Updated 11/10/2020

MET CS682 TERM PROJECT PART 4

Detailed Design

Sneha

The purpose of this exercise is to give you practice specifying design details. It will help you see how the various UML models come together for use by developers.

The following characteristics apply:

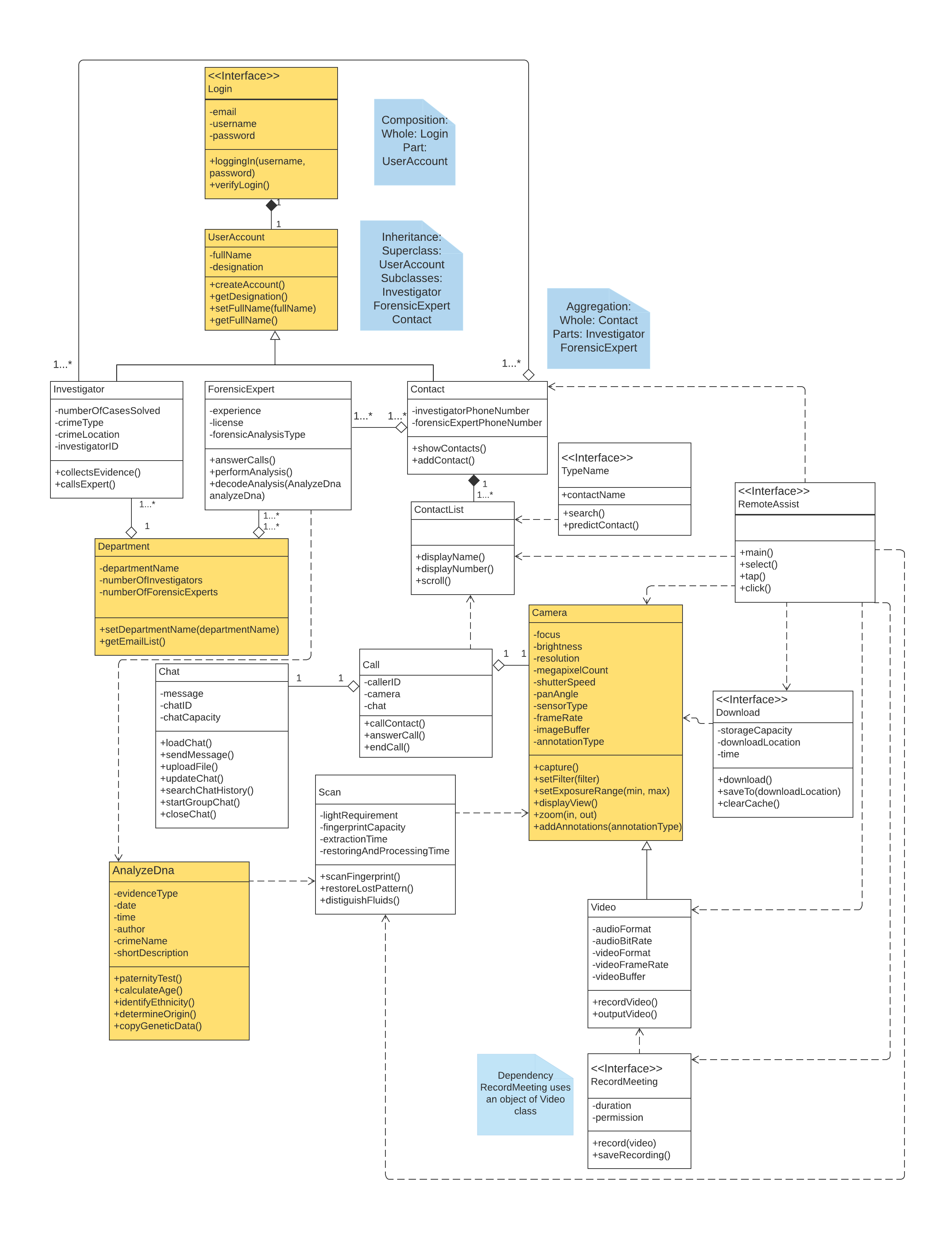
* Your scope from term project part 1. *RemoteAssist* has a potentially large scope but your solution should focus on an aspect or aspects of your choice.
* Your solution should focus on software-intensive aspects.

The last section below contains numerous hints.

