Quality Assurance Plan- UniPal

This is the Quality Assurance Plan for UniPal – a high functioning communication tool between students, lecturers, administrators, similar to QMPlus, with an emphasis on reliability, availability, and mobile use. This solution would be a non-university-specific system that multiple universities could adopt to solve their communication problems in the post-COVID, hybrid learning environment.

In this plan I speak talking about the software engineering techniques including design, estimation, configuration management and the SDLC. I also discuss the reviews that are taking place and the testing strategies that we will implement. Within each section I will be including how all the points mentioned will contribute to the quality assurance of UniPal, in making sure that the verification and the validation (product and process correctness) is being done appropriately.

Software Engineering Techniques:

We will be using agile software development methodology develop UniPal. This is to allow maximum productivity and maximum user involvement and feedback throughout the process. This way we can constantly improve our product to satisfy the user's needs by responding to changes appropriately, therefore including the quality of the project and to ensure that it does what it needs to do. We will be developing an initial working prototype version of UniPal, which will be presented to a university as a test user. Upon gathering their feedback, we will be making the necessary changes and improvements to the software iteratively until all the requirements have been met and all the functionalities are there. After this, the final product would have been made, and we will begin the mainainance, testing and installation phase of the life cycle. We will be using audits which will contain a checklist, and within this checklist are criteria used to determine that the product has been developed using the correct procedures and processes. This is to assure quality and software verification and process correctness.

We will be using scrum, where each use case is broken down into tasks and these will be in sprints. In the beginning of each sprint there will be sprint plan where we identify a goal as to what we will be developed from the use cases, and by the end of the sprint those deliverables would have been developed. This allows us to release parts of the projects quicker to then demo to the stakeholder thus increasing user satisfaction. Because of this we can evaluate product correctness as when we are involving the user in the scrum process with the demos they are able to give feedback as to what the system is missing, which we can then use to go back and make improvements to the system, therefore meeting the requirements and increasing the quality of the product, increasing software validation.

We will be designing UniPal as a web application that is usable in the browsers mentioned in the non-functional requirements. We will also be a designing a mobile compatible version of UniPal with IOS and android with the suitable front end and backend that is user friendly. We will be using UML to demonstrate the classes and objects present in the application and the overall workflow of UniPal. We will also be using a use case diagram to demonstrator all the use cases of the system, and which user type corresponds to those use cases. This allows us to evaluate product correctness as the use cases (from the UC diagram) and the objects in the UML diagram are all linked to the requirements. So if all the use cases and objects are implemented fully then we know that the system does what it is supposed to do and are able to validate the product. This also allows process correctness (verification) as using these diagrams, we can make sure that the code is developed in the correct way for e.g. the use cases are developed for the correct user in our case for example only the administrators can remove students from a course.

Estimation: There are 2 aspects of estimates to consider for this project. Firstly, the estimation of the tasks. We have made the estimates for the tasks based on the Developers bandwidth and competency within the task that they are working on. We will be using working days which will include any meetings involved for the project and excluding any holidays. The other is regarding the metrics for the project. The metrics that we plan to measure and estimate will be the Availability, Ease of Use, Speed/Response Time, Robustness/Recovery, Reliability, and Maintainability. These metrics must be measured to ensure that we have met the Non Functional requirements. This will allow us to evaluate product correctness as we can determine if the requirements have been met with these measurements and estimates. I will now discuss how we will make the estimates of these metrics to evaluate product correctness.

Availability

- Availability we will count the days that the system is down for maintenance and estimate the percentage of the year that it has available for by subtracting it with the time it is down for maintenance. If it is at least 99% available, we know we have satisfied the non-functional requirements.

Response time

- We will measure the time it takes to for the user to log in. We will be logging in 50 times and getting the average time it takes for the user to log in and we will check if this average Is <=1second.
- We will then try to stress test the system by giving it 5,000 login requests and if the system is unable to less than 5000 of these requests then we know the requirement hasn’t been met.
- We will be timing how long the system processes a submit action from the user when they have a max sized submission and making sure that this is <= 5 seconds.
- We Will fill in all inputs with maximum size for the module feeds and timing how long it takes to load and making sure that that it is <= 1 second.
- We will be timing how long it takes for the system to give visual feedback and checking that it is <=0.5 seconds.

