|  |  |
| --- | --- |
| Name: | Edwin Rodriguez (ED04) |
| ID: | Mustafa Abbasi (D07) |
| Project: | Mini-Project 2 |
| Class: | CMPUT 291 |
| Due Date: | Nov 27, 2015 |
| No Collaboration was done for this project. | |
|  |  |

C291G40’s Review Database System:

A1. General Overview

The primary purpose of the Information Retrieval System(IRS) is to facilitate searching for user-defined queries using the physical layer. This is done by parsing the input of the system (a single file containing one or many user reviews) to simply the input and eventually creating four queryable Berkley DB indices. These indices will then be used as the basis for the user queries allowed by the system.

The IRS is composed of three main phases:

Phase 1: Prompts the user for the review file. Once the file is found, it parses the file and creates four sub-files (reviews.txt, pterms.txt, rterms.txt and scores.txt) containing key information about the reviews.

Phase 2: The IRS sorts the sub-files created in phase 1 and creates four database indices (rw.idx, pt.idx, rt.idx, and sc.idx) using the sorted sub-files.

Phase 3: The IRS will prompt users to enter a query. Once entered, the IRS will try to retrieve all the information that fits the query using the databases indices. Once done, it will prompt the user for another query.



If   
query   
!=   
exit

*Figure A: Flowchart of the data flow in the Information Retrieval System (IRS)*

A2. User Guide

In order to run the IRS, three important scripts need to be executed in the following order:

1. PrepData.py (Phase 1)
   1. This script will prompt the user for the location of the review file.
      1. This will read the review file and create the sub-files (\*.txt)
2. Phase2.sh (Phase 2)
   1. This script will delete all existing indices and it will re-create them using the most recent versions of the sub-files (\*.txt)
   2. If you are unauthorized to run the script due to linux permissions, type in the following command: chmod +x phase2.sh.
3. Phase3.py (Phase 3)
   1. This script will endlessly prompt the user for a query.
      1. The query will return nothing if the query is unable to find any information
      2. The query will return the index followed by the summary of the review of any hit.
   2. To exit, type exit as a query.

(b) a description of your algorithm for evaluating queries, in particular evaluating queries with multiple conditions and wild cards and range searches and an analysis of the efficiency of your algorithm,

(c) your testing strategy,

The testing strategy discusses your general strategy for testing, with the scenarios being tested and the coverage of your test cases.

(d) your group work break-down strategy.

The group work strategy must list the break-down of the work items among partners, both the time spent (an estimate) and the progress made by each partner, and your method of coordination to keep the project on track.

The design document should also include any assumption you have made or any possible limitations your code may have.

Detailed Design of Software:

The detailed design of your software should describe the responsibility and interface of each primary class (not secondary utility classes) and the structure and relationships among them. Depending on the programming language being used, you may have methods or functions instead of classes.

The program is composed of the following classes modules:

1. main.py:
   1. Responsibility:  
      The responsibility of this class is to create the main menu and create the contents of the menu based on the type of user that logged in.   
      It also ensures that once users are logged in to the system, they remain or return to this main menu until they decide to logout.
   2. Interface:

As this is the main arterial of the program, this interfaces with the other 4 components.

This interaction is done within the processMenuSelection method.

* 1. Relationship:

Hub to the entire program.

1. splashScreen.py:
   1. Responsibility:

To direct log in existing users or register new users.

To give the program with sufficient knowledge so that the program may differentiate between normal users and airline agents.

* 1. Interface:

The start() function starts the splashpage and returns true once a user logged in.

getEmail() returns the email of the user

getPassword() returns the password of the user

isAgent() returns true if the user is an airline agent.

* 1. Is accessed by Main.py only

1. booking.py:
   1. Responsibility:

Responsible for making, viewing and canceling all bookings.

* 1. Interface:

makeBooking(email): begins the process of making a booking

cancelBooking(email): begins the process of canceling a booking

listExistingBookings(email): begins the process of listing all bookings

* 1. Is accessed by Main.py only.

1. search.py:
   1. Responsibility:

Responsible for searching for all flight information.

* 1. Interface:

Main(email, connectionString): starts the process of searching for flights

* 1. Is accessed by Main.py only.

1. updateTime.py:
   1. Responsibility:

Responsible for updating actual arrival and departure time of scheduled flights.

* 1. Interface:

updateTime(connectionString, TypeOfFlight): Updates the actual departure or arrival time of a scheduled flight. TypeOfFlight is a string containing “Arrival” or “Departure”.

* 1. Is accessed by Main.py only.

Testing Strategy:

The program was developed under control database conditions where the database had a small data sample. This allowed us to manually verify the output of each of the queries required by the program. Manual verification of the contents of the database was also done to verify updates or deletions from the database that occurred in the program.

Basic input and output from the program was also tested to insure that the program behaved as expected. This led to the development of error checking methods for data inputted from the user. This can mostly be seen in the case of the splash menu, the user menu and date entries into the program.

There was also integration testing which occurred to insure that the program worked after combining all the modules/classes together.

The tests had a small data coverage overall and regression testing ensured that the program maintained its working integrity after big changes occurred.

Bugs:

Most of the bugs that were found within the program were basic input/output bugs based on improper handling of user inputs. Some basic bugs were also found that led the user to input invalid data into the system which led to the program being stuck in an infinite loop. Some bugs also occurred through the translation of user inputs into Oracle queries.

Group Break-Down Strategy:

The project was divided based on the provided requirements. To improve coordination between the group, the separation of tasks revolved around the exclusive ownership of classes (files). This meant that an individual had full control over their work. The project relied on git and github as a way to give the project version control and the ability to allow the group to work with the most current code. For a close look at the github repo please see the following link:

https://github.com/c291g40/project2

Break-down:

|  |  |
| --- | --- |
| Edwin | Mustafa |
| Splash Screen and familiarizing with Python:  6.5 hrs – Oct 20 | Search Requirement: 10.5 hrs |
| Main menu (main.py): 2 hrs – Oct 21 | Airline Agent requirements: 2 hrs |
| Logout feature: .5 hrs – Oct 21 | Fixing bugs: 4 hrs |
| Listing Bookings: 2.5 hrs – Oct 22-23 | Implementing Return Flights: 4.5 |
| Making Bookings 1way and 2 way bookings:  4.5 hrs – Oct 24 |  |
| CancellingBookings: 1 hr – Oct 24 |  |
| Merging project: 1 hr – Oct 26 |  |
| Documentation:3.5 hr – Oct 26-27 |  |
| Total: 21.5 hours | Total: 20.5 hours |

Other Design Decisions:

In order streamline the program, the database credentials required command-line arguments. This was done so that the program can quickly recover in-case of an unexpected program termination as the database username and password will hinder the recovery of the program.