Module Code: CS2QT19

Assignment Report Title: Quality Plan

Student Number: 26011251

Date of Completion: 11/02/20

Actual Hours spent for the assignment: 12

Table of Contents

Introduction

This document details the quality plan for the development of an Event Management Assistant. The application will contain the features commonly found in other management applications: a calendar, a check list, budgeting tools, and a mailing list. These features will interact with each other and will be customisable as per the needs of the user.

The project will be conducted by one developer, following the development pipeline stipulated within the document.

Product Plans

The product contains four core functions which must be developed in sequence. Each stage must be completed and fully tested before proceeding to the next stage. Each function must fully work with preceding functions.

Each function of the system has a set of high-level requirements which must be implemented to ensure purposeful functionality and inclusion within the system. ‘Must’ features are necessary for the function to perform as required. ‘Could’ features are not necessary for the function to perform as required and as such should only be implemented if the stage has been completed and tested, with enough time to spare to properly implement the additional feature to the preexisting function. If not implemented during the initial development, these features could be implemented after release with user feedback dictating order of addition to the application.

1. Calendar Function
    - Users must be able to add/remove events onto/from the calendar for each date
    - Users must be able to set reminders for each date and receive notifications (via Email/ text/phone popup)
    - Users could be able to share their calendar with other users to compare schedules

2. Check List Function
    - Users must be able to create and edit check lists with items
    - Users could be able to share their check list with other users to compare and distribute tasks

3. Budgeting Function
    - Users must be able to make calculations based upon ingoing and outgoing money
    - Users must be able to construct a budgeting plan based upon ingoing and outgoing money
    - Users could be able to share their budgeting plan with other users

4. Mailing List Function
    - Users must be able to add email contacts to a mailing list
    - Users must be able to send emails automatically to users on their mailing list
    - Users must be able to sync automatic emails with events from their calendar and check list, as well as their budget plan

These requirements represent features common across scheduling applications, such as monday.com [1] and doodle.com [2]. A successful application should emulate these popular features without plagiarism or uninspired copying. The application is to be developed by a single developer and as such the vital requirements should be prioritised to avoid a loss of quality by trying to complete all tasks within the allocated timeframes.

Process Descriptions

As defined in ISO 9001, process need to be “aligned and understood by everyone in the business in order to increase productivity and efficiency” [3]. As such the processes through which the application will be developed and the tools used are to be described. The application is to be developed by one developer however it is still important to document the processes so the same steps can be repeated by other parties in an effort to improve and optimise the processes.

Component Development Pipeline

To ensure quality and consistency throughout the development, each core component will be developed following the development pipeline shown below in Figure.1.

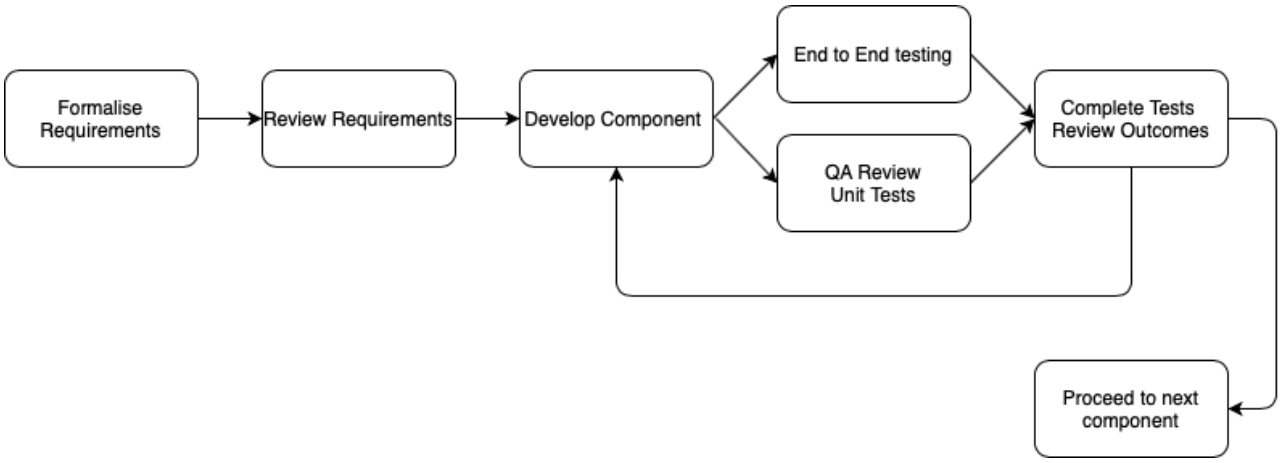

Figure.1 Process Development Pipeline

The initial stage is to assess the purpose and functions of the planned component. These requirements should then be reviewed and decided upon according to the product brief. Furthermore a hierarchy of requirements should be created to determine which requirements are to be given priority. If issues are faced during the development, the development of non-essential requirements can be postposed to post-release content patches in order to meet project deadlines. This relative leeway should not be abused however as all requirements should be completed within the deadlines given.

