# Travers SAMS Project Implementation Plan

Thomas Travers

8/1/20

CS406 Advanced Software Engineering

Grantham University

NOTE: Please see associated documents referenced for much better image and table quality.  Git address will be included in the bottom of the document...

## a. Project Description

*Describe the SAMS project.*

## b. Agile Effort Estimation

*Describe agile effort estimation performed.  Discuss why effort estimation is critical for a successful project.  Discuss differences between agile effort estimation and conventional effort estimation.  Add the agile effort estimation for the SAMS project performed in from the Week 1 lab.*

*I used a story card for each use case and made an estimation using a number from the fibonacci sequence.  I also listed my rationale for the effort estimation and the number of hours estmated.  Effort estimation in a traditional style is usually task based and performed by the project manager.  Agile estimation is usually team based and performed by the group as a whole.  The method used in the SAMS project is an agile method and is meant to be done by everyone involved on the team.  This should also be more accurate as it gauges estimation based on experience of a group rather than guesses of an individual.  It also has the benefit of enhanced perspective.  For example, one person in the group might have a higher estimation because they foresee having to code around a common security vulnerability in the hardware platform selected that the others don’t and they might include this reasoning on their story card.  This added perspective can help shape the estimate to include this crucial piece of information that others might have missed resulting in a more accurate estimation in the end.*

**Agile Estimation**

Apply Agile Estimation to approximate the effort required to satisfy the following use cases for the Study Abroad web application, described in appendix D, on page 667. Keep in mind that the estimation should include the time that it takes to design a solution that satisfies the use case, code, unit test the solution, perform integration, and user acceptance testing.

1. Place each use case on a story card or post-it note.

2. For each story card, or post-it note estimate the number of hours required for development. The number must be in the Fibonacci sequence.

3. List for each UC, your rationale for the effort estimation and the number of hours estimated.

4. Review your estimates, reconsider your rationale and revise estimates, if needed.

5. Sum the number of hours for a total level of effort.

6. Submit the list of UCs, your rationale, the number of hours estimated, and the total hours estimated.

For example:

UC01. Search for Programs | 2 hours | Rationale - The developer will reuse an already written a search algorithms for exchange programs.

UC02. Display Program Detail …

SAMS Project use cases

UC01. Search for Programs (Actor: Web User, System: SAMS) This use case allows a user to search overseas exchange programs using a variety of search criteria such as subject, semester, year, country and region. The system searches the database and displays a list of programs, each of which includes a link to display the detail of the program.

***UC01. Search for Programs | 8 hours | I chose 8 assuming writing algorithms from scratch. We should be able to accomplish this task using XML or JSON to communicate data to/from the database.***

UC02. Display Program Detail (Actor: Web User, System: SAMS) This use case retrieves and displays the detailed description of a program, selected by clicking the link of the program displayed by the Search for Programs use case, or by giving the program ID or program name.

***UC02. Display Program Detail | 3 hours | I chose 3 because all the data communication stuff should be completed in UC01 (functional dependency) leaving just the UI work here.***

UC03. Submit Online Application (Actor: Student, System: SAMS) "This use case allows a student to apply to an over- seas exchange program. The student fills in an application form and submits it. The system verifies the application, saves it in the database, and sets the status of the application to “submitted.” The system also sends e-mail to the two faculty members requested by the student to write recommendation letters, and the academic adviser to approve the course equivalency form.

***UC03. Submit Online Application | 21 hours | I chose 21 here because this use case has many tasks and functions that can be broken down.***

UC04. Login (Actor: Student, System: SAMS) This use case allows a registered student to login to the system.

***UC04. Login | 34 hours | I chose 34 because this task requires us to design a database for our project. The database must have carefully designed tables to prevent data corruption. It might be beneficial to also add LDAP as the authentication mechanism as this standard can be used for other applications on campus as well including campus email, student portal access, thus limiting account creations. If this system is already implemented, we can probably save a few hours by utilizing the existing LDAP system.***

UC05. Logout (Actor: Student, System: SAMS) This use case allows a student to logout from the system.

***UC05. Logout | 1 hour | I chose 1 because this is as simple as the front end terminating the connection and "forgetting" user credentials.***

UC06. Edit Online Application (Actor: Student, System: SAMS) This use case allows the student to edit an application that is not yet submitted.

***UC06. Edit Online Application | 3 hours | I chose 3. Assuming some previous tasks are dependencies, this is simply a form that auto-loads fields with existing values, allows edits, and submit or cancel changes.***

UC07. Check Application Status (Actor: Student, System: SAMS) This use case allows a student to check the status of an application, such as in-preparation, submitted, under-review, accepted, and rejected.

***UC07. Check Application Status | 2 hours | This is simply setting and retrieving a value in the database so I chose 2.***

UC08. Submit Recommendation (Actor: Faculty, System: SAMS) This use case lets a faculty member submit a recommendation on behalf of a student. The system saves the recommendation in the database.

***UC08. Submit Recommendation | 3 hours | This is simply a form for upload that records and registers in the database as well as links the record for the file to a pending application.***

UC09. Approve Course Equivalency Form (Actor: Advisor, System: SAMS) This use case lets an academic adviser review and approve course equivalency forms submitted by students when they submit their online applications. Each form specifies the overseas courses that the student plans to use to substitute for the courses of the academic department."

