Github Page For More Information: https://github.com/abenezerBelachew/DBGroupProject1

Task

The goal of this assignment is two folds: (1) to teach the use of SQL in a host programming language, and (2) to demonstrate some of the functionalities that result from combining SQL with a host programming language. Your job in this project is to build a system that keeps the enterprise data in a database and to provide services to users. You will be storing data in a SQLite database and will be writing code in Python to access it.

User Guide

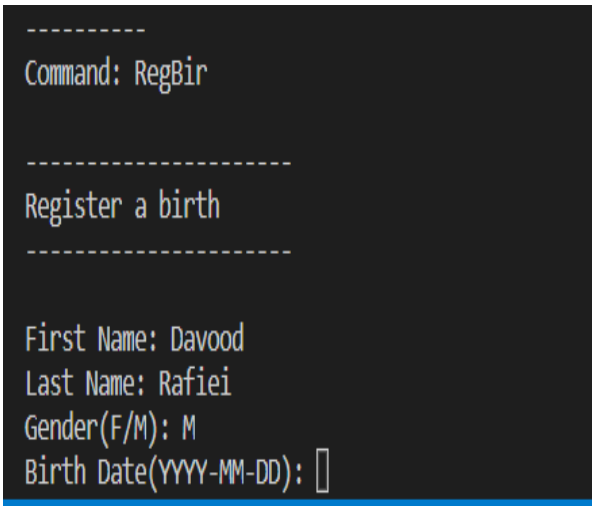
Run the program by typing

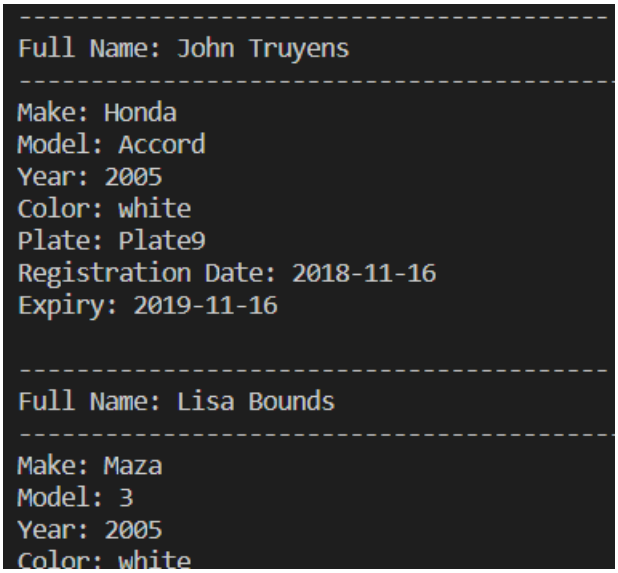
1. "py main.py [database_name]" where database name is the database to be tested or queried on
2. Write in credentials (uid) and (pwd).
    a. Note: The password will not be visible when you type it. Once you're done typing the password press enter.
3. Among the provided commands, type in the command of your wish and provide required responses.
4. If everything goes well, you should see a message saying "All done :)"
5. You can check the database to check to see updated or added changes.

```
py main.py testdb.db
['REGBIR', 'REGMAR', 'RENVEH', 'PROBIL', 'PROPAY', 'DRIABS', 'ISUTIC', 'CAROWN', 'HELP', 'QUIT']
Your uid: user1
Your Password:
----------------------------
You're logged in.
----------------------------
User in main:  user1
----------------------------
```

```
------------------------------
User in main:  user1
------------------------------
List of Commands:
RegBir to Register a Birth
RegMar to Register a Marriage
RenVeh to Renew a vehicle
ProBil to Process a bill of sale
ProPay to Process a payment
DriAbs to Get a driver abstract
IsuTic to Issue a ticket
CarOwn to Find a car owner
Help to see available commands
Quit to quit

----------
```