# Updated Class Model

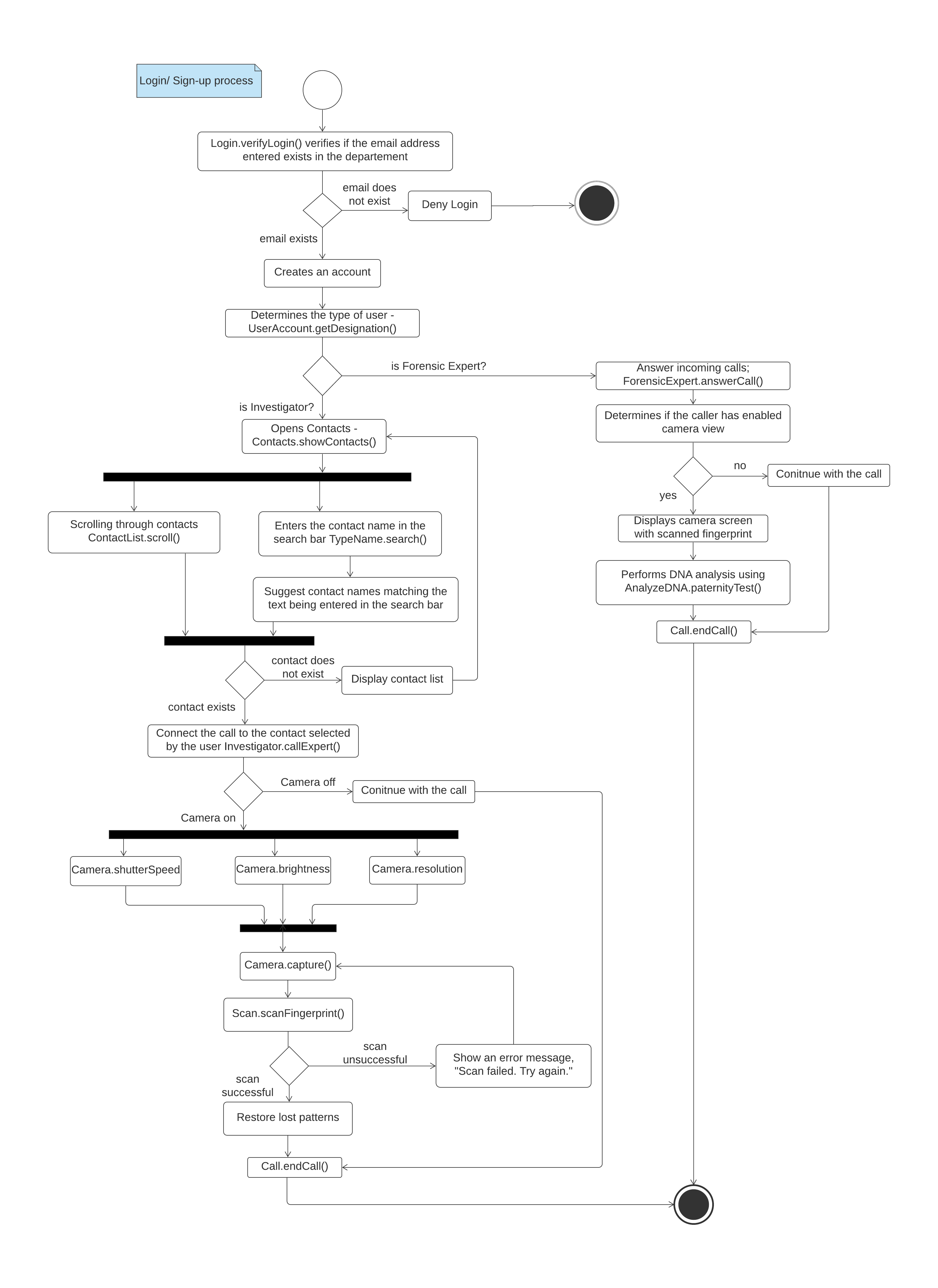
Update your class model to be part of a detailed design, including classes, illustrative attributes, and illustrative methods. Add classes, attributes, and methods appropriate for the activity diagram and pseudocode requested in the sections below. If additional classes are needed, try to limit them to two. It is not necessary to show every conceivable attribute and method, only those that you consider important in communicating your detailed design. Use colors to show the updated design elements.

The classes highlighted yellow in the following diagram have some methods/ attributes added or modified. (Also attached as a PDF)



# Activity Diagram

Select an appropriate method (function) which you introduced in your class model above and draw an activity diagram for it. The diagram may reference other functions, possibly from objects of other classes. Make sure that the latter appear in your class model but you do not need to elaborate on them beyond their (expressive!) names and possibly clarifying comments if necessary. **(Also attached as a PDF)**



# Pseudocode

Provide pseudocode for the method that you outlined in 6.2. This method may call on other functions, possibly methods from objects of other classes.

Your pseudocode, including comments, should not exceed three quarters of a page (using 12-point type). To help you develop the logic, you may want to use design techniques you are now familiar with (use cases, sequence diagrams, specific design goals, etc.).