Ease of Use

- We will be sampling 100 users. And for each user, we will be checking how long it takes for them to complete the key functionalities listed and making sure it is done within an hour with no further instructions. More than half of the sample should be able to do so in order to meet the requirement.
- We will be making sure that more than 95% of the sample can figure out how to use the functionalities within an hour when they are given a user manual.

Reliability

- When the system fails, we will be checking how long the system had been operating before failing. If it is at least 1 year, we know we have met the reliability requirement.

Robustness/Recovery

- We will be timing the system once it crashes to determine how long it takes to restart and return to its normal state. If it is <= 30minutes we know we have met the requirement.

It is important that we measure these metrics and estimate them accurately as It allows us to determine if we have met the non-functional requirements clearly as it is simply comparing the numeric values. This ensures that we are completing software validation and product correctness, and that UniPal is displaying the correct features and does what it is supposed to do.

Configuration management: using git. This is to make sure the product is consistent with the requirements, and design information throughout the lifecycle. There will always be a baseline, which in this project will be the first working prototype of UniPal. The baseline is a formally reviewed and agreed upon version of the product, that is used as a base for further development. Changes can only be made through the correct change control procedures, which is listed below.

Our process of configuration management for UniPal

- Developer creates a fork from the git development environment and clones to their local machine so they can access the code and start making changes
- Developer will make the relevant changes to the code based on what ticket they are working on and will push to code review once the changes are made. The ticket will then be marked as "in code review"
- The tech lead will review the code against the standards and once it passes it will be marked as "ready for build". The build will then begin, where the code is merged with the existing code base from the dev branch to the master branch.
- Quality control is performed in the integration environment and is then pushed to user accept testing. The ticket should then be marked as "ready for testing"
- User acceptance testing is performed in the environment by SQA staff.
- Once testing has passed and approved the code from the master will be deployed into the production environment will be tagged in git. The ticket will now be marked as done

We have opted to go with this process as it allows for faster problem resolution. This reduces the cost and increases the quality of the product as there is less likely to be errors. This is key in product evaluation as there is a process in place to check if there are any errors in the application. If there are it cannot go onto the next stage until those errors are fixed. This allows us to make sure that the UniPal has met the requirements before making any further developments/releases and therefore gives more certainty that the product does what it is supposed to do. We can also evaluate process correctness as this will be the process that everyone is required to follow before making any changes to the software. This means that any changes that are made will only be made if they have gone through this process, therefore ensuring that we are developing the software in the correct way (software validation) [1].

Executing Technical Reviews:

At the beginning of the sprints, we will be reviewing the requirements with the user to gain extra clarification that that is what they want from the product, and we will be creating user stories out of these requirements, which will be broken down into tasks for the developers to complete. This needs to be done regularly so that we can adapt to any changes in requirements without compromising quality, as a common quality problem within many projects is having poor and unclear requirements. If the requirements are reviewed and confirmed accordingly, this allows to evaluate the product correctness as we are then able to tell if the requirements have been met or not.

Reviews must take place in order to ensure that we are meeting the correct standards that have been set. During the implementation phase of UniPal we will be conducting Code reviews. These are standards regarding how to develop the code so the code is at an optimal quality. Every time user pushes changes their changes will be checked against the standards to see if passed or not. This will be done during the implementation phase of the software development lifecycle. Code reviews have been shown to reduce errors by 80%, therefore making it more important to complete these to make sure there are less errors, therefore increasing the quality of the code and the product UniPal. [2].

These are some of the standards we will be including for the code review [3]

- All functions and objects must be commented above with a line between the comment and the function/object to increases readability.
- Variable names must be relevant and be in line with the purpose of the variable. Variable names should all be in lower case, short and not include any metadata.
- Class and function names should be in Camel Casing.
- Constants should be in upper case lettering
- Indentation: use from 2-4 tabs for indentation and make sure to be consistent with indentations so all parts of code are indented in the same manner.
- Where lines exceed more than 120 characters, use line breaks.

