K. J. Somaiya College of Engineering, Mumbai-77

(Autonomous College Affiliated to University of Mumbai)

Batch: B3 Roll No.: 1811111, 1811112, 1811113
Experiment / assignment / tutorial No. 4
Grade: AA / AB / BB / BC / CC / CD /DD
Signature of the Staff In-charge with date

Title: To develop UML diagrams for selected project

Aim: To learn and understand the way of creating various UML diagrams for requirement analysis.

CO: Analyze the software requirements and model the defined problem with the help of UML Diagrams

---

Books/ Journals/ Websites referred:

1. Roger Pressman, “Software Engineering”, sixth edition, Tata McGraw Hill.
2. System Analysis & Design by Satzinger, Jackson and Burd, Cengage Learning, 2007
3. System Analysis and Design Methods by Jeffery l. Whitten, Lonnie D Bentley,McGraw Hill, 7th edition.
4. System Analysis and Design by Alan Dennis, Barbara H. Wixom, Roberta M. Roth,Wiley India 4th edition
5. http://en.wikipedia.org/wiki/Software_requirements_specification
6. http://en.wikipedia.org/wiki/Use_case

Pre Lab/ Prior Concepts:

In software and systems engineering, a **use case** is a list of steps, typically defining interactions between a role (known in Unified Modeling Language (UML) as an "actor") and a system, to achieve a goal. The actor can be a human or an external system.

In systems engineering, use cases are used at a higher level than within software engineering, often representing missions or stakeholder goals. The detailed requirements may then be captured in Systems Modeling Language (SysML) or as contractual statements.

As an important requirement technique, use cases have been widely used in modern software engineering over the last two decades. Use case driven development is a key characteristic of process models and frameworks like Unified Process (UP), Rational Unified Process (RUP), Oracle Unified Method (OUM), etc. With its iterative and evolutionary nature, use cases are also a good fit for agile development.

A sequence diagram is a graphical view of a scenario that shows object interaction in a time-based sequence of what happens first, what happens next. Sequence diagrams establish the roles of objects and help provide essential information to determine class responsibilities and interfaces.

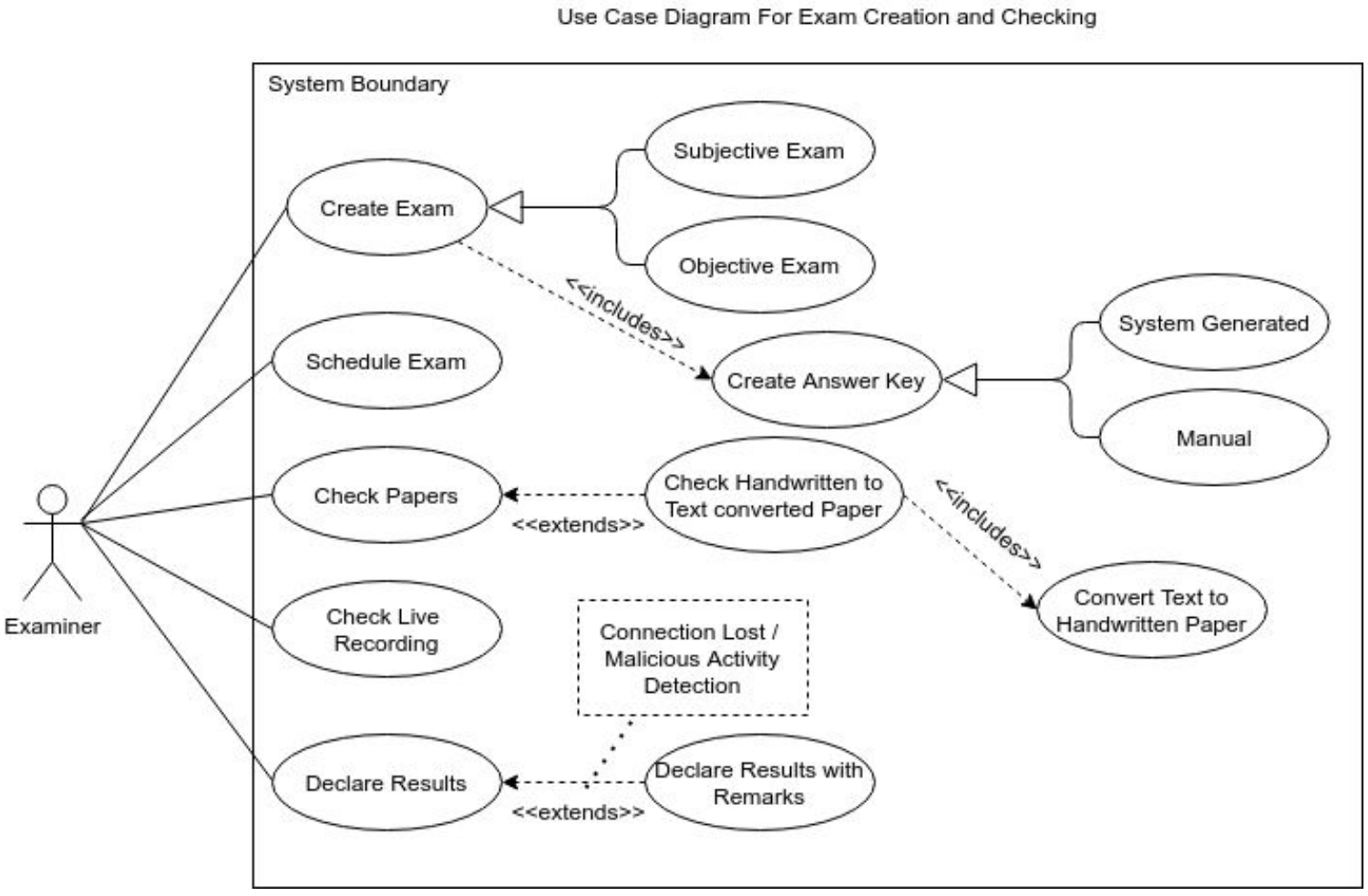
In UML, class diagrams are one of six types of structural diagram. Class diagrams are fundamental to the object modelling process and model the static structure of a system. Depending on the complexity of a system, you can use a single class diagram to model an entire system, or you can use several class diagrams to model the components of a system.
Class diagrams are the blueprints of your system or subsystem. You can use class diagrams to model the objects that make up the system, to display the relationships between the objects, and to describe what those objects do and the services that they provide.

