Analysis

1. Requirement

The software requirements are description of features and functionalities of the target system. Requirements convey the expectations of the users from the system product. The requirement can be obvious or hidden, known or unknown, expected or unexpected from the client point of view.

Functional and Non- functional requirement

* Functional Requirement

Functional requirement specify the operation and activities that a system must be able to perform. This specification done so that the user can understand the system, which do not need any particular technical knowledge to understand the system. Functional requirement includes:

* Descriptions of data to be entered into the system
* Descriptions of operations performed by each screen
* Descriptions of work-flows performed by the system
* Descriptions of system reports or other outputs
* Who can enter the data into the system
* Non-Functional Requirement

A requirement of a software that is not specific, measurable aspect of performing a particular task. Non-functional requirements are vital to success of software systems. If non-functional requirements are not included, then they may occurred unsatisfied users, developers, clients, and budget overruns to correct the system with the help of it the system will be time that is more realistic and cost estimates, have few errors and are more comprehensive.

* MoSCow Patriotization

MoSCoW techniques is one of the easiest methods for the prioritization. MoSCoW method is the prioritization technique that is originating from the dynamic software development method (DSDM). According to this technique, you can categorize your list of requirements into the following groups:

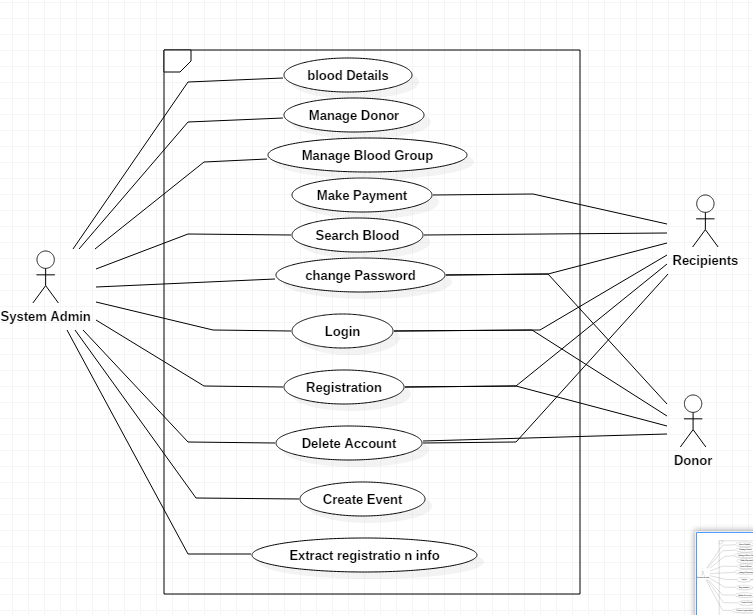
* **M – Must have.** This point describes requirements that must be satisfied in the final solution. These requirements are non-negotiable. In addition, the project will fail without them.
* **S – Should have.** A high-priority feature that is not critical to launch. However, it is considered to be important and of a high value to users. Such requirements occupy the second place in the priority list.
* **C – Could have.** A requirement that is desirable but not necessary.
* **W – Won’t have.** A requirement that would not be implemented in a current release but may be included in a future stage of development. Such requirements usually do not affect the project success.

|  |  |  |
| --- | --- | --- |
| Functional and Non-Functional | Requirement | MoSCoW |
| R1  (NF) | Login of admin | Must have |
| R2 (NF) | Login of Blood Donor | Must have |
| R3 (NF) | Change Password | Should have |
| R4 (NF) | Registration | Must have |
| R5 (F) | Change profile | Should have |
| R6 (F) | Extract Information | Could have |
| R7 (f) | Report of Blood | Should have |
| R8 (f) | Detail of new Blood | Should have |
| R9 (f) | Removing of Blood details | Could have |
| R10 (f) | Blood purchase details | Must have |
| R11 (f) | Blood sales details | Must have |
| R12 (f) | Billing of Blood sales | Must have |
| R13  (f) | Delete Account | Could have |
| R14 (f) | Create event | Must have |
| R15(f) | Delete event | Could have |
| R16 (f) | Update event | Should have |

1. Use case

A use case is a software and system engineering term that describes how a user uses a system to accomplish a particular goal. A use case acts as a software modelling technique that defines the features to be implemented and the resolution of any errors that may be encountered. Use cases define interactions between external actors and the system to attain particular goals. Three basic elements make up a use case:

* Actors: Actors are the type of users that interact with the system.
* System: Use cases capture functional requirements that specify the intended behaviour of the system.
* Goals: a user to fulfil goals describing the activities and variants involved in attaining the goal typically initiates Use cases.

