Blood Bank

Supplementary Specification

Version 1.1

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 21/03/2018 | 1.0 | Supplementary specifications for the first version of the system | Dănilă Vlad-Mihai |
| 14/05/2018 | 1.1 | Updated design constraints | Dănilă Vlad-Mihai |

Table of Contents

1. Introduction 4

2. Non-functional Requirements 4

2.1 Availability 4

2.2 Performance 4

2.3 Security 4

2.4 Usability 4

3. Design Constraints 4

Supplementary Specification

# Introduction

The document aims to present the requirements of the Blood Bank system. This Supplementary Specification lists the requirements that are not contained in the use cases of the use-case model. The Supplementary Specifications together with the use-case model compose a complete set of requirements on the system.

The Blood Bank System will enable users to register and share medical data and medical requirements. BBS allows donors to be registered in a database according to their blood group and facilitate fast interaction between a patient needing blood transfusion and a donor.

This specification defines the non-functional requirements of the system, such as usability, availability, performance and security.

# Non-functional Requirements

## Availability

The BBS will maintain a constant reliability by being available 24 hours a day, 7 days a week, due to the importance of data and the damages incorrect or incomplete data can do. There shall be a significantly low percentage of downtime due to eventually addition new features, not exceeding 1% of the time.

## Performance

The system shall have no more than 3 seconds delay while directly accessing the database and no more than 1 second delay when an action response is expected.

## Security

The system shall impose users to be logged in order to perform the available operation and access its features.

## Usability

The user interface shall be design for ease-of-use and appropriate for users with no additional training on the system. The user interface shall be Windows compliant.

# Design Constraints

* The system must be implemented in a Java IDE
* Layers and another architectural style should be used
* Minimum two Design Patterns must pe present
* 2-3 One-to-Many / Many-to-Many entity relationships
* Usage of frameworks and libraries
* Input validation
* Testing using unit tests