|  |  |  |  |
| --- | --- | --- | --- |
| ID and name | **UC-1: Login** | | |
| Primary actor | Employee | Secondary actors | - |
| Description | An employee logs into the software to use the functionalities that they have access to. | | |
| Trigger | An employee wants to use the software. | | |
| Preconditions | PRE-1: The employee already is in the system. | | |
| Postconditions | - | | |
| Normal flow | 1. **Successful Login** 2. The employee enters their credentials correctly 3. The software recognizes the credentials and the employee logs in successfully. | | |
| Alternative flows | - | | |
| Exceptions | - | | |

|  |  |  |  |
| --- | --- | --- | --- |
| ID and name | **UC-2: Programmer/Tester visualizes tasks** | | |
| Primary actor | Programmer/Tester | Secondary actors | - |
| Description | A programmer/tester can see all tasks that have not been completed and/or have the deadline in the future and that are assigned to them. | | |
| Trigger | A programmer/tester wants to complete a task. | | |
| Preconditions | PRE-1: A programmer/tester is logged in. | | |
| Postconditions | - | | |
| Normal flow | 1. **Visualization of task list** 2. After login, the software shows the programmer/tester the list updated in real time. | | |
| Alternative flows | - | | |
| Exceptions | - | | |

|  |  |  |  |
| --- | --- | --- | --- |
| ID and name | **UC-3: See Task Conversation Tree** | | |
| Primary actor | Programmer/Tester | Secondary actors | - |
| Description | A programmer/tester watches the conversation tree of a task that is assigned to him. | | |
| Trigger | A programmer/tester wants to work on a task/verify the conversation tree of a task. | | |
| Preconditions | PRE-1: Programmer/tester in logged in and the task is assigned to them. | | |
| Postconditions | - | | |
| Normal flow | **1.0 See Task Conversation Tree**  1.Programmer/Tester clicks on a task id/task name  2.The software then shows the programmer/tester the conversation tree of the task they clicked. | | |
| Alternative flows | - | | |
| Exceptions | - | | |

|  |  |  |  |
| --- | --- | --- | --- |
| ID and name | **UC-4: Enter new Feature** | | |
| Primary actor | Programmer | Secondary actors | - |
| Description | The programmer enters a new feature in the system based on the task id that the software automatically enters, a link to their implementation, and details on the implementation. | | |
| Trigger | A programmer finished a task and wants to enter it in the system. | | |
| Preconditions | PRE-1: The programmer is logged in  PRE-2: The programmer is on a conversation task tree page.  PRE-3: He didn’t enter a feature implementation on this task that hasn’t been tested or is the first time he implements this task.  PRE-4: The task isn’t closed after the last test if there are bugs. | | |
| Postconditions | POST-1: The assigned tester can now enter a test in the system. | | |
| Normal flow | **1.0 Enters a feature implementation successfully**  1. The programmer pushes the “Add a New Entry” button.  2. The system pops up a form.  3. The programmer enters the link to their implementation of the feature and details that explain the implementation.  4. The software takes the automatically enters the task id and all the data that the programmer entered and adds it to the data base and updates the conversation tree. | | |
| Alternative flows | **1.1 The last implementation wasn’t tested**  2.a The system determines that the last feature implementation wasn’t tested.  2.a.1 The programmer waits until the tester added a test and then if there aren’t bugs he can return to step 1 in Normal Flow | | |
| Exceptions | - | | |

