Milestone 3

Team 5

Bohlokoa Tilo (1852529)

Chico Vicente (1844374)

Bolekwa Mboniswa (1397266)

Xolani Magoloza (1412049)

Rahube Seabi (1834316)

# Action Points

|  |  |
| --- | --- |
| **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). |
| 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. |

# 2. Purpose

This document is the finale iteration in mapping and showing our 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. This document certifies that the organisation, #fundMe, has read and reviewed the document and is satisfied with the information presented within the document at this phase of the project. If there are issues, they are to be communicated back to us (Kelello) for modification and/or clarification. As this document details most of the necessary information needed before the design and implementation phases can begin. It therefore serves as a sign-off document between the two abovementioned parties to supply each with a form of legal protection.

# Executive Summary

This Milestone, being the final iteration of the domain modelling and system design; it is a culmination of all the build-up and feedback from the previous Milestones. We drew a lot from the action points above to make improvements to our models in this milestone, and also made changes based on information in the updated case study.

Based off the alterations to the case study, certain attributes in the Class Diagram. This can be seen in how we no longer store information on the student’s financial mean’s test. We also made changes to the relationships that fail to model your environment, this can be seen with how we changed the relationship between Administrator and Donor to a one to one as there will always be an Administrator creating a Donation.

This document contains the final form of the Use Case Set. You will note that there are more use cases being invoked (for example how Create Donation now invokes Read Application, Read Donor and Update Application as they are essential to its functionality). We also specify if a Use Case extends or includes another in each respective case.

The Use Case Diagram in this document has also been updated to better represent the users in the system. This can be seen with the Update Applicant, which now includes the Administrator as an actor as opposed to the Applicant alone.

The document contains 2 Fully Dressed Use Cases to give you a less abstract perspective of how those Use Cases work. These include an in depth look at the post conditions, the actors involved, the stakeholders and their interests; and most importantly the flow of activities. The flow of activities then informed the SSD’s that we built for each use case. Showing the functionality of the flow of activities in a way that is most understandable to the UI design team, but also holds value to all stakeholders as they have a clear view of how the system works.

We also added a State Machine Diagram to model the changes the Applicant class undergoes in its lifeline. This will aid you as the client in understanding the changes we make to this class in the use cases and throughout the system.

Our team added an activity diagram for the Create Donation Use Case. This model shows the flow of data in this use case.

# Final Class Diagram for #fundMe system

A screenshot of a cell phone

Description automatically generatedThe following is the final class diagram which serves as a representation of the structure of the system’s database. It also shows the relationships among the tables of the database, the relationships of which we elicited from the given business rules. In this case, the tables are Applicant, Application, Donor, Donation and Administrator, as can be seen in the diagram below.

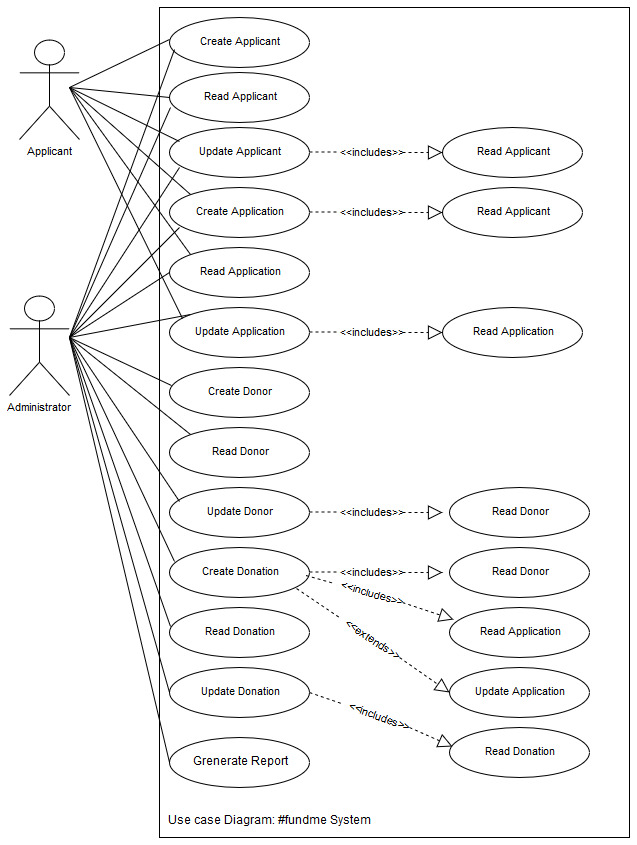
# 5. Final Use Case Set for #fundMe system