The reason we are doing a code review is because there are many benefits that allow us to ensure process correctness including [4]:

- Allows us to fix bugs early on when they are cheaper to fix. Reduces costs without compromising the quality of the application.
- Allows other developers to gain more knowledge of the codebase, allowing them to produce better quality code that fits with the codebase, therefore improving the quality of the project.
- helps to maintain a level of consistency in software design and implementation.

This ensures process correctness and software verification because we are putting standards in place to make sure that the software is developed in the correct way. Even though the software may meet the requirement, if there are no standards (like what is listed above) in place to make sure it is developed correctly, then the overall quality of the application will decrease and may cause errors later on in the process.

We will be inspecting each functionality of the software and checking the extent to which it meets the requirements and the specification. We will be using the ISO 9000 definition of quality which is “the degree to which a set of inherent characteristics fulfils the requirements”, to determine the quality of the project.

At the end of each sprint, we will have a sprint review. This involves a demo of the working functionality that was developed in the implementation stage to the user. This is Important to check if it meets the functional and non-functional requirements, therefore we can determine the quality of the product better and allow us to complete software validation. The stakeholders and test users will be in this meeting to review the latest changes and improvements to UniPal. The developer or the SQA staff that was responsible for testing the functionality out will present the working solution to the user. The user is then free to give any necessary feedback. This allows for better communication with the stakeholder and allows us to take into consideration their feedback and improve the quality of UniPal in order to ensure the product correctness with its requirements set in the beginning.

At the end of each sprint, we will also be having a sprint retrospective, where we will review the previous sprint in terms of the processes and the activities completed. We will be reflecting on what went well, and what could be improved in order to ensure more productivity. Each developer and SQA staff can share their thoughts on what went well and what we can do to improve processes next sprint. If the processes are improved this will directly impact results based on the first principle of kaizen for quality: "before results can be improved the processes must be improved." [5] This, therefore makes it important for the sprint retrospectively to be conducted at the end of each sprint for optimal results and quality. This is how we will verify that the correct steps were taking during the development process to evaluate process correctness/software verification. If there were any deviations from the correct procedures, this will be addressed during this meeting. We will then suggest methods to make sure this doesn't happen again and how we can stick to the correct procedures so that in next sprints we stick to the correct processes. This will increase software verification and overall quality of UniPal.

Testing Strategy:

Unit testing

Each module developed that corresponds to a functionality will be required to have a unit test which tests for a particular functionality. This is to isolate the code and determine if that functionality is working appropriately. These functionalities could include for e.g. making sure the student signs in with the correct information, or the student only accessing modules they have been given permission to access, or lecturers only being able to edit modules that they teach for. The unit testing will be completed by the developers as well as the SQA staff members during the development phase.

Benefits

- Increases the quality of the individual components which therefore contributes to the overall system resilience.
- Makes and safer and easier to refactor the code, as It takes the risk out of changing older source code.
- Ensures that code meets quality standards before deployment which saves time from having to make changes later in the project.

Unit testing ensures product correctness as we are making sure that the test cases are satisfied with the unit that we are testing. If the unit test fails, we know that the functionality does not do what it is supposed to do and therefore the software validation for that component has failed. Unit testing also ensures process correctness as if the unit has not been developed in the correct way and there are code errors or logical errors, it will fail as you haven't developed the code properly. Unit tests are made to verify that the unit can perform independently from other parts. So if the unit satisfies the product requirements but fails the unit test it means it has not been developed properly as it is not fully independent. This shows that unit testing can be used to ensure product correctness(validation) as well as process correctness (verification)

Integration testing

This will be a more macro testing procedure as supposed to the unit testing. We will be testing how the units of the code interact with each other. Examples of interacting functionalities for UniPal could include submitting of assignments from the students and the grading of assignments from the lecturers, or the lecturers grading assignments and the students viewing their grades for an assignment. Another example could include the lecturer editing the times for lesson and the student viewing their timetable for that lesson. Our focus will be on checking the data communication between the units, such as module information, timetable information etc. These units will be exchanging data between one another, so it is imperative to test that they are interacting accordingly.