Sample result

Testing

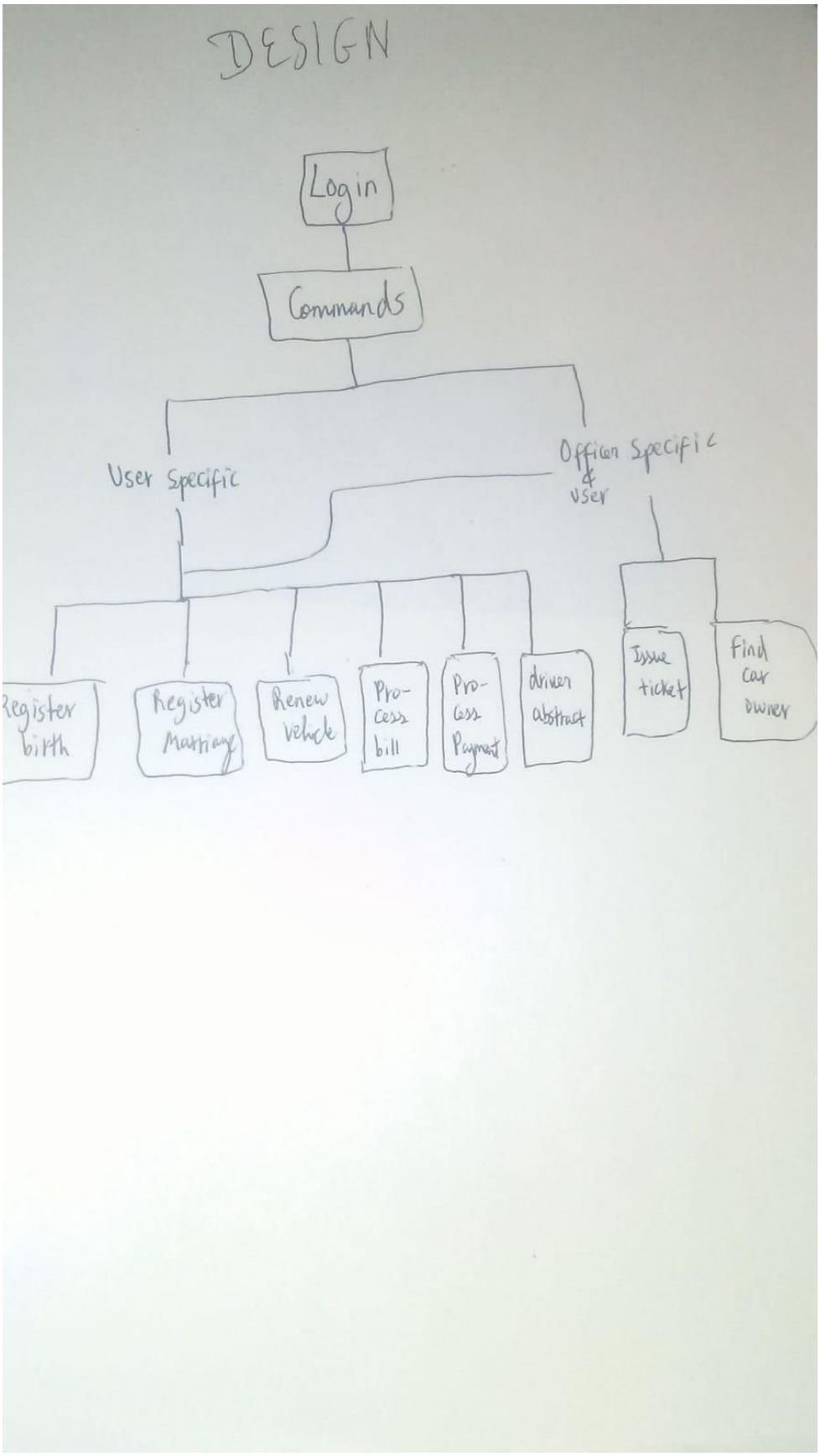
For testing, we made already given test databases from previous assignments and databases we made on our own for the purpose of testing. The files can be found in the submission folder. We thought of different scenarios that the code may break in and have tried preventing and 'leakage' to the best of our abilities.

