Table of Contents

[Introduction 2](#_Toc479672878)

[Purpose 2](#_Toc479672879)

[Overview of Entity-Relationship Diagram 2](#_Toc479672880)

[Entity-Relationship Diagram 3](#_Toc479672881)

[Relational Model 4](#_Toc479672882)

[SQL Code for Table Creation 5](#_Toc479672883)

[SQL Code for Pre-Filling the Tables 7](#_Toc479672884)

[Tables Created 7](#_Toc479672885)

[Registration of Clients 9](#_Toc479672886)

# Introduction

## Purpose

This report will describe the planning process and the creation of the initial tables that the client’s database will be created using. It will also show the login and register screens and them working, adding users to the database and allowing them to sign in. The ER Diagram that is being used to base the entities and relations on will be shown, along with the relational model that will be created using the ER diagram as the base. The relational model that is created will then be used to create the tables for the database. There will be four tables that will be pre-filled, the car, type, instructor, and admin tables. These will be pre-filled as they are tables that the client will fill before any data is put into the database. The tables will be included along with any other SQL code that is used to create the database, all documented where necessary, to show the client what is happening with each of the steps in the creation. There will also be some screenshots of the database with some prefilled slots to show how it would work for the client in practice. The registration and logging in of users will be shown as a series of screenshots that will be annotated explain how it works.

## Overview of Entity-Relationship Diagram

Note the implied workflow for the processes of registration, assigning of instructors and cars, booking and billing.

**Client:** stores the clients’ information

**Type:** information about the type of instruction that is available (pre-filled). Clients select which instruction they wish to enrol for when they register.

**Document:** documents will be held outside of the DB but linked. Documents may be of type ‘bill’ or ‘certificate’. They are provided to clients over time by the administrators, available when the client has logged in.

**Admin:** contains both admins and manager (pre-filled)

**Timeslot:** is created by admins a month or more in advance, key is id + time together so that for each time, the id counts through the available slots at that time (e.g., only one per time on Saturday and up to four during the week)

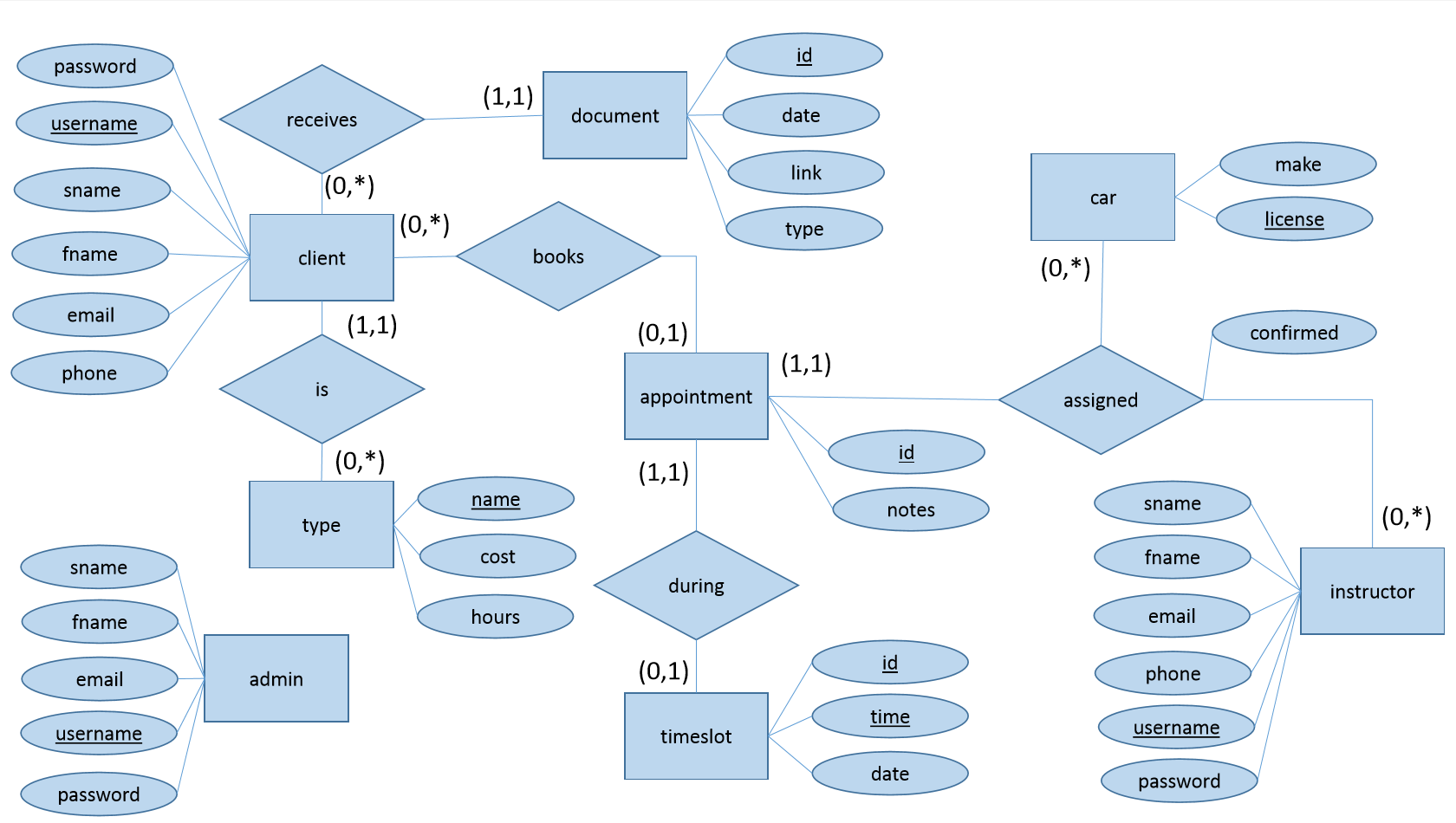
**Instructor:** contains all four instructors (pre-filled)

