Updated 11/10/2020

MET CS682 TERM PROJECT PART 3

Design Goals and Data Flows

Sneha

The purpose of this exercise is to give you practice performing system design. It also gives you additional practice with UML diagrams.

You are to continue to specify UML design for *RemoteAssist* that assists employees in their remote work.

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.

# Packages

In at most one and a half single-space pages (using 12-point type) not including appendices, apply design goals to provide packages, showing which classes belong to which package. Explain how your packages fulfill applicable design goals—or how they could be enhanced to do so. Your design goals should focus on Sufficiency, Flexibility, and Reusability, explaining cohesion and coupling in your design and their relationship with design goals. Packaging choices require trade-offs. Explain these as appropriate.

1. **Package name**: Camera

**Design goals for this package**: The Camera package includes six classes: Camera, Video, Scan, Recording, Download, and AnalyzeDna. The Camera package is very vital to the system and its design goals are stable with designs existing in *RemoteAssist*. The classes in the package are **highly** **cohesive** as all the classes and their methods are single-minded. There exists **moderate** **coupling** as the package shares an aggregation relationship with the Contact package for the camera module to work.

The Camera class encourages **reusability** by sharing an inheritance relationship with its subclass Video. The ways in which reusability can be achieved for this design is listed below.

* *Inheritance between classes*: extending the properties of a parent class (Camera) in the child class (Video).
* *Object code*: Sharing libraries between the camera and the scanner.

The Camera package offers high **flexibility** as well. Aspects of flexibility include:

* *Allows new functionalities to be added*: Add *delete* function to the existing *capture* function.
* *Changing functionality*: allow *recordVideo* to create an *hyperlapse* video.

**Design tradeoffs for this package**: *Time vs. Cost* - The Camera package will have to compromise on the cost of the development to save the time taken for the development of software. Even if the company has its own software development unit, or subcontracts the project, the cost of development is relative to the development time. The project will be slamming throughout as overruns arise, with the budget and the time. To overcome the timeline risk, the company shall invest in a third-party software. The company shall reduce the time it takes to develop a software program by buying a third-party software, thereby increasing the cost of development.

1. **Package name**: Contact

**Design goals for this package**: The Contact package contains the classes that address the main functionality of the system. The Contact package includes four classes: Contact, ContactList, Call, and Chat. There exists **high** **cohesion** as the related code is grouped together as classes in the package. The **coupling** is **low** as the dependency between the classes in the package is weak.

The Contact package encompasses of classes that define what the design intends to do. Thereby, the package supports ***sufficiency*** by managing the requirements for the design. The design implementation for the Contact package is ***robust***. Robustness provides the design with the power to deal with appropriate and erroneous inputs. The following scenarios briefly explain robustness of a design.

* A user enters a name in the search bar for contacts. The contact the user is searching for does not exist in the directory so the design will display the message “Contact does not exist.”
* A user enters incorrect inputs in the search bar multiple times. The incorrect inputs can be special characters, symbols, or junk characters. The design shall be robust enough to deal with such errors without crashing.

The Contact package is highly **reusable**. The answerCall() and endCall() methods and phoneNumber attributes of the Contact class can be adopted by other classes.

**Design tradeoffs for this package**: *Rich features vs. simple features* – Adding rich features to the design to contend with the company’s competitors is not always the best choice. If the design focuses too much on introducing more features, it burdens the users to learn and be up to date with the new features. This can even lead to lessening the satisfaction of the users. It is ideal for the design of this package to factor in simplicity over richness. Hence, this is a crucial design trade-off to consider.

1. **Package name**: Department

**Design goals for this package**: The Department package encompasses of three classes: UserAccount, Investigator, and ForensicExpert. The Investigator class and the ForensicExpert class contain information about the main users for the system being developed. The Investigator class contains attributes and methods that define an investigator and their actions. The ForensicExpert class contains attributes and methods that define a forensic expert and their actions.

