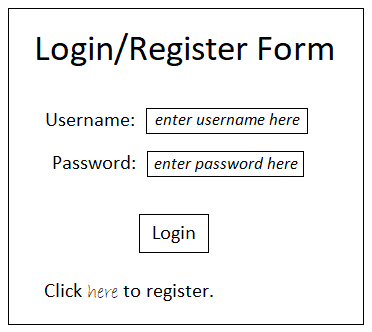
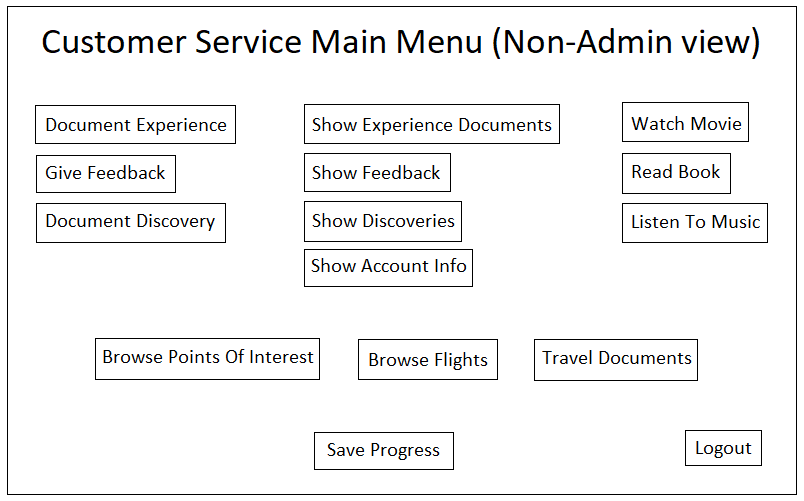
# Customer GUI

The Customer service GUI will be used by passengers and administrators of the ISTS and is the initial point of entry into the ISTS GUI service. Administrators can click on navigational buttons on the screen in order to get to and from the Resource Management, and Flight Manager GUIs while non-administrative passengers can only use the Customer service GUI. For administrators, the service will remember which interface was being used at logout so they can pick up where they left off when they sign back in. When users first sign in to the service, the most up-to-date Customer service data for the user is automatically downloaded from the remote persistence storage location on the IPFS to local memory using the “pullFromIpfsRepo” API method. Below is what the login/register form should approximately look like:



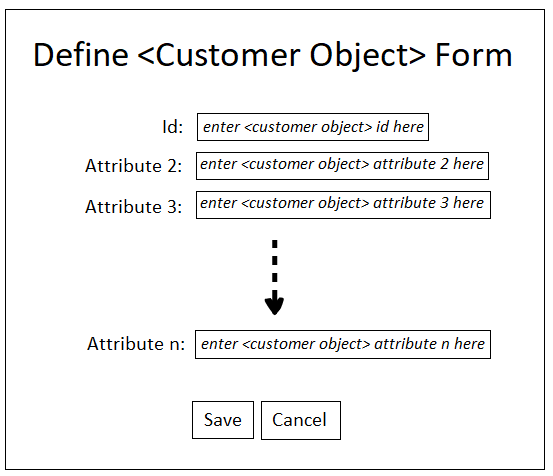
Caption: The login/register form for the Customer service GUI. The “Click here to register.” link at the bottom opens up a “Define <Customer Object>” form (defined later in this section) when clicked where passengers can fill out registration information to create an account.

After users login, they should be brought to the main menu. Below is what the main menu screen of the Customer service GUI should approximately look like:



Caption: The main menu screen of the Customer service GUI for non-administrative passengers. The Admin GUI is shown at the end of this Customer GUI section.

