VU LEARNING HUB

CS611 Assignment 1 Solution

Fall 2026

📢 Join our official WhatsApp Channel for instant VU updates, past papers & assignment solutions!

➡ **https://whatsapp.com/channel/0029VbDOzXMC6ZvZBUbW8k2n**

💬 Questions? Chat with us on WhatsApp for quick help & guidance!

➡ **https://wa.me/923135320064**

🌐 Visit our website for more past papers, notes & assignment solutions!

➡ **https://vulearninghub.xyz**

CS611 — Software Quality Engineering

Assignment No. 1 — Fall 2026

A retail company hires a software team to build a warehouse management app. The team builds the app exactly as described in the contract and tests every feature, yet warehouse staff find it cannot handle product returns, a task they perform every single day. The team calls the app a success because it meets the written contract, while the workers call it incomplete and unusable. Both sides are judging quality, but they are judging it by different measures.

a) Why the producer calls it good quality and the users call it poor

The software team (the producer) judges quality as conformance to specification: the app does everything the written contract requires, and every specified feature passes its tests. By that measure — building the product right, which is verification — the app is good quality. The warehouse workers (the users) judge quality as fitness for use: whether the app lets them do their real daily work. Because product returns are missing, they cannot finish their daily work efficiently, so in actual use the app fails them. This is the classic gap between verification and validation: the team verified the product against the contract, but nobody validated it against the users' real tasks. A product can therefore pass all its specified tests and still be poor quality in the hands of its users.

b) How user expectations differ from the customer's expectations

The users are the warehouse workers who operate the app all day. They care about usability and task fit: is it easy to learn, quick to use, forgiving of mistakes, and does it support every step of their daily work, including returns? The customer is the company manager who bought the app. The manager cares about the purchase decision: was it delivered on time and within budget, does it match the signed contract, and will it pay for itself? The manager may therefore approve the app on contract and cost grounds while the workers struggle with it daily. This is why the purchaser and the end user must be treated as different stakeholders: satisfying the buyer's contract does not automatically satisfy the worker's work.

c) How modifiability helps solve this problem

Modifiability means designing software so that it can be changed easily, cheaply and safely after delivery. It is achieved through modular design, low coupling between parts, high cohesion inside each part, clear interfaces, and configurable workflows instead of hard-coded rules. If the warehouse app had been built this way, the missing returns feature could be added as one new module, plugged into the existing screens and stock records, without rewriting the whole app. The change would stay local, so existing features would keep working and regression testing would be small and quick. Modifiability also supports learning from users: as workers report gaps, each release can add or adjust one task at a time, so the app grows towards real fitness for use instead of freezing at the first contract.

d) Why following the contract is not always enough

Written specifications are never complete. Some needs are tacit: warehouse staff never thought to mention returns because it is simply part of every day for them, so it never reached the contract. Specifications can also be ambiguous, they change as work changes, and they usually describe functions, not qualities in use such as usability, efficiency and satisfaction. Users often cannot state all their needs until they see a working system. That is why good quality needs more than contract conformance: observing real work, involving users early, building prototypes, and acceptance testing based on real tasks rather than only on listed features. The contract is a starting point for quality, not its full measure.

In short, the team should treat the missing returns feature as a validation failure, add it as a modular change, and involve warehouse staff in acceptance testing of the next release.