XERO – Refactor Task

# Name: Rakesh c. Mehta

Email : [r\_mehta00@yahoo.co.nz](mailto:r_mehta00@yahoo.co.nz), [rakesh22.mehta@gmail.com](mailto:rakesh22.mehta@gmail.com)

Phone: 021 214 5971

# ***Details to Improve this Project***

I have made changes to the existing project with best practices and recommendations in mind, as I still don’t have correct context on how and who will use the system I have kept it at a very high level.

* Developed this solution under VS2017 using latest framework and toolset.
* As usual I created Interface IProductTask where I created signatory for all possible task. This is the way I have an advantage to develop solutions without any cyclic dependencies, possible to add new functionalities & easy to maintain & update solution
* Rewrite controller “Product” which can take care of all actions with proper comment as existing Product controller methods are bit confusing & not completed.
* Added new module ProductTasks.cs which executes all actions(methods) called from Controller Product & returned the data (i.e. Adding, updating, Deleting Product/s or Product Option/s or searching Product/s or Product Option/s based on given criteria. This is the best way to maintain each individual Action(method) & able to maintain code for future.
* Model Helper.cs been removed as it represents existing Data connection (Database & Server) because it is not wise to keep under Code file (1) Security Concern (2) future maintenance i.e. if someone must change database & server in future don’t need to update model Helper.cs (3) For various deployment Dev, staging or Production don’t need to update Helper.cs & don’t require separate release every time. Rather I kept this data connection details in Web.config file
* Created (rewritten) Model Product.cs which represent only data type for Product/s like Product ID, Name, Price & Delivery Price etc. Removed the data entity related to Product option/s Also removed all methods like Get, Delete, Save as it is not practical to write all methods in Model. Model is basically Data Class & represent Data Entity. All actions will take care by ProductRepository.cs
* Added new Model ProductOption.cs which represent data type for various Product Option/s for each Product like ProductOptionId, ProductId, Name, Description etc. All methods will take care by ProductRepository.cs
* Added new Model ProductsDB.cs which represent Proucts & ProductOptions and tied it to Entity framework to make DB operations atomic. EF is considered as best-practice for low level product/details kind of inquiry and suits better but of course it depends on the context and other restrictive factors depending on existing infrastructure, processes, and technologies in place.
* This will be used in future when we have complex tables with foreign key relationships. If data is coming from more than 3 tables, it's better to use storeprocedure and consume storeprocedure data at ProductTasks level. Utilize methods defined at ProdTasks level. To get final result here (service files) and send it to Controller.
* In addition, if provided with a genuine productivity tool like ReSharper then this can be further improved from code styling and other improvements.