**Tracking and Control Mechanisms**

* 1. **Quality assurance and control**

**Introduction**

This Software Quality Assurance plan (SQA) will outline the processes, methodologies, standards, and procedures of how work will be performed to assure a timely and quality delivery of the pharmacy system.

**Overview**

The SQA will be organized as follows:

* Management: Description of major organization components and the SQA tasks and relationships.
* Documentation: Identify documents related to development, versioning and verification.
* SQA plan Requirements: Define the SQA reporting, review and auditing procedures to ensure deliverables are on time, with quality and in accordance with the SQA plan and project requirements.
* Training: The necessary skills needed to communicate with team members, document artifacts and develop code.

**Management**

Organization: There are four team members involved in the process. Work will be divided into tasks for each team member. An equal amount of tasks (effort or amount) will be distributed among team members. Team must report to their project managers about the status of the project. Project managers will be required to check on each task assigned to team members and report back as soon as possible.

Roles:

* Two project managers
* One team lead
* Three other members (equal amount of work as team lead)

Responsibilities:

* Project managers: Provide feedback and guide team members. Will act as middle man between client(Professor) and team members.
* Team lead: Must send weekly progress reports to the Professor, submitted to canvas every Monday.
* All team members: Develop a SPMP that will guide the development of the project. Update the SPMP as the project progresses. Involve project managers at all stages of the process. Evaluate Project managers, themselves and other team members on their performance throughout the project.

SQA implementation throughout all the software life cycles: The implementation of the quality assurance plan will be carried out until the release of the software.

* Software Requirements phase: During the development of the SRS, team members must ensure that the required functionalities are included. Required functionality is of more importance than wanted functionality. The required functionality should be refined until they are completely understood and stated.
* Planning phase: SPMP artifacts should be reviewed daily until the end of the project.
* Design phase: Code and algorithms should be reviewed daily and weekly to ensure quality.

**Documentation**

Software Project Management Plan: Will state the problem statement, the scope of the project, major software functions, performance/ behavior constraints, project estimates, risk management, project schedule, staff organization, and tracking and control Mechanisms.

Software requirements specification: Will list the essential requirements of the software. Essential requirements being: introduction, usage scenarios(s), data/functional/behavioral model description, functionalities, and constraints and validation criteria.

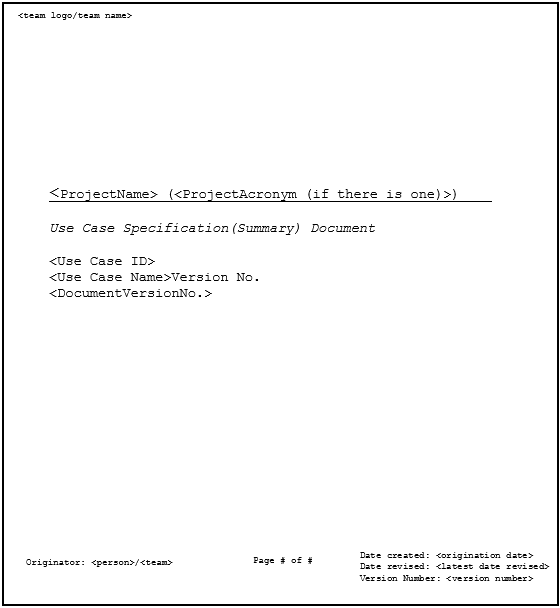
Uses cases:

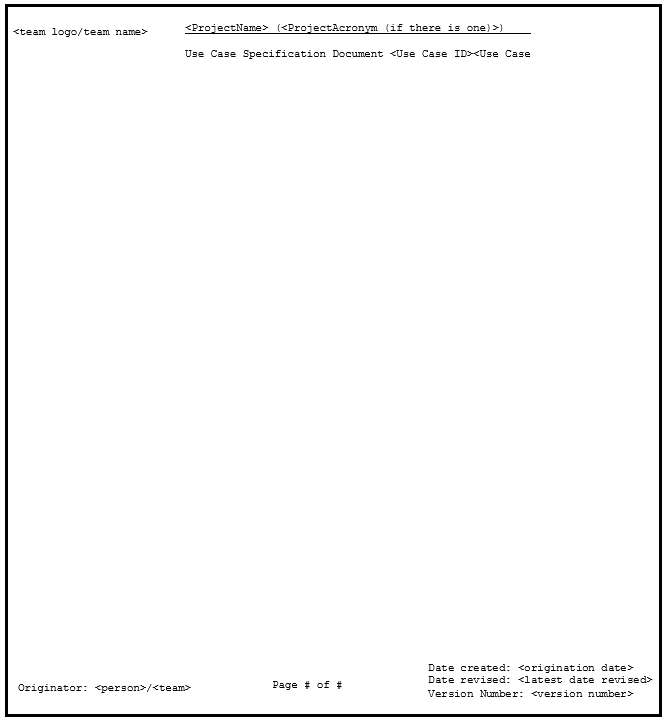
* Specification: Captures the detailed functional and non-functional business requirements. Technical or application requirements should not be detailed here.
* Summary: This document captures business requirements at a high level by identifying all the use cases in a project/application.

