|  |  |
| --- | --- |
| Project Starbase  University of Calgary, CPSC 471, Winter 2019 | Abstract  Project Starbase is a database orientated solution for the problem of managing and administrating a spaceborne transportation company. Our system allows end users to track and inform their spaceship crew members, organize cargo, plan interplanetary flights, and more. In this report, we explain our website in more detail, then go over its planning, implementation and functionality, including a user-friendly manual.  John Ngo, Celina Ma, Matthew Lee – Group 27  30020834, 10151336, |

Introduction:

The commercialization of space is a fast-approaching prospect. We foresee that transportation of commercial clients and their cargo to and from various interplanetary destinations will be a valuable economic niche within this prospect of commercializing space. As such, it becomes conceivable that an organization, such as a corporation, will want to keep track of their clients, employees, cargo, spaceships, space stations, astronomical bodies, and the plans for flights between various locations. Furthermore, clients will want an easy interface with which to book their flights and register their cargo, where these interactions themselves generate and move data, making it reasonable to bundle the client interface together with this problem.

Managing this information, particularly with countless instances of the data outlined above, will be a data-intensive task as large amounts of information about a small set of types of entities will flow within this system – a situation uniquely suited for a database approach. Companies which faced similar problems before include aircraft transportation services, and shipping services. However, due to how young this spaceborne transportation industry is, there are no readily available and dedicated solutions to this problem. As such, we set out to offer a solution in the form of Project Starbase.

Our Project Starbase System is a web application which is meant to provide a smooth interface for a spaceborne transportation company to manage its data. It consists of a web interface and database server tailored to solve the problem outlined above. The web interface manages interactions between the users and the database from a user-centric perspective. It is focused on delivering to each of the three types of end-users exactly the functionality they may desire - from booking flights and registering cargo for the clients, to managing astronomical data and corporate registries for the Ground Control crew. The database server is what stores and handles the wide array of relevant data that the web interface passes to and from it. Further information will be available in the upcoming sections.

The Design of Project Starbase:

Overview: Project Starbase is designed around the three different external users of the system – clients, flight crew, and the ground control crew. Each of the three types of external users are streamed towards their own web interface following a centralized log in screen. These web interface modules for each of the three end clients contain, clustered together, functions which correspond to the duties and responsibilities of each of the three end users. For example, a member of the flight crew will want to know what spacecraft they’re assigned to but should be incapable of reassigning themselves to a new spacecraft – that would be the job of the ground crew.

Transaction Collection: We’ll show our set of transactions possible within this project via its updated HIPO diagram. As can be observed below, the design is such that a centralized log-in hub then divides the users into three separate web interfaces, depending on if they are clients, ground control, or flight crew. After this, we will go into specifics as to what each of the three end users can see, what they can do, and what the overall functionality of the system looks like:

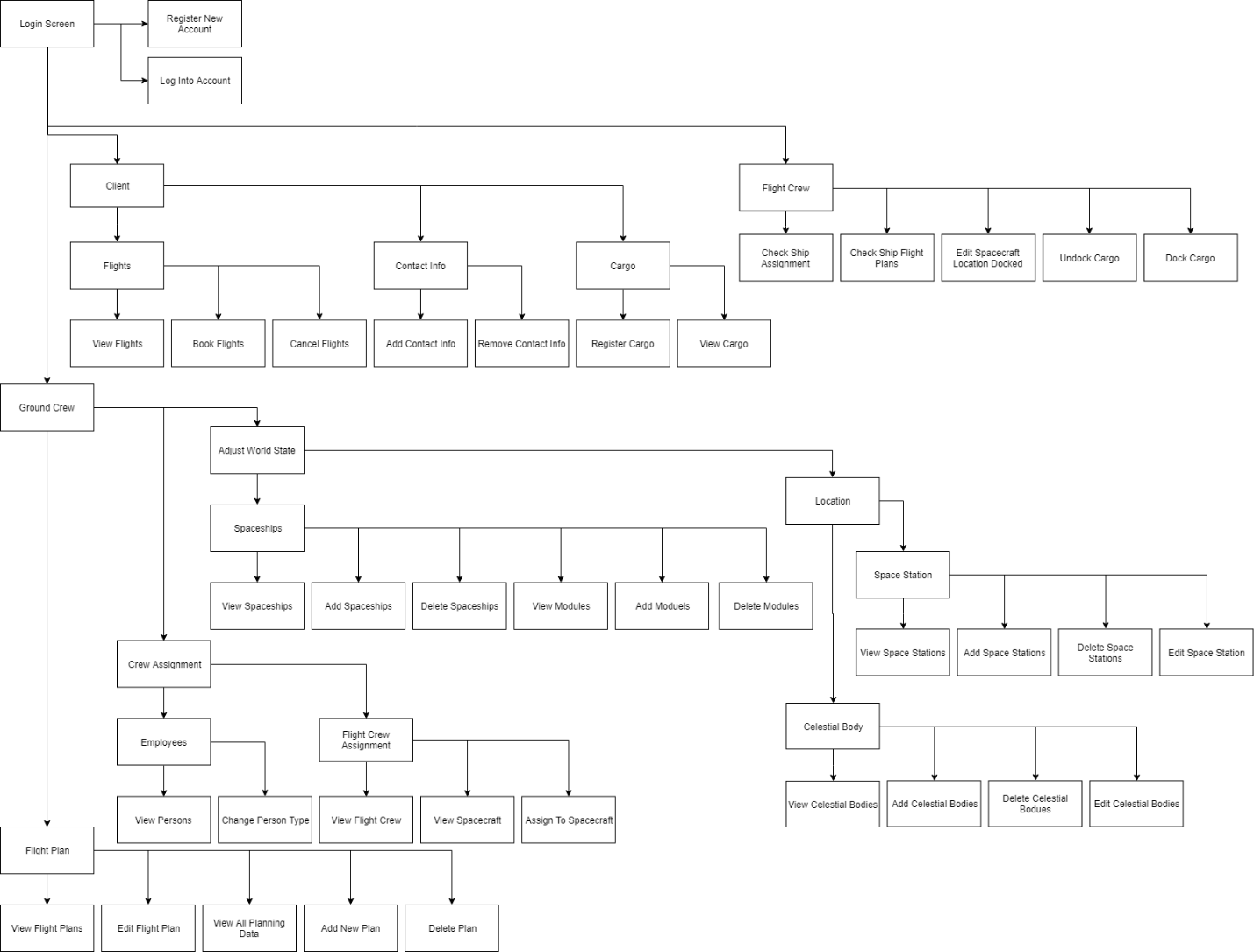


Figure 1: HIPO Diagram of the Implemented System. A larger edition of the flow-chart is available as a .png picture at: <https://photos.app.goo.gl/VGu7gfJtx1aNxwKY9>

Now, onto the users. Here, we will detail what we mean by each of the of the end-users, their expected use-cases for this system, the segment of the diagram this corresponds to, and the expected user flow. First, all end-users need to go through the ‘Login Screen’ module, where they register and log into the system. They are taken to a different page providing unique functions depending on their role. Now, onto the details of each user:

End-Users:

**Clients**: Clients are the customers of the space transportation company, who wish to transport themselves and their cargo from place to place. In general, we envisioned that clients will want to be able to view, book, and cancel their flights, manage their contact info, and add in or check what cargo they have registered with the company.

