Data Definition Language (DDL) (90pts) – represents the entire DB schema, including table structures, primary and foreign key constraints, datatypes, data constraints, table indexes and relationships. Your DB should consist of at least 12 entities of suitable complexity (3-10 attributes).

1. Table Structures (12)

Data Modeling Language (DML) (90pts) – Represents views (5), queries as stored procedures with varying levels of difficulty (12) and triggers (3) to facilitate information retrieval across your system. Think about the DML in terms of business needs and data administration. In essence, queries are algorithms for information retrieval and should therefore be of a certain complexity to justify their existence. Therefore, a large percentage of your grade will consider such complexity and will be based on your understanding of table joins and query manipulation. Consequently, you are expected to create at least 20 database objects (views, stored procedures and triggers) of varying levels of complexity and provide a rationale for each query.

**Views (5)**

create view republic as

select people\_name, force\_value

from WARRIORS join PEOPLE on

warrior.people\_id = people.people\_id

where faction = 'republic' order by force\_value desc;

Creating a view for Republic is essential for the database as it is one of the most desired views for fans using the database. The Republic versus the Sith is what makes the story of Star Wars. We are able to capture this view easily with joins from our Warriors and PEOPLE entities, searching out for the factions attribute.

create view sith as

select people\_name, force\_value

from WARRIORS join PEOPLE on

WARRIORS.people\_id = PEOPLE.people\_id

where faction = 'sith' order by force\_value desc;

Continuing on the rationale of our making or Republic, the Sith view is the perfect addition to compare attributes amongst both views.

create view useTheForce as

select warriors\_id, people\_name, force\_value

from WARRIORS join PEOPLE on

WARRIORS.people\_id = PEOPLE.people\_id

where force\_value > 0

order by force\_value desc;

We decided to make a useTheForce view as it allows for easy access to show which warriors have force values. The view also showcases who has the top force value.

* 1. Queries as stored procedures (12)
     1. Select \* from species where species = ‘yodas\_race’; *(Easy)*
     2. Select people.name from people join weapon where people.weapon\_name = “lightsaber” (Medium)

select people\_name

from WARRIORS join PEOPLE on

WARRIORS.people\_id = PEOPLE.people\_id

AND force\_value = (

select WARRIORS.People\_id from WARRIORS

where max(force\_value);

This query would be used by the DB manager to search up the name of the person who has the strongest force\_value in the WARRIORS table.

select people\_name, planets\_name

from PEOPLE join PLANETS on

PEOPLE.people\_planet\_of\_origin = PLANETS.planet\_of\_origin

order by people\_name desc;

This query would be used by the DB manager to search up the names of people and their respective planets.

select people\_name, battles\_name

from PEOPLE join WARRIORS on

PEOPLE.people\_id = WARRIORS.People\_id

join BATTLES on

BATTLES.warriors\_id = WARRIORS.warriors\_id

order by people\_name desc;

This query would be used by the DB manager to search up the warriors and all the battles that are cataloged within the database.

**Triggers (3)**

delimiter //

create trigger good\_guy\_points\_trigger

before insert on warriors

for each row

begin

if WARRIORS.faction = 'republic' then

set WARRIORS.force\_value += 100;

end if;

end; //

delimiter ;

Rationale:

The good\_guy\_points\_trigger offers homage to our heroes of the galaxy. The force will always benefit those who follow the proper teachings. We upgrade force\_values to be a part of our game-like simulator in which we can have appropriate entities battling each other.

delimiter //

create trigger bad\_guy\_points\_trigger

before insert on warriors

for each row

begin

if WARRIORS.faction = 'sith' then

set WARRIORS.force\_value -= 100;

end if;

end; //

delimiter ;

Rationale:

The bad\_guy\_points\_trigger punishes warriors hell bent on conquering the galaxy. As a part of fan fiction, this trigger helps with the storyline of the good side beating the dark side.

System (45pts) – Your database system should be integrated into a working system, which allows users to insert, modify and/or delete data and information from the database. Your system must also provide for an additional business constraint that it is not possible to handle with your database alone. A graphical user interface (GUI) will be needed for this requirement. The goal is to let a non-expert user manipulate your system.