***UC09. Approve Course Equivalency Form | 2 hours | This is 2 because it simply is a form that changes a database value.***

| **Use Case List** | **Estimated Hours** |
|---|---|
| *UC01: Search For Programs* | *8* |
| *UC02: Display Program Detail* | *3* |
| *UC03: Submit Online Application* | *21* |
| *UC04: Login* | *34* |
| *UC05: Logout* | *1* |
| *UC06: Edit Online Application* | *3* |
| *UC07: Check Application Status* | *2* |
| *UC08: Submit Recommendation* | *3* |
| *UC09: Approve Course Equivalency* | *2* |
| | |
| **Total Hours Estimated:** | **77** |

## c. Agile Iteration Planning

*Describe agile iteration planning. Discuss why scheduling and resource allocation are necessary for a successful project. Discuss differences between agile iteration planning and convention scheduling and resource allocation techniques. Add the agile iteration planning completed in Week 1 assignment.*

*Agile iteration planning helps us produce a product within our deadline and budget in an efficient manner and allows us to publish with basic functionality and produce updates with added functionality. This is crucial because it prioritizes needed functionality over non-essential useful ones and extra ones. This also cuts down initial production costs of the software system. The benefits of this over traditional iteration planning is better cost effectiveness, quicker time to market, and better planning for future updates.*

**1. Dependencies:**

Identify the dependencies among the nine use cases for the study abroad application:

- UC1: NONE
- UC2: UC8
- UC3: UC9, UC1
- UC4: UC3
- UC5: UC4
- UC6: UC4
- UC7: UC2
- UC8: NONE
- UC9: NONE
- UC10: UC5, UC7

UC1

- points: 5
- priority: 1

UC2

- points: 4
- priority: 4

UC3

- points: 2
- priority: 1

UC4

- points: 5
- priority: 2

UC5

- points: 3
- priority: 4

UC6

- points: 6

- priority: 5

UC7

- points: 5
- priority: 4

UC8

- points: 5
- priority: 3

UC9

- points: 4
- priority: 2

UC10

- points: 6
- priority: 3

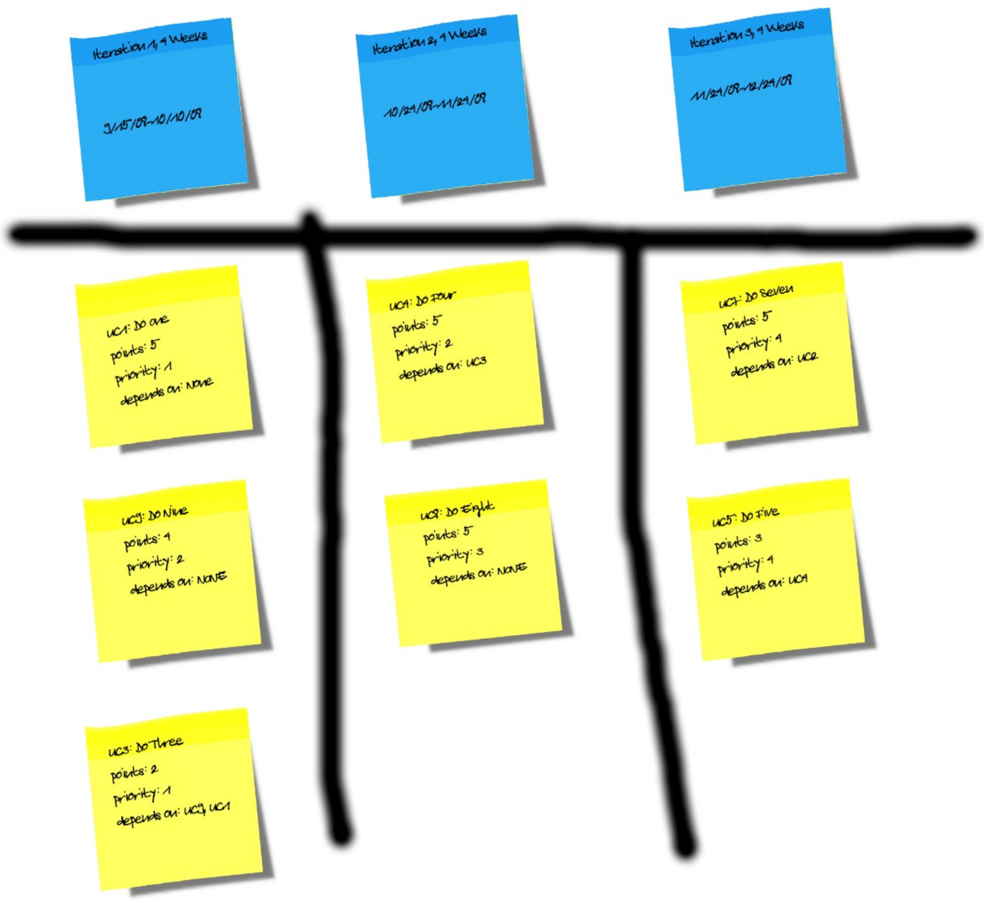
| **Month 1** | **Points** | **Priority** | **Month 2** | **Points** | **Priority** | **Month 3** | **Points** | **Priority** |
|---|---|---|---|---|---|---|---|---|
| *UC1* | *5* | *1* | *UC4* | *5* | *2* | *UC7* | *5* | *4* |
| *UC9* | *4* | *2* | *UC8* | *5* | *3* | *UC5* | *3* | *4* |
| *UC3* | *2* | *1* | | | | | | |
| ***Total Points:*** | ***11*** | | ***Total Points:*** | ***10*** | | ***Total Points:*** | ***8*** | |