As with all users, first the client goes through the ‘Login Screen’ module, seen on the top left, going through the processes of ‘Register New Account’, and ‘Log Into Account’.

From there, they will be recognized as a client, and linked to the webpage corresponding to clients, shown under the label ‘client’ above. On this webpage, they may adjust their contact information via the ‘Contact Info’ module’s ‘Add Contact Info’ and ‘Remove Contact Info’. They may register and examine what cargo they have recorded by the company’s system via the ‘Cargo’ module’s ‘Register Cargo’ and ‘View Cargo’. Finally, they can handle which flights they want to personally be on via the ‘Flights’ module’s ‘View Flights’, ‘Book Flight’ and ‘Cancel Flight’ options. This is all that is expected of the client end-user.

**Flight Crew**: Flight crew are the employees assigned to a given spacecraft within the company’s database. Their responsibilities include knowing which ship they are assigned to, docking/undocking cargo from the starship, and knowing the spacecraft’s flight plan and updating information accordingly.

After going through the ‘Login Screen’ Module, Flight Crew users are streamed towards the ‘Flight Crew’ Module’s webpage.

From the ‘Flight Crew’ module, the flight crew member can check which ship they are assigned to with ‘Check Ship Assignment’, check flight plans for the ship they are assigned to with ‘Check Ship Flight Plan’, adjust where their ship is docked with ‘Edit Spacecraft Location Docked’, and handle cargo transfer in and out of the ship with ‘Dock Cargo’ and ‘Undock Cargo’. Overall, the responsibilities and flow of the flight crew are simple, which fits given that the majority of their task is likely to be minutiae aboard the spacecraft they are assigned to as opposed to broad things which need to be tracked by the database.

**Ground Control**: Ground control are the ground-based employees of the transportation company and oversee managing the bulk of the system. They are responsible for planning and creating flight plans, managing which of the persons in the system is an employee or not and of which kind, assigning flight crew to their spacecraft, and detailing what spacecraft and locations are in the database or not.

Past the log-in screen, the Ground Control user is streamed towards the ‘Ground Crew’ module, where they have a massive host of responsibilities.

Their first responsibility is to plan out flight plans. They need to know what flight plans and what entities exist for them to plan flights around. Their tasks include making, changing and removing flight plans when necessary. From the ‘Ground Control’ module, they then can go into the ‘Flight Plan’ module, where they can view the plans with ‘View Flight Plan’, view the information needed to plan new plans with ‘View All Planning Data’, and Add/Edit/Delete plans with ‘Add Plan’, ‘Edit Plan’ and ‘Delete Plan’, respectively.

Their second responsibility is to manage the company’s employees. They do this through the ‘Crew Assignment’ module. There are two things they must manage – first, what people in the database are of what type of employee or client, and second, which flight crew are assigned to which spacecraft. The first they can do with the ‘Employees’ submodule, with the functions ‘View Person’ and ‘Change Person Type’. The second they can do with the ‘Flight Crew Assignment’ module, with ‘View Flight Crew’, ‘View Spacecraft’ and ‘Assign to Spacecraft’.

Their third responsibility is to manage the world-state represented in the database. This they do through the ‘Adjust World State’ module. There are two types of entities they must manage: Spaceships and Locations, of which there are two types of locations – Celestial Bodies and Space Stations. To manage Spaceships, they go to the Spaceships submodule, where they can Add/View/Delete the spaceship from the registry, along with Add/View/Deleting the modules on a spacecraft. To manage Locations, they go to the Locations submodule, where they can view Celestial Bodies and Space Stations, along with adding/deleting each location as necessary. All of this is via similarly labelled leaf nodes.

To facilitate the implementation of these end-uses, the Entity-Relationship diagram of the database was drawn up with these functionalities and requirements in mind. Below is the diagram:

EER-Diagram:

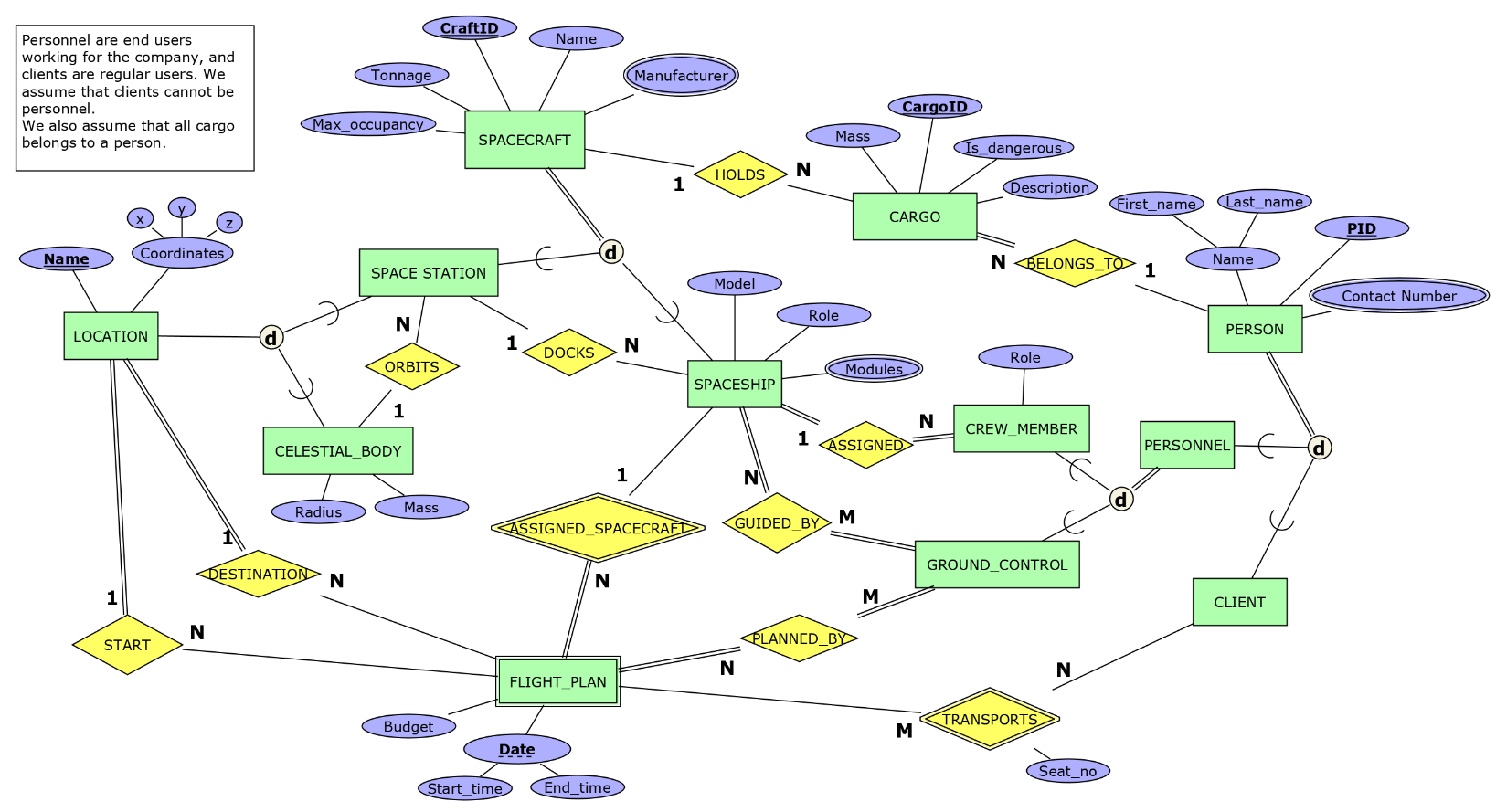


Figure 2: The EER Diagram of the Planned Database System.

Notes on Implemented Modifications: Since the last presentation, no significant changes to the ER diagram have occurred. The most significant change is the inclusion of a type attribute to person, to allow us to track which kind of end user the person belonged to. A few other attributes were renamed, such as CraftID to Craft\_ID, for the sake of consistency, and a Username and Password attribute were added to the Person entity for the purposes of managing and handling the three types of end users. Digital Copy with Higher resolution available at: <https://photos.app.goo.gl/VGu7gfJtx1aNxwKY9>

Project Implementation:

The following is documentation of the implementation of the project in functionality. We will detail the structure of the database, the implementation of its functional SQL queries, and the details of the interface itself. Starting with the structure of the database:

Relational Model – the Structure of the Database:

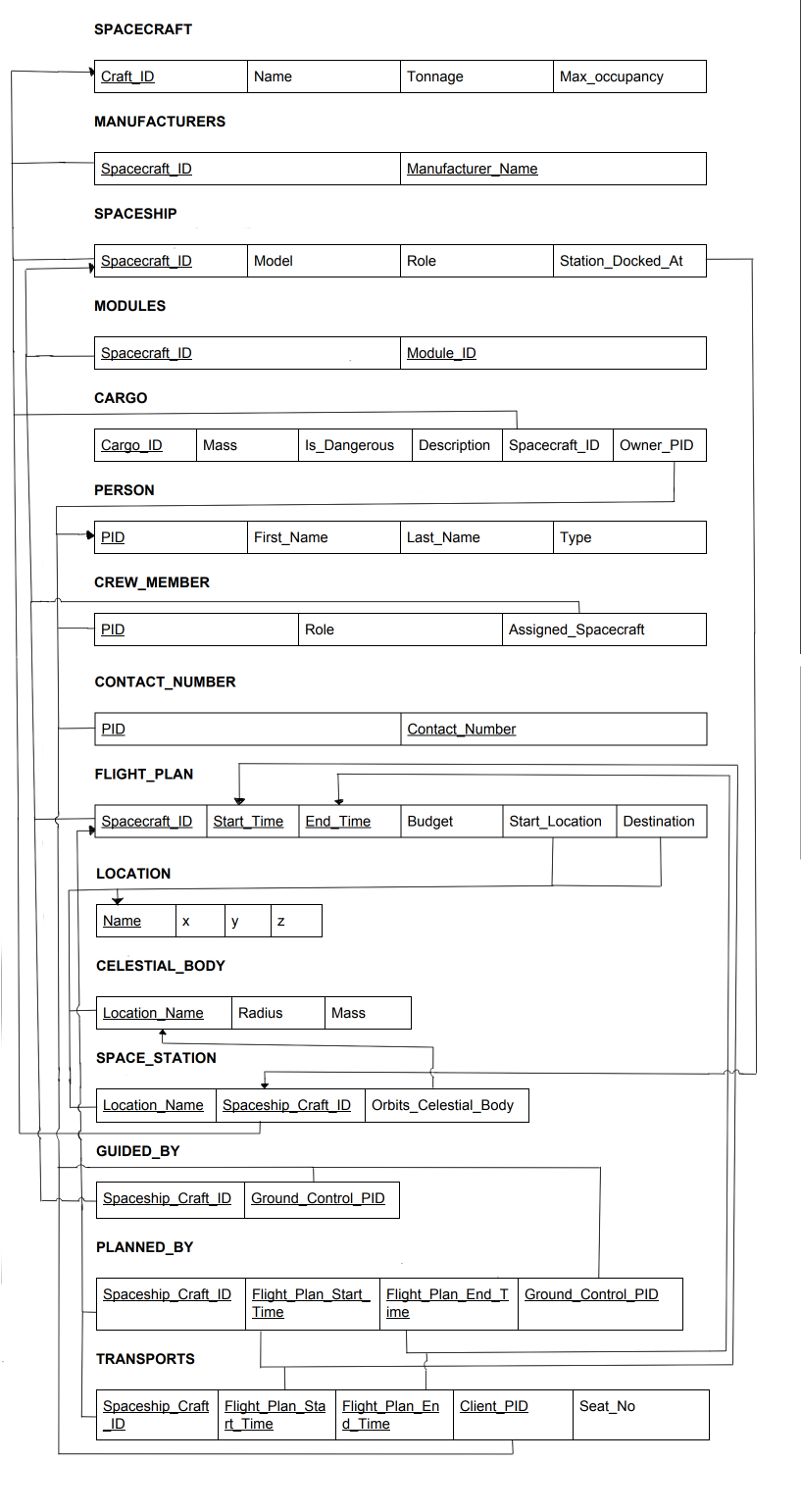


Figure 3: The Relational Model of the Database, as Constructed Via the Entity to Relational Algorithm

Notes on Implemented Modifications: Since the last presentation, no significant changes to the Relational diagram have occurred. A few attributes were renamed, such as CraftID to Craft\_ID, or Spaceship\_Craft\_ID to Spacecraft\_ID, for the sake of consistency, and a Username and Password attribute were added to the Person relation for the purposes of managing and handling the three types of end users. Furthermore, Person IDs and Spacecraft IDs were set to auto-increment, for ease of adding in new entries. Digital copy with higher resolution available at: <https://photos.app.goo.gl/VGu7gfJtx1aNxwKY9>

Database Type Selected:

To implement our database solution, we utilized the DBMS MySQL, installed with a local server via Appserv (1). Appserv was used to set up our local server and MySQL since it was recommended during our tutorials. Our website code was written in HTML/CSS and PHP.