# BEGIN

# INPUT: email\_address;

//allows only authorized email addresses to login to *RemoteAssist*

# IF (email\_address exists?){

# creates an account;

# determines the type of user;

# }

# ELSE{

# return deny login;

//Aborts operation

# }

ENDIF

//The flow splits here and operates differently for different types of users

IF(UserAccount.designation == "Investigator"){

call Contact.showContacts();

call TypeName.search();

call ContactList.scroll();

IF (contactname exists?){

call Contact.call();

}

ELSE{

System out("Contact does not exist");

}

ENDIF

IF(cameraON?){

//The camera is ON in this case and the attributes of the camera are automatically set to the following values.

Camera.resolution = 3300 \* 4200;

Camera.brightness = "medium";

Camera.shutterSpeed = 20";

call Camera.capture();

call Scan.scanFingerprint();

//checks if the fingerprint scan is successful

IF (scan successful?){

return restored fingerprint;

}

ELSE{

system out (“Scan failed. Try again!”);

}

ENDIF

}

ENDIF

ELSE{

/\*If the camera is off, the user will continue to stay on the call.\*/

Stay on call

}

call Call.endCall();

}

ENDIF

ELSE {

/\* Identifies user as forensic expert if the statement IF(UserAccount.designation == "Investigator")evaluates to FALSE.\*/

Answer call;

IF (cameraView enabled?){

Displays camera screen with scanned fingerprint;

performs DNA test;

}

ELSE{

stay on call;

}

ENDIF

call Call.endCall();

}

# Appendix

Since Term Project Part 4 builds on Term Project Part 1,2, and 3 Provide requirements and design you have outlined last week in Term Project Part 1. Not everything needs to be included, only the parts you feel are helpful in supporting the updated UML design. This section is not graded.

## First Use Case

|  |  |  |
| --- | --- | --- |
| **Use case Name** | Investigator (Caller) Use Case | |
| **Actor:** | Investigator | |
| **Description:** | RemoteAssist shall allow the investigator to contact a user in another department. | |
| **Pre-condition:** | The investigator logs into the application on a smart display device. RemoteAssist shall open the home screen displaying the menu. | |
| **Step #** | **Actor** | **System** |
| **1** | The investigator clicks on Contacts. | *RemoteAssist* displays a contact list. |
| **2** | The investigator starts typing the contact’s name in the search bar. | *RemoteAssist* starts searching the name and predicts the name, simultaneously. |
| **3** |  | *RemoteAssist* returns a list of contacts matching the name entered. |
| **4** | The investigator selects a contact and places a call. | *RemoteAssist* calls the selected contact. |
| **5** |  | *RemoteAssist* returns the call response. In this case, the recipient answers the call. |
| **6** | The investigator clicks on the camera icon. | *RemoteAssist* opens the camera and displays field view. |
| **7** | The investigator clicks on the icon that scans fingerprint. | *RemoteAssist* scans and processes the fingerprint and restores any lost patterns. |
| **8** | The investigator clicks on the video icon. | *RemoteAssist* switches to video mode. |
| **9** | The investigator clicks on the record button. | *RemoteAssist* records the video as a meeting. |
| **10** | The investigator speaks and clicks on the End Call button. | *RemoteAssist* ends the recording and disconnects from the call. |
| **Alternate Courses:** | #4 The investigator’s call is not answered— display message “Call Disconnected.”  #7 If *RemoteAssist* fails to restore lost patterns of a fingerprint, display “Unable to restore fingerprint patterns! Try again.” | |
| **Implementation Constraints:** | 1. The request for call must be managed within 3 seconds. 2. *RemoteAssist* shall have a high-resolution camera of 50 megapixels or higher. 3. *RemoteAssist* shall not take more than 10 seconds to process the fingerprint patterns. | |

