**NorthEastWorst Airline Documentation**

for

Dr. Gerstl

Database Instructor

Farmingdale State College

Farmingdale, NY

by

Jay R. Kaplan

BCS 260

April 30, 2015

**Table of Contents**

Introduction ………………..……………………………………..………………………………….2

ER Diagram ………………..……………………………………..………………………………….3

Entities ………………..……………………………………..………………………………….4

Decomposition ………………..……………………………………..………………………………….6

Table Description ………………..……………………………………..………………………………….7

SQL Query/Updates ………………..……………………………………..………………………………….9

Application Guide ………………..………...…………………………..…………………………………12

References ………………..……………………………………..………………………………...16

**Introduction**

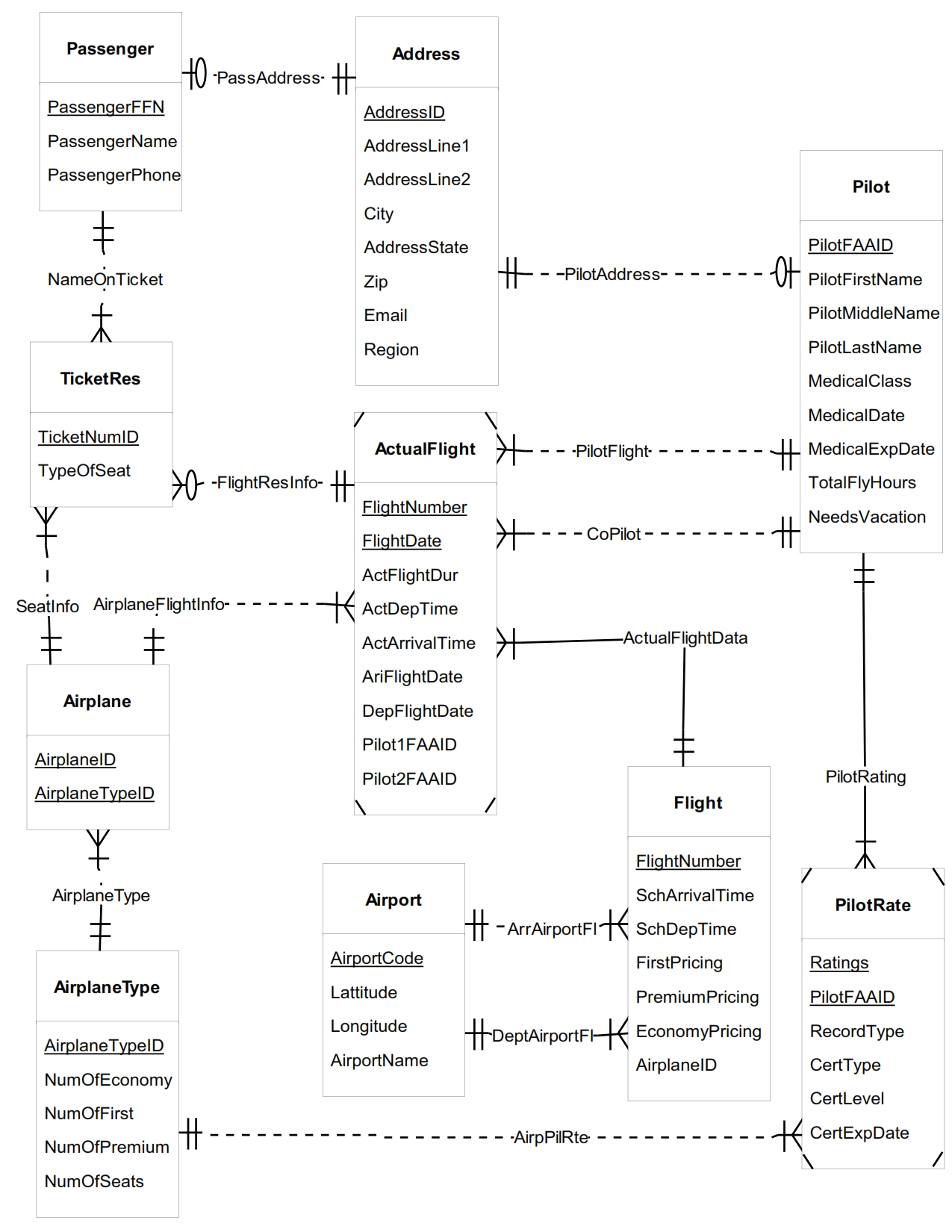
Farmingdale Airways has requested a new database be built in order to save on the costs of purchasing and subscribing to Sabre Airline Solution. There are four components of the database system: pilot assignments, airplane and flight schedules, and a ticketing system.

The pilot assignment system is required to have two pilots per flight. Each pilot is given an FAA rating. That rating gives them the ability to fly specific aircraft and each pilot is rated per each plane in the Farmingdale fleet. There is also a need to calculate the flight duration of each plane for each pilot to whom it is assigned. Therefore, each pilot can be given vacation time after they have flown over a certain amount of hours in a given month.

The airline has a fleet of 8 planes of which there are 4 different types. The planes fly 20 flights per day through 6 routes using 4 airports. Each of the 4 different airplanes has their own seating structure. Two of the plane types offer only economy seats while the larger two offer economy, premium, and first class seats.

The project also requires a ticketing system with the ability to sell tickets for specific flights. Every seat has a fixed price per its specific route. There is no dynamic pricing based on seat availability. Each of the four plane types has unique seating structures; two of the four offer only economy for the short route flights and the other two offer first class for the higher paying airfares. When a new customer registers in the system, they automatically will become part of the frequent flyer program.