**Implementation of the Functional SQL Statements:**

The implementation of SQL functions largely consists of the leaf nodes of the HIPO diagram above. Listed here, by their hierarchy of inheritance and function, they are:

**Registration:**

Register New Account:

Insert Into Person (Username, Password, First\_Name, Last\_Name, Type)

Values ('@username, @password, @fname, @lname, @type);

Register Crew Member:

Insert Into Crew\_Member Values (@pid, @role, @assignedspaceship);

**Login:**

Retrieve User Information:

Select \* From Person Where Username = @user;

**Client Module:**

Flights:

View Your Flights:

Select \*

From (Flight\_Plan as f JOIN Transports as t ON(f.Spacecraft\_ID = t.Spacecraft\_ID

and f.Start\_Time = t.Flight\_Plan\_Start\_Time and f.End\_Time = t.Flight\_Plan\_End\_Time))

Where Client\_PID = @PID;

View Available Flights:

Select f.\* From (Flight\_Plan as f) Where Not Exists

(Select \* from Transports as t Where f.Spacecraft\_ID = t.Spacecraft\_ID

and f.Start\_Time = t.Flight\_Plan\_Start\_Time and f.End\_Time = t.Flight\_Plan\_End\_Time

and t.Client\_PID = $pid);

Book Flight:

Insert into Transports (Spacecraft\_ID, Flight\_Plan\_Start\_Time, Flight\_Plan\_End\_Time, Client\_PID)

Values (@spacecraftid, @starttime, @endtime, @ClientPID)

Cancel Flights:

Delete From Transport Where Spacecraft\_ID = @Spacecraft\_ID and

@Start\_Time = @Flight\_Plan\_Start\_Time and @End\_Time = @Flight\_Plan\_End\_Time

and Client\_PID = @PID;

Cargo:

View Your Cargo:

Select \* From Cargo Where Owner\_PID=@Client\_PID;

Add Cargo:

Insert Into Cargo (Mass, Is\_Dangerous, Description, Owner\_PID)

Values (@Mass, @Is\_Dangerous, @Description, @Owner\_PID);

Remove Cargo:

Delete From Cargo Where Cargo\_ID = @CargoID;

**Flight Crew Module:**

View Crew Member Info:

Select \* From (Crew\_Member Natural Join Person) Where PID = @CrewPID;

View Ship’s Flight Plans:

Select \* From Flight\_Plan Where Spacecraft\_ID=@AssignedSpacecraftId;

View Ship’s Cargo:

Select \* From Cargo Where Spacecraft\_ID = @AssignedSpacecraftId;

Dock Cargo:

Update Cargo Set Spacecraft\_ID = @AssignedSpacecraftId Where Cargo\_ID = @ChosenCargoID;

Undock Cargo:

Update Cargo Set Spacecraft\_ID = NULL Where Cargo\_ID = @ChosenCargoID;

**Ground Crew Module:**

Crew Assignment:

Employees:

View Persons:

Select p.PID, p.First\_Name, p.Last\_Name, p.Type, p.Username

From Person as p;

Change Person Type:

Update Person as p

Set p.Type = @Type

Where p.PID = @PID or p.Username = @Username;

Flight Crew:

View Flight Crew:

Select p.PID, p.First\_Name, p.Last\_Name, p.Type, p.Username, c.Role, c.Assigned\_Spaceship

From Person as p Natural Join Crew\_Member as c

Where p.Type = 'Flight Crew';

View Spaceship:

Select \*

From Spacecraft Natural Join Spaceship;

Assign to Spaceship:

Update Crew\_Member as c

Set c.Assigned\_Spaceship = @Spacecraft\_ID

Where c.PID = @Our\_PID;

Flight Plan:

View Flight Plans:

Select \*

From Flight\_Plan;

Edit Flight Plan:

Update Flight\_Plan as f

Set f.Start\_Time = @Start\_Time, f.End\_Time = @End\_Time, f.

Where f.Spacecraft\_ID = @Our\_Spacecraft\_ID and f.Start\_Time = @Our\_Start\_Time and f.End\_Time = @Our\_End\_Time;

Update Planned\_By

Set Start\_Time = @Start\_Time, End\_Time = @End\_Time

Where Spacecraft\_ID = @Our\_Spacecraft\_ID and Start\_Time = @Our\_Start\_Time and End\_Time = @Our\_End\_Time;

Insert into Planned\_By Values(@Our\_Spacecraft\_ID, @Start\_Time, @End\_Time, @Ground\_Crew\_PID);

View All Planning Data:

View Spaceship:

Select \*

From Spacecraft Natural Join Spaceship;

View Locations:

Select \*

From (Locations Natural Left Join Space\_Station) Left Natural Join Celestial\_Body;

Add New Plan:

Insert Into Flight\_Plan Values(@Spacecraft\_ID, @Start\_Time, @End\_Time, @Budget, @Start\_Location, @Destination);

Insert Ignore into Planned\_By Values(@Our\_Spacecraft\_ID, @Start\_Time, @End\_Time, @Ground\_Crew\_PID);

Delete Plan:

Delete From Planned\_By

Where Spacecraft\_ID = @Our\_Spacecraft\_ID and Start\_Time = @Our\_Start\_Time and End\_Time = @Our\_End\_Time;

Delete From Flight\_Plan

Where Spacecraft\_ID = @Our\_Spacecraft\_ID and Start\_Time = @Our\_Start\_Time and End\_Time = @Our\_End\_Time;

Adjust World State:

Spaceships:

View Spaceships:

Select \*

From Spacecraft Natural Join Spaceship;

Add Spaceships:

Insert Into Spacecraft Values(@Name,@Tonnage,@Max\_Occupancy);