The Department package is **highly cohesive** and **loosely coupled** as classes do not know much about each other’s inner functionality. This package is highly **flexible** so shall allow different types of investigators (fraud investigator and surveillance investigator) and different components of forensic investigation (cyber forensics and accounting forensics) to be added without modifying the current design. The Department package is **sufficient** as it contains the qualifying users and their departments required to use the system.

**Design tradeoffs for this package**: *Usability vs. Security* - This package shall focus more on security than on usability. The system shall prioritize security by adding additional safety measures so only the authorized users are able to login and use the system. Ease of use is crucial when considering design as well, but security is very essential for the system.

1. **Package name**: Task

**Design goals for this package**: The Task package contains two classes: Agenda and Journal. This package has **high cohesion** as the classes are composed with one responsibility. The methods and objects in the Agenda class belong to the class itself. The Journal class cannot use the objects or the methods of the Agenda class and vice versa. This means the Task package is **loosely coupled**.

**Design tradeoffs for this package**: Java vs. Python – If the back-end processing module uses Python, then it is best to develop the system in the same language. Choosing a language that best suits the design is an efficient way to complete a project.

1. **Package name**: Notify

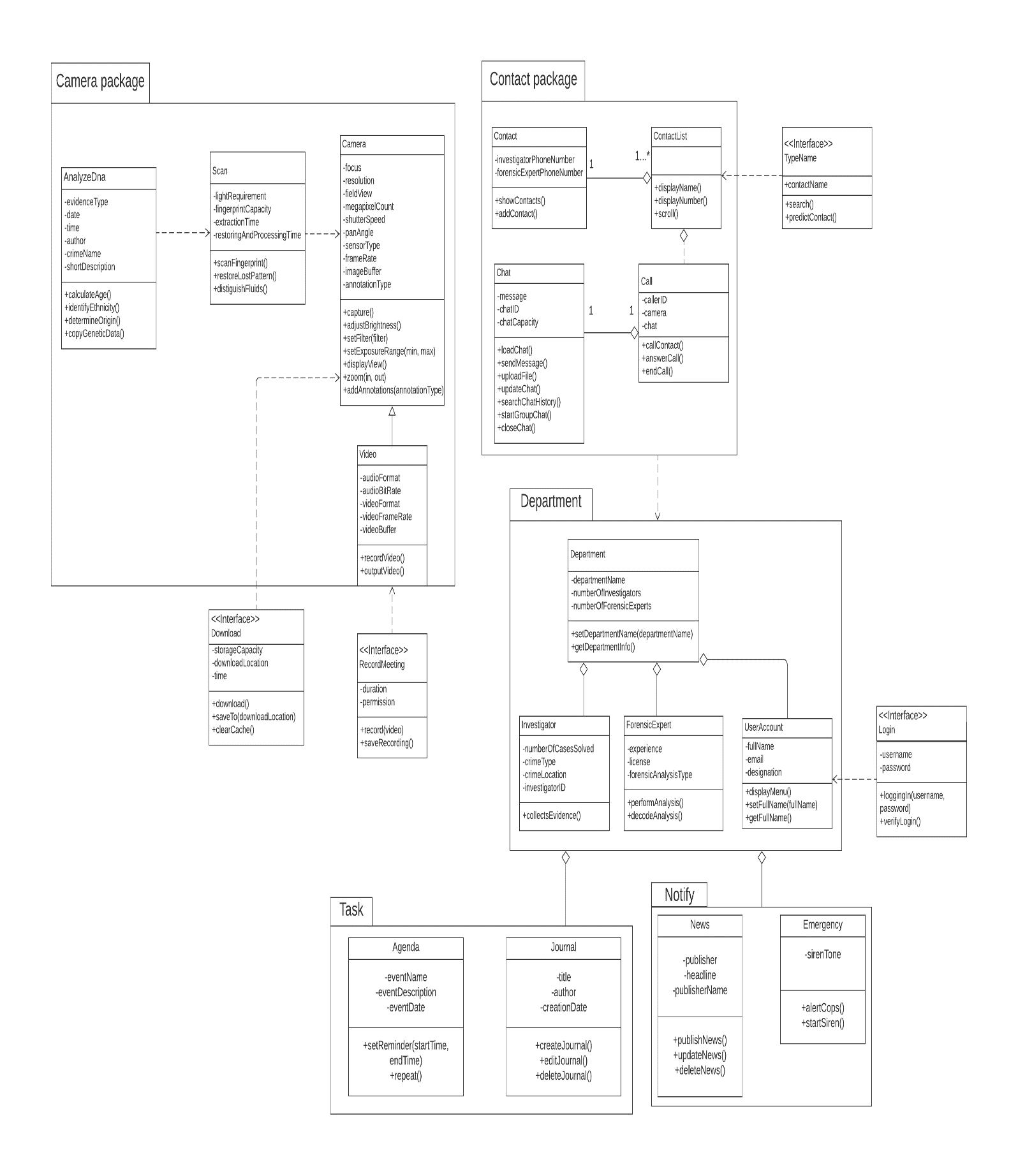
**Design goals for this package**: The Notify package contains two classes: News and Emergency. The classes, their objects, and their methods are totally independent. The News class calls only the methods (updateNews(), deleteNews()) within itself. The Emergency class calls only the methods (alertCops(), startSiren()) within itself. This means the package maintains **low coupling** and **high cohesion**.

