview over the functionality of your application features, and explaining the technology, frameworks, etc. that you have used. This video must be made **available through a link** (for instance, recorded with MS Teams and provided is the link to MS Stream). **Do NOT submit video files on CANVAS or in your repo! Commit the link in a text file in your repo.**

## *Details for Sprint 1 - Deadline March 9th, 2024*

For Sprint 1 you will have to design and implement all the REST API services you will need so that your app can interact with your database. (**ALL CRUD operations for all the tables.)**
You will also need a simple **Login API**. You do **not** have to do a complete account creation setup. Single user with pre-configured username and password is fine. You do **not need a users table** either, and you can simply authenticate the user in memory on the backend server. These REST Services need to be implemented using the **Python Flask Framework**. You will be testing your backend code with **Postman**, without the need to have the UI pages implemented.
On CANVAS, **submit the link** to your GitHub repo. If you work as a **team of two,** one person needs to submit the **GitHub classroom link on CANVAS**, and the other person **MUST** submit a note stating the **name, student ID, and section of their partner.**
You are allowed to make modifications and changes to your Sprint 1 code even after the submission deadline, but only the last commit before the deadline will be considered for grading.

## *Details for Sprint 2 - Deadline April 27th 2024*

First, there needs to be a **Login page** that allows the user to log in. Then, the user needs to be able to perform all CRUD operations on all four tables, each on a separate page.
You can use any of the templates introduced in class if you like.

**NOTE: Under no circumstances should the user ever see the ids used as primary keys on any of the pages! (You can, however, introduce secondary Ids of your choice of you want).**

## *Other Requirements*

- Make sure the username and password for the database you setup are credentials you're willing to share. Do not use personal passwords for this homework, which you might be using anywhere else.
- Make sure your project compiles and runs. If the project doesn't run, you will forfeit points.
- Submit early and often! **Only code that has been submitted to your repo can be graded.** Unexpected circumstances can occur (power outages, etc) – make sure you have commits to fall back to if need be.

- Using GitHub Classroom to setup the remote repo for this assignment
  Use the provided link to create your private GitHub repo:

  https://classroom.github.com/a/WpQRsNaU

# Please push your commits on a regular basis.

## What to Turn in

Commit your source code (**properly commented**) to your private repository in the GitHub classroom. **Do not zip the files together.** Your GitHub repository should show multiple meaningful commits illustrating how you worked through the project.
There will be **2 separate assignments** listed in CANVAS - one for each sprint, so that 2 separate grades can be assigned. Each assignment/sprint will be worth **50 points max**. Together, they are worth **100 points max** for the final project (not counting the extra credit).
For **both sprints,** on CANVAS simply submit the **same link** to your GitHub repository containing your project. When working in a team, don't forget to submit a **project document 24h before due date for BOTH sprints**, outlining the work distribution among the team members. It should be **part of your repository** (no separate document submission on CANVAS). For **Sprint 2** submit an **updated project document**. Also remember, **both team members must make a CANVAS submission** as stated above, to be assigned a grade. **This applies to both Sprint 1 and Sprint 2!**
Additionally, you will need to submit the link to your **video presentation**. The video **MUST** be **accessible** for everyone **to stream/view** (verify the link before submitting). **Do NOT upload or commit a video file (.mp4, .avi, etc.)** The presentation video does not have to be narrated/showcased by both team members. However, both team members must be aware of each others code. The video should **first showcase the functionality** of the project (without any technical details, 1-2 minutes), followed by a **short technical review** of the project, detailing the concepts, frameworks, techniques, etc used for this project (2-3 minutes). The **total length** of the video should be around **5 minutes**. Submit the link to your video in a text file as part of your repository. **Do NOT submit anything via email!**

Both team members will receive the **same grade for the project**.

## Extra Credit

You can earn **extra credit** if you are able to deploy the Python Flask API and/or the web front-end to the cloud. **Do not attempt this until you have finished the other parts of the project.** There are many options of cloud deployment (AWS, MS Azure) and you are free to choose one that support the technologies of this project (Python Flask backend + NodeJS/EJS based front-end). S**ubmit your public URL of the deployed project in a text file as part of your repository.**

## Basic Grading rubric for the project

| Item | Points in Sprint 1 | Points in Sprint 2 |
|---|---|---|
| Remote DB Tables setup | 10 | - |
| Code for API | 35 | - |
| Correct submission (GitHub commits, CANVAS link) | 5 | - |
| Code for UI (login + 4 pages) | - | 20 |
| Code for API communication (both ways) | - | 25 |
| Correct submission (GitHub, CANVAS, including video) | - | 5 |
| **Total** | **50** | **50** |
| ***Extra credit for cloud deployment*** | **15 points** | **15 points** |