Use case for Bug Application

ID and name	UC1 – Login		
Primary actor	Tester, Developer	Secondary actors	System
Description	An employee for the BA application can login using his credentials provided by the software company and he can use them to log in to the application.		
Trigger	An existing employee that is a tester goes to the terminal and logs in.		
Preconditions	PRE 1 – The tester is an employed person at the company and already has the credentials to enter the application.		
Postconditions	POST 1 – The user is logged into the application.		
Normal flow	Logging into an employee account 1. The employee clicks on the log in button 2. The system loads the login page 3. The employee enters his login credentials into the application 4. The employee presses on the button “Log In” 5. The system goes into the startup page.		
Alternative flows	He can exit the form at any time by clicking exit on the form.		
Exceptions	The employee has introduced invalid data about his login credentials. • The BA application gives an error message stating that the credentials are incorrect • Return to step 3 in the normal flow.		

ID and name	UC2 – Place a bug		
Primary actor	Tester	Secondary actors	Tester

Description	The option for a tester to add a bug in the list of bugs that a developer should fix.
Trigger	A tester indicates that a bug should be fixed.
Preconditions	PRE-1 The tester needs to be logged in to the application.
Postconditions	POST- 1 The bug will be placed into a list of errors, with the status of “unsolved”. POST – 2 The list is updated in real time with any changes. POST –3 A trigger like notification will also appear noticing the logged users that a new bug has been placed.
Normal flow	**1. 0 Post a bug** 1. The tester uses the button “report a bug” in the BA Application. 2. After the initial press, the BA application will create a new form. 3. In the form created the tester can input the information about the bug, the source and the effect in the application. Additionally, he can input a priority status based on how fast the bug needs to be fixed. 4. The tester confirms the bug or requests to modify an existing bug. (returns to step 2)
Alternative flows	• The user can exit the form page by clicking on the exit button in the form.
Exceptions	

ID and name	UC2 – Solve a bug		
Primary actor	Developer	Secondary actors	Developers
Description	The option for a developer to remove a bug, or to replace the status of the bug in “resolved”.		
Trigger	A developer states that a bug has been fixed.		
Preconditions	PRE-1 The developer needs to be logged in to the application.		
Postconditions	POST- 1 The bug will either be removed from the list depending on the security of the bug or the status can be changed to “solved” (the severity of the bug can determine the developer to not remove the bug if it contains critical information worth noted) POST – 2 The list is updated in real time with any changes.		
Normal flow	**1. Solve a bug** 1. The developer presses on a bug in the existing list. 2. The pressing again on the item in the list he can modify the status of the bug, add additional information, and change the status from “unsolved” to “solved”, or he can remove it.		
Alternative flows	Delete or change the status of the bug. 1. From step 2 of the normal flow there are 2 bifurcations. • He can either change the state of the bug, by modifying it, while also adding extra information in the ticket • He can remove it if the threat does not represent a real threat, or is it a false positive (it is not actually a bug)		
Exceptions			

ID and name	UC2 – Modify a bug		
Primary actor	Tester, Developer	Secondary actors	-
Description	The option for both the tester and the developer to modify a bug.		
Trigger	A tester indicates that a bug should be updated.		
Preconditions	PRE-1 The tester needs to be logged in to the application.		

Postconditions	POST- 1 The BA application will open a new form and the user can modify the existing bug with different values. POST – 2 The list is updated in real time with any changes.
Normal flow	**1. 0 Post a bug** 1. The tester clicks on a bug from the list then on the button “modify the bug” in the BA Application. 2. After the initial press, the BA application will create a new form. 3. In the form created the tester can input the information about the bug, the source, and the effect in the application. Additionally, he can input a priority status based on how fast the bug needs to be fixed. 4. The tester modifies an existing bug.
Alternative flows	• Upon entering invalid data, or deleting values from the existing forms, he will pe prompted with an error. • He will return to the initial stage where he is in the startup screen.
Exceptions	If other users want to modify the current bug, the button “modify a bug” will be closed for all the other users, if a user is currently modifying a bug.

ID and name	UC2 – Sort bugs		
Primary actor	Tester, Developer	Secondary actors	-
Description	The option for both the tester and the developer to sort the list of bugs.		
Trigger	A developer wants to sort the list.		
Preconditions	PRE-1 The tester needs to be logged in to the application.		
Postconditions	POST- 1 The BA application will sort the existing bugs by priority.		
Normal flow	**1. Sort a bug** **1.** The user will have a button next to the list displayed of bugs. 2. By pressing it the list will become sorted by their priority.		
Alternative flows	• There will also be an option that sorts all the old, recent bugs if there are existing bugs with a similar level of priority.		
Exceptions	If the list is empty, it will return an error message stating that the list of bugs is currently empty.		

ID and name	UC1 – Log out		
Primary actor	Tester, Developer	Secondary actors	System
Description	An employee can log out of the BA application.		
Trigger	An existing employee that is a tester goes to the terminal and logs out.		
Preconditions	PRE 1 – The tester is an employed person at the company and has already entered the application.		
Postconditions	POST 1 – The user closed the application		
Normal flow	The user will simply press on the exit button on the startup page and the application will close.		
Alternative flows	-		
Exceptions	-		

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.

