**Analysis Specification**

**On**

**Orion Blood Bank**

Roshan Shrestha

00163453

Computing Project

Level 5 Diploma in Computing

Softwarica College of IT & E-Commerce

Kathmandu, Nepal

23rd April, 2017

Contents

[Introduction 3](#_Toc480823245)

[Use Case Diagram 4](#_Toc480823246)

[Requirements 7](#_Toc480823247)

[Prioritization 13](#_Toc480823248)

[Architecture 15](#_Toc480823249)

[ System Architecture 15](#_Toc480823250)

[ Initial Class Diagram 16](#_Toc480823251)

[Conclusion 17](#_Toc480823252)

[References 18](#_Toc480823253)

# **Introduction**

Analysis can be defined as a thorough inspection of something complex in order to value nature or to find out its important features. It can be known as the foremost pace of project or a plan. It provides decisive facts sustain to a technological panel. Project analysis provides important presentation parameter and creates usual inconsistency information for tracking progress. It is the foremost pace for creating the improvement of any task. With the help of analysis specification, we can know useful and non-useful needs for the system.

Rough draft of Blood bank website

Orion Blood banking is a web application created for storing the blood, separation of different components, providing the blood to the accident victims, people undergoing surgery and patients receiving treatment for leukemia, cancer or other diseases, testing of blood. As creating this system there occurs various benefit to the human being where they can easily find the required blood. The person who requires the blood does not have to post on social media or call different hospitals where it not only saves the time, it will also save the life of many people.

It contains the subsequent division:

1. Use case diagram
2. Requirements

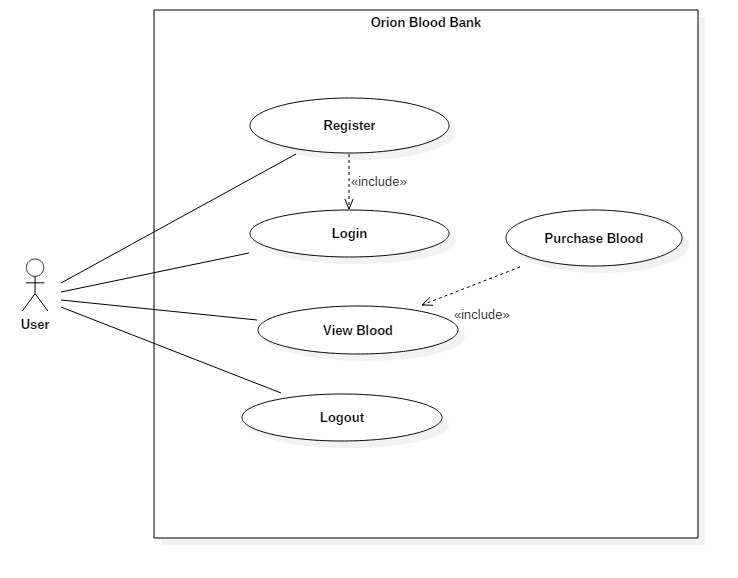
# **Use Case Diagram**

A use case is a methodology used in system analysis to identify, clarify, and organize system requirements. The use case is made up of a set of possible sequences of interactions between systems and users in a particular environment and related to a particular goal. It consists of a group of elements (for example, classes and interfaces) that can be used together in a way that will have an effect larger than the sum of the separate elements combined. The use case should contain all system activities that have significance to the users (Rouse, 2017). It is significant on the following basis:

1. Also, various functional and non-functional requirements are distinct by this diagram.
2. Dependencies and constraint in numerous use cases is acknowledged by this diagram.
3. According to their roles it identifies the actor character concerned with the system.

The use case representation of Orion Blood Banking as user perspective is presented beneath:

1. To purchase the requisite, blood
2. Registration in web-application by inputting detail of the respective person in the registration form can be made by the user.
3. The required blood group can be viewed and purchase by the user.
4. By using the registered username and password blend the user can sign in into the system for the endorsement.