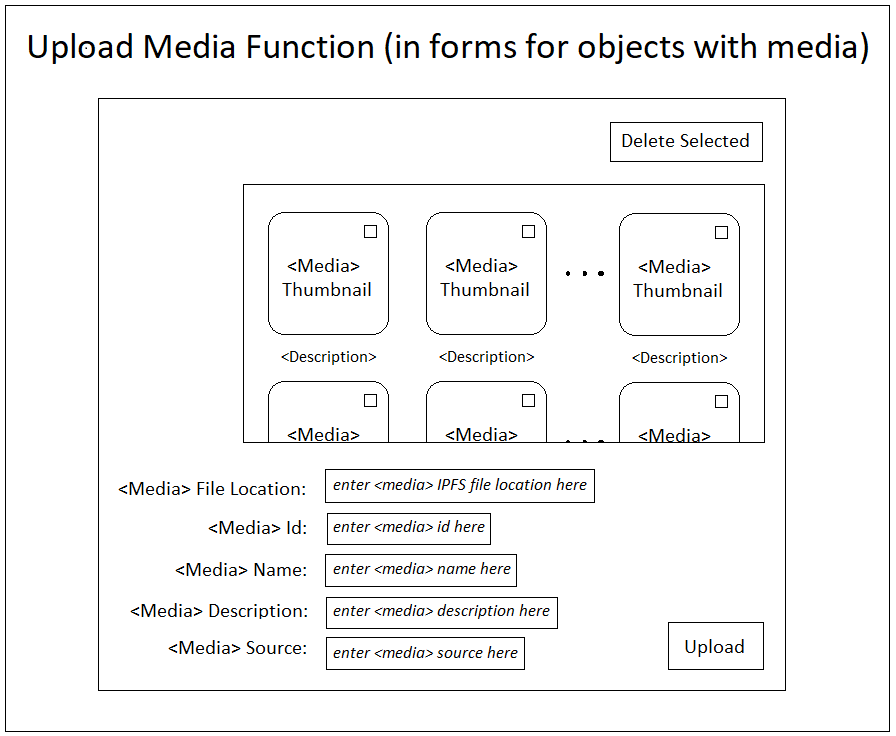
The top left column of buttons on the main menu that allow the user to create documents when clicked on bring up a form window that should look approximately like the following:



Caption: The generic “Define <Customer Object>” form of the Customer service GUI where n is the number of attributes. It allows users to create a document such as a discovery made on a trip by defining its attributes in a form and then hitting the save button.

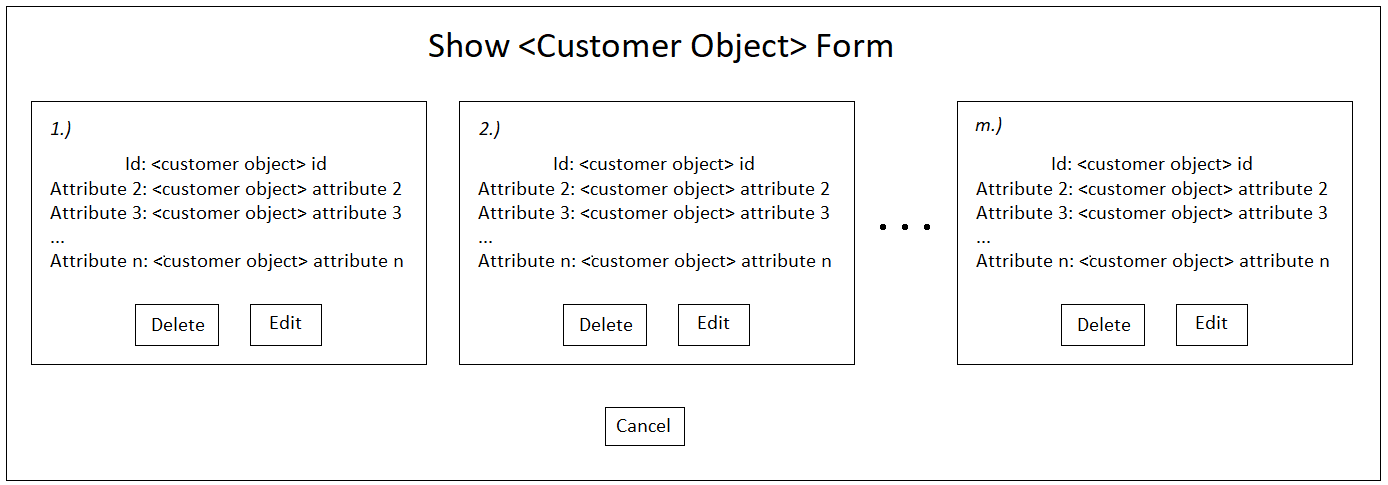
After hitting the save button, the GUI will use the appropriate Customer service API method to create the new type of Customer object. For instance, if the form was for defining a new Discovery, then the “defineDiscovery” method would be called with the inputted attributes as parameters.