In its basic form, an activity diagram is a simple and intuitive illustration of what happens in a workflow, what activities can be done in parallel, and whether there are alternative paths through the workflow. Activity diagrams as defined in the Unified Modeling Language are derived from various techniques to visually illustrate workflows. Activity diagrams are used to visualize the workflow of a business use case. A complete workflow description will have a basic flow, and one or several alternative flows. This workflow has a structure that we can define textually, using informal if, if-then-else, or does-until statements of various kinds. For a simple workflow with a simple structure such textual definitions may be quite sufficient, but in the case of more complex structures, activity diagrams help to clarify and make more apparent what the workflow is. Historically, activity diagramming techniques have mostly been used in the business process modeling domain, but this article will also briefly discuss how you can use it in the system modeling domain.

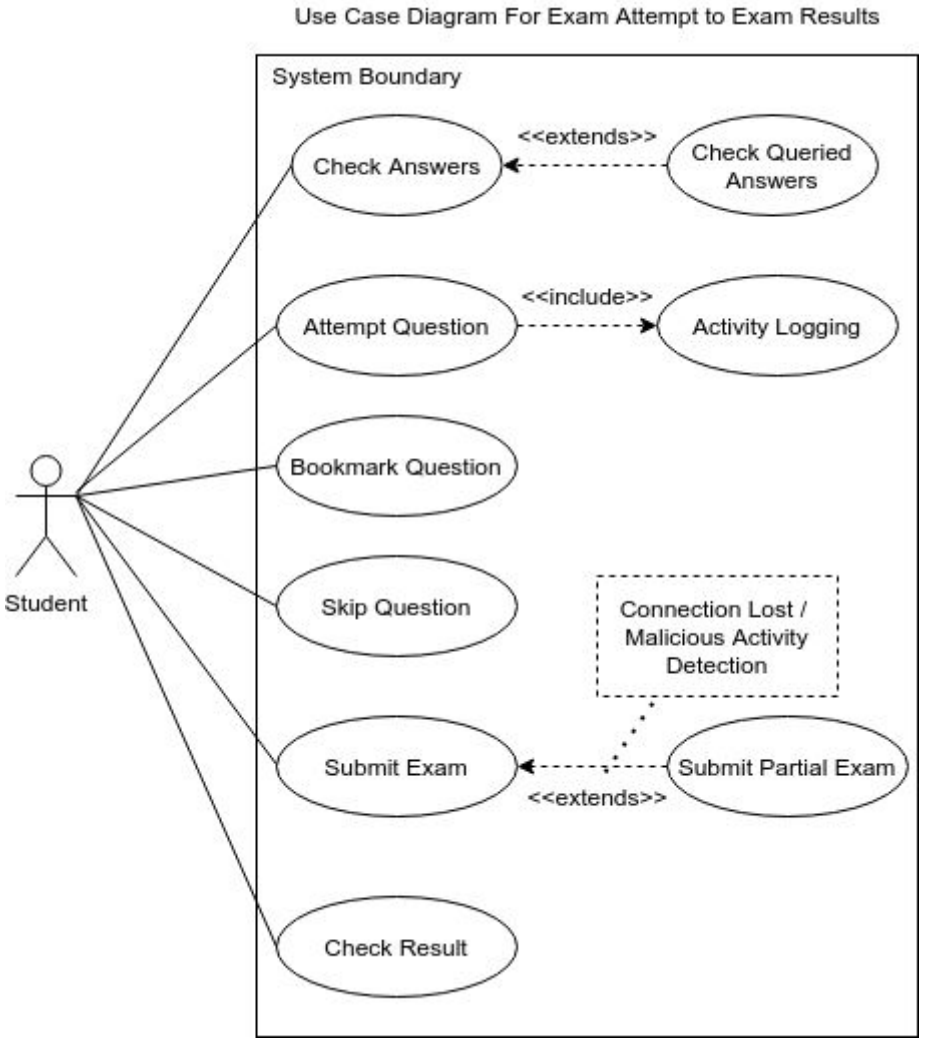
Requirement Modeling:
Significance of every diagram is to be written