|  |  |  |  |
| --- | --- | --- | --- |
| ID and name | **UC-5: Enter new Test** | | |
| Primary actor | Tester | Secondary actors | - |
| Description | The tester enters a new test in the system based on the task id that the software automatically enters, a link to their implementation, and details on the implementation. If there are any bugs he must check yes and enter a detailed explanation of the bugs that appear. | | |
| Trigger | A tester finished a task and wants to enter it in the system. | | |
| Preconditions | PRE-1: The tester is logged in  PRE-2: The tester is on a conversation task tree page.  PRE-3: There is a feature implementation that isn’t tested. | | |
| Postconditions | POST-1: The assigned programmer can now enter a new feature implementation in the system if the test returned bugs. | | |
| Normal flow | **1.0 Enters a test successfully**  1. The tester pushes the “Add a New Entry” button.  2. The system pops up a form.  3. The tester enters the link to their test and details that explain the test and checks the no option for bugs.  4. The software takes the automatically enters the task id and all the data that the tester entered and updates the conversation tree. | | |
| Alternative flows | **1.1 The last implementation doesn’t exist or was tested**  2.a The system determines whether the last feature implementation was tested or there isn’t a feature implementation.  2.a.1 The tester waits until the programmer adds a new feature implementation and then he returns to step 1 in Normal Flow.  **1.2 There are bugs in the tested implementation**  3.a The teste checks the yes option and enters a detailed explanation of the bugs encountered.  3.a.1 The flow returns to step 4 of the normal flow | | |
| Exceptions | - | | |

|  |  |  |  |
| --- | --- | --- | --- |
| ID and name | **UC-6: Update a Feature** | | |
| Primary actor | Programmer | Secondary actors | - |
| Description | The programmer clicks on the dropdown on his last feature implementation and then clicks on the option to update the implementation. He enters all the data that he wants to change, and the system will update the conversation. | | |
| Trigger | A programmer reworked a task and wants to update the system. | | |
| Preconditions | PRE-1: The programmer is logged in  PRE-2: The programmer is on a conversation task tree page.  PRE-3: He has an untested feature implementation | | |
| Postconditions | - | | |
| Normal flow | **1.0 Updates a feature implementation successfully**  1. The programmer pushes the “V” button on the corner of his last feature implementation.  2. The system pops up a form.  3. The programmer updates the link to their implementation of the feature and the details that explain the implementation.  4. The software takes the automatically enters the task id and all the data that the programmer entered and updates the conversation tree. | | |
| Alternative flows | **-** | | |
| Exceptions | - | | |

|  |  |  |  |
| --- | --- | --- | --- |
| ID and name | **UC-7: Update Test** | | |
| Primary actor | Tester | Secondary actors | - |
| Description | The tester clicks on the dropdown on his last test and then clicks on the option to update the test. He enters all the data that he wants to change, and the system will update the conversation. | | |
| Trigger | A tester finished a task and wants to enter it in the system. | | |
| Preconditions | PRE-1: The tester is logged in  PRE-2: The tester is on a conversation task tree page.  PRE-3: There is a test. | | |
| Postconditions | - | | |
| Normal flow | **1.0 Updates a feature implementation successfully**  1. The tester pushes the “V” button on the corner of his last test.  2. The system pops up a form.  3. The tester updates the link to their test of the feature and the details that explain the test.  4. The software takes the automatically enters the task id and all the data that the programmer entered and updates the conversation tree. | | |
| Alternative flows | **-** | | |
| Exceptions | - | | |

|  |  |  |  |
| --- | --- | --- | --- |
| ID and name | **UC-8: Deletes a Feature Implementation** | | |
| Primary actor | Programmer | Secondary actors | - |
| Description | The programmer presses de “V” button on the last feature implementation and then presses the delete option. | | |
| Trigger | A programmer wants to delete an untested feature implementation | | |
| Preconditions | PRE-1: The programmer is logged in  PRE-2: The programmer is on a conversation task tree page.  PRE-3: He has an untested feature implementation on this page. | | |
| Postconditions | POST-1: The assigned tester has to wait to enter a new test. | | |
| Normal flow | **1.0 Deletes the last feature implementation successfully**  1. The programmer pushes the “V” button on the corner of his last feature implementation.  2. The system deletes the last feature implementation. | | |
| Alternative flows | **-** | | |
| Exceptions | - | | |

|  |  |  |  |
| --- | --- | --- | --- |
| ID and name | **UC-8: Deletes a Test** | | |
| Primary actor | tester | Secondary actors | - |
| Description | The tester presses de “V” button on the last test and then presses the delete option. | | |
| Trigger | A tester wants to delete a test that hasn’t had a response feature implementation. | | |
| Preconditions | PRE-1: The tester is logged in  PRE-2: The tester is on a conversation task tree page.  PRE-3: There is a test that has no response feature implementation | | |
| Postconditions | POST-1: The assigned programmer must wait to enter a new feature implementation. | | |
| Normal flow | **1.0 Deletes the test successfully**  1. The tester pushes the “V” button on the corner of his last test.  2. The system deletes the last test. | | |
| Alternative flows | **-** | | |
| Exceptions | - | | |

