***Project Design - Data Dictionary***

***Airport – page 2***

***Route – page 3***

***Aircraft – page 4***

***Schedule – page 5***

***Flight – page 6***

***Customer – page 7***

***TravelAgent, ServiceItem – page 10***

***FlightService – page 11***

***Booking – page 12***

***Staff – page 13***

***ProfileManager , FlightManager – page 14***

***BookingManager, ServiceManager – page 15***

***Databse – page 16***

***ProfileSystem – page 20***

***ReservationSystem – page 21***

***ReportSystem, ServiceSystem – page 22***

***FlightSystem – page 23***

***Class Name:*** Airport

***Superclass:*** None

***Attributes:***

**Id :** A number that Uniquely identifies an airport

**Name :** Alphanumeric

**City :** Alphanumeric

**Country :** Alphanumeric

**IATA/FAA :** Alphanumeric, an abbreviation of name.

**Latitude :** A Decimal number representing latitude

**Longitude:** A decimal number representing longitude.

**Altitude:** A number representing altitude in meters.

**Timezone:** A number representing timezone related to GMT time. Example this value is 12 for NZ. This represents GMT+12.

**DST:** A character that represents daylight saving time

**Tz:** Alphanumeric, represents timezone from the tz database.

**Methods:**

**getID():** This method allows access to an airports ID.

**getName():** This method allows access to an airports name.

**getCity():** This method allows access to an airports City.

**getIATA():** This method allow access to an airports IATA code.

**getLat():** This method allows access to an airports Latitude.

**getLong():** This method allows access to an airports Longitude.

**getAlt():** This method allows access to an airports Altitude.

**getTimeZone():** This Method allows access to an airports time zone(GMT).

**getDST():** This method allows access to an airports Daylight saving code.

**getTZ():** This method allows access to an airports TZ database time zone.

**getByIATA():**  This method will return a single airport based on its IATA code.

**getByName():**  This method will return a single airport based on an airport name.

**setID(id):** This method allows us to set an airports ID.

**setName(name):** This method allows us to set an airports Name.

**setCity(city):** This method allows us to set an airports City.

**setIATA(IATA code):** This method allows us to set an airports IATA/FAA abbreviation code.

**setLat(latitude):** This method allows us to set an airports latitude.

**setLong(longitude):** This method allows us to set an airports Longitude.

**setAlt(altitude):** This method allows us to set an airports altitude.

**setTimeZone(amt\_hours):** This method allows us to set an airports time zone(GMT)

**setDST(DST\_code):** This method allows us to set an airports daylight saving code.

**setTZ(timezone\_name):**This method allows us to set an airports TZdatabase time zone.

**CreateAirport():** This method will insert a row into the database with this objects information.

**DeleteAirport():** This method removes the airport data from the databased based on an ID.

**AlphabeticList()** This method will display an alphabetic list of available airports in the database.

***Class Name:* Route**

***Superclass:*** None

***Attributes:***

ID: A number that uniquely identifies a route.

SourceAirport: An IATA/FAA code that corresponds to an airport

DestinationAirport: An IATA/FAA code that corresponds to an airport

codeshare: A character that can be either ‘NULL’ or ’Y’. Defines whether codesharing occurs on this route.

stops: A number representing the number of stops on this route.

***Methods:***

getID(): This method allows access to a routes ID.

getSrc(): This method allows access to a routes source airport IATA/FAA code.

getDest(): This method allows access to a routes destination airport IATA/FAA code.

getCodeShare(): This method allows access to a routes codeshare status.

getStops(): This method allows access to the amount of stops on a route.

setID(id): This method allows us to set the ID of a route.

setSrc(IATA\_code): This method allows us to set the source airport IATA/FAA code.

setDest(IATA\_code): This method allows us to set the destination airport IATA/FAA code.

setCodeShare(codeshare\_status):This method allows us to set the codeshare status for a route.

setStops(): This method allows us to set the amount of stops on a route.

createRoute(): This method adds the data about a Route to the database.

updateRoute(): This method modifies the data about a route in the database.

deleteRoute(): This method deletes information about a route from the database.

IsInternational(): This method compares the 2 airports associated with this route and identifies whether the route travels internationally.

