Use case table template

Use Case Number:	
Use Case Name:	
Participating Actors:	
Goal:	
Trigger:	
Precondition:	
Postcondition:	
Basic Flow	
Exceptions	
Qualities:	
Functionality:	
Constraints:	
Includes:	
Extends:	
Related Artifacts:	
Notes:	
Open Issues:	

Use Case Number:	1
Use Case Name:	***Start Skin Observer application***
Participating Actors:	main user (patient)
Goal:	initialize storage methods/check for previously stored data, open up the application for the user
Trigger:	this user clicks the app from the android app menu list
Precondition:	none
Postcondition:	The application is now at start up
Basic Flow	1/ User click application icon 2/ Application is launched
Exceptions	2.1/ application fails to launch display error message
Qualities:	System responds within 1s
Functionality:	The application has: search/add/delete/view/edit ---> 1/ search: photo, skin condition, group ---> 2/ create: a photo, a skin condition(entry) ---> 3/ add: entries to group, photos to entries ---> 4/ delete: photos, entries, groups ---> 5/ view: photos, skin conditions, groups, ---> 6/ edit: photos, skin conditions, groups
Constraints:	The UI looks aesthetically pleasing
Includes:	
Extends:	
Related Artifacts:	As a user, I want to keep a visual log of any skin conditions I might . The purpose could be to gather a personal history of a medical condition as to provide evidence to a physician.
Notes:	We'll add details into each function later ***-Tests for use case 1-*** 1/ check if the application launches properly 2/ check if all the GUI and make sure it works 3/ check if the application closes without errors

Open Issues:	

Use Case Number:	2
Use Case Name:	***Take Photo***
Participating Actors:	main User (patient)
Goal:	Allow the User to take a photo, and then have the application store the photo with a storage method.
Trigger:	The User chooses *take photo* option.
Precondition:	User knows the target area; there is a camera.
Postcondition:	On success, all relevant information of the captured photo is stored.
Basic Flow	1/ User takes a photo 2/ Use case 3: Timestamp A Photo 3/ System displays the taken photo 4/ System prompts the User to enter the name of a new *skin condition* or to choose some existing *skin condition* that the taken photo refers to 5/ User enters the name of a new *skin condition* or choose some existing *skin condition* 6/ System prompts the User to enter the name of a new *group* or to choose some existing *group* for the taken photo 7/ User enters the name of a new *group* or choose some existing *group* 8/ System prompts the User to enter Annotation to be associated with the taken photo 9/ User enters some annotation about the photo 10/ System saves the taken photo
Exceptions	1/ If the camera malfunctions 1.1/ System displays an error 1.2/ System returns to the previous activity 8/ If the System can't save the taken photo due to perhaps insufficient amount of storage 8.1/ System displays an error 8.2/ System prompts the User to clear some memory or to discard the

	taken photo 8.3/ If the User chooses *clear some memory* return to step 3 8.4/ If the User chooses *discard the taken photo* return to the previous activity
Qualities:	
Functionality:	
Constraints:	
Includes:	Timestamp A Photo, Tag a photo, add skin conditions to photo
Extends:	
Related Artifacts:	As a user, I want pictures I take to be stored so that I may recall them and view them later. Camera
Notes:	***-Tests for use case 2-*** 1/ verify the application can capture an image with its camera 2/ verify the photo can be displayed on screen 3/ test to see if all user interaction with the User Interface works correctly such as user input, and the user selections on the skin condition, and group options. 4/ verify that the picture can be stored 5/ verify that the picture can be reloaded after being stored 6/ verify how the program handles running out of memory to store a new photo
Open Issues:	Can we implement a better looking camera?