**Car:** contains all five cars (pre-filled)

**Appointment:** this is the appointment that a client books and for which an instructor leaves notes on how the driving went

**Assign:** Instructors indicate for which timeslots they are available or not available by creating assignments. These get confirmed by admins. Admins link cars to the instructors for a given time (= their assignments). Once a confirmed instructor and a car is available for an assignment, it can be offered to the clients to be booked.

## Entity-Relationship Diagram



# Relational Model

**client**(username, password, fname, lname, email, phone, type\_name)

**admin**(username, password, fname, sname, email)

**instructor**(username, password, fname, sname, email, phone)

**document**(document\_id, date, time, link, type, client\_id)

**drivertype**(name, cost, hours)

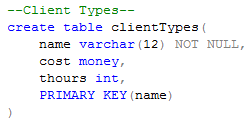
**car**(licence\_no, make)

**timeslot**(slot\_id, time, date)

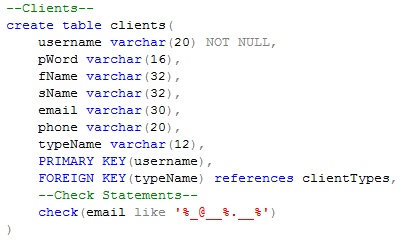
**appointment**(appointment\_id, notes, slot\_id, client\_id, car\_id, instructor\_id)

