Project Design Document

Airline System Design

Version 1.2
February 24, 2005

Acceptance	
Network name	
Network Representative	Sarah Tshila and TJ Dorsey
Signature	
Date	

Submitted as partial fulfillment of the requirements for CS 4322.

Table of Contents

1.0 Introduction

This design document will explain in detail the aspects of the airline flight system. The user interfaces will provide a clear understanding of how the user will interact with the system. The protocol portion will explain how the clients and server will interact with each other to send and request data from the user and the database. The class designs explain how each part will work in detail and how data is handled between the database and server, between the server and client, and the client and user.

2.0 User Interface Design

The focus of the user interface design is to make navigation of the client and server as easy as possible. The interface was broken down into four sections according to the user's login status: guest, passenger, employee, and administrator. The guest user will possess the least power, while the administrator will possess the most power. The user interfaces are designed to inherit features from lower interfaces.

2.1 General User Interface

The General Interfaces will include information that is not user-specific, but is significant to the navigation of the system. Each interface will have menu options displayed at the bottom of each screen to give the user choices to complete a task.

Guest User Interface will have the following options to choose from:

```
(S)earch (L)ogin (R)egister (Q)uit
```

Passenger User Interface will have the following options to choose from:

```
(S)earch (L)ogout (B)ook Flight (R)eservation (P)rofile (Q)uit
```

Employee User Interface will have the following options to choose from:

```
(S)earch (L)ogout (B)ook Flight (R)eservation (P)rofile (W)ork Assignment (Q)uit
```

Admin User Interface will have the following options to choose from:

```
(S)earch (L)ogout (B)ook Flight (R)eservation (P)rofile (Q)uit
(F)light Control: Create, Modify, Delete
(T)ime Schedules: Modify
```

2.2 Non-Specific Interfaces

The Welcome screen will be displayed each time the client program starts up. From this screen, the user can choose to search for flights, log into the system, or register with the system.

```
WELCOME SCREEN
*********************************************************************
Welcome to the VSU Airline System

--------------------------------
```

```
Please choose an option below to proceed.

(S)earch (L)ogin (R)egister (Q)uit

*********************************************************************************
```

The Search screen is where the user will have the option to input flight parameters for the system to try and find a matching flight according to the user input.
* If all three input lines are left blank, then no search will be done.

```
SEARCH SCREEN
*********************************************************************************
Welcome to the VSU Airline System

Search for a Flight
---------------------------------

*NOTE: MUST be a registered user to book flights

Departure City:  STRING
Departure Date:  STRING in this format, MM/DD/YYYY
Destination City:  STRING

*Note: this is a one-way trip

(S)earch (L)ogin (R)egister (Q)uit

*********************************************************************************
```

The Search Results screen will display all available flights with the matching parameters that the user entered at the Search screen.

```
SEARCH RESULTS SCREEN
*********************************************************************************
Welcome to the VSU Airline System

Search Results
---------------------------------

Flight Number, Departure City and Date
Destination City
Price

Examples:
C14, from Atlanta, GA to Las Vegas on February 28, 2005 for $204

A121, from Atlanta, GA to Las Vegas on February 28, 2005 for $251

B52, from Atlanta, GA to Las Vegas on February 28, 2005 for $237

(S)earch (L)ogin (R)egister (Q)uit

*********************************************************************************
```

2.3 Guest User Interface Screens

The guest user is anyone that wishes to use the system is either not a registered user or not logged in. Guest users cannot book flights or check reservations. They can only search for flights, register with the system and login as a registered user.

The user login screen is displayed, when a user chooses the Login command from the menu. This will ask for a username and password to be entered. If the login information is correct and matches an exiting username and password, then a confirmation screen will be displayed. If the login information is incorrect, then a notice will be displayed and the user will be taken back to the login screen.

```
USER LOGIN SCREEN
*********************************************************************************
Welcome to the VSU Airline System

User Login
---------------------------------

Enter your username:
Enter your password:

(Message will appear either confirming or denying access)

(S)earch (L)ogin (R)egister (Q)uit

*********************************************************************************
```

The user registration screen is where new users will be added to the system. The user will input their personal information and enter a user name and password also. After the user confirms his information, the system will add the new user. If the username entered is already in use, then the use will be asked to submit a different username.

```
USER REGISTRATION SCREEN
*********************************************************************************
Welcome to the VSU Airline System

User Registration
---------------------------------

Name:
Address:
City:
State:
Zip Code:
Phone Number:
E-mail address:

Please enter in a unique username and password to log into the system.
Username:
Password:
Verify Password:

Please review User Information and Confirm.
Is the information correct? (Y/N):

Thank you for registering.  (if information is correct)
```

```
(S)earch (L)ogin (R)egister (Q)uit

*********************************************************************************
```

Other message(s):

```
Please enter a different username:
```

2.4 Passenger User Interface

The passenger user is anyone that is registered with the system and not and employee or administrator. This user has the same search power as a guest user, but passengers can book flights, check reservations, and change their user information.

