# Vertex Take Home Technical Challenge

## Background

For this technical exercise, **we ask you to timebox yourself to no more than two (2) hours of your time,** it is valuable to you and we don't want to take away from that. We are not looking for completeness, we are looking for how you think and how you solve problems, We also want to see you explain your solution so make sure you understand what you do submit completely.

Work on these requirements in order. If you don't know how to solve a requirement, skip it instead of taking time to learn a new skill specifically for this exercise.

## General Instructions

- Create a public GitHub Repository for this exercise
- Do NOT mention Vertex anywhere in the repository, the code, the README.md, anywhere
- Ensure all code is on main, but if you develop with branches do not delete them, again we like to see how you think and work so those branches give us valuable insight. Do not squash merge your commits to main, do a simple merge. Working on a single branch is fine, just commit often.
- When you are ready to submit, send an email to your recruiter with the link to your GitHub repository
- Provide proper instructions in your README.md of how to setup prerequisites, build your code, and ultimate run and test your code

## General Thoughts

- Use documentation to your advantage not for us
- Show off your skills. Do not try and match a language or skill you perceive, show us what you know
- Don't let deficiencies stop you, we all have them. Our growth is based on identifying our own deficiencies and improving on them, not hiding them or running from them

# Requirements

The following requirements should be executed in order based on the above instructions and background information.

## Build an API

Create a RESTful API that provides basic CRUD and other relevant operations on the following domain entity object. You can use an embedded database or external database. Be sure you explain how to create and/or connect to the database and any other relevant information. Note that your thoughts on SQL (or database specific language) is relevant. Make the attribute names relevant and accessible as well, do not simply copy the names we have provided below. Don't forget unit testing.

Entity: Customer

| Attribute | Type | Constraints | Notes |
|---|---|---|---|
| Id | UUID | PK | |
| First Name | String | | |
| Middle Name | String | | Null is acceptable |
| Last Name | String | | |
| Email Address | String | Unique | |
| Phone Number | String | | Can be composite |

## Integration and/or Acceptance Testing

Create a pattern and implement it for integration and or acceptance testing of your REST API. Be sure to provide clear instructions on how to run it and possibly expand on how you might automate this for CI/CD.

## Provide Observability

To your API from the previous step, provide common and well constructed observability into the functioning of your application. Assumptions on collectors of this telemetry data is fine, just document them.

## Containerization

Containerize your API. Be sure to document how you need to run the container and what inputs you need to provide to run the application.

## Kubernetes

Prepare your application for deployment to Kubernetes. You can leverage technologies like Kind or MiniKube to test this as you see fit, but be sure to create the pathway for deployment that is extensible. Include instructions needed for deploying your application to a Kubernetes instance.

## CI/CD

Design and document a CI/CD pipeline for your application. Put this documentation in your repository in a location appropriate for you. Be sure to call out any manual or automated gates in your process.