@Spacecraft\_ID =

(Select s.PID

From Spacecraft as s

Where s.Name = @Name, s.Tonnage = @Tonnage, s.Max\_Occupancy = @Max\_Occupancy;)

Insert Into Spaceship Values(@Spacecraft\_ID, @Model,@Role,@Station\_Docked\_At);

Delete Spaceships:

Delete From Spaceship

Where Spacecraft\_ID = @Spacecraft\_ID;

Delete From Spacecraft

Where Spacecraft\_ID = @Spacecraft\_ID;

Edit Spaceships

Update Spaceship

Set Model = @Model, Role = @Role, Max\_Occupancy = @Max\_Occupancy

Where Spacecraft\_ID = @Spacecraft\_ID;

Update Spacecraft

Set Name = @Name, Tonnage = @Tonnage, Max\_Occupancy = @Max\_Occupancy

Where Spaceship = @Spacecraft\_ID;

View Spaceship Modules

Select m.Module\_ID

From Modules as m

Where m.Spacecraft\_ID = @Spacecraft\_ID;

Add Spaceship Modules

Insert Into Modules Values(@Spacecraft\_ID, @Module\_ID);

Delete Spaceship Modules

Delete From Modules

Where Spacecraft\_ID = @Spacecraft\_ID and Module\_ID = @Module\_ID;

Locations:

Space Station:

Add Spacestation:

Insert Into Spacecraft Values(@Name,@Tonnage,@Max\_Occupancy);

@Spacecraft\_ID =

(Select s.PID

From Spacecraft as s

Where s.Name = @Name, s.Tonnage = @Tonnage, s.Max\_Occupancy = @Max\_Occupancy; )

Insert Into Location(@Location\_Name, @X, @Y, @Z);

Insert Into Space\_Station(@Location\_Name, @Spacecraft\_ID, @Orbits\_Celestial\_Body);

Delete Space station:

Delete From Space\_Station

Where Spacecraft\_ID = @Spacecraft\_ID and Location\_Name = @Location\_Name;

Delete From Spacecraft

Where Spacecraft\_ID = @Spacecraft\_ID;

Delete From Location

Where Location\_Name = @Location\_Name;

Edit Space Station:

Update Spacecraft

Set Name = @Name, Tonnage = @Tonnage, Max\_Occupancy = @Max\_Occupancy

Where Spacecraft\_ID = @Spacecraft\_ID;

Update Location

Set Location\_Name = @Location\_Name, x = @X, y = @y, z = @z

Where Location\_Name = @Location\_Name;

Update Space\_Station

Set Location\_Name = @Location\_Name, Orbits\_Celestial\_Body = @Orbits\_Celestial\_Body

Where Spacecraft\_ID = @Spacecraft\_ID;

View Space stations:

Select \*

From Spacecraft Natural Join Space\_Station;

Celestial Body:

Add Celestial Body:

Insert Into Locations Values(@Location\_Name, @x, @y, @z);

Insert Into Celestial Body Values(@Location\_Name, @Radius, @Mass);

Delete Celestial\_Body:

Delete From Celestial\_Body

Where Location\_Name = @Location\_Name;

Delete From Location

Where Location\_Name = @Location\_Name;

Edit Celestial Body:

Update Celestial\_Body

Set Radius = @Radius, Mass = @Mass

Where Location\_Name = @Location\_Name;

Update Location

Set x = @x, y = @y, z = @z

Where Location\_Name = @Location\_Name;

View Celestial Bodies:

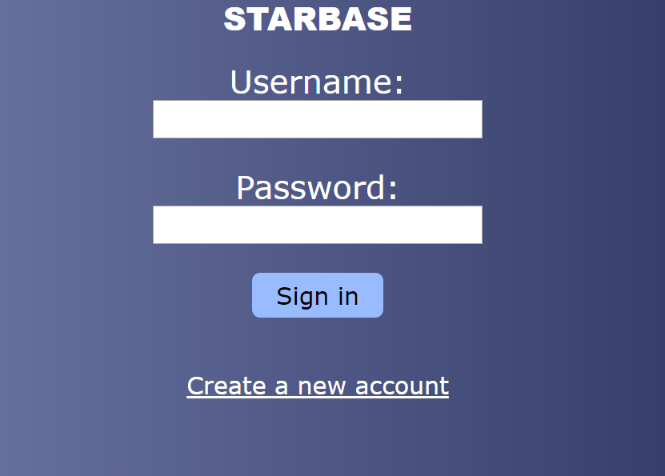
Select \*

From Celestial\_Body Natural Join Location;

**User Manual:**

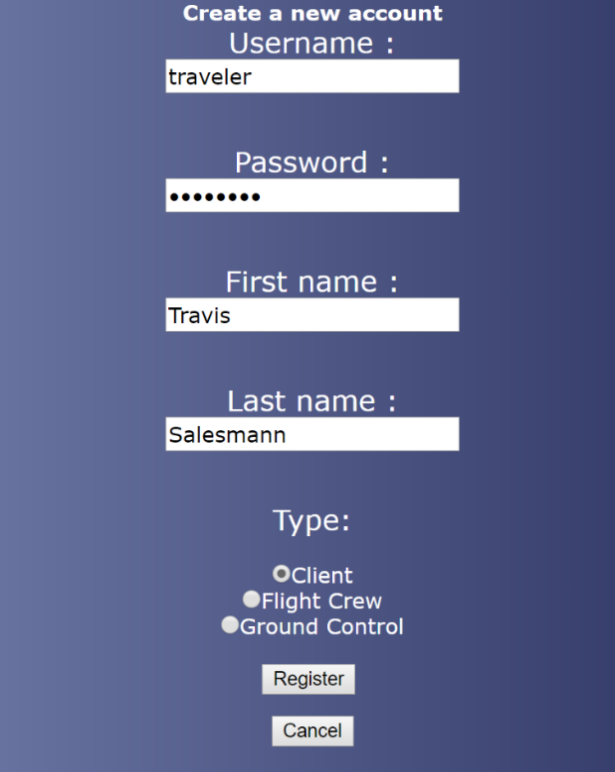
Welcome, new user, to the Starbase flight system. This user manual will guide you through the main processes of this software. Main page and registration sections will be encountered by all. Afterwards, if you are a client, skip to the client section. If you are a ground crew member, skip to the ground crew section. If you are a member of the flight crew, go to the flight crew section. If you are seeing the wrong screen after logging in, please contact a ground crew member.

