***Design Document Project 1***

***CMPUT 291 – Group 35 – Project***

a.) General overview of the system:

This application is based upon the usage of the terminal window and command line within UNIX systems. This application is written in python (3.4x). To interact with the system, a user must type in a response when prompted into the command line, and then hit the enter key to submit their answer to the prompt. The application will give details to the user through statements on the terminal window.

Part 1 – Initialization Block:

Where we initialize all the things to connect to the database. When run will connect to the database Gwynne.

Part 2 – Login, Register or Exit decision

Show in the terminal window display commands that a user can input into the terminal window for the login screen. These commands are as follows:

1 – Already registered user, proceed to login page

2 – New user, proceed to registration page

3 – Exit from the application

Entering the number, representing the decision, and then hitting enter will cause the program to accept the answer and then to move on from that point to the next step.

Part 3 – Login (for an old user / a registered user):

A registered user is going to be prompted to enter their username, and then password.

Upon hitting the enter key at this point we will then have the database process their username and password and if they match what we have currently stored for them then we will allow them to proceed, if not they will get an error message and be returned to the login prompt.

Part 4 – Exit (from the application, before login):

This will exit the application without performing any other action. It is assumed at this point we need not maintain the data of people that are not logged in.

Part 5 – Registration (For new users):

On this screen the user will be prompted to enter a username and a password. These (once we have the pair) will then be placed into the database. We then allow the user entry to the main menu screen. However if the username already is in the database then we will warn the user that this is not allowed and return them to the options for login, register or exit screen.

Part 6 – Main Menu (The major part!):

After a successful login. This is going to be the part where the user is granted the ability to select what actions they desire to actually perform. The terminal window will print the list of options and then the user selects the number of the action they desire and then put it into the terminal window and hit enter and then the action is processed and has an action taken in accordance to this! This will take the user to the various functions we are specified to have.

Part 7 – The various unique functions:

This part is dedicated to all of the various functions that the user can select. From the main menu they selected an option and whatever option is selected, they will have the same similar effect. The user in any one of these options will be prompted for various series’ of information that they must supply by making a choice from a list of options or entering in information to the console and then submitting the information by hitting enter. Upon completion the system will return the desired information and then send the user back to the main menu.

Please note: The Record a flight departure and the record a flight arrival are for airline agents ONLY. Users without the airline agent privileges will be denied access to these functions and told so upon attempt.

b.) Detailed Design

Python 3.4 was used, with the intent to make the application text based. So whenever something was desired to be printed on the screen basic print statements were used. Whenever we wanted to gain user input we used the “input” python command, aside from the passwords because we used the more defensive getpass.getpass() command.

When commanding to the databases we created a connection to gwynne.cs.ualberta.ca and then passed this connection around the program. When we actually wanted to perform some sort of action on the database we would create a cursor with it using the connection.cursor() command and evoke SQL commands using it.

Please refer to the functional flowchart for the way that the functions interact in this program. It will help make this make more sense.

THE main function, this will initialize a host of variables such as the connection to the database and useful Booleans that will manage if the user exits the program, and if we run into a cx\_Oracle error we have a block dedicated purely towards creating proper output from the terminal screen. This function evokes other functions (big ones) such as the loginMenu and then the mainMenu functions. It will shuffle essential variables like username and userpassword around through these functions as well. But the bulk of the responsibility is transferred to other functions.

loginMenu function is going to be dedicated purely towards the login screen (trapping the user in a while loop) passed the connection this will display the options the user has to login, register or exit and then through a series of Booleans dictated by the user input will then cascade down to the desired functions. Throughout the login, registration or exit process the user has a series of prompts that then interact with the database to check if the user is valid or not or if the user is actually trying to register an account that does not exist.

Upon successful completion this function then throws the username, password and a successful or not flag to the main function.

mainMenu: This function is going to present the user with a series of options (trapping them in yet another while loop) however this mainMenu function acts moreso as if it were a massive launchpad to other functions that handle their respective functionalities.

c.) Testing strategy. Since our application is a small python application we happily iteratively tested the program through each step of development. We’d implement a small function and test it over and over again to ensure it didn’t work and then implement another. For the SQL queries we isolated the queries into sqlplus and then populated the database tables with test data and then ran the queries there and iteratively tested them until they worked as well. With both a functioning python application and functioning SQL commands we would then integrate the two together SLOWLY and iteratively. One statement at a time. Using this method we could easily isolate where issues arose and tackle them properly before moving onto the next step.

Most bugs found were trivial python bugs or trivial SQL bugs such as syntactical errors, or trying to put a password field into a database that had too small a space allocated for a password. Very easily resolved small minor things- there were no massive Earth-shattering bugs whatsoever. Mindyou if you want an exact quantity that is something not specifically kept by us, likely over 100 small syntactical errors.

d.) Group Work Break-Down Strategy.

The strategy of this project was that there are some fairly obvious breaking points.

#1: We needed a basic framework for an application to function in. As specified by the professor this didn’t have to be fancy or pretty. The goal was to have something that would be functional and then allow us to focus on the primary core goals of the course (Databases!). Thankfully due to my enlightened choice of picking python- this was not difficult to implement.

#2: We needed a design document, so that was another breaking point. Detailed with specific requirements on the assignment page it was fairly straightforward.

#3: Clarity, clarity, clarity. We had to comment code, clearly define functions, ensure indentiations were properly maintained, give appropriate names to variables and functions. Visual aids like flowcharts also were of assistance.

#4: We had multiple queries, insertions, etc of varying difficulty to implement using cx\_Oracle! How to approach this? We honestly opted to throw some queries at eachother… and if needed helped eachother through them.

Coordination: Constant access to eachother through very open/active e-mails, text messages and face to face meetings. We didn’t need anything further (Eg: Slack) in order to get momentum rolling.

e.) (? Not listed SPECIFICALLY but mentioned) Descisions we made:

Keep the application simple. Our professor stated that we don’t need to make the UI pretty, just that it had to work. So we decided to keep it simple and focus on the real stuff.

Keep it tidy, marks were being given for commented code and a well made designed document- which includes flowcharts.