**Design tradeoffs for this package**: Swiftness vs. configuration – It is important to determine how swiftly the features work over how they can be configured. How soon will the cops be alerted when a user hits the emergency feature? This is a safety-critical trade off, so it is worth prioritizing the speed and minimizing emphasis on configurations for this package.

# Class Model

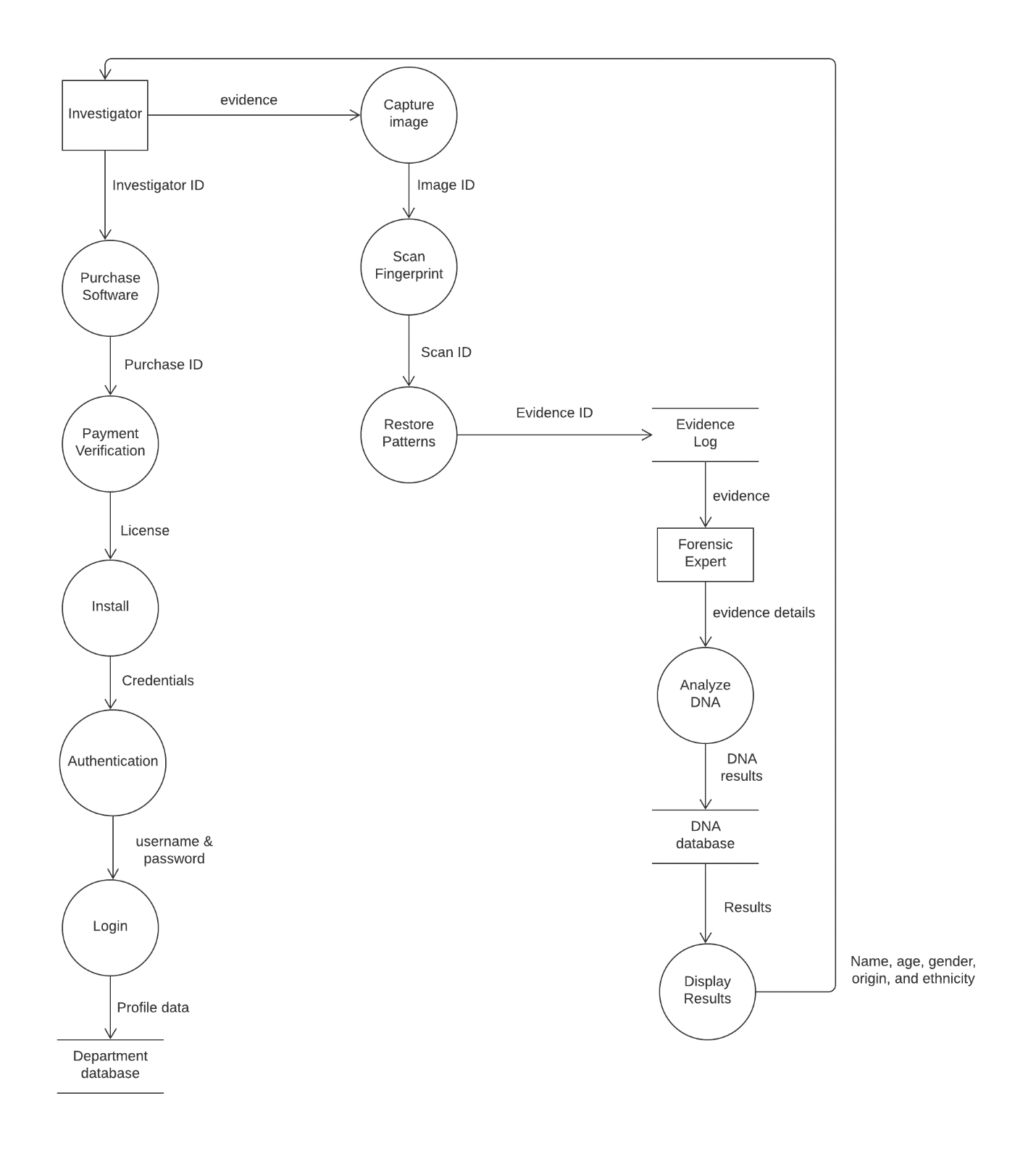
Update your class model to reflect the design goals you have cited above. Incorporate the packages in the model. Consider including attributes and methods to add clarity to the diagram. Label everything appropriately and clearly. Explain the parts of your class diagram that are not obvious. Use colors to show the updated design elements.

