# **Problem D**

Create a rest service that uses the database design from problem C.

## Expected Deliverables

1. There must at least be a client application and a server application. The client application can be as simple as a set of integration tests, but could be any of the following:
   1. Console application
   2. Windows forms application
   3. Web forms application
   4. MVC application
   5. WPF application
2. You can use any third party technologies you’d like to aid in development. Provide instructions on how to install them if it’s not obvious.
3. The required operations the web service should support are
   1. Enroll policy
      1. Should generate a policy number utilizing the logic described in problem Problem C, item e.
   2. Cancel policy
   3. Add a pet to the policy
   4. Remove a pet from the policy
   5. Transfer a pet to a different owner.
4. The data source need not be more complicated than an in memory storage facility. You can use localdb or some other technology if you’d like.

**Answer:**

Unfortunately, due to time constraints I won’t be able to finish this by Friday (1/20/2017). I decided to use the .Net Core version of WebAPI and it slowed me down a bit due to subtle differences (I’m used to WebAPI 2 and Java APIs using SpringBoot). I wanted to use Entity Framework (EF) because it does some speedy scaffolding but even that has undergone some subtle changes to run on the .Net Core platform.

**What I do have:**

1. I have all the stored procedures required to cover points a-e above.
2. I have controllers and repositories set up to call these sprocs. The API is about 85% complete.
3. I didn’t do TDD for this solution so I was planning to write the unit tests. So, there are not unit tests.
4. Some of the API is usable in the Policy controller. I think only the Enroll method works. I was getting ready to wire up all the rest of the methods and write unit tests for them. Some tweaks need to be made to the routes and such and generally clean up, and testing of course.
5. I was going to slam out a simple WinForms application to the call the API methods, but alas, I have not. ☹
6. I have an updated schema script in the TrupanionAPICore folder which includes all the new sprocs I added today.

e

**If you want to test the API:**

1. First, create the database using the SQL Script (V2) included in this drop. Note, I’ve also included an MDF so you can just attach it to your SQL Server 2012 instance and you’ll be up and running right away….your choice. **Make note of your server name because you will need it to update the connection strings in the Policy and Pet repos. The connection string is trusted ls to make your life even easier.**
2. Related to the above, I of course, would not ordinarily sprinkle connection string code all over the place. I wanted to use DI in conjunction with the app properties JSON config file.
3. Open the TrupanionAPICore solution and Run it either in Debug of Release mode. A page should come up which shows an erroneous endpoint. It contains the port number that is being used. If you want to try it out with PostMan of another Rest client you can try:

<http://localhost:[portnumber]/controller/resource>.....

Ex:

[http://localhost:1626/api/policy/1/"USA](http://localhost:1626/api/policy/1/%22USA)"

The URL about is a PUT call to Enroll a customer (see the Enroll method in the Policy controller)

**Conclusion:**

Thanks for allowing me to attempt to solve these problems. I really wanted to get ALL of them done. If you have any question please let me know.

Hopefully we can meet for an interview soon.

Thanks,

Kurt