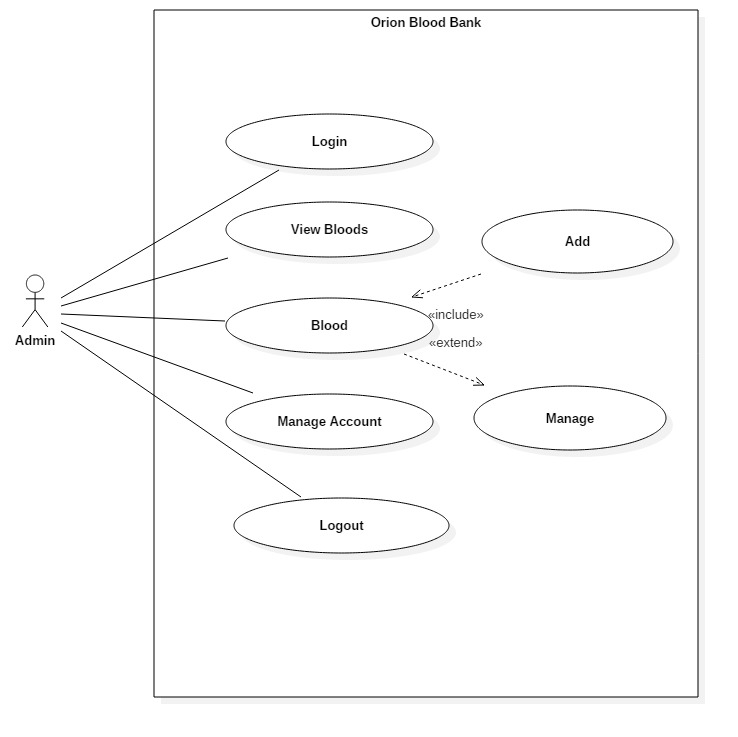


*Fig-1: Use case diagram from user perspective*

The use case description of the user is detailed beneath:

1. Register: Registration can be done by the user by providing the information required.
2. Login: The user can login into his/her account by logging in the username the password.
3. View Blood: After the user have login in he/she can view the required blood as well as purchase the blood required.
4. Logout: After carrying out all the necessary task the user can logout at any time provided by the logout button in the system.

The use case representation of Orion Blood Banking as administrator perspective is presented beneath

1. For endorsement, the admin can sign-in using the personal username and password.
2. Using the slot in form the admin can insert and manage different blood groups.
3. The log of transactions executed in the system by customer and admin, can be witness by the admin himself/herself.
4. If required the admin can witness, modernize and call off any blood group.

*Fig-2: Use case diagram from admin perspective*

The use case representation of Admin is described beneath:

1. Login: By using the personal username and password the admin can login onto the system.
2. View Bloods: If required the admin can witness, modernize and call off any blood group.
3. Blood: After viewing the blood the admin can add or manage the respective bloods in any required cases.
4. Manage Account: The admin can manage the accounts of the user in any cases if necessary.
5. Logout: The log of transactions executed in the system by customer and admin, can be witness by the admin himself/herself.

# **Requirements**

One of the essential sectors used in the system development is the Requirement assembly as it resolves the user from system anticipation. With the help of requirement assembly, the extent of the system and the recognition of the likely prospect hitch or possibility can be decisive. Hence, ingenuity in conversation and perceptive is essential.

The necessity can be collected by visiting the Orion Blood Banking. To meet their prospect through the website we have arranged a set of questionnaires intended for the Owner. Also to diagnose the uncertainty and private requirement in the system a prototype can be advanced.

The 2 types of requirements for the system are enlisted below:

* **Functional requirement**

It is the fundamental philosophy and functioning of the system which is required for the advancement of a system.

