Batch: A1	**Roll No.: 16010123012**
Experiment No.: 4	
Grade: AA / AB / BB / BC / CC / CD /DD	
Signature of the Staff In-charge with date	

Title: Study of Umbrello Unified Modelling Language tool Or Lucidchart: flowcharts and diagram drawing tool.

Aim: To learn and understand the way of creating various UML diagrams for requirement analysis. Prepare Use case, Class diagram and ER diagram for the miniproject.

CO: Analyse 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 case is 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.

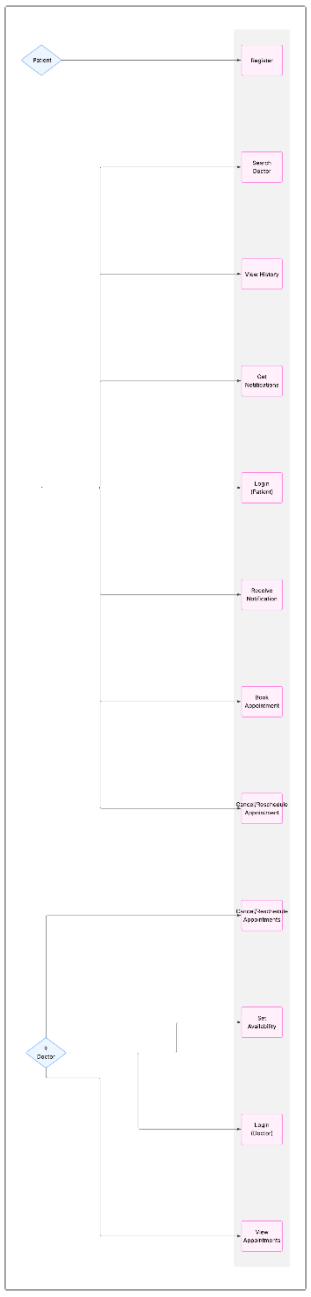
In UML, class diagrams are one of six types of structural diagram. Class diagrams are fundamental to the object modelling process and model the static structure of a system. Depending on the complexity of a system, you can use a single class diagram to model an entire system, or you can use several class 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:

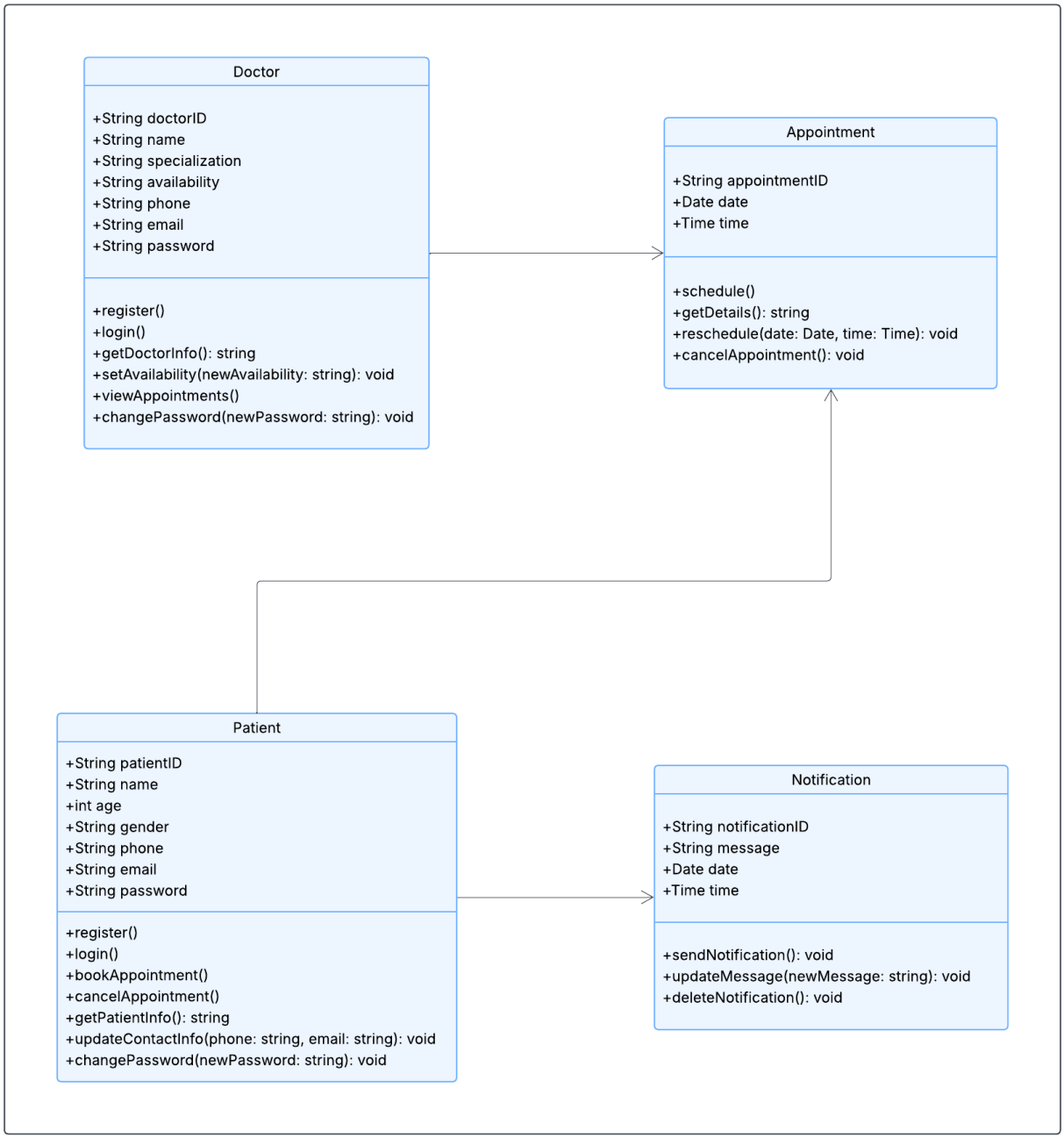
1. **Use Case Diagram**

The Use Case Diagram highlights the functional requirements of MediSlot by showing how different users (patients, doctors) interact with the system. It helps capture the main services provided by the application, such as booking appointments and setting availability. This diagram is particularly significant in understanding the system from the user's perspective and ensuring all required functionalities are covered during development.