The book flight menu will display the flight results from the previous search and give the user the option to choose a flight to book. The user will select the corresponding number next to the flight they wish to book and the system will ask for confirmation of his selection.

```
BOOK FLIGHT
*********************************************************************************
Welcome to the VSU Airline System

Book Flight
---------------------------------

(1) [Flight Number], [Departure City] to [Destination City] on [Date] for [Price]

Examples:
(1) C14, from Atlanta, GA to Las Vegas on February 28, 2005 for $204

(2) A121, from Atlanta, GA to Las Vegas on February 28, 2005 for $251

(3) B52, from Atlanta, GA to Las Vegas on February 28, 2005 for $237

Choose the flight you want to book:

*********************************************************************************

You have chosen the following flight

A121, from Atlanta, GA to Las Vegas on February 28, 2005 for $251

Please confirm choice (Y/N):

Your flight has been booked. Thank you for choosing VSU Air.

(S)earch (L)ogout (B)ook Flight (R)eservation (P)rofile (Q)uit

*********************************************************************************
```

The profile screen will display any personal information about the user and also allow the user to change his information. After the information is displayed, the system will ask if the user wishes to change his information. If ‘Y’ is selected, then the user will be able to change his information line-by-line. If ‘N’ is selected, then is will return to the menu options.

```
USER PROFILE
********************************************************************************
Welcome to the VSU Airline System

User Profile
---------------------------------

Username: bcosby
Password: jellypudding

Status: Passenger

Name: Bill Cosby
Address: 800 Old School Road, Chicago, IL  31754
Phone Number: 345-092-3958
E-mail Address: bcosby@aol.com

Do you wish to update your profile (Y/N):

********************************************************************************

Choose the feature you wish to change by selecting the line number.
(1)Username: bcosby
(2)Password: jellypudding
(3)Name: Bill Cosby
(4)Address: 800 Old School Road, Chicago, IL  31754
(5)Phone Number: 345-092-3958
(6)E-mail Address: bcosby@aol.com

Line to change: 3
Enter new Name: William Cosby
Do you wish to change another line? (Y/N): N

Profile updated.

(S)earch (L)ogout (B)ook Flight (R)eservation (P)rofile (Q)uit

********************************************************************************
```

2.5 Employee User Interface

An Employee user is anyone that is working for the airline. These users have the same powers as the passenger user, but employees can also check work assignments. Employees are already in the system and do not register.

The work assignment screen displays information about when an employee is schedule to work. This information will include dates and flights.

```
WORK ASSIGNMENT
********************************************************************************
Welcome to the VSU Airline System

Work Assignment
---------------------------------

Name: John Doe
You are scheduled to work on the following days:
```

```
[Date] on flight [flight number]

2/22/2005 on flight F1

2/25/2005 on flight A561

(S)earch (L)ogout (B)ook Flight (R)eservation (P)rofile (W)ork Assignment (Q)uit

********************************************************************************
```

2.6 Administration User Interface

The administrator has the all the options as a regular passenger, plus the power to create, modify and delete flights and the ability to create, modify and delete work schedules.

The admin will choose ‘F’ for Flight Control and then choose ‘C’ to create a flight. The administrator’s flight creation screen will ask for a flight number, route information and a price to set for the flight.

```
(F)light Control: Create, Modify, Delete
Flight Control: (C)reate, (M)odify, (D)elete

ADMIN - CREATE FLIGHT
********************************************************************************
Welcome to the VSU Airline System

Flight Creator
---------------------------------

Flight Number:
Departure City:
Departure Date:
Destination City:
Price:
Flight Time:  *optional feature to be added later if time permits

(S)earch (L)ogout (B)ook Flight (R)eservation (P)rofile (Q)uit
(F)light Control: Create, Modify, Delete
(T)ime Schedules: Modify

********************************************************************************
```

The admin will choose ‘F’ for Flight Control and then choose ‘M’ to modify a flight. The administrator’s flight modify screen will ask for a flight number and display the airline flight information. The system will ask the administrator to choose which line to modify.

```
(F)light Control: Create, Modify, Delete
Flight Control: (C)reate, (M)odify, (D)elete

ADMIN - MODIFY FLIGHT
********************************************************************************
Welcome to the VSU Airline System

Flight Modifier
---------------------------------
```

```
Enter a flight number to modify: F1
Flight Number: F1
(1)Departure City: Atlanta, GA
(2)Departure Date: 04/28/2005
(3)Destination City: Las Vegas
(4)Price: $251

Choose the line to modify:
Enter New Departure City:
Enter New Departure Date:
Enter New Destination City:
Enter New Price:

(S)earch (L)ogout (B)ook Flight (R)eservation (P)rofile (Q)uit
(F)light Control: Create, Modify, Delete
(T)ime Schedules: Modify

********************************************************************************
```

The admin will choose ‘F’ for Flight Control and then choose ‘D’ to delete a flight. The administrator’s flight deletion screen will ask for a flight number and display the flight information. It will then as for confirmation to delete the flight.

```
(F)light Control: Create, Modify, Delete
Flight Control: (C)reate, (M)odify, (D)elete

ADMIN - DELETE FLIGHT
********************************************************************************
Welcome to the VSU Airline System

Flight Deletion
--------------------------------

Enter a flight number to delete: F1
Flight Number: F1
Departure City: Atlanta, GA
Departure Date: 04/28/2005
Destination City: Las Vegas
Price: $251

Do you wish to delete this flight? (Y/N):

(S)earch (L)ogout (B)ook Flight (R)eservation (P)rofile (Q)uit
(F)light Control: Create, Modify, Delete
(T)ime Schedules: Modify

********************************************************************************
```