Following this review, the main development of the code shall take place. Pair programming will not be able to take place and as such care should be taken to properly implement comments and documentation to allow future developers to assess the code without assistance, important if the contract stipulates ownership of the source code for the client. Consistent naming conventions should be used for variables, the layout of functions should also be consistent.

Components cannot be developed concurrently nor can testing occur concurrent to development. As such a cycle of: “develop, test, assess” will take place. The coding of the component will take place, followed by assessment of quality and unit tests. Additionally End to End testing will take place to ensure cooperation between components within the system. These outcomes will be assessed and necessary changes to the code will then be made, followed by further testing until satisfaction that the component meets the specifications has been reached.

Development Platform

The project will developed using Java within the IntelliJ IDEA. IntelliJ is the preferred development environment for Java as it offers helpful features (eg.predictive syntax) [4] that are lacking in other IDEs, such as Eclipse. Java will be used instead of other languages as the principles of Object

Oriented Programming will allow for interaction between components within the system. Furthermore, Java is one of, if not the most popular programming language [5] and as such detailed videos and blog posts may be viewed online when in need of help, alleviating one of the problems of working individually on a project. Each core feature will have its own class, with as many subclasses as necessary. Dependencies and inheritance should be kept to a manageable level as to not overcomplicate the debugging process (the greater the dependencies, the longer the trail of code to follow).

Source Control

The code will be stored on Git. There will be a master branch and and four main branches for each function of the system.

- Master
    - Calendar
        - (as many branches as necessary for function)
    - Check List
        - (as many branches as necessary for function)
    - Budgeting
        - (as many branches as necessary for function)
    - Mailing List
        - (as many branches as necessary for function)

Daily commits should be made to ensure the code is up to date and secure in case of hard drive failures. Furthermore progress can be logged at constant intervals; the difficulty of certain tasks may be assessed relative to the amount/quality of code committed and thus improvements to methodology can be made accordingly.

Code containing errors should be assessed and debugged while coding, the IntelliJ IDEA provides assistance in tracking any errors, and as such it should be possible to commit code without any errors at the end of each day. If problems cannot be solved by the end of the day, errors and bugs should be commented on with the commit so work can be resumed the next work day without overlooking any previous errors or bugs.

Quality Goals

For the product to meet its specification, certain quality goals must be adhered to. Quality goals are defined as " a collection of individual metrics that helps to quantify quality of application as well as quality of testing" [6].

The ISO 25010 Quality Model details the eight quality characteristics which must be considered in order to meet the stakeholders' needs. It can be applied to the development of the system as shown in Figure.3.

Figure.2 ISO 25010 Software Product Quality Model [7]

Quality	Application to the project
Functional Sustainability	Each component of the system achieves its purpose and provides expected results.
Performance Efficiency	Each component operates with reasonable response times and recourse utilisation. Resources must be optimised to allow the storage of large amounts of user data.
Compatability	The project works cross-platform and can share data between components. A common environment and resources can be shared with other products.
Usability	The application is intuitive to operate and control for the intended demographic with an easily understood interface. The application is accessible to those with certain special-requirements (eg. Colour-blindness).
Reliability	The application allows for stable, everyday use under the required conditions (eg.Offline for singular usage or Online for shared usage). In the event of an error or crash, user data can be recovered and the application brought back to its previous state.
Security	User data is used in accordance to the Data Protection Act 2018 [8]. The integrity of the application cannot be compromised by external agents. Databases are secure and compliant with OAuth 2.0 standards [9].
Maintainability	The application can readily be modified and updated in future releases. Code is sufficiently commented on and documented to allow future use by other developers. Assets are available for reuse in other systems. Are tests implemented automatically or must they be maintained by developers?
Portability	Can the software be adapted for use in different environments? (eg. Different Hardware or Software configurations)

Figure.3 Application of ISO 25010 SQM to the project Table

Figure.3 provides a basis for which the quality goals within the system can be assessed. It is not possible to optimise a system for all of these attributes, and as such the most important quality goals must be identified.