The following table is an illustration of a use case set, which essentially serves as a framework of the events that are expected to take place within the #fundMe system. Applicant, Application, Donor, Donation and Administrator are the tables within the system’s database as seen in the Class Diagram above. The Create, Read and Update are the core operations that are to trigger the attention of the system.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Use Case | Description | Used use cases |
| APPLICANT | | | |
|  | Create Applicant | Registration is required for first time applicants. |  |
| Read Applicant | Allows applicant to view applicant details. |  |
| Update Applicant | Allows applicant to update applicant details. | <<includes>>Read Applicant |
| APPLICATION | | | |
|  | Create Application | Allows user to apply for required funding. | <<includes>> Read Applicant |
| Read Application | Allows user to view application from the database. |  |
| Update Application | Allows user to update application details. | <<includes>> Read Application |
| DONOR | | | |
|  | Create Donor | Each new donor must be created by an administrator. |  |
| Read Donor | Allows admin to view a donor’s details. |  |
| Update Donor | Allows admin to update a donor’s details. | <<includes>> Read Donor |
| DONATION | | | |
|  | Create Donation | A new donation is created when a match has been made between applicant and donor. | <<extends>>Update Application  <<includes>> Read Donor  <<includes>>Read Application |
| Read Donation | Allows user to view donation details. |  |
| Update Donation | Allows user to update donation details. | <<includes>> Read Donation |
| ADMINISTATOR | | | |
|  | Create Administrator | Creates new administrator. |  |
| Read Administrator | Reads administrator’s details. |  |
| Update Administrator | Updates administrator’s details. | <<includes>> Read Administrator |
| REPORT | | | |
|  | Generate report | | |

# 6. Revised Use Case Diagram for #fundMe system

The use case diagram illustrates the interaction of the users of the system, Applicant and Administrator, with the system, as well as which operation, according to the use case set above, concerns which users. In this particular case both the Applicant and the Administrator are primary actors of the system.

The donor plays no role in the system and thus they do not appear as an actor of the system in the diagram.

# 7.1.1 Fully dressed use case description for Create Applicant

The below fully-dressed use case description provides detailed information on create applicant. It provides necessary steps on process on how the actor interacts with the system, what must be true/false for the activities to occur.

|  |  |  |
| --- | --- | --- |
| **Use case name:** | Create Applicant | |
| **Scope:** | #FundMe System | |
| **Triggering Event:** | Applicant requests to create applicant | |
| **Brief Description:** | When the student wants to create applicant in the system they will enter their demographic data which will be captured by the system into the applicant table. If students are not comfortable in their technology skills they can submit necessary documents containing their demographic data to the administrator who will then create applicant on their behalf. The system then sends login details to the applicant with instruction to provide important documentation and sets the applicant’s funding suitability status to “Red”. | |
| **Actors:** | Applicant (Primary)  Administrator (Primary) | |
| **Related Use Cases:** | N/A | |
| **Stakeholders and interests:** | Applicant: Desires to be captured in the system thus records their Data  Administrator: Desires to keep track of new applicants | |
| **Pre-conditions:** | N/A | |
| **Post Conditions:** | Applicant data recorded in the APPLICANT table  Confirmation message sent to applicant with login credentials along with instructions to provide the necessary documents | |
| **Flow of Activities** | Actor | System |
| 1. Requests to Create Applicant.   2. Enters applicant\_firstname, applicant\_lastname, applicant\_dateofbirth, applicant\_id/passport, applicant\_institution, applicant\_degreetype, applicant\_studentnumber, applicant\_emailaddress, applicant\_contactnumber and applicant\_postaladdress | 1.1 Prompts for applicant\_firstname, applicant\_lastname, applicant\_dateofbirth, applicant\_id/passport, applicant\_institution, applicant\_degreetype, applicant\_studentnumber, applicant\_emailaddress, applicant\_contactnumber and applicant\_postaladdress  2.1 Generates login credentials (applicant\_username and applicant\_password)  2.2 Stores the data into the APPLICANT table  2.3 Sets funding suitability status to “Red”.  2.3 Sends login credentials along with instructions to provide necessary documentation (certified copies of ID document/passport and proof of registration). |

