Kyle Bell
Jeremy Tsang

Project Step 6

Executive Summary

We did not make many changes to our project after our original ideas in step 1 since there was nothing major that did not fit. We did make some smaller changes, however, mainly for consistency, readability, or grammar. Other changes involved changing the schema to have correct dual attribute primary keys for the composite relationship tables and we changed the relationship between loadouts and troopers from M:M to M:1 since it makes more sense that way.

Later on we received some excellent suggestions from Rachel Orrell in step 2 which we implemented. We updated the schema to simply use arrows instead of crow's feet notation since crow's feet notation is already used in the ER diagram. We specified that loadouts are for weaponry for the stormtroopers. We fixed the trooper loadout relationship to consistently be 1:M and now loadouts can be shared among troopers. And finally we reworded and cut down parts of our project summary for better readability.

In step 3 we received some more good feedback from students that we were happy to implement. We fixed column names on a couple pages to be more user friendly, added a query to remove a trooper for when they die (RIP), made the tables larger to be nicer to look at, and fixed a typo on the garrison page.

Now at step 4 is where we started to make some more significant changes to our project. We were under the impression that our tables were to contain information from the database just for the specific entity in question. This made things quite unintelligible due to the usage of foreign key identifiers which were all integers, which lead to some tables just being integers in every column. It was pointed out that we should be using joins to have more intelligible user friendly information so we gladly implemented that in troopers by including the garrison name that the trooper belongs to as well as the weaponry in their loadout. We also added a footnote that specified which attributes were from a separate entity.

We also received some feedback from the TAs and added more specifics to our project description. Originally it was a bit more like a sales pitch as opposed to a summary so we included more specific details on how the database functioned as well as added more numerical data to the description.

When it came time to actually connect front end with back end many problems that we didn't initially realize started becoming obvious. One of the most prominent changes was our front end initially did not account for dropdowns but due to feedback in the piazza reviews and from TAs we decided to do it for input fields where there would be a predefined quantity of choices. For example, there were 11 types of ships so it made sense to use a dropdown for ship types during the ships INSERT. However, it would not be user friendly to implement a similar dropdown when adding the M:M pairs between ships and troopers (or ships and droids) since this was dependent on the number of rows. If the current state of the database had 1000 rows, this would mean a user would be scrolling an irritatingly long time to find their desired IDs. It would have been better to implement something like autocomplete, but it seemed out of scope of the project. In the end we added extra DMQs for the predefined dropdowns but the latter omitted dropdowns.

Another change made during implementation was we started using names for the foreign keys even though we did not bother with them in the initial formation of the DDQs. This change was to allow easier validation since we could just search the query result for the named constraint and then assign actions based on if they were found or not. Other subtleties were brought to our attention during the piazza reviews as well. During testing we had forgotten to check that the troopers garrison could be set to NULL so when a reviewer tried setting that attribute to NULL the entire row would disappear due to the inner join mentioned earlier. Thankfully the fix was simple and all we needed was to change the relevant DMQs to use LEFT JOINs instead.

Finally the last hurdle was validation due to the lack of dropdowns for certain fields. We needed validation in 3 parts: before attempting the query (don't want SQL injection), if the query failed (e.g.searching for a FK that doesn't exist or adding a duplicate primary key), and even after the query succeeded (a query succeeding with 0 modified row). Due to the similar nature of the queries across all pages, we thought it best to implement a single validation class whenever dealing with redirects to cut out some duplicate code.

Kyle Bell
Jeremy Tsang

Project Outline

Project Title: Empire Employees
Team: Rakghoul Serum
URL: http://flip3.engr.oregonstate.edu:6600/

Empire Employees is a database driven web application designed to manage stormtroopers, garrisons, ships, and droids under the Galactic Empire. It does this by providing a listing of each of the above entities as well as allowing users to add/modify/delete entities. For modifying these entities, we allow troopers to be moved around to various garrisons (or no garrisons at all if they are in transit). Their equipment, modeled by the Loadouts entity, can also be updated as well. We can move troopers/droids from ship to ship as well and in this case these relationships are modeled by a M:M relationship. Stormtroopers can also be deleted from the database if they have fulfilled their purpose and given their lives for the Empire.

