CSCI 370 Project

Final Report

April 14, 2019

Matthew Hird
545366650

Introduction **3**

Application Requirements **5**

Database Design **7**

Database Schema 9

Sample Data **11**

Testing **12**

References **15**

Introduction

The purpose of this application is to act as an information resource for the "Star Wars: X-Wing" miniatures tabletop game. In the game, players simulate aerial dogfights between spacecraft by moving miniature model ships on a table top and rolling dice to decide the result of attacks.

Before the game starts, each player selects the ships and pilots they plan to take into combat. Each player selects one of three **factions** to play as, select the **ships** they plan to use from a pool based on their faction, and select which **pilot** will fly each craft. The list of pilots to choose from is based on the both the ship type and faction. Each ship/pilot combination has a point cost associated with it. The players decide on a point limit, which is the total number of points worth of ships/pilots each player can bring to the battle. Additionally, each pilot and ship have additional rules and stat values assigned to them to used throughout the game.

Much of this information is shown on specialized **pilot cards** that come with each purchased ship model (Figures 1, 2 and 3). One card exists for each (faction, ship, pilot) set. The parts of the card involving the pilot are labelled in red in Figure 1.

Figure 1: *Pilot components of pilot card*

The **pilot's name** is listed at the top. Names without a black dot next to them are generic (such as "Red Squadron Pilot"), meaning one player may use more than one pilot with that name within one game. A name with a black dot means it is a **unique name**, meaning a player can only use one pilot with that name in a single game. This is important as sometimes the same pilot name exists on cards for multiple ship types, on multiple factions, or even on a different pilot card

with the same faction and ship type. The **elite** (medal) icon indicates the pilot can take an elite upgrade ability. The **pilot skill** and **point cost** are integer values showing the quality and cost of the pilot.

Figure 2 shows the aspects of the pilot cards tied to the ship. These values are the same for every pilot within a faction flying that ship type. At the top are the **ship name** and **faction** icon. The **ship stats** on the left are numeric values that dictate the ship's combat effectiveness. The **action bar** contains symbols indicating what types of **actions** that ship can take during a turn. The **upgrade bar** at the bottom indicates what **upgrade** types each ship can be assigned. The **elite** status is the only upgrade not based on the ship aspect of the card.

Figure 2: *Ship components of pilot card*

Figure 3: Pilot card for *rear half of a huge ship*

Ships also come in different **sizes** -- small, large and huge. Small and large ships have mostly the same rule, with a few small exceptions, but those differences are in how some rules are applied differently, not differences in the type of information tied to the ships. **Huge ships** on the other hand can be very different. Some huge ships have separate pilot cards for the front half and rear half. The rear half (shown in Figure 3) has an **energy value** instead of an attack value, while the front half has a different attack range than smaller ships.

Figure 4: *Ship maneuver dial*

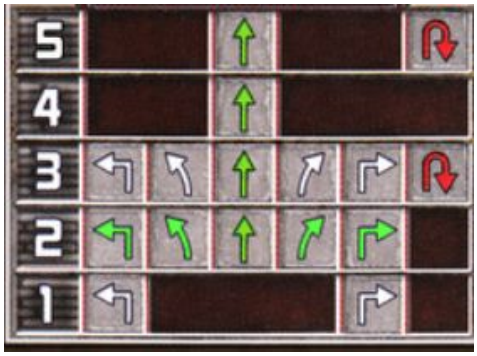


Figure 5: *Ship maneuver table*

Each ship also has a set of **maneuvers** the the player has to select from each turn. Each maneuver consists of a **maneuver type** symbol, a **difficulty** colour, and a **speed** value. In the actual game, these maneuvers are shown on round cardboard dial (shown in Figure 4), and a maneuver is selected by turning an indicator component (not shown) to point at the correct maneuver. To maneuver options easier to read, official reference material shows the maneuver dial information in a table (shown in Figure 5).

Application Requirements

There are 3 core functionalities this application needs to perform: displaying the individual ship and pilot information described in the introduction, searching for ships or pilots based on various search fields, and inserting new ship and pilot information into the system.

