ECE 322 Project Proposal

Group members

Zachary Schmidt, CCID: zschmidt
Adib Nasreddine, CCID: nasreddi
Nam Son Pham, CCID: npham
Ajoku Nnamdi, CCID: najoku
Huu Dinh Ky Nguyen, CCID: huudinhk
Levi Thomas, CCID: llthomas

Project Repository: https://github.com/leviLThomas/CMPUT-291-Project-1

System Under Test

We will be testing a project developed by Levi Thomas, Huu Dinh Ky Nguyen, Tyler Singh and Ivan Stolypin for CMPUT 291 - Introduction to File and Database Management. The project is written entirely in Python and mimics aspects of Twitter such as composing tweets, searching for tweets, following users, etc. The project uses the InquirerPy package (https://github.com/kazhala/InquirerPy) to implement a Terminal User Interface. This allows the user to navigate and select options using the arrow keys and enter button, additionally the program only allows the user to provide text input when prompted. Data associated with the program is stored in a SQLite3 database with the following relational schema:

- users(usr, pwd, name, email, city, timezone)
- follows(flwer, flwee, start_date)
- tweets(tid, writer, tdate, text, replyto)
- hashtags(term);
- mentions(tid, term)
- retweets(usr, tid, rdate)
- lists(lname, owner)
- includes(lname, member)

Testing Strategies

We intend to use all major testing strategies covered in the course to test the application. From black box testing, we will invoke error guessing, checklists based on requirements from the project specifications, input domain testing (equivalence classes and boundary value analysis including EPC and weak nx1), and decision tables. Many of these methods will blend together when we form our test suite, but test cases that clearly fall in one category will be well-marked. Since most of the input to the application are strings, it is not very practical to use pairwise testing since there are an infinite number of strings to consider. To begin white box testing, we will develop a high-level control flow graph for the program. The major methods we will focus on are statement coverage, branch coverage, all p-uses, and all c-uses (to keep things simple). Additionally ChatGPT will be used to generate sample data, test cases for the SQL queries being used and SQL injection handling.

We will also implement mutation testing to evaluate the effectiveness of our test suite(s).

When developing test cases/the code for our test suite, we will rely heavily on ChatGPT as this is the primary goal of this project. We intend to write a minimal amount of code ourselves, instead leaving the implementation details to ChatGPT. Of course, there will be a significant amount of editing/re-tooling of what ChatGPT outputs since it is not always 100% correct. We will report on principles and practices we find helpful when interacting with ChatGPT in our report and presentation.

Each group member will focus on developing test cases for one or two testing strategies.