For Customer objects that contain media (e.g., Discovery), the “Define <Customer Object>” form should include media management functionality that should look approximately like the following:



Caption: The upload-media functionality that’s included on forms for defining Customer objects that contain media, e.g., experience documentation can have video recordings. Clicking on a thumbnail opens the media in another window for an enlarged view and/or to be listened to or read. Uploading new media creates a new clickable thumbnail in the list for the media. Media can also be checked and deleted from the list.

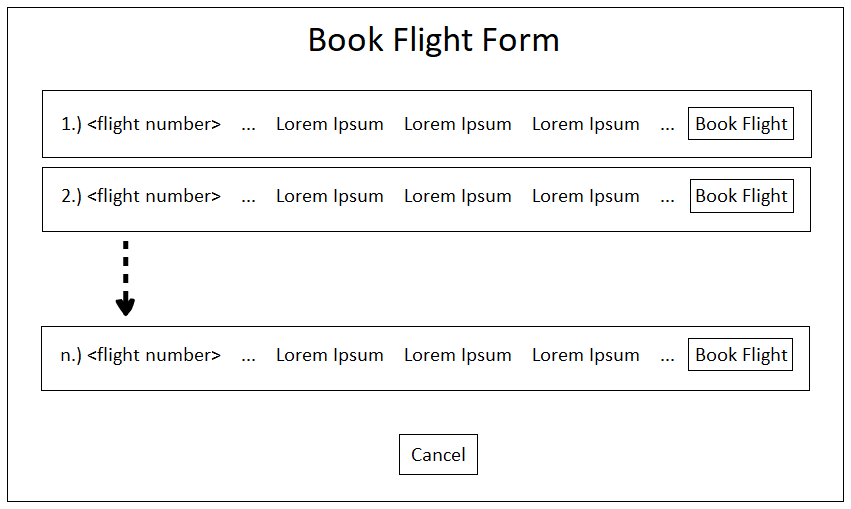
The “Show <Customer Object>” buttons in the top middle column of the main menu show the information each of the types of Customer objects contain when clicked on. For instance, “Show Account Info” should bring up a form window with the passenger’s account registration information that should look approximately like the following:



Caption: The generic “Show <Customer Object>” form of the Customer service GUI where n is the number of attributes and m is the number of the specific Customer object. It allows users to view, update, and delete Customer objects. When clicking on an edit button, the corresponding Customer service object’s “Define <Customer Object>” form should be brought up in a new window to allow for editing.

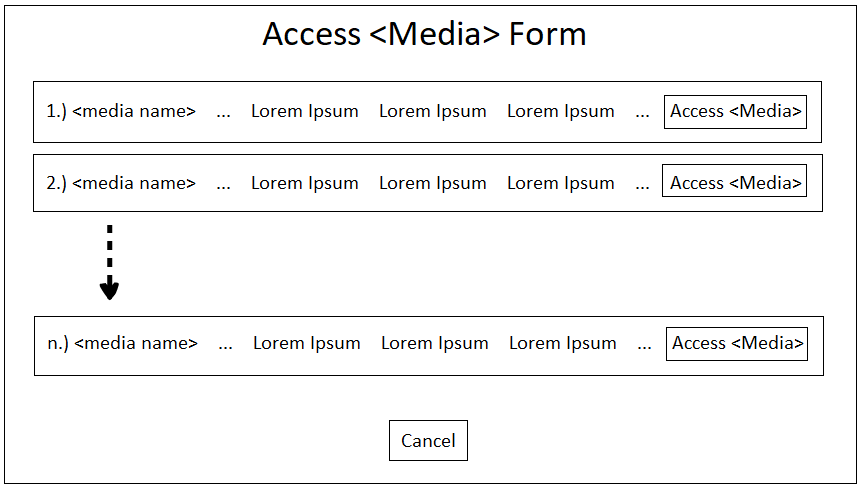
When objects are shown and potentially updated, the GUI utilizes the Customer service accessor API methods. For instance, to show all the user’s discoveries, it would need to call the “getDiscoveries” method.

The middle row of buttons (above the “Save Progress” button) in the main menu are for browsing points of interest and flights, booking flights, and accessing important travel documents and flight information after booking a flight. These can’t be edited by passengers; only read (except passengers can upload their passport and visa information on the form that opens when they click on the “Travel Documents” button). The “Browse Flights” button brings up another window for booking flights that looks approximately like the following:



Caption: The form for booking flights in the Customer service GUI where n is the number of flights. It allows users to browse available flights and book them by clicking the “Book Flight” button. When a user books a flight, the “bookFlight” API method is called with the listed parameters.