### Database Submission Guidelines

For the submission, you will need to include the following:

1. An exported copy of the SQL DDL used to create your database.
2. An exported copy of the SQL inserts necessary to recreate your database.
   1. Or appropriate CSV files and an import script.
3. An exported copy of the SQL DML commands used by your database.
4. A zipped collection of the program application and all related files
   1. If you are using Python, please include a requirements.txt generated by the pip "freeze" command, this will increase the likelihood of getting your program running
   2. If you are using Java, please include the compiled java application
5. A walk-through of your application's features as a multi-page PDF document.
   1. This must include screenshots of your interface interactions that display its functionality.

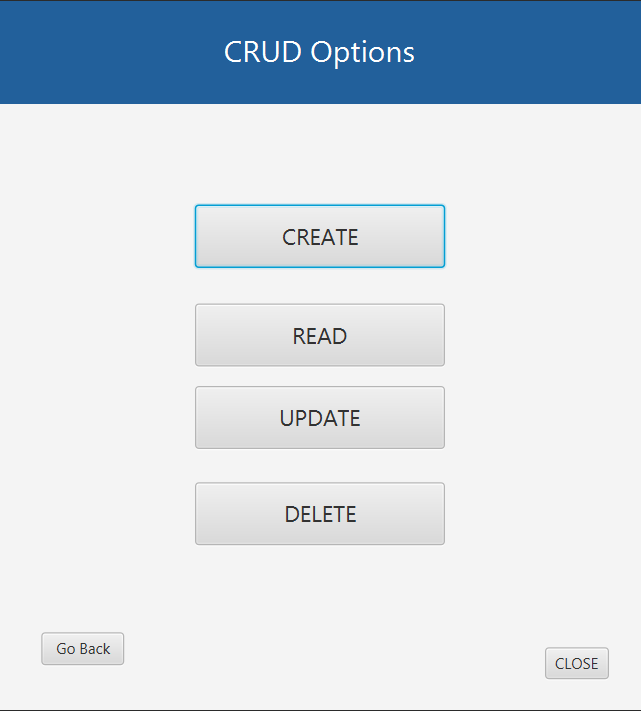
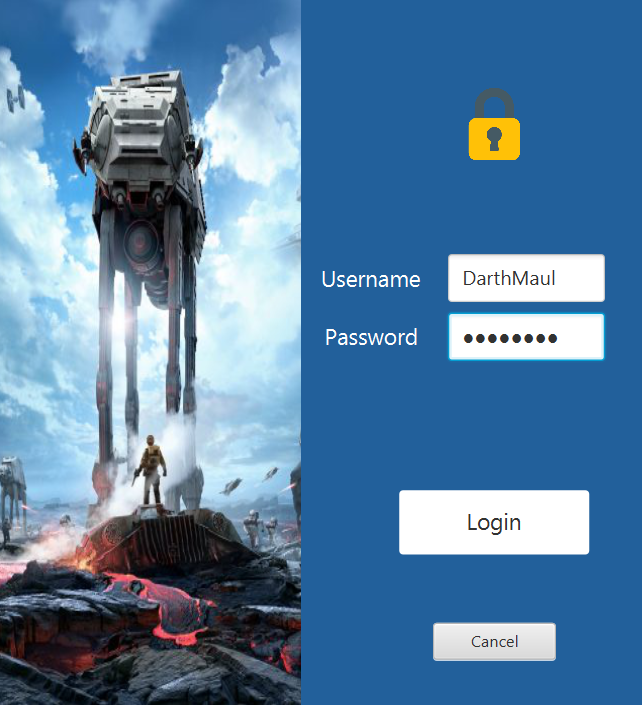
***Make sure that your application provides the ability for me to connect to your database.***

If you are using your CIKeys MariaDB instance, provide me with a user id and password (and let the application make use of it so I can log in.) I should have IP permissions as I will access your database application from the campus network. (You should already have this close to setup as a result of Homework 3.)