**(Figure 1.1)**

## d. Risk identification and resolution

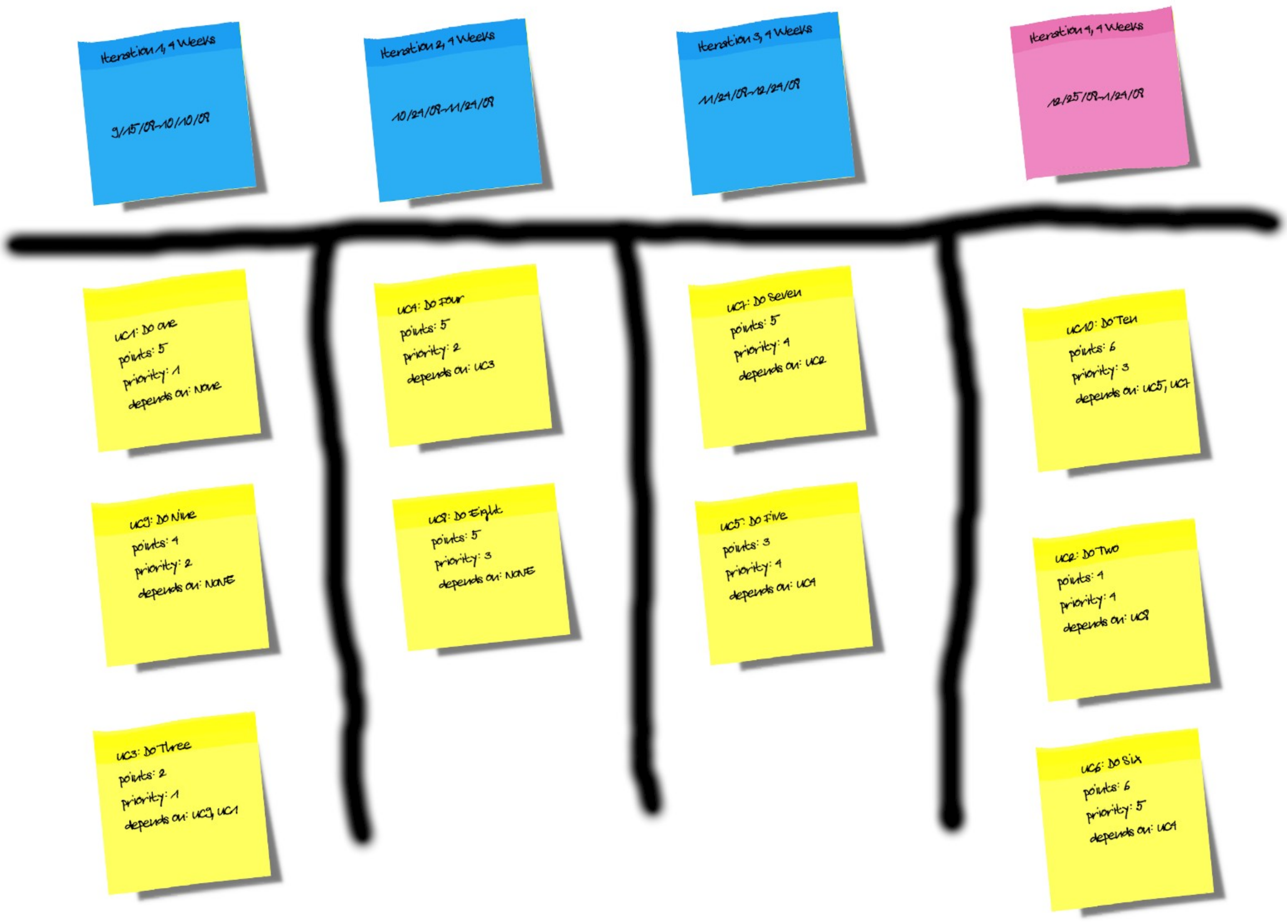
*Describe Risk identification and resolution. Discuss why risk identification and resolution are essential for a successful project. Add the risks identified and the resolutions determined in the Week 1 assignment.*

*Because of dependency issues and priority, I followed the simple rules. Max points per week were 3 equaling to max points per month being 12. I started with cutting out each UC(x) into cards, then placed them all in order by priority. I then found any cards with dependencies. If the dependency was not in front of the card on the list, I placed it in front regardless of it's priority (because that item can't be completed without the dependency). If the next item on the list places the month over the 12 points, I placed that item on the top of the list for the following month. I ended up with the table above when completed. Since I can't go over 12 points per month, the next items in queue would be UC10, UC2, and UC6 in that order.*



**(Figure 1.2)**

In figure 1.2, the board displays the organization of my chart in figure 1.1. As previously mentioned, some of the tasks are left out due to dependency issues, and points adding up more than allowed.



**(Figure 1.3)**

Figure 1.3 shows the same chart as Figure 1.2 but it adds a column at the end in pink depicting all the tasks not completed in a theoretical 4th month. The points add up to more than 12 allowed for a month but this is just to show the items that got bumped due to priority and dependency issues.

Because the project can not be completed in three one month iterations, there are two possibilities to fix this issue.

(1) Hire more staff - This will add administrative overhead but will allow us to complete the project on time. Since we currently have three staff members and can handle 3 points per week, this would equate to a point per staff member. We can calculate the staffing needed to complete the project on time and hire the correct number of staff members as well as get an accurate measure of cost by knowing their salary. Since some projects come with an expensive penalty for late projects, this would be my most likely of the two solutions.

*(2) Extend the project deadline – Since the staff might not be temporary, if there is no staff within the company to allocate to the project, extending the deadline might be a suitable alternative as the company would not have to lay off the extra hired help if they are not needed after the job is completed. Many states including mine allow automatic approval of unemployment benefits for staff layed off without valid termination reason other than lack of work. This sometimes reflects in unemployment insurance cost which adds up over time.*

## e. Configuration Management Baselines and ICs

*Describe Baselines and configuration items. Discuss differences in configuration management between agile and conventional project. Discuss why configuration management is important for a successful project. Add the baselines and ICs identified in the Week 2 assignment.*

1. Determine baselines for an agile approach to creating the SAMS website. Baselines are milestones and associated Configuration Items

a. Identify the baselines (milestones)

i. The start of the project, each iteration, and the project close are milestones - Identify each iteration by the iteration number.

ii. SAMS (The overall project)

1. Planning

2. Iterations

a. Iteration_1

Database - this task is to create a relational database that would be essential for our project. Since our project is a network project with accounts, item submissions, status's, and response, it would make sense to utilize a relational database to make keeping track of records seamless, easier to maintain, and scalable for the future.

b. Iteration_2

Backend - Since we are using a database, we could possibly use PHP or JavaScript to interact with it. We will need a backend, or simply a bunch of useful coded functions that can retrieve or store information from our front ends to and from the database as well as perform searches by scraping sites from study abroad programs for their offerings. The backend could even include the code for submitting applications, editing applications, and replying to applications. The backend could also utilize technologies like JSON or XML for communicating data exchanges between the scraped content and the UI along with data pulled from the SQL server.

c. Iteration_3

Frontend - this is the user interface. It has to be simple and intuitive. The more straight forward it is makes it more appealing and pleasant of an experience for the users. This also makes for fewer mistakes and less frustration. The interface should also be responsive. A responsive interface would also make it much easier to produce specialty and mobile applications making it much more accessible to students and easier for them to check the status of their applications. A mobile application for this could simply be creating a WebView in the mobile application and locking it to the SAMS webpage URL.

i. Development phases within each iteration are milestones.