Although we are a group of two and need not have done the securing of the system from SQL injections, we did it anyway. The system is built so it can't be withstand sql injections.

Group Work Breakdown Strategy

We are a group of two. We classified the work based on the list of commands we had to implement. We both consulted each other and tested each others work. We used github as a primary source to keep track of what other person was up to. We also had a chat system in which we regularly communicated about the work.

Design


DESIGN
Login
Commands
User Specific
Officer Specific & User
Register birth
Register Marriage
Renew Vehicle
Process bill
Process Payment
driver abstract
Issue ticket
Find car owner

new a vehicle registration

Existing reg-number : ?

Are you sure you want to renew your reg_number? Yes

└ new expiry date (one year from today's date if current registration has expired or expires today)

Existing reg-number: ?

Are you sure you want to renew your reg-number?

Yes

if expiry date has passed or expires today:

new-expiry-date: one year from today

else:

new-expiry-date: one year from current expiry date

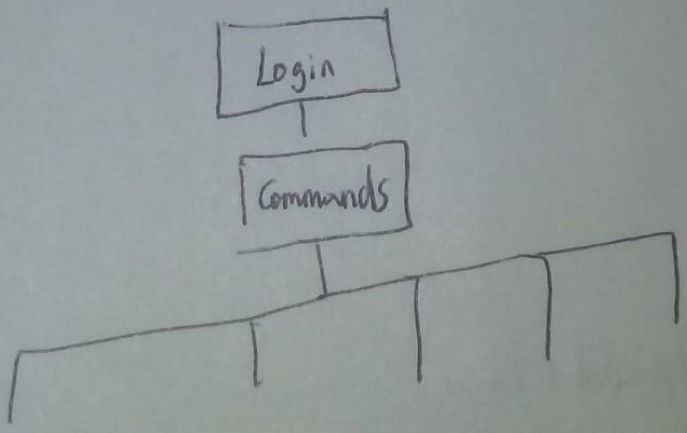


Table payments: Keep all payments made towards tickets

Users: list the users of the system

Users (uid, Pwd, Utype, fname, lname, city)

'a' agent 'o' officer

(location of registing agent or the traffic officer

User opens app

Is asked to login:

Uid:

Pwd:

Must be able to logout

User opens app

Please provide db to be used:

Login

1. Register a birth

- ask for
  - first name
  - last name
  - gender
  - birth date
  - birth place
  - First & lname of parents
  - registration date: today
  - registration place: user's city
- assign unique registration no. to the birth record
  - address: same as mothers
  - Phone: same as mothers
- If parents not in db, system should get fn, ln, bd, bp, addd & Phone.
  - for each parent, any column other than fn and ln can be null if not provided

~~birth (Persons~~

birth → Persons → Us

birth → FK (fn, ln) ref Persons
FK (f_fn, f_ln) ref persons
FK (m_fn, m_ln) ref persons
PK (regno)

Users → PK (uid)
FK (fname, lname) ref persons

Persons → PK (fname, lname)

Documentation

Every function has a docstring describing the use of the function.

All command functions take in database and user as arguments. The database is used to write into or update the main database file given as an argument at the start of the program.

Bugs We Faced

Most of the bugs we faced were mostly typos in the source code or accessing a different table and getting the results from them or even misreading the assignment text.

We were able to fix all of them to the best of our abilities.

Group Work breakdown [done by]

System Functionalities

- ☑ Login Screen [Abenezer]
- ☑ Register a birth [Abenezer]
- ☑ Register a marriage [Abenezer]
- ☑ Renew a vehicle registration [Abenezer]
- ☑ Process a bill of sale [Arthur]
- ☑ Process Payment [Arthur]
- ☑ Get a driver abstract [Arthur]

Traffic officers

- ☑ Issue a ticket [Abenezer, Arthur]
- ☑ Find a car owner [Abenezer]

General

- ☑ String matching [Abenezer, Arthur]
- ☑ Error checking [Abenezer, Arthur]
- ☑ Counter SQL injection attacks [Abenezer]
- ☑ Non-visible password when typing [Abenezer]