Summary

A) Question A

i. I did at least one test for each of the requirements.I fount this efficient since we are supposed to carry unit testing, since most of the modules wer specific in nature, a test for each proved to be satisfactory

ii. All the tests were done in the same package, since we were doing unit testing and the scope was not that big , A single package testing was feasible.

B) Question B

i. I ensured that the code was sound by limiting the number of assertions, a technically sound code should have the fewest number of assertions possible

ii. To ensure code efficiency I used a method with the notation **@after** this method will ensure that all the held resources are released. I found this efficient since most of the variables were static, this meant that resources were held as class variables hence made testing difficult

C) Question C

Reflection

A) Testing Techniques

i. **Unit testing** is the first level of testing and is often performed by the developers themselves. It is the process of ensuring individual components of a piece of software at the code level are functional and work as they were designed to. Developers in a test-driven environment will typically write and run the tests prior to the software or feature being passed over to the test team. Unit testing can be conducted manually, but automating the process will speed up delivery cycles and expand test coverage. Unit testing will also make debugging easier because finding issues earlier means they take less time to fix than if they were discovered later in the testing process. TestLeft is a tool that allows advanced testers and developers to shift left with the fastest test automation tool embedded in any IDE.

ii.

- **Integration testing :** After each unit is thoroughly tested, it is integrated with other units to create modules or components that are designed to perform specific tasks or activities. These are then tested as group through integration testing to ensure whole segments of an application behave as expected (i.e, the interactions between units are seamless). These tests are often framed by user scenarios, such as logging into an application or opening files. Integrated tests can be conducted by either developers or independent testers and are usually comprised of a combination of automated functional and manual tests.
- **System testing :** System Testing is the testing of a complete and fully integrated software product. Usually, software is only one element of a larger computer-based system.

Ultimately, software is interfaced with other software/hardware systems. System Testing is actually a series of different tests whose sole purpose is to exercise the full computer-based system.

System Testing involves testing the software code for following

- Testing the fully integrated applications including external peripherals in order to check how components interact with one another and with the system as a whole. This is also called End to End testing scenario.
- Verify thorough testing of every input in the application to check for desired outputs.
- Testing of the user's experience with the application.

- **Acceptance testing :** beta testing of the product done by the actual end users.The purpose of this test is to evaluate the system's compliance with the business requirements and assess whether it is acceptable for delivery.
- **Performance testing :** Performance Testing is defined as a type of software testing to ensure software applications will perform well under their expected workload.

The focus of Performance Testing is checking a software program's

- Speed - Determines whether the application responds quickly
- Scalability - Determines maximum user load the software application can handle.
- Stability - Determines if the application is stable under varying loads

- **Security testing :** is a type of software **testing** that intends to uncover vulnerabilities of the system and determine that its data and resources are protected from possible intruders.

iii._ **Unit testing** : This testing technique is used to test the smallest components of the system, It is a very key testing for the project as if it does not succeed the rest of the project cannot succeed.

Integration testing : Used to test the combined units and find faults

System testing :

Acceptance testing : verifies whether the end to end the flow of the system is as per the business requirements or not and if it is as per the needs of the end user.
The implication of this testing is that the user can reject it, this would lead to the team either redoing or revising the project

Security testing : of a sensitive software with high chances of security bridge, it is a crucial test to perform. The implications to a software project could be a go ahead or a review of the security measures put in place. If the security measures are weak a review would definitely be key

B) Test Management Practices

i. _1). It's essential since it makes sure of the Customer's reliability and their satisfaction in the application.
2). It is very important to ensure the Quality of the product. Quality product delivered to the customers helps in gaining their confidence.
3). Testing is necessary in order to provide the facilities to the customers like the delivery of high quality product or software application which requires lower maintenance cost and hence results into more accurate, consistent and reliable results.
4). Testing is required for an effective performance of software application or product.
5). It's important to ensure that the application should not result into any failures because it can be very expensive in the future or in the later stages of the development.
6). It's required to stay in the business.

ii.

iii. Squish
features
Powerful integrated development environment (IDE)
- A wide range of popular script languages for test scripting
- Full support for Behavior Driven Development (BDD)
- Full control via command line tools
- Integrations with Test Management and CI-Systems

C) Mindset

i. It's important to consider interrelation between the units as the specifications are brought together by the units. It is impossible to test one component and neglect the other and achieve the results, for example there is no way we can test for adding a doctor without having a doctor in place, for us to test for allergy and medication we first have to have a patient added

ii. I tried to eliminate bias by testing the code without considering how it is going to flow, that is not first checking for the errors or expected problems in the code, This ensured that all the units were tested on a level based standard.

iii. **Requirements are Crucial for Good Testing:** As a software tester we need to test what we know, this means digging deep into code to find what to test,Requirement precision and completeness is a must. It is therefore crucial not to cut corners because users using the system always have zero clue of what they want. If a software tester cuts corners, the system is most likely to fail during use by expected audience