***Class Name:* Aircraft**

***Superclass:*** None

***Attributes:***

ID: A number used to uniquely identify an aircraft.

name: Alphanumeric. The name used for the type of aircraft.

inService: A number representing how many of this type of aircraft are in service.

fClass: A number representing how many first class seats are on this aircraft.

bClass: A number representing how many busi ness class seats are on this aircraft.

peClass: A number representing how many premium economy seats are on this aircraft.

eClass: A number representing how many economy seats are on this aircraft.

total: A number representing the total number of seats available on the aircraft.

***Methods:***

getID(): This method allows access to an aircrafts ID.

getName(): This method allows access to an aircrafts type.

getInService(): This method allows access to the amount of aircrafts of this type that are in service

getFClass(): This methos allows access to the amount of first class seats on this aircraft.

getBClass(): This method allows access to the amount of business class seats on this aircraft.

getPEClass(): This method allows access to the amount of premium economy class seats on this aircraft.

getEClass(): This method allows access to the amount of economy class seats on this aircraft.

getTotal(): This allows access to the total amount of seats on this aircraft.

setID(id): This method allows us to set an aircrafts ID.

setName(name): This method allows us to set an aircrafts name.

setInService(numOfAircraft): This method allows us to set the amount of aircraft of this type that are in service.

setFClass(fName): This method allows us to set the amount of first class seats on this aircraft

setBClass(lName): This method allows us to set the amount of business class seats on this aircraft.

setPEClass(numOfseats): This method allows us to set the amount of premium economy class seats on this aircraft.

setEClass(numOfSeats): This method allows us to set the amount of economy class seats that are on this aircraft.

setTotal(): This method calculates and sets the total number of seats on this aircraft by adding together the values in fClass,bClass,peClass,and eClass.

createAircraft(): This method will add this information about the aircraft to the database.

deleteAircraft(): This method will remove an aircraft from the database based on an ID.

getSClassStartPoint(): This method will return a number between 0-5 that refers the start point of the seat class.

Round(): This method rounds a number up or down depending on the decimal value. A utility function for getSClassStartPoint().

***Class:* Schedule**

***Superclass:*** None

***Attributes:***

ID: A number used to uniquely identify a schedule in the database.

flightID: Alphanumeric. A string used to uniquely identify a Flight.

Plane: A number that identifies the type of plane to be used for this schedule. This number refers to a plane in the Aircraft database.

Route: A number that identifies the route to be used in this schedule. This number refers to a route in the route database.

departDay: A string representing the day of departure (eg “Thu”).

departDate: A string representing the date and time of departure (eg ‘2015-01-01 00:00:00’).

departTimezone: A string representing the timezone of departure.

arriveDay: A string representing the day of arrival

arrive: A string representing the date of arrival.

arriveTimezone: A string representing the timezone of arrival.

***Methods:***

getID(): This method allows us access to a schedules database ID.

getFlightID(): This method allows us access to the Flight ID.

getPlane(): This method allows us to access the plane type used in a schedule.

getRoute(): This method allows us to access the route used in a schedule.

getDeparture(): This method allows us to access the departure date/time of the schedule.

getArrival(): This method allows us access the arrival date/time of the schedule.

setID(): This method allows us to set a schedules database ID.

setFlightID(): This method allows us to set a schedules flight ID.

setPlane(): This method allows us to set a schedules plane type.

setRoute(): This method allows us to set a schedules route.

setDeparture(): This method allows us to set the schedules departure date/time.

setArrival(): This method allows us to set the schedules arrival date/time

createSchedule(): This method adds information about a schedule to the database

updateSchedule(): This method modifies information about an existing schedule in the database.

deleteSchedule(): This method deletes a schedule from the database.

**Class Name: Customer**

***Superclass:*** None

***Attributes:***

ID: A number used to uniquely define a person

Title: Alphanumeric.

fName: Alphanumeric. Represents a persons first name.

lName: Alphanumeric. Represents a persons last name.

gender: A character that represents a persons gender. Can be either ‘m’ or ‘f’.

DOB: Instantiation of a ‘Date’ class. Represents a persons date of birth.

Phone: Alphanumeric. Represents a persons contact phone number.