1. **Use Case**

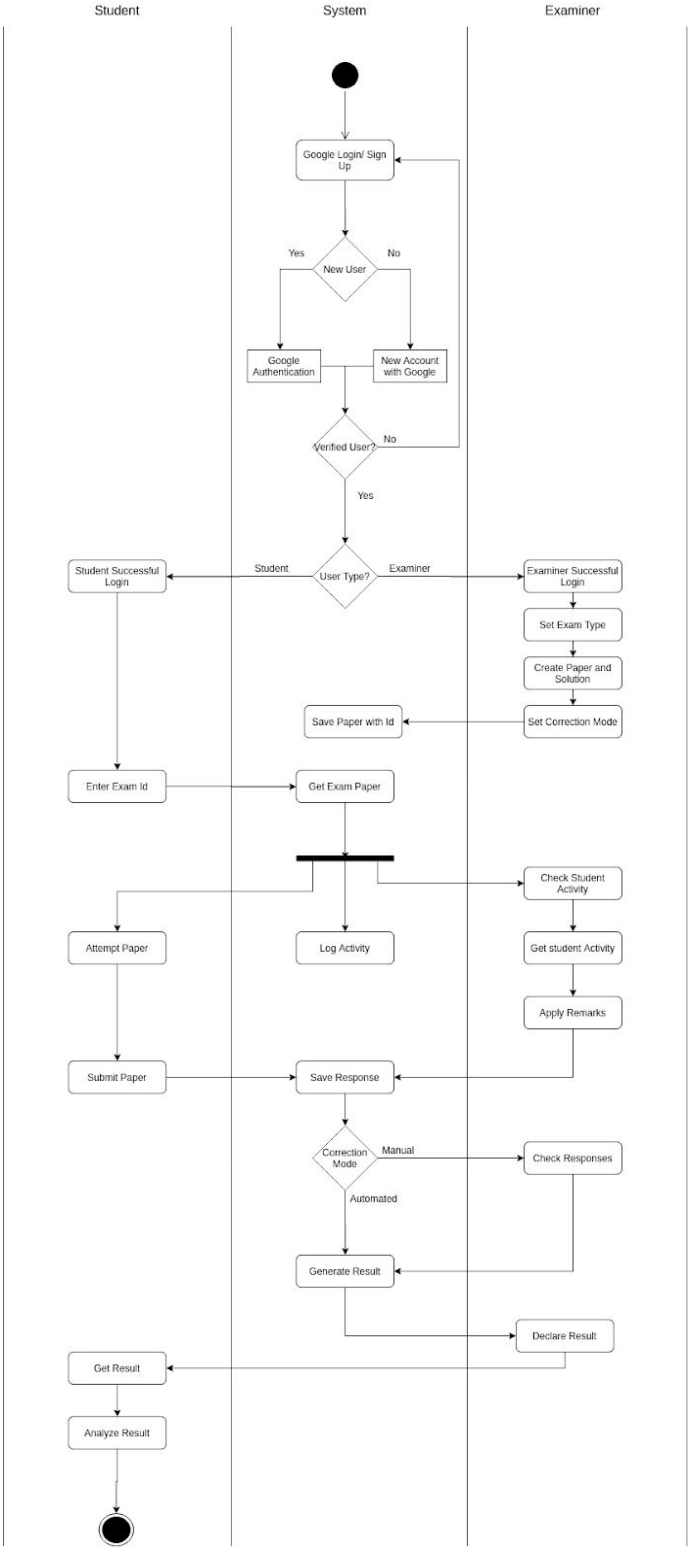
   Use case diagram specifies users point of view of the system, The below diagram shows the examiner's interface with the system and the actions performed by the system in the examiner's perspective.



Use case diagram specifies users point of view of the system, The below diagram shows the student's interface with the system and the actions performed by the system in the examiner's perspective.

