**Resource Manager**

Elaboration Phase Status Assessment (LCAM)

1. Assessment Against Objectives of the Elaboration Phase
   1. Did we complete the expected outcomes for LCAM?

The aim of the project is to build a resource management tool for an organisation. This is shown in the completed *Critical Core Risky Difficult* (CCRD) demonstrator.

The main Use Case in the system is understood to be *Allocate Resource.* This is shown in the completed Full Use Case Description *Allocate Resource.*

The main functions of this use case are:

* Ensure skilled resources available for a project can be located quickly
* Ensure resources cannot be double-booked
* Ensure resources can only be booked if their allocation is appropriate

To achieve our goals within the development of this system we are going to use a web application that stores data in a database, using various persistence frameworks. This is shown in the completed *Architecture Notebook.*

Project-specific risks and their respective methods to deal with them have been acknowledged. The risks are as follows:

* Appropriately securing information
* Support maintenance by sponsor
* Adequate skillset of the team
* Scope creep/under-scoped.

Our evolving understanding of risks is shown in the ongoing *Risk List* and discussed further below in *Section 4*.

We acknowledge the methods on which we will check the system integrity upon delivery. To ensure the functionality and system properties are delivered fittingly, we will use the testing strategies as follows:

* Functional testing
* Unit testing
* Integration Testing
* User acceptance testing

This is shown in the *Master Test Plan*.

Dependencies and completion times for certain areas of the project have been acknowledged by the team. Target completion dates for key aspects of the project are as follows:

* **Architecture** (13/05/2019)
  + Implement 1st & 2nd highest priority to support CCRD Use Case.
* **Functionality** (24/06/2019)
  + Top 2 CCRD Use Cases to be implemented.
* **Testing** (ongoing)
  + Iterative testing will be performed as new functionality is added. Once stable, a UAT of that functionality will be performed.
* **Documentation** (26/08/2019)
  + Complete user documentation - how to use the system for each user, admin documentation that explains how to configure and setup system.

This is shown in the *Project Plan.*

* 1. Skills Required

Our project requires skills using the following key tools and technologies:

* MySQL
* Java
* Maven
* Git
* Tomcat
* SpringBoot
* JPA
* Hibernate
* JavaScript
* HTML5
* CSS

We have demonstrated that we have the skills to use these technologies through the implementation of the CCRD Use Case.

1. Deliverables
   1. Project Vision

This document only contains a rough outline of the intended application. There were no issues producing the document.

* 1. Requirement Model

Further requirements for this component were acknowledged throughout the *Elaboration* phase. These will be implemented during the *Construction* phase. Outlining the CCRD Use Case was a high priority that was achieved by the team.

* 1. Proposed Architecture

This document states how we will combine all required components. There was no issue here.

* 1. Risk List

As new risks and issues can occur at any point, this document cannot be completed until the project is complete. The document is well written and contains key components in which the team must look out for. There were no issues producing this document.

* 1. Master Test Plan

There were no issues here.

* 1. Initial Project Plan

There were some issues involving unclear outlines to what work was needed to be done. These issues have been corrected and the plan now thoroughly outlines the work required.

* 1. CCRD Implementation

We experience issues implementing the persistence layer with JPA and Hibernate as they are quite strict. They do not permit having objects in the memory cache that do not match the persistence layer after an update. This means all relationships in terms of one-to-one, one-to-many and many-to-many and their directionality of uni/bi direction must be correctly defined in each of the models. The code must reflect the directionality by cascading changes to the children of a parent and the parent of a child if an update is made to one. This ensures the state is consistent at the memory cache and persistence layer levels, which is great in that it supports our requirements for reliable and bug free code. That level of thoroughness is a lot more difficult and time consuming to implement. We spent an inordinate amount of time just tweaking the relationships to make sure everything cascaded as it should without throwing an error. This was especially an issue for the CCRD as it involved a three way many-to-many relationship, many projects which require many skills and allocate many resources that are proficient with many skills. Unit and Integration testing also had many growing pains, due to the SpringBoot architecture chosen. Spring is noteworthy in that it uses automated annotation-based dependency injection to build the applications ‘context’. This means at any level the programmer can’t just instantiate a new object that uses Spring’s @Autowired annotation, that must be done by Spring for it to be aware of that object in the application context. This took some time to adjust to, especially for unit tests as typically in unit tests one would mock all objects in the context except for the Class under test. Once we had worked that out, it was a great time saver setting up and tearing down the testing environment before and after each test, ensuring there was no contamination between tests.

1. General Issues
   1. Team Communication

The team had difficulties during the elaboration phase where we did not communicate when tasks were completed or when tasks were available for other members to work on. This led to at least one wasted iteration. This issue has now been resolved and the team is communicating more frequently.

1. Risks

4.1 Poor Use of Version Control

This risk can cause project delays if the team does not appropriately use version control, causing the project to fail.

The mitigation strategy is to use a common version control system with available help online.

This risk is resolved.

4.2 Team has Inadequate Skillset

This risk can cause project delays in the form of team members only learning requisite technologies when they need them which can make developing the application take longer than required.

The mitigation strategy for this is to use skills taught during the CSU Computer Science Bachelors and to occasionally have skill quizzes to ensure the teams skills are up to date.

This risk is still open but being managed.

4.3 Team Members Leave the Project

If any members of the team leave the project, due to the small project team, some use cases will have to be removed.

The mitigation strategy is to ensure that team members are all well communicated to and for the team to be open with each other to manage expectations.

This risk is still open and being managed through ensuring the team discusses issues when the occur.

4.4 Inadequate Security

If the program is not secure enough to store data, then the users will not use it.

We will ensure the security of the data is appropriate to the requirements.

This risk has been mitigated through HTTPS and implementing a login page.

4.5 Incapable of Providing Needed Functionality

The risk of not meeting the client’s needs can lead to refusal of product purchase thus a waste of time and money on part of the developers, alongside a potential loss of reputation.

The mitigation strategy is to establish the project requirements and selecting technologies that can deliver the services required by the client.

This risk is being mitigated by iteratively performing UAT with the client.

4.6 Team Unable to Identify or Procure Technologies

If team members are unable to determine the technologies that should be used, the project will cease to proceed past the planning phase.

The mitigation strategy is to investigate early into the technologies to be used.

This risk is resolved.

4.7 Team continues to communicate poorly.

During the Elaboration phase the team had communication issues which are now resolved.

If the team continues this trend the project will likely fail.

This risk is being managed by ensuring an iteration manager is responsible to ensure delivery of all artefacts.

1. Summary

Overall our progress against the Elaboration phase is complete.

The team needs to ensure that we better focus our attention to the required skill sets specifically:

* Hibernate: JPA persistence.
* Java Servlet pages.

The team has had some issues with communication and so attention should be focussed on ensuring communication is maintained into and throughout the Construction phase.

While the team did some upskilling early on a lot of the development of the current CCRD use case was performed by a key team member, the rest of the team needs to ensure they upskill over the break to ensure that they can equally contribute to development.