Email: String of characters that represent a persons email address.

Address: Alphanumeric. Represents a persons street address.

State: Alphabetic. Represents which state a person resides in.

Country: Alphabetic. Represents which country a person resides.

CardType: Alphanumeric. Represents what type of card the customer has.

CardNum: Numeric. A Number that represents the customers card number.

freqFlierPoints: Numeric. Represents the number of frequent flier points a customer has accredited to them.

Passport: Either ‘TRUE’ or ‘FALSE’. Represents whether a customer has a passport.

Status: Either ‘NULL’, ‘no fly’, or ‘watch’. Represents whether a customer is allowed to book a flight based on previous flight behaviour.

TravelAgent: Alphanumeric. A String that corresponds to the name of a travel agent in the agents database.

***Methods:***

getID(): This method allows access to a customers ID.

getTitle(): This method allows access to a customers title.

get FName(): This method allows access to a customers first name.

getLName(): This method allows access to a customers last name.

getGender(): This method allows access to a customers gender.

getDOB(): This method allows access to a customers date of birth.

getPhone(): This method allows access to a customers phone number.

getEmail(): This method allows access to a customers email Address.

getAddress(): This method allows access to a customers street address.

getState(): This method allows access to a customers state of residence.

getCountry(): This method allows access to a customers country of residence.

getCardType(): This method allows access to a customers credit card type.

getCardNum(): This method allows access to a customers credit card number.

getfreqFly(): This method allows access to the amount of frequent flier points accredited to the customer.

getPassport(): This method allows access to a customers passport status.

getNoFly(): This method allows access to a customers fly status.

getAgent(): This method allows access to the name of a travel agent that the customer used to book a flight.

setID(id): This method allows us to set the ID of a customer.

setTitle(title): This method allows us to set the title of a customer.

setFName(fName): This method allows us to set the first name of a customer.

setLname(lName): This method allows us to set the last name of a customer.

setGender(gender): This method allows us to set the gender of a customer.

setDOB(date): This method allow us to set the date of birth of a customer.

setPhone(ph#): This method allows us to set the phone number of a customer.

setEmail(emailAddress): This method allows us to set the email address of a customer.

setAddress(stAddress): This method allows us to set the street address of a customer.

setState(state): This method allows us to set the state of residence of the customer.

setCountry(country): This method allows us to set the country of residence of the customer.

setCardType(cardType): This method allows us to set the credit card type of the customer.

setCardNum(cardNum): This method allows us to set the credit card number of the customer.

setFreqFly(freqFlyPts): This method allows us to set the frequent flier points for a customer.

setPassport(passportState): This method allows us to set the passport status of the customer.

setNoFly(status): This method allows us to set the no fly status of the customer.

setAgent(name): This method allows us to set the agency name that the customer used to book a flight.

getByEmail(email): This method gets a customer object from the database and allocates its attributes to this object.

Update() This method will update the information about this customer in the database based on an email.

deleteCust() This method will delete information about a customer from the database based on an email.

**Class Name: Travel Agent**

***Superclass:*** None

***Attributes:***

ID: A number used to uniquely identifies a Travel Agent

Name: Alphanumeric. Represents the name of the agency.

Phone: Numeric. Represents the travel agents contact phone number.

Email: Alphanumeric. Represents the travel agents contact email address.

***Methods:***

getID(): This method allows access to a travel agencies ID.

getName(): This method allows access to a travel agencies name.

getPhone(): This method allows access to a travel agencies phone number.

getEmail(): This method allows access to a travel agencies email address.

setID(id): This method allows us to set the ID of an agency.

setName(name): This method allows us to set the name of an agency.

setPhone(phone#): This method allows us to set the phone number of an agency.

setEmail(emailaddress): This method allows us to set the email address of an agency.`

setByEmail(email): This method allows us to retrieve information from the database about a travel agency based on an email.

Update() : This method updates an existing travel agent in the database.

**Class Name: ServiceItem**

***Superclass:*** None

***Attributes:***

ID: A number used to uniquely identify a service item

Item: Alphabetic. Represents a description/name of a service item.

Cost: Numeric. Represents the price of an item.

Availability: Either ‘All’ or ‘International’. Represents what type of flights this item is available on.