In this context, a ship refers to a unique ship entity, meaning a **ship name** plus any other characteristics that are different between 2 or more ships with the same name. The **ship variant** is an arbitrary addendum to the name to make each ship's name/variant pair unique. Similarly, a pilot refers to a unique pilot card, meaning the set of a **pilot name**, **ship name**, and **faction** plus any other characteristics that are different between pilot cards with the same pilot name, ship name and faction, and the **pilot variant** is an arbitrary addendum to the name to make the pilot's name/variant pair unique.

Each ship has a *ship page* that displays information for that ship. This includes the basic identifiers and statistics (**name**, **variant**, **faction**, **size**, **agility**, **hull**, **shield**), the set of **actions** the ship can perform, the **upgrades** the ship has access to, the set of **maneuvers** available to that ship, and the **list of pilots** that can fly that ship. Also, if the ship is huge and has an **energy value** instead of an attack value, the energy value is displayed instead.

Each pilot has a *pilot page* that displays the pilot's **name**, **variant**, pilot's **unique name**, **ship name**, **faction**, **pilot skill**, **elite status** and **point cost** for that pilot.

There is a dedicated search page with various search fields and filters to choose from. First, the user can select whether the results will be for *Ships* or *Pilots*. They can then can then filter by "pilot/ship name contains...", **pilot skill** (disabled if *ship* results selected), **faction**, **size**, **attack**, **agility**, **hull**, or **shield**. Fields with numeric values can be compared with ==, >=, <= or !=. **Size** can be filtered by Small, Large, Huge, Not Huge, or Any.

There is also a quick search function that is available on the top navigation bar of every page. This simply lets the user select *Ships* or *Pilots* and enter a search term into the text field. For all searches, if no other fields are filled out, searching for *Ships* returns all ships in the system and searching for *Pilots* returns all pilots in the system.

A *Ship* search returns a list of (**ship name**+**variant**, **faction**) pairs, while a *Pilot* search returns a list of (**pilot name**+**variant**, **ship name**+**variant**, **faction**) triples. Each **ship name** links to that ship's page, each **pilot name** link's to that pilot's page, and each **faction** name does a search for all ships of that faction.

On each ship page, each **pilot name** in the *pilot list* is linked to that pilot's page, and clicking the ship's **faction** does a search for all ships for that faction. On each pilot page, the *ship name* is linked to that ship's page, clicking the **faction** does a search for all ships from that faction, and clicking the **unique name** searches for all pilots with the same unique name.

The final core functionality is the ability to add new ships and pilots to the system.

Ships have the input fields:

text fields=>{**ship name**, **ship variant** (optional)};
enum dropdowns=>{**faction**, **size**};
numeric values (or N/A)=>{**attack**, **energy** (if size is huge), **agility**, **hull**, **shield**}.

Pilots have the input fields:

text fields=>{**pilot name**, **pilot variant** (optional), **unique name** (existing or new),
ship name, **variant name** (ship name/variant pair must already exist in system)};
boolean dropdown=>{**elite**};
numeric values (or N/A)=>{**pilot skill**, **point cost**};

