Bug Tracking

|  |  |  |  |
| --- | --- | --- | --- |
| ID and name | 1a. Login | | |
| Primary actor | Tester, Programmer | Secondary actors | - |
| Description | The client logs in to his account | | |
| Trigger | The actor wants to use the application | | |
| Preconditions | None | | |
| Postconditions | Credentials are checked and the user is logged in the account, | | |
| Normal flow | 1. The actor fills in the username and password 2. The actor clicks on log in 3. The soft opens the main window if the credentials are correct | | |
| Alternative flows | None | | |
| Exceptions | 3a.The soft shows the error for invalid credentials and asks the actor to check the filled in credentials | | |

|  |  |  |  |
| --- | --- | --- | --- |
| ID and name | 1b. Bug list visualisation | | |
| Primary actor | Tester, Programmer | Secondary actors | - |
| Description | The client logs in to his account | | |
| Trigger | The actor wants to see the list of bugs | | |
| Preconditions | Logged in | | |
| Postconditions | None | | |
| Normal flow | 1. When the soft opens, the list of bugs is presented. | | |
| Alternative flows | 1. If the account belongs to a programmer, there is a second tab with “Currently working on” bugs. | | |
| Exceptions | None | | |

|  |  |  |  |
| --- | --- | --- | --- |
| ID and name | 2 Register new bug | | |
| Primary actor | Tester | Secondary actors | - |
| Description | The tester finds a bug and can register it at any computer with the application | | |
| Trigger | The tester finds a bug and wants to register it | | |
| Preconditions | Logged in (the account must belong to a Tester) | | |
| Postconditions | The Bug is saved and marked as “Unsolved” and added to the overall list of bugs | | |
| Normal flow | 1. The tester clicks on the tab “Register new bug” 2. The soft presents to form to fill for the new bug 3. The tester fills the form with the name and the description of the bug 4. The tester clicks on the button “Register” 5. The soft shows the tester a pop up with the confirmation of the added bug | | |
| Alternative flows | 5a. The tester declines the confirmation | | |
| Exceptions | None | | |

|  |  |  |  |
| --- | --- | --- | --- |
| ID and name | 3. Register to a bug | | |
| Primary actor | Programmer | Secondary actors | - |
| Description | The programmer can search and select a bug from the list and clicks on the register button. | | |
| Trigger | The programmer wants to work on a new bug. | | |
| Preconditions | Logged in as a programmer, visualisation of the list | | |
| Postconditions | The Bug is saved and updated with the programmer’s name at the ones working on it | | |
| Normal flow | 1. The programmer searches for a bug 2. The programmer clicks on the wanted bug 3. The programmer clicks on the button “Register” 4. A confirmation pop-up appears. 5. The programmer confirms the register. | | |
| Alternative flows | 5a. The programmer declines the register | | |
| Exceptions | 4a. The programmer is already registered to the bug | | |

|  |  |  |  |
| --- | --- | --- | --- |
| ID and name | 4. Solve bug | | |
| Primary actor | Programmer | Secondary actors | - |
| Description | The programmer marks a bug as “WaitingValidation” | | |
| Trigger | The programmer finished solving a bug and wants to mark it. | | |
| Preconditions | Logged in (the account must belong to a Programmer), bug list visualisation, the programmer must be registered to the said bug | | |
| Postconditions | The Bug is marked as “WaitingValidation” | | |
| Normal flow | 1. The programmer clicks on the “Currently working on” tab. 2. The programmer clicks on the wanted bug 3. The programmer clicks on the button “Mark as solved” 4. A confirmation pop-up appears. 5. The programmer confirms the pop-up. | | |
| Alternative flows | 5a. The programmer declines the register | | |
| Exceptions | 4a.The programmer tries to mark a bug as “WaitingValidation”  A “SOLVED” or another “WaitingValidation” bug | | |

|  |  |  |  |
| --- | --- | --- | --- |
| ID and name | 5. Validate bug | | |
| Primary actor | Tester | Secondary actors | - |
| Description | The tester verified a bug marked as WaitingValidation and marks it as “SOLVED” or “unsolved” | | |
| Trigger | The tester verified a bug and wants to mark the bug as “SOVLED” or “Unsolved” | | |
| Preconditions | Logged in (the account must belong to a Tester), bug list visualisation | | |
| Postconditions | The Bug is marked as “Unsolved” or “SOLVED” | | |
| Normal flow | 1. The tester searches for the wanted bug. 2. The tester selects the bug 3. The tester marks the bug with “SOLVED” or “Unsolved” by pressing the buttons Solved or Unsolved 4. A confirmation pop-up shows up. 5. The tester confirms his choice. | | |
| Alternative flows | None | | |
| Exceptions | 4a.The tester tries to mark a “SOLVED” bug as “Unsolved”  4b.The tester tries to mark a “SOLVED” bug as “SOLVED”  4c.The tester tries to mark an “Unsolved” bug as “SOLVED”  4d.The tester tries to mark an “Unsolved” bug as “Unsolved”  In all cases, a pop-up error will be shown. | | |

Descriptions of template fields:

* **ID and name:** Title should be descriptive and should usually begin with a verb, e.g. order, calculate, input, etc. ID can have any format but must be unique among all use cases.
* **Primary actor:** Person that wishes to accomplish a goal through the use of the system. Only a single primary actor per use case.
* **Secondary actors:** Actors that have an interest in the completion of the goal but that do not directly interact with the system.
* **Description:** Concise description of the purpose of the use case.
* **Trigger:** Condition internal or external to the system that prompts the use case to start.
* **Preconditions:** Conditions that must be true before the use case starts. Each should be labeled with an ID unique to the use case.
* **Postconditions:** Conditions that must be true after the use case ends normally. Each should be labeled with an ID unique to the use case.
* **Normal flow:** Detailed step-by-step description of the logical flow of the use case. It should describe an explicit two way interaction, with the system prompting for input and the actor responding accordingly. Each step should be numbered.
* **Alternative flows:** Flows that achieve the same goal as the normal flow but are expected to be less common or lower priority.
* **Exceptions:** Conditions that result in the normal flow ending prematurely due to an unrecoverable condition in the system. The condition that causes the flow should be clearly stated, as should be any other decisions that the actor must make in this situation.

# Iterations:

1. ID 1 - Autetificare, ID 2 - Inregistrare bug
2. ID 3 - Alegere si inregistrare bug gasit, ID 4 - Rezolvare bug
3. ID 4 - Eliminare bug