***Methods:***

getID(): This method allows access to a Service Items ID.

getItem(): This method allows access to a service items description.

getCost(): This method allows access to a service items price.

getAvail(): This method allows access to a service items availability.

setID(id): This method allows us to set the ID of a service Item.

setItem(desc): This method allows us to set the description of a service item.

setCost(price): This method allows us to set the price of a service item.

setAvail(availability): This method allows us to set the availability of a service item.

createServiceItem(): This method creates a serviceItem object in the database.

updateServiceItem(): this method modifies an existing serviceItem in the database.

deleteServiceItem(): This method deletes information about a serviceitem in the database.

displayAll(bool intl): This method displays all serviceItems in the database based on availability. True = international menu, false = national menu.

**Class: FlightService**

***Superclass:*** none.

***Attributes:***

ID: A number that uniquely identifies a FlightService in the databse.

scheduleID: A string that identifies the flight this service belongs to. Refers to a Flight class’s FlightID.

ServiceItemID: A number that identifies a ServiceItem. This refers to a ServiceItem class’s ID.

BookingID: A number that identifies the booking this flightService belongs to.

itemAmount: A number that identifies how many of the ServiceItem will be on the flight.

***Method:***

getID(): This method allows us to access a FlightServices ID.

getFlightID(): This method allows us access to a FlightService’s Flight ID.

getServiceItem(): This method allows us access to a FlightService’s service item.

getItemAmount(): This method allows us to access to the amount of serviceItem’s that will be on a flight.

setID(): This methos allows us to set the ID of a flight service object.

setFlightID(): This method allows us to set the flightID of the flight this FlightService belongs to.

setServiceItem(): This method allows us to set which item this flightService is for.

setItemAmount(): This method allows us to set the amount of ServiceItems that will be on the flight.

createFlightService(): This method creates a FlightService object in the database.

deleteFlightService(): This method deletes a flightService in the database based on an ID.

**Class: Booking**

***Superclass:*** none

***Attributes:***

***ID: A integer that uniquely identifies a booking.***

custEmail: A email address that uniquely identifies a customer.

scheduleID: A string that identifies the schedule this booking is flying on.

travelAgent: A string that uniquely identifies a travel agent in the database.

Price: A number that represents the price of this booking.

***Methods:***

getCustEmail(): This method allows us access to the Customers email.

getScheduleID(): This method allows us access to the bookings scheduleID.

getID(): This method allows us access to the booking ID

getTravelAgent(): This method allows us access to the travel agent that booked

getMRE(): Retrieves the most recently added booking ID from the database. A utility function used in makeBooking().

setServices(): This method allows us to set the customers requested services for this booking.

setPrice(): This method allows us to set the price of this booking.

**Class Name: Staff**

***Superclass:*** None

***Attributes:***

ID: A number used to uniquely identify a staff member.

Password: A sting of characters representing a staffs login password.

userType: A string that identifies the type of staff member the object represents. This can be either “Staff”,”ProfileManager”,”ServiceManager”,”FlightManager”, ”BookingManager” or “Admin”.

***Methods:***

getID(): This method allows us to access a staff members ID.

getPassword(): This method allows us to access a staff members password

setID(id): This method allows us to set the id of a staff member.

setPassword(password):This method allows us to set the password of a staff member.

setByEmail(): this method gets the staff data from the database.

**Class: ProfileManagerUI**

***Superclass:*** Staff

***Attributes:***

String userType: A string that identifies the user using this UI

***Methods:***

Run(): This method displays the main menu of the user interface

customerAccessMenu(): This method displays the menu to access customer profile functions.

**Class: FlightManagerUI**

***Superclass:*** Staff

***Attributes:***

String UserType: A sting of characters representing a the staff member type

***Methods:***

Run(): This method runs the main interface fot FlightManagerUI

accessAircraftMenu(): This method displays the menu for aircraft modifications.

accessAirportsMenu(): This method displays the menu for Airport modifications.

accessRoutesMenu(): This methos displays the menu for route modifications.

accessScheduleMenu() This methos displays the menu for the schedule modification.

airportWeatherMenu(): This method allows the flight manager access to the weather reporting system.