1. Planning
2. Requirement
3. Design
4. Implementation

5. Integration

6. Validation

3. Close Project

## a. Change Control

*Describe change control. Discuss why change control is necessary for a successful project. Add the change control auditing portion from the Week 2 assignment.*

Any changes to requirements or any document / code after the baseline is established should require a change proposal at this point. A change proposal should include a description of the change requested, a reason for the change requested, a brief description of how this change will effect other aspects of the project / what changes would have to be completed in the project to implement this change.

Tests will be created for validation. When parts of the project are finished and function correctly, it's corresponding test should then pass. If one fails, then it will have to be fixed to continue.

*Change control auditing will help ensure the changes in approved ECP's are implemented in a timely manner by ensuring that changes are requested in a formal process allowing each request to be properly assessed. This also allows proper planning and design of the changes to complete giving a fairly accurate time estimate for completion. Following the same agile process, this change can be implemented on time. It also allows the change to be assessed for completeness and compliance ensuring quality of service as well.*

## f. Coding Standards

*Describe coding standard. Discuss how coding standards contribute to a successful project. Add the coding standards created in the Week 3 lab.*

**PURPOSE:**

*The purpose of this document is to set a standard and guideline so that expectations can be accurate as to the responsibilities and duty of each developer and team member working on the project. The standard set forth will also build consistency in coding style making it easier to review code and read through code as well as help reduce coding errors and reduce the bugs that slip through the cracks. Improving coding efficiency also increases our chances of reducing the time to market for* *product completion.*

**Required Items:**

*Required items will not be followed by an "*"*

**Optional Items:**

*Optional items will be followed by an "*"*

**File header *:**

*Begin files by describing attributes of its content (Tailor the amount of content in the file header to type of organization, and process used conventional, or agile)*

**File header format *:**

*File Location: Path to the file*

*File Name: Full path name of the file that contains the program including the device name / location.*

*Version number: The version number and release number (useful for versioning, maintenance, and software reuse)*

*Author name: Name of the programmer who creates the program file and writes the first version of the program.*

*Project name: Full name of the project for which the program is initially written.*

*Organization: Full name of the organization*

*Copyright: Copyright information*

*Related Requirements: A list of requirements implemented by the programmer*

*List of classes*: A list of the names of the classes contained in the program file*

*Related documents: A list of related documents, and their URLs if available.*

*Update history: A list of updates, each specifies the date of the update, who performs the update, what is updated, why the update, and possibly the impact, risks, and resolution measurements.*

*Reviewers*: A list of reviewers and what each of them reviews*

*Test cases*: A list of test scripts along with their URLs if available*

*Functional Description: An overall description of the functionality and behavior of the program-that is, what the program does and how it interacts with its client.*

*Error Messages: A list of error messages that the program may generate along with a brief description of each of them.*

*Assumptions: A list of conditions that must be satisfied, or may affect the operation of the program.*

*Constraints: A list of restrictions on the use of the program including restrictions on the input, environment, and various other variables (Kung, 2014 p. 452).*

**File contents:**

*List descriptions of sections, classes, operations in the file*

**Description of classes:**

*Class description – provide a functional description of each class*

*(a) Purpose – a statement of the purpose of the class*

*(b) Description of methods – a brief description of the purpose of each method, the parameters, return type, and other input and output such as files, database tables, and text fields accessed and updated by the method. The list should not include the ordinary get and set functions.*

*(c) Descriptions of fields – a brief description of each field including the name of the field, data type, range of values, initialization requirements, and values of significance.*

*(d) In-code comments-comments that are inserted at places of the code to facilitate understanding and tool processing (Kung, 2014 p. 451).*

**Naming conventions:**

*These protocols specify the rules for naming packages, modules, paths, files, classes attributes, functions, variables, constants, and the like. Naming conventions should help program understanding and maintenance. The names that are selected for the classes, attributes, functions, and variables should convey their functionality and semantics, and facilitate understanding of the program. Moreover, the names should be the same or easily relate to the names in the design. Meaningless names such as "flag", "temp", "x", "y", "z" and the like, must not be used. Also, names must not contain space, control characters, or special letters (Kung 452)*

**Naming convention formats:**

*Packages: Names must be meaningful and use lowercase alphabet only.*

*Modules: Names must be meaningful and use lowercase alphabet only.*

*Paths: All files should be kept in their respective directory paths. For example, source kept in the "src" directory. Subdirectories to organize files in a directory will be allowed if*

> *(1) The directory has many files*
>
> *(2) The sub directory must have a meaningful and generic names such as lib for all class libraries, or res for all resource files*

*Files: Names must be meaningful and use lowercase alphabet only.*

*Classes: Names must be meaningful and use camel case alphabet only.*

*Attributes: Names must be meaningful and use camel case alphabet only.*

*Functions: Names must be meaningful and use camel case alphabet only.*

*Variables: Names must be meaningful and use camel case alphanumeric only.*

*Constants: Names must be meaning and use camel case alphanumeric only.*

**Source code formatting rules:**

*Formatting conventions specify formatting rules used to arrange program statements. These include line break, indentation, alignment, and spacing. Most integrated development environments (IDE) define a default format, which can be changed per the needs of the software development organization. In Figure 18.3, the formatting conventions are illustrated (Kung, 2014 p. 453).*

**Source code formats:**

*Line break: To keep code neat and legible, line breaks will be used according to the standards set forth in the Linux Kernel variant of K&R style.*

*Indentation:To keep code neat and legible, indentation will be used according to the standards set forth in the Linux Kernel variant of K&R style.*

*Alignment:To keep code neat and legible, alignment will be used according to the standards set forth in the Linux Kernel variant of K&R style.*

*Spacing:To keep code neat and legible, spacing will be used according to the standards set forth in the Linux Kernel variant of K&R style.*

