# CMPUT 291 MINI PROJECT 2

# DESIGN REPORT

## USER GUIDE:

This user guide helps the user to run the application from the beginning to the end. The application takes in an xml file as an initial argument and form 4 '.txt' files , which are different filtered out data based on the given specification in the project. The application then takes the .txt files as in input makes '.idx' files which are index files of these '.txt' files. The user then enters the chooses full/brief output. Full will give the entire email where as Brief will give jus the row id and the subject.

## LAUNCH APP

Type the following command : `python3 PHASE1.py` in the terminal after navigation to directory with all the files. This command will prompt the user to enter the .xml file .


The users enter the desired.xml file. And presses the enter key This will create the 4 text files email.txt,recs.txt,terms.txt and dates.txt . These are 4 different txt files based on different specification.


Now the user is supposed to the type in the command as given above. This will create the 4 .idx files da.idx,re.idx,em.idx. and te.idx  based on either btree/hash access methods.The user the enters the following command:


Now the user can enter the query as desired and the type of output desired.

## ALGORITHM and DESIGN:

The application is broken down into 4 parts the . “PHASE1.py” uses cElementree as one of the main libraries. This file converts .xml to .txt files for the given specification. “phase2.py” uses python os library to perfom command line fuction sort: to sort the text file and produce and output file of the same name., perl to execute the break.pl script for the given text files and db_load the form the idx files based on different access methods and insert into db. Interpreter.py breaks down the input query into 2-d array.

Main.py is the front end of the application and calls the interpreter.py and makes a connection between the input query and the output by getting the different row ids for different tags using either te.idx,em.idx or da.idx depending on the tag and then finding their intersection and uses re.idx to give out the corressponding email.

**TESTING:** was done using the example inputs given on eclass:

1. *subj:gas*
2. *subj:gas body:earning*
3. *confidential%*
4. *from:phillip.allen@enron.com*
5. *to:phillip.allen@enron.com*
6. *to:kenneth.shulklapper@enron.com to:keith.holst@enron.com*
7. *date:2001/03/15*
8. *date>2001/03/10*
9. *bcc:derryl.cleaveland@enron.com cc:jennifer.medcalf@enron.com*
10. *body:stock confidential shares date<2001/04/12*

and many other .

## GROUP WORK AND TASK BREAKDOWN:

Yanlin : completed phase1.py , debugging phase2.py, and main.py. TIME > 9hrs

Shohan: completed Interpreter.py, debugging main.py and testing. TIME> 9hrs

Sirjan : completed phase2.py , main.py , debugging and testing. TIME>9hrs

The group meetings took place in the CSC building during mutual free time and on weekends