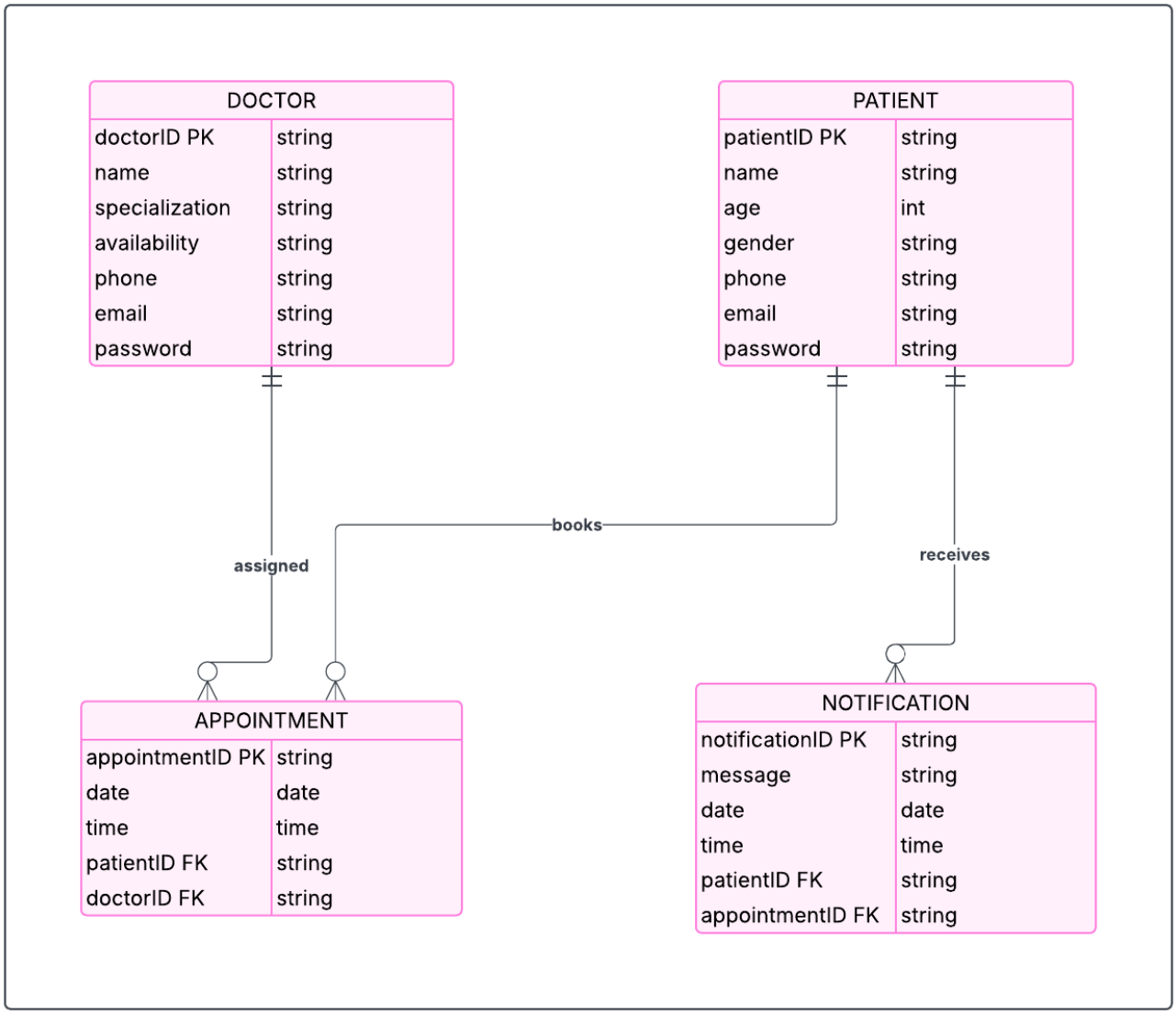
2. Class Diagram

The Class Diagram provides a blueprint of the system's structure by defining classes, their attributes, operations, and relationships. In MediSlot, it shows entities like Patient, Doctor, Appointment, Notification, and their interactions. This diagram is significant as it helps in object-oriented design, ensuring code reusability, maintainability, and clarity of the system's internal logic. It also serves as a bridge between requirements and actual coding implementation.

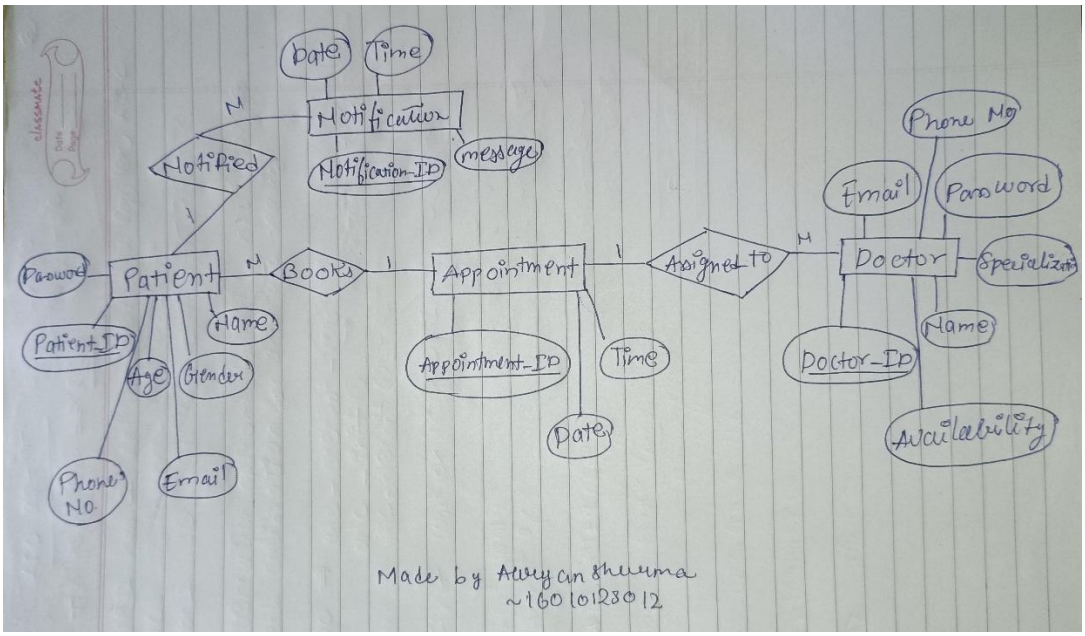


3. **ER Diagram**

The ER (Entity Relationship) Diagram models the database design of MediSlot. It represents entities such as Patient, Doctor, Appointment and Notification, and their relationships (e.g., Patient books Appointment, Doctor manages Appointment). This diagram is significant because it helps in designing a normalized, efficient, and consistent database schema, ensuring that the system can handle large volumes of data and provide accurate results.



Lab Work:



Conclusion:

In this experiment, we created Use Case, Class, and ER diagrams for the MediSlot system using Lucidchart. The Use Case diagram captured functional requirements, the Class diagram provided the structural design, and the ER diagram represented the database schema. Together, these diagrams helped us understand the system's functionality, structure, and data flow, forming a clear bridge between requirements and implementation.

Post Lab Descriptive Questions:

1. **Where do use cases fit in the software development life cycle?**
   Use cases fit in the requirement analysis phase of the SDLC. They are used to capture functional requirements and define interactions between users and the system. They also guide design, testing, and validation.

2. **List different notations used in Class diagram with example**
   - Class: Represented as a rectangle divided into three compartments (name, attributes, methods).
   - Association: Line connecting classes (e.g., Student — borrows → Book).
   - Multiplicity: Numbers at line ends (1, 0..1, 1..*, etc.).

- Inheritance (Generalization): Line with a hollow triangle pointing to the parent class.
- Aggregation: Line with a hollow diamond at the parent side.
- Composition: Line with a filled diamond at the parent side.
- Dependency: Dashed arrow showing one class depends on another.