**Show the ER Diagram and explain the entities and relationships**



**There are 10 entities in this ERD:**

1. Address
2. Airport
3. Pilot
4. Passenger
5. Airplane Type
6. Airplane
7. Pilot Rating
8. Flight
9. Actual Flight
10. Ticket Reservation
11. Address(1) table is a strong entity with its own surrogate key representing addresses of both pilots and passengers. Address has a one-to-one relation with both the pilot(2) and passenger(4) table. Pilot(2) and passenger(4) have a mandatory relationship with address while the address table is optional because a particular addressID can only have a relationship with either a pilot or a passenger, not both at the same time.
12. Airport(2) table is a strong entity with its own airport code. The airport code is a 4 letter code which represents the four different airports in the current database. This table has a direct relationship with the flight(8) table. Airport(2) table has two relationships with flight(2) table, as it provides the arriving and departing airport data. It has a one to many relationship with the flight(8) table, as a flight must have a minimum of one airport, but an airport can show up many times in the flight(8) table.
13. Pilot(3) table is a strong entity that contains the information for individual pilots. The primary key of this table is the pilot’s FAA license number. This table has a relationship with the address(1) table, pilot rating(7) table, and actual flight(9) table. The pilot table contains both pilot and co-pilot data. A pilot has a mandatory one-to-one relationship with address, as each pilot must have an address and can only have one address. The pilot(3) table has a pilot relationship with actual flight(9) table. This relationship takes the place of Pilot1FAAID. It is a one to many table, with a minimum of 1 pilot assigned per actual flight and a maximum of many pilots assigned to multiple flights. The co-pilot relationship goes to Pilot2FAAID, as it is a minimum of 1 co-pilot assigned per flight, but many co-pilots are assigned to each flight in the data base. Pilot(3) table has an identifying relationship with the pilot rating table because the primary key of Pilot(3) walks across the solid line and becomes part of the composite primary key of pilot rating(7).
14. Passenger(4) table is a strong entity that has a relationship with both the address(1) table and the ticket reservation(10) table. The passenger(4) table has a one-to-one relationship with address(1) because every passenger must have one and only address. The passenger(4) table has a one to many relationship with the ticket reservation(10) table. A passenger must have their information on at least one ticket, and can have multiple tickets purchased with their name on it.
15. Airplane Type(5) table holds the information about the type of airplane and the seats on a particular plane. It is a strong entity that has a one to many relationship with the Airplane(6) table and a one to many relationship with the Pilot Rating(7) table. Airplane type must have a minimum of one in the Airplane(6) table, and can have a maximum of many as the airplane types can be duplicated in the airplane(6) table.
16. Airplane(6) table is a table that matches up an airplane model number with its type of airplane. For example: Airplane ID Notfun is an ERJ145 airplane type. It has strong entity relationships with the Ticket Reservation(10) table, Airplane Type(5) table, and Pilot Rating(7) table. Airplane (6) has a one to many with the ticket reservation(10) table because a purchased flight must have one plane per ticket, but an airplane can be associated with different tickets purchased. Airplane(6) has a one to many relationship with actual flight(9) because each flight must have one airplane associated with it, and an airplane can have more than one flight in a day. Airplane(6) has a one-to-one relationship with airplane type(5). The fleet has 4 different types of airplanes, each airplane has to have one type, and the airplane id will not be repeated in the database, thereby making it a one to one relationship.
17. Pilot Rating(7) is a weak identity because it uses the primary key of the pilot(3) table as part of the composite primary key in pilot rating(7). Pilot rating has two relationships. One comes from the pilot(3) table. It’s an identifying relationship because the key from the pilot(3) table walks across and becomes part of the primary key of the pilot rating table. Pilot rating(7) also has a one-to-one relationship with airplane type(5). A pilot rating id can only happen for one and only one plane. If a pilot is rated for more than one plane, they would get another rating in the database. The pilot and rating are combined here to make up the composite key.
18. Flight(8) table is a strong entity as it uses a flight number as its primary key. Currently there are 20 flights in a given day. Each flight has a set route, schedule, and seat pricing. The flight(8) table has three relationships, two of which are shared with the airport(2) table. Flight has a one-to-one relationship with airport(2) for both arriving and departing flights. Each flight number can only leave from one airport and arrive in one airport. The information in this table provides general flight information and is used to provide information to the actual flight table.
19. Actual Flight(9) is a weak entity because its composite primary key uses the primary key from the flight(8) table. This table has 5 relationships, a one-to-one with airplane(2), a zero to many with ticket reservation(10), one-to-one with pilot(3), one-to-one with co-pilot, and an identifying relation of one-to-one with flight(8). Actual flight’s(9) one-to-one with airplane(2) is a mandatory relationship because each actual flight must have one and only one flight associated with it. Actual flight(9) can have zero or many relationships with a ticket reservation, because there is a possibility that no tickets are purchased on an actual flight, such as when it is created in the database. An actual flight(9) must have one pilot and one co-pilot per flight. This is represented by two one-to-one relationships with the pilot(3) table. Actual flight(9) gets its data from the flight(8) table. It is this table that lends its primary key to create the actual flight(9) table. It has a one-to-one relationship because an actual flight must have one and only one set of flight data. It is the relationship from the other direction because of fight date that Flight(8) has a one to many relationship with actual flight(9).
20. Ticket Reservation(10) is a strong entity that generates a unique ticket number to a specific flight for a passenger in a seat on the plane. Ticket reservation(10) has three relationships. It has a one-to-one relationship with airplane(2) because it uses this table to retrieve seat information for the flight. There is only one airplane per ticket reservation. Ticket reservation(10) has a one-to-one relationship with actual flight(9) because a ticket reservation can have one and only one actual flight per reservation number. Ticket reservation(10) has a one-to-one relationship with passenger, because one and only one passenger is allowed per ticket.

**Decomposition: Discuss the data as originally supplied, the functional dependencies discovered, and the decomposition. Show that the table is in BCNF.**

The airplane project brief stated 8 planes in total. Each plane is one of four types of planes. The model number is functionally dependent on the airplane type. Each type of plane has a unique seat structure. The seats are functionally dependent on the type of plane.

Every flight has two certified pilots. Each pilot is rated on a particular type of plane. A pilot’s assignment to a flight is functionally dependent on the pilot’s ratings. There was a spreadsheet supplied that had a tab called Pilot’s Certificate. Inside of Pilot’s Certificate, it had a type of rating which corresponded to the type of plane for which a particular pilot was rated. The pilot FAAID number is functionally dependent on the airplane type. The other fields that are functionally dependent on types of ratings are: record type, certificate type, certificate level, certificate expiration date, and ratings.

The pilot tab has a unique id which contains data in the row that is functionally dependent on the unique id. The address, city, state, and zip were moved from the Pilot’s basic information tab and into their own table and a surrogate primary key was created which connected back to the pilot information. The address table was also used to separate out the customer’s address. Medical class, medical date, record type and region are functionally dependent on the FAAID.

A flight table was provided that showed 20 different flights that flew 7 days a week, 365 days a year. Each flight had its own from-and-to airports and a departure time. The from, to, and departure times were all dependent on the flight number.

The pricing table had a similar structure as the flight table. It provided a unique flight number with first, premium, and economy pricing. The pricing per seat is functionally dependent on the flight number.

The December reservation tab had the largest amount of information in one tab and became normalized into multiple tables:

* Airport Table: airport code → latitude and longitude, and the airport names. This information became the airport table.
* Actual Flight: Flight Number and Flight Date → these two became the primary keys of the actual flight table and had a number of columns which were functionally dependent on it. Actual departure time, scheduled arrival time, scheduled departure time, flight duration, airport departure, airport arrival, pilot1, pilot2.
* Airplane: Airplane ID → airplane type, seats on plane.
* Flight Table: flight → functionally dependent on flight was flight number, departure airport, arrival airport, and scheduled departure.
* Ticket Reservation Table: Ticket Reservation → price paid, type of seat, airport departure, airport arrival, seat number.

**Table Description: Discuss every table and every field. Explain what they model and any database constraints (e.g., keys, foreign keys, etc).**

**Table:**

Address - this table is an address table that models the address for customers and pilots.

**Fields:**

AddressID - primary key for the address table.

AddressLine1 - captures 1st line of an address

AddressLine2 - captures 2nd line of an address; this is null.

City - captures the city of an address

Email - captures the email of a customer and pilot

Region - this was part of the excel sheet and added to the table as null

AddressState - due to state being a reserved word, I used AddressState. It is used for State.

Zip - ZipCode for an address

**Table:**

AirportCode – this table is used to hold all the airport codes, names, and geo location.

**Fields:**

AirportCode - this is a 4 letter code to represent the airport; it is a primary key.

AirportName - this is the name of the airport

Lattitude - Latitude is the North/South coordinates of an airport; it’s part of the geo location

Longitude - Longitude is the East/West Coordinates of an airport; it’s part of the geo location

**Table:**

Pilot – this table holds the information from the FAA documents. The fields represent the data that was obtained from the client.

**Fields:**

PilotFAAID - Primary key and unique number given by the FAA for each pilot: the pilot’s license #

MedicalClass - three kinds of medical classes ranked by the FAA for each individual pilot

MedicalDate - Date of medical class ranking

MedicalExpDate - Expiration date of medical class ranking

NeedsVacation - calculated field that determines when a pilot has flown too many hours

PilotFirstName – Pilot’s first name

PilotLastName – Pilot’s last name

PilotMiddleName – Pilot’s middle name; this field is a null value

TotalFlyHours - total amount of hours flown

AddressID - this is a foreign key to the address table, it holds the pilot’s address

**Table:**

Passenger - this is the passenger/customer table.