## Second Use Case

|  |  |  |
| --- | --- | --- |
| **Use case Name** | Forensic Expert (Recipient) Use Case | |
| **Actor:** | Forensic Expert | |
| **Description:** | *RemoteAssist* shall allow forensic expert(s) to answer calls to solve problems in real-time, pull crucial evidence into view, and support collective problem-solving. | |
| **Pre-condition:** | The forensic expert will always be logged in to the application on a smart display device. The device (system) will display the menu on the home screen. | |
| **Step #** | **Actor** | **System** |
| **1** | The forensic expert receives a call. | *RemoteAssist* notifies the user by ringing. |
| **2** | The forensic expert answers a call. | *RemoteAssist* displays the call screen. |
| **3** |  | *RemoteAssist* displays a camera view showing the image of fingerprints captured by the investigator. |
| **4** |  | *RemoteAssist* displays the processed image of the fingerprints. |
| **5** | The forensic expert clicks on the download icon. | *RemoteAssist* makes a copy of the image by downloading it. |
| **6** | The forensic expert clicks on DNA test icon. | *RemoteAssist* starts analyzing fingerprints and returns DNA test results. |
| **7** | The forensic expert decodes the test results to the investigator by sending the details in the chat box. | *RemoteAssist* sends the message to the investigator and displays the message to both the users. |
| **8** |  | *RemoteAssist* notifies the expert the call is being recorded. |
| **9** | The forensic expert ends the call. | *RemoteAssist* ends the recording and disconnects from the call. |
| **Alternate Courses:** | #2 The forensic expert may not answer the call.  #5 If the DNA test does not return any data—display “Match not found” | |
| Implementation Constraints: | 1. The system shall not take more than 15 seconds to provide the DNA test results. 2. Messages shall not take more than 2 seconds to deliver. | |

# References

[1] Module 6: Object-Oriented Designs. (n.d.). Retrieved from https://onlinecampus.bu.edu/bbcswebdav/pid-8604526-dt-content-rid-50582650\_1/courses/21sprgmetcs682\_o1/course/module06/allpages.htm.

[2] Dennis, A., Wixom, B. H., & Roth, R. M. (2015). Systems Analysis & Design. Wiley.

# Evaluation



**Please do not include Hints section in your solution.**

# Hints

## Overall Assignment Notes and Grading Criteria

* You may use Visio, LucidChart, or another design tool of your choice (please check with your facilitator in advance if not using Visio or LucidChart).
* The module notes are a primary source for explanations and examples; we also encourage you to do outside reading and research to gain additional perspective.
* **Clarity:** Are diagrams clear to read? (e.g., avoid overlapping lines, non-polished designs).
* **Relevance**: Make sure to consider the right scope for the activity diagram and pseudocode. This is not system or use case level such as one you put together previosuly. This is specific to a single method which you have identified as important in your design to expand on.
* **Relevance**: We develop activity diagrams and pseudocode to help us with important and complex functionality. Make sure to choose a significant method for the activity diagram/pseudocode. Significant means something important to the application you are designing and your design (i.e. something you have been focusing on all along, or something that you have expanded on in this assignment).
* **Thoroughness and Coverage:**
  + Refer the parts back to design goals at every opportunity. For example, if you add a class, make sure it does not contradict goals of design you have selected previously.
  + Consider what to place in the Appendix. For example, it may be helpful to add a sequence diagram to complement your activity diagram and pseudocode—make sure to choose appropriate scope.
* **Consistency and Clarity:** Is your design consistent? For example, all function calls and attributes listed in pseudocode are provided in class model.

## Class Model

* **Clarity:** Is it clear how your added (if applicable) classes/methods/attributes relate to the existing model? Add explanatory notes if not obvious. For example, do these new additions appear in the pseudocode?
* **Technical Soundness:** At this point, cohesion, coupling, and encapsulation choices really matter. Make sure the right attributes and methods belong to the right class, explain what may not be obvious.
* Additional hints and notes on approaching class modeling are covered in Assignment 4.

## Activity Diagram

* A good place to start is to review the “Activity Diagrams” section in Module 4.
* Activity diagrams are discussed on pages 129-138 in the text.
* It may be helpful to develop/update a use case for the function you decide to work with, which you are welcome to include in the Appendices section.
* **Clarity**: Make sure your diagram is clearly labeled, e.g., are your decision points clear?
* **Technical Soundness:** Don't confuse activity diagrams with state diagrams—you can show parallel processes and decision choices. They are not the same as data flow diagrams either, because activity diagrams indicate the flow of control rather than of data.
* **Thoroughness and Coverage:** Does your activity diagram cover all reasonable branches?  This is a good opportunity to add robustness.

## Pseudocode

* Module 6 provides two ways to approach writing Pseudocode within the “Specifying Methods and Algorithms in Detail” section.
* It may be helpful to develop a use case or sequence diagram for the function outlined in part 2, which you are welcome to include in the Appendices section.
* **Clarity:** 
  + Pseudocode is flexible; if you introduce something that is not obvious (i.e. syntax that is Java-like, Python-like, SQL-like) provide an explanation (i.e. "Note to reader "++" means increment variable by 1")
  + Use colors and/or indentations to distinguish various parts of your pseudocode.
* **Technical Soundness:** Review and understand loops, If/Then, and case statements, and know when to use one over another.
* **Depth and Coverage:** Practice both verbal and arithmetic expressions in your pseudocode.