Main Page



The main page of the website is login.php. Here, you may sign in with an existing account or create a new account. After logging in, you will be redirected to the Client, Flight Crew or Ground Crew portal depending on your account role.

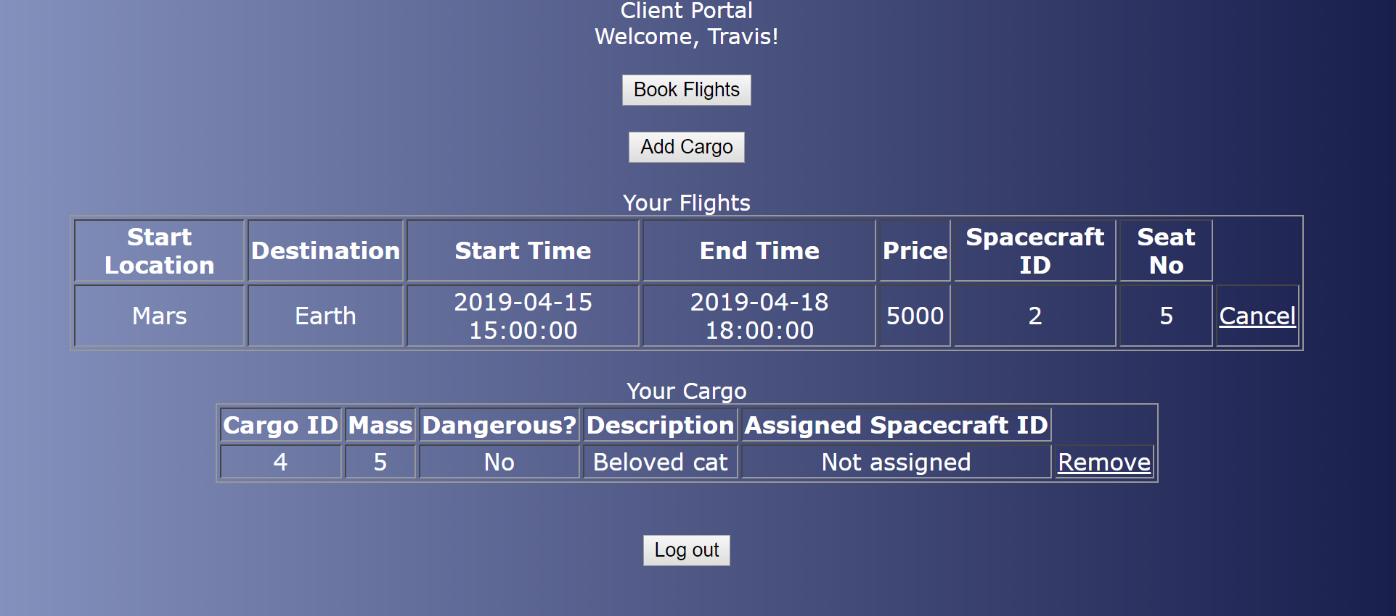
Registration



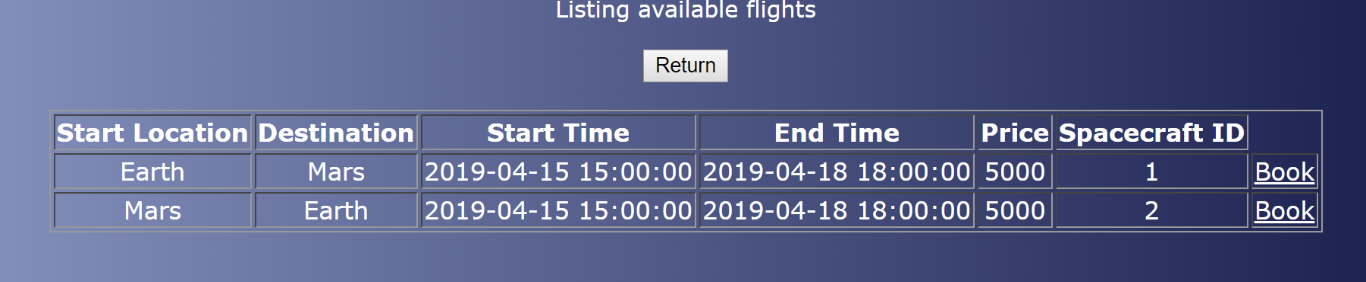
To create a new account, enter your information in the form shown above on register.php.

Clients

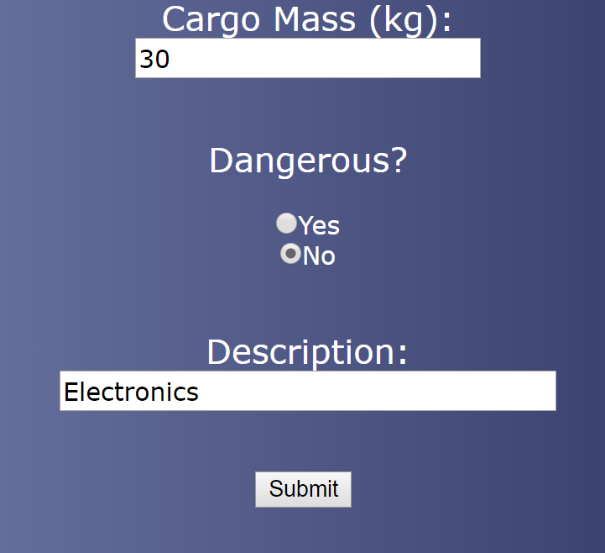
From the Client Portal, clients can view flights they have booked and cargo they have registered.



To book a flight, clients can select from the available flight plans. The chosen flight can then be viewed from the main client page.



Clients can add and remove their cargo as well. Cargo is added by filling the cargo information form.