The glorious Galactic Empire has 1.5 million planets under its control which includes 69 million different colonies and states. Without proper organization and resolve the galaxy will turn to chaos, or worse, become under the control of rebel scum. Empire Employees is capable of managing all 1 billion Stormtroopers, 25,000 Star Destroyers and other smaller vehicles, as well as droids that the Empire employs. There are 11 vehicles types and 20 droid types currently supported. Each stormtrooper is equipped with a loadout that contains 1 out of 7 blasters and 1 out of 2 detonator grenades. To deploy a specific number of troops to ensure effective defense of each garrison spread such large cosmic distances it is critical to always have the correct number of supplies and personnel available. Empire Employees does this by allowing the users current troop, ship, and droid counts for each garrison and providing sufficient warnings when said counts drop below an acceptable level needed for adequate defense. For some ships, such as tie fighters, there may only be 1 trooper on board whereas for larger ships, such as star destroyers, may have as much as 9,700 Stormtroopers. With our simple to use online interface, we will meet all your Stormtrooper relocation management needs!

Database Outline

Entity Tables:

troopers - an individual soldier of the Galactic Empire that upholds peace and prosperity for all. They have a garrison and load out assignments that are related to the garrisons and loadouts entities respectively.

- id: int, auto_increment, unique, not NULL, PK
- garrison: int, FK
- loadout: int, not NULL, FK

Kyle Bell
Jeremy Tsang

- Relationship: a 1:M relationship between garrisons and troopers, a M:M relationship between troopers and ships (implemented with the ships_troopers composite entity), and a 1:M relationship between loadouts and troopers.

ships - The Galactic Empire has ships in order to transport troopers and droids throughout the galax. Each ship has a ship type and is related to troopers and droids.

- id: int, auto_increment, unique, not NULL, PK
- type: varchar(255), not NULL
- Relationship: a M:M relationship between ships and troopers, a M:M relationship between ships and droids.

droids - Robust, top of the line droids help repair ships for our glorious fleet. They have a droid type, and have a relationship with ships.

- id: int, auto_increment, unique, not NULL, PK
- type: varchar(255), not NULL
- Relationship: a M:M relationship between droids and ships implemented with the ships_droids composite entity.

loadouts - Each unique weapon loadout contains a proprietary blaster as well as an option for a detonator in order for each trooper to get the job done right their own way! A loadout has a relationship with a trooper.

- id: int, auto_increment, unique, not NULL, PK
- blaster: varchar(255), not NULL
- detonator: bool, not NULL
- Relationship: a 1:M relationship between loadouts and trooper

garrisons - Our Empire keeps the galaxy safe with garrisons throughout the galaxy. Each of them has a name and a maximum capacity

- id: int, auto_increment, unique, not NULL, PK
- Name: varchar(255), not NULL UNIQUE
- capacity: int, not NULL
- Relationship: a 1:M relationship is implemented between troops and garrisons as a FK inside of troops.

Relationship Tables:

ships_troopers - Relationship between ships and troopers.

- ship: int, not NULL, PK, FK
- trooper: int, not NULL, PK, FK
- Relationship: facilitates M:M relationship between ships and troopers.

ships_droids - Relationship between ships and droids.

Kyle Bell
Jeremy Tsang

- ship: int, not NULL, PK, FK
- droid: int, not NULL, PK, FK
- Relationship: facilitates M:M relationship between ships and droids.

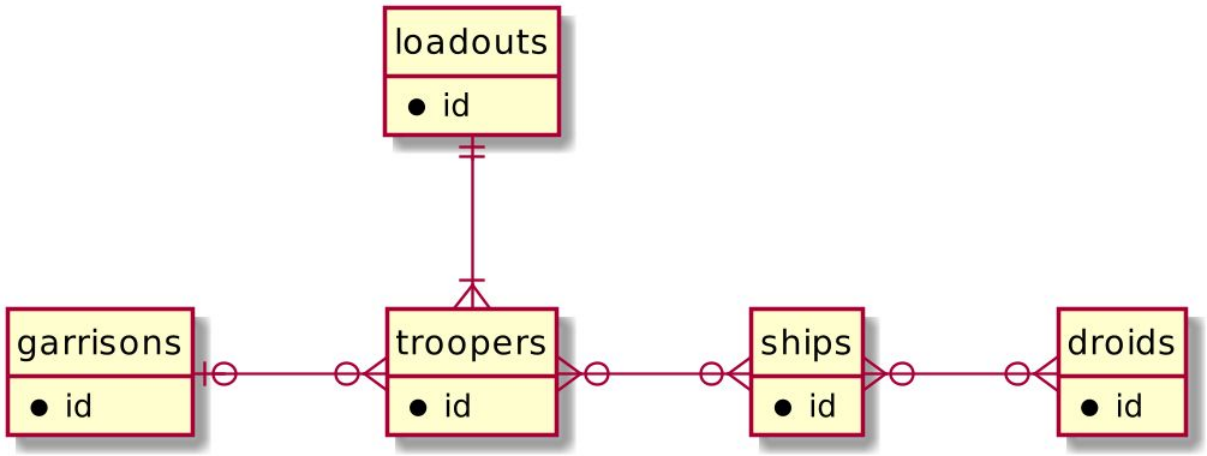
Team Assignments:

We will be implementing all of the entities described above:

- troopers
- loadouts
- garrisons
- ships
- droids
- ships_droids
- ships_troopers

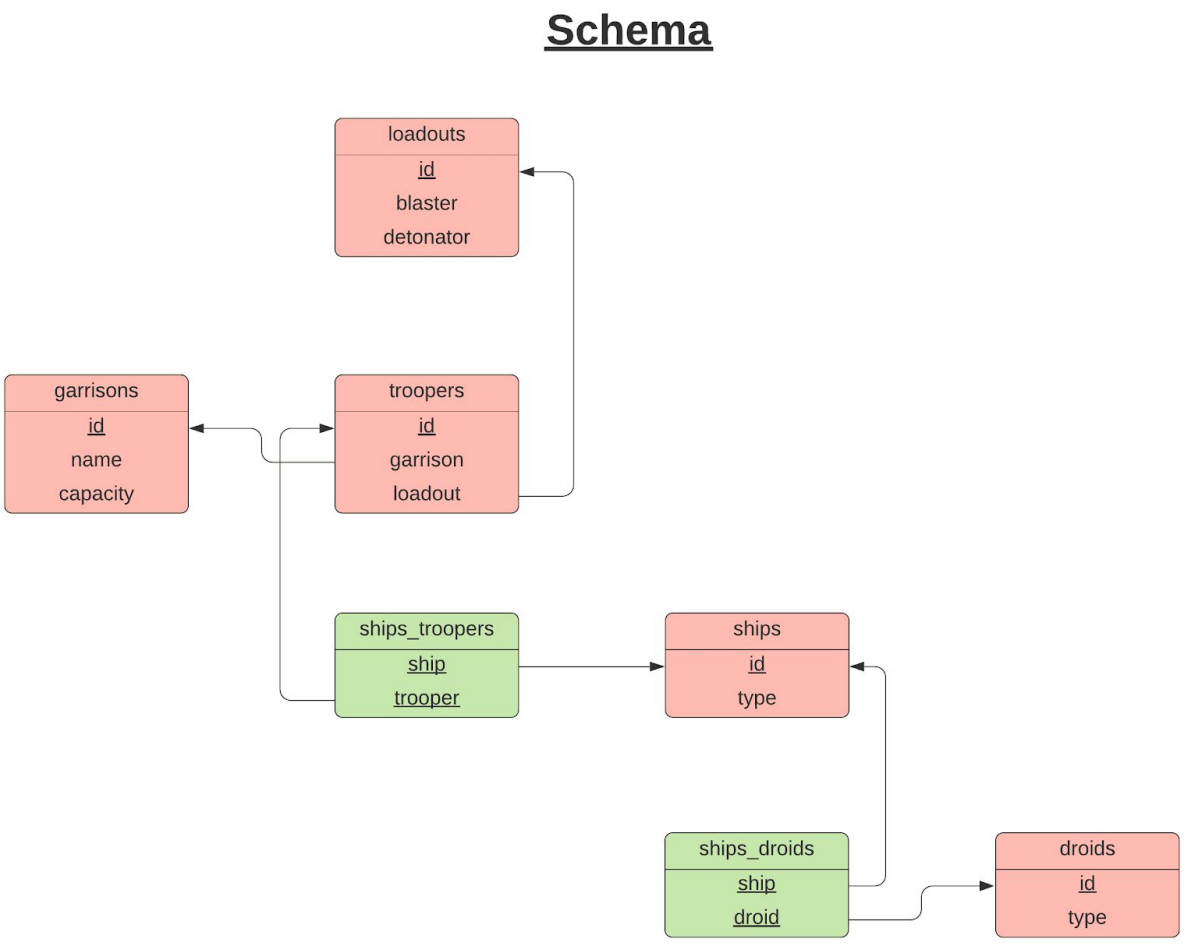
- Kyle will be responsible for the code and webpages for the following entities:
  - ships_droids (M:M relationship)
  - droids
  - garrisons
  - ships
- Jeremy will be responsible for code and webpages for the following entities:
  - ships_troopers (M:M relationship)
  - troopers
  - loadouts