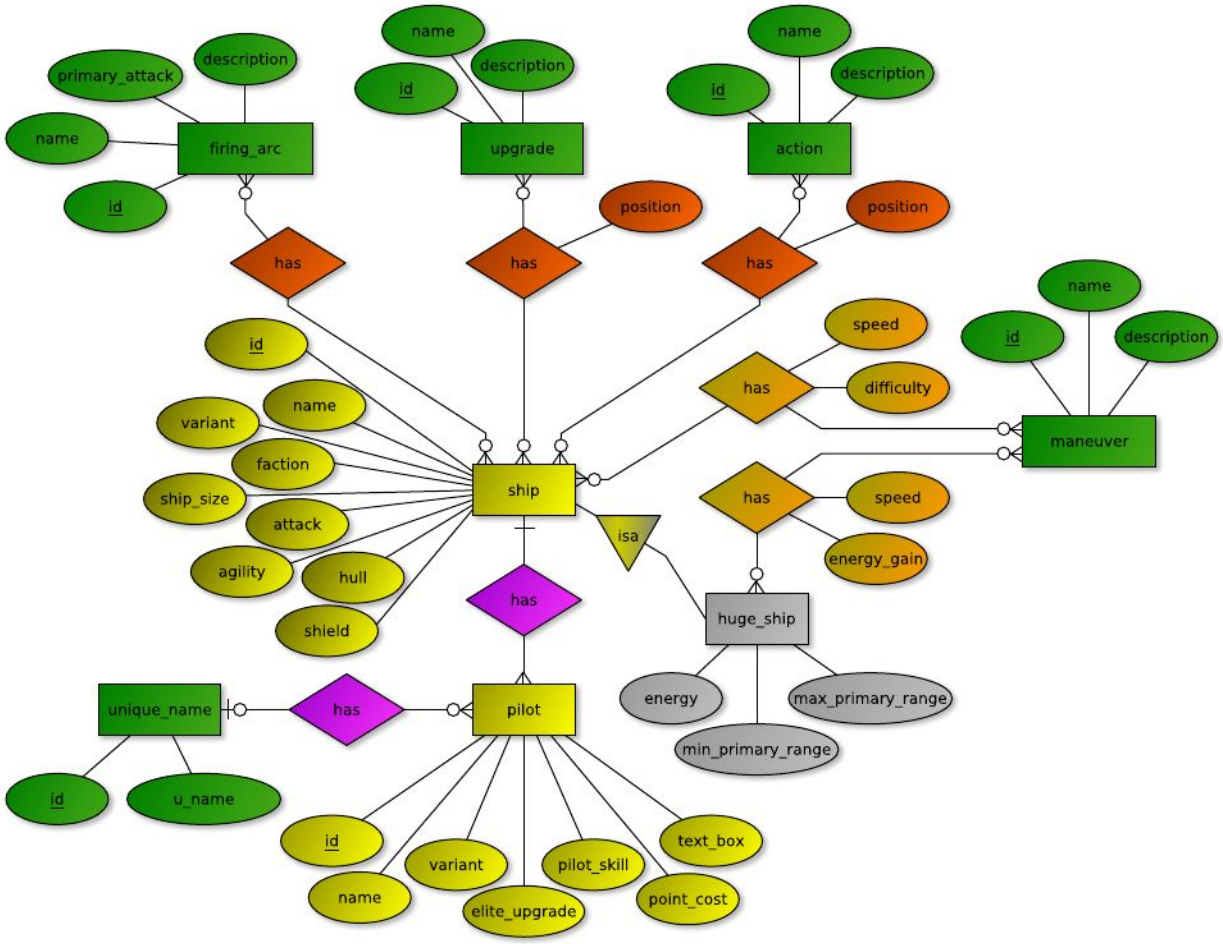
Confirmation messages are sent if data is correctly inserted. If invalid data is entered or the product addition fails for any other reason, any pending database transaction is rolled back and an abort message is sent to the user.

In order to limit scope due to time constraints, there is no ability to search on or insert new **actions**/**action bars**, **upgrades**/**upgrade bars** or **maneuvers**/**maneuver tables**. Also, there is no way to delete data from within the client application.

Database Design

Figure 6 shows my ER diagram for this application domain.

Figure 6: *ER diagram for X-Wing application*



Each entity in the table has been given an arbitrary integer primary key field called *id*. This was to make entering foreign key data easier (for me), and to make comparisons more quickly and to reduce data storage space (for the database).

Even though they lack fields outside of their name, id and unused description field, I believe the existence of the entities *firing_arc*, *upgrade*, *action*, *maneuver* and *unique_name* is justifiable. The first 4 are all in a many to many relationships, and their names are meaningful in both the system and the game (they are referenced by rules and represented by symbols and images). As for *unique_name*, it is the 1 in a 1 to many relationship. Additionally, all of these entities will be expanded upon in future iterations of this project. All of the description and text box fields will be filled with rules text, and each item in those entities will be given it's own webpage in the application. Also, there are *upgrade cards* with unique names, and some of the names are shared with pilot names.