Use Case Number:	3
Use Case Name:	***Timestamp A Photo***
Participating Actors:	This is done automatically by the system

Goal:	Timestamp A Photo
Trigger:	The User has taken a photo
Precondition:	The photo is taken successfully by the camera
Postcondition:	On success, System automatically records the time stamp of the taken photo
Basic Flow	1/ Use case 2 2/ System automatically records the time stamp on and add to the taken photo
Exceptions	2/ If the System cannot add a time stamp 2.1/ System notifythe user that it cannot record the time stamp. 2.2/ System does not record time stamp of the photo.
Qualities:	Time stamp is recoded with the following format DD/MM/YYYY HH:MM
Functionality:	
Constraints:	
Includes:	
Extends:	
Related Artifacts:	As a user, I want all pictures I take to be timestamped so that I can measure the time between photos and know when an event occurred.
Notes:	This use case is an activity inside of the "take a photo" class, it will automatically add a timestamp to any photos taken by the user that they wish to save to memory, there are many simple java methods to implement this task. ***-Tests for use case 3-*** 1/ check that stored photos have a time stamp 2/ check that they have the correct time stamp

Open Issues:	

Use Case Number:	4
Use Case Name:	***View Photo***
Participating Actors:	main User (patient)
Goal:	Allow the User to review a taken photo
Trigger:	The User chooses *view photo* option
Precondition:	User knows which photo he/she wishes to review
Postcondition:	On success, all relevant information of the captured photo (*skin condition, body part and timestamp)* is displayed
Basic Flow	1/ System prompts the User to choose a photo to view 2/ User chooses a photo to view 3/ System displays the chosen photo 4/ System gives option to view photo annotation and other misc information about the photo
Exceptions	3/ If the System cannot display the chosen photo because the photo may not exist or it may be broken 3.1/ System notify the user that it cannot display the chosen image 3.2/ System returns to step 1
Qualities:	
Functionality:	
Constraints:	

Includes:	
Extends:	
Related Artifacts:	As a user, I want to be able to review any photos I have taken and view them so that I can see any changes.
Notes:	***-Tests for use case 4-*** 1/ verify the user may view a photo
Open Issues:	

Use Case Number:	5
Use Case Name:	***Compare Two Photos***
Participating Actors:	main User
Goal:	View two photos simultaneously to observe any change of some skin condition
Trigger:	The User chooses *compare multiple photos* option
Precondition:	The User knows up front which photos he/she wishes to compare
Postcondition:	On success, System displays two photos simultaneously
Basic Flow	1/ User choose to view all photos OR to choose a *skin condition* in a list to show all photos of the chosen *skin condition (use case View skin condition)* OR to choose a *group* in a list to show all photos of the chosen *group (use case View group)* 2/ System prompts User to choose two photos for comparison 3/ User choose to compare photos 4/ System shows two photos simultaneously by default setting 5/ User to choose type of view (side-by-side or overlay) 6/ System shows photos side-by-side or overlay depending on User's choice

Exceptions	2/ User chooses one or more than two photos 2.1/ Ask the user to choose two photos to compare
Qualities:	
Functionality:	
Constraints:	Only 2 photos can be viewed simultaneously
Includes:	
Extends:	
Related Artifacts:	As a user, I want a method to organize these pictures so that I can view pictures of the same bit of skin over time and see if there is any progression or growth. For instance I might want to organize some photos that are related to my "gross mole that occurs on my right hand".
Notes:	***-Tests for use case 5-*** 1/ verify the application can display two photos properly 2/ verify the program acts appropriately when the user only has 0 images, and when the user only has 1 image 3/ verify the user can choose two photos 4/ verify the view works properly for displaying the two photos
Open Issues:	

Use Case Number:	6
Use Case Name:	***Tag a photo***
Participating Actors:	main User (patient)
Goal:	Add a photo to an existing group or a new tag
Trigger:	User selects *tag a photo*

Precondition:	User knows which photo to add to some tag
Postcondition:	On success, the chosen photo is added to some tag
Basic Flow	1/ System prompts the User to choose a photo 2/ User selects a photo 3/ System prompts User to assign some existing *tag* or create a new *tag* for the chosen photo 4/ The user chooses existing *tag* or the newly created *tag* for the taken photo
Exceptions	3/ If there is an error when making a new tag 3.1/ System shows the error message 3.2/ Return to the previous activity
Qualities:	
Functionality:	
Constraints:	
Includes:	
Extends:	
Related Artifacts:	As a user if I have an organizational tag or group, I want to be able to add photos to that group so that I can further track the progression of something relevant to that group. # This use case happens when the user hasn't placed some photo in a *group* when the photo was taken
Notes:	***-Tests for use case 6-*** 1/ verify that a photo can be added to a tag 2/ handle the null case, where the user tries to add null to a tag?
Open Issues:	