**SQA plan Requirements:**

Standards:

* Coding standards: C++
* Write-ability
  + Documents**:**
    - Software Project Management Plan(SPMP): High level language will be used in during part one of the SPMP process. As the project progress, the language will evolve to a more business based language.
    - Uses cases: Business language will strictly be used in use case specification and summary to capture business requirements.
* Read-ability
  + Artifact templates:
    - Software Project Management Plan (SPMP) template: The SPMP will follow the template provided in class. It will also include a table of contents for readability purposes.
    - Use cases: Use case summary and use case speciation will follow the template provided in class. The layout of each page of the use case specification and summary is as follows:





* Evaluations: Team members will be required to evaluate themselves, other team members and their project managers using the template provided in class. Names will be in alphabetical order by last name, you can assign any amount of points per person but the total sum must equal 100, describe what task where done and tasks that should have been done but were not, and describe how work was completed.
* Font and size**:** All documents will use Courier New font. Section headers will be in 14pt and boldface. The text that follows will be in 12pt, any titles of a major topic will be boldfaced.

Reviews and audits:

* Communication plan
* Software documentation
* Estimate review
  + Metrics: Lines of code, function point and task estimate will be used to determine the size of the project and software.
* Traceability review
  + Every artifact has to have traceability.
* Design review
* Code review
* Test plan review
* Test cases review

Testing**:** Both testing and error/defect collection and analysis will require every line of code to be testes as a part of some test case.

* Unit testing
* Integration testing
* System testing
* Acceptance testing

Error/defect collection and analysis:

* Types of data
* Types of defects

Change management:Make sure a disciplined process is defined. Only make changes when they are needed, not wanted.

* Process method
* Change handling

Safety:

* Environment

Security management:

* Access to artifacts
* Access to tools

Risk management:Risks must be mitigated, monitored and managed.

Noncompliance documentation:The hierarchy of reportingnoncompliance must be followed.

**Training**

Team members will be required to obtain or already have the skillset to use the following tools that will be used through the software development process:

* Visual studio 15
* Google docs
* Google calendar
* Microsoft word
* Wrike task manager
* Slack online group messenger
* GitHub

**6.2 Change Management and control**

Communication of changes and how they will be handled will be accomplished through both Slack and GitHub.

**Source code:**

* Slack will be used as the informal method of communicating a change. On Slack a team member will notify the others of a change to a file and inform them of any pull request review.
* GitHub includes a functionality that allows for a brief description of the changes made to the file when attempting to commit to the master. Team members will be required to write an adequate and general overview of what changes have been made to the file when requesting a pull request review of a file. Doing so will allow for fluid traceability of any bugs that may arise through the merge.
* Merge-ability of pulled requests can be accomplished through protected branches. Protected branches will ensure that no irrevocable changes will be made to the master branch without the having at least one administrator review the request along with other team members not involved in the file changes. One administrator will sign off and submit detailed reviews (if any) with a unanimous team decision to either approve, defer or reject pull requests. Administrators must ensure that all branches that are created are protected branches.

**Non-source code(documents):**

* For documents such as, but not limited to, the Software Project Management Plan (SPMP), communication of a change will also be accomplished through Slack and GitHub.
* On Slack, a team member will inform the others of a changed or added document to the GitHub repository. These documents will not require review by the other team members and can be uploaded directly to the master during the first part of the SPMP process. After part one of the SPMP, all requests to commit artifacts to the master will require at least one administrator approval.

**6.3 Tools**

The tool for control access and versioning of artifacts will be GitHub and Wrike.

**Source code:**

* Versions of the source code will be found on the GitHub repository through “branches”. Changes are proposed in a branch to avoid overwriting the master branch. The master branch will act as the baseline and will only contain published and approved work.
* The team leader will have admin access to the repository. With the exception of one team member who will also have admin access, all other team members will have read and write access to the repository.

**Non-Source Code(documents):**

* To differentiate versions of the documents on GitHub, an underscore “\_” in the document name will indicate the document is still in progress, a document name without underscores “\_” will indicate that that is the final version of the document.