Test Plan

Since I am using the sample project I can't use test driven development as I am not developing this project and am only changing the testing aspects of it.

Priority and Pre-requisites

While I have many more requirements outlined in the requirements document I have focused on three main requirements for testing.

R1: User data is stored securely
R2: Keep stock updated
R3: Have delays of maximum 3 seconds

R1 includes users data so is the highest priority as this relates to GDPR:

- This suggests that there should be reasonably large amount of resources put into this.
- Since this is a high priority we should consider at least two different T&A approaches
- The redundancy principal suggests making intentions explicit – this being that the data is stored very securely.
- The sensitivity principal suggests that it is better to fail every time than some, this works because the safety needs to be secure every time and can only have a very small margin of error. So it is best that this system is very consistent.
- This can be tested by SQL injection and checking that they don't return unexpected results
    - The points of injection are the input of user details
    - Synthesise data to test these points
    - Document the results
    - Feed the data back into testing

R2 Testing that the stock has been updated as orders are placed:

- While not as important as making sure user data is secure it is still a high priority as not having the stock update could lead to orders being made with items that don't exist.
- This could be tested throughout the creating of the project
- Testing that this works would include synthesising test data and checking that the database will change and if it will change as expected.

R3 having delays of maximum 3 seconds:

- This is the lowest priority of the 3 requirements as a slow system that works is better than a quick one that doesn't.
- This can only be completely tested on the full system so would be part of the later testing in the project.
- There are validation and verification issues with this as
- To verify we need some synthetic data to run tests on

- To validify we need some logging of performance of the system
- This suggests that the following tasks need to take place:
    - Synthesisng data to test the ordering system
    - Scaffolding to simulate orders being placed by customers
    - Designing a logging system to capture the real performance
    - Feeding data back into testing

Scaffolding and Instrumentation

For R1 scaffolding and instrumentation are not used as they don't directly apply to SQL injection testing. Despite this at a later stage in the development instrumentation may be used to add code snippets that log or alert when certain security-related issues occur.

For R2 scaffolding would be used to create a controlled environment where the stock updating can be tested. This would include creating a database with pre-defined stock levels and simulating orders being placed in order to see how the system works. Then instrumentation will be used to record stock levels to see if they are changing as expected.

For R3 scaffolding can be used similarly to before where it would create an environment where orders could be placed and then instrumentation can be used to time how quickly the orders are placed. Thus giving more information about how quickly the system is working.

Process and Risk

Testing should start as early as possible for all requirements as this will mean that they won't become more expensive fixes later on in the testing lifecycle as it is much cheaper if a buig is caught early in the development lifecycle.

R1: Security should be tested all throughout the process and can start early in the process at unit testing. Although the system

R2: Testing that stock changes as expected can be tested early in the project as this only concerns the ordering of items and the stock database.

R3: This will have to be tested later in the project as it concerns timings and how quickly parts of the project can return the desired outcome.