Use Case Number:	7

Use Case Name:	***Create Transparent Layer***
Participating Actors:	User
Goal:	Use some photo as a transparent layer to help user take a new photo in the same position as the layer
Trigger:	User selects *transparent layer* option inside the *taking photo* view
Precondition:	User knows the photo to make it a transparent layer
Postcondition:	On success, a *transparent layer* appears on a screen
Basic Flow	1/ System prompts the User to choose a photo 2/ User selects a photo 3/ System displays the *transparent layer* on the screen
Exceptions	3/ If there is an error when making a new group 3.1/ System shows the error message 3.2/ Return to the previous activity
Qualities:	
Functionality:	
Constraints:	
Includes:	
Extends:	Take Photo
Related Artifacts:	As a user, I want some method of helping me take consistent photos, so that when I show the doctor any progression such as the growth of a mole is evident.

Notes:	This use case happens when the user wishes to take a new photo based on some existing photo of the same skin condition. ***-Tests for use case 7-*** 1/ verify that the layer works 2/ verify a photo can still be capture with the layer 3/ handle the case when no photos are present for the user to use as a layer
Open Issues:	This use case requires knowledge of image processing, thus it will be added later if possible.

Use Case Number:	8
Use Case Name:	***Create Reminder***
Participating Actors:	User
Goal:	Remind user of taking photo of some skin condition regularly
Trigger:	User choose *set reminder* option
Precondition:	User knows the skin condition and the group
Postcondition:	The System sets the alarm to remind User of taking photo and specify the skin condition and group automatically based on the info of the alarm
Basic Flow	1/ System prompts the User to choose the skin condition in a list 2/ User chooses some skin condition 3/ System prompts the User to choose the group in a list 4/ User chooses some group 5/ System prompts the User to enter other information of the alarm: time, repetition, name, additional info. 6/ User change the default value of these information if needed 7/ System sets the reminder
Exceptions	7/ If System cannot set the reminder 7.1/ System displays an error message 7.2/ Return to previous activity
Qualities:	
Functionality:	

Constraints:	
Includes:	
Extends:	
Related Artifacts:	As a user, I want some method of helping me take consistent photos, so that when I show the doctor any progression such as the growth of a mole is evident.
Notes:	***-Tests for use case 8-*** 1/ verify a reminder can be created
Open Issues:	

Use Case Number:	9
Use Case Name:	***Retake Photo***
Participating Actors:	User
Goal:	Retake an erroneous photo
Trigger:	User choose to retake a photo
Precondition:	The user has and able to modify the photo he or she want to retake
Postcondition:	The old unwanted photo is replaced by a new photo
Basic Flow	1/ User choose the option to retake that photo 2/ System invokes the function in use case 2 3/ New photo overwrites the old photo
Exceptions	3/ If the new photo cannot overwrite the old photo 3.1/ System displays an error message 3.2/ The user will then have to go and delete the previously taken photo

Qualities:	The process of retaking a photo is simple and quick.
Functionality:	
Constraints:	The new photo must reuses any appropriate information attached in the old photo.
Includes:	Take Photo
Extends:	
Related Artifacts:	# As a user, I should be able to retake photos I am taking, if I fail to take a photo # I want to correct it, so that I do not have erroneous photos. --Camera
Notes:	***-Tests for use case 9-*** 1/ verify the user is able to use the retake photo option 2/ verify that the previous photo has been removed from the application 3/ verify the new photo is able to be stored by the application
Open Issues:	

