Table of Contents

1 Welcome to ARS

ARS (Auto Registration System) is a program written in Python3 for use with oracle database systems. The system allows a user access to the whole database defined by a username and the corresponding password (e.g. [username]/[password]@gwynne.cs.ualberta.ca:1521:CRS will lead to the database stored at the given host). The interface allows you to login and access the various apps needed to maintain the ARDB (Auto Registration Database).

2 Usage

ARS requires no installation and minimal setup. Simply unzip the project package into the directory you wish to run and use ARS in and type:

python3 ARS.py

to begin running ARS.

2a Main Window and Login

The main window consists of 6 buttons in order: New Vehicle Registration, Auto Transaction, Driver License Registration, Violation Record, Search Engine, and Login. To use any application the user must log in. The login page is a simple username and hidden password entry with a login button. After logging in the user is free to use any application in the program, but they may not login again unless they close the program. **Note: please only use the X in the top right corner of the ARS interface to close the program,** this ensures that the connection to the database is properly terminated when exiting the application. ARS’s main interface calls each of the five major programs detailed below.

The main application file source code is stored in ARS.py

2b [APP1] New Vehicle Registration

In APP1 the user is presented with several entries to register a new vehicle. Many of the entries will ensure you are trying to submit data according to constrictions of the create table statements. There is also a “Add New Person” widget, which allows you to create a new person in the database for use with the application(s).

APP1’s source code is stored in ./apps/new_vehicle_registration.py

2c [APP2] Auto Transaction

APP2 allows a user to record auto transactions that occur. The current system date will be preloaded into the “Sale Date” entry to allow the user to easily follow the format or quickly use today’s date if needed. Otherwise the app will allow a primary owner to sell their vehicle(s) to new owners and ensures all corresponding tables in the database are up to date. This app makes use of the “Add New Person” widget.

APP2’s source code is stored in ./apps/auto_transaction.py

2d [APP3] Driver License Registration

In APP3 the user can enter information to create a new license. The app ensures the unique constraint on SIN and License # are maintained when submitting data. This app also makes use of the “Add New Person” widget. You can open the photo file you have selected for the license and make sure that the picture is appropriate. There is also a small conditions widget to add new conditions to the database for use with the app, and the “?” button will let you see all conditions in the database.

APP3’s source code is stored in ./apps/driver_license_registration.py

2e [APP4] Violation Record

The user can use APP4 to enter a Violation that has been issued. The format for date is preloaded and pressing the “?” beside the “Date Issued” column will allow the user to set the entry to the system time. Pressing the “?” by the vType entry will allow the user to pull up a list of all the types of violations and their associated fines. You are allowed to submit a violation without a description and without a violator SIN. If you choose to submit without a violator SIN, the application will place the ticket on the primary owner of the entered VIN (Vehicle Identification Number).

APP4’s source code is stored in ./apps/violation_records.py

2f [APP5] Search Engine

APP5 presents the user with several text entries and buttons corresponding to a specific search. You type the appropriate search term into the entry above the button specifying the search you want. The three types of searches are:

i. Searching for personal information (i.e. Address, Birthday, driving class, etc) by searching on an exact name or license number.

ii. Searching for violation history (i.e. Ticket Number, Issuing Officer, location, fine, etc) by searching on a SIN or license number.

iii. Search the vehicle history (i.e. Number of violations received with the vehicle, the average sale price, and number of sales) by searching on a VIN (Vehicle Identification Number)

Note: These searches are case un-sensitive.

APP5’s source code is stored in ./apps/search_engine.py

3 Development Process

This section details elements about the development processes.

3a Code Development

3a i Design Choices

We went with a GUI approach because it is easier for a user to manipulate and allows the user to easily respond to any errors in their data entry.

We wanted to keep a similar feel between all the applications, most feature a horizontal priority with two columns for entry boxes and submit buttons in the bottom right corner. This allows a user to quickly pick up the basic scheme of any app they may use.

We used a numbering system for our error messages to easily find the location of the source of an error message in our program. For example, a login error on APP5 is 0xa5-1 (general form: 0xa#-#, where the first number is the app and the second is the error number).

We ensured that the SQL statements were sanitized using a Python dictionary entry method. This allows for the user to search for strings that may contain “’” and prevents the user from performing SQL injection.

We check the database for your entries after trying to enter them, if there was a submission failure. This helps keep data entry transactional because if we checked beforehand, the data could have changed between the time we check and submit.

3a ii GitHub

Our code was stored in a private repository on GitHub, we used feature branches to develop each application independently. This allowed us to keep a consistent central file location and maintain a version control in case of accidental data deletion.

3a iii Work Breakdown

APP1, APP2, APP3, NewPersons App were produced by Devon Upton.
APP4, APP5, the ARS.py, and tableWidget.py was developed by Bennett Hreherchuk.

3b Testing

Both team members had a general scheme to building and testing the applications. It followed this pattern:

1. Start with building the fundamental frame for the application in TKinter, test that each entry and button are accessible and call the correct commands.
2. Build verification for each entry as defined by the CREATE TABLE statements in *p1_setup.sql*, ensuring no entry will violate these terms.
3. Build SQL statements to send to the database. Test that all error codes are caught in case of an error and that the database properly rolls back when there is an error.