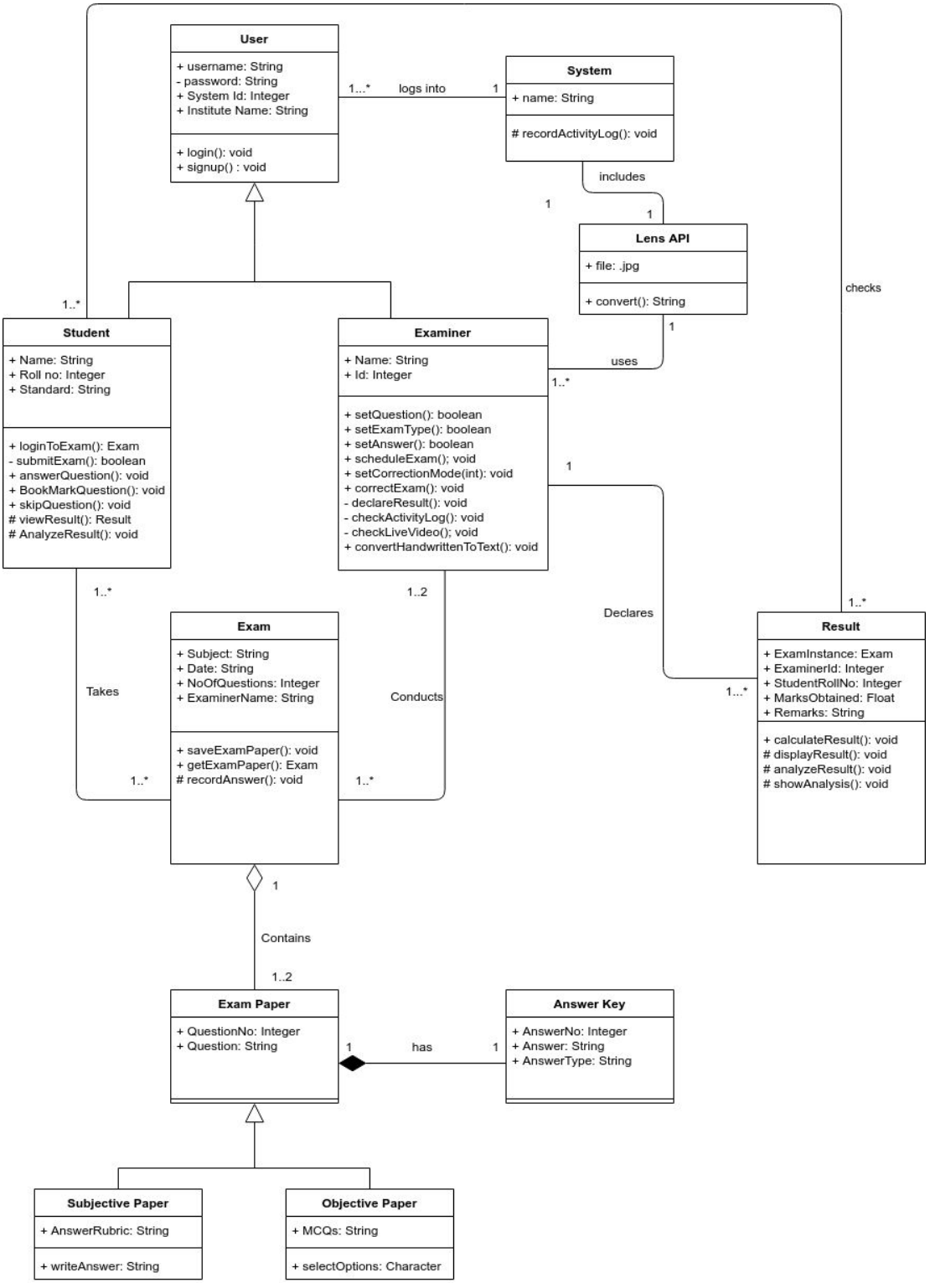


2. Activity Diagram

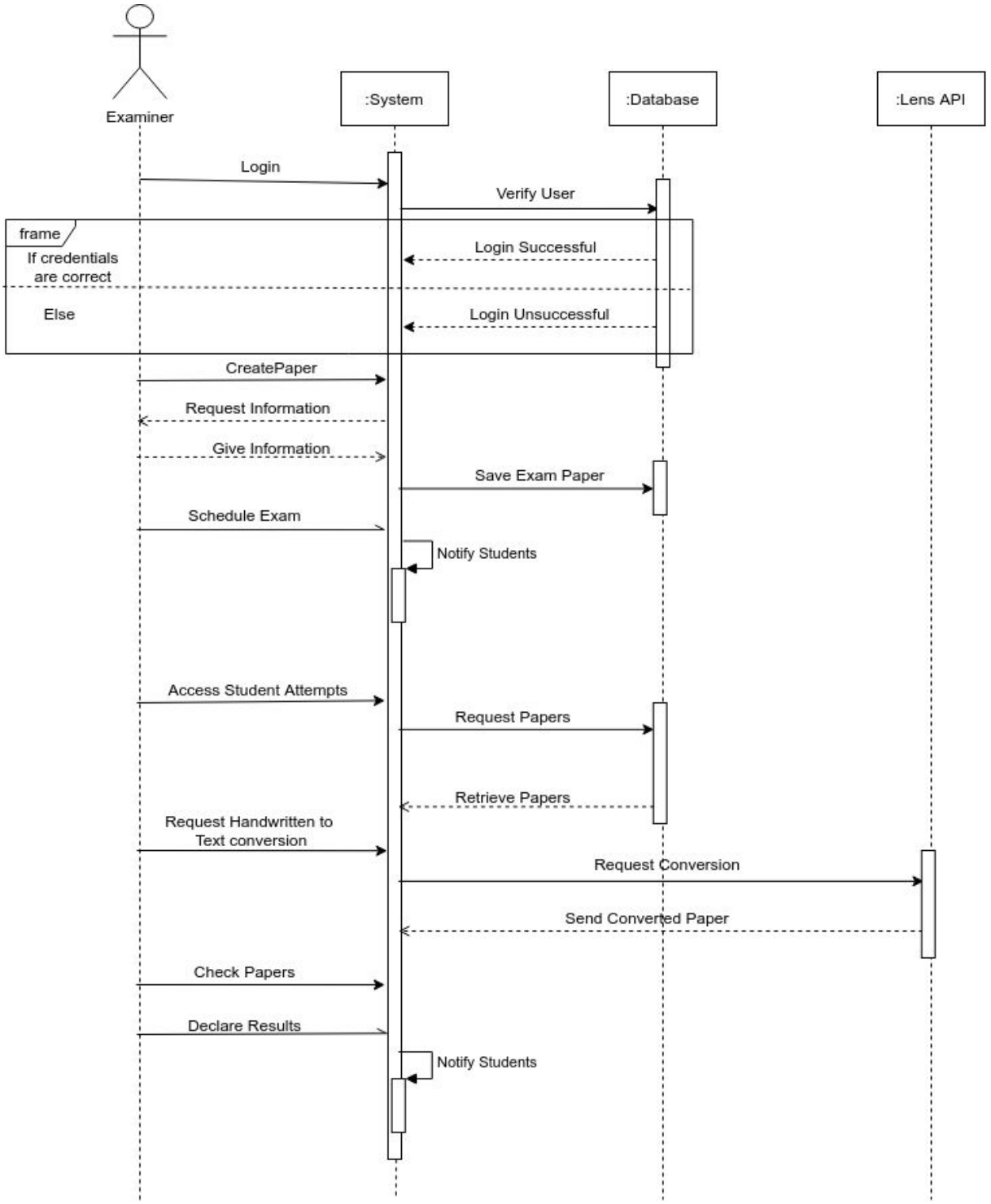


The activity diagram is drawn for the whole activity of the conduction of an online examination in online mode it shows the actions performed by student teacher and examiner.