Entity-Relationship Diagram



Kyle Bell
Jeremy Tsang

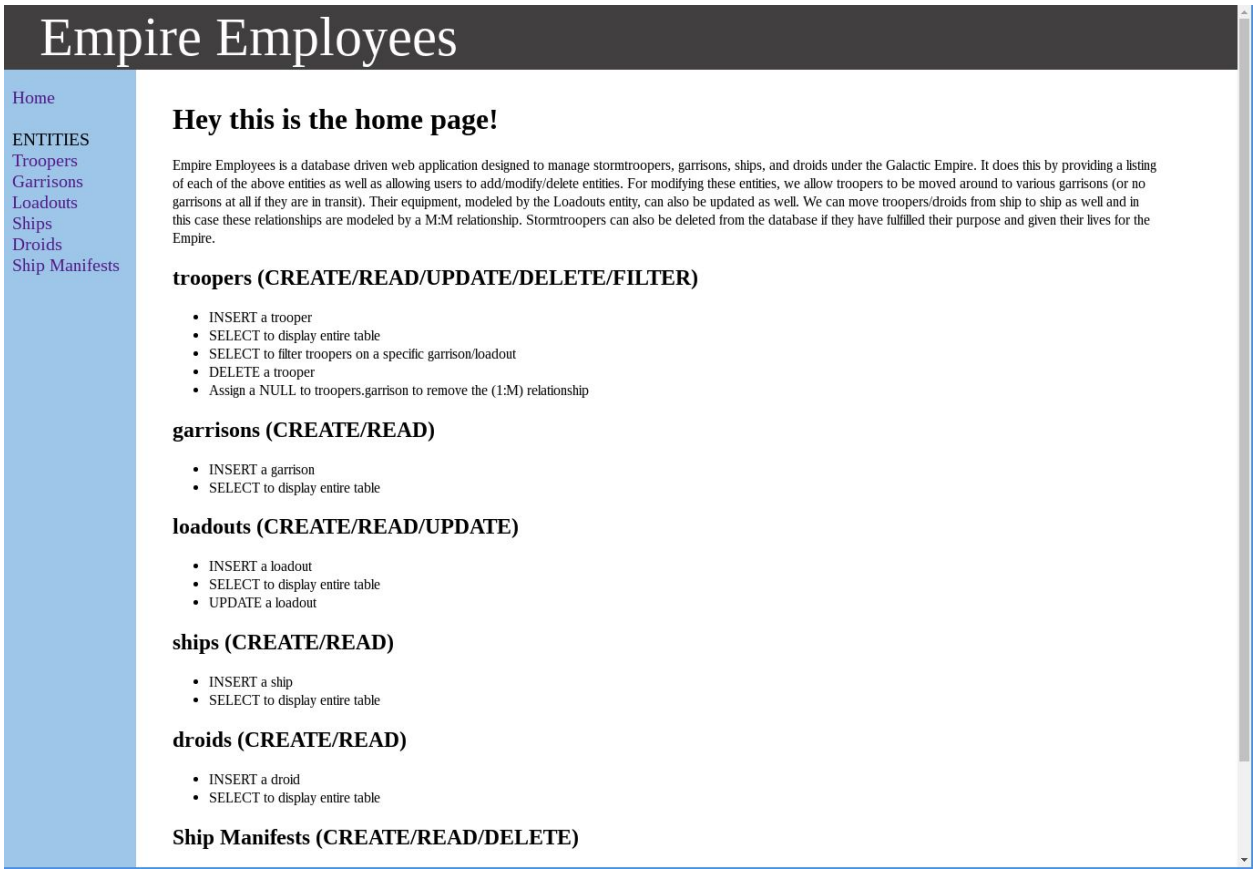
Schema



Kyle Bell
Jeremy Tsang