# SQL Code for Table Creation

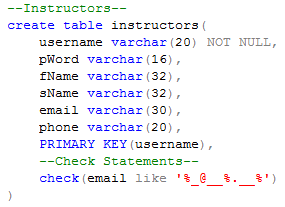
**Client Type:**



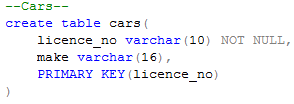
**Clients:**



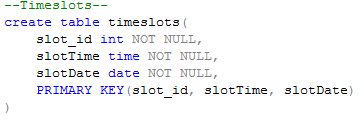
**Instructors:**



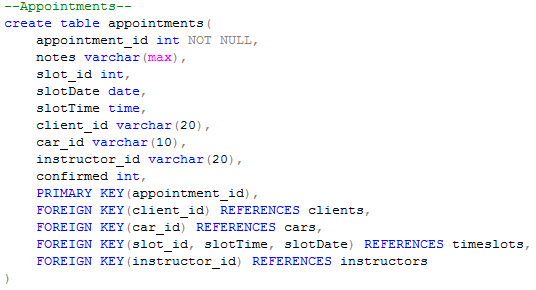
**Cars:**



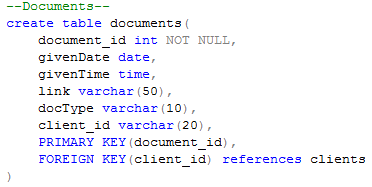
**Timeslots:**

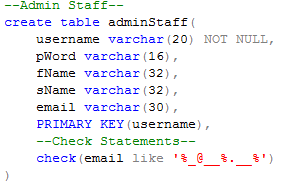


**Appointments:**



**Documents:**



**Admin Staff:**

# SQL Code for Pre-Filling the Tables

The tables that need to be pre-filled are:

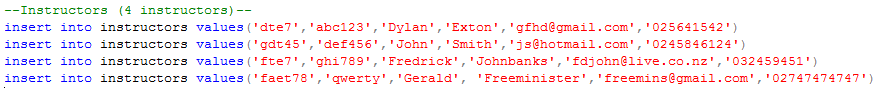
* Instructors
* Client Types
* Cars
* Admin Staff

All data has been fabricated and doesn’t represent the true data that the database will hold, i.e. the Instructors and Admin Staff are made up.