The admin will choose ‘C’ for Crew Schedules and then choose ‘M’ to modify a schedule. The administrator’s flight creation screen will ask for a flight number, route information and a price to set for the flight.

```
(C)rew Schedules: Create, Modify, Delete
Crew Schedules: (C)reate, (M)odify, (D)elete
```

```
ADMIN - MODIFY CREW SCHEDULE
********************************************************************************
Welcome to the VSU Airline System

Modify Crew Schedule
--------------------------------

Which Employee do you want to reassign? John Doe

(Display John Doe's Schedule)

What Day to you want to change? 04/28/2005

(From here, the admin will change the employee schedule for that day.)

(S)earch (L)ogout (B)ook Flight (R)eservation (P)rofile (Q)uit
(F)light Control: Create, Modify, Delete
(T)ime Schedules: Modify


********************************************************************************
```

3.0 Database Design

The database will store information on users, employees, and administrators ranging from personal information about the individual users to the flights they have booked. The database will also contain information on various flights available for booking and reservations that have been made by the users. The database will be located on the server side. Text files have been chosen for this project due to the simplicity of the data storage. There are three separate database files: user, flights, and reservations. The information in these files will be loaded at the runtime of the server and then re-saved when the server is shutdown.

3.1 USERS Database

The user database information will be store in the "USERS.TXT" text file. Semicolons will be used as delimiters.

The elements of the user database are as follows:

- Username – the username used by the passenger, employee or admin
- Password – the password
- userStatus – defines the type of user that is logging into the system
    - P for Passenger
    - E for Employee
    - A for Administrator
- Name – name of the person
- Address – Permanent place of residence
- Phone Number – Current home number
- E-mail address – most frequently used e-mail address

```
USERS.TXT
username;password;userStatus;Name;Address;Phone Number;E-mail Address

bbthorton;hollywoodman;P;Bill Bob Thorton;902 Hollywood Lane, Los Angeles, CA
90210;350-293-4747;bbthorton@yahoo.com
bcosby;jellypudding;P;Bill Cosby;800 Old School Road, Chicago, IL  31754; 345-092-
3958;bcosby@aol.com
jdoe;iamlost;A;John Doe;1800 Lost World Drive, Tampa, FL  46468; 921-438-
9511;jdoe@yahoo.com
```

3.2 FLIGHTS Database

The flights database information will be store in the "FLIGHTS.TXT" text file. This file contains all the information for each flight except the people on it. Semicolons will be used as delimiters.

The elements of the user database are as follows:

- flightNumber – the flight number of the flight from point A to point B
- deptCity – the departure city of the flight
- deptDate – the departure date of the flight
- destCity – the destination city of the flight
- price – the price of the flight

```
FLIGHTS.TXT
flightNumber;deptCity;deptDate;destCity;price

F1;Atlanta, GA;04/28/2005;Las Vegas, NO;251
A251;Atlanta, GA;04/28/2005;Las Vegas, NO;204
B2;Atlanta, GA;04/28/2005;Las Vegas, NO;238
```

3.3 RESERVATIONS Database

The reservations database information will be store in the "RESERVATIONS.TXT" text file. This file contains information on a flight and all the passengers on that flight. Semicolons will be used as delimiters.

The elements of the user database are as follows:

- flightNumber – the flight number of the flight from point A to point B
- username – username of the passengers of this flight
- resNumber- the reservation code

```
RESERVATIONS.TXT
resNumber;flightNumber;username;username;username

R001;F1;Atlanta, GA;04/28/2005;Las Vegas, NO;251
R002;A251;Atlanta, GA;04/28/2005;Las Vegas, NO;204
R003;B2;Atlanta, GA;04/28/2005;Las Vegas, NO;238
```

3.4 WORKASSIGNMENT Database

The work assignments database information will be store in the "WORKASSIGNMENT.TXT" text file. This file contains information on who is working on a flight and what day they are working. Semicolons will be used as delimiters.

The elements of the user database are as follows:

- username – the person being assigned work
- flightNumber – the flight they are working on
- date – the date they are working the flight previously noted

Employees could have multi work assignments next to their name, but will not have multiple assignments for a given day.

```
WORKASSIGNMENT.TXT
username;flightNumber;date;flightNumber;date

janedoe;F1;05/28/2005;A251;04/28/2005
```

4.0 Protocol Design

Available on all interfaces:

S – The client recognizes this command as "Search". It sends the message to the server. The server searches the data structure and returns the results to the client which then displays them on the console.

R – The client recognizes this command as "Register". It gets the pieces of information needed one by one; Name, Address, Phone number, on the client side and concatenates this information and sends it to the server, which adds this information to the user's data.

L - The client recognizes this command as "Login". The client gets the username and password and sends them to the server. The server runs a password checker. If the user is already logged in, the server will send that message to the client which will inform the user that they are already logged in.

O - The client recognizes this command as "Logout". The client gets the username and password and sends them to the server. The server removes the user's name from the logged in list.

Q – The client recognizes this command as "Quit". It then informs the server that it is quitting and disconnects.

Available on only the interface to registered users:

B– The client recognizes this command as "Book". It can only be accepted if the user is registered in the system. Therefore, the server checks the system for the user information and if it is not available, the person is prompted to register. If the person is registered, they are prompted to enter the flight they want to book and the booking is added by the server to both the person's information and the flight information.