Use Case Number:	10
Use Case Name:	***View Skin Conditions***
Participating Actors:	main User
Goal:	Show the list of skin conditions
Trigger:	The User chooses *view list of skin conditions*
Precondition:	
Postcondition:	On success, System displays the list of skin conditions or an empty list if there is no skin condition
Basic Flow	1/ System displays the list of skin conditions
Exceptions	1/ If System cannot display the list of skin conditions 1.1/ System displays an error message 1.2/ Return to previous activity

Qualities:	The list must be easy to read.
Functionality:	
Constraints:	
Includes:	View Multiple Photos
Extends:	
Related Artifacts:	
Notes:	***-Tests for use case 10-*** 1/ verify all the skin conditions are displayed correctly 2/ verify the case where no skin conditions exist is handled correctly
Open Issues:	

Use Case Number:	11
Use Case Name:	***View Groups***
Participating Actors:	main User
Goal:	Show the list of groups
Trigger:	The User chooses *view list of tags*
Precondition:	
Postcondition:	On success, System displays the list of groups or an empty list if there is no tag
Basic Flow	1/ System displays the list of tags
Exceptions	1/ If System cannot display the list of tags 1.1/ System displays an error message 1.2/ Return to previous activity

Qualities:	The list must be easy to read
Functionality:	
Constraints:	
Includes:	View Multiple Photos
Extends:	
Related Artifacts:	
Notes:	***-Tests for use case 11-*** 1/ verify all the appropriate tags are displayed 2/ verify the case is handled where there are no tags
Open Issues:	

Use Case Number:	12
Use Case Name:	***View Photos in a Skin Condition***
Participating Actors:	main User
Goal:	Show the list of photos in a chosen skin condition
Trigger:	The User clicks on a skin condition in the list of skin conditions
Precondition:	
Postcondition:	On success, System displays the list of photos in the chosen skin condition
Basic Flow	1/ User chooses a skin condition in the list 2/ System displays a list of photos in the chosen skin condition 3/ User click on a photo to view it bigger
Exceptions	The list of photos must be loaded fast and correctly

Qualities:	
Functionality:	
Constraints:	
Includes:	
Extends:	View Skin Conditions
Related Artifacts:	
Notes:	***Tests for use case 12-*** 1/ verify the case is handled when no skin condition is selected 2/ check to see if all the photos with that skin condition are displayed
Open Issues:	

Use Case Number:	13
Use Case Name:	***View Photos in Tag***
Participating Actors:	main User
Goal:	Show the list of photos in a chosen tag
Trigger:	The User clicks on a group in the list of tags
Precondition:	
Postcondition:	On success, System displays the list of photos in the chosen tag
Basic Flow	1/ User chooses a group in the list of tags 2/ System displays a list of photos in the chosen tag 3/ User click on a photo to view it bigger
Exceptions	

Qualities:	
Functionality:	
Constraints:	
Includes:	
Extends:	View tags
Related Artifacts:	
Notes:	***-Tests for use case 13-*** 1/ verify the case where no tags are selected 2/ verify that the correct photos are displayed corresponding to the tag
Open Issues:	

Use Case Number:	14
Use Case Name:	***Edit Photo Info***
Participating Actors:	main user (patient)
Goal:	Edit information of a photo
Trigger:	User chooses the *edit* option in the menu
Precondition:	User know what information he/she want to edit
Postcondition:	On success, all relevant information of the chosen photo is updated
Basic Flow	1/ System has displayed info of the chosen photo at the end of use case *View photo* 2/ User chooses some piece of information of the entry to edit (time stamp is not editable) (optional) 3/ User confirms new information 4/ System saves and updates the entry

Exceptions	3/ If the new information is not valid 3.1/ System displays an error 3.2/ System returns to step 2
Qualities:	System assists user in editing by providing appropriate keys
Functionality:	
Constraints:	Use DD/MM/YYYY format for the date of a log entry; a short mnemonic name is no more than 100 characters
Includes:	
Extends:	View Photo
Related Artifacts:	
Notes:	***<u>-Tests for use case 14-</u>*** 1/ verify the photo is saved properly after being edited 2/ verify the new fields are updated correctly 3/ handle the case where the new fields are null

