# Tecsys Retail Coding Exercise

The goal of this test is to show us which techniques, styles, patterns and practices you use when developing web applications, not how much you can code. It should take around an hour to finish Step 1 while Step 2 is more open ended. If you spend more than 3 hours total on this, you are probably spending more time than necessary. You can use any online resources (Google, MSDN, etc.) to complete the test. If you require any clarifications on the requirements, do not hesitate to email me at [john.hinz@tecsys.com](mailto:john.hinz@tecsys.com)

Once you are done, email the link to it online (e.g. Github, Dropbox, et. al) to [john.hinz@tecsys.com](mailto:alexander.sjahlim@tecsys.com) specifying which part of Step 2 you have chosen to implement. We will confirm once we receive it. If we believe it is acceptable, we will schedule an in-person interview. When you come in for your interview, plan to spend around 15 minutes walking us through your code explaining what you did and why.

## Guidelines

* Treat this like production code. Although you will be judged on how much you get done, we will be looking more at the quality, craftsmanship and creativity of what you have delivered.
* Treat this like a shippable project.
  + The first thing we are going to do is start the application and see if it works (black box testing). A non-working application will be heavily penalized.
  + The second thing we are going to do is look at the quality of the code in terms of design, cleanliness, maintainability and performance (white box testing).
* If there is something you think should be done, but you don’t have time, please add a TODO comment. E.g.
  + // TODO: Should add better error handling here
  + // TODO: This is a performance bottleneck and I would normally use XYZ
* There are at least 3 ways to have access to the database:
  + Mount the included wingtiptoys.mdf and wingtiptoys\_log.ldf under SQL Server 2012 Express (or any other edition)
  + Put the wingtiptoys.mdf and wingtiptoys\_log.ldf in the App\_Data folder and use a local connection string
  + There is also a script file that you can run to create the database, with data, under SQL2008 or SQL2008R2 if you don’t have access to a SQL 2012 server.

## Background

There is a single HTML page for Wingtip Toys. Your job is to make the product catalog dynamic and load it from the database.

## Step 1 – Setup an ASP.Net Site

Do any **one** of the following

### ASP.Net

It is up to you on whether you want to use WebForms or ASP.Net MVC. Load the cars on the right hand side of the page from the Product table of the Wingtiptoys database.

* You should only return cars and not any other product categories
* Load the product name, unit price and image. You should also include the ProductID as a hidden field.
* Do not change the database. We will use your source code and connect it to a local database at our office to verify it.

### Single Page Application

Use a JavaScript SPA framework (e.g. Angular, Ember, Backbone, etc.) combined with a Microsoft technology based RESTful web service (e.g. APIController, WCF, etc.). Load the cars on the right hand side of the page from the Product table of the Wingtiptoys database.

* You should only return cars and not any other product categories
* Load the product name, unit price and image. You should also include the ProductID as a hidden field.
* Do not change the database. We will use your source code and connect it to a local database at our office to verify it.

## Step 2 – Enhance the Product

Do any **one** of the following. Pick whichever you feel is most relevant to the position you are applying for or your skills:

### Search

1. Add a search textbox. This will search across the product names and descriptions and return back a list of the items.
2. This should include validation to make sure the search has at least 2 characters.
3. Add unit tests to make sure the search validation works as intended

### Add To Cart

Using JavaScript along with any JavaScript libraries you prefer, implement the following:

1. Add an Add To Cart link that sends a request to the server which includes
   1. A CartId as a GUID. This can be a hardcoded GUID and does not have to be generated on the client.
   2. The Product ID
2. The server should take the request and insert/update the record in the CartItems table.
3. Add unit tests that that validate the server reacts properly to incoming input

### Dependency Injection

1. Use any dependency injection pattern or framework to separate the website from the data access.
2. Add unit tests that test the website logic, without communicating with the database.