P– The client recognizes this command as "Profile". The client prompts for the username and the server checks the system for the user information and if it is not available, the person is prompted to register. If the person is registered, the server returns the person's information and the person is prompted to update.

U– The client recognizes this command as “Update”. It is displayed by the client after the user accesses their profile. They can either enter Y or N and if Y is entered, then the new information is written back to the database through the server. If N is entered, the user is taken back to the previous options.

Available on the interface to Employees:

W– The client recognizes this command as “Work Assignments”. It returns the schedule from the server that is only available to be viewed.

V- The client recognizes this command as “Vacation”. It is available to employees to view how many sick and vacation days they have left. They can not use the system, however, to report a future absence or schedule days off.

Available on interface to Administrators:

F- The client recognizes this command as “Flight”. The client sends a message to the server, which sends back all available flights that are then displayed for the administrator. The administrator can then choose from a menu of **C, E**, or **D** in order to create, edit, or delete.

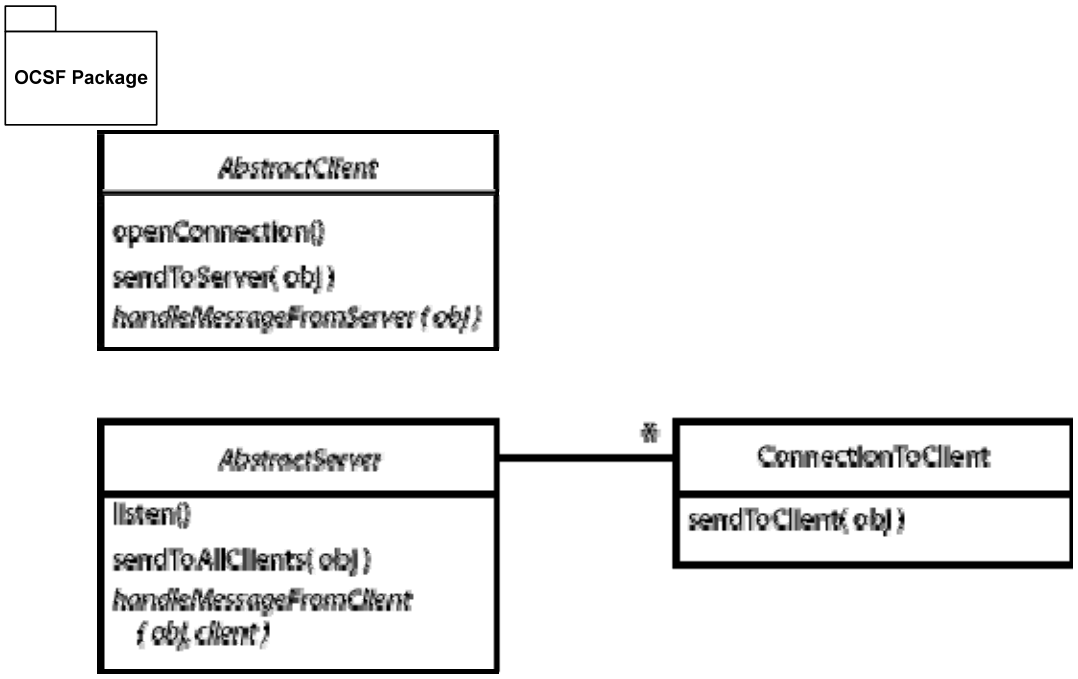
P - The client recognizes this command as “Set Price”. This is presented to the administrator in the Flight menu as well.

T- The client recognizes this command as “Time Schedule” for workers. It sends a message to the server which in turn sends back the time schedule for all employees. This also presents a menu where the administrator is requested to choose **M**, if he wants to modify the schedule.

5.0 Architectural Design

This diagram shows how the OCSF framework interacts with the airline classes. The AbstractServer, AbstractClient, and ConnectionToClient come within the OCSF package. These classes are then inherited to create the airline system in which the server sits on one side, and the clients have access to this server and can interact with it through client interfaces. The ClientIF class will be inherited to present user interfaces that will be available to different types of users of the system.

5.1 The **OCSF Package** has the following classes and this diagram shows just the **most important methods.**



6.0 Class Design

Structure

The classes will be designed to either work on the client side or on the server side. The AirlineServer inherits functionality from the AbstractServer and adds the attributes and methods needed to work with the airline clients. Most of the work will be done on the server. The AirlineClient class is a very thin client, acting as a mere channel of correspondence between the clients and the server. It mostly sends and receives messages from the server and also sends and receives from the client consoles, using the method.

Data

The data will be loaded into the AirlineServer's data structures at run time. It will have linked lists for flight, the schedule, reservations, and users. These data structures store objects created from the database text files shown above. Clients will also be able to add, delete or modify the information depending on what authority they have.

Runtime

At run time, the server will be started up with theAirlineServer which has a main and the ClientConsole, which also has a main. The ClientConsole class will obtain the username and password of the person logged in and determine whether they are a customer, employee, or administrator from the users list on the server. It will then use the given information to display a ClientConsole that is either a CustomerConsole, EmployeeConsole, or AdminConsole. If the user does not exist in the system, they will be prompted to register.

If they are registered with the system they will be able to perform commands like searching, booking flights and more.