The *ship* entity has attributes for each of its basic stats and traits. The *variant* attribute represents the arbitrary discriminator I've given to ships that share names in order to make the name/variant pair unique. The way I see it, the *id* value is the unique identifier the machine easily recognizes, and the name/variant pair is the unique identifier that humans easily recognize. Its upgrade bars and action bars are represented in a many to many relationship with the *upgrade* and *action* entity. The *position* attributes on those relations refers to the left-to-right 0-based index of those symbols on the pilot cards. The list of firing arcs each ship has is represented in the many to many relationship with the *firing_arc* entity. The maneuver dial/table for a ship is represented by the many to many relationship with the *maneuver* entity. Each (maneuver symbol, speed, difficulty colour) triple on each ship's dial is represented by the *speed* and *difficulty* attributes with the maneuver foreign key.

The *huge_ship* entity is a specialization of *ship*, as each member will need all of the attributes and relations from *ship*. If a huge ship has an attack value, it will have a special *min_attack_range* and *max_attack_range*, and *energy* will be NULL. If it doesn't have an attack value, it will have an *energy* value, and *min_attack_range* and *max_attack_range* will be NULL. The *huge_ship* entity has its own relationship with the *maneuver* entity, except maneuver dials for huge ships have *energy_gain* instead of *difficulty*.

The *pilot* entity represents unique pilot cards. Its attributes represent each of the bits of data shown in the pilot sections of the pilot cards (described in the "Introduction").

Database Schema

```
CREATE TABLE firing_arc (
  id             TINYINT(2)              NOT NULL,
  name           VARCHAR(50)             NOT NULL,
  primary_attack BOOL                    NOT NULL,
  description    VARCHAR(1000) DEFAULT '' NOT NULL,
  PRIMARY KEY (id)
);

CREATE TABLE action (
  id          TINYINT(2)    NOT NULL,
  name        VARCHAR(50)   NOT NULL,
  description VARCHAR(1000) NOT NULL,
  PRIMARY KEY (id)
);

CREATE TABLE maneuver (
  id          TINYINT(2)              NOT NULL,
  name        VARCHAR(50)             NOT NULL,
  description VARCHAR(1000) DEFAULT '' NOT NULL,
  PRIMARY KEY (id)
);

CREATE TABLE upgrade (
  id          TINYINT(2)              NOT NULL,
  name        VARCHAR(50)             NOT NULL,
  description VARCHAR(1000) DEFAULT '' NOT NULL,
  PRIMARY KEY (id)
);

CREATE TABLE unique_name (
  id   SMALLINT(3) NOT NULL AUTO_INCREMENT,
  name VARCHAR(50) NOT NULL UNIQUE,
  PRIMARY KEY (id)
);

CREATE TABLE ship (
  id      TINYINT(2)                         NOT NULL,
  name    VARCHAR(50)                        NOT NULL,
  variant VARCHAR(20) DEFAULT ''             NOT NULL,
  faction ENUM ('Rebel', 'Imperial', 'Scum') NOT NULL,
  `size`  ENUM ('Small', 'Large', 'Huge')    NOT NULL,
  attack  TINYINT(1),
  agility TINYINT(1),
  hull    TINYINT(1),
  shield  TINYINT(1),
  PRIMARY KEY (id),
  CONSTRAINT uq_ship UNIQUE (name, variant)
);
```

```
CREATE TABLE huge_ship (
  ship_id           TINYINT(2) NOT NULL,
  energy            TINYINT(1),
  min_primary_range TINYINT(1),
  max_primary_range TINYINT(1),
  PRIMARY KEY (ship_id),
  FOREIGN KEY FK_huge_ship_01 (ship_id) REFERENCES ship (id)
);

CREATE TABLE pilot (
  id          SMALLINT(3)            NOT NULL AUTO_INCREMENT,
  name        VARCHAR(50)            NOT NULL,
  variant     VARCHAR(20) DEFAULT '' NOT NULL,
  u_name_id   SMALLINT(3)            NOT NULL,
  ship_id     TINYINT(2)             NOT NULL,
  pilot_skill TINYINT(1)             NOT NULL,
  elite       BOOL,
  point_cost  TINYINT(2),
  text_box    VARCHAR(100)                   DEFAULT '',
  PRIMARY KEY (id),
  CONSTRAINT uq_pilot UNIQUE (name, variant),
  FOREIGN KEY FK_pilot_01 (u_name_id) REFERENCES unique_name (id),
  FOREIGN KEY FK_pilot_02 (ship_id) REFERENCES ship (id)
);

CREATE TABLE maneuver_dial (
  ship_id     TINYINT(2)           NOT NULL,
  speed       TINYINT(1)           NOT NULL,
  maneuver_id TINYINT(2)           NOT NULL,
  difficulty  ENUM ('G', 'W', 'R') NOT NULL,
  PRIMARY KEY (ship_id, speed, maneuver_id),
  FOREIGN KEY FK_maneuver_dial_01 (maneuver_id) REFERENCES maneuver (id),
  FOREIGN KEY FK_maneuver_dial_02 (ship_id) REFERENCES ship (id)
);

CREATE TABLE huge_maneuver_dial (
  ship_id     TINYINT(2) NOT NULL,
  speed       TINYINT(1) NOT NULL,
  maneuver_id TINYINT(2) NOT NULL,
  energy      TINYINT(1) NOT NULL,
  PRIMARY KEY (ship_id, speed, maneuver_id),
  FOREIGN KEY FK_huge_maneuver_01 (maneuver_id) REFERENCES maneuver (id),
  FOREIGN KEY FK_huge_maneuver_02 (ship_id) REFERENCES huge_ship (ship_id)
);

CREATE TABLE upgrade_bar (
  ship_id    TINYINT(2) NOT NULL,
  upgrade_id TINYINT(2) NOT NULL,
  position   TINYINT(1) NOT NULL,
  PRIMARY KEY (ship_id, upgrade_id, position),
  FOREIGN KEY FK_upgrade_bar_01 (upgrade_id) REFERENCES upgrade (id),
  FOREIGN KEY FK_upgrade_bar_02 (ship_id) REFERENCES ship (id)
);
```