The type of integration testing we will use is the incremental approach, where we integrate two or more modules that are logically related to each other and then tested for proper functioning of the application. This is then repeated with another 2-3 modules that are linked until all the units/modules that are linked are integrated and tested appropriately. This is better than the big bang approach, where all the components or modules are integrated together at once and then tested as a unit, because it will be easier to identify where the faults are as we are doing things in smaller steps. Also, this is a larger scale application, so integrating all the components at one can be overwhelming and take up more time and effort.

Integration testing is necessary because different parts of the code would have been developed by different developers who may have a different understanding of the requirements. Therefore, we must do unit testing to make sure that these units are able to work together correctly. UniPal is one application that is composed of many units linked to each other, so to only test one unit independently without testing its integration with other sections may lead to unfound errors later, which will increase cost and compromise the quality of the product. [7]. Integration testing ensures process correctness (software validation) as even if the code meets the requirements independently if the modules are unable to interact with each other it means you have not implemented them correctly. Therefore, with a correct integration test plan we can ensure that the product is developed correctly by making sure that different parts of UniPal can interact with each other properly so that we can ensure that the overall system has been developed in the correct way.

User acceptance testing:

This will take place after the unit and integration testing. The end User/client will test the product and will give feedback and we can make the relevant changes accordingly. Our test users will be the administrators, students, and teachers from a university. They will test the usage of their product for their university. This is how we will conduct user acceptance testing for UniPal:

- Analyze the requirements from the documents such as the use cases, and System Requirements Specification (SRS) to develop test scenarios
- Create User Acceptance Testing plan, in which we will write the strategy used to confirm that the requirements have been met
- Identify the test cases
- Prepare the test data
- Record Results

- Confirm the business objectives being met: here the tester will sign off that all testing has passed, and the product is now ready to go into production/release provided there are no defects. Any defects that arise from the testing will be logged into a defect log.

We decided to include user acceptance testing in our testing strategy because normally developers create functionalities from what is their understanding of the requirements. This might not represent what the user needs from UniPal. That's why we must confirm this through user acceptance testing to evaluate whether we have made the correct product or not. This can also give us more opportunities to fix any issues or make any improvements based on the user feedback, therefore increasing the quality of the product. This shows that User Acceptance testing is key in assuring quality and evaluating product correctness. [8]

Overall, we are keen on adopting a multiple testing strategy so that more angles are covered. We can cover the angles from the developers, as well as the users and stakeholders. The more testing done on UniPal, the sooner we can identify defects in the software and make the necessary improvements. This saves time, costs, improves and assures quality.

Reference

[1] Ideas2IT, Git configuration. Available at https://www.ideas2it.com/blogs/git-configuration-management-2/

[2] Jeff Atwoof, Code Reviews: Just Do It. Available at https://blog.codinghorror.com/code-reviews-just-do-it/

[3] Javascript Coding Standards, Available at https://developer.wordpress.org/coding-standards/wordpress-coding-standards/javascript/#:~:text=Indentation%20with%20tabs.,indicate%20unreadable%20or%20disorganized%20code.

[4] Devart, Code Review benefits. Available at https://www.devart.com/review-assistant/learnmore/benefits.html

[5] Dr. Mahesha Samaratunga, Kaizen for Quality https://qmplus.qmul.ac.uk/pluginfile.php/3478769/mod_resource/content/1/Lecture%20Kaizen%20for%20quality.pdf

[6] Jason Boog, Unit Testing: Advantages and Disadvantages https://theqalead.com/test-management/unit-testing/#:~:text=Advantages%20of%20Unit%20Testing&text=Unit%20tests%20make%20it%20safer,quality%20assurance%20of%20the%20code.

[7] Thomas Hamilton, Integration Testing: What is, Types with Example https://www.guru99.com/integration-testing.html

[8] Thomas Hamilton, What is User Acceptance Testing [UAT] Examples. Available at https://www.guru99.com/user-acceptance-testing.html

**This is part of a large project plan for UniPal where I collaborated with 3 other developers. This document was my contribution to the Overall Project Plan