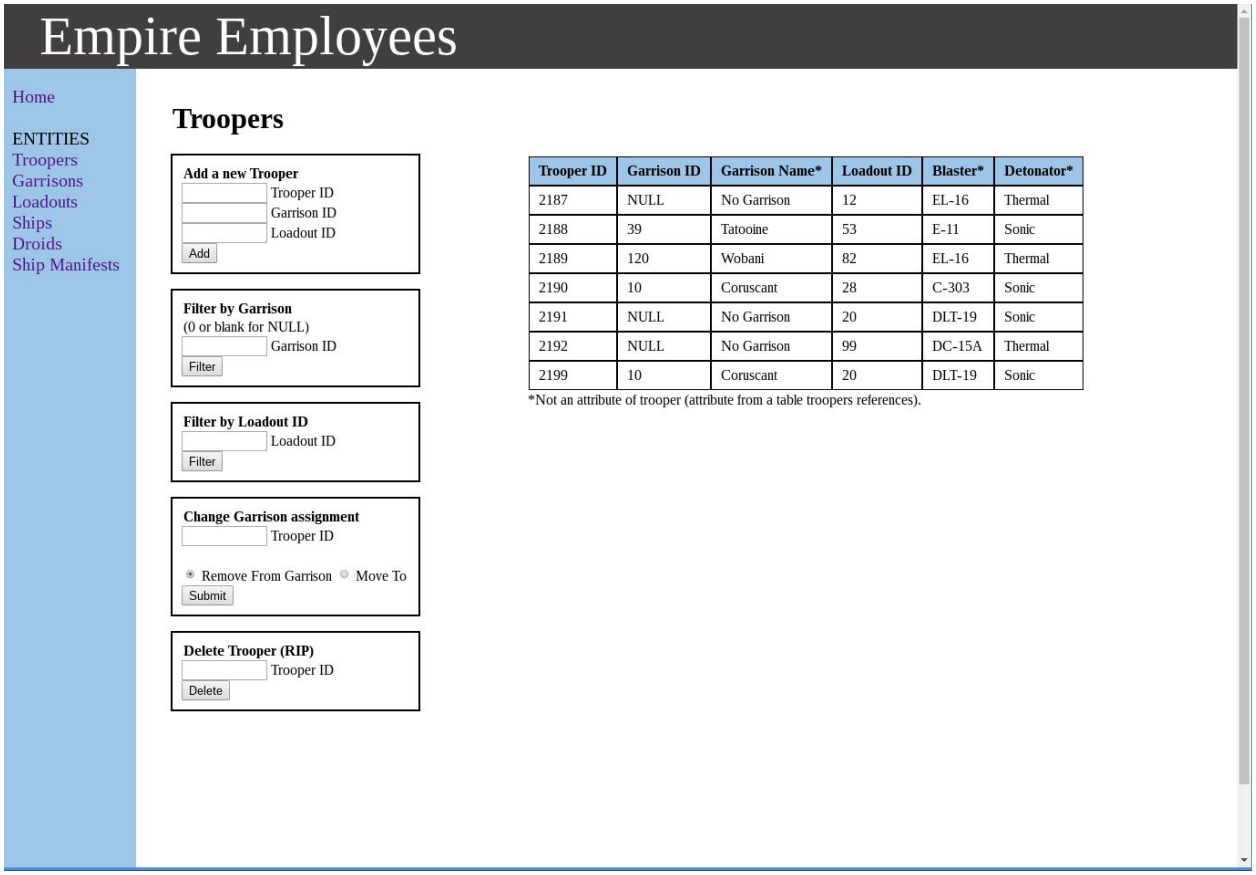
Screen Captures

Home page



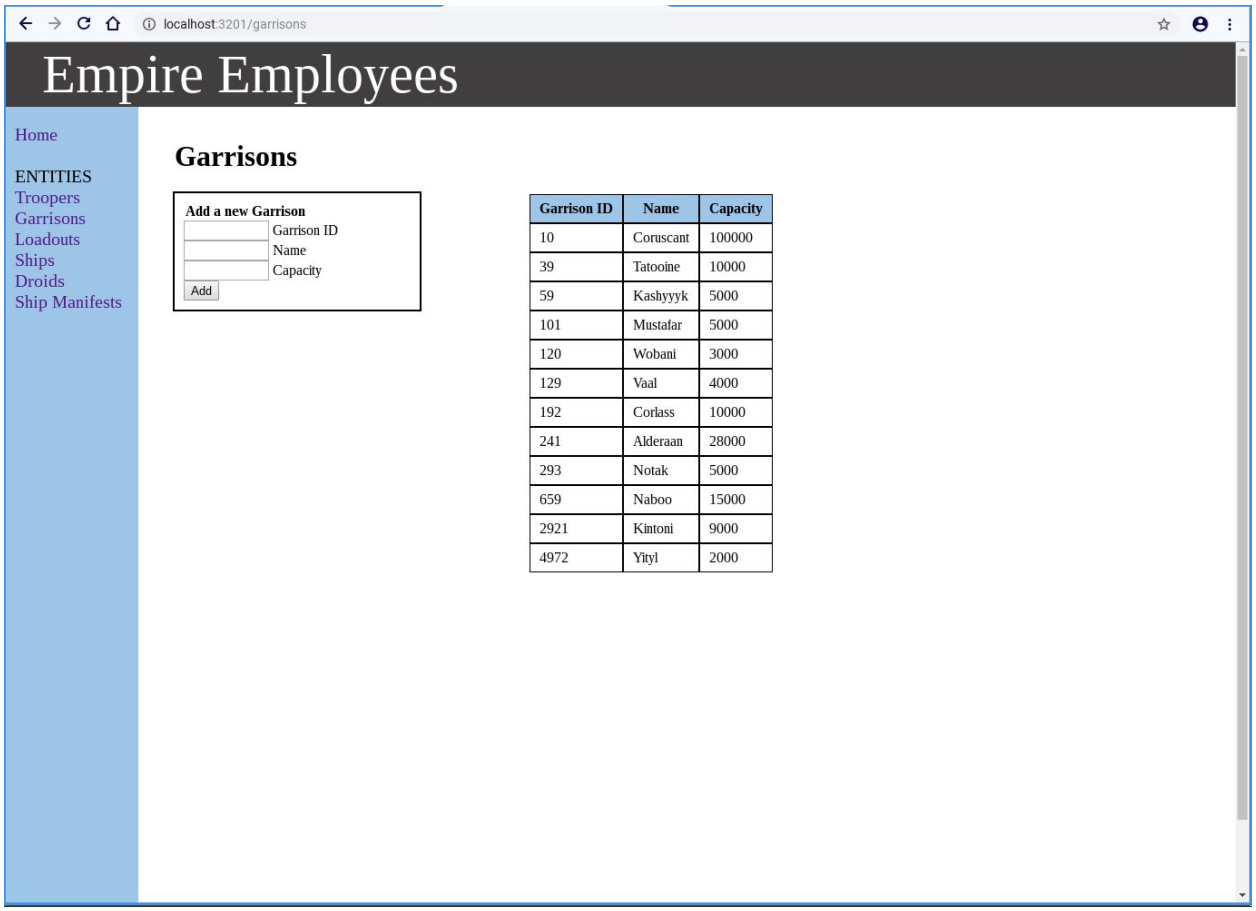
Kyle Bell
Jeremy Tsang