Navigation

The clients will only be able to view what is available on their console. They will navigate using the commands and have the option to go back to the previous screen. The commands available on the screen are dependent on the type of person logged in. The client will determine what kind of console to display using the authority() method. The display() method will be inherited by all consoles from the ClientConsole and implemented differently.

6.1 Class Diagrams

The diagram below shows the methods and attributes in the client and the server parts of the system. The AbstractServer and AbstractClient classes have more methods than the ones shown in the diagram. The diagram simply shows the key methods in the ocsf classes. The ClientConsole class is associated with the EmployeeConsole, AdminConsole and CustomerConsole classes and will provide them functionality as well. It contains the main method, and starts at runtime. When started, it determines from the client what kind of user has logged in and displays the appropriate console.

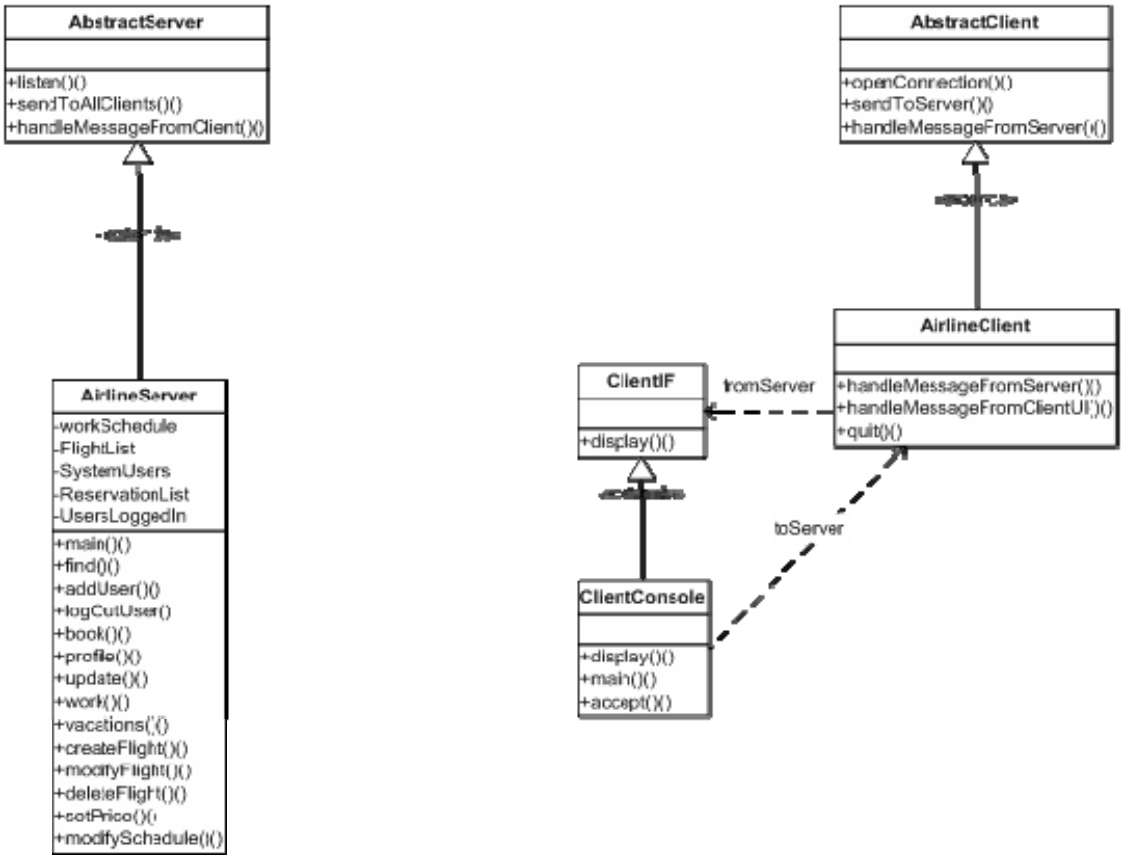
The Server Side:

The AirlineServer will load up all data at run time and store the data from the text files into data structures so that the data is available in memory. The classes associated with the AirlineServer class will enable it to store objects in the data structures like flights, persons, bookings, which will encapsulate information about each entity and be easier to access.