**Class 1: User**

**ID**: FR1

**Title**: Registration

**Description**: It is mandatory for both administrator and user to register for the use the website.

**Rational**: Information and privacy of the user.

**Dependencies**: NA

**ID**: FR2

**Title**: Login

**Description**: The user name and password record on the register page are required to login.

**Rational**: Authentication and confidentiality of user information

**Dependencies**: FR1

**ID:** FR3

**Title:** View Blood

**Description:** The user cam effortlessly view the blood groups and different information.

**Rational:** View blood groups

**Dependencies: FR2**

**ID:** FR4

**Title:** Logout

**Description:** After saving his task the user can log out of the system.

**Rational:** Log out of the system.

**Dependencies: FR2**

**Class 2: Admin**

**ID:** FR10

**Title:** Login

**Description:** Admin should place user name and password to manage the blood groups.

**Rational:** Confidentiality and information of the admin

**Dependencies: NA**

**ID:** FR11

**Title:** View blood

**Description:** The admin will be able to view the different kinds of blood available.

**Rational:** To view blood.

**Dependencies: FR10**

**ID:** FR12

**Title:** Blood

**Description:** The admin can add and manage the blood needed.

**Rational:** Adding and managing the blood.

**Dependencies: FR11**

**ID:** FR13

**Title:** Manage Account

**Description:** Admin will manage the account of different users respectively.

**Rational:** Managing accounts.

**Dependencies: FR10**

**ID:** FR14

**Title:** Logout

**Description:** After saving his task the user can log out of the system.

**Rational:** Website monitoring.

**Dependencies: FR10**

* **Non-functional requirement**

As it is not required system to subsist although it cater enhancement to the system.

**ID:** NFR1

**Title:** Performance.

**Description:** The system should be fast without any lag on performance.

**Rational:** Productivity maintenance.

**Dependencies:** N/A

**ID:** NFR2

**Title:** Usability

**Description:** The website should be user friendly.

**Rational:** Easily accessible.

**Dependencies:** N/A

**ID:** NFR3

**Title:** Scalability

**Description:** The system should be flexible for any kind of changes that may occur.

**Rational:** Capacity to encompass.

**Dependencies:** N/A

**ID:** NFR4

**Title:** Response time

**Description:** The system should be quick on response to the user.

**Rational:** Rapid responding.

**Dependencies:** N/A

**ID:** NFR5

**Title:** Portability

**Description:** The website should run on any mobile devices.

**Rational:** Adaptability on any system.

**Dependencies:** N/A

**ID:** NFR6

**Title:** Security

**Description:** All the data should be encrypted and should be confidential.

**Rational:** Upholds security.

**Dependencies:** N/A

**ID:** NFR7

**Title:** Reliability

**Description:** The system must be accurate and reliable.

**Rational:** Maintenance of reliability.

**Dependencies:** None

**ID:** NFR8

**Title:** Availability.

**Description:** System should be active whenever used by the user.

**Rational:** Upholding the availability.

**Dependencies:** N/A

**ID:** NFR9

Title: Capacity

Description: Multiple handling of the task should be performed by the system.

Rational: Sustain Capacity.

Dependencies: N/A

**ID:** NFR10

**Title:** System Dependability

**Description:** The system should act independently awaring the user before for any faults.

**Rational:** Independent decision.

**Dependencies:** N/A

# **Prioritization**

Requirement prioritization is used in Software product management for determining which candidate requirements of a software product should be included in a certain release. Requirements are also prioritized to minimize risk during development so that the most important or high risk requirements are implemented first. Several methods for assessing a prioritization of software requirements exist. (Wikipedia, 2017) It helps in allotment of time and funds to various needs as well as in the project organization. One of the ways of doing prioritization is by interacting with the Proprietor, faculty and also the user. The requirements should be prioritized for the subsequent reasoning:

1. Evolution of fresh system and application encountering the anticipation of the user.
2. Better perspective of the application or system is permitted.
3. Identification of irrelevant necessities and decisive requirements.
4. Helps in provision of time and assets.

Moscow technique for prioritizing the requirement is cast off for the prioritization:

* Must Have: These are the essential requirement of the system before starting the system.
* Should Have: These are features that maybe included in the system according to it benefits to the system
* Could Have: These features are not essential but may be added for attraction.
* Won’t Have: These features are not required for the system.