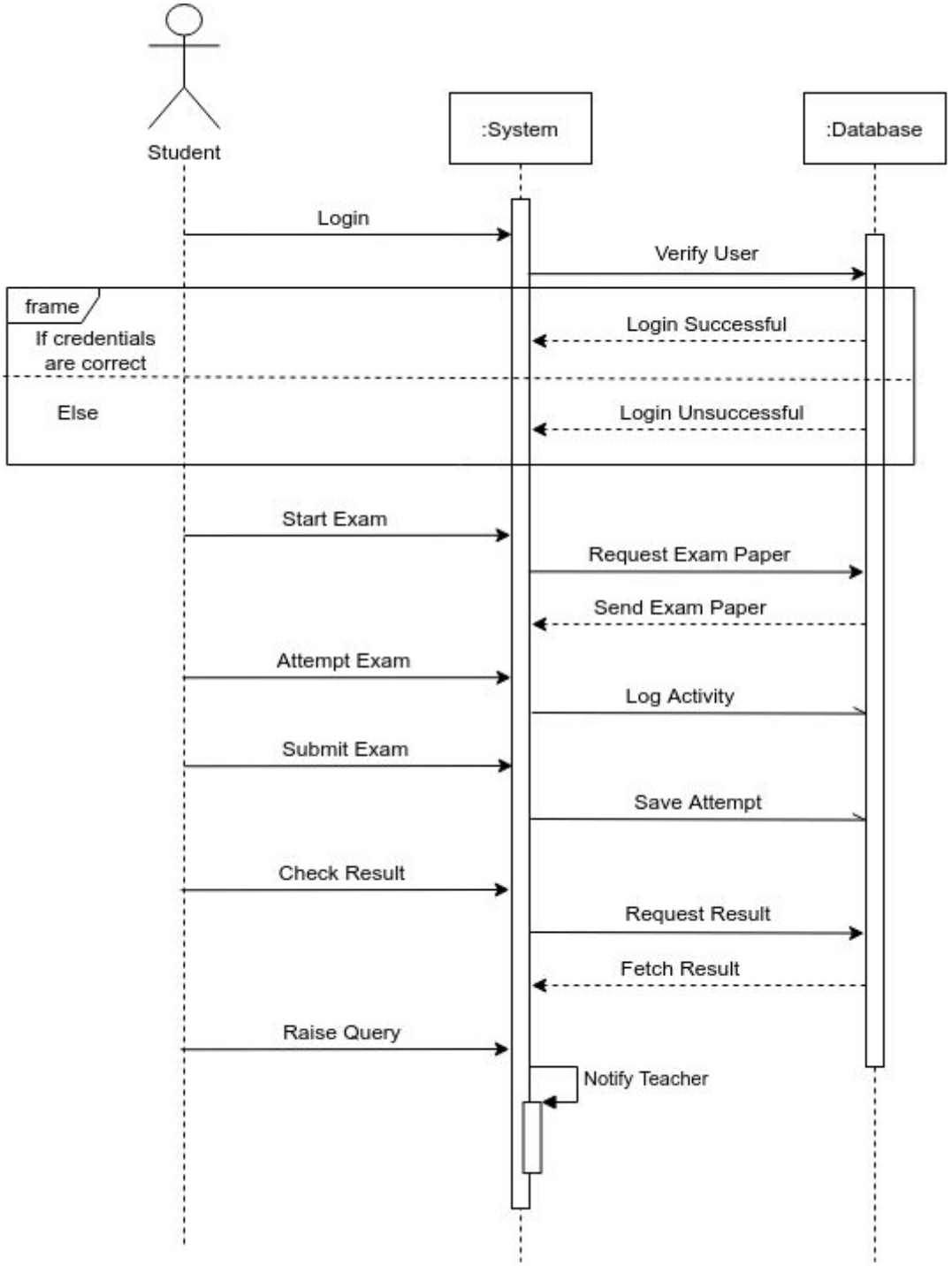
3. **Class Diagram:** Class diagram specifies the static view of the system and shows interaction between different models of the system. Following diagram shows interaction between different classes and their respective functions



