## **Bug Detection System**

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

|  |  |  |  |
| --- | --- | --- | --- |
| ID and name | UC-1: Add a Bug | | |
| Primary actor | Test Engineer | Secondary actors | Bug Tracking System |
| Description | A Test Engineer accesses the BTS from corporate intranet, views the menu, and fills a form with a name for the bug and a description and submits it, and after that the list of bugs is updated with the one introduced and can be seen by everyone. | | |
| Trigger | A Test engineer discovers a new bug. | | |
| Preconditions | PRE-1. The user is logged as Test Engineer  PRE-2. The bug that he wants to add is not in the list already. | | |
| Postconditions | POST-1. The bug is stored in BTS(Bug Tracking System) with a status of “Accepted”.  POST-2. The bug list is updated and the new bug can be seen by everyone. | | |
| Normal flow | 1. **Tester identifies an issue** and decides to log a bug.  2. **Tester opens the Bug Tracking System (BTS)** and selects "New Bug."  3. **Tester enters bug details** (title, description, steps to reproduce, expected vs. actual results).  4. **Tester attaches supporting files** (screenshots, logs, or recordings).  5. **Tester sets bug attributes** (severity, priority, affected components, environment).  6. **Tester submits the bug report.**  7. **BTS updates the bug list**, making the new bug visible to all developers. | | |
| Alternative flows | None | | |
| Exceptions | The bug with the same name already exists so the tester gets an error. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| ID and name | UC-2 Resolve a Bug | | |
| Primary actor | Software Engineer | Secondary actors | Bug Tracking System |
| Description | A Programmer accesses the BTS from the corporate intranet, views the list of reported bugs, selects a bug to work on, updates its status when fixing, and finally marks it as resolved. The bug is then removed from the active bug list visible to other programmers. | | |
| Trigger | |  | | --- | |  |   A Software Engineer sees a new bug in system that needs to be resolved. | | |
| Preconditions | PRE-1. The user is logged in as Programmer.  PRE-2. The bug exists in the system with an open status. | | |
| Postconditions | POST-1. The bug is updated with a new status .  POST-2. The bug is removed from the active bug list for all programmers. | | |
| Normal flow | 1. **Software Engineer accesses the Bug Tracking System (BTS)** and views the list of reported bugs.  2.**Software Engineer selects a bug** from the list to work on.  3.**Software Engineer analyzes the bug details** (description, reproduction steps, logs).  4.**Software Engineer starts working on the fix** and updates the bug status to "In Progress."  5.**Software Engineer resolves the bug** by implementing and testing the fix.  6.**Software Engineer updates the bug status to "Resolved."**  7.**BTS removes the bug from the active bug list,** making it no longer visible to other programmers. | | |
| Alternative flows | None | | |
| Exceptions | None | | |

|  |  |  |  |
| --- | --- | --- | --- |
| ID and name | UC-3: Assign a bug to a specific Developer | | |
| Primary actor | Test Engineer | Secondary actors | Developer, Bug Tracking System |
| Description | A Test Engineer accesses the BTS from corporate intranet, views the menu, and fills a form with a name for the bug and a description and submits it for a specific DEVELOPER, and after that the list of bugs is updated with the one introduced and can be seen only by that one DEVELOPER. | | |
| Trigger | A Test engineer discovers a new bug and wants a certain DEVELOPER to solve it. | | |
| Preconditions | PRE-1. The user is logged as Test Engineer  PRE-2. The bug that he wants to add is not in the list already.  PRE-3. The Dev that is implied exists and have an account. | | |
| Postconditions | POST-1. The bug is stored in BTS(Bug Tracking System) with a status of “Accepted”.  POST-2. The bug list is updated and the new bug can be seen by the assigned DEVELOPER only and gets a notification. | | |
| Normal flow | 1. **Tester identifies an issue** and decides to log a bug.  2. **Tester opens the Bug Tracking System (BTS)** and selects "New Bug."  3. **Tester enters bug details** (title, description, steps to reproduce, expected vs. actual results).  4. **Tester attaches supporting files** (screenshots, logs, or recordings).  5. **Tester sets bug attributes** (severity, priority, affected components, environment).  6. **Tester assign to a single developer the bug**.  6. **Tester submits the bug report.**  7. **BTS sends a notification to the Developer** | | |
| Alternative flows | None | | |
| Exceptions | **The bug with the same name already exists so the tester gets an error.**  **The developer doesn t exist.**  **The developer doesn t have an account.**  **The tester doesn t have the right to assign bugs to that specific developer**. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| ID and name | UC-4: Software Engineer refuses a bug. | | |
| Primary actor | Software Engineer | Secondary actors | Bug Tracking System |
| Description | A Software Engineer accesses the BTS from corporate intranet, views the menu, and sees a bug that is assigned to him/her. He is busy with other bugs and refuses the bug. | | |
| Trigger | A Teste Engineer assigns to a specific Software Engineer a bug. | | |
| Preconditions | PRE-1. The user is logged as Software Engineer  PRE-2. The bug is specifically assigned to him. | | |
| Postconditions | POST-1. The bug is stored in BTS(Bug Tracking System) with a status of “Rejected”.  POST-2. The bug list is updated and the new bug can be seen by everyone. | | |
| Normal flow | 1. **Software Engineer accesses the Bug Tracking System (BTS)** and views the list of reported bugs and receives a notification from a certain bug assigned to him.  2. **Software Engineer selects the bug** and set his status to rejected.  3. **BTS removes the bug from the developer,** making it visible to other programmers. | | |
| Alternative flows | None | | |
| Exceptions | The System fails to process the refuse.   1. If, due to a technical error (e.g., lost connection, inaccessible database), the BTS cannot save the changes, nothing will happen. 2. Solution:   BTS displays an error message and allows the Developer to retry: *"Failed to refuse bug. Please try again."* | | |
|  |  | | |

|  |  |  |  |
| --- | --- | --- | --- |
| ID and name | UC-5: Admin Assigns role to a new created account. | | |
| Primary actor | Admin | Secondary actors | Bug Tracking System |
| Description | Admin accesses the BTS and see at the request table new account that are waiting to get a role and access to BTS. | | |
| Trigger | A new person who is either Software Engineer either Test Engineer creates a new account. | | |
| Preconditions | PRE-1. The user is logged as Admin.  PRE-2. The list of new users pending is not empty. | | |
| Postconditions | POST-1. The new account gets permission of Software Engineer/ Test Engineer. | | |
| Normal flow | 1. **Admin** accesses the Bug Tracking System (BTS) and navigates to the request table for new accounts.  2. **Admin** selects a new user from the list of pending accounts.  3. **Admin** assigns a role to the new user (Software Engineer / Test Engineer).  4. **BTS** updates the user’s permissions and grants access based on the assigned role.  5. **The new user** receives a notification that their access has been approved and they can now use BTS. | | |
| Alternative flows | **The Admin rejects the request**   1. Instead of assigning a role, the Admin decides to reject the request (e.g., due to incorrect details or unauthorized access). 2. **BTS removes the pending account request** and notifies the user of the rejection. | | |
| Exceptions | **The system fails to update the user's role**  1. If, due to a technical error (e.g., lost connection, inaccessible database), the BTS cannot save the changes, the role will not be assigned.  **2. Solution:** BTS displays an error message and allows the Admin to retry: *"Failed to assign role. Please try again later."* | | |