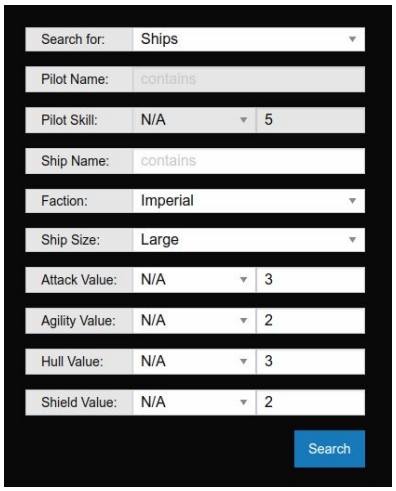
```
CREATE TABLE action_bar (
  ship_id   TINYINT(2) NOT NULL,
  action_id TINYINT(2) NOT NULL,
  position  TINYINT(1) NOT NULL,
  PRIMARY KEY (ship_id, action_id, position),
  FOREIGN KEY FK_action_bar_01 (ship_id) REFERENCES ship (id),
  FOREIGN KEY FK_action_bar_02 (action_id) REFERENCES action (id)
);

CREATE TABLE ship_firing_arc (
  ship_id       TINYINT(2) NOT NULL,
  firing_arc_id TINYINT(2) NOT NULL,
  PRIMARY KEY (ship_id, firing_arc_id),
  FOREIGN KEY FK_ship_firing_arc_01 (ship_id) REFERENCES ship (id),
  FOREIGN KEY FK_ship_firing_arc_02 (firing_arc_id) REFERENCES firing_arc (id)
);
```

Sample Data

My data set includes the relevant data of all of the currently released products for *Star Wars: X-wing (first ed.)*. My data set includes 10 firing_arc entries, 13 action entries, 17 maneuver entries, 17 upgrade entries, 286 unique_name entries, 68 ship entries, 11 huge_ship entries, 301 pilot entries, 882 maneuver_dial entries, 84 huge_maneuver_dial entries, 206 upgrade_bar entries, 187 action_bar entries, and 66 ship_firing_arc entries.

