Usage Scenarios

Usage scenario for Login as a Donor:

|  |  |
| --- | --- |
| **Use case name** | Login |
| **Paticipating actor** | Donor (User of the application) |
| **Flow of events** | 1. The Donor enters the username and password in the required fields of the Login form 2. The Donor presses the “Login” button 3. The Donor is logged in to his/her accounts |
| **Entry condition** | The Donor has a valid account and access to the application (and internet connection) |
| **Exit condition** | The Donor is successfully logged into the application and can access his/her account |
| **Quality requirements** | If the Login fails or the data entered by the Donor is invalid, the application will notify him/her about it |

Usage scenario for Sign Up as a Donor:

|  |  |
| --- | --- |
| **Use case name** | Sign Up |
| **Paticipating actor** | Donor (User of the application) |
| **Flow of events** | 1. The Donor enters data in the required fields of the Sign Up form (including its email and password) 2. The Donor presses the “Sign up” button 3. The data provided by the Donor is processed and validated in the application |
| **Entry condition** | The Donor has access to the application and can use it (e.g. has internet connection) |
| **Exit condition** | The Donor can log in to his/her account |
| **Quality requirements** | If the Register fails or the data entered by the Donor is invalid, the application will notify him/her about it |

Usage scenario for Blood Donation as a Donor:

|  |  |
| --- | --- |
| **Use case name** | Apply for donation |
| **Participating actor** | Donor (User of the application) |
| **Flow of events** | 1. The Donor enters data in the required fields of the Donation form (e.g. data regarding his health condition) 2. The Donor selects a Donation Center 3. The Donor presses the “Send” button 4. The data provided by the Donor is processed and stored in a repository in order to be updated |
| **Entry condition** | The Donor has access to the application and is already logged into his account |
| **Exit condition** | The Donor receives a notification from the application showing that operation has been successful |
| **Quality requirements** | If the operation of sending the donation form fails, the application will notify him/her about it |

Usage Scenario for Doctor: Request blood

|  |  |
| --- | --- |
| **Use case name** | Request blood |
| **Participating actor** | Doctor |
| **Flow of events** | 1.The doctor enters the details of the request(e.g. no. red cells, plasma, priority etc.)  2.Presses the “Send” button  3.The data provided in the request is stored in a repository in order to be later processed |
| **Entry condition** | The doctor is logged into his account |
| **Exit condition** | The doctor receives a notification from the application showing that operation has been successful |
| **Quality requirements** | If the operation of sending the request fails, the application will notify him/her about it |

Usage Scenario for Donation Center Personnel: Update profile

|  |  |
| --- | --- |
| **Use case name** | Update Profile |
| **Participating actor** | Donation Center(personnel) |
| **Flow of events** | 1.The donation center personnel presses the “Update Profile” button  2.Edits his/her details  3.Presses the “Update” button  4.The data of the personnel is changed according to the update request |
| **Entry condition** | The dc personnel is logged into his account |
| **Exit condition** | The dc personnel is alerted about the success of the operation |
| **Quality requirements** | In case of unsuccessful update(ex. invalid data) the personnel is notified about the issue |

Testing scenarios

|  |  |
| --- | --- |
| Test-case identifier | AddModify\_OnAdd\_ReturnsOne |
| Test location | Vivus.Core.Administration.UnitTests/AdministratosViewModelTests |
| Feature to be tested | Add an admin |
| Feature pass/fail criteria | Test passes if the admin is added successfully |
| Means of control | The AddModifyAsync() method is called via a viewModel |
| Data | The attributes of the viewModel are filled with mock data  After the method is called the number of admins is 1 |
| Test procedure | The test is started by selecting Test -> Run. The test will run without further intervention until completion. The test should not take more than 8 seconds. |
| Special requirements |  |

|  |  |
| --- | --- |
| Test-case identifier | ApproveOrRejectDonation\_OnNoSelection\_ReturnsAll |
| Test location | Vivus.Core.DCPersonnel.UnitTests/BloodDonationRequestsViewModelTests |
| Feature to be tested | Approve or reject a donation |
| Feature pass/fail criteria | Test passes if all the approved or rejected donations are returned |
| Means of control | The ApproveOrRejectDonation() method is called via a viewModel |
| Data | The list of donation requests is filled with mock data  After the method is called the donation request is approved/rejected and the number of donations is checked |
| Test procedure | The test is started by selecting Test -> Run. The test will run without further intervention until completion. The test should not take more than 8 seconds. |
| Special requirements |  |

|  |  |
| --- | --- |
| Test-case identifier | Add\_OnApply\_ApplyForDonationFormScenario1 |
| Test location | Vivus.Core.Donor.UnitTests/ApplyViewModelTests |
| Feature to be tested | Apply for donation |
| Feature pass/fail criteria | Test passes if all there are no errors and the number of donation forms has increased |
| Means of control | The ApplyAsync() method is called via a viewModel |
| Data | The attributes of the viewModel are filled with mock data  After the method is called the number of donation forms is checked |
| Test procedure | The test is started by selecting Test -> Run. The test will run without further intervention until completion. The test should not take more than 8 seconds. |
| Special requirements |  |

|  |  |
| --- | --- |
| Test-case identifier | AddModify\_OnAddEmpty\_ReturnsZero |
| Test location | Vivus.Core.Administration.UnitTests/DoctorsViewModelTests |
| Feature to be tested | Add a doctor with no attributes |
| Feature pass/fail criteria | Test passes if the doctor is not added |
| Means of control | The AddModifyAsync() method is called via a viewModel |
| Data | The attributes of the viewModel are not filled  After the method is called the number of doctors is the same |
| Test procedure | The test is started by selecting Test -> Run. The test will run without further intervention until completion. The test should not take more than 8 seconds. |
| Special requirements |  |