*Capitalization: Use of capital letters will conform to the standards set forth in the naming conventions.*

**Comments:**

*In-code comment conventions. If it is written correctly, in-code comments facilitate program understanding and maintenance. Thus, the coding standards should define the requirements and guidelines including the locations where in-code comments must, or should be provided, and the formats to write the comments so that tools can process them (Kung, 2014, p. 455).*

**Comments formatting:**

*Comments must be provided in the following places within the source code:*

*(a) Above a code line*

*comments must have the following format:*

*(a) For Java, C, and C++, preceded by "/*" and ended with "*/"*

*(b) For HTML, preceeded by "<!--" and ended with "-->"*

*comments must be used for the following purpose only:*

*(a) To describe or explain a block of code*

*(b) To describe or explain a function / method / class / object / template*

*(c) To display an information header*

*(d) To address or point out sections of code that are questionable and need attention*

*(e) To describe the entire file, it's contents, and it's purpose (at the beginning of the document)*

**Rules for implementing, changing coding standards:**

*Rules and responsibilities for creating and implementing the coding standards as well as reviewing and improving the practice must be defined. Before any change can occur, a developer must submit a request using a predefined request form. The request must first be reviewed and signed off by a team lead if not done by a team lead before it can be given to a project manager. Request will be evaluated and only approved if the change can make a meaningful and positive impact in compliance with the project requirements.*

## g. Software Quality Assurance Plan

*Describe software quality assurance. Discuss how a quality assurance plan contributes to a successful project. Add the quality assurance plan created during the Week 4 lab. If not already in the plan, add the quality attributes, metrics, verification, and validation, and QA techniques describe in the Week 3 assignment.*

### i. Purpose

The purpose of the SAMS project is to connect students with data about study abroad programs available and allow the students to interact with staff, apply for postings, check application status, and request letters of recommendations. It also allows the staff to manage the students and receive requests, post postings and letters of recommendations and so on.

ii. **Management**

We will be using an Agile development methodology to implement the project. Each role will be assigned to appropriate staff, and the project will be planned out before development begins starting with a draft of the business requirements and UML class models along with the other design documents discussed in the course and the previous course (such as GANTT and PERT charts). The UML class models along with other design documentation give an expectation of every aspect of the program so that work can be divided among the developers and each one knows what to expect from the others when coding the program. It also gives us a skeleton to work with to design unit testing.

iii. **Standards and Conventions**

There will be templates describing what information is required in the header of all source files, header files, and any other code documents. There will be documentation specifying what information is to be put in code comments, and what is to be left out. There will also be standards on coding styles such as bracket placement. I am a huge fan of Linux software development and would one day like to contribute to mainline kernel, so I try to use the Linux Kernel variant of K&R style on my work these days to build the habit. If running a project, I would lay this style down as the foundation and expect everyone to use the same style as to keep code consistency and neatness.

iv. **Reviews and Audits**

Reviews and audits will be mandatory as well as unit testing. This will ensure that our code is functional and does not have any bugs or errors. Reviews and audits also help expose security vulnerabilities as well. If problems are found, then the code is obviously not going to meet business requirements and must be corrected immediately.

v. **Configuration management**

An SCM will define every process and product used during development life cycle to ensure that the end product meets the business software requirements.

vi. **Processes, Methodologies, Tools and Techniques**

We will use Agile Methodology for project planning and implementation. We will use the Linux Kernel variant of K&R for coding style guidelines. We will have templates for information required in the header of source and header files. Developers should use a private GitLab server with CI/CD pipelines configured and enabled. They should be familiar with the use of CI/CD techniques. Since this is a web technologies project, I would expect that the project be developed with the intentions of running on a LAMP server (one of the more popular industry standards). Since the code will be running in LAMP servers, the CI/CD should all be using a docker image of the same LAMP server OS in order to maintain compatibility. Developers should also be using gcc as the compiler for their code. Testing should be done on a distributed VM image of the same OS used for the server to ensure compatibility. A preconfigured VM image can be distributed to ensure that everyone has the same environment.

vii. **Metrics and Indicators**

We will use the CMMI development model to determine base measurement requirements for our SQA plan.

**Context/Overview:**

Assume OO design and an agile process for delivery of the SAMS project. Consider the software quality assurance needs of the SAMS project.

## 1. Specify quality attributes applicable to the SAMS project

**Quality Attributes:** A software quality attribute is a software attribute that describes, or characterizes, a quality-related aspect of the software. (Kung, p470)

**Reliability Attributes:** Ability of the software to perform its required functions under stated conditions and produce correct and consistent results.

**Robustness**: The ability of the software to perform it’s required functions under rough or exceptional conditions.

**Efficiency:** The ability of the software to perform its functions and produce desired results with minimum expenditure of time and resources

**Interoperability:** The ability of the software to interact and exchange information with other software.

**Maintainability:** The aptitude of the software to undergo repairs and evolution.

**Testability:** The aptitude of the software to permit all desired and applicable forms of assessment, including inspection, peer review, white-box testing, and black-box testing.

**Portability:** The aptitude of the software to permit itself to be transported to run on different operating environments or platforms.

**Reusability:** The aptitude of the software to facilitate integration or arrangement of its component modules.

**Modularity:** The aptitude of the software to facilitate integration or arrangement of its components modules.

**Cohesion:** The degree of relevance of the functions of a software module with respect to the module’s core functionality.