The datasets can be seen in .csv files stored in the following directory:
csci370Project-MatthewHird/xwingProjectDB/csv/

Testing



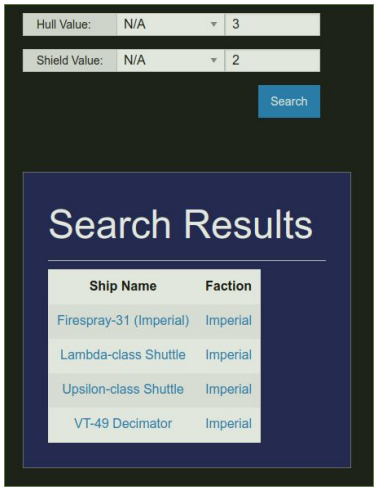
Search:

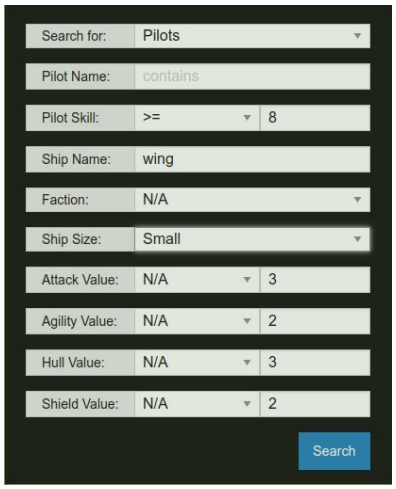
Type: Ships

Faction: Imperial

Size: Large

Result:





Search:

Type: Pilots

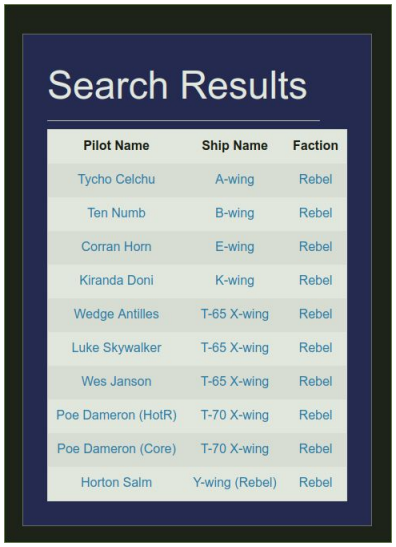
Pilot Skill: >= 8

Ship Name

Contains: wing

Size: Small

Result:



⇩

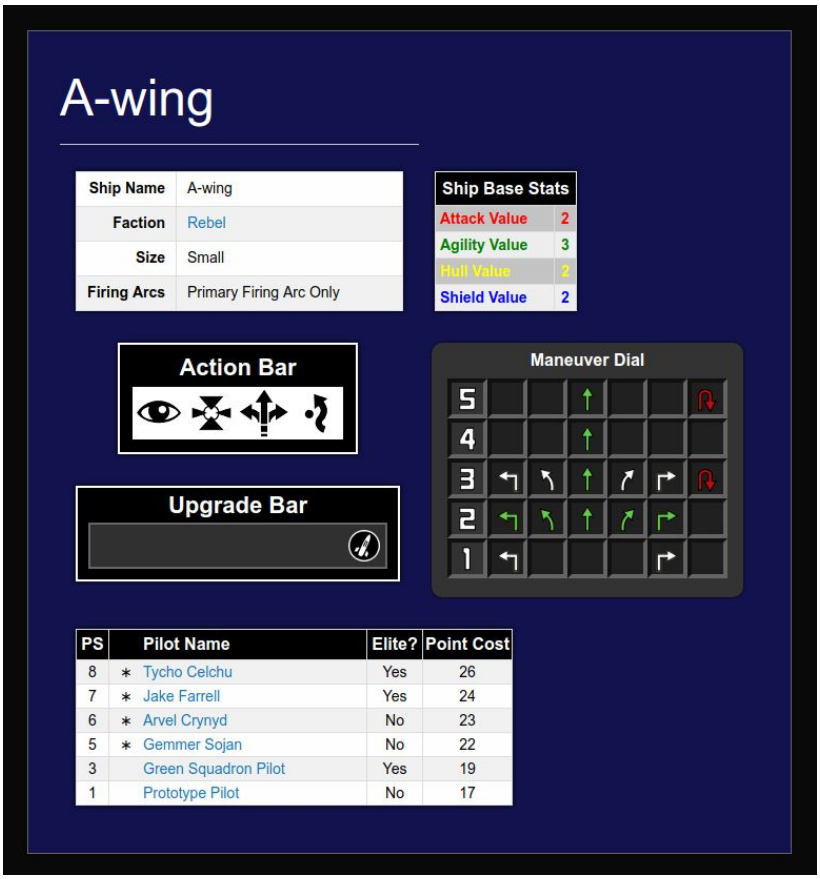
Select: Tycho Celchu

Result:

⇩

Select: A-wing

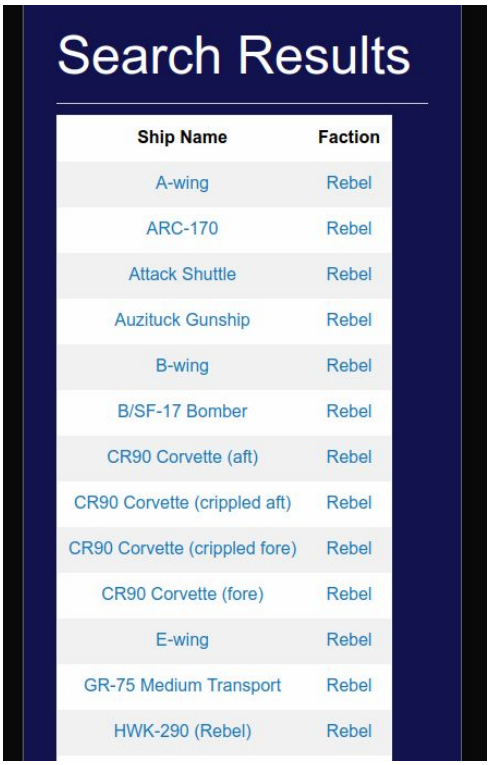
Result:



