Vidyavardhini's College of Engineering & Technology

Department of Computer Engineering

Academic Year : 2024-25

Experiment No. 3
Create UML Diagrams for the given case study
Name: Krisha Chikka
Std/Div: TE/1 Roll no. : 30
Date of Performance: 22/07/2024
Date of Submission: 29/07/2024

Aim: To create UML diagrams for the given case study

Objective: To design Use case and Activity diagram for Currency Detector for the Visually Impaired

Theory:

Unified Modeling Language (UML) is a general-purpose modeling language. The main aim of UML is to define a standard way to **visualize** the way a system has been designed. It is quite similar to blueprints used in other fields of engineering. UML is **not a programming language** , it is rather a visual language.

UML helps in specifying, visualizing, constructing, and documenting the artifacts of software systems. We use UML diagrams to portray the behavior and structure of a system.

UML helps software engineers, businessmen, and system architects with modeling, design, and analysis.

The Object Management Group (OMG) adopted Unified Modelling Language as a standard in 1997. It's been managed by OMG ever since.

The International Organization for Standardization (ISO) published UML as an approved standard in 2005. UML has been revised over the years and is reviewed periodically.

Need of UML

- Complex applications need collaboration and planning from multiple teams and hence require a clear and concise way to communicate amongst them.
- Businessmen do not understand code. So UML becomes essential to communicate with non-programmers about essential requirements, functionalities, and processes of the system.
- A lot of time is saved down the line when teams can visualize processes, user interactions, and the static structure of the system.

Types of UML Diagram

1. *Class Diagram*: It is the building block of all object oriented software systems. We use class diagrams to depict the static structure of a system by showing system's classes, their methods and attributes. Class diagrams also help us identify relationship between different classes or objects.
2. *Instance Diagram*: An Object Diagram can be referred to as a screenshot of the instances in a system and the relationship that exists between them. Since object

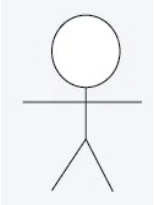
diagrams depict behaviour when objects have been instantiated, we are able to study the behaviour of the system at a particular instant.

3. *Component Diagram*: Component diagrams are used to represent how the physical components in a system have been organized. We use them for modelling implementation details.
4. *Deployment Diagram*: Deployment Diagrams are used to represent system hardware and its software.It tells us what hardware components exist and what software components run on them.
5. *State Diagram*: A state diagram is used to represent the condition of the system or part of the system at finite instances of time. It's a behavioral diagram and it represents the behavior using finite state transitions.
6. *Activity Diagram*: Activity Diagrams to illustrate the flow of control in a system. We can also use an activity diagram to refer to the steps involved in the execution of a use case.
7. *Use Case Diagram*: Use Case Diagrams are used to depict the functionality of a system or a part of a system. They are widely used to illustrate the functional requirements of the system and its interaction with external agents (actors).
8. *Sequence Diagram*: A sequence diagram simply depicts interaction between objects in a sequential order i.e. the order in which these interactions take place.
9. *Collaboration Diagram*: It shows sequenced messages exchange between objects

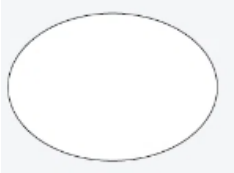
Use Case Diagram Notations

UML notations provide a visual language that enables software developers, designers, and other stakeholders to communicate and document system designs, architectures, and behaviors in a consistent and understandable manner. Following are the elements of the Use case diagram.

a. *Actor:* Actors are external entities that interacts with the system. These can include users, other systems, or hardware devices. Below is the symbol for actor



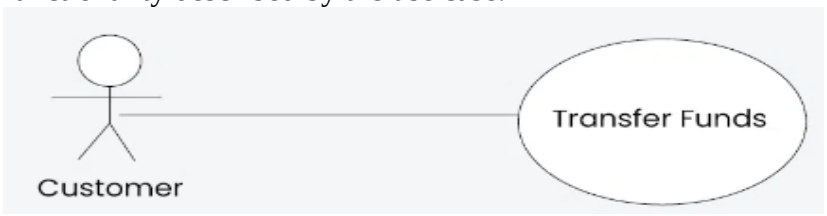
b. *Use Case:* Use case represents the functionality of the system. Below is the symbol for the Use case.



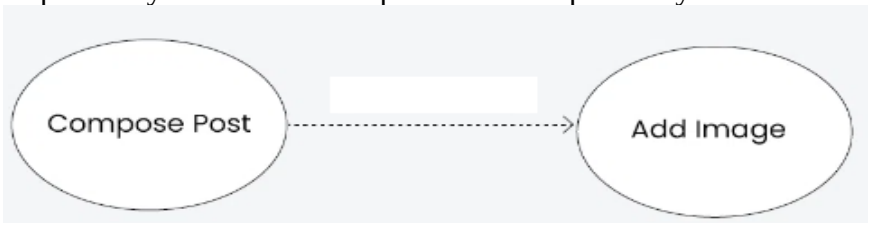
c. *Relationship:* Relationships play a crucial role in depicting the interactions between actors and use cases.

1. Association: The Association Relationship represents a communication or interaction between an actor and a use case. It is depicted by a line connecting the actor to the use case. This relationship signifies that the actor is involved in the functionality described by the use case.

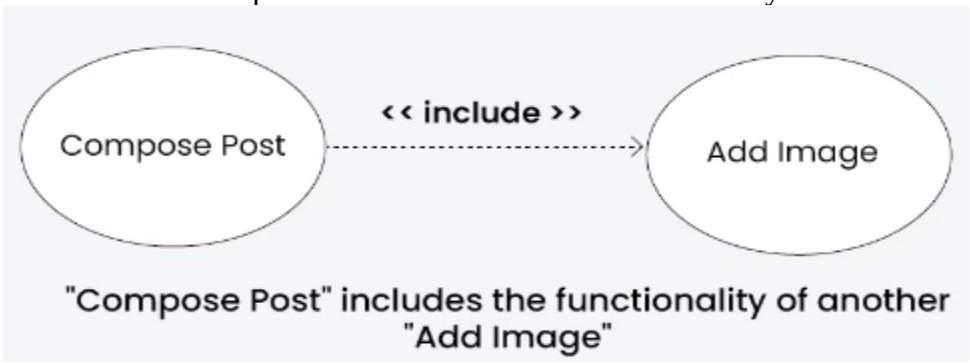
2.