**Fields:**

PassengerFFN - the primary key that holds the address info, passenger name, and phone.

AddressID - foreign key to the address table that holds the address information

PassengerName – passenger’s name

PassengerPhone – passenger’s phone number

**Table:**

AirplaneType - Holds the airplane type information, number, and type of seats.

**Fields:**

AirplaneTypeID - Holds the type of airplane - airline currently has 4 different types

NumOfEconomy - number of economy seats on a particular plane

NumOfFirst - number of first class seats on a particular plane

NumOfPremium - number of premium economy seats on a particular plane

NumOfSeats - number of total seats on a particular plane

**Table:**

Airplane - table for Airplane Type and Airplane Model ID.

**Fields:**

AirplaneID - this is the model number of a specific airplane

AirplaneTypeID - this is the type of airplane in the fleet, a foreign key to Airplane Type table.

**Table:**

PilotRate – this table shows how a pilot is rated on a particular airplane type.

**Fields:**

PilotFAAID - foreign key to the pilot table

AirplaneTypeID - foreign key to the airplane type table

CertExpDate - expiration date of the pilot’s FAA license

CertLevel - Level of FAA certification

CertType - Type of FAA certification

Ratings - Instrument ratings

RecordType - indicates whether the record contains Basic or Certificate data

\*PilotFAAID and AirplaneTypeID make up a composite primary key for this table

**Table:**

Flight – this table holds the daily flight data information.

**Fields:**

FlightNumber - number of each flight, 20 flights per day

ArrivalAirport - airport that the flight is arriving at, foreign key to airport table

DeptAirport - airport that flight has taken off from, foreign key to airport table.

SchArrivalTime - scheduled arrival time of flight

SchDepTime - scheduled departure time of flight

EconomyPrice - price of economy seat on a set flight

PremiumPricing - price of premium seat on a set flight

FirstPricing - price of first class seat on a set flight

AirplaneID - airplane that is flying a certain route, foreign key to airplane table.

**Table:**

ActualFlight - table that holds all flight information by date; retains info in database.

**Fields:**

FlightNumber - flight number of a route, foreign key to flight table

FlightDate - date of flight in database

ActArrivalTime - time that flight arrived in arriving airport

ActDepTime - time that flight left departing airport

ActFlightDur - total time of flight

AriFlightDate - arriving flight date

DepFlightDate - departing flight date

AirplaneID - airplane that was used on flight

PilotFAAID1 - FAA # of Pilot on flight, foreign key to Pilot table

PilotFAAID2 - FAA # of Co-Pilot on flight, foreign key to Pilot table

**Table:**

TicketRes - Creates a ticket in the ticketing system.

**Fields:**

TicketNumberID - creates a unique ticket number; this is the primary key of this table.

AirplaneID - Airplane that will be used on flight

PassengerFFN - unique customer number that is either used or created at time of purchase, foreign key to the flight table.

FlightNumber - flight number that was assigned to purchased flight, composite foreign key to flight table.

FlightDate - date of chosen flight, composite foreign key to flight table

TypeOfSeat - type of seat customer selected

**SQL: In part 2 you completed an application for database access. Give each query/update from part 2 and explain its operation.**

1) Get the list of flights between two airports, including the flights and number of seats given a date and the two airports (you can hardcode these in the submission)

*select Flight.FlightNumber, FlightDate,ArrivalAirport, DeptAirport,ActualFlight.AirplaneID, Airplane.AirplaneTypeID, NumOfSeats, EconomyPrice*

*from Flight*

*join ActualFlight on Flight.FlightNumber = ActualFlight.FlightNumber*

*join Airplane on Airplane.AirplaneID = ActualFlight.AirplaneID*

*join AirplaneType on Airplane.AirplaneTypeID = AirplaneType.AirplaneTypeID*

*where ArrivalAirport = 'KPHL' and DeptAirport = 'KISP' and FlightDate = '2015-4-16';*

This query was completed by combining 8 columns from multiple tables to produce a table that could show the data and satisfy this request. Flight number exists in fight and actual flight. Actual flight is required to get the flight date. Arrival and departure airports are in flight table. The airplane that flew the actual flight is also in the actual flight table. When actual flight is joined with airplane type, the query is able to acquire number of seats and pricing of those seats.

2) Book a flight given the date, class, frequent flyer number (you can hardcode these in the submission)

*INSERT INTO TicketRes (TicketNumberID, AirplaneID, PassengerFFN, FlightNumber, FlightDate, TypeOfSeat) VALUES ('19916818','N00dle','90218','8','2015-04-16','First');*

This was an insert statement that booked a flight for a specific date and customer. The insert statement was hard coded with values, including one that manually created a new reservation number.

3) Register a customer for the frequent flyer program given the customer information (you can hardcode these in the submission)

*INSERT INTO Address (AddressID,AddressLine1,AddressLine2,City,Email,Region,AddressState,Zip) VALUES (32,'123 Main St.','','Bellmore','abc@123.com','','NY','11710');*

*INSERT INTO Passenger (PassengerFFN,PassengerName,AddressID,PassengerPhone) VALUES (18976,'Bobby Jones',32,'(917) 613-3956');*

In order to register a customer for the frequent flyer program, two insert statements are required. One to the address table, which creates an address id that is then used in the passenger table to create a passenger FFN, which address id is a foreign key to.

4) Get a list of the pilots who are qualified to fly a specific plane given the airplane id (you can hardcode these in the submission)

*select Airplane.AirplaneID, PilotRate.PilotFAAID*

*from Airplane*

*join PilotRate on Airplane.AirplaneTypeID = PilotRate.AirplaneTypeID*

*where Airplane.AirplaneID = 'N00dle';*

The airplane table and pilot rating table are queried together to acquire the names of the airplanes on which the pilot is rated to fly. This query takes a specific airplane id and gives back the pilots that are rated for that specific plane.

5) Find out which plane is flying a specific flight given the date, time, and route (you can hardcode these in the submission)

*select Flight.FlightNumber, FlightDate, SchDepTime ,ActualFlight.AirplaneID*

*from ActualFlight*

*join Flight on ActualFlight.FlightNumber = Flight.FlightNumber*

*where FlightDate = '2015-04-16' and SchDepTime = '15:00:00' and ActualFlight.FlightNumber = '3';*

Flights are based on scheduled departure, not the actual departure. Actual flight table has the date of a specific flight, while flight table has the basic daily information. The table returns the flight number, flight date, scheduled departure time, and airplane id on a specific date for that query.

6) Assign a pilot to a flight given the flight information and the pilot id (you can hardcode these in the submission)

*INSERT INTO Flight (FlightNumber,ArrivalAirport,DeptAirport,SchArrivalTime,SchDepTime,EconomyPrice,PremiumPricing,FirstPricing,AirplaneID) VALUES (21,'KDCA','KISP','19:30:00','18:30:00','$249 ','$299 ','$700 ','N0000');*

*INSERT INTO ActualFlight (FlightNumber,FlightDate,ActArrivalTime,ActDepTime,ActFlightDur,AriFlightDate,DepFlightDate,AirplaneID,PilotFAAID1,PilotFAAID2) VALUES (21,'2015-04-16','19:30:00','18:30:00',1,'2015-04-16','2015-04-16','N0000','13086596','90181863');*

*UPDATE Pilot Set TotalFlyHours = (select ActFlightDur from ActualFlight where FlightNumber = 21 and FlightDate = '2015-04-16') where PilotFAAID = 13086596;*

This query inserts a new flight into the flight table, then inserts that new flight into the actual flight table. It then assigns two pilots to that flight through the actual flight table. Finally, the query updates the total pilot flight hours for that flown flight using the actual flight data table.