# 7.2.1 System sequence diagram for Create Applicant

A screenshot of a cell phone

Description automatically generatedThis diagram maps the communication between the system and the applicant during the Create Applicant process.

# 7.1.2 Fully dressed use case description for Update Applicant

The fully dressed use case description below gives a more detailed elucidation of the Update Applicant use case/process. It sheds light upon the steps necessary to perform the process successfully and exceptions/alternatives in the case of a deviation from the main path, as well as what must be true before and after the process takes place.

|  |  |  |
| --- | --- | --- |
| **Use case name:** | Update Applicant | |
| **Scope:** | #fundMe System | |
| **Triggering Event:** | Applicant requests to update applicant | |
| **Brief Description:** | When an Applicant/Administrator wishes to make changes to applicant, they can request to update applicant. The system will enable the Applicant/Administrator to make changes to applicant. The Applicant/Administrator then makes the desired changes. The system captures the changes into the APPLICANT table. After which it displays the updated version of the demographic data. The system then sends a confirmation message to the Applicant, pertaining to the changes made.  **N.B.** The Applicant can only change their first name, last name, birth date, student number, institution, degree type, email, contact number and postal address.  Whereas the Administrator can change an applicant’s first name, last name, birth date, student number, institution, degree type, email, cell number, postal address and eligibility status. | |
| **Actor(s):** | Applicant (Primary)  Administrator (Primary) | |
| **Related use case(s):** | <<includes>> Read Applicant | |
| **Stakeholder and interests:** | Applicant: Desires to update their details efficiently and with ease.  Administrator: Desires to update data as accurately and time-efficiently as possible. Desires to always have up-to-date information about the applicant to avoid providing inaccurate information to donors and generating reports based off outdated data.  The Organization: Desires to be provided with accurate and up-to-date reports and information for decision making with regards to the impact and success of #fundMe. | |
| **Pre-conditions:** | The applicant’s record must already exists within the APPLICANT table in the #fundMe database. | |
| **Post-conditions:** | Updated applicant data are captured into the APPLICANT table.  Confirmation message detailing the changes made to applicant is sent to the Applicant. | |
| **Flow of activities:** | Actor | System |
| 1. Requests to update applicant 2. Enters applicant\_id/passportnumber 3. Makes the necessary changes to the data | * 1. Prompts for applicant\_id/passportnumber   2. Uses applicant\_id/passportnumber to invoke the Read Applicant use case   3. **A**.(*if Applicant*)Displays applicant\_firstname, applicant\_lastname, applicant\_dateofbirth, applicant\_studentnumber, applicant\_institution,applicant\_email, applicant\_contactnumber, applicant\_degreetype and applicant\_postaladdress from the APPLICANT table   **B**.(*if Administrator*) Displays the applicant\_firstname, applicant\_ lastname, applicant\_dateofbirth, applicant\_studentnumber, applicant\_degreetype, applicant\_institution, applicant\_email, applicant\_contactnumber, applicant\_postaladdress and applicant\_eligibilitystatus from the APPLICANT table   * 1. Stores the changes in the APPLICANT table   2. Displays updated applicant data   3. Sends a confirmation message to the Applicant, pertaining to the changes made. |

# 7.2.2 System sequence Diagram for Update Applicant

A screenshot of a cell phone

Description automatically generatedThe following sequence diagram represents the order in which communication takes place between the applicant and the system during the update applicant process.

