## **Use cases for Bug Tracking Application**

|  |  |  |  |
| --- | --- | --- | --- |
| ID and name | UC - 1 : Log - in | | |
| Primary actor | Tester, Developer ,Admin | Secondary actors | System |
| Description | Any employee with access to the application can log in, using their personal credentials, provided by the software company and specifically assigned to each individual. | | |
| Trigger | An employee opens the log-in window. | | |
| Preconditions | PRE 1 – The person who logs in works at the company the software is made for  PRE 2 – The person who logs in has credentials given to them by the said company  PRE 3 – The account has to be active | | |
| Postconditions | POST 1 – The user successfully logs into the main board of the application  POST 2 – Depending on their position, the next pop up window can be for Tester, Developer or Admin | | |
| Normal flow | **1.0 Logging into the application:**   1. The user inputs his credentials into the text fields. 2. The user clicks the log in button. 3. The user is redirected to a new pop-up window that will show the functionalities according to their position in the company. | | |
| Alternative flows | * 1. Exit by closing the application:   1.a. The user can close the window by clicking the X button on the top right corner  1.b. The user can also close the window by pressing ALT + F4 on their own keyboard  2. There will not be any progress saved(neither of the input fields) | | |
| Exceptions | 1. E1: User inputs wrong credentials:   1. The user writes in the text fields wrong credentials.  2. The user clicks the log in button.  3. The application will pop un an error message box stating that the credentials were wrong.   1. E2 : User leaves blank text fields:   1. The user decides to leave one or both text fields empty.  2. The user clicks the log in button.  3. The application will pop un an error message box stating that the credentials were left empty. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| ID and name | UC - 2 : Add a new Bug | | |
| Primary actor | Tester | Secondary actors | Testers, Developers, Admins |
| Description | A tester who found a new bug will upload it, alongside its new given name and a description | | |
| Trigger | A tester founds a new bug and posts it. | | |
| Preconditions | PRE 1 – The tester has to be logged in the application  PRE 2 – The bug has to have a new name and description | | |
| Postconditions | POST 1 – The bug will be added to the list(the solvedBy attribute will be NULL)  POST 2 – The bug list will be refreshed | | |
| Normal flow | 1. **Adding a new bug into the application:**   1. The user inputs the necessary information about the new bug into the name text field and, respectively, into the description text field.  2. The user pushed the Add New Bug button.  3. The active bug list is refreshed and the Developers and Admins will see the newly added bug | | |
| Alternative flows | * 1. Exit by closing the application:   1.a. The user can close the window by clicking the X button on the top right corner.  1.b. The user can also close the window by pressing ALT + F4 on their own keyboard.  2. There will not be any progress saved, unless the Add New Bug button has been pressed. | | |
| Exceptions | 1. E1: The tester leaves empty text fields:   1. The user chooses not to put any text in the respective fields.  2. The user clicks the Add New Bug button.  3. The application will pop un an error message box stating that the text fields were left blank. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| ID and name | UC - 3 : Solve a pre-existing Bug | | |
| Primary actor | Developer | Secondary actors | Developers, Testers, Admins |
| Description | A developer who decided that they fixed a bug can update the status of the said bug to “solved” and delete it from the active bugs list | | |
| Trigger | A developer found a solution to a bug. | | |
| Preconditions | PRE 1 – The developer has to be logged in the application  PRE 2 – The bug has to be present in the active bugs list(its status should be “active”) | | |
| Postconditions | POST 1 – The status of the bug will be updated to “solved”  POST 2 – The bug will disappear from the active bugs list | | |
| Normal flow | 1. **Solving a pre-existing bug from the application:** 2. The developer selects the bug he wants to solve. 3. The description and the name of the bug will appear in the right side of the screen. 4. The developer presses the Solve Bug button. 5. The active bugs list is refreshed and the solved bug is removed from the said list. | | |
| Alternative flows | * 1. Exit by closing the application:   1.a. The developer can close the window by clicking the X button on the top right corner.  1.b. The user can also close the window by pressing ALT + F4 on their own keyboard.  2. There will not be any progress saved, unless the Solve Bug button has been pressed. | | |
| Exceptions | 1. E1: No bug chosen:   1. The developer doesn’t choose any bugs from the active bugs list.  2. The developer clicks the Solve Bug button.  3. The application will pop un an error message box stating that no bug was selected for fixing.   1. E2: Concurrent bug solving:   1. Two users simultaneously choose the same bug for fixing.  2. The users press the Solve Bug button at the same time.  3. Both developers will get a warning that someone is trying to fix the same bug and they will get the username of the person to get in contact.  4. The bug will not be updated and will remain in the active bugs list. | | |
| ID and name | UC - 4 : Modify a pre-existing Bug | | |
