**Underwriter Application Architecture**

My underwriter application is comprised of 4 main tables which represent all of the information needed to store and manage a user’s details regarding car insurance. The table are: Person, Claim, Vehicle and Policy.

The Person table stores the user’s title, forename, surname, email, date of birth, telephone number, address details, license type, license period occupation, number of claims in the past 3 years and a randomly generated string. The address details are split into four separate fields: Address 1, 2, 3 and Post Code. All but telephone number address 2 and address 3 are compulsory fields, and the randomly generated string is generated upon adding a record to the Person table.

The Claim table stores the claim date, value, type, description and the primary key (id) of the related Person table entry for each claim the person has made in the past 3 years. All but the claim description field are compulsory.

The Vehicle table stores the registration, annual mileage, value, parking location, policy start date and the primary key of the related Person table entry. All fields in this table are compulsory.

The Policy table stores the excess, breakdown cover, windscreen cover and the primary key of the related entry in the Person table. All fields in this table are compulsory, with the excess field being generated when the user has filled all details out.

According to the specification; the Person, Vehicle and Policy tables could have been merged into one due to the one to one nature of the required data storage method. However, this would not adequately represent the real world where one person can own multiple vehicles and take out multiple policies on these vehicles, so for the purpose of this assignment I have designed the database as if the relationships between these three tables could be one to many rather than one to one.

The Underwriter application takes advantage of HTTPS for receiving requests from browsers or broker applications by being run using the command “bundle exec ruby bin/secure\_rails s”. The “bin/secure\_rails” file takes advantage of an RSA certificate to encrypt the connection using a 1024 bit encryption.