7) Create flights for a new day (create all the necessary entries--you may hardcode the day)

INSERT INTO ActualFlight (FlightNumber,FlightDate,AirplaneID) VALUES (1,'2015-04-17','Notfun');

INSERT INTO ActualFlight (FlightNumber,FlightDate,AirplaneID) VALUES (2,'2015-04-17','Notfun');

INSERT INTO ActualFlight (FlightNumber,FlightDate,AirplaneID) VALUES (3,'2015-04-17','Notfun');

INSERT INTO ActualFlight (FlightNumber,FlightDate,AirplaneID) VALUES (4,'2015-04-17','Notfun');

INSERT INTO ActualFlight (FlightNumber,FlightDate,AirplaneID) VALUES (5,'2015-04-17','Nearly');

INSERT INTO ActualFlight (FlightNumber,FlightDate,AirplaneID) VALUES (6,'2015-04-17','N00dle');

INSERT INTO ActualFlight (FlightNumber,FlightDate,AirplaneID) VALUES (7,'2015-04-17','Nearly');

INSERT INTO ActualFlight (FlightNumber,FlightDate,AirplaneID) VALUES (8,'2015-04-17','N00dle');

INSERT INTO ActualFlight (FlightNumber,FlightDate,AirplaneID) VALUES (9,'2015-04-17','Nearly');

INSERT INTO ActualFlight (FlightNumber,FlightDate,AirplaneID) VALUES (10,'2015-04-17','N00dle');

INSERT INTO ActualFlight (FlightNumber,FlightDate,AirplaneID) VALUES (11,'2015-04-17','Nearly');

INSERT INTO ActualFlight (FlightNumber,FlightDate,AirplaneID) VALUES (12,'2015-04-17','N00dle');

INSERT INTO ActualFlight (FlightNumber,FlightDate,AirplaneID) VALUES (13,'2015-04-17','N0tgrt');

INSERT INTO ActualFlight (FlightNumber,FlightDate,AirplaneID) VALUES (14,'2015-04-17','N0tbad');

INSERT INTO ActualFlight (FlightNumber,FlightDate,AirplaneID) VALUES (15,'2015-04-17','Ntg00d');

INSERT INTO ActualFlight (FlightNumber,FlightDate,AirplaneID) VALUES (16,'2015-04-17','N0000');

INSERT INTO ActualFlight (FlightNumber,FlightDate,AirplaneID) VALUES (17,'2015-04-17','N0tgrt');

INSERT INTO ActualFlight (FlightNumber,FlightDate,AirplaneID) VALUES (18,'2015-04-17','N0tbad');

INSERT INTO ActualFlight (FlightNumber,FlightDate,AirplaneID) VALUES (19,'2015-04-17','Ntg00d');

INSERT INTO ActualFlight (FlightNumber,FlightDate,AirplaneID) VALUES (20,'2015-04-17','N0000');

This hard codes a new day’s worth of flight information into the actual flight table using flight number, airplane id, and a flight date.

8) Print the manifest (list of passengers) on a specific flight

*Select \**

*from TicketRes, Passenger, Address*

*where TicketRes.PassengerFFN = Passenger.PassengerFFN and Address.AddressID = Passenger. AddressID and FlightNumber = '1' and FlightDate = '2015-04-16';*

Using the date of 4-16-2015, this query selects all columns from ticket reservation, passenger, and address tables who are flying on a specific flight.

9) Get the personal information for a passenger from their ID

*select PassengerName, PassengerPhone, AddressLine1, AddressLine2, City, AddressState, Zip, Email*

*from Passenger*

*join Address on Passenger.AddressID = Address.AddressID*

*where PassengerFFN = 12476;*

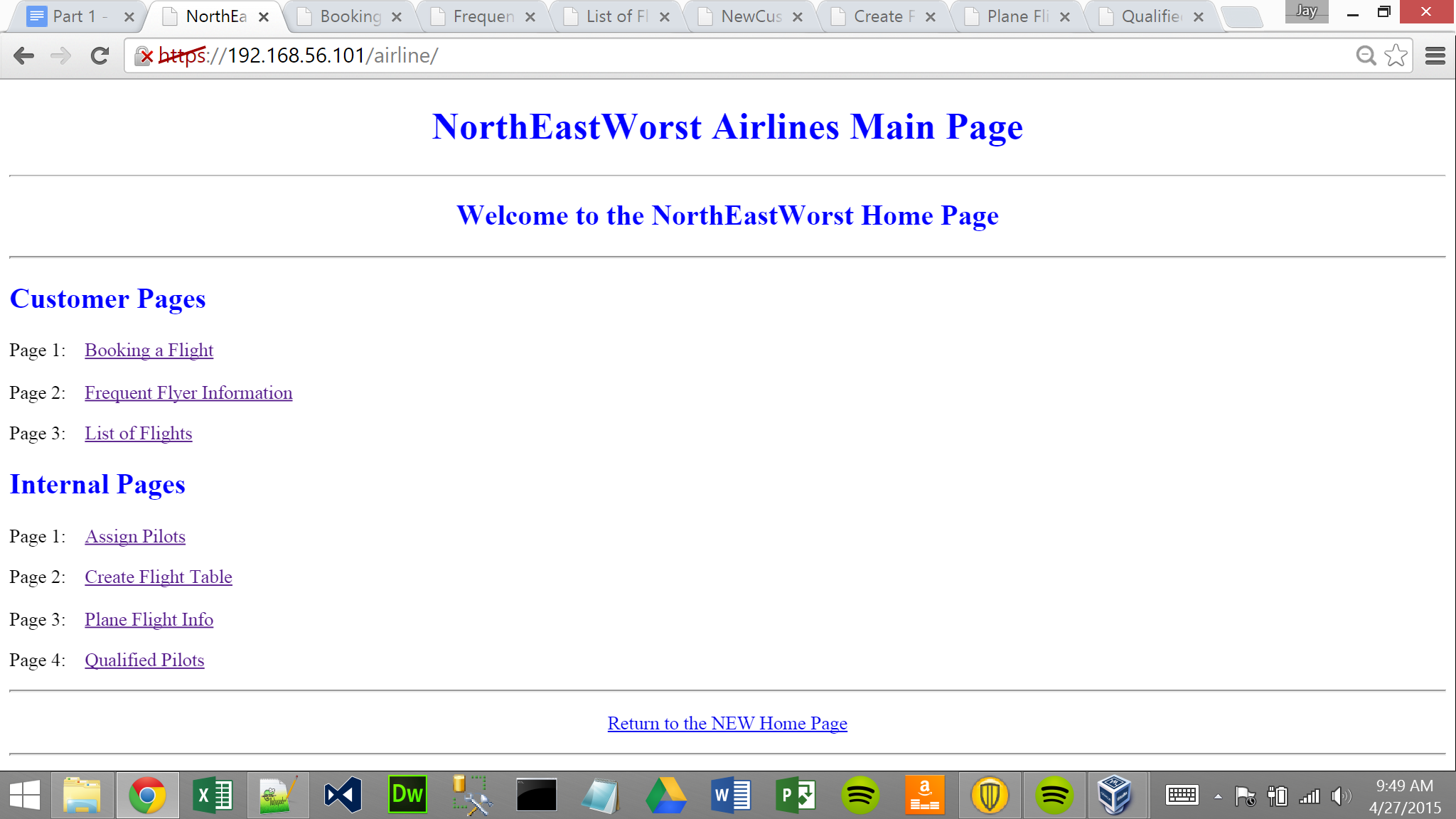
This query selects passenger name, passenger phone number, address line 1 and 2, city, state, zip and email for specific passenger id.