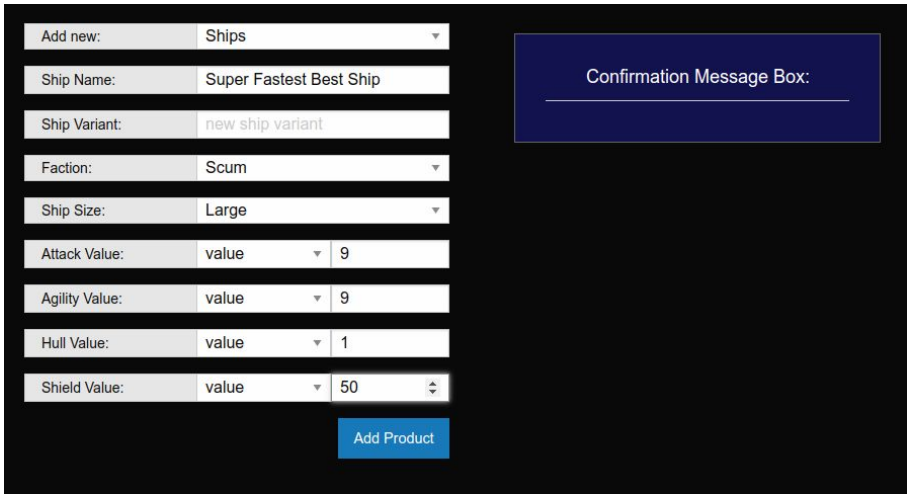
⇩

Select: Rebel

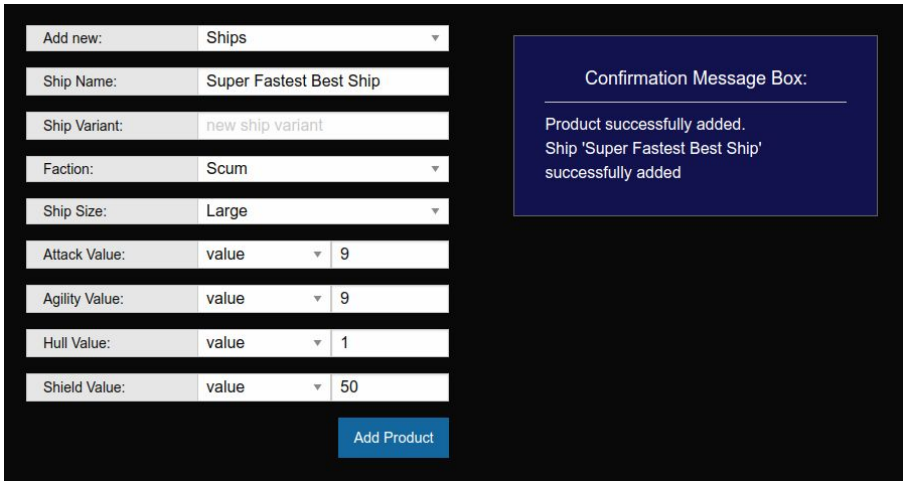
Result:



Add Product:
Type: Ship
Name:
Super Fastest Best Ship
Variant: none
Faction: Scum
Size: Large
Attack: 9
Agility: 9
Hull: 1
Shield: 50

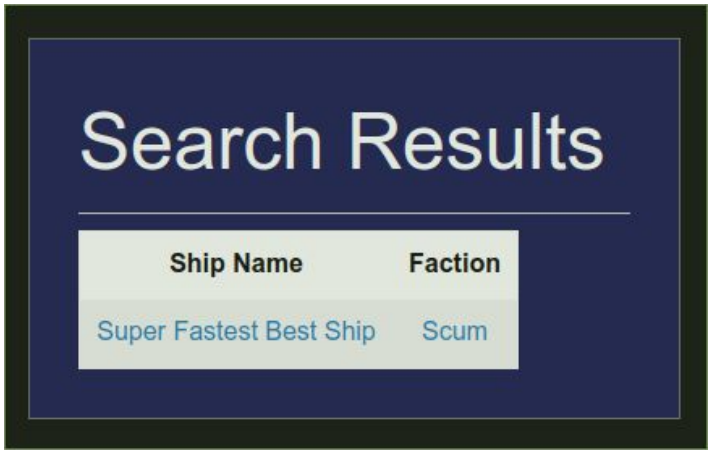


Result:



Search: super fastest

Result:



References

Figure 1. *Captain Nym Scum Card*. Accessed April 12, 2019. https://vignette.wikia.nocookie.net/xwing-miniatures/images/f/f4/Swx65-captain-nym-scum.png/revision/latest?cb=20170617005307

Figure 2. *Captain Nym Rebel Card*. Accessed April 12, 2019. https://vignette.wikia.nocookie.net/xwing-miniatures/images/1/11/Swx65-captain-nym-rebel.png/revision/latest/scale-to-width-down/210?cb=20170617005630

Figure 3. *CR90 Corvette Aft Card*. Accessed April 12, 2019. https://vignette.wikia.nocookie.net/xwing-miniatures/images/9/9a/Cr90-corvette-aft.png/revision/latest?cb=20140425131330

Figure 4. *TIE Defender Maneuver Dial*. Accessed April 12, 2019. https://xwing-miniatures.fandom.com/wiki/Glaive_Squadron_Pilot

Figure 5. *A-Wing Maneuver Table*. Accessed April 12, 2019. https://vignette.wikia.nocookie.net/xwing-miniatures/images/0/06/A-Wing_Move.png/revision/latest?cb=20130926024415