# 7.1.3 Fully dressed use case diagram Create Donation

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:** | <<includes>> Read Application  <<extends>> Update Application  <<includes>> Read Donor | |
| **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 update 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 Update Application use case is invoked to update the Application status to “Funded” or “Partially Funded”  Confirmation message is sent to respective Donor and Applicant pertaining the details of the Donation | |
| **Flow of Activities:** | 1. The Administrator requests to create a Donation 2. The Administrator enters the search criteria 3. The Administrator enters the credentials of the Donation matching an Application to a Donor | * 1. The system prompts for search criteria to read the list of Donors and Applications   2. The system invokes the Read Application 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   5. The system prompts for details on the Donation   6. The system enters the record of the Donation into the DONATION table with the credentials   7. The system invokes Update Application to change the status of the Application to “Funded” or “Partially Funded”   8. The system sends a confirmation to the respective Applicant and Donor with details pertaining to the donation. |

# 7.2.3 System sequence diagram for Update Application

The following sequence diagram represents the order in which communication takes place between the administrator and the system during the create donation process. The SSD mostly serves the purpose of portraying the communication between the system and the user, and it is very useful to the UI design team.

A screenshot of a cell phone

Description automatically generated

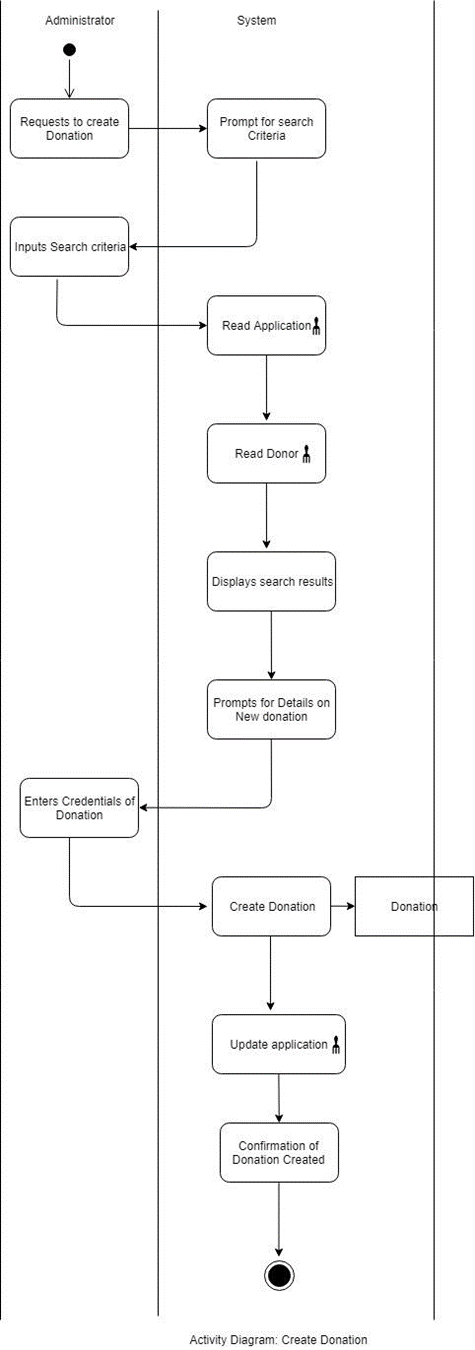
# 8.State Machine Diagram for APPLICANT class

A screenshot of a cell phone

Description automatically generated

# 9.Activity Diagram for Create Donation

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



# Peer Review and team sign-off

|  |  |  |  |
| --- | --- | --- | --- |
| **Student Number** | **Student Name** | **Contribution(%)** | **Signature** |
| 1844374 | Chico | 22.3 | A drawing of a person  Description automatically generated |
| 1852529 | Bohlokoa | 23.3 | A drawing on a necklace  Description automatically generated |
| 1397266 | Bolekwa | 22.3 |  |
| 1412049 | Xolani | 18 |  |
| 1834316 | Rahube | 14 | A picture containing sky, boat, outdoor  Description automatically generated |