4. **Sequence:** Sequence Diagram is an interaction diagram that describes how and in what order a group of objects work together. It helps to model the logic of a system or its operations based on time and messages.
   The following sequence diagram logically models the actions undertaken by the examiner while using the Advanced Online Examination System.



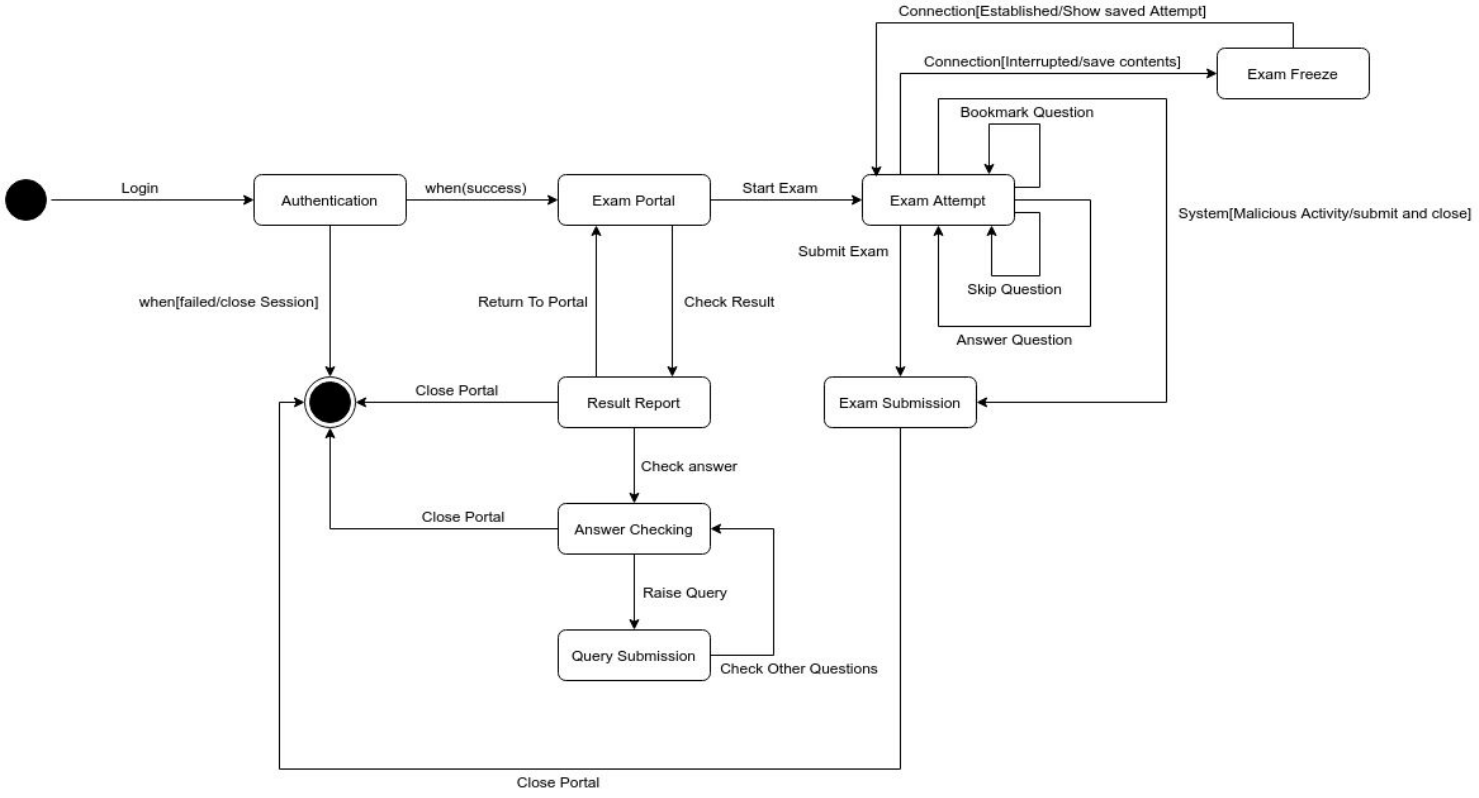
The following sequence diagram logically models the actions undertaken by the student while using the Advanced Online Examination System.