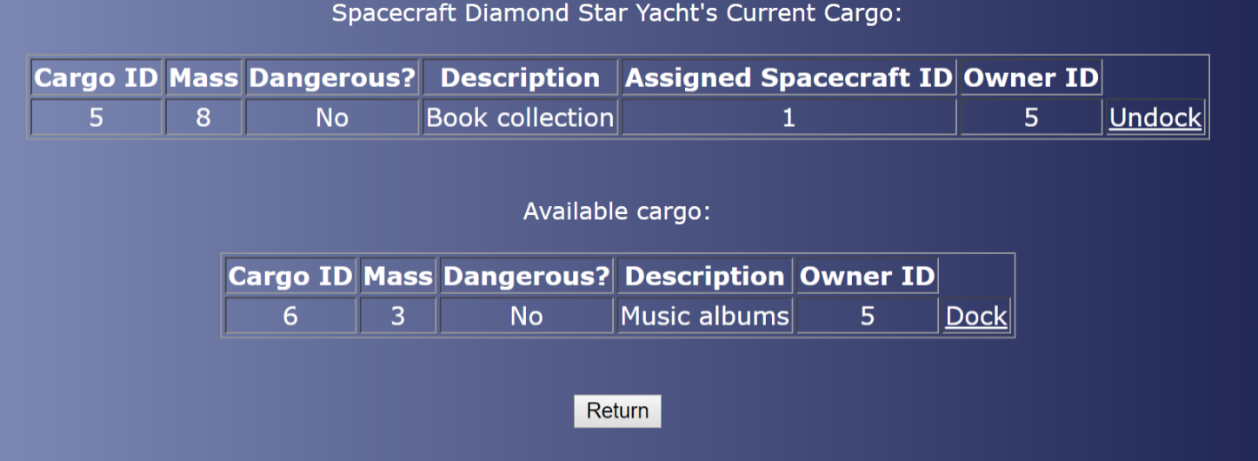


Flight Crew

From the Flight Crew Portal, crew members may view their assigned spaceship, their role, and upcoming flights for their ship.

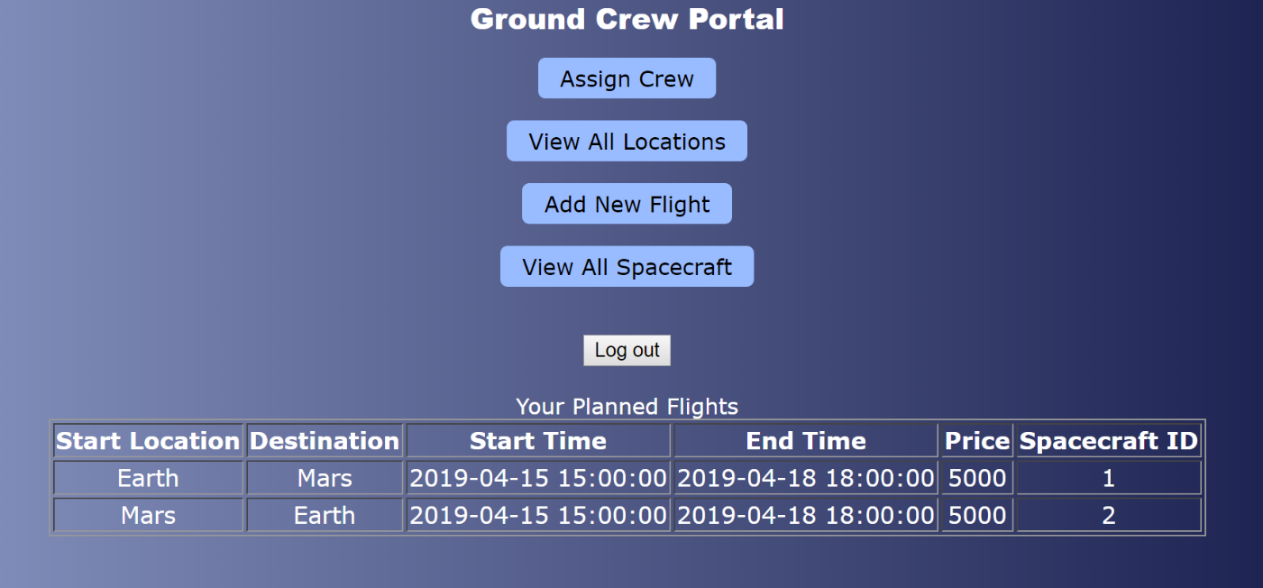


They can manage cargo by selecting items to dock and undock from their assigned ship.

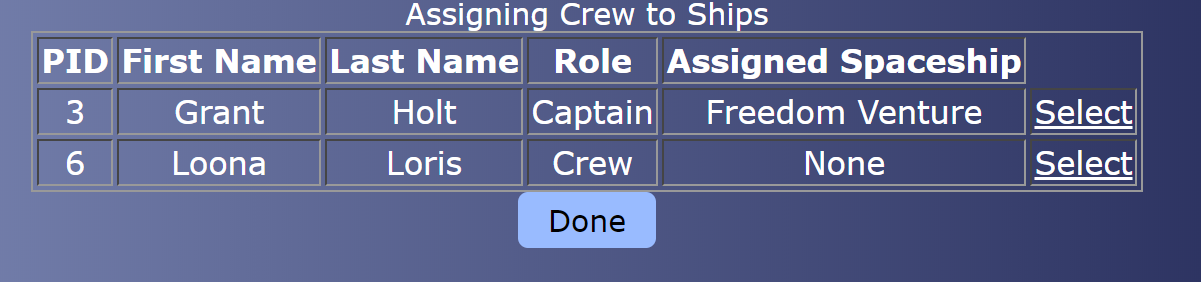


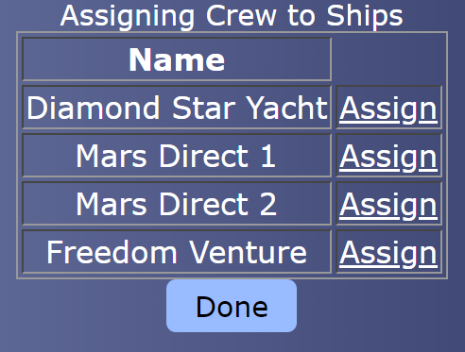
Ground Crew

Ground Crew members are responsible for updating the world state of the database. They have many tasks, such as adding new locations, spacecraft and flight plans to Starbase.