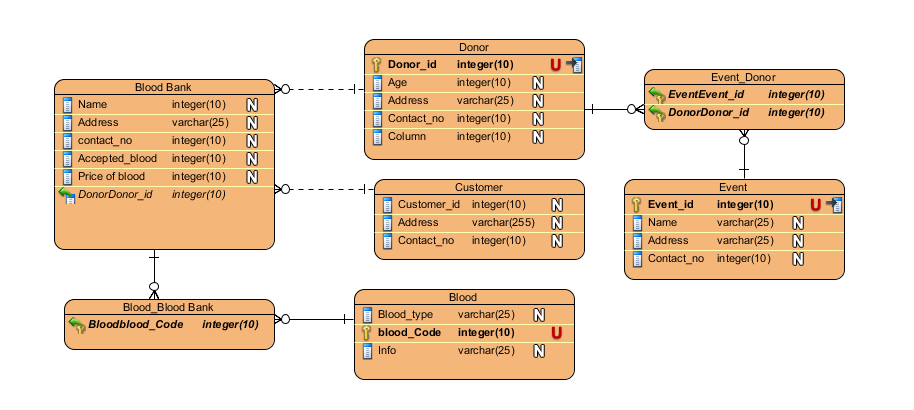


Use case diagram is used for the simplest representation of all user interaction with the system. This helps to view the different users and the different functions performing with the system. This system of viewing make any members can instantly gain an idea of the functionality of the system and its boundaries.

1. ER-Diagram

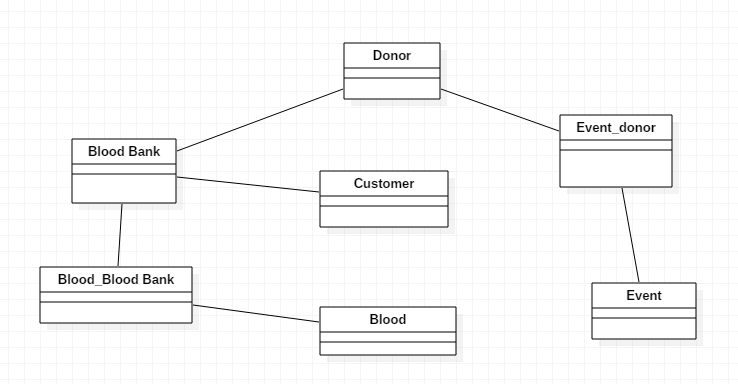
An entity relationship diagram (ERD) shows the relationships of entity sets stored in a database. An entity in this context is an object, a component of data. An entity set is a collection of similar entities. These entities can have attributes that define its properties.

When documenting a system or process, looking at the system in multiple ways increases the understanding of that system. ERD diagrams are commonly used in conjunction with a data flow diagram to display the contents of a data store. They help us to visualize how data is connected in a general way, and are particularly useful for constructing a relational database.

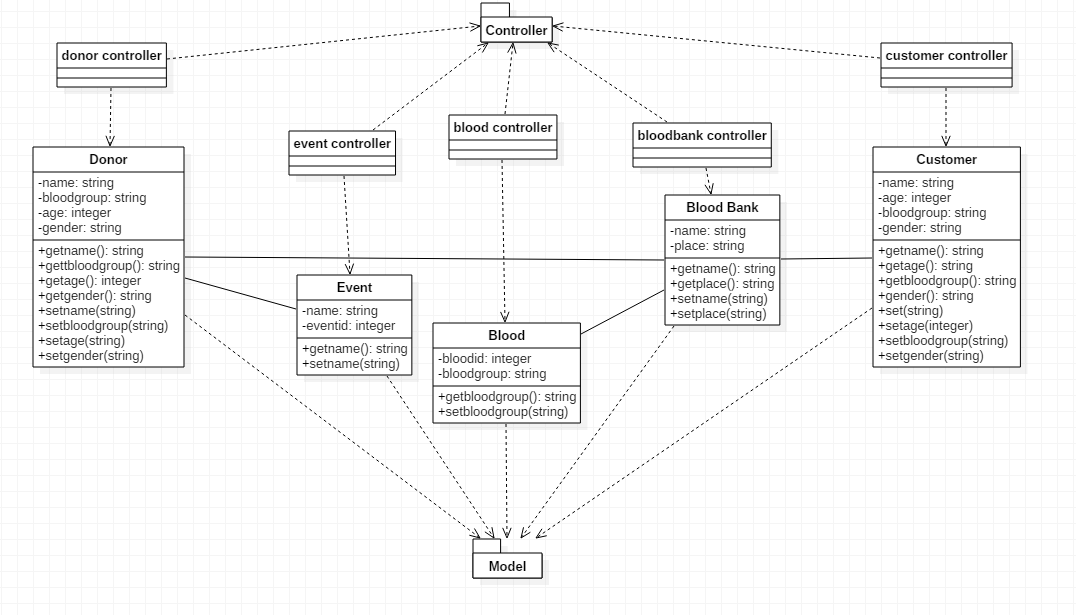


1. Initial Class Diagram

In software engineering, a class diagram in the Unified Modeling Language (UML) is **a type of static structure diagram** that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.



1. Final Class diagram



Final initial diagram Shows static structure of classifiers in a system. Diagram provides basic notation for other structure diagrams prescribed by UML. It is helpful for developers and other team members. Business Analysts can use class diagrams to model systems from business perspective

1. Activity Diagram

An activity diagram is a graphical representation of an executed set of procedural system activities and considered a state chart diagram variation. Activity diagrams describe parallel and conditional activities, use cases and system functions at a detailed level. Shapes that are connected by arrows represent an activity diagram. Arrows run from activity start to completion and represent the sequential order of performed activities. Black circles represent an initial workflow state. A circled black circle indicates an end state. Rounded rectangles represent performed actions, which are described by text inside each rectangle. An activity diagram is used to model a large activity's sequential work flow by focusing on action sequences and respective action initiating conditions. The state of an activity relates to the performance of each workflow step.