**Coupling:** The degree of dependency and interaction between a module and other modules.

- **Operating Systems:** To keep results consistent, all the web services should be hosted on the same operating system with the same software setup, and security settings. This could also provide the benefit of setting up one LAMP server and duplicating this setup for rapid deployment across all servers for the project. It also makes load balanced services identical on each server.
- **Network Load:** This project will deal with several aspects that could be placed under load. For this example, network load will be a big one to consider. This project deals with uploading and distributing documents such as letters of recommendations and applications. These are shared to the server and distributed among select students and staff. A network load test to determine the max capacity of the network in terms of users uploading at once will give valuable analytical data as to when to scale (on a load balanced design).
- **User Capacity Load:** This attribute will test when the service reaches the max capacity of users logged in at once.
- **Database Load:** The database for the SAMS project should be tested for query load.
- **Hardware:** A change in hardware could change capacity and load analytical data. Keeping the system specs as close as possible would provide greater accuracy in determining when to scale the project.
- **Communication Protocols:** Much of the project will communicate via TCP/IP stack as it is a network application project. Some aspects of the project will communicate between services and servers on the network as well as perform network load balancing tasks. Some aspects might utilize network sockets. Backend services might also utilize ports other than port 80.
- **Unit Testing:** Testing functions for things like out of range values can help determine the quality of error checking and enforcing correct input values.
- The system must also be designed to be easy to integrate updates and integrate into current systems (such as the 3rd party sites where it polls for data about available classes from partner programs abroad).
- We also want to keep coupling low as to prevent errors across the board when we modify a line of code in one module.

**2. Evaluate the requirements, design, implementation and system metrics in figure 19.2. Which metrics will you use to assess the quality of the SAMS project, and why?**

To assess the quality, I will use a combination of Reliability and Availability metrics, and Object Oriented Quality Metrics. I chose these because failure to comply with these metrics could produce the most noticeable bad results to the end user of the software project. These metrics can help us direct resources to critical areas of the project.

**3. Which static and dynamic verification and validation techniques will you use in the SAMS project?**

For static, we can use inspection, walkthrough, and peer review. For dynamic, we can develop functional and non-functional tests for the sams project to ensure that it meets the requirements.

**4. What QA techniques will you use to ensure the quality of the SAMS software?**

Software Development Activities, Software Requirements analysis, Software Design, Software Implementation, Integration and system testing, and Software operating and maintenance will be the basic process used to ensure the quality of the SAMS software.

## h. Software Test Plan

*Describe software testing. Discussion how software testing contributes to a successful project. Add the software test plan create in Week 5 assignment.*

*1. Select one use case in the SAMS project, and follow steps on pages 519 -521 in the textbook to generate test cases.*

*i. Submit:*

·1 *The expanded use case, like figure 20.12*

·2 *A table identifying input values for use case testing, like figure 20.13*

·3 *Test case generation table, like figure 20.14*

·4 *Use case based test data, like figure 20.15*

2. *Follow guidance on pages 526-532 to create a test plan for the SAMS website.*

   i. *A test plan should include:*

·1 *Test objectives*

·2 *Types of tests*

·3 *Test methods and techniques*

·4 *Test cases—Include just the test case created in 1*

·5 *Test coverage criteria*

·6 *Documents needed*

·7 *Required resources*

·8 *Effort estimation and schedule*

***1. Use Case:***

UC04. Login (Actor: Student, System: SAMS) This use case allows a registered student to login to the

system.

UC04. Login | 34 hours | I chose 34 because this task requires us to design a database for our

project. The database must have carefully designed tables to prevent data corruption. It might be beneficial to also add LDAP as the authentication mechanism as this standard can

be used for other applications on campus as well including campus email, student portal

access, thus limiting account creations. If this system is already implemented, we can

*probably save a few hours by utilizing the existing LDAP system.*

***1.1 Expanded Use Case***

| **Actor: Existing User** | **System: SAMS Web Login** |
|---|---|
| | 0. System displays homepage with login / register |
| 1. User clicks login link. | 2. System displays login form |
| 3. User fills in login ID & password, then clicks login | 4. System verifies the login ID, password and |
| | 4.1 displays student portal page or |
| | 4.2 displays error message and stays on login page. |

***1.2 Identify Input Values***

| **Input Element** | **Type** | **Value Specification** | **Valid** | **Invalid** | **Exceptional Cases** |
|---|---|---|---|---|---|
| Login ID | String | Length must be between 8 to 20 characters | Login ID meets input validation criteria and is a registered LDAP user | Login ID does not meet acceptable criteria or is not an LDAP user | Length of 0 |
| Password | Password | Length must be between 8 to 20 characters | Password meets input validation criteria and its hash is | Login password does not meet input criteria or is not associated with the user | Length of 0 |

validated with associated user

### *1.3 Test Case Generation Table*

| Test Case | Login ID | Password | Expected Outcome |
|---|---|---|---|
| 1 | Valid | Valid | display student portal |
| 2 | Valid | Invalid | display error and stay on login page |
| 3 | Valid | Exceptional | display error and stay on login page |
| 4 | Invalid | Valid | display error and stay on login page |
| 5 | Exceptional | Valid | display error and stay on login page |
| 6 | Exceptional | Exceptional | display error and stay on login page |

### **1.4 Use case based test data**

| Test Case | Login ID | Password | Expected Outcome |
|---|---|---|---|
| 1 | johndoe@myuniversity.edu | j4h6PcZ9xdsSnx5e | display student portal |
| 2 | johndoe@myuniversity.edu | pass | display error and stay on login page |
| 3 | johndoe@myuniversity.edu | VHhQkt8Tu9ZyYDJGD7aUb | display error and stay on login page |
| 4 | johndoe | vBHxZ3gMa5h | display error and |

| | | | |
|---|---|---|---|
| | | U3sAa | stay on login page |
| 5 | VHhQkt8Tu9ZyYDJGD7aUbVHhQkt8Tu9ZyYDJGD7aUbVHhQkt8Tu9ZyYDJGD7aUb@myuniversity.edu | hVpQj75KmUJWttxZ | display error and stay on login page |
| 6 | VHhQkt8Tu9ZyYDJGD7aUbVHhQkt8Tu9ZyYDJGD7aUbVHhQkt8Tu9ZyYDJGD7aUb@myuniversity.edu | a | display error and stay on login page |

### *2.1 Test Objectives*

*The objective of the tests are to validate user input so that we can ensure effective and secure login practices.*

### *2.2 Types of tests*

*We will be using user validation input testing.*

### *2.3 Test methods and techniques*

*For this, we can use HttpUnit to submit our tests. With this, we can automate most of the tasks. We know our inputs and our expected outputs. If our inputs and expected outputs don't match, we will know that we have a problem in our code.*

