# Overview

Thanks again for applying with us! We want to get to know you a little better and find out more about your background and experience. Please provide well-formed answers that provide real insight into your professional experience and approach to delivering working software. After we get a chance to read through your answers, we will contact you in two to three business days and set up your virtual interview with our team. Thanks again, and we’re looking forward to talking with you more soon!

# General Questions

1. What are the first things you do when reviewing someone else’s code?

**I first review the requirements for the Change Request (CR) or the Incident (bug) report to determine if the code provides an effective solution. I also check if it is the correct solution or project that is being changed, or added to, or merged to. I validate that any PR’s are tied to a specific user story and that all commits can be traced back to a user story/task.**

1. How do you determine if the code is “good” or “bad”?

**I check if the code abides by SOLID, DRY and or KISS principles to determine if the code is good for addition to the repository. I find that decoupling of logic into separate classes by sticking to an architectural pattern helps to mitigate any changes that might break any integration testing. Bad code is unnecessarily complex and tightly coupled.**

1. What are your guiding principles and practices for software development and team management? How did you learn them? How do you share them with your team?

**My guiding principles for software development are to effectively breakdown any product backlogs into small user story tasks. Make sure that the team members acknowledge the effort estimates required for each task and that the members have all available resources to accomplish the assigned tasks. Once development is done it is important to have test coverage run for the build every time so there is continuous integration. I learned most of them from experience. I share them at the sprint planning stage to make sure we are all on agreement throughout the current iteration of the sprint.**

1. Tell us about a challenging bug you had to troubleshoot and resolve in a production environment. How did you find it, reproduce it, resolve it, and deploy it?

**First of all, you should not debug in production. I had one instance where I did not test for legacy browser compatibility. The production bug was found by a user who called a help desk and eventually bubbled up to me. I was able to reproduce the incident in the QA environment once I installed the legacy browser with the specific configuration. I resolved it by coding no cache headers to the http response. At that project there was an on-premise infrastructure team that took care of deployments at that time. All I had to do was direct them to a shared location of the DLL’s on the QA environment.**

1. Tell us about a time you had to effectively manage your time between team member needs, your own development commitments, and client relationships.

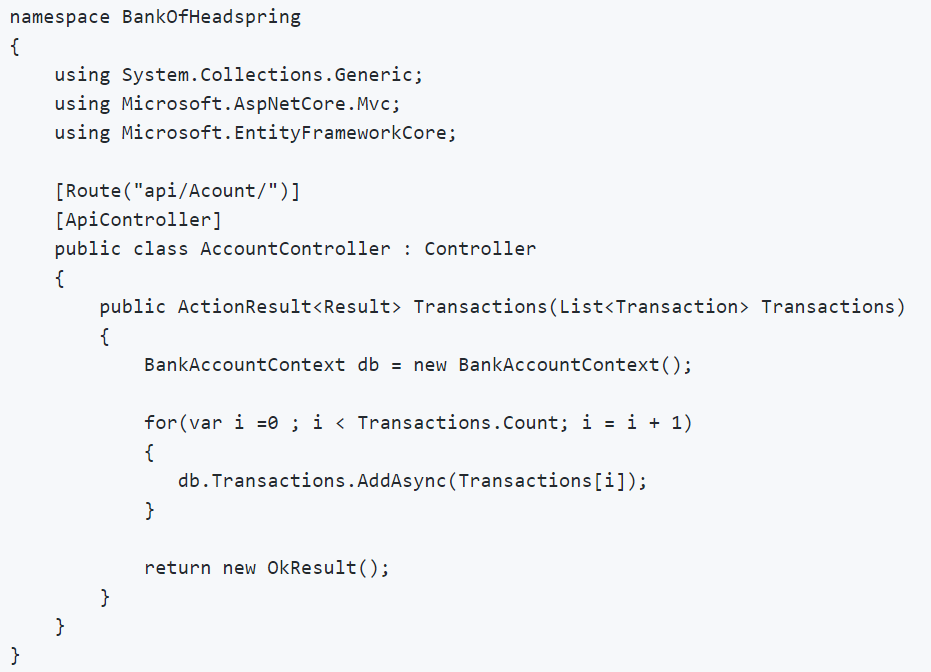
**Given my consultant background this happens quite often. In a recent project, I would dedicate the greater portion of my day getting to understand all the requirements from the client product owners and stakeholders. My development commitments would revolve on feedback gained from the client and I would email my team members any commits I would make to the solution. I would prioritize any client/product owner meetings and would either pull in team members or email them updates after. My development commitments would then ensue only after all client meetings were done or team members were also working on development commitments/user stories.**

1. Tell us about a time you had to organize a team to meet an aggressive deadline. What steps did you take to ensure everyone could effectively meet the deadline?

**In a recent project, I was the lead developer with a team in India and needed to make sure that the deliverable was to be released on time. We would meet during my night to make sure that the requirement estimates were feasible and try to see if we can cut or push any tasks that were not related to the deliverable to the next iteration. I would then report back with the client product owner to get an approval and focus solely on the tasks for the deadline. This would eliminate additional scope creep requests from the client.**

# Code Review

For the following questions, please review the code sample below. Imagine a developer on the team submitted this code in a pull request:



1. Please provide a comprehensive list of any issues you see with the code.

* **It is missing the HttpPost attribute on the method Transactions.**
* **The ModelState is not checked for a bad request.**
* **The Transaction object is not mapped to the Transactions object in the context. Yes, I have used Headspring Jimmy’s AutoMapper Nuget package.**
* **There is no exception checking.**
* **You can use a foreach on the Transactions list given it’s an Enumerable object.**
* **It’s not performed asynchronously. The method signature is missing the async designation.**
* **The dbcontext is not disposed properly or there should be one instance of the context.**
* **It does not check if the AddAsync operation was successful after saving changes.**

1. What feedback would you give to the developer and how would you message it to them?

**I would inform them of the ModelState dictionary and to make sure that data validation happens before an add/save, post/put operation takes place. After making these changes, the developer should debug in a lower environment to validate that transaction data is saving to the database.**

# Coding Exercise

Tell us about your solution to the Coding Exercise.

1. What design tradeoffs did you make?

**I limited the custom divisors and tokens by three. The library only accepts three pairs of custom divisors and tokens. The library can only accept either an enclosed range or a set of integers to iterate through but not both.**

1. How did you decide what testing framework to use?

**I used the default MSTest framework as it supports unit testing for .NET Core.**

1. What parts of the requirements were ambiguous? What assumptions did you make when the requirements were not clear?

**I did not understand where the input was coming from until I read the final pitfalls section.**

**I assumed that there will only be three custom divisors and tokens to iterate through.**

**The custom token combinations were vaguely mentioned in the requirements, but became clearer in the second console application demo.**

1. What errors did you run into and how did you resolve them?

**I initially used a Dictionary object to keep track of the divisors/tokens, but decided to create a DivisorToken class that contains the matching pairs. This was easier to instantiate and to iterate though a list of these DivisorToken objects. I added conditions that prevent wrong developer input such as supplying a range and a set of integers at the same time.**

1. Is there anything you’d especially like to highlight about your solution that you felt demonstrated your design and development skills particularly well?

**I do not repeat code and made separate custom classes for the Range and DivisorTokens.**

1. What would you differently if you had to do this over again?

**I did several designs before my final submission. I think this approach is the simplest one I could come up with. Also, the console demos do not make use of the requirement for the user supplied set of integers. I did test this functionality in the unit test project.**