If you are not using your CIKeys MariaDB, provide me with the necessary files and scripts to reconstruct your database on a local mysql instance. (This is in addition to the submission requirements above if needed.)

***If I cannot get your application to work, and you did not include an application walkthrough PDF, I cannot give you a final grade for the project or the class.***

WALK THROUGH

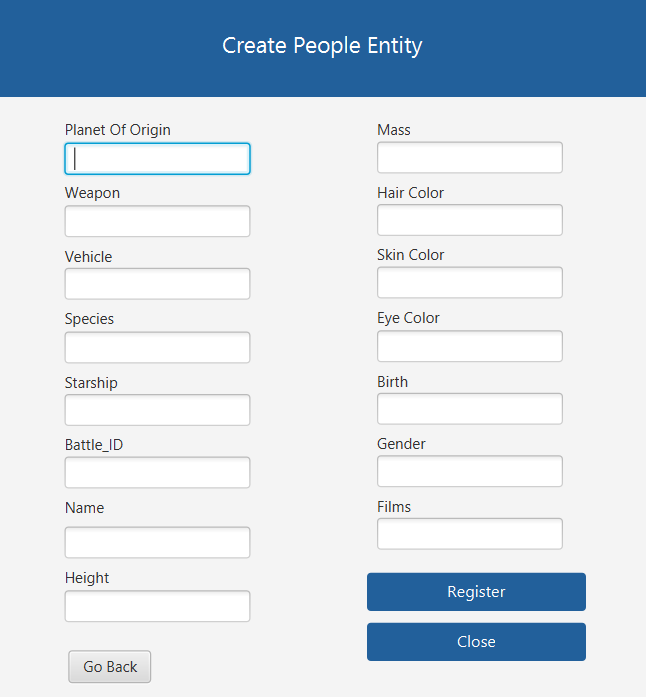
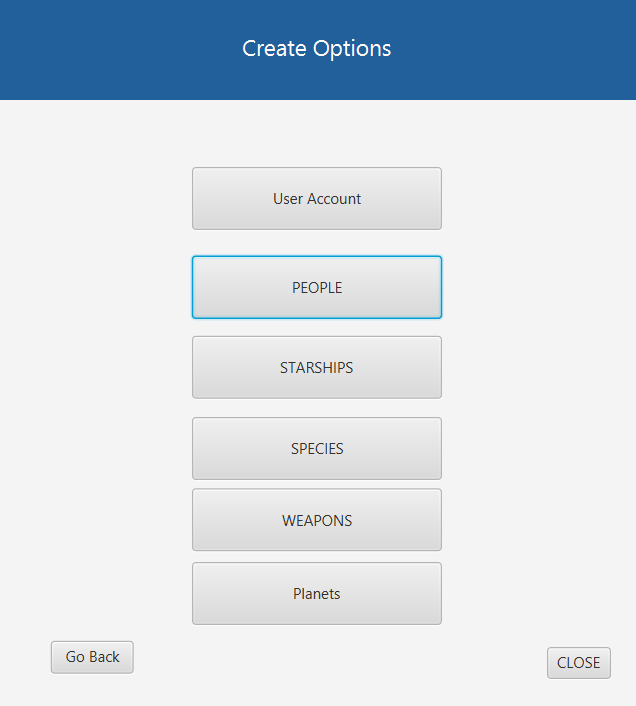


[Login screen] [CRUD options screen]

The Star Wars Database GUI was made using JAVAFX 11, along with Java 11. At the time of this writing, we are working on deploying an executable but are running into some barriers with installing Maven late in the game.

TL, DR: youtube link

Upon running the GUI application, the user is prompted with a login prompt. This login feature serves as our security barrier to protect the data within. We made logging in easy in some cases (for speed of testing), such as typing DarthMaul for username and typing password for the password field. After a successful login, the user is prompted with the CRUD options of: create, read, update and delete.



[create entities screen] [create people screen]

If the user selects create, they will then be asked to choose which entity that they would like to create. In our example, the user wishes to create an entity in the PEOPLE table. We have the available text fields for inputting data. It’s important to note that a primary key is not necessary to add in, we have our database table to automatically create one.