## **Use case template**

An adaptation of the standard Cockburn template will be used. The template and examples follow:

|  |  |  |  |
| --- | --- | --- | --- |
| ID and name | UC-1: Login | | |
| Primary actor | User (Tester or Programmer) | Secondary actors | System |
| Description | The User logs into the application by providing their credentials and selecting their role (Tester or Programmer). | | |
| Trigger | User indicates the intention to log into the system. | | |
| Preconditions | PRE-1. The application is accessible.  PRE-2. User has a registered account. | | |
| Postconditions | POST-1. User is successfully logged into the system. | | |
| Normal flow | 1. **Provide Credentials** 2. User accesses the login interface. 3. User provides their username/e-mail and password and selects their role (e.g. Tester or Programmer). 4. System verifies the provided credentials against the stored user data and grants access to the user’s account. (see 1.0.E1) | | |
| Alternative flows | - | | |
| Exceptions | **1.0.E1 Invalid Credentials**  1. System prompts the user to re-enter their credentials and their role.  2a. If User cancels or refuses to provide the required information, the use case is terminated.  2b. Else if User provides all the required information, the use case is restarted. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| ID and name | UC-2: Record Bug | | |
| Primary actor | Tester | Secondary actors | Programmers |
| Description | The Tester accesses the system and records a bug by providing a name and description. Upon recording the bug, all Programmers receive an updated list of bugs including the newly added item. | | |
| Trigger | The Tester indicates the need to record a bug. | | |
| Preconditions | PRE-1. The Tester is logged into the system. | | |
| Postconditions | POST-1. Bug is successfully recorded in the system.  POST-2. The bug list is updated on every Programmers’ window. | | |
| Normal flow | 1. **Record a Single Bug** 2. Tester accesses bug recording functionality. 3. Tester provides a name and description for the bug. (see 1.0.E1, 1.0.E2, 1.0.E3) 4. Tester indicates that the bug recording is complete. (see 1.1) 5. System updates and displays the list of bugs with the newly recorded bug. | | |
| Alternative flows | * 1. **Record multiple bugs**  1. Tester asks to record another bug. 2. Return to step 1 of normal flow. | | |
| Exceptions | **1.0.E1 Incomplete Bug Information**   1. System prompts the Tester to complete all required fields before proceeding.   2a. If Tester cancels or refuses to provide the required information, the bug recording process is terminated.  2b. Else if Tester provides all the required information, the use case is restarted.  **1.0.E2 Duplicate Bug**  1. System notifies the Tester that a bug with identical name and description already exists in the list.  2a. If Tester is sure that the bugs are separate instances, they may choose to proceed with the recording.  2b. Else if Tester cancels the bug recording process, the use case is terminated.  **1.0.E3 Invalid Characters in Provided Information**  1. System prompts the Tester to correct the input before proceeding.  2a. If Tester updates the input, the use case is restarted.  2b. Else if Tester cancels or refuses to update the input, the use case is terminated. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| ID and name | UC-3: Update Bug | | |
| Primary actor | User (Tester or Programmer) | Secondary actors | System |
| Description | The User accesses the system and updates a bug by changing its information. Upon updating the bug, all Programmers receive an updated list of bugs including the newly updated item. | | |
| Trigger | The User indicates the need to update a bug. | | |
| Preconditions | PRE-1. The User is logged into the system. | | |
| Postconditions | POST-1. Bug is successfully updated in the system.  POST-2. The bug list is updated on every Programmers’ window. | | |
| Normal flow | 1. **Update a Single Bug** 2. User accesses bug updating functionality. 3. User provides new information for the bug. (see 1.0.E1, 1.0.E2, 1.0.E3) 4. User indicates that the bug updating is complete. (see 1.1) 5. System updates and displays the list of bugs with the newly updated bug. | | |
| Alternative flows | * 1. **Update multiple bugs**  1. User asks to update another bug. 2. Return to step 1 of normal flow. | | |
| Exceptions | **1.0.E1 Incomplete Bug Information**   1. System prompts the User to complete all required fields before proceeding.   2a. If User cancels or refuses to provide the required information, the bug updating process is terminated.  2b. Else if User provides all the required information, the use case is restarted.  **1.0.E2 Duplicate Bug**  1. System notifies the User that a bug with identical name and description already exists in the list.  2a. If User is sure that the bugs are separate instances, they may choose to proceed with the updating.  2b. Else if User cancels the bug updating process, the use case is terminated.  **1.0.E3 Invalid Characters in Provided Information**  1. System prompts the User to correct the input before proceeding.  2a. If User updates the input, the use case is restarted.  2b. Else if User cancels or refuses to update the input, the use case is terminated. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| ID and name | UC-4: Eliminate Bug | | |
| Primary actor | Programmer | Secondary actors | Tester |
| Description | The Tester accesses the system to view the list of bugs. Additionally, Programmer can select a specific bug from the list and declare it as eliminated. Upon declaring the bug as eliminated, it is removed from the list for all Programmers. | | |
| Trigger | The Programmer indicates the need to view and possibly eliminate a bug. | | |
| Preconditions | PRE-1. The Tester is logged into the system.  PRE-2. List of bugs is available for viewing. | | |
| Postconditions | POST-1. If Programmer declares a bug as eliminated, it is removed from the list for all Programmers. | | |
| Normal flow | * 1. **View List of Bugs**   2. Programmer asks to view the list of bugs.   3. System displays the list of bugs along with relevant details such as bug name and description.   4. **Select Bug**   1. Programmer selects a specific bug from the list.  2. System shows the selected bug.  **3.0 Declare bug as Eliminated**  1. Programmer triggers a designated button to declare the selected bug as eliminated.  2. System prompts for confirmation. (see 3.0.E1)  3. Programmer confirms the elimination of the bug. (see 3.1)  4. System removes the bug from the list for all Programmers. | | |
| Alternative flows | **3.1 Cancel Elimination**  1. Programmer decides not to eliminate the bug.  2. System returns to the bug list without removing the bug. | | |
| Exceptions | **3.0.E1 Confirmation Timeout**  1. System times out the confirmation process and returns to the bug list (step 1.0.2) view without removing the bug. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| ID and name | UC-5: Logout | | |
| Primary actor | User (Tester or Programmer) | Secondary actors | System |
| Description | User logs out of the system, terminating their current session and ending access to the application’s features. | | |
| Trigger | User indicates the intention to log out of the system. | | |
| Preconditions | PRE-1. User is logged into the system. | | |
| Postconditions | POST-1. User’s session is terminated.  POST-2. User is logged out of the system. | | |
| Normal flow | **1.0 Logout**   1. User triggeres the logout action. 2. System prompts the user to confirm their decision to log out. 3. User confirms the logout action. (see 1.1) 4. System terminates the user’s current session. | | |
| Alternative flows | * 1. **Cancel Logout**  1. User decides not to log out of the system. 2. System terminates the use case. | | |
| Exceptions | **-** | | |