**Application- In part 3 you will produce an application that uses 2 of the SQL statements with IIS. Show how to use these in this section.**

# **NorthEastWorst Airline**

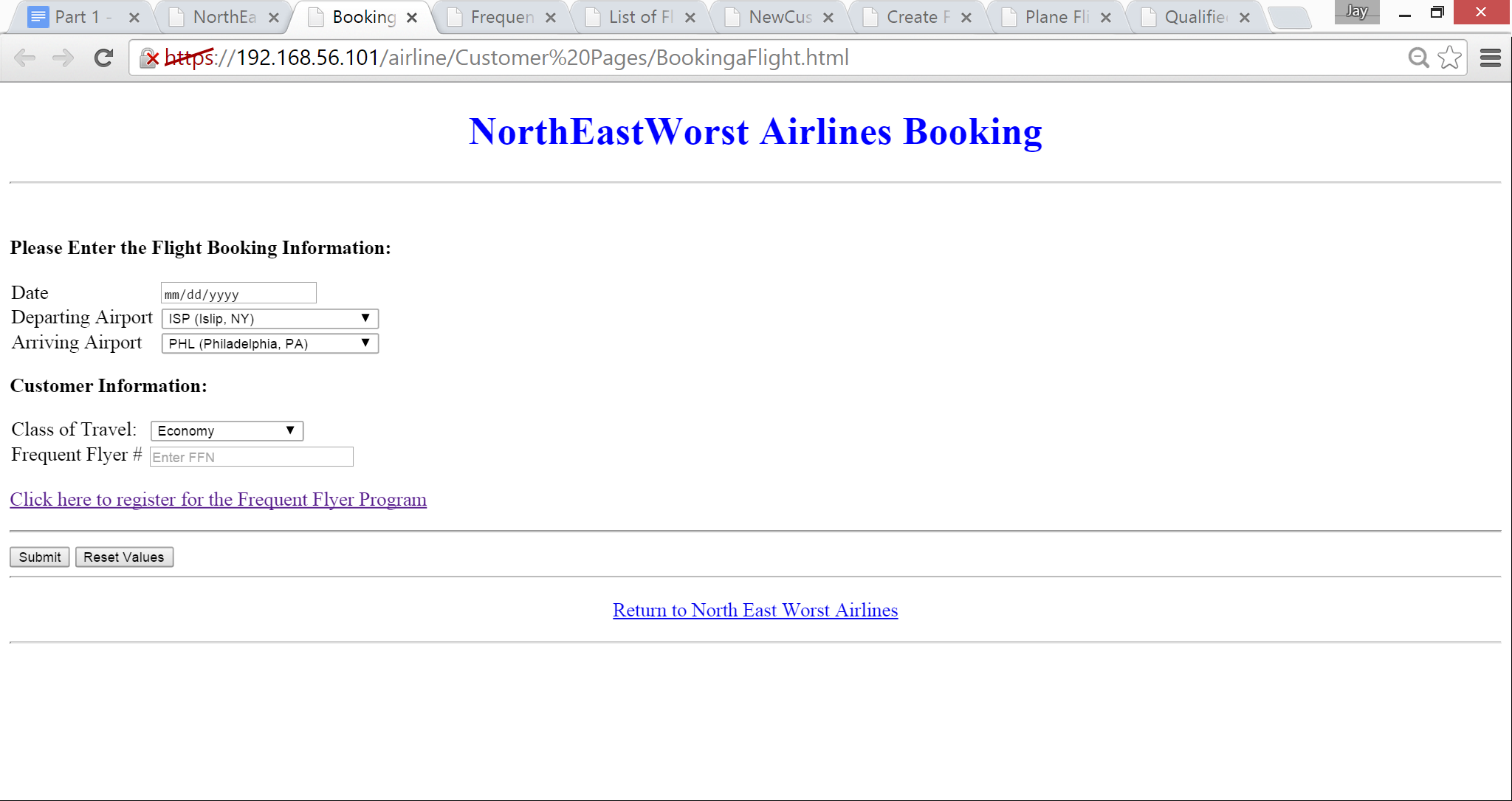
**Application Guide for Airline Staff**

# 



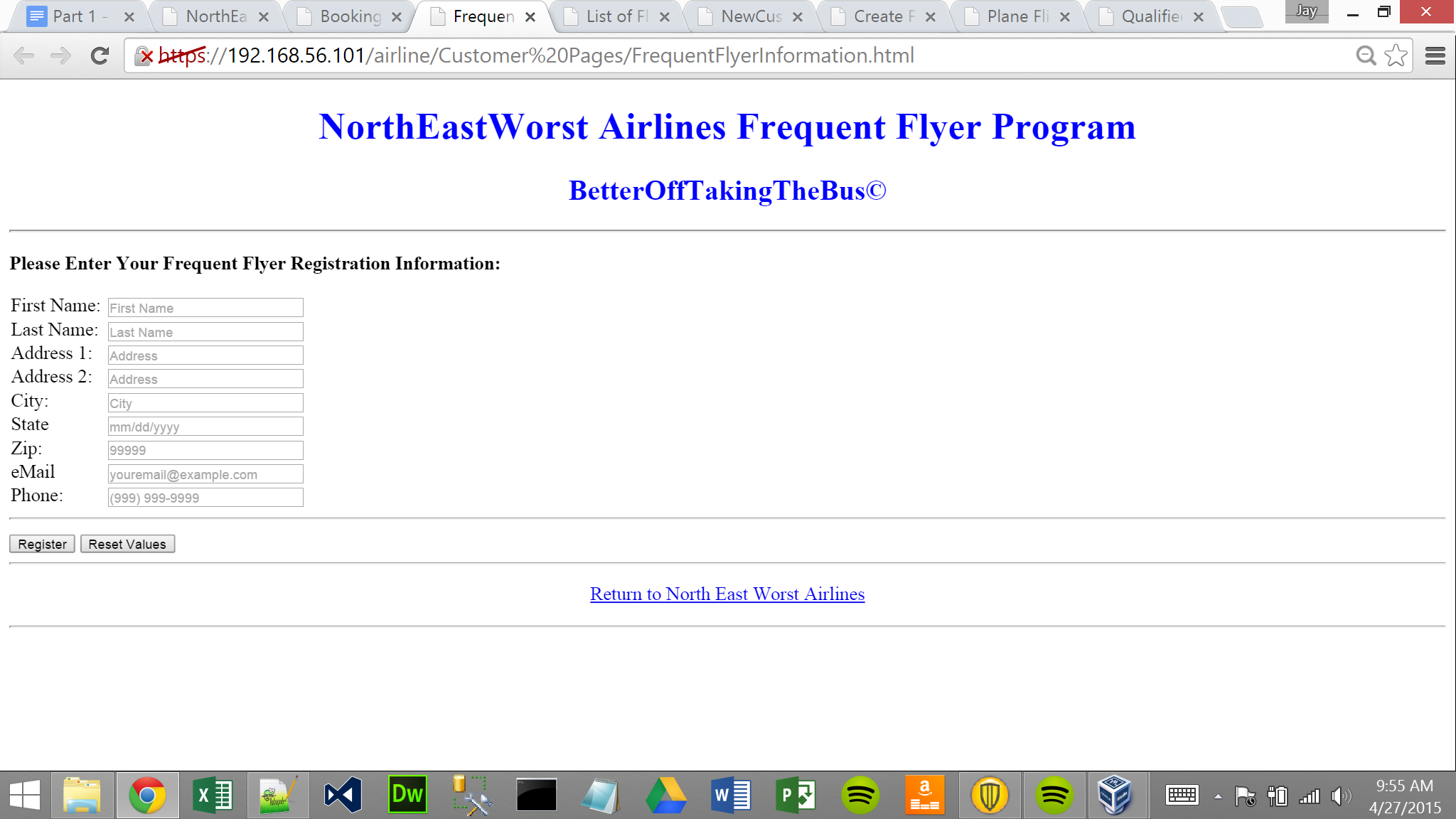
**Main Page**

This is the main page for NorthEastWorst Airlines. This page has 3 customer pages and 4 internal pages, making a total of 8 pages including the main page. On this page, you will click a link and that link will take you to the correct page.