|  |  |  |  |
| --- | --- | --- | --- |
| ID and name | **UC-9: Manager visualizes teams** | | |
| Primary actor | Manager | Secondary actors | - |
| Description | A manager can see all teams that are led by them. | | |
| Trigger | A manager wants to enter a team’s page. | | |
| Preconditions | PRE-1: A manager is logged in. | | |
| Postconditions | - | | |
| Normal flow | 1. **Visualization of teams** 2. After login, the software shows the manager the list updated in real time. | | |
| Alternative flows | - | | |
| Exceptions | - | | |

|  |  |  |  |
| --- | --- | --- | --- |
| ID and name | **UC-9: Manager visualizes a team** | | |
| Primary actor | Manager | Secondary actors | - |
| Description | A manager can visualize a team that they are leading | | |
| Trigger | A manager wants to enter a team’s page. | | |
| Preconditions | PRE-1: The manager is logged in.  PRE-2: The manager leads at least one team. | | |
| Postconditions | - | | |
| Normal flow | 1. **Visualization of team’s page** 2. The manager clicks on a team’s profile. 3. The system shows the teams page, which has a list of tasks and all actions that a manager can take. | | |
| Alternative flows | - | | |
| Exceptions | - | | |

|  |  |  |  |
| --- | --- | --- | --- |
| ID and name | **UC-10: Enter a new Account** | | |
| Primary actor | Manager | Secondary actors | - |
| Description | The manager enters a new programmer/tester in the system based on the name, username and email, password being generated by the system and sent it to the programmer/tester and sends it to the programmer/tester trough mail. | | |
| Trigger | A manager wants to enter a new programmer/tester in a team. | | |
| Preconditions | PRE-1: The Manager is logged in.  PRE-2: The Manager is on a team profile. | | |
| Postconditions | POST-1: The programmer/tester is now part of a team. | | |
| Normal flow | **1.0 Enters a Programmer successfully**  1. The manager pushes the “Add a New Account” button.  2. The system pops up a form.  3. The manager enters the name, email and username preferred by the new programmer and checks the programmer option for role.  4. The software generates the passwords and take all the data that the manager entered and updates team’s composition. | | |
| Alternative flows | **1.1 Enters a Tester successfully**  3.1 The manager checks the tester option for role. The system returns to step 4 of the Normal Flow. | | |
| Exceptions | **1.0.E1 There already is a programmer/tester with this email/username**  3. The system informs the manager that there already is a programmer/tester with that email/username  4a. If the manager enters new data we go back to step 3 in Normal Flow.  4b. Else the use case is canceled. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| ID and name | **UC-11: Update Account** | | |
| Primary actor | Manager | Secondary actors | - |
| Description | The manager presses de update an account button and he then enters an already existing username or email and updates the rest of the data. | | |
| Trigger | A manager wants to update a programmer/tester from a team. | | |
| Preconditions | PRE-1: The Manager is logged in.  PRE-2: The Manager is on a team profile. | | |
| Postconditions | - | | |
| Normal flow | **1.0 Updates a programmer successfully**  1. The manager pushes the “Update an Account” button.  2. The system pops up a form.  3. The manager enters the new data preferred by the programmer and checks the programmer option.  4. The software checks the new data with the old data and takes all the data that the manager entered and updates team’s composition. | | |
| Alternative flows | **1.1 Updates a tester successfully**  3.1 The manager checks the tester option. The system continues with step 4 of the Normal Flow. | | |
| Exceptions | **1.0.E1 There is not a programmer/tester with this email/username**  3. The system informs the manager that there is not a programmer/tester with that email/username  4a. If the manager enters new data we go back to step 3 in Normal Flow.  4b. Else the use case is canceled. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| ID and name | **UC-12: Delete Account** | | |
| Primary actor | Manager | Secondary actors | - |
| Description | The manager presses de delete an account button and he then enters an already existing username or email and delete all data related. | | |
| Trigger | A manager wants to delete an account from a team. | | |
| Preconditions | PRE-1: The Manager is logged in.  PRE-2: The Manager is on a team profile. | | |