| Primary actor | Tester | Secondary actors | Testers, Developers, Admins |
| Description | A tester can modify the description of a pre-existing bug from the active bug list. | | |
| Trigger | A tester modifies the description of an already existing bug. | | |
| Preconditions | PRE 1 – The tester has to be logged in the application  PRE 2 – The bug has to be present in the active bugs list(its status should be “active”) | | |
| Postconditions | POST 1 – The description of the bug will be updated  POST 2 – The updates bug will appear in the active bugs list | | |
| Normal flow | 1. **Modifying a pre-existing bug from the application:** 2. The tester selects the bug he wants to modify. 3. The description and the name of the bug will appear in the right side of the screen. 4. The tester modifies the description to their liking. 5. The tester presses the Modify Bug button. 6. The active bugs list is refreshed and the selected bug is now modified. | | |
| Alternative flows | * 1. Exit by closing the application:   1.a. The tester can close the window by clicking the X button on the top right corner.  1.b. The tester can also close the window by pressing ALT + F4 on their own keyboard.  2. There will not be any progress saved, unless the Modify Bug button has been pressed. | | |
| Exceptions | 1. E1: Empty bug description:   1. The developer chooses to erase the previous description and leave it blank.  2. The developer presses the Modify Bug button.  3. The application will pop un an error message box stating that a bug cannot have an empty description.   1. E2: Concurrent bug solving:   1. Two users simultaneously choose the same bug for modifications.  2. The users press the Modify Bug button at the same time.  3. Both developers will get a warning that someone is trying to modify the same bug and they will get the username of the person to get in contact.  4. The bug will not be modified and will remain in the active bugs list. | | |
| ID and name | UC - 5 : Sort bugs | | |
| Primary actor | Tester, Developer | Secondary actors | - |
| Description | A tester or a developer can sort the active bug list by date(ascending or descending) or alphabetically, by name (ascending or descending). | | |
| Trigger | A tester/developer chooses a sorting filter. | | |
| Preconditions | PRE 1 – The tester/developer has to be logged in the application.  PRE 2 – The active bugs list has to have at least 2 elements. | | |
| Postconditions | POST 1 – The bugs in the active bugs list will be sorted by the filter of choice for every active user | | |
| Normal flow | 1. **Sorting the active bugs list:** 2. The user selects the sorting filter(by date ascending/descending, by name ascending/descending). 3. The active bugs list is automatically refreshed when the choice has been made. 4. The active bugs list is sorted for every logged application user. | | |
| Alternative flows | * 1. Exit by closing the application:   1.a. The user can close the window by clicking the X button on the top right corner.  1.b. The user can also close the window by pressing ALT + F4 on their own keyboard. | | |
| Exceptions | \_ | | |

|  |  |  |  |
| --- | --- | --- | --- |
| ID and name | UC - 6 : Change Password | | |
| Primary actor | Tester, Developer, Admin | Secondary actors | System |
| Description | A logged user can change their password to their liking. | | |
| Trigger | An user changes their password. | | |
| Preconditions | PRE 1 – The user has to be logged into the application.  PRE 2 – The password has to be valid(contain at least 1 capital letter, at least 1 lowercase letter, at least 1 number and at least 1 special character and has to be at least 8-characters long). | | |
| Postconditions | POST 1 – The password of the user will be changed. | | |
| Normal flow | 1. **Changing the password:** 2. The user presses Change Password button. 3. A new screen pops up, with a text field where the new password will be placed. 4. As the user inputs their new password, the system checks it’s validity (it must satisfy the requirement listed above). 5. The tester presses the Change My Password button in the new pop up window. 6. The password of the user who requested the change is now modified and for further access, they have to use the new password. | | |
| Alternative flows | * 1. Exit by closing the application:   1.a. The user can close the window by clicking the X button on the top right corner.  1.b. The user can also close the window by pressing ALT + F4 on their own keyboard.  2. The new pop-up window will close automatically if the parent window is closed, and the password will not be changed if the Change My Password button has not been pressed.   * 1. Close the new pop up wind:   1.a. The user can close the pop-up window by clicking the X button on the top right corner.  1.b. The user can also close the pop-up window by pressing ALT + F4 on their own keyboard.  2. There will not be any progress saved, unless the Change My Password button has been pressed. | | |
| Exceptions | 1. E1: Change password without passing the requirements:   1. The password chosen by the user does not meet the criteria.  2. The user presses the Change My Password button.  3. The application will pop un an error message box stating that the password did not pass the requirements. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| ID and name | UC - 7 : Add a new Employee | | |
| Primary actor | Admin | Secondary actors | System |
| Description | An admin can add a new employee in the system by specifying: name, surname, date of birth, username(has to be unique), password and role(Tester or Developer). | | |