Functional Sustainability

The application must perform consistently without errors in a way suited to the target user base. As such a key quality goal is Functional Sustainability. To ensure each component works with each other, End to End testing will occur. Error handling must also be tested. All tests must be fully completed.

Usability

The target demographic must be able to use the application with minimal assistance or reference to documented instructions. Documented instructions must be available for users if needed and must be written in simple English. Translated versions for other languages may be made available if necessary.

Reliability

The application must allow stable, everyday use. User data must be stored and readily available. Error reports can be sent by the user during the event of an error or crash so any widespread issues can be solved.

Security

User data must be correctly handled as per the Data Protection Act 2018. Account information must not be available to the public and must not be abused by the operators of the platform. Accounts must be password protected, with two-factor authentication optional to users. User data must be removed from the database when account deletion has been requested.

Objectivity in Quality Testing

It is common practice to have a team to conduct quality assessments separate from the development team, this ensures objectivity as the quality team are not influenced by development issues. As this project will be conducted by one developer, it will be important to test for quality against determined metrics as to be consistent.

Figure.4 shows five core metrics based upon external quality attributes: maintainability, reliability, reusability, and usability.

Internal Attribute	Description
Depth of inheritance tree	Inheritance of classes should be utilised to reduce overall amount of code required. The depth of the inheritance tree should be kept to the minimum required amount as to not overcomplicate debugging.
Cyclomatic complexity	The number of linearly independent paths within a section of source code. Complexity should be limited to a value of 10 as per common practice [10].
Program size in lines of code	The program should have the minimum amount of code necessary to function as to reduce complexity and the total size of the program.
Number of error messages	Errors recognised during testing should have noted error codes in the event of an error occurring. These errors should be listed for users and developers to access.
Length of user manual	The user manual should be clear and concise, informing the user of the functions of the program.

Figure.4 Internal Attribute Metrics Table

Risks and Risk Management

Risk should be identified and documented throughout the project. By doing so, current and future issues may be resolved, minimising potential error. Figure.5 shows the risks to the quality criteria as defined by ISO 9126 [11].

Criteria	Example Risk	Likelihood (3 Low - 1 High)	Severity (3 Low - 1 High)
Maintainability	- Code is too complex for future additions and improvements	2	2
	- Post release support is lacklustre or absent	2	2
Efficiency	- Application runs slowly with great demand on the system	3	3
Portability	- Application is not available on multiple hardware/software configurations	1	3
	- User data is not shared across devices	3	2
	- Application requires ‘always online’ connectivity	3	2
Reliability	- Components are buggy and do not function consistently	2	2
	- Error occurs without error code or proper error handling	2	2
Functionality	- Component does not function as required	2	1
	- Component is unused by user	2	2
Usability	- The interface is unintuitive and does not clearly convey meaning to the user	2	1
	- The user manual does not provide sufficient detailed instruction or is too obtuse and complex	3	2

Figure.5 Quality Criteria Risk Example Table

Bibliography:

1. https://monday.com/lp/aw/calendar/long/?marketing_source=adwordssearch&marketing_campaign=il-s-calendar-e-desk-monday&aw_keyword=calendar%20app&aw_match_type=e&gclid=EAIaIQobChMIt8e2tL695wIVyrTtCh2QmgXeEAAYASAAEgIATvD_BwE Date accessed: 06/02/20
2. https://doodle.com/en_GB/schedule-app Date accessed: 06/02/20
3. https://www.iso.org/standard/62085.html Date accessed: 06/02/20
4. https://dzone.com/articles/why-idea-better-eclipse Date accessed: 10/02/20
5. https://stackify.com/popular-programming-languages-2018/ Date accessed: 10/02/20
6. https://www.infostretch.com/blog/quality-goalsdefine-and-measure-quality-of-your-application/ Date Accessed: 10/02/20
7. https://iso25000.com/images/figures/en/iso25010.png Date accessed: 10/02/20
8. https://www.gov.uk/data-protection Date Accessed 10/02/20
9. https://oauth.net/2/ Date Accessed 11/02/20
10. McCabe (December 1976). "A Complexity Measure". IEEE Transactions on Software Engineering (4): 308–320. doi:10.1109/tse.1976.233837
11. https://www.iso.org/standard/22749.html Date Accessed 11/02/20