CREATE/READ/UPDATE/DELETE/FILTER troopers Page



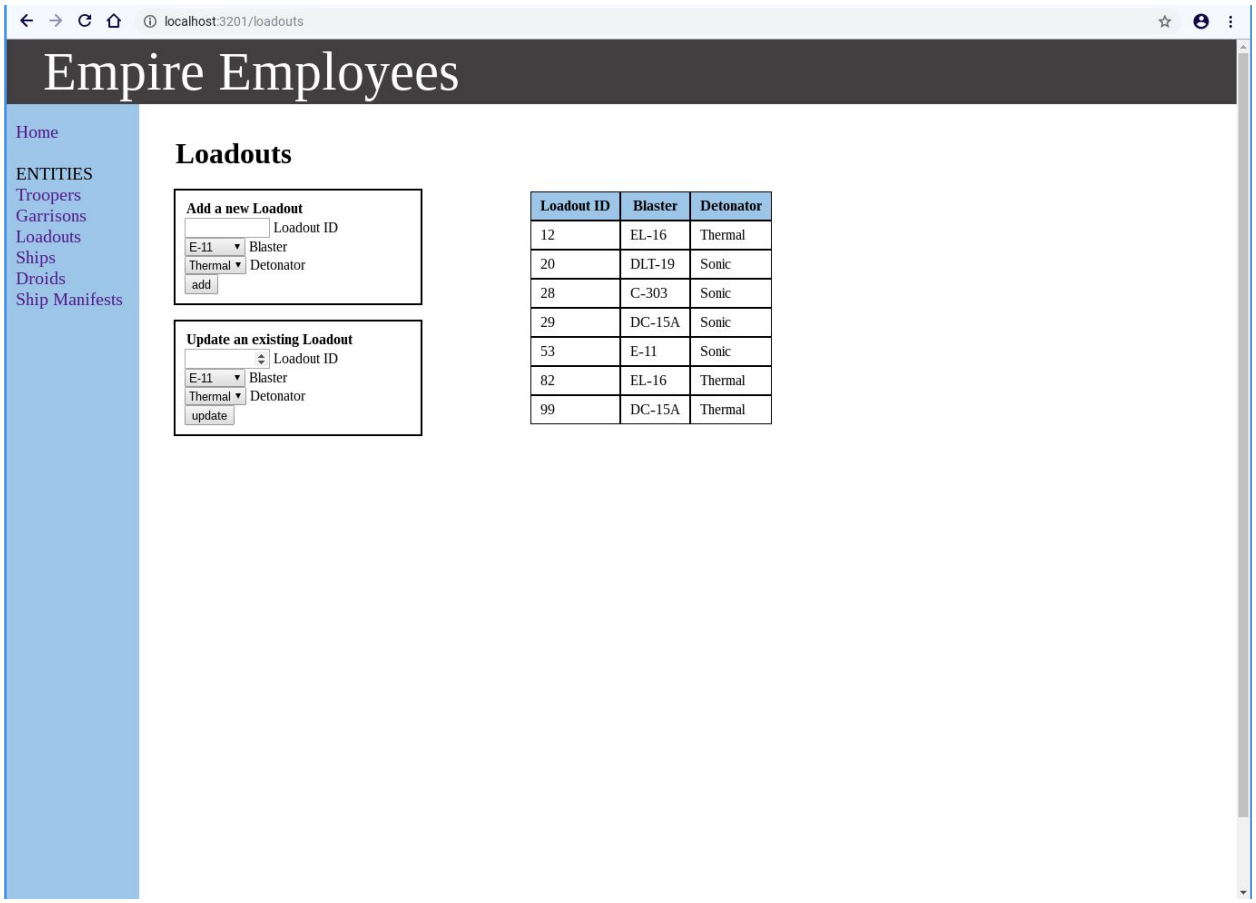
Kyle Bell
Jeremy Tsang

CREATE/READ garrisons page