**Class: BookingManagerUI**

***Superclass:*** Staff

***Attributes:***

String UserType: A sting of characters representing a the staff member type

***Methods:***

Run(): this method runs the main BookingMAnager menu.

**Class: ServiceManagerUI**

***Superclass:*** Staff

***Attributes:***

String UserType: A sting of characters representing a the staff member type ***Methods:***

Run(): this method runs the main ServiceManagerUI menu.

serviceItemUI(): This method runs ther service item management UI.

**Class: BookingController**

Attributes:

String userType: represent the type of user.

Methods:

makeBooking(): This method starts the booking process

setType(): This method sets the userType.

getType(): This method returns the type of user.

displaySeating(): This method displays the available seating on a flight

searchSeatingArray(): This method searches the seating plan on a given schedule/seat class

viewCustomerBookings(): view all bookings for a customer.

ChooseSeating(): This method allows a user to choose their seating option for a flight.

chooseServices(): This method allows a user to choose their services for a flight.

Class: BookingManagerController

Attributes:

String userType: a string that identifies which member is using the controller.

Methods:

checkNoFly(): returns all persons on the no fly list.

bookingReport(): outputs all reports from booking system.

Class: CustomerProfileController

Attributes:

Sqlite3\* database: a reference to the database,

Methods:

findCustomer(): This method outputs a customers details based on their ID.

registerExistningCustomer(): This method is a handler for users who already exist on in the database but have not set their password yet.

createCustomer(): This method provides the interface and entity calls to add a customer to the database,

editCustomer(): This method handles the editing of a customer personal details.

deleteCustomer(): This method will remove a customer from the database.

**Class: CustomerUI**

Attributes:

String userType: represent the type of user.

String username: a string that represents the username of the user.

Methods:

run(): This method runs the main menu for CustomerUI.

setUsername(): This method sets the usename of a customer.

manageFlights(): Allows the user access to flight management menus

manageServices(): allowsthe user to access Service Item management

manageDetails(): allows the user to access personal details management.

viewCustomerBookings(): view all bookings for a customer.

ChooseSeating(): This method allows a user to choose their seating option for a flight.

chooseServices(): This method allows a user to choose their services for a flight.

**Class: F lightManagemerController**

Attributes:

String userType: represent the type of user.

Methods:

void findAircraft(): This method finds an aircraft based id ID

void createAircraft();This method creates a new aircraft

void editAircraft(): This method allows the user to edit a aircraft object.

void deleteAircraft(); This method deletes an aircraft from the database.

void findAirport(): This method finds an airport based on ID

void createAirport(): This method creates a new Airport in the database.

void editAirport(): This method allows the user to modify an existing airport.

void deleteAirport(): This method deletes an airport from the database.

void findRoute(): This method finds a round form the dtabase based on ID.

void createRoute(): This method creates a new route in the database.

void editRoute(): This method allows the user to modify an existing route in the database.

void deleteRoute(): This method deletes a route from the database.

void findSchedule(): This method finds a schedule from the database based on ID,

void createSchedule(): This method creates a new schedule in the database.

void editSchedule(): This method allows the user to modify an existing checule object.

void deleteSchedule(): This method delete a schedule from the dtabase.

void flightReport(): This module outputs all the reports for flightmanager.

***Class: GuestUI***

***Attributes:***

Sqlite3\* db: a reference to the database.

String usertype: a string representing the type of user.

Methods:

getType(): returns the type of user controlling this interface.

setType(): set the type of the user controlling this interface.

run(): this method runs the main menu for a guest user.

login(): the overarching login function that handle differentiation between customers and staff members.

registerExistingCustomer(): This function handles the case if a customer already exists in the database, but they have never logged into the system before. This will enable them to do so.

guestSearch(): This method allows an unregistered user to search for flights.

Getpass\_lin(): linux specific function to handle password masking.

Getpass\_win(): windows specific method to handle password masking.

***Class: LoginController***

Attributes:

Sqlite3\* db: a reference to the database.

Methods:

validateLogin(): the login validation function. It checks the database to see if the password/username is valid.

validateStaffLogin(): The login function that validates staff accounts. Called from validateLogin().

validateCustomerLogin(): The login function that validates customer accounts. Called from validateLogin().

validateTravelAgentLogin(): The login function that validate travel agent accounts. Called from validateLogin().