The top right column of buttons in the main menu allow passengers to access in-flight entertainment. These can’t be edited by passengers; only read. Clicking on them should bring up a form that looks approximately like the following:



Caption: The generic form for allowing users to pick media out from a list for entertainment purposes during the flight where n is the number of list items. Users can click on the “Access <Media>” button to stream it. Media offered includes movies, music, and books. The appropriate accessor/getter API method is called per the type of media being accessed.

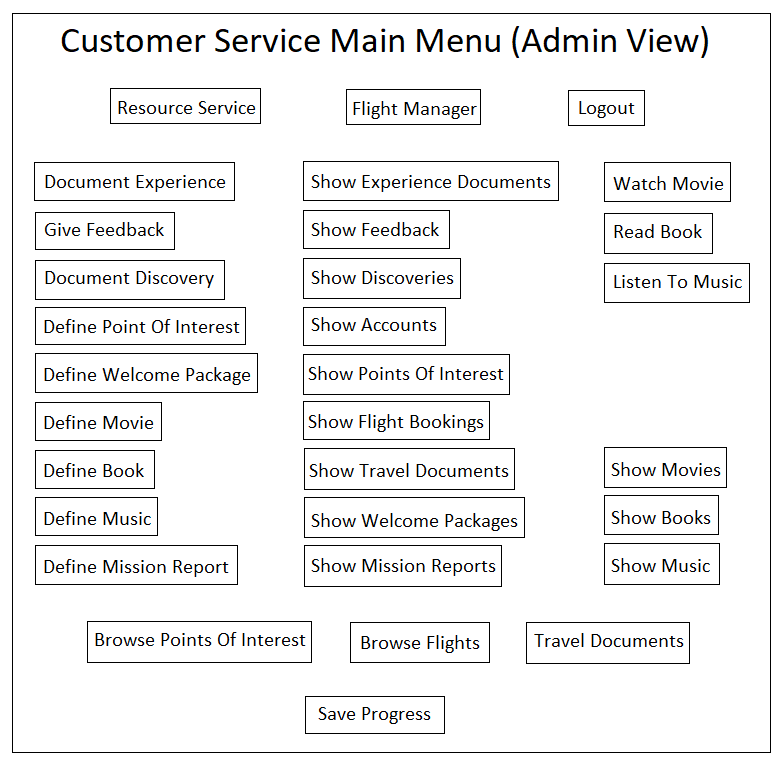
The bottom “Save Progress” button on the main menu uploads any local data updates/changes made in local memory to the remote IPFS for persistence, so that it can be retrieved again at a later time. This calls the “pullFromIpfsRepo” and “pushToIpfsRepo” API methods in succession.

**Admin Customer GUI**

The Admin view of the Customer service GUI is the same as non-admin passenger’s except for the following:

* Admins not only get back a view of their own data but every passenger’s when showing experience documents, feedback, discoveries, and accounts.
* Admins can navigate to other services (Resource and Flight Manager).
* Admins can define more types of Customer objects and edit them than regular passengers, e.g, points of interest, travel documents, in-flight entertainment, welcome packages, and mission reports.
* Admins can view (and edit) all passenger account information, and all flight booking records.

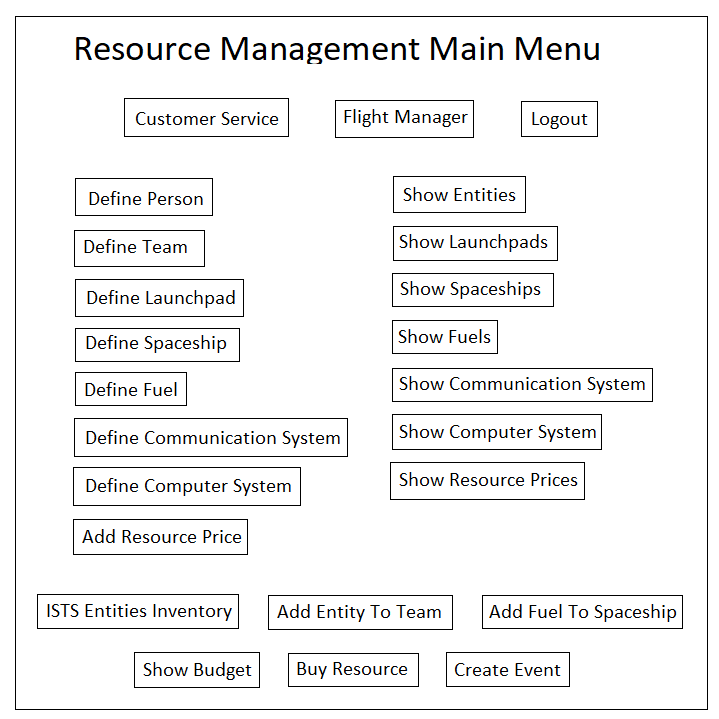
The following is what the Admin’s main menu should approximately look like:



Caption: The main menu screen of the Customer service GUI for Admins. Admins are able to have their own personal user objects and also act administratively over all objects. For instance, the “Travel Documents” button accesses the Admin’s personal user travel documents while the “Show Travel Documents” button would return every ISTS passenger’s travel documents since the Admin is allowed to read, update, and delete any of them. For another example, Admins can watch movies by clicking the “Watch Movie” button and also update and delete movies by clicking the “Show Movies” button or define a new movie by clicking the “Define Movie” button.

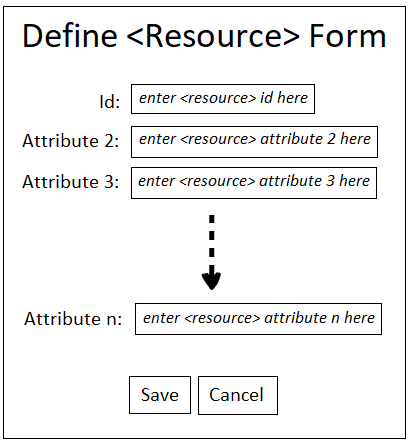
# Resource Management GUI

The Resource Management GUI will only be used by administratrative users of the ISTS. Please refer to the Customer service’s GUI section for an explanation of how logging in to the service and navigation works. Below is what the main menu screen of the GUI should approximately look like:



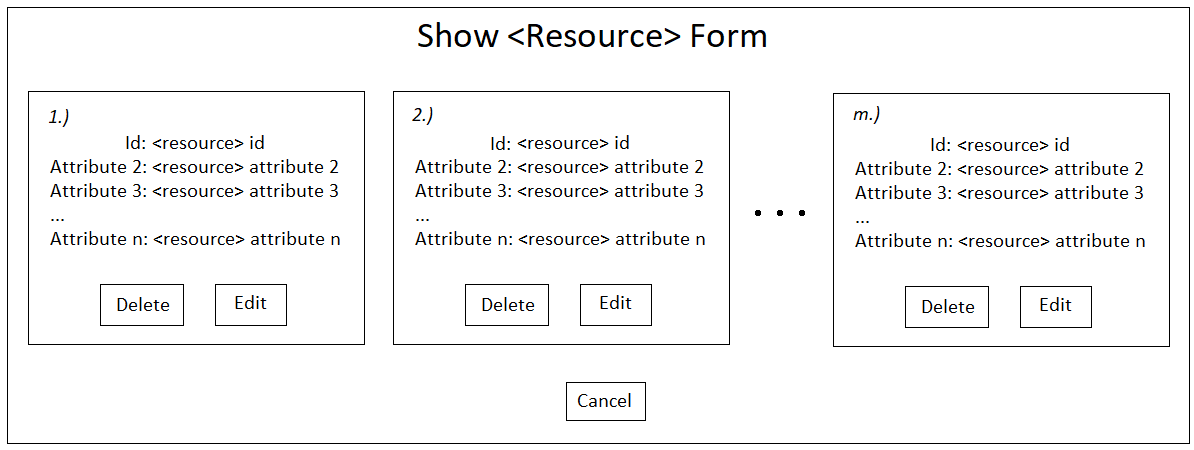
Caption: The main menu screen of the Resource Management service GUI.

The middle left column of buttons on the main menu that allow the user to define the state of each of the resources when clicked on bring up a form window that should look approximately like the following:



Caption: The generic “Define <Resource>” form of the Resource Management GUI where n is the number of attributes. It allows users to create a resource by defining its attributes in the form and then clicking the save button which would call the resource’s “define<resource>” API method with the inputted attributes as parameters.

The middle right column of buttons on the main menu that show the information each of the types of resources contain when clicked on bring up a form window that should look approximately like the following:



Caption: The generic “Show <Resource>” form of the Resource Management GUI where n is the number of attributes and m is the number of the specific resource. It allows users to view, update, and delete resources. When clicking on an edit button, the corresponding resource’s “Define <Resource>” form should be brought up in a new window to allow for updating.