### *2.4 Test cases*

| **Test Case** | **Login ID** | **Password** | **Expected Outcome** |
|---|---|---|---|
| 1 | Valid | Valid | display student portal |
| 2 | Valid | Invalid | display error and stay on login page |
| 3 | Valid | Exceptional | display error and stay on login page |
| 4 | Inva | Vali | display error and stay on |

| | | | |
|---|---|---|---|
| | lid | d | login page |
| 5 | Exceptional | Valid | display error and stay on login page |
| 6 | Exceptional | Exceptional | display error and stay on login page |

***2.5 Test coverage criteria***

*The tests will cover valid, invalid, and exceptional input of all user input fields. All fields are required to pass testing in order to meet requirements*

***2.6 Documents needed***

*We will require documents for use cases as well as UML models of the SAMS project. These will be useful for drafting our test case scenarios.*

***2.7 Required resources***

*We will require access to the current LDAP system employed on campus to create a functional dummy account to test our SAMS project.*

***2.8 Effort estimation***

*Since most of the testing can become automated and implemented into the other development phases of the project, I would estimate no more than an hour or two to write / implement tests for UC04.*

## i. Security Plan

*Describe software security. Discuss how software security contributes to a successful project. Add the security plan created in the Week 6 assignment.*

*Context/Overview:*

*Use the security-related activities in the life cycle shown in figure 24.4 to create at security test plan.*

*Resources to consult:*

*Chapter 24*

*Specific questions or items to address:*

***1. Domain Model from Figure 24.8***

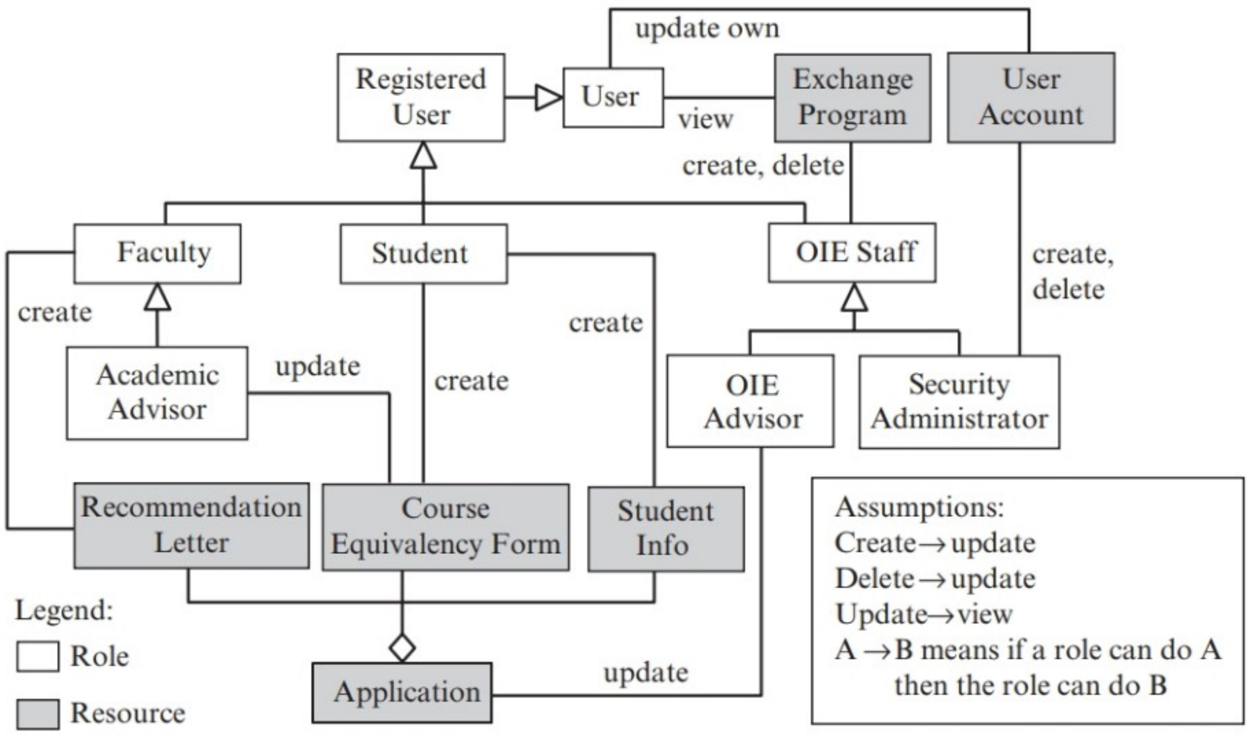


**FIGURE 24.8** Part of a domain model showing security information

***2. Identify security threats and derive security requirements and misuse cases. See examples 24.2 and 24.3***

*Security Threats*

*Every aspect of the system can be miss used if security measures are not properly implemented. Here are the threats that I can identify for this project…*

- *Brute force attack on servers (Web Logins, SSH and SQL Services)*
- *Exploit of default user names and passwords*
- *Man in the middle attacks*
- *Social Engineering attacks*
- *Phishing Attacks*
- *Denial of Service Attacks*
- *Domain Name hijacking / ransom holds*

- *Input manipulation*

*To combat these misuse cases… the following requirements shall be set*

*(a) All servers in use shall deploy a properly configured IDS/IPS + firewall to combat brute force attacks, vulnerability scanning and port scanning.*

*(b) All servers in use shall deploy a properly maintained antivirus and antimalware system to combat rogue payloads that can compromise the network.*

*(c) The web forms and interfaces shall validate all input types.*

*(d) No default username's and passwords will be in use. All of these will be disabled immediately after an admin account is created with a unique username and password.*

*(e) Root login will not be allowed on any SSH server. An admin will use their user account and escalate with su or sudo during remote connections.*

*(f) All SSH connections will use a generated SSH key. No password authentication will be allowed.*

*(g) Domain names will be set to auto renewal to prevent laps where the name can be hijacked and held for ransom.*

*(h) No links will be sent to any messenger or email. Any notice will require the user to manually login from a browser or approved web application. This will help prevent phishing.*