***Class: ProfileManagerController***

Attributes:

Sqlite3\* db: a reference to the database.

String usertype: a string refering to the type of user using this function.

Methods:

findCustomer(): This method allows the user to get information about a customer from the database based on email.

profileReport():This method outputs all reporting information for the profile system.

***Class: RouteController***

Attributes:

Sqlite\* db: a reference to the database.

Methods:

alphabeticList(): This method outputs a list of available airports in alphabetical order.

***Class: ScheduleController***

Attributes:

Sqlite3\* db: a reference to the database.

Methods:

Search(): This method searches the database for schedules that fall within a date bracket on a particular route.

***Class: SearchController***

Attributes:

Sqlite3\* db: a reference to the database.

String usertype: A string referring to the type of user who is using this function.

Methods:

Search(): This method enables the user to search for a flight. It will return all flights on a particular route within a given timeframe.

getType(): returns the type of the user.

setType(string): assigns the type of the user.

***Class: Seat***

***Attributes:***

Sqlite3\* db: a reference to the database.

Int id: an integer which uniquely identifies a seat

Int scheduleID: an integer which identifies the schedule that this sead is associated with.

Int bookingID: an integer which identifies which booking this seat is associated with.

String seatClass: The class of seat this is (eg economy)

String seatNum: The number of seat on the plane.

Methods:

getID(): this method gets the ID of the seat from the DB.

getScheduleID(): this method gets the scheuldeID of the seat.

getBookingID(): this method gets the Booking ID of the seat.

getSeatClass(): this method gets the classification of seat.

getSeatNumber(): this method gets the seat number of a seat.

setDB(): assign the database reference.

setID(): this method assigns the ID of a seat.

setScheduleID(): This method assigns the scheduleID of a seat.

setBookingID(): this method assigns the BookingID of a seat.

setSeatClass(): this method assigns the seat classification of a seat.

setSeatNum(): this method assigns the seat number of a seat.

createSeat(): This method created a new seat and inserts its information into the database.

updateSeat(): This function update the information of an existing seat in the database.

getByScheduleID(): this function returns all seats allocated to a schedule.

checkExists(): This function check whether a seat already has been booked on a flight.

getByBooking(): This method gets a seat for a particular booking

convertSeatNum(): This is a utility method for converting user input of a seat number into a useable index.

**Class: ServiceManagerController**

Attributes:

Sqlite3\* database: a reference to the database.

String usertype: A string representing the type of user using this controller.

Methods:

findFlight(): returns all flights based on a FlightCode.

findService(): return all the services associated with a particular flight.

createService(): This method allows the user to create a new service item and add it to the database.

EditService(): This method allows a user to edit the information about an existing ServiceItem in the database.

deleteService(): This method allows the user to delete a service item from the database.

serviceReport(): This method will output all reports associated with the service system.

**Class: StaffUI**

Attributes:

Sqlite3\* db: a reference to the database.

String userType: a string referring to the type of user.

Methods:

Run(): This method displays the main menu for a staff member.

***Class: TravelAgentUI***

***Attributes:***

Sqlite3\* db: a reference to the database.

String userType: a string referring to the type of user.

***Methods:***

Run(): This method displays the main menu for a travel agent

bookingsMade(): This method returns all bookings made by a customer.

addCustomer(): This method allows a travelAgent to add a new customer to our system.

editDetails(): This method enables a travel agent to edit their own details.

***Class: WeatherController***

Attributes:

Sqlite3\* db: A reference to the database.

String weatherQuery: A string that stores thet HTTP request query.

Float lat: float to store latitude

Float lon: float to store longitude

Int numDays: An integer to store the amount of days the query should request

Methods:

setWQuery(): takes the values in lat,long,weatherQuery and numDays and creates the HTTP request string.

setLat(): Assign a value to latitude.

setLon(): Assign a value to longitude.

setNumDays(): Assign the number of days the query request is for.

executeQuery(): send HTTP request and store results in XML file.

displayResults(): Display results from executeQuery().

getAirportWeather(): gets the weather conditions as a specified airport.