# Testing Overview

Unit tests test logic. For dependencies, use the real objects when possible. Mock dependencies to enforce consistency in testing the logic.

Infrastructure tests are unit tests that use actual resources such as databases, files, web services, etc. These resources are typically dedicated to a specific environment or generated on demand. If the environment is shared, then you run the risk of the data not being in pristine condition for your tests to pass or worse pass for the wrong reasons. This can happen due to failed tests, manual tests, etc. leaving the expected entries in a bad state. Reset the environment before tests are run to avoid issues.

In-memory databases may be used in place of real databases (like a SQL server) in some scenarios. They do not work the same nor do they support the same features, so unless your implementation is very basic, this is probably not a good option.

As a rule of thumb, unit test critical logic and typically anything discovered as a bug to verify the issue is fixed. Create integration tests to verify everything was hooked up. If the infrastructure is critical then create an integration test for it.

You should have integration tests that test the actual deployed application as well. Remember to perform integration tests, manual and automated to ensure your app works as expected and is hooked up properly in the deployed environment.

There's a lot to be said on testing and there are plenty of perspectives on what should and should not be tested. I encourage you to do your own research and form your own conclusions.