# Action Points

|  |  |
| --- | --- |
| Feedback | Action Point |
| The purpose of our milestone was lacking in terms of actively pushing the client towards sign off |  |
| The problem statement came across as a solution and not as a problem due to a lack of understanding of what it is |  |
| The Class Diagram had two erroneous relationships between entities | The Application to Applicant relationship should be 1..1 as we can’t have an Applicant without an Application  The same can be said for the relationship between Donor and Admin |
| The Case Set was not clearly stated to define the “read” methods | We should use the word Read instead of Retrieve as this does not clearly express the method |
| The team failed to distinguish between functional and non-functional requirements | Most of the non-functional requirements we listed are functional requirements and should be changed as such. |
| Our team included a reference to the system creating a match within 5 days, which is wrong as this is not in line with what the case study says | Our team should instead change this to the system prompting the administrator to make a match within that time |
| The Features and Benefits includes features that are not linked to the Use Case Set | These should be changed as the Use Case Set is supposed to map out the functionality and features of the whole system |

# 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 will help implement a solution to your problems. We will accomplish this, firstly by adding depth and accuracy to the diagrams presented to you in the previous milestone. And secondly, we will provide fully fleshed out descriptions of important methods the proposed system will have to perform.

# //Executive Summary

We used a Class Diagram to model the domain of the client, and discovered that the main classes of the database will be: Donor, Administrator, Applicant, Application and Donation. This includes

We then modelled the system usage through Use Case Set and Use Case Diagram based off the classes. The administrator will be granted access to most of the uses as they process the matches and deal with the system on behalf of the donor. The student will only have access to creating their applications and profiles.

We also defined the main features of the system. Amongst which is the system being able to aid and manage the process of matching donors and applicants. Another important feature is the system being able to generate overview reports on the system effectiveness; also processing the data and displaying it.

We explored different methods for requirement elicitation, with Requirement Workshops emerged as the clear favorite. This is because it gives access to a lot of stakeholders in a limited amount of time. We supplemented this with more user specific methods like interviews and questionnaires.

We broke down some of their use cases to give you a clear functional requirements. We also detailed 5 non-functional requirements that will make the system more reliable. We identified that a big constraint to us building an optimal system is that we have to build a desktop application specifically for the kiosks (as opposed to a multi-platform system).

# Additional Fully Dressed

|  |  |  |
| --- | --- | --- |
| **Use Case Name:** | Update Application | |
| **Scope:** | #fundMe system | |
| **Triggering Event:** | When an Applicant or Administrator selects the option to edit an existing application | |
| **Brief Description:** | The Applicant or the Donor chooses the option on the system to update an existing application. The user/ actor then enters the number of the Application, which prompts the system to recover the record from the database. The system then displays the record and gives the actor the chance to edit the details they want to change (except the Application ID and reference number), and the option to override the existing agreement copy. After the actor indicates that they are done making changes, they can save changes, and the system will give a conformation of the successful change. | |
| **Actor(s):** | Applicant (Primary)  Administrator (Primary) | |
| **Related Use Cases:** | Read Applicant | |
| **Stakeholders and Interests:** | Applicant: Wants to edit the details of their application  Administrator: Wants to link an Application to a Donor through a Donation | |
| **Pre-Conditions:** | The record must exist in the database  The Application should not already have a Donation in the case of an Applicant, as this will cause glitches in the system. Otherwise this strays from the happy path | |
| **Post-Conditions:** | The Application entry is updated in the database  The system gives confirmation that the entry has been updated | |
| **Flow of Activities:** | 1. User requests to Update Application 2. User enters the Application Number 3. User makes the necessary changes to the Application | * 1. Prompts the User to enter the Application Number   2. Invokes Read Application   3. Displays the details of the application   4. Prompts the user to make changes to the displayed details   5. Updates the Application record in the database   6. Displays a message for the user that the update has been completed |