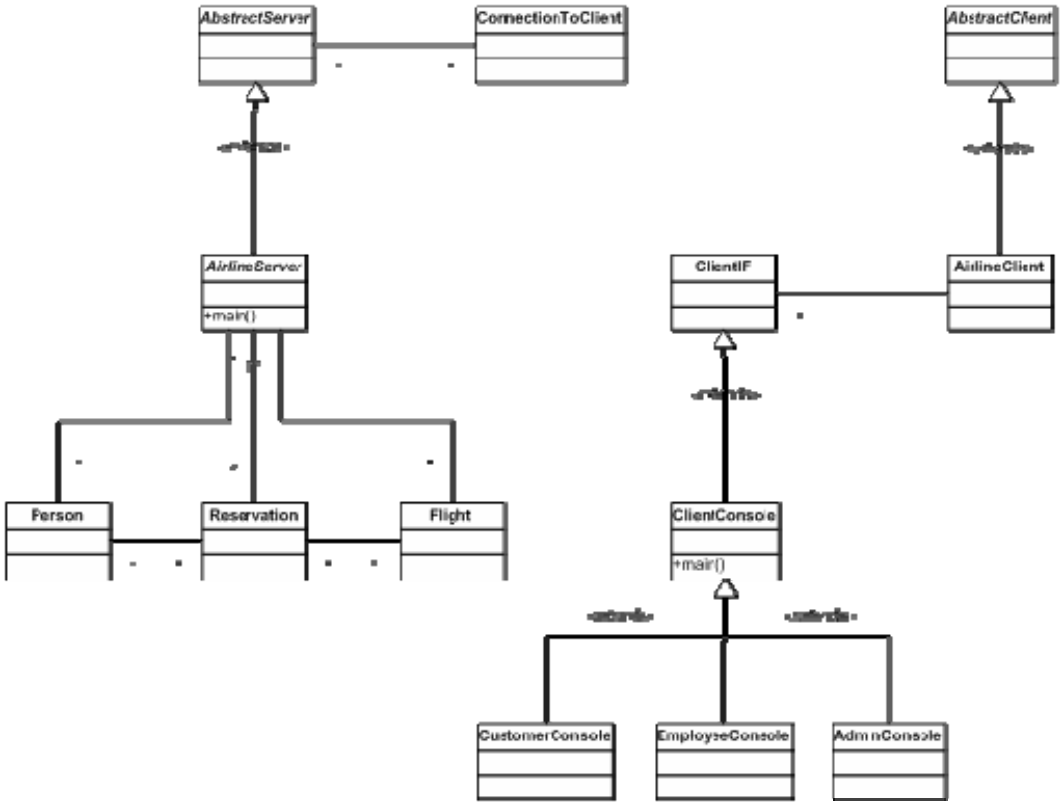
If an object, for example a flight is requested by a client, the Airline server will toString() that object and send it to the client as it's string representation and the client interfaces will handle what and how the information is displayed.

The client Side:

The AirlineClient will communicate with the AirlineServer to enable users to access the data on the server as well as send data to it. The client will be able to perform some functions beyond the AbstractClient. It will handle requests made by users and convert them into commands that the server can recognize and execute, and it will have some methods as well, but it will not store any data.


AbstractServer
+listen()()
+sendToAllClients()()
+handleMessageFromClient()()
AirlineServer
-workSchedule
-FlightList
-SystemUsers
-ReservationList
-UsersLoggedIn
+main()()
+find()()
+addUser()()
+logOutUser()
+book()()
+profile()()
+update()()
+work()()
+vacations()()
+createFlight()()
+modifyFlight()()
+deleteFlight()()
+setPrice()()
+modifySchedule()()
AbstractClient
+openConnection()()
+sendToServer()()
+handleMessageFromServer()()
AirlineClient
+handleMessageFromServer()()
+handleMessageFromClientUI()()
+quit()()
ClientIF
+display()()
fromServer
ClientConsole
+display()()
+main()()
+accept()()
toServer

The following diagram shows all the classes in the system, but without attributes and methods listed. It shows how all the airline system classes interact with the client and server structure.



6.2 Class Descriptions

Classes defined:

6.2.1. AbstractClient(Not listed)
6.2.2. AbstractServer(Not listed)
6.2.3. AdminConsole
6.2.4. AirlineClient
6.2.5. AirlineServer
6.2.6. ConnectionToClient(Not listed)
6.2.7. ClientConsole
6.2.8. CustomerConsole
6.2.9. Day
6.2.10. EmployeeConsole
6.2.11. Flight
6.2.12. Person
6.2.13. Reservation

6.2.1-AbstractClient, 6.2.2-AbstractServer, 6.2.5-ConnectionToClient, are from the OCSF framework and none of the attributes or methods will be changed in the airline system. To avoid packing the document with so much, their methods and attributes will not been listed.

The 'getter' and 'setter' methods in all the classes have also not been listed either because almost all the attributes will have them. They will be used to access the attributes in order to maintain data integrity.

To avoid further repetition, the methods and attributes of child classes that are inherited from parent classes are not listed as well. Each class has a description of the class(es) from which it inherits and therefore, is assumed to have all the attributes and methods from that class in addition to its own.

6.2.3 Class: AdminConsole

Description: This class will inherit all its methods and attributes from the ClientConsole class. It will be called by the ClientConsole class to display the administrator interface. The only method that will be different is the display method.

Attributes:

No attributes yet

Methods:

Name	Scope	Return Type	Description	Arguments	Algorithm
display	public	string	This method will show the options available to an administrator on the console.	None	It will have different conditional statements to determine where the administrator is in the system and what to display.

6.2.4 Class: AirlineClient

Description: This class will inherit all its methods and attributes from the AbstractClient class. It will be called by the ClientConsole to send and receive information from the server. It will only implement 4 methods from the AbstractClient, handleMessageFromClientUI, handleMessageFromServer, level and quit. Authority is a security feature that ensures that only authorized users access certain information

Attributes:

No attributes yet

Methods:

Name	Scope	Return Type	Description	Arguments	Algorithm
handleMessageFromClientUI	public	none	This method will be used	string	It will parse the entered

			to read messages from the UI and send them to the server unless they are client messages		messages to determine what kind of command they are and let the server know what to do.
handleMessageFromServer	public	none	This method will be used to read messages from the server and send them to the clients.	string	It will send the message to the UIs.
authority(level l)	private	string	Makes sure the user is authorized to execute a command	level	Checks the Level attribute of the console and returns a string that says whether or not the client is allowed to access that information
Quit	public	none	It closes the client	none	this method will communicate to the server about logging the client off first and then close the client

6.2.5 Class: AirlineServer

Description: This class will inherit all its methods and attributes from the AbstractServer class. It will execute all the requests from clients and have data structures to store all client data. It will receive and send messages to the client.