(Also attached as PDF)



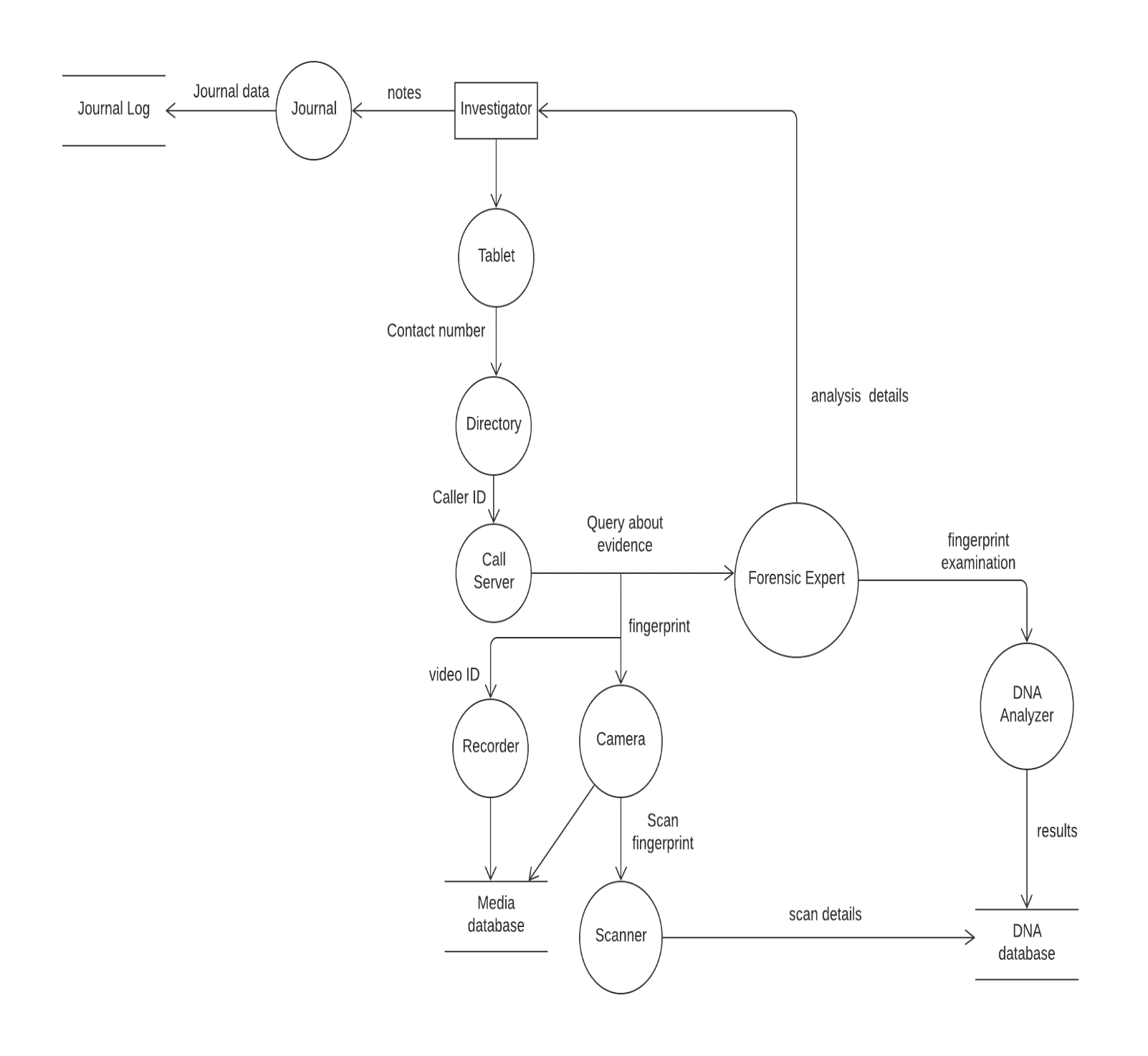
# Logical Data Flow Diagram

Draw a logical data flow diagram for your design. Explain the parts of the diagram that are not obvious. Your diagram should have roughly 8-10 logical processing elements. Make sure to identify scope for your design. Explain the parts of your logical data flow diagram that are not obvious.



