SET09102 Coursework

40281448

**Requirement Specification**

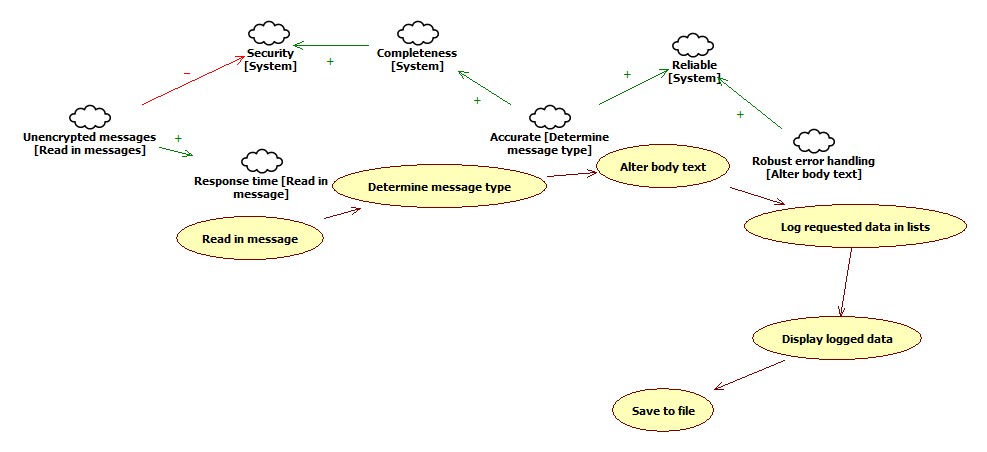


Figure NFR Use Case Softgoal Diagram

The NFR softgoals describe a system that should perform accurately, reliably, and reasonably secure. A quick response time when reading in the messages is required so as to return feedback of any errors as soon as possible to the user, with helpful error messages. The unencrypted nature of these messages aid the response time greatly, however, this leaves the messages, and therefore the information held within those messages, open to attack in multiple ways. This, without doubt, would negatively contribute to the overall security of the system, as is documented above.

**Version Control Plan**

An online repository for code, GitHub, has been used continuously throughout the development of the software to allow a seamless and reliable integration of design iterations. As this project has been completed by one individual, the assumption will be made, for the purpose of the version control plan, that multiple team members are involved.

Team members will be separated by task, based on the current iteration’s requirements. The VCS will be utilised in a way that branches are created, preferably in a 3-tier manner. The team will branch from the master to create a development branch before branching from that to create their own individual branches.

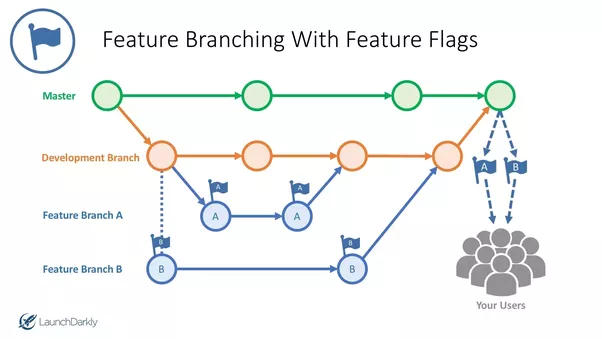


Figure 2 Branch Layout in VCS

By using this approach, the team will be able to intricately examine every increment to the app with complete control. The master branch (the live branch) is unaffected from any development work, allowing users to access the software without disruption. The development branch is taken from the master and allows the teams to communicate and build the latest increment. It acts as a hub for the teams, allowing everyone to see changes and make changes if necessary. This allows teams to be re-assigned to different tasks with ease, as the code in development is accessible by all.

From this development branch, teams create their own individual branches that are designated for the development of their task, referred to as ‘feature’ in the above figure. This keeps tasks separate from each other and allows a safe environment for teams to develop, test, and push the boundaries of code without having any effect on anything out with the current branch.

When teams complete their tasks, they are able to push their changes to the development branch, where code can be collated and reviewed before being deployed on the master branch. Before deployment, the teams can set up automated tests that run before a push is accepted in the VCS. This extra step can ensure the branches only consist of clean and usable code, with no undesirable surprises for those who may be pulling from the branch.

**Evolution Strategy**

It would be a fair assumption that before deployment of this application, legal issues with the transfer of personal information to the software carried in the messages, through whatever means, would invoke the implementation of an encryption feature. The messages would need to be securely delivered to the software and unencrypted within it, before saving the files securely – the necessity of re-encryption at this point is dependant on the manner in which they are stored, i.e. within a database that is already encrypted or other means of effective security blocking access to the sensitive information.

The evolution process would involve a mix of some corrective, adaptive and perfective maintenance, before eventually requiring at least some re-engineering to keep methods and functionality up-to-date in the processes that are used. The corrective maintenance would assess the clarity of the classes in the first regard, before expanding to ensure that the rest of the program is as easy to maintain as possible.

The system has been split into methods as far as seemed sensible to increase the readability of the software as a whole, in an attempt to follow best practice and decrease future maintenance costs. Commenting and clear indication of the purpose of methods both play imperative roles in increasing the ease of understanding of the project and allow debugging to accurately pinpoint any potential bugs or defects without complication.

Most of the maintenance cost would be during the adaptive maintenance, in which functionality would be altered to keep the software up-to-date with the current demands of the industry. Enhancements that are made after this maintenance has been performed would consume the largest part of the maintenance budget, as new requirements come into play.

# References

Justin Baker, S. (2016, August 1). *What is the best way to incorporate feature flags and toggles with version control branching?* Retrieved from Quora: https://www.quora.com/What-is-the-best-way-to-incorporate-feature-flags-and-toggles-with-version-control-branching