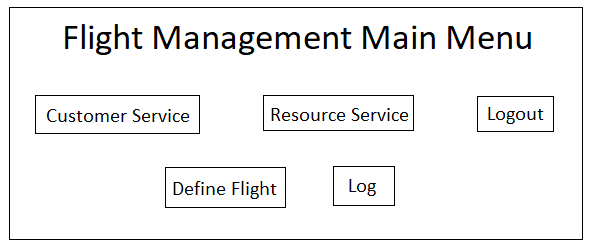
The bottom two rows of buttons in the main menu involve functions that perform or provide support for some type of behavior on or manipulation of resource data. For instance, the “ISTS Entities Inventory” button brings up a window that shows an organized form view of the people and teams that make up the ISTS organization. This could be used to organize groups of people, e.g., flight crews and passengers, in order to support managing flights. More interactive functionality could also be included.

The top row of buttons on the main menu are for navigational purposes such as directing users to other ISTS services or logging out.

The “Create Event” button on the main menu, brings up a form that allows the user to create/simulate an event by typing in an event with the appropriate event command-line syntax. The event and resulting actions are logged.

# Flight Management GUI

The Flight Management GUI will be used by administrators of the ISTS. Please refer to the Customer service’s GUI section for an explanation of how logging in to the service and navigation works. Below is what the main menu screen of the GUI should approximately look like:



Caption: The main menu of the Flight Management service GUI.

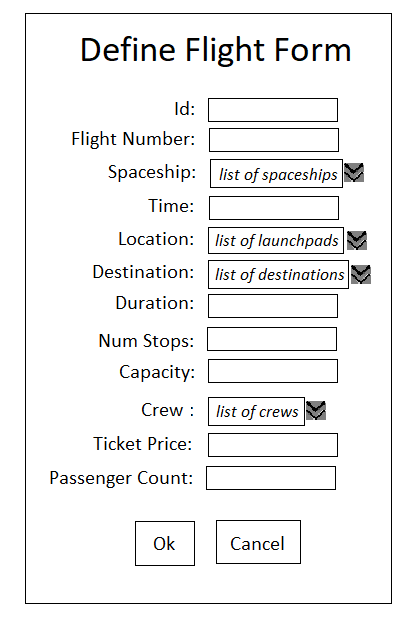
The main menu of the Flight Management service is relatively simple. This is mainly because it does not store state. Therefore, an administrator would need to navigate to the other services’ GUIs in order to monitor and update objects there. The top row of buttons navigates to the service GUI that’s clicked.

To manage resources, an administrator would navigate to the Resource Service GUI and use its interface (see the Resource Service main menu GUI defined previously). There, they could click the “Show Spaceships” button which would bring up a list of all the ISTS spaceships and their information (see the Resource Service GUI section for how this would appear). An administrator could then read, update, and delete any information of any spaceship. This includes the messages that a spaceship has, fuel level, and the number of available spacecraft. The Resource Service GUI also allows for access to the communication system (by clicking the “Show Communication System” button), the operating budget (by clicking the “Show Budget” button), and more.

To manage flight status, an administrator would navigate to the Customer service GUI (see the Customer service main menu GUI defined previously). There they could click on the “Show Flights” button which would bring up a list of all the ISTS spaceships and their information (see the Customer service GUI section for how this would appear). An administrator could then read, update, and delete any information of any flight including flight status.

The “Log” button on the Flight Management main menu should print to stdout a log of the events and actions that have happened in the automated control system when clicked by calling the “getLog” method on the Flight Management’s API.

The main function of the Flight Management GUI is flight creation. When the “Define Flight” button on its main menu is clicked, a form window should be brought up that should look approximately like the following:



Caption: The “Define Flight” form of the Flight Management GUI. It allows administrators to create a new flight by defining its attributes in the form and then clicking the Ok button which would call the Flight Management API’s “defineFlight” method with the inputted attributes as parameters. It includes a drop-down list function for some attributes to be selected for convenience.

For the above Flight Form to be created, the GUI would call the Flight Management service’s “getResourceService” and “getCustomerService” API methods in order to get the lists of external objects that are available and that are part of creating a flight where applicable. For instance the GUI would need a reference to the Resource Service to find all available spaceships in order to populate the “Define Flight” form’s list of spaceships for selection.