# Physical Data Flow Diagram

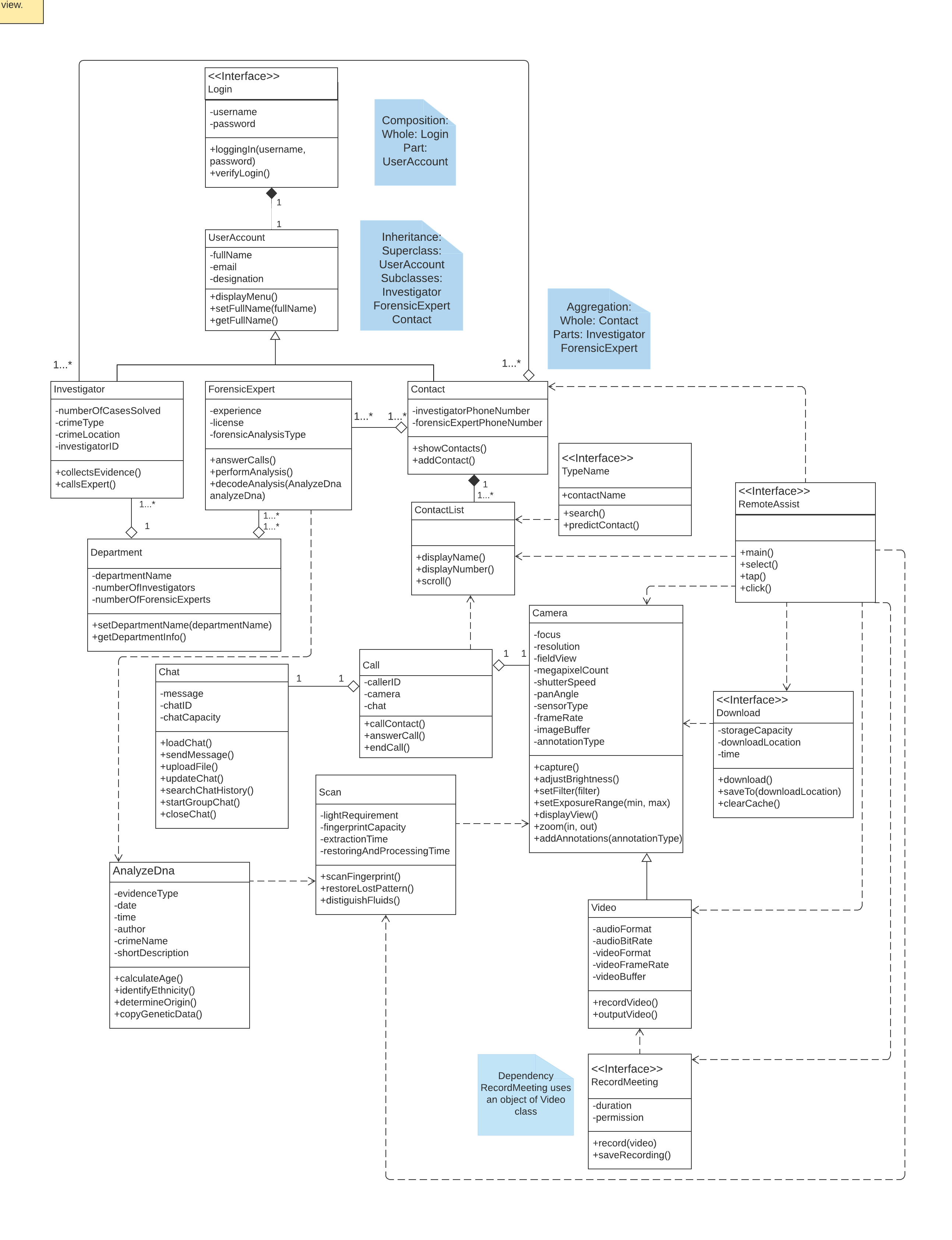
Draw a physical data flow diagram for your design. Explain the parts of the diagram that are not obvious. Your diagram should have roughly 3-5 physical processing elements. Make sure to identify scope for your design. Explain the parts of your physical data flow diagram that are not obvious.



# Appendix

Since Term Project Part 3 builds on Term Project Part 1 & 2, 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.

Previous Class Model:



**Functional Requirements:**

1. *RemoteAssist* shall allow users to login with their departmental credentials.
2. *RemoteAssist* shall allow the Investigator to contact other user(s) in real-time by chat, audio, or video.
3. *RemoteAssist* shall have a camera that will incorporate most basic settings of modern cameras.

* Panorama mode
* 3600 view
* Live Focus and Selective focus
* Slow Motion
* Hyper-lapse
* Self-timer and more

1. *RemoteAssist* shall have a camera feature with the following advanced image processing techniques.

* Capture and restore fingerprint patterns.
* Provide 3D images of shoe/ footprints.
* Describe hair strands accurately by indicating the color of the strand.
* Provide filters to set out unique wavelengths which projects substances from their environments. (The substances can be blood, water, alcohol, chemicals, etc.)

1. *RemoteAssist* shall allow the forensics expert to zoom in and out of the video during the call.
2. *RemoteAssist* shall allow both the investigator and the forensics expert to record a call.
3. *RemoteAssist* shall allow both the investigator and the forensics expert to send messages, images, documents, or videos in the chat.
4. *RemoteAssist* shall allow both the investigator and the forensics expert to download and screenshot images.
5. *RemoteAssist* shall save the downloaded images and videos and the call recordings in Files.