Open Issues:	

Use Case Number:	15
Use Case Name:	***Delete Photo***
Participating Actors:	main user (patient)
Goal:	Delete an existing photo
Trigger:	User chooses the *Delete* option in the menu after choosing some photo
Precondition:	User knows what photo he/she want to delete
Postcondition:	On success, delete the photo

Basic Flow	1/ System has displayed the chosen photo at the end of *View photo* use case 2/ User chooses “delete” option to delete the photo 3/ System prompts the user to confirm action 4/ Confirmation of delete action
Exceptions	4/ if System cannot delete the photo 4.1/ System shows an error message 4.2/ Return to step 1
Qualities:	
Functionality:	
Constraints:	
Includes:	
Extends:	
Related Artifacts:	

Notes:	***-Tests for use case 15-*** 1/ verify that the photo is removed, and all other data relating to that photo is also removed
Open Issues:	

Use Case Number:	16
Use Case Name:	Delete Tag/Skin Condition
Participating Actors:	main user (patient)
Goal:	Delete some tag or some skin condition
Trigger:	User chooses the *Delete* option in the menu after the System shows a list of tags or a list of skin conditions
Precondition:	

Postcondition:	On success, remove the chosen tag or skin condition out of the information of all photos that belong to the tag or skin condition
Basic Flow	1/ System has displayed the list of groups or the list of skin conditions at the end of *View tags* or *View skin conditions* 2/ User chooses *delete* option in the menu 3/ System prompts User to choose tags or skin conditions to delete 4/ User chooses tags or skin conditions to delete 5/ System prompts User to confirm deletion 6/ User confirms deletion 7/ System identifies all photos that belong to the deleted tag or deleted skin condition and remove the group or skin condition out of the information of those photos; remove the tag or skin condition
Exceptions	7/ if System cannot modify info of the photos 7.1/ System shows an error message 7.2/ Not delete the tag or skin condition 7.3/ Return to previous activity
Qualities:	
Functionality:	
Constraints:	
Includes:	

Extends:	
Related Artifacts:	
Notes:	***-Tests for use case 16-*** 1/ verify that the tag/skin condition is deleted 2/ verify that all photos who were in the tag or had the skin condition tags now no longer have the tags
Open Issues:	

Use Case Number:	17
Use Case Name:	View Multiple Photos
Participating Actors:	main User
Goal:	Show the list of photos

Trigger:	The User chooses a criteria to view a list of photos
Precondition:	
Postcondition:	On success, System displays the list of photos or an empty list if there is no photo meet the criteria
Basic Flow	1/ System displays a list of photos meet the criteria 2/ User click on a photo to view it bigger
Exceptions	
Qualities:	
Functionality:	
Constraints:	
Includes:	

Extends:	
Related Artifacts:	
Notes:	***<u>-Tests for use case 17-</u>*** 1/ verify that the list of photos is displayed correctly 2/ verify that the list of photos for 0 photos doesn’t create errors
Open Issues:	

Use Case Number:	18
Use Case Name:	***Password encryption***
Participating Actors:	Main user
Goal:	Allow returned user to log in with his/her password

Trigger:	This user clicks the app from the android app menu list
Precondition:	None
Postcondition:	Allow user to log in if the password is correct, reject otherwise
Basic Flow	1/ User click application icon 2/ System prompts user for the password 3/ User enters password 4/ System checks the password 5/ System starts the application if password is correct. Otherwise, display a wrong password message. Return to step 2
Exceptions	2.1/ application fails to launch Display error message
Qualities:	System responds within 1s
Functionality:	
Constraints:	

Includes:	
Extends:	
Related Artifacts:	As a user, I am worried that if my phone gets stolen people will see my photos. I want my photos secured via a password
Notes:	Implement encryption of the folder containing photos
Object Oriented Analysis	

Use Case Number:	19
Use Case Name:	***Reminder***
Participating Actors:	Main user