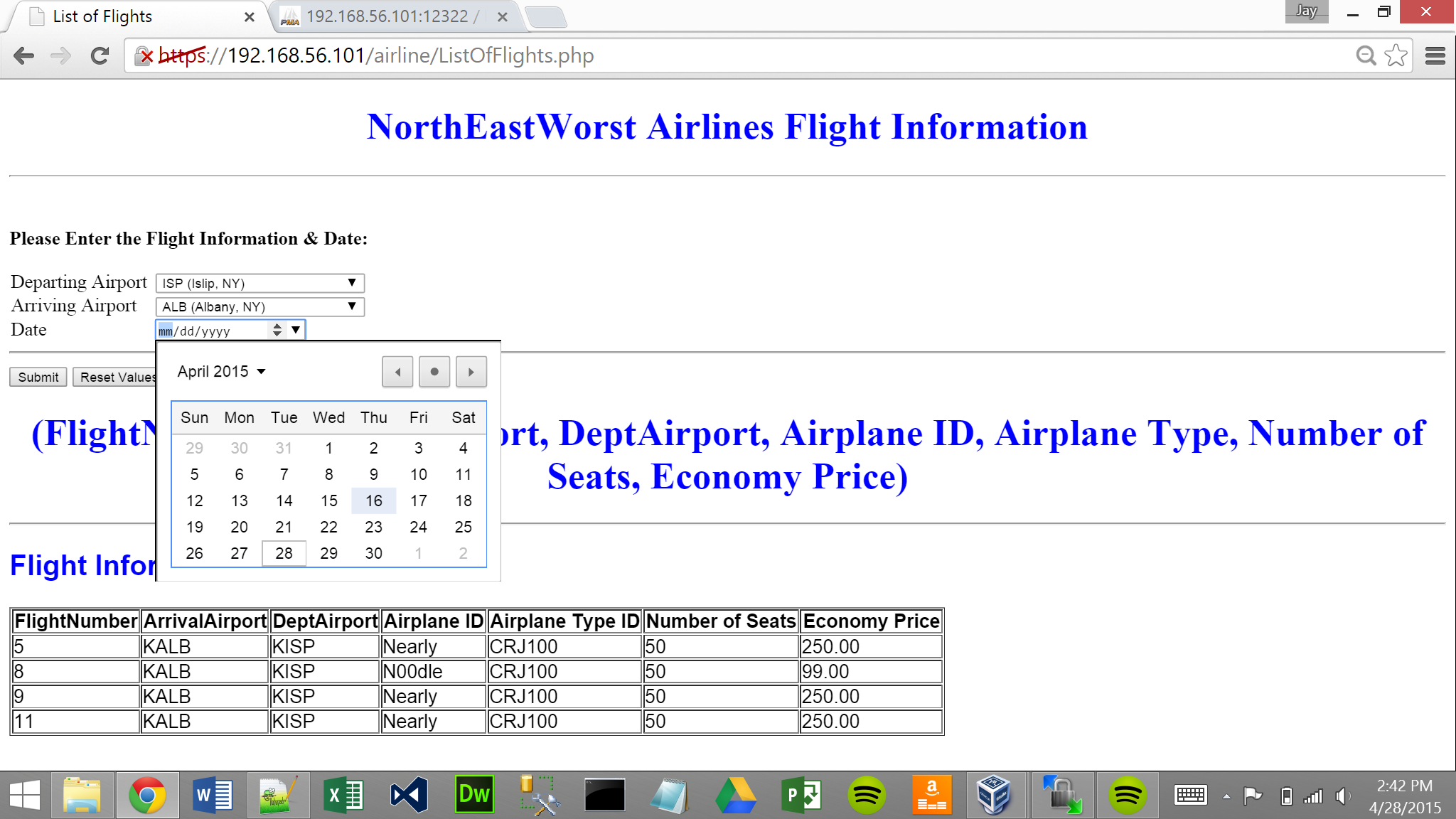
**Page 1: Booking a Flight.**

This is the page to book a customer’s flight. Select a date using the date box and a date widget will appear which allows you to select the date. Then, select the departing and arriving airports by using the drop down arrow to select the appropriate destination. Next, select the class of travel (economy, premium, or first) and enter a customer’s frequent flier number. If the customer does not have a number, click the link to create a frequent flyer number.