# References

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

[2] Module 5: System Architecture. (n.d.). Retrieved from https://onlinecampus.bu.edu/bbcswebdav/pid-8604588-dt-content-rid-50582586\_1/courses/21sprgmetcs682\_o1/course/module05/allpages.htm

[3] Otto, Kevin N., and Erik K. Antonsson. “Trade-off Strategies in Engineering Design.” Research in Engineering Design, vol. 3, no. 2, 1991, pp. 87–103., doi:10.1007/bf01581342.

[4] “Interactions.” Exploring Design Trade-Offs for Quality of Life in Human-Centered Design | ACM Interactions, interactions.acm.org/archive/view/january-february-2018/exploring-design-trade-offs-for-quality-of-life-in-human-centered-design.

[5] “Types and Components of Data Flow Diagram (DFD).” GeeksforGeeks, 22 Apr. 2020, [www.geeksforgeeks.org/types-and-components-of-data-flow-diagram-dfd/](http://www.geeksforgeeks.org/types-and-components-of-data-flow-diagram-dfd/).

[6] Rudyy, Eugene. “Tradeoffs in Product Design.” Medium, Muzli - Design Inspiration, 27 Mar. 2019, medium.muz.li/tradeoffs-in-product-design-e2ad6d71a465.

# Evaluation



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

# Hints

## Overall Assignment Notes

* 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.
* **Thoroughness and Coverage:** Accompany each requested section with a brief description that explains your thinking and the choices that you made.
* **Consistency and Clarity:** Make sure that you have a consistent design throughout the solution. You will want to iteratively review all four sections to make sure that together they are consistent as a single solution.
* **Clarity:** Are diagrams clear to read? (e.g., avoid overlapping lines, non-polished designs).
* **Relevance**:
  + Is the scope of your design relevant to the solution? For example, choosing the data flow scope, classes, and methods that are secondary (for example, authentication might be important – but with limited scope of your assignment it may not be the focus).
  + This is the usual editing process. It is tempting to repeat important points but this often weakens your work and can appear to the reader like padding.

## Design Goals

* Your focus should be the beginning sections of the Module 5 Notes “The Goals of System Architectures and Designs”
* **Technical Soundness:**
  + Solution shows understanding in context of goals of system design (i.e., a thorough discussion of how sufficiency is satisfied in the design, with links to the solution’s class model and links back to the requirements).
  + Do you show distinction and understanding between cohesion and coupling of classes and packages within your design to support the selected design goals?
* **Thoroughness and Coverage:**
  + Make sure to explain how the selection of your packages specifically supports the goals of system design for this scenario, avoid being generic.
  + Are tradeoffs between goals of design covered?
  + Is cohesion/coupling covered within the context of goals of system design?
* **Clarity:** Are tradeoffs between cohesion and coupling clearly explained? For example, mentioning high cohesion is not enough - a clear link between the design needs to be made, and explained (e.g., classes and relationships in the class model show cohesion in package).

## Class Model

* **Clarity**: Did you add a new class or relationship – is it aggregation or composition? Look to explain your updated design and show changes in a different color.
* Other hints and notes on approaching class modeling are covered in Assignment 4.

## Logical and Physical Data Flow Diagrams

* Physical data flow diagrams are described in the Module 5 Notes within the “Data Flow Diagrams” section.
* **Technical Soundness:** 
  + Make sure to understand the difference between logical and physical data flow diagrams
  + Is the focus on the physical processing elements and not on the logical processing elements in the physical data flow?
  + Are all the data flows correct (e.g., external actor not sending data flow directly to data store, but through a processing element, data flows appropriately labeled with data and not a process (action).
  + No “dead-ends” in processing elements. Each processing element should have an input and an output.
* **Clarity:** Explain the scope of your data flow diagrams.