*(i) No staff will ever ask for personal information or account information via any communication type. The user will be required to contact staff directly in order to prevent social engineering attacks.*

*(j) All users will be made aware of staff procedures by receiving notices of what the staff will not do in order to maintain privacy. Copies of this will also be a part of the privacy notice.*

*(k) All users will be required to read and agree to the user agreement, privacy notice, and security policy in order to use their account prior to first login.*

*(l) All users will be required to read and agree to any privacy policy, security policy, or user agreement updates, changes, and notices upon login before being allowed to proceed using the app.*

*(m) Changes to user data will require 2-factor validation of the following by sending a code that must be entered into a data field on the web app...*

1. *user email*
2. *user phone textbook*

*(n) Users will ONLY be assigned access and privileges based on their job duties*

*and roles. They will not receive any higher access to the system than what is needed. Access not needed will be stripped away after (see section 3 for details)*

## ***3. Specify role-based access rights***

*Users will ONLY be assigned access and privileges based on their job duties and roles. They will not receive any higher access to the system than what is needed. Access not needed will be stripped away after.*

*The main user roles will be…*

*(1) Administrator – has access to all accounts as well as code, database, logs, and analytic data required to maintain the site.*

*(2) Student – the student can browse available abroad courses and open seats. They can apply to them, submit documentation in the portal, and check status of pending applications along with view the account actions history including documents submitted and dates submitted.*

*(3) Advisor / Counselor – A student advisor will be given access rights to log into a student's account to assist the student in setup and search. The student will have the ability to grant and revoke access from the settings. Any changes an advisor makes will appear in the students dashboard on the portal and will require the student to review and submit manually themselves giving them the opportunity to review the changes / additions made to their application.*

*(4) Staff – Staff will have the ability to upload recommendation letters to the students portal, as well as have access to analytical data and insights as to the status of applications and documents.*

*(5) Admissions – This type of user will have the ability to register classes with the system as well as approve or deny "their applicants". They will have the ability to review and request pending applications as they come in. They also have the ability to suspend application submissions when their seats are filled. They can even have one click access for direct contact with the students and staff that recommended them and send recommendation letters in order to request more information or address followup questions or concerns.*

*(6) Outsiders – This type of access can be granted without login in the form of a single web page that gives statistics and analytical data without any personal information attached. This could be useful for board members, staff, and students to view real time analytical data about applications submitted, applications approved, and they can even sort data by location allowing them to see where their specific school or desired location fits into the analytical data. This data along can be used to determine funding, or popularity of a program. For example, if students are not applying and the cost to maintain the program is too high, a board might vote to cut a particular program that isn't receiving well. A school can decide to grant more*

*funding if they find that they are constantly overbooked with foreign students and need to gain more seating in the class rooms in order to accommodate more studnets.*

***4. Discuss what will need to happen when requirements change. See section 24.7.2 Security in the Iterative Phase.***

*When requirements change, a change proposal must first be made. Before the change can be implemented, we must first draft a new domain model to show security related relationships and formulate a list of security threats. Me also assess how this could effect the components of the system and the role based access rights.*

***5. Discuss security in implementation, testing, and deployment***

*Security will be implemented just as stated. It will be tested and deployed in a physical work environment. If a security related test fails, the system will not deploy until it is fixed. If a security related bug or vulnerability is found after deployment, a new security requirement will be made, and a test will be made from this requirement. A bug fix will be coded and tested against this requirement and once it is met, an update will be pushed to correct the issue. The same goes for the human element as well, anyone who's account has been compromised due to fishing and social engineering will receive an email explaining how these type of exploits work and how to avoid them as well as their account requiring them to review and agree to the privacy policy all over again.*

## j. Software Maintenance Plan

*Describe software maintenance. Discuss how software maintenance contributes to a successful project. Add the software maintenance plan created in the Week 7 lab.*

***I.Description of Software Maintenance***

*Software maintenance is modifying a software system or component after delivery to correct faults, improve performance, add new capabilities, or adapt to a changed environment. (IEEE Standard 610.12-1991)*

*Software maintenance is very important for keeping a project fully functional in an ever evolving world and changing environment. Some software systems are supported decades after release making them legacy software. These software systems must undergo changes to make them compatible with newer hardware, operating systems, and associated components.*

## *II.Factors that Mandate Change*

*We will mandate change when we need*

- *Bug Fixes*
- *There is a change in operating environment*
- *Change in government policies and regulations*
- *Change in business procedures*
- *Changes to prevent future problems*

## *III.Maintenance Process Models*

*The IEEE process model would be best suited for this type of project. Here are the phases of the model...*

*Problem Identification / Classification, Analysis, Design, Implementation, Regression / System Testing / Acceptance Testing, and Delivery. This process model can fit in nicely with our agile techniques and allow us to continuously maintain our project in an efficient manner ensuring quality code along the way.*

## *IV.Types of Software Maintenance*

*This project will employ Corrective maintenance, adaptive maintenance, perfective maintenance, and emergency maintenance. All of these types of maintenance are crucial to providing a smooth and functional experience to all users of the finished product as well as protect their user data.*

## *V.Software Reverse Engineering*

*Software reverse engineering is a process that converts the code to recover the design, specification, and a problem statement (Kung, 2013). This could be useful in examining 3rd party components and designing better integration tools to work with them. The better we understand them, the better the system as a whole can be.*

## ***VI.Software reengineering***

*Software reengineering is a process that restructures a software system or component to improve certain aspects of the software (Kung, 2013). As stated in the previous section, this will be useful in developing better integration with third party components of the system.*

***Project Git Address:***

*https://github.com/tomtravers2002/TT_CS406_SAMS*

***Sources:***

*610.12 (H)-1990 - IEEE Standard Glossary of Software Engineering Terminology (HyperCard Stack).*

*(n.d.). Retrieved August 03, 2020, from https://standards.ieee.org/standard/610_12(H)-1990.html*

*Kung, David. (2013). Object-Oriented Software Engineering: An Agile Unified Methodology (1st ed).*

*New York, NY: McGraw-Hill Education*

*ISBN: 9781264056149*