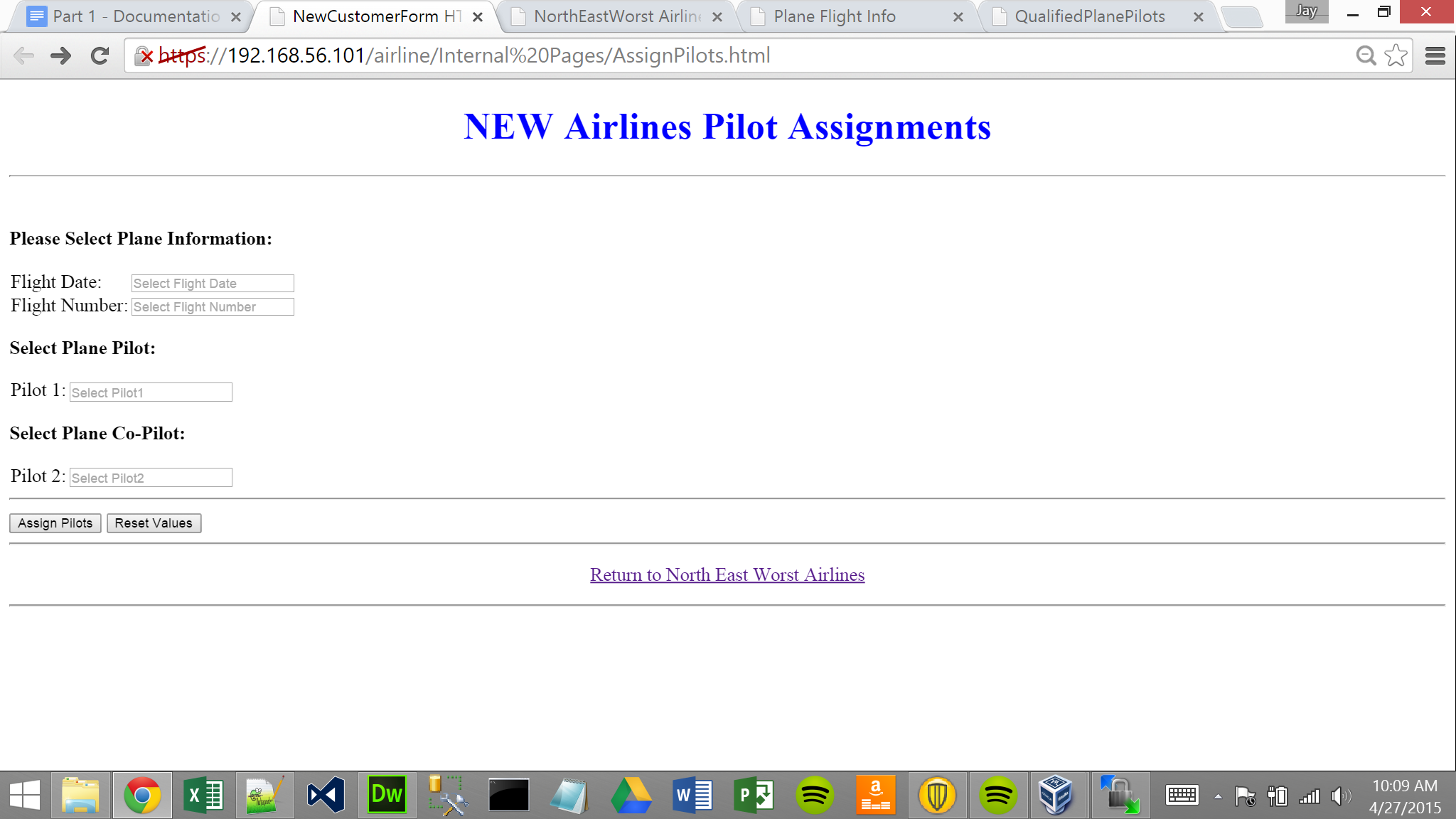
**Page 2: Frequent Flyer Information.**

Use this page to enter the customer’s information. It allows you to enter first name, last name, address 1, address 2 if needed, city, state, zip, email, and phone number. Each white box has a hint of how to enter the information. Then click register. You can click the reset values if there is a mistake and it will erase all the data on the page.



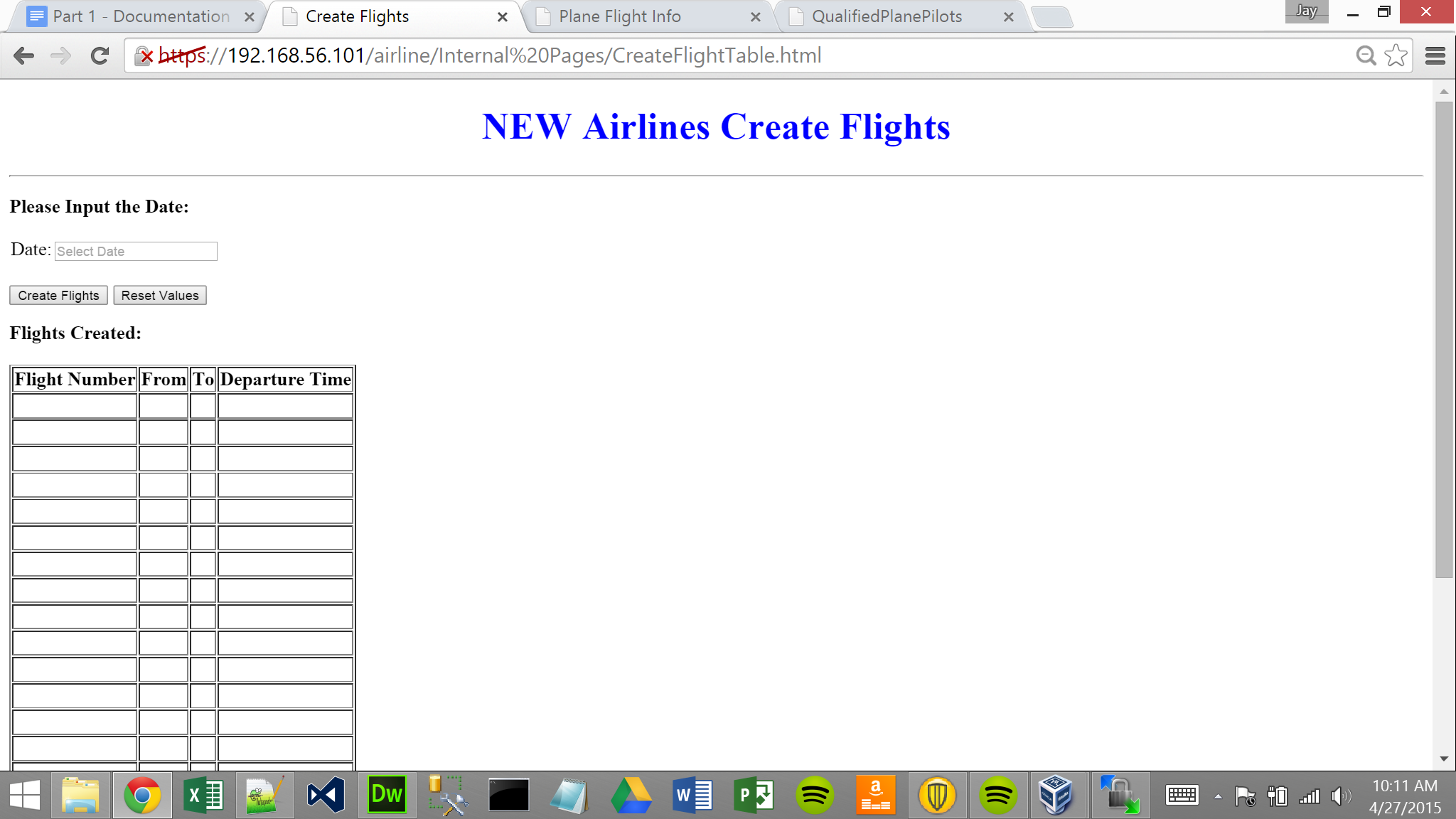
**Page 3: Flight Information and Date.**

Select departing airport from the drop down menu. Then select arriving airport from the drop down menu. Click the date drop down menu for the data picker widget. For this test data only 4/16/2015 and 4/17/2015 are currently available. Finally click submit. This will return the flight information on the bottom of the page.