| Postconditions | - | | |
| Normal flow | **1.0 Delete a programmer successfully**  1. The manager pushes the “Delete an Account” button.  2. The system pops up a form.  3. The manager enters the username or email of a programmer account and checks the programmer option.  4. The software checks the username or email to exist and deletes the data related to it. | | |
| Alternative flows | **1.1 Delete a tester successfully**  3.1 The manager checks the tester option. The system continues with step 4 of the Normal Flow. | | |
| Exceptions | **1.0.E1 There is not a programmer/tester with this email/username**  3. The system informs the manager that there is not a programmer/tester with that email/username  4a. If the manager enters new data we go back to step 3 in Normal Flow.  4b. Else the use case is canceled. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| ID and name | **UC-13: Enter a new Task** | | |
| Primary actor | Manager | Secondary actors | - |
| Description | The manager enters a new task based on details regarding the task the deadline, priority, programmer, tester and type. | | |
| Trigger | A manager wants to give a programmer tester pair a new task. | | |
| Preconditions | PRE-1: The Manager is logged in.  PRE-2: The Manager is on a team profile. | | |
| Postconditions | POST-1: The task is now assigned. | | |
| Normal flow | **1.0 Enters a Task successfully**  1. The manager pushes the “Add a New Task” button.  2. The system pops up a form.  3. The manager enters the details regarding the task the deadline, priority, programmer, tester and type.  4. The software assigns the task to the 2 employees mentioned. | | |
| Alternative flows | **-** | | |
| Exceptions | **1.0.E1 The deadline is not of a date type**  3. The system informs the manager that the deadline is not of an correct type.  4a. If the manager enters new data we go back to step 3 in Normal Flow.  4b. Else the use case is canceled. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| ID and name | **UC-14: Update a Task** | | |
| Primary actor | Manager | Secondary actors | - |
| Description | The manager updates a task based on details regarding the task the deadline, priority, programmer, tester and type. | | |
| Trigger | A manager wants to update for a programmer tester pair a task. | | |
| Preconditions | PRE-1: The Manager is logged in.  PRE-2: The Manager is on a team profile.  PRE-3: There is a task in the list of this team. | | |
| Postconditions | POST-1: A task is now updated. | | |
| Normal flow | **1.0 Update a Task successfully**  1. The manager pushes the “Update a Task” button.  2. The system pops up a form.  3. The manager enters the details regarding the task the deadline, priority, programmer, tester and type.  4. The software update the task of the 2 employees mentioned. | | |
| Alternative flows | **-** | | |
| Exceptions | **1.0.E1 The deadline is not of a date type**  3. The system informs the manager that the deadline is not of an correct type.  4a. If the manager enters new data we go back to step 3 in Normal Flow.  4b. Else the use case is canceled. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| ID and name | **UC-15: Archive a Task** | | |
| Primary actor | Manager | Secondary actors | - |
| Description | The manager archives a task clicked from the list. | | |
| Trigger | A manager wants to archive a task for a programmer tester. | | |
| Preconditions | PRE-1: The Manager is logged in.  PRE-2: The Manager is on a team profile.  PRE-3: There is a task in the list of this team. | | |
| Postconditions | POST-1: A task is now archived. | | |
| Normal flow | **1.0 Update a Task successfully**  1. The manager clicks a task from the list and then the “Delete a Task” button.  2. The system pops up a form.  3. The manager pushes the Archive a Task button  4. The software archives the task . | | |
| Alternative flows | **-** | | |
| Exceptions | - | | |

|  |  |  |  |
| --- | --- | --- | --- |
| ID and name | **UC-16: Logout** | | |
| Primary actor | Employee | Secondary actors | - |
| Description | An employee logs out of the software. | | |
| Trigger | An employee wants to log out. | | |
| Preconditions | PRE-1: The employee is logged in. | | |
| Postconditions | - | | |
| Normal flow | **1.0 Successful Logout**   1. The employee presses the logout button. 2. The software logs him out and shows the login page. | | |
| Alternative flows | - | | |
| Exceptions | - | | |