5. State Chart Diagram:

State Chart Diagram specifies how the state of an object changes while using a software application. The following diagram shows the different states and transitions of the student object.

State Chart Diagram For Object Student



Conclusion:

Thus the software requirements were analyzed

Post Lab Descriptive Questions:

1. Where do use cases fit in the software development life cycle?

   Ans.

   A use case is a methodology used in system analysis to identify, clarify and organize system requirements. The use case is made up of a set of possible sequences of interactions between systems and users in a particular environment and related to a particular goal. The method creates a document that describes all the steps taken by a user to complete an activity. Use cases are typically written by business analysts and can be employed during several stages of software development, such as planning system requirements, validating design, testing software and creating an outline for online help and user manuals. A use case document can help the development team identify and understand where errors may occur during a transaction so they can resolve them. Every use case contains three essential elements:

   - The actor: This can be a single person, a group of people interacting with the process or the system.
   - The goal: The final successful outcome that completes the process.
   - The system: The process and steps taken to reach the end goal, including the necessary functional requirements and their anticipated behaviors.

   A single use case can benefit developers by revealing how a system should behave while also helping identify any errors that could arise in the process. Other benefits of use case development include:

   - The list of goals created in the use case writing process can be used to establish the complexity and cost of the system.
   - By focusing both on the user and the system, real system needs can be identified earlier in the design process.
   - Since use cases are written primarily in a narrative language they are easily understood by stakeholders, including customers, users and executives not just by developers and testers.
   - The creation of extending use cases and the identification of exceptions to successful use case scenarios saves developers time by making it easier to define subtle system requirements.
   - By identifying system boundaries in the design scope of the use case, developers can avoid scope creep.
   - Premature design can be avoided by focusing on what the system should do rather than how it should do it.

In addition, use cases can be easily transformed into test cases by mapping the common course and alternate courses and gathering test data for each of the scenarios. These functional test cases will help the development team ensure all functional requirements of the system are included in the test plan. Furthermore, use cases can be used in various other areas of software development, including project planning, user documentation and test case definitions. Use cases can also be used as a planning tool for iterative development.

2. Compare sequence diagram with collaboration diagram. Explain pros and cons of each.
   Ans:

Sequence Diagrams	Collaboration Diagrams
1. The sequence diagram represents the UML, which is used to visualize the sequence of calls in a system that is used to perform a specific functionality.	1. The collaboration diagram also comes under the UML representation which is used to visualize the organization of the objects and their interaction.
2. The sequence diagrams are used to represent the sequence of messages that are flowing from one object to another.	2. The collaboration diagrams are used to represent the structural organization of the system and the messages that are sent and received.
3. The sequence diagram is used when the time sequence is the main focus.	3. The collaboration diagram is used when object organization is the main focus.
4. The sequence diagrams are better suited to analyse activities.	4. The collaboration diagrams are better suited for depicting simpler interactions of the smaller number of objects.

Advantages of Sequence Diagrams:

- Sequence diagrams are used to explore any real application or a system.
- Sequence diagrams are used to represent message flow from one object to another object.
- Sequence diagrams are easier to maintain.
- Sequence diagrams are easier to generate.

- Sequence diagrams can be easily updated according to the changes within a system.
- Sequence diagram allows reverse as well as forward engineering.

Disadvantages of Sequence Diagrams:

- Sequence diagrams can become complex when too many lifelines are involved in the system.
- If the order of message sequence is changed, then incorrect results are produced.
- Each sequence needs to be represented using different message notation, which can be a little complex.
- The type of message decides the type of sequence inside the diagram.

Advantages of Collaboration Diagrams:

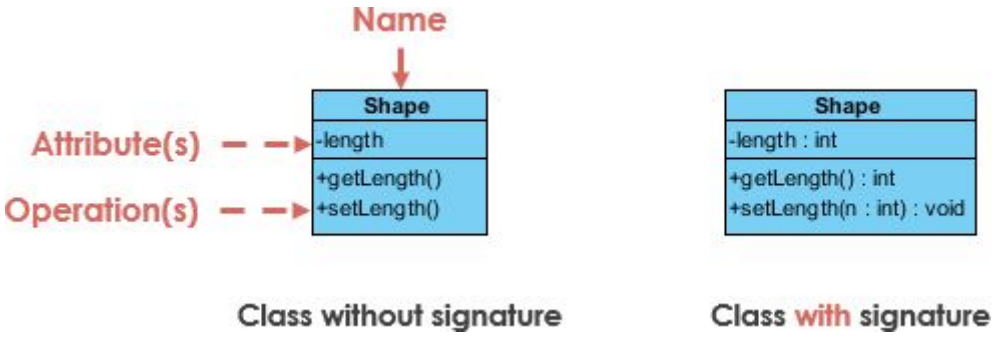
- It emphasizes the structural aspects of an interaction diagram - how lifeline connects.
- Its syntax is similar to that of a sequence diagram except that liflines don't have tails.
- Object diagrams are special cases of communication diagrams.
- It allows you to focus on the elements rather than focusing on the message flow as described in the sequence diagram.
- It gives flexibility to add new objects in two dimensions.