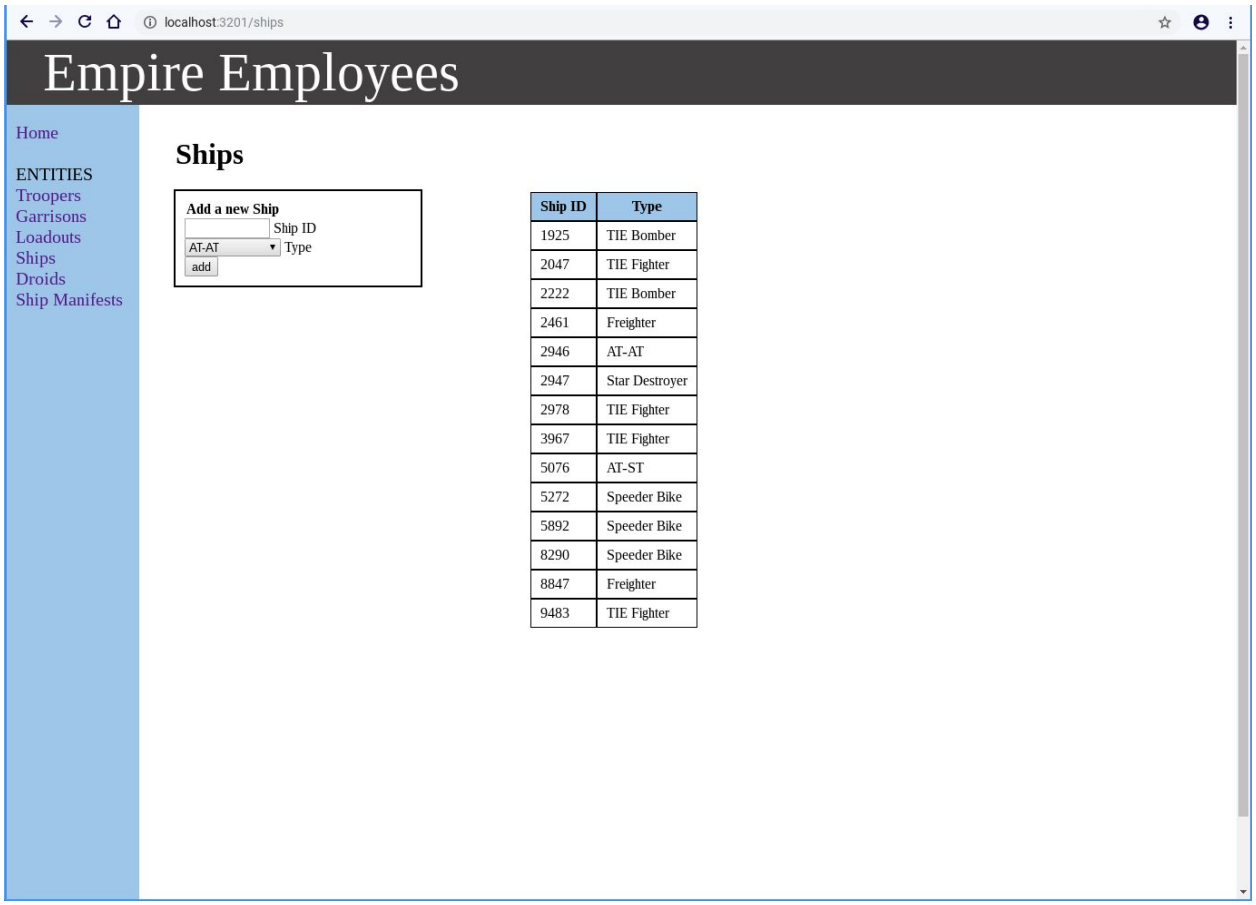
Kyle Bell
Jeremy Tsang

CREATE/READ/UPDATE loadouts page



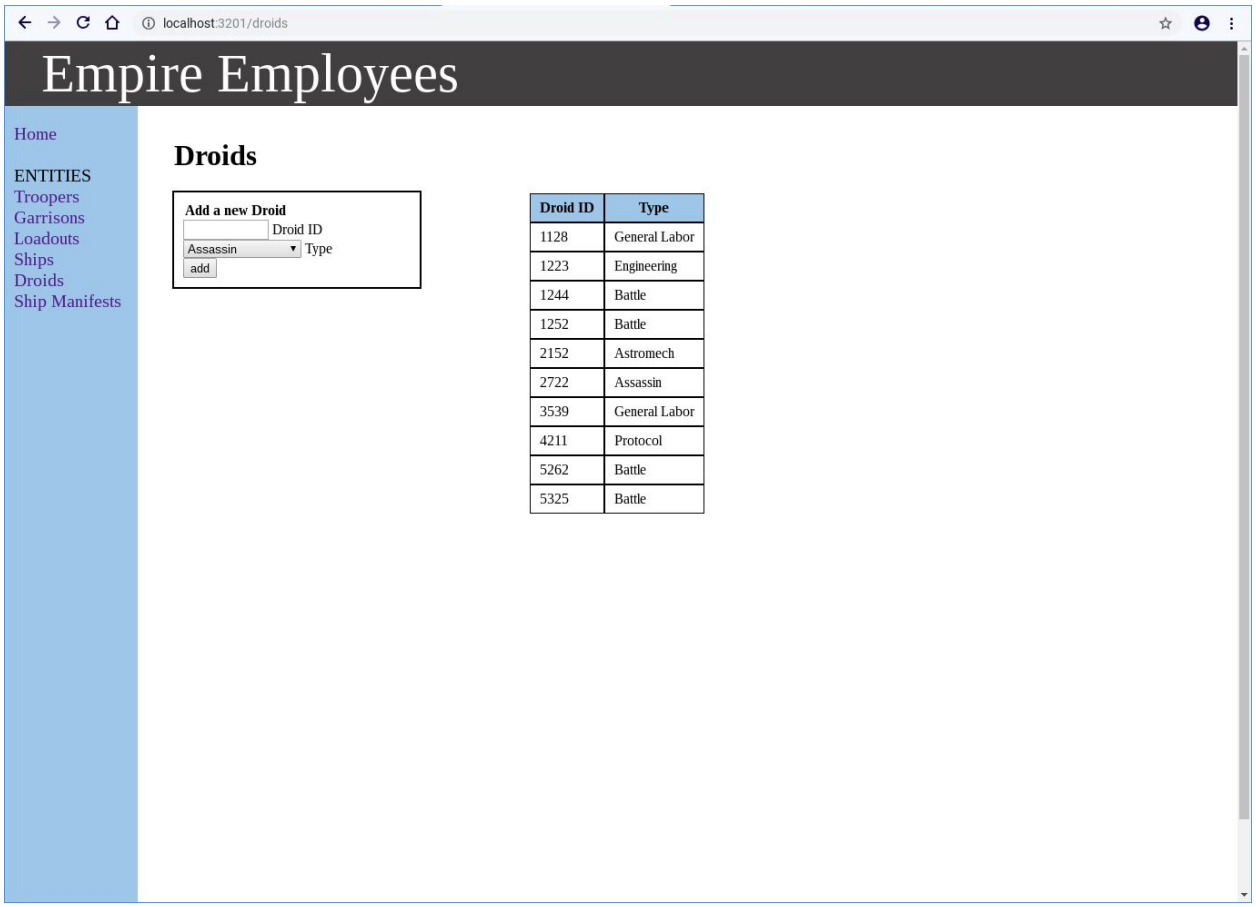
Kyle Bell
Jeremy Tsang

CREATE/READ ships page



Kyle Bell
Jeremy Tsang

CREATE/READ droids page



Kyle Bell
Jeremy Tsang

CREATE/READ/DELETE manifests page