Attributes:

Name	Scope	Type	Description
WorkSchedule	public	listMap(tentatively)	This will be a data structure to store all the working **Days**

			and who is scheduled to work on those **Days.**
FlightList	public	listMap(tentatively)	This will be a data structure to store all the **Flight** objects that are available.
UsersList	public	listMap(tentatively)	This will be a data structure to store all the **Person** objects that are available.
ReservationList	public	listMap(tentatively)	This will be a data structure to store all the **Reservation** objects that are available.
LoggedInList	public	listMap(tentatively)	This will be a data structure to store all the **Person** objects that are logged into the system at any one moment

Methods:

Name	**Scope**	**Return Type**	**Description**	**Arguments**	**Algorithm**
main()	public	none	This method will execute at run time and create a new AirlineServer	String []args	Not yet written.
find()	public	none	This method will be used to search FlightList for a specified flight	string	It will search the data list and send the found Flight back to the client or nothing.
addUser(Person p)	public	none	Adds a Person to the UsersList and	Person	Appends the Person object to the end of the UsersList data

			brands them as a customer.		structure
logInUser(Person p)	public	none	Adds a Person to the LoggedInList in order to know who is logged in.	Person	Appends the Person object to the end of the LoggendInList data structure
logOutUser(Person p)	public	none	Removes a Person from the LoggedInList	Person	Removes the person from the LoggedInList
book(Person p, Flight f)	public	String	This method books is used to book someone on a flight	Person p, Flight f	It takes a **Flight** and **Person** objects and creates a **Reservation** that it adds to the ReservationList. Then it updates the Person information by adding this Reservation the Person's Reservations arrayList
printProfile(Person p)	public	String	Returns a Person's profile	Person	It creates a string out of the Person's profile by calling the Person.toString() method
update(String attribute, String msg)	public	none	Updates a person's information.	String attribute, String msg	It takes the attribute string and makes Person."attribute" = msg
work()	public	string	Returns the entire work schedule	None	Creates a string from the WorkSchedule to display.
vac(Person p)	public	string	Returns the person's available day for vacation.	None	Creates a string from the Person's vacation information. This is obtained from the Array, VacationDays that is in the Person class. Not applicable to Customers
createFlight (String flightNumber, String	public	None	It creates a flight from	String	Flight f1 = new Flight(flightNumber,

deptCity, String deptDate, String destCity, String Price)			given information.		deptCity, deptDate, destCity, Price).
Quit	public	none	It closes the client	none	this method will communicate to the clients about closing and then close the server

6.2.6 Class: ClientConsole

Description: This class will start at runtime and will provide an interface with which clients can communicate with the server. It will be inherited by the AdminConsole, EmployeeConsole, and CustomerConsole classes. The only method that will be overridden in those classes is the display method.

Attributes:

No attributes yet

Methods:

Name	Scope	Return Type	Description	Arguments	Algorithm
main()	public	none	This method is responsible for the creation of the Client UI.	String[] args	The algorithm will be created at implementation time.
display()	public	string	Method that when overriden in child classes is used to display objects onto a UI.	String message	The display method will be used by the client to display messages on the console
accept()	public	none	This method reads input from the user.		It will wait for messages from the user

6.2.7 Class: CustomerConsole

Description: This class will inherit all its methods and attributes from the ClientConsole class. It will be called by the ClientConsole class to display the customer interface. The only method that will be different is the display method.

Attributes:

No attributes yet

Methods:

Name	Scope	Return Type	Description	Arguments	Algorithm
display()	public	string	This method	None	It will have

			will show the options available to an customer on the console.		different conditional statements to determine where the customer is in the system and what to display.

6.2.8 Class: Day

Description: This class will encapsulate a date. It will be used in order to store day objects in the WorkSchedule linked list so that the schedule can be accessed a lot easier.

Attributes:

Name	Scope	Type	Description
Date	public	string	This will be a string attribute containing the date.
dayFlights	public	Array of strings	It will be an array containing flight numbers
Crew	public	Array of strings	This will be an array of crew members and it can be a list with names or the employee's user-IDs

Methods:

Name	Scope	Return Type	Description	Arguments	Algorithm
toString()	public	string	This method will show the date.	None	It will display the date object as a string in the format mm/dd/yyyy.

6.2.9 Class: EmployeeConsole

Description: This class will inherit all its methods and attributes from the ClientConsole class. It will be called by the ClientConsole class to display the customer interface. The only method that will be different is the display method.

Attributes:

No attributes yet

Methods:

Name	Scope	Return Type	Description	Arguments	Algorithm
display()	public	string	This method will show the options available to an employee on the console.	None	It will have different conditional statements to determine where the employee is in the system and what to display.

6.2.10 Class: Flight

Description: This class provides functions that will be inherited by every flight object.

