# Action Points (I have 4 solid one’s so far, you can read through and even add your own)

|  |  |
| --- | --- |
| **Action Point** | **Steps to fix it** |
| The previous iteration of the use case set showed a very poor understanding of invocation of other use cases. Firstly, because we did not specify whether invoked use cases were “extends” or “includes” | This is corrected in the final Use Case Set below. We were forced to evaluate each invoked use case closely for whether it is essential to the use case that invokes it (extends) or if it was only invoked upon certain conditions (includes). |
| The second critique of our Use Case Set was that some use cases did not invoke the necessary Use Cases. | This is also corrected in the final iteration below. We broke down how use cases like Create Application work, and realized how the functionality cannot be completed without invoking another use case (which is Read Applicant in this case) |
| In every one of our fully dressed use cases there was an actor missing. | This highlighted an underlying problem with our understanding of the actors in this case study and in general. Upon consulting with our notes and you, we got to understand what qualifies as an actor. The corrected Fully Dressed Use Case reflects this transformation |
| A problem that showed up frequently in the previous milestone was the inconsistency our document had. Especially between models that build from each other, and between models of the same kind. | The first step we took towards fixing this was assigning a team member the responsibility of quality assurance and consistency through the document. The second was trying to understand the models better, as a lot of the sloppiness came from a lack of understanding. |
| A huge factor in how underwhelming our previous document was, was the time management. The rushed piece of work led to inconsistencies and a lack of quality control. | We worked to combat this by managing our time and allocated tasks better. We worked towards our own deadlines before the submission, thus ensuring that closer to the submission date we can focus more on checking our contributions. We also utilized our meeting time more efficiently by delegating tasks before the meeting; this meant that the time during the meeting was spent more efficiently compiling the final document and not on creating diagrams from scratch. |
| In previous milestones our team did not take full advantage of the feedback process that comes with each iteration of the project. We did not consult or fully engage with the feedback that we received and thus fell behind the other teams in terms of our understanding of compiling professional client documents | Although too little too late, we consulted for this milestone and got a lot of useful feedback off our previous work, and how to proceed going forward. Consulting was not only beneficial to improving the final milestone, but it also helped us better understand the pedagogy of this course, and how additional feedback plays a very huge role in our ideal transformation into better system analysts. |
| There was consistently a problem with the actors listed in the models. And this points to an underlying problem with our understanding of actors. | During consultation and going through our notes this concept was better understood by the whole team. An example of this is how we added additional users to most of the use cases as we better understand how it can be utilized by different actors. |
|  |  |
|  |  |
|  |  |
|  |  |

# Purpose

This document is the second iteration in mapping and showing our current understanding of your organization’s domain and presenting the framework of a system that aims help combat your current problem. Its purpose is to act as a form of contractual agreement between Kelello and the organisation, #fundMe. The document certifies that the organisation, #fundMe has read and reviewed the document and are satisfied with the information presented within the document, thus far in the project; and if not, the issues are to be communicated back to us (Kelello) for further modification. It therefore serves as a sign-off document between the two abovementioned parties to supply each with a form of legal protection.

# //Executive Summary (will compile better after viewing all the components)

A large portion of this milestone is reflecting, and iterating based off the shortcomings of Milestone 2. The

We then proceeded to revise the Class Diagram, Use Case Set and Diagram according to these action points. Notable changes include changing the type of relationships in the Class Diagram and making the wording we use in the Use Case Set more consistent with UML standards.

We then developed some of the use cases into Fully Dressed Use Cases. The Fully Dressed Use Cases contains: a brief description on the task the use case is supposed to perform, a list of the actors on the system, a list of all the stakeholders in the use case and their interests, pre and post conditions, and most importantly the flow of activities which is central to developing the SSD. The Fully Dressed Use Cases are each supported by their respective Use Case Diagram and SSD, the latter which is mainly to model the communication between the system and involved actors. The isolated Use Case is mostly used to model the invocation of other Use Cases.

# Final Class Diagram

A screenshot of a cell phone

Description automatically generated

# Fully Dressed Use Case (Create Donation) and it’s SSD

The Fully Dressed Use Case serves a very important role in communicating the system analyst’s (us) understanding of the client domain. This user-centric model allows for input on whether the system being designed successfully executes the task required of it whilst sticking to the business rules. Though easy for the client to understand, there are certain details to highlight in the Fully Dressed Use Case below. We designed the Create Donation Use Case with close attention to the Administrator’s matching task, and we included the functionality to search and view Application and Donor information as a core component.

|  |  |  |
| --- | --- | --- |
| **Use Case Name:** | Create Donation | |
| **Scope:** | #fundMe system | |
| **Triggering Event:** | When the Administrator matches an Application to a Donor | |
| **Brief Description:** | This use case serves as a means of creating the Donation entity which links a Donor to an Application. It will prompt the Administrator for information on the donation, and this will be acquired from the respective Donor and Application. When all data has been captured the new Donation record will be recorded to the system database. | |
| **Actor(s):** | Administrator (Primary) | |
| **Related Use Cases:** | Read Application (includes)  Update Application (includes)  Read Donor (includes) | |
| **Stakeholders and Interests:** | Application: the application isn’t a typical stakeholder (person) but we feel the entity has a place as the use case acts in the interest of matching each Application to a Donor. The Application status also needs to updated after a Donation has been made  Donor: the interest of the Donor is to be matched with an Application, and in the latest version of the case study there are no preferences for the donor  Administrator: The Administrator matches the Donor to an Application, and their interest in this use case is retrieving the necessary data (using invoked use cases) to perform this task and being able to record the result by saving the new Donation record to the database. | |
| **Pre-Conditions:** | The Administrator must find a suitable match between Donor and Application in order to create a Donation | |
| **Post-Conditions:** | The Administrator must be notified upon the successful execution of the use case  A new Donation record must be added to the database with the relevant details  The Application Status of the Application in the match must be updated to Green | |
| **Flow of Activities:** | 1. Makes a request to the system to create a new Donation 2. The Administrator inputs the search criteria and returns it to the system 3. The Administrator enters the credentials of the new Donation matching an Application to a Donor | * 1. The system prompts the Administrator for search criteria on the Donor and Application   2. The system invokes the Read Applicant use case to search for a suitable Application   3. The system invokes the Read Donor use case to search for a suitable Donor   4. The system displays the search results to the Administrator   5. The system prompts the Administrator for details on the new Donation   6. The system enters the record of the new Donation into the system database with the credentials input by the Administrator   7. The system invokes Update Application to change the status of the Application   8. The system displays confirmation to the Administrator that a Donation has been created. |

A close up of a map

Description automatically generated

# State Machine Diagram

A screenshot of a cell phone

Description automatically generated

# Activity Diagram

# 8. The Activity Diagram serves an important role in representing a use case as it shows both control and flow of data between the actor and the system, though the Diagram is easy for the client to understand there are certain highlights that must be made. We use rake symbols to show activities that invoke related use cases in separate diagrams.