Goal:	Remind the user of taking consistent photos to keep track of his/her skin conditions.
Trigger:	This user chooses “reminder” option on the main screen of the application
Precondition:	None
Postcondition:	Allow user to create alarms that go off at set-up times.
Basic Flow	1/ User chooses “reminder” option 2/ System shows a list of alarms created 3/ User chooses to create a new alarm or edit some existing one 4/ System creates a new alarm or modifies the existing one
Exceptions	2/ Fails to load the list of alarms 2.1/ Display an error message 2.2/ Return to the main screen
Qualities:	
Functionality:	The alarm goes of at a specific point of time that is set by the user. If the time set up already passes the current time, the alarm will go off immediately. One-time or repeating alarm types are supported.
Constraints:	

Includes:	
Extends:	
Related Artifacts:	As a user, I want notifications about when I should take photos again. I'm forgetful and being told every so often to take new photos would be helpful. Certain photos I don't have to retake again.
Notes:	The alarm is not bound to any particular photos. The user needs to annotate the alarm to know which skin condition he/she would like to observe by taking consistent photos. The alarm name is stored in a database
Open Issues:	

Use Case Number:	20
Use Case Name:	***Emailing photos***
Participating Actors:	Main user

Goal:	Email a set of photos to the patient’s physician
Trigger:	This user chooses “email photos” option to email a set of chosen photos to his/her physician
Precondition:	There is a list of photos being displayed on the screen
Postcondition:	Successfully email the physician a set of chosen photos
Basic Flow	1/ User chooses “email photos” option 2/ System prompts user to choose multiple photos within the list 3/ User choose a list of photos 4/ User confirms his/her chosen photo set 5/ System displays pop-up window to ask for the user’s confirmation of sending photos via email 6/ Upon confirmation of the user, system prompts the user to enter the physician’s email 7/ System sends the chosen set of photos to the provided email 8/ System notifies the user of successful or failed email
Exceptions	
Qualities:	

Functionality:	
Constraints:	
Includes:	
Extends:	
Related Artifacts:	As a user, I want to email a set of photos to my physician
Notes:	Checking the network connection before sending photos via email to provide descriptive feedback
Open Issues:	

Use Case Number:	21

Use Case Name:	***Annotate images***
Participating Actors:	Main user
Goal:	Annotate images to provide more relevant information about the images
Trigger:	This user chooses “add annotation” option when the newly created photo is displayed on the screen right after it is taken
Precondition:	User takes a photo
Postcondition:	Successfully annotate the photo
Basic Flow	1/ User chooses “add annotation” option 2/ System prompts user to input some information 3/ User inputs an annotation and confirms his/her input 4/ System attaches the annotation to the photo
Exceptions	
Qualities:	

Functionality:	
Constraints:	
Includes:	
Extends:	
Related Artifacts:	As a user I would like to annotate images with annotations to describe the scenario that I made this photo
Notes:	The annotation is stored in the database. It will be loaded when the user chooses “show annotation” option when viewing the photo
Open Issues:	

Use Case Number:	22

Use Case Name:	***Tag photos***
Participating Actors:	Main user
Goal:	Tag the images to provide queries images by tag
Trigger:	The user chooses some existing tag to tag the photo to or create a new tag for the photo
Precondition:	User has just taken a photo or some tag is chosen
Postcondition:	Successfully tag the photo
Basic Flow	When a photo is just taken 1/ System loads the list of tags 2/ User chooses some existing tags to tag the photo 3/ User confirms his/her chosen tags 4/ System tags photo with chosen tags The target tag can be chosen from the list of existing tags 1/ User chooses “connect photos” option to tag a chosen photo to the current selected tag 2/ System loads a list of photos that are currently not attached to the tag 3/ User chooses some photo(s) to tag 4/ System tags the chosen photos

Exceptions	
Qualities:	
Functionality:	
Constraints:	
Includes:	
Extends:	
Related Artifacts:	As a user I would like to tag the images so I can query the image by tag
Notes:	
Open Issues:	