Hi Theo,

The coding exercise should take you about an hour to complete, but you can go for longer if you wish – what I would say is that your solution should be clean, easily maintainable and not over-engineered. You should follow principles like SOLID for example and follow a TDD approach.

FYI - There is a bug in the code, please refactor the solution to fix this bug so that the payment will be accepted.  As far as I’m aware, they’d like it refactored. There is a bug in the code that was put there on purpose I think. Your solution should be built to now accept the payment from my understanding.

Take a read through previous feedback for tests that were not passed.

No based on tech test

* **Candidate 1 -***Was looking for the validation to be separated into different classes per scheme, then injected either with IOC or factory. This would allow each scheme to have its own control over validation rules which would be easily unit testable. It would also allow separation of the payment service from the validation rules, this would mean the validation result could be mocked allowing for easy unit testing of the method. Also only one unit test written, it does mention that not much previous experience with unit testing but that is unfortunately a core part of our platform.*
* **Candidate 2 -***not easy to follow, cannot understand the rationale*
* **Candidate 3 -***Validation is part of the main MakePayment method, expected this to be separated into separate classes per scheme, this would enable easy editing of validation rules for each payment scheme as well as separate unit testing. MakePayment unit tests could then mock the validation result and have a smaller more concise unit test around making payments.*
* **Candidate 4 -***Validation has been separated into separate class however the switch statement remains. Expected each payment scheme to be separated into separate validation methods/classes. This allows concise unit testing of rules for a specific scheme, also allows rules per scheme to be expanded on separately with clear inter-scheme boundaries.*
* **Candidate 5 -**Good that DI used. Good that spotted missing transaction and suggested using immutable ledger. Validation service still uses large switch statement for scheme validation rules, expected these to be separated into separate methods/classes. Unit tests could be more specific, one test for one outcome.

Lastly, can you please return your solution via a Dropbox link? [www.dropbox.com](http://www.dropbox.com/) If you select “share” can you click on “anyone with this link can open the file” – this way I can simply send them the Link and they can access it.