Disadvantages of Collaboration Diagrams:

- Collaboration diagrams can become complex when too many objects are present within the system.
- It is hard to explore each object inside the system.
- Collaboration diagrams are time consuming.
- The object is destroyed after the termination of a program.
- The state of an object changes momentarily, which makes it difficult to keep track of every single change that occurs within an object of a system.

3. List different notations used in Class diagrams with example.

   Ans:



Class Name:

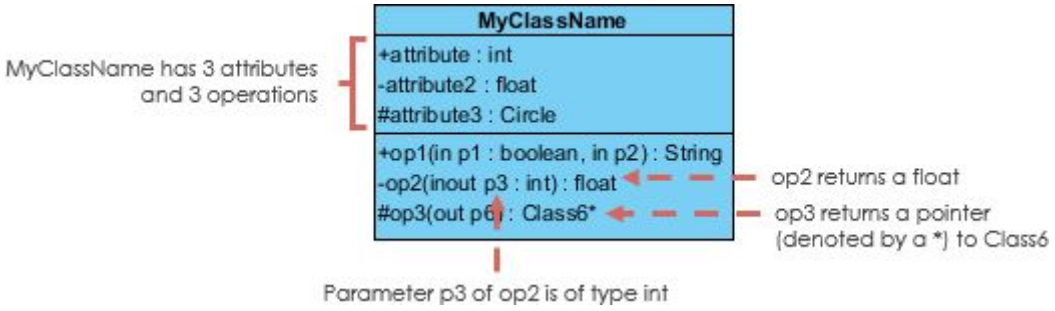
The name of the class appears in the first partition.

Class Attributes:

- Attributes are shown in the second partition.
- The attribute type is shown after the colon.
- Attributes map onto member variables (data members) in code.

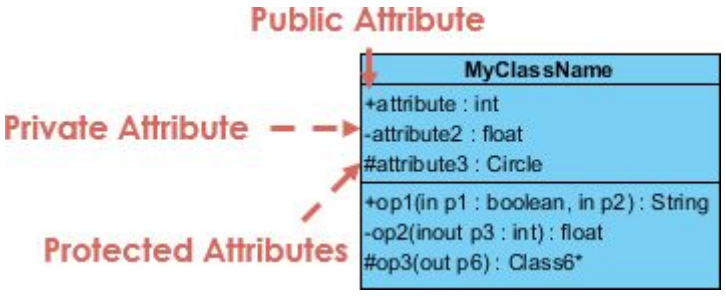
Class Operations (Methods):

- Operations are shown in the third partition. They are services the class provides.
- The return type of a method is shown after the colon at the end of the method signature.
- The return type of method parameters are shown after the colon following the parameter name. Operations map onto class methods in code.



Class Visibility:

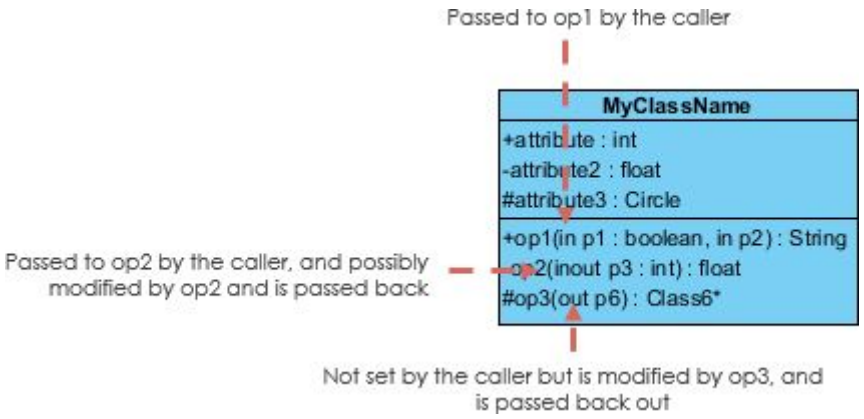
The +, - and # symbols before an attribute and operation name in a class denote the visibility of the attribute and operation.



- + denotes public attributes or operations
- - denotes private attributes or operations
- # denotes protected attributes or operations
- ~ denotes package

Parameter Directionality:

Each parameter in an operation (method) may be denoted as in, out or inout which specifies its direction with respect to the caller. This directionality is shown before the parameter name.



Relationship between Classes:

Classes are interrelated to each other in specific ways. In particular, relationships in class diagrams include different types of logical connections.