The following prioritization requirement of the application is enlisted below:

**User**

|  |  |  |
| --- | --- | --- |
| **ID** | **Functional Requirements** | **MoSCoW** |
| FR1 | Registration | Must Have |
| FR2 | Login | Must Have |
| FR3 | View blood | Should Have |
| FR4 | Logout | Must Have |

**Admin**

|  |  |  |
| --- | --- | --- |
| **ID** | **Functional Requirements** | **MoSCoW** |
| FR10 | Admin Login | Must Have |
| FR11 | View Blood | Should Have |
| FR12 | Blood | Must Have |
| FR13 | Manage Account | Should Have |
| FR14 | Logout | Must Have |

**Non-functional Requirement**

|  |  |  |
| --- | --- | --- |
| **ID** | **Non-functional Requirements** | **MoSCoW** |
| NFR1 | Performance | Wont Have |
| NFR2 | Usability | Wont Have |
| NFR3 | Scalability | Wont Have |
| NFR4 | Response Time | Wont Have |
| NFR5 | Portability | Wont Have |
| NRF6 | Security | Wont Have |
| NRF7 | Reliability | Wont Have |
| NRF8 | Availability | Wont Have |
| NRF9 | Capacity | Wont Have |
| NRF10 | System Dependability | Wont Have |

# **Architecture**

Architecture specify to the course and creation of mounting a system or an application. In other ways, it defines the structure effort for the operation in the system or application enhancement. It also influences the system safety, presentation, and trustworthiness.

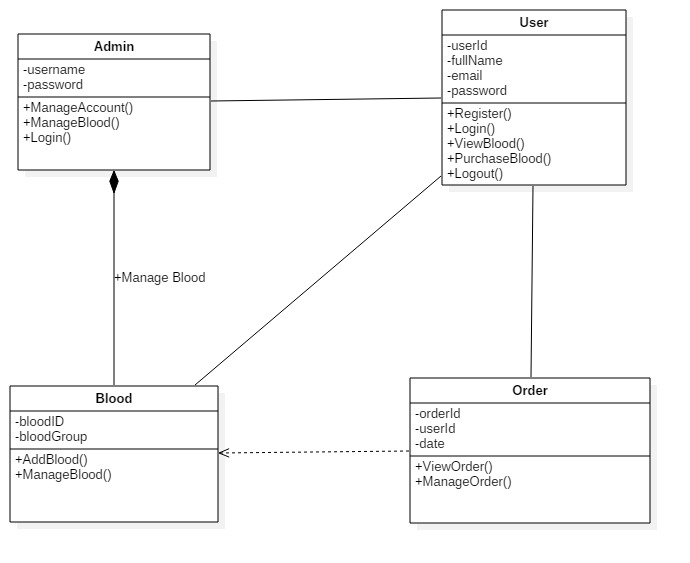
# **System Architecture**

The Model-View-Controller (MVC) is an architectural pattern that separates an application into three main logical components: the model, the view, and the controller. Each of these components are built to handle specific development aspects of an application. MVC is one of the most frequently used industry-standard web development framework to create scalable and extensible projects. (point, 2017)

For this, I ought to value Model View Controller (MCV) blueprint sample. Also desire to use PHP along with laravel Framework for expanding application as well as MySQL in the process Database.

# **Initial Class Diagram**

The class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing and documenting different aspects of a system but also for constructing executable code of the software application. (point, 2017)



*Fig-3: Class diagram*

# **Conclusion**

Hence, we can figure to build up the system amidst the assist Analysis Specification. We further scheduled the user case among user and system as well as obtained a synopsis of the system along with the yield expected against the system. In distinction to functional and non-functional requirement we collected user requirement for the system and preferred appropriate architecture for the system and generated class diagram that is the prototype of the system from use cases.

# References

point, T., 2017. *MVC Frameawork: Introduction.* [Online]   
Available at: https://www.tutorialspoint.com/mvc\_framework/mvc\_framework\_introduction.htm  
[Accessed 12 April 2017].

point, T., 2017. *UML - Class Diagram.* [Online]   
Available at: https://www.tutorialspoint.com/uml/uml\_class\_diagram.htm  
[Accessed 22 april 2017].

Rouse, M., 2017. *use case.* [Online]   
Available at: http://searchsoftwarequality.techtarget.com/definition/use-case  
[Accessed 20 april 2017].

Wikipedia, 2017. *Prioritization.* [Online]   
Available at: https://en.wikipedia.org/wiki/Requirement\_prioritization  
[Accessed 20 april 2017].