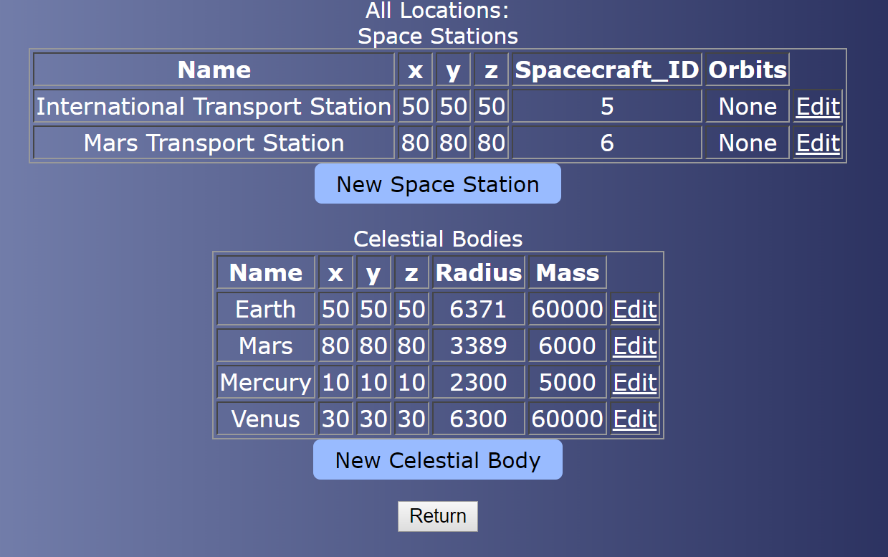


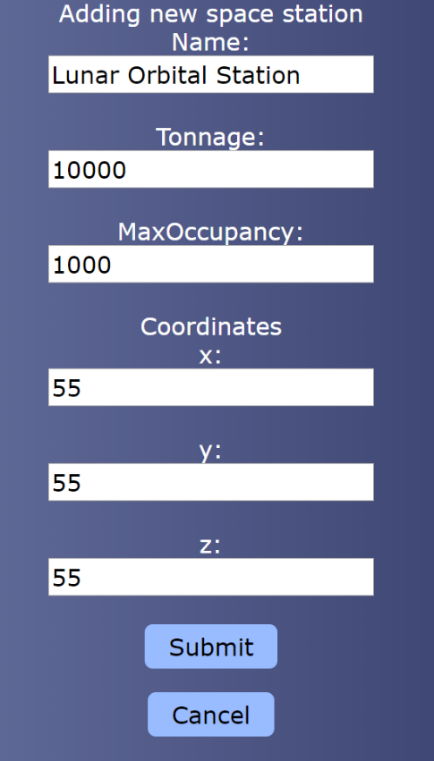
On the Ground Crew Portal screen, ground crew members can view flights that they have created.



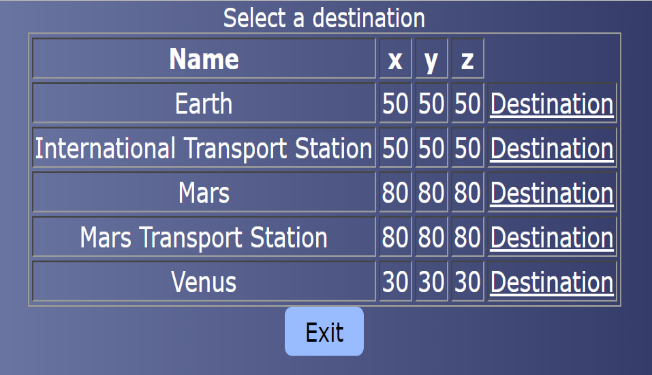
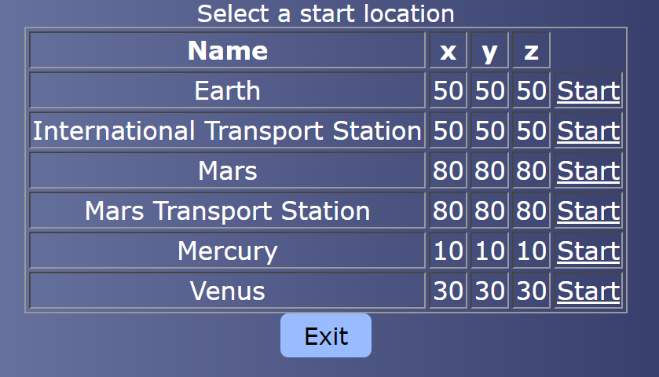


Ground crew members can assign flight crew to ships. This is done by selecting the crew member to update and choosing which ship they should be moved to.





They can also view and add new locations and spacecraft to the database.





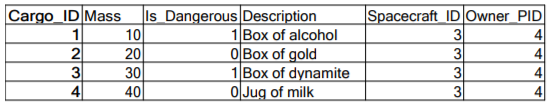
To create new flight plans, ground control selects departure and destination locations and times, as well as the transporting spaceship and budget.

That’s it for the User Manual. Now you should be able to have a grasp on how exactly to utilize this piece of software!

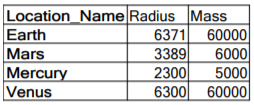
**Appendix:**

Sample Data Records Utilized to Populate the Database:

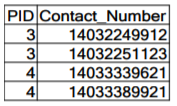
Cargo:



Celestial Body:



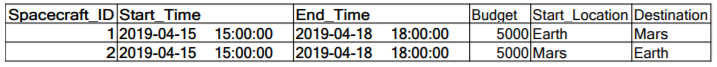
Contact Number:



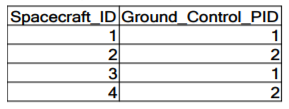
Crew Member:



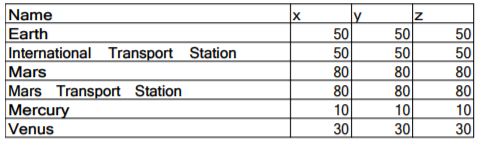
Flight Plan:



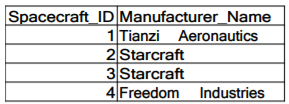
Guided By:



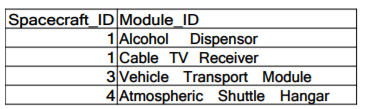
Location:



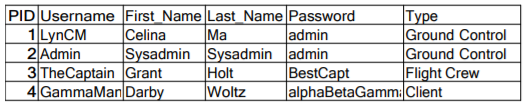
Manufacturers:



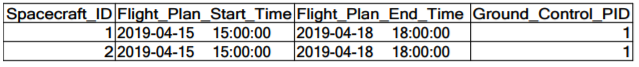
Module:



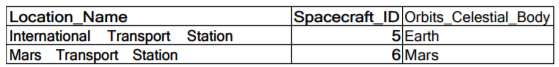
Person:



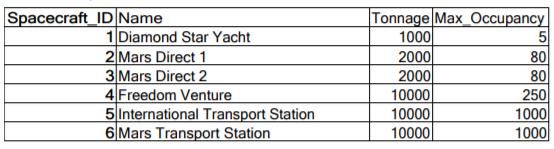
Planned By:



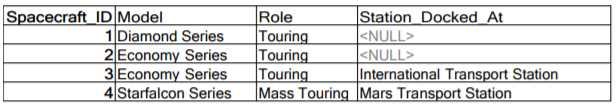
Space Station:



Spacecraft:



Spaceship:



Transports:



**References**

1. Appserv. (published 2001, Oct 9). *Appserv: Apache + PHP + MySQL*. Retrieved from <https://www.appserv.org/en/about/>