Attributes:

Name	Scope	Type	Description
flightNumber	public	string	The flight number will be a 3 digit number followed by a letter. For example 112A and we will consider it as a string.
deptCity	public	string	This will have the information about where the flight is departing from.
deptDate	public	string	Includes information about the date on which the flight is departing.
destCity	public	string	This will have the information about where the flight is arriving.
price	public	string	The price set by the administrator for this flight.
Reservations	public	arrayList	This arrayList will hold the reservation numbers that are

			linked to this flight

Methods:

Name	Scope	Return Type	Description	Arguments	Algorithm
setPrice	public	string	Sets the price to the argument passed in.	float	Gets the given float value and makes this.price equal to that value.
toString	public	string	Returns a print out of the flight information as listed in the flights.txt file.	None	Concatenate all the information about the flight from the txt file or data structure for display

6.2.11 Class: Passenger

Description: This class will represent a customer that is already registered with the system.

Attributes:

Name	Scope	Type	Description
Flights	public	array	This will be an array containing flights that the person is booked on
userName	public	string	This will be a string containing the person's userID
resNumber	public	string	This will contain some kind of number to identify the reservation with.

Methods:

Name	Scope	Return Type	Description	Arguments	Algorithm
toString	public	string	Returns a print out that says something like this is booking number x for	None	Concatenate all the information about the reservation from the txt file

			passenger y on flight z.		or data structure for display

6.2.12 Class: Person

Description: The person will provide functionality for the Employee, Admin and Passenger entities. It holds the basic functions and attributes that any of the clients will be able to perform. The person will have a status that will identify them to the system.

Attributes:

Name	Scope	Type	Description
Username	public	string	The username, will be a string that the object carries in order to identify the user in the system.
password	public	string	The password will be a maximum string of 8 digits and minimum of 6.
Status	public	string	The user status will be a letter to symbolize what type of user it is.
Name	public	string	This will hold the person's name and can be a string of any number of characters.
Address	public	string	This again will be a string containing the person's full address which includes street address, city, state, and zip code.
Phone	public	string	This will be a string containing 10 characters without spaces or hyphens to represent the person's phone number.
email	public	string	This will be a

			string containing the person's email address. It will have to be in the format *@* (1 or more characters before the @ sign, and 1 or more after)
Reservations	public	ArrayList	This will be an ArrayList to store all the reservations **this** Person has made.
VacationDays	public	Array	This will be an array containing a Person's days off. This will be empty for a customer.

Methods available to every Person:

Name	Scope	Return Type	Description	Arguments	Algorithm
search	public	string	Used to search for a flight in the database	string deptCity, string date, string city	It sends the message to the server. The server searches the data structure and returns the results to the client which then displays them on the console.
login	public	none	Logs the person into the system	string username, string password	The client gets the username and password and sends the information to the server. The server runs a password checker. If the user is already logged in, the server will send that message to the client.
logout	public	none	Logs the person out of the system	none	This disconnects the client from the server.
register	public	none	Adds this user's information to the system data	string name, string address, string phone, string email, string userName, string password	Uses the data provided in the arguments to add the user to the users database. This information will be added as a string

quit	public	none	Completely disconnects the client and exits	none	It makes the client exit the system.
toString	public	string	Returns a print out that holds all the information about the client	None	Concatenate all the information about the Person from the UsersList data structure for display.

Methods available to a Person with status 'E' and 'A':

logout	public	none	Logs the person out of the system	none	This disconnects the client from the server.
register	public	none	Adds this user's information to the system data	string name, string address, string phone, string email, string userName, string password	Uses the data provided in the arguments to add the user to the users database. This information will be added as a string
quit	public	none	Completely disconnects the client and exits	none	It makes the client exit the system.

6.2.13 Class: Reservation

Description: This class will represent a flight booking that will contain information about both the flight and the passenger.

Attributes:

Name	Scope	Type	Description
flightNumber	public	string	The flight number will be a 3 digit number followed by a letter. For example 112A and we will consider it as a string.
userName	public	string	This will be a string containing the person's userID

resNumber	public	string	This will contain some kind of number to identify the reservation with.

Methods:

Name	Scope	Return Type	Description	Arguments	Algorithm
toString	public	string	Returns a print out that says something like this is booking number x for passenger y on flight z.	None	Concatenate all the information about the reservation from the txt file or data structure for display

7.0 Glossary

- Administrator – Person that control almost every aspect of the system
- Airline System – the entire collection of clients and the server defined in this document
- Client – Handles the interaction between the system and the user
- Employee – Person employed by the airline system; this person does not register with system like the passenger
- Flight – Airline flight from point A to point B
- Guest – Anyone that is not logged into or registered with the system
- Passenger – A person that is registered with the system
- Server – This part of the system manages the database and communication of the clients
- User Interface – Provides a visual means by which the user can easily interact with the system

Appendix A: Document Control

Document Revision History

Version	Release Date	Revised By	Revision Description
1.0	January 25, 2004	Sarah Tshila TJ Dorsey	Initial version
1.1	February 24, 2004	Sarah Tshila TJ Dorsey	Final version
1.2	March 6, 2004	Sarah Tshila TJ Dorsey	Revised Final version

Document Owner

Sarah Tshila and TJ Dorsey

Appendix B: Change Control Request Form

Change Request Form

Project Name:	VSU Air Airline System	**Original Delivery Date:**		**Status:**	
Request Author:		**Change Order Date:**			
Owner:		**Date Resolved:**			

Change Request Description (to be filled out by request author)

Resolution Description (to be filled out by Project Manager)

Activity Log

Date	Name	Notes