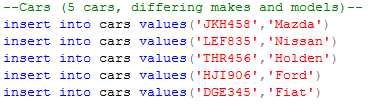
**Client Types:**



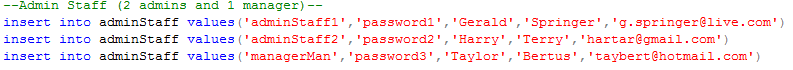
**Instructors:**



**Cars:**



**Admin Staff:**



## Tables Created

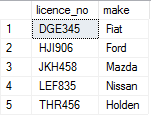
**Client Types:**



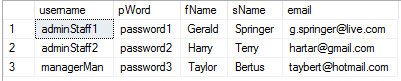
**Instructors:**



**Cars:**



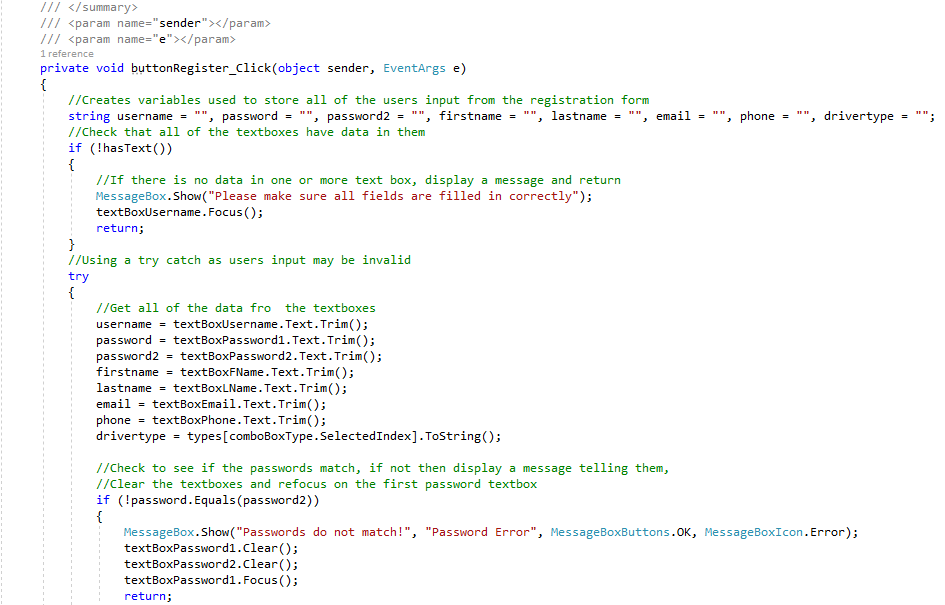
**Admin Staff:**

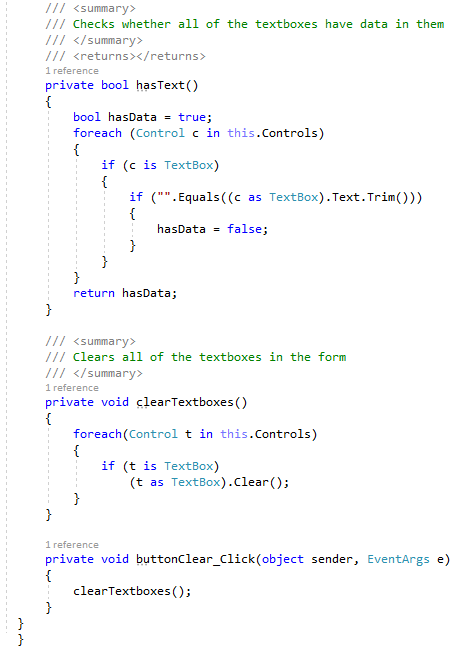


# Registration of Clients

The clients must be able to register for the database to be useful. This registration program does most of the checking, making sure there is information where it is needed. The database checks if the email is in the correct format, and that the primary keys are not null, but the registration program makes sure all the rest of the attributes have valid information in them.

## C# Code for Registration





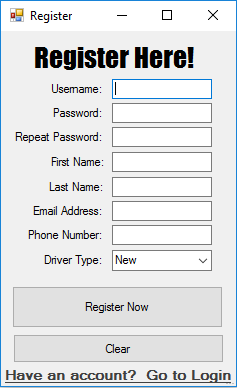
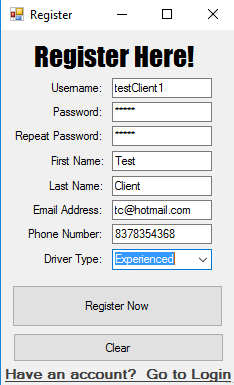


Figure Registration Window GUI

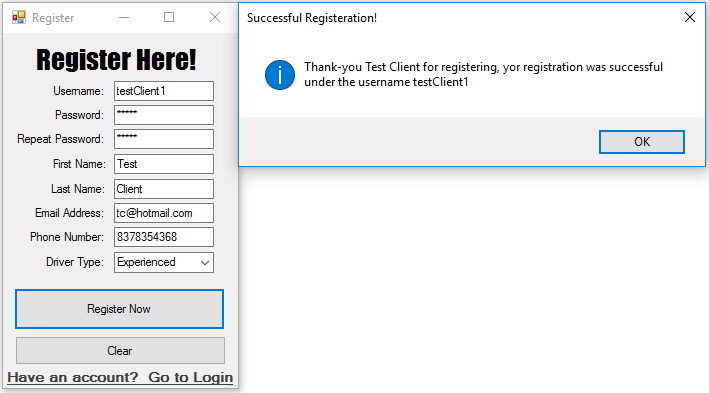
## Walkthrough for Registration of Clients

Table of Clients before the user has registered:

Client adds the relevant personal details into the register screen:



Client clicks on the “Register Now” button once all information has been added

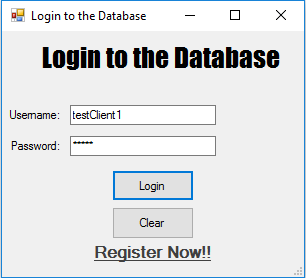


Once the client has clicked on “Ok” in the Message Box, they are taken back to the login screen. Their information has been added to the database as shown here in the table from SQL Server:

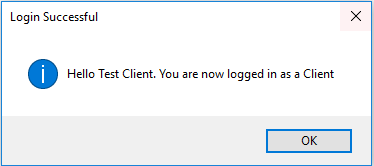


## Walkthrough for Login

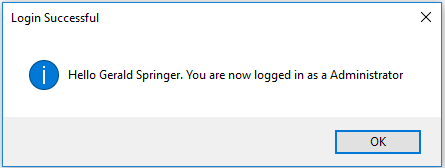
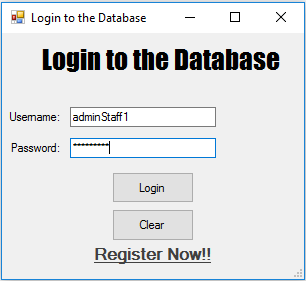
For all account types, the username and password are checked against the “adminStaff”, “instructors” and “clients” tables. This assumes that there is virtually a 0 chance that a client and a member of staff will have the exact same login credentials. This means that all users of the database can login using the same screen but then have different privileges in the database

 This is using the client we just created.