Examples

For a hypothetical *Cafeteria Ordering System*[1]:

ID and name	UC-1: Order a Meal		
Primary actor	Patron	**Secondary actors**	Cafeteria Inventory System
Description	A Patron accesses the Cafeteria Ordering System from either the corporate intranet or external Internet, views the menu for a specific date, selects food items, and places an order for a meal to be picked up in the cafeteria or delivered to a specified location within a specified 15-minute time window.		
Trigger	A Patron indicates that he wants to order a meal.		
Preconditions	PRE-1. Patron is logged into COS. PRE-2. Patron is registered for meal payments by payroll deduction.		
Postconditions	POST-1. Meal order is stored in COS with a status of "Accepted." POST-2. Inventory of available food items is updated to reflect items in this order. POST-3. Remaining delivery capacity for the requested time window is updated.		
Normal flow	**1.0 Order a Single Meal** 1. Patron asks to view menu for a specific date. (see 1.0.E1, 1.0.E2) 2. COS displays menu of available food items and the daily special. 3. Patron selects one or more food items from menu. (see 1.1) 4. Patron indicates that meal order is complete. (see 1.2) 5. COS displays ordered menu items, individual prices, and total price, including taxes and delivery charge. 6. Patron either confirms meal order (continue normal flow) or requests to modify meal order (return to step 2). 7. COS displays available delivery times for the delivery date. 8. Patron selects a delivery time and specifies the delivery location. 9. Patron specifies payment method. 10. COS confirms acceptance of the order. 11. COS sends Patron an email message confirming order details,		

[1] Examples adapted from Wiegers, K. E. & Beatty, J. (2013) Software requirements . 3rd ed. Redmond, WA: Microsoft Press.

	price, and delivery instructions. 12. COS stores order, sends food item information to Cafeteria Inventory System, and updates available delivery times.
Alternative flows	**1.1 Order multiple identical meals** 1. Patron requests a specified number of identical meals. (see 1.1.E1) 2. Return to step 4 of normal flow. **1.2 Order multiple meals** 1. Patron asks to order another meal. 2. Return to step 1 of normal flow.
Exceptions	**1.0.E1 Requested date is today and current time is after today's order cutoff time** 1. COS informs Patron that it's too late to place an order for today. 2a. If Patron cancels the meal ordering process, then COS terminates use case. 2b. Else if Patron requests another date, then COS restarts use case. **1.0.E2 No delivery times left** 1. COS informs Patron that no delivery times are available for the meal date. 2a. If Patron cancels the meal ordering process, then COS terminates use case. 2b. Else if Patron requests to pick the order up at the cafeteria, then continue with normal flow, but skip steps 7 and 8. **1.1.E1 Insufficient inventory to fulfill multiple meal order** 1. COS informs Patron of the maximum number of identical meals he can order, based on current available inventory. 2a. If Patron modifies number of meals ordered, then return to step 4 of normal flow. 2b. Else if Patron cancels the meal ordering process, then COS terminates use case.

ID and name	UC-5 Register for Payroll Deduction		
Primary actor	Patron	**Secondary actors**	Payroll System
Description	Cafeteria patrons who use the COS and have meals delivered must be registered for payroll deduction. For noncash purchases made through the COS, the cafeteria will issue a payment request to the Payroll System, which will deduct the meal costs from the next scheduled employee payday direct deposit.		
Trigger	Patron requests to register for payroll deduction, or Patron says yes when COS asks if he wants to register.		
Preconditions	PRE-1. Patron is logged into COS.		
Postconditions	POST-1. Patron is registered for payroll deduction.		
Normal flow	**5.0 Register for Payroll Deduction** 1. COS asks Payroll System if Patron is eligible to register for payroll deduction. 2. Payroll System confirms that Patron is eligible to register for payroll deduction. 3. COS asks Patron to confirm his desire to register for payroll deduction. 4. If so, COS asks Payroll System to establish payroll deduction for Patron. 5. Payroll System confirms that payroll deduction is established. 6. COS informs Patron that payroll deduction is established.		
Alternative flows	None		
Exceptions	5.0.E1 Patron is not a full time employee. 5.0.E2 Patron is already enrolled for payroll deduction.		

Extra step: Traceability

For this extra step, you will add traceability information for each use case by adding a new field to the template:

Method-level traces	<fully.qualified.ClassName>#<methodName> ...

Any method that implements the functionality described in the normal flow, alternative flow or exceptions should be included in this field. This means that the method that is initially executed and any methods of any classes that the work is delegated to should be included.

Examples for previous use cases:

UC-1:

Method-level traces	my.company.ordering.MenuWidget#dateClicked my.company.ordering.MenuWidget#completeOrder my.company.ordering.InventoryInterface#checkInventory ...

UC-5:

Method-level traces	my.company.payroll.PayrollInterface#checkEligibility my.company.payroll.RegistrationForm#confirm ...