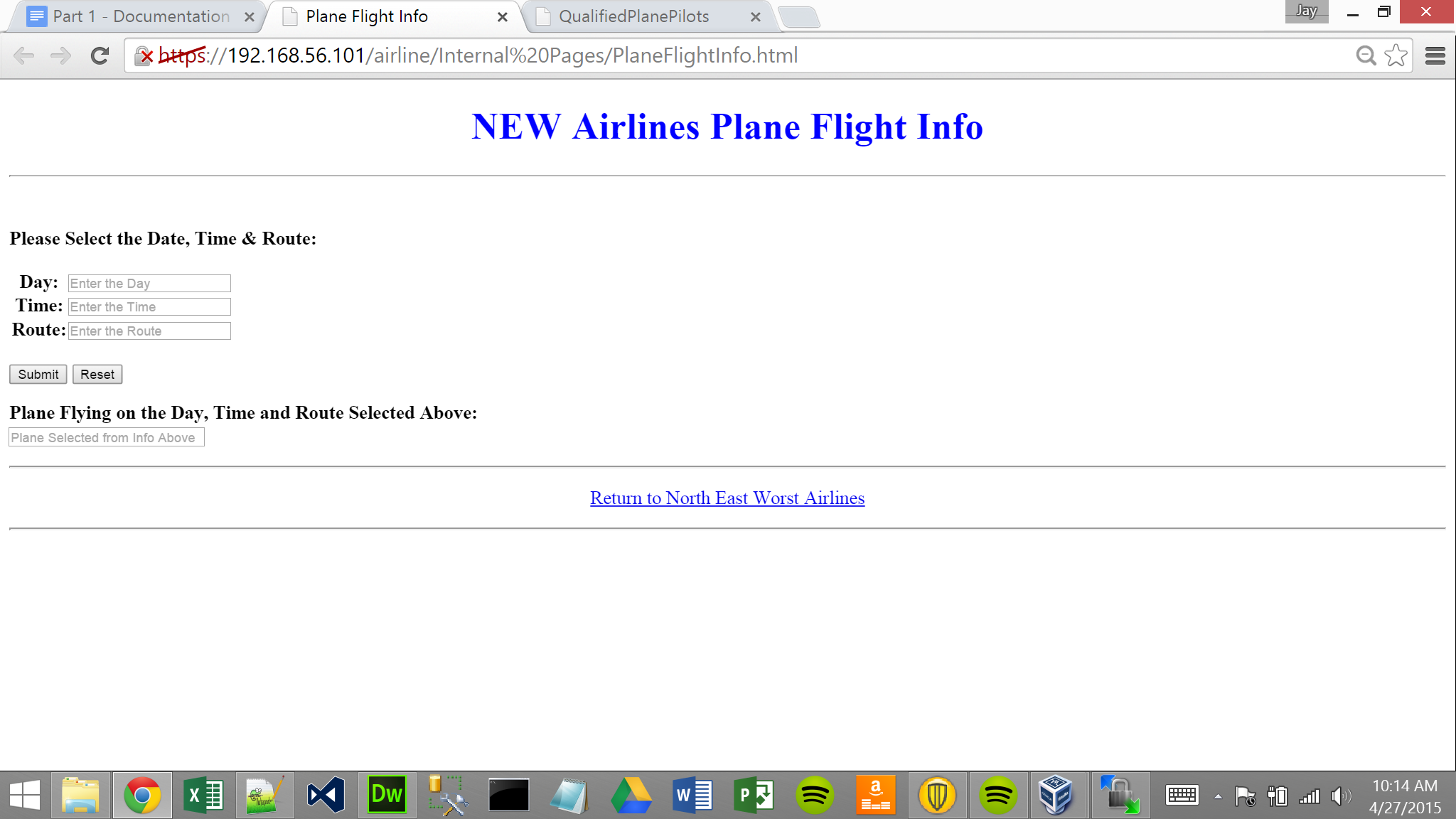
**Page 4: Assign Airline Pilots.**

This page assigns a pilot and co-pilot. Enter the flight date and flight number under plane information, then select the pilot and co-pilot, and click assign.



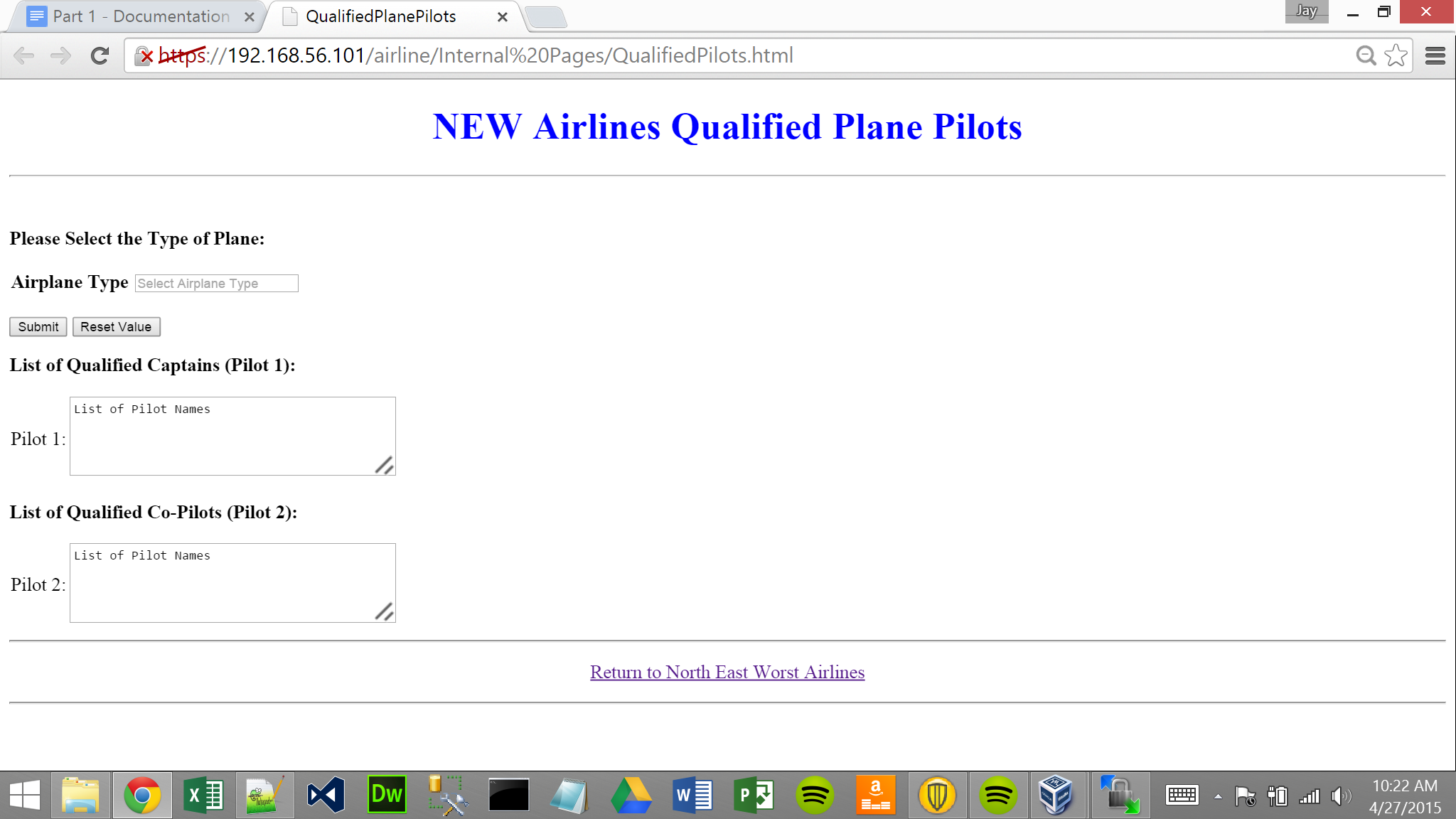
**Page 5: Create Flights for One Date.**

Select a date to create flights, then click on create flights and the flight table will populate with data from the flight table and insert into the actual flight table.



**Page 6: Plane Flight Information.**

Select the day, time, and route, then click the submit button. If you make a mistake, you can select the reset button which will clear the boxes. After clicking the submit button, the plane that is flying on the selected day, time, and route will appear in the box on the bottom of the page.



**Page 7: Find Qualified Airplane Pilot & Co-Pilot.**

Select the airplane type from the airplane type select box. Then click the submit button, wherein a listing of qualified pilots and co-pilots will appear in the respective boxes.

References

Forta, Ben. *Teach Yourself SQL in 10 Minutes*. 4th ed. Sams Publishing, 2012. Print.

Gerstl, David, ed. "Topic 016 - Project Background and Details." *Database - 24242*. N.d. Web. 24 Apr 2015.

Gerstl, David, ed. "Airline Background Info." *Database - 24242*. N.d. Web. 24 Apr 2015.

Kroenke, David M., and David Auer. *Database Concepts*. 6th ed. Prentice Hall, 2012. Print.