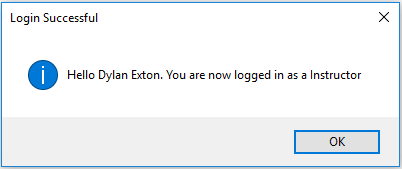
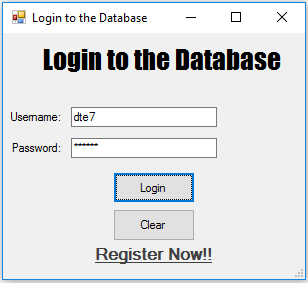
The user will then get a message box pop up that states their first and last names and the account type they are logged into



**So for an admin staff account it would look like this**

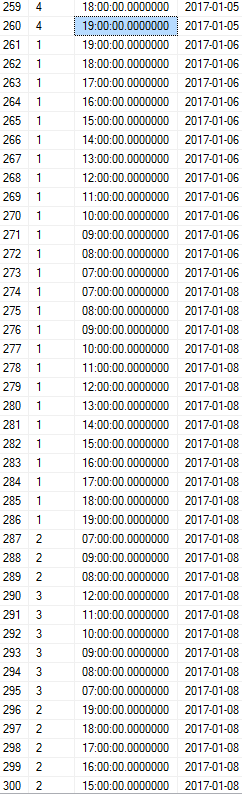
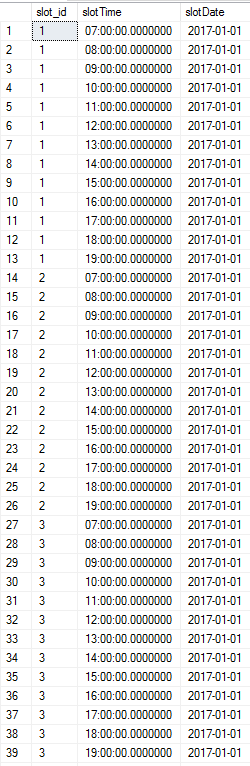


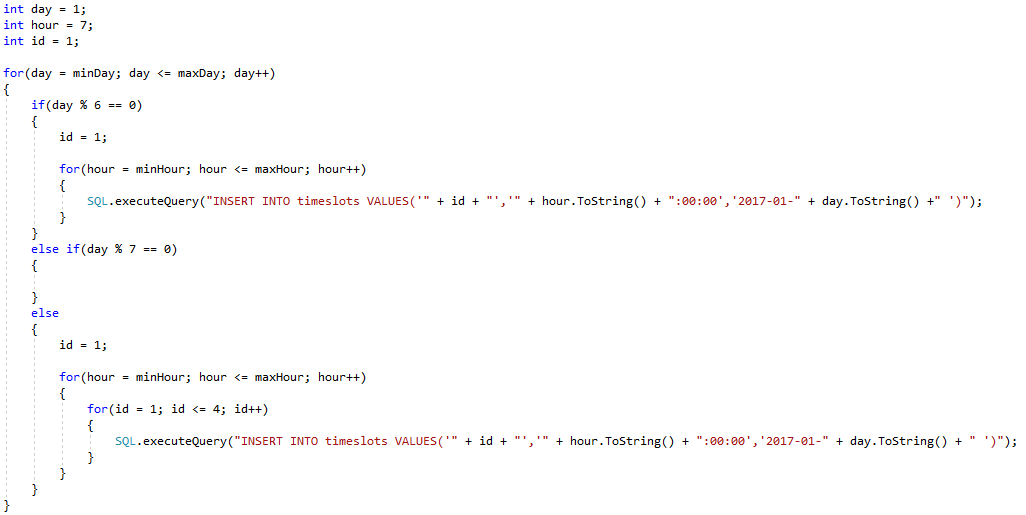
**And an instructor would look like this**



# Filling Timeslots for a Month

To do this I created a C# program to enter all the available timeslots for the month of January 2017. This assumes that the 1st of January is a Monday as it skips every 7th day, as Sunday doesn’t need to be entered into the database.



This is the code that I wrote to add all the timeslots in for the month. This would need to be altered to create all the timeslots for the year given each month’s differing start day, changing the day at which a Sunday falls and the differing number of days per month.