| Trigger | An admin adds a new employee in the system. | | |
| Preconditions | PRE 1 – The admin has to be logged into the application.  PRE 2 – The new employee data has to be valid from a grammatical point of view, the username has to be unique and the role can have one of two options: Tester or Developer | | |
| Postconditions | POST 1 – A new employee will have access to the application, based on their role in the company.  POST 2 – The admin will be able to see only the newly added user in the system. | | |
| Normal flow | 1. **Changing the password:** 2. The admin presses the Add New Employee button. 3. A new screen pops up with text fields for each data: name, surname, username, password, a date picker for birthday and a choice pick for role. 4. The admin inputs all the data for the employee. 5. The admin presses the Add Employee button in the new pop-up window. 6. The new employee is now added to the application and can log in. | | |
| Alternative flows | * 1. Exit by closing the application:   1.a. The admin can close the window by clicking the X button on the top right corner.  1.b. The admin can also close the window by pressing ALT + F4 on their own keyboard.  2. The new pop-up window will close automatically if the parent window is closed, and no employee will be added in the system in the Add Employee button has not been actioned(in the pop up window).   * 1. Close the new pop up window:   1.a. The admin can close the pop-up window by clicking the X button on the top right corner.  1.b. The admin can also close the pop-up window by pressing ALT + F4 on their own keyboard.  2. There will not be any progress saved, unless the Add Employee button has been pressed. | | |
| Exceptions | 1. E1: Add a new employee without valid data:   1. The data input by the admin does not meet the criteria.  2. The admin presses the Add Employee button in the pop-up window.  3. The application will pop un an error message box stating all the invalid data. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| ID and name | UC - 8 : Delete a pre-existing Employee | | |
| Primary actor | Admin | Secondary actors | System |
| Description | An admin can choose and permanently delete a pre-existing employee from the system. After removal, the credentials of the deleted user will not be valid anymore, so logging in the application with those credentials will not be possible anymore.   * Admins cannot delete other admins; | | |
| Trigger | An admin deletes a pre-existing employee from the system. | | |
| Preconditions | PRE 1 – The admin has to be logged in the application  PRE 2 – The employee has to be active | | |
| Postconditions | POST 1 – The employee’s account will be deactivated.  POST 2 – The credentials of the said employee will no longer be available. | | |
| Normal flow | 1. **Deleting a pre-existing employee:** 2. The admin chooses the employee he wants to delete. 3. The admin presses the Delete Employee button. 4. The employee is now deleted from the system and their credentials will no longer be available for use. | | |
| Alternative flows | * 1. Exit by closing the application:   1.a. The admin can close the window by clicking the X button on the top right corner.  1.b. The admin can also close the window by pressing ALT + F4 on their own keyboard.  2. No employee will be deleted from the system if the Delete Employee button has not been pressed. | | |
| Exceptions | 1. E1: No user chosen for deletion:   1. The admin does not choose any user for removal.  2. The developer presses the Delete Employee button.  3. The application will pop un an error message box stating that no employee has been chosen for deletion. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| ID and name | UC - 9 : View the bug history | | |
| Primary actor | Admin | Secondary actors | System |
| Description | An admin can watch the entire bug history and the developer that solved it.   * Admins cannot edit/modify the bug history. | | |
| Trigger | An admin views the bug history. | | |
| Preconditions | PRE 1 – The admin has to be logged in the application | | |
| Postconditions | POST 1 – The admin will see all the bugs that have been fixed. | | |
| Normal flow | 1. **Viewing the entire bug history:** 2. The admin presses the View Bug History button. 3. A new window will pop up where all the fixed bugs will appear, alongside the developer that solved them. | | |
| Alternative flows | * 1. Exit by closing the application:   1.a. The admin can close the window by clicking the X button on the top right corner.  1.b. The admin can also close the window by pressing ALT + F4 on their own keyboard. | | |
| Exceptions |  | | |

|  |  |  |  |
| --- | --- | --- | --- |
| ID and name | UC - 10 : View the employees list | | |
| Primary actor | Admin | Secondary actors | System |
| Description | An admin can watch the entire employees that have worked/are still working for the company. | | |
| Trigger | An admin views all the employees that have ever worked for the company. | | |
| Preconditions | PRE 1 – The admin has to be logged in the application | | |
| Postconditions | POST 1 – The admin will see all the employees. | | |
| Normal flow | 1. **Viewing the employee list:** 2. The employee list is initialized when the admin logs in. | | |
| Alternative flows | * 1. Exit by closing the application:   1.a. The admin can close the window by clicking the X button on the top right corner.  1.b. The admin can also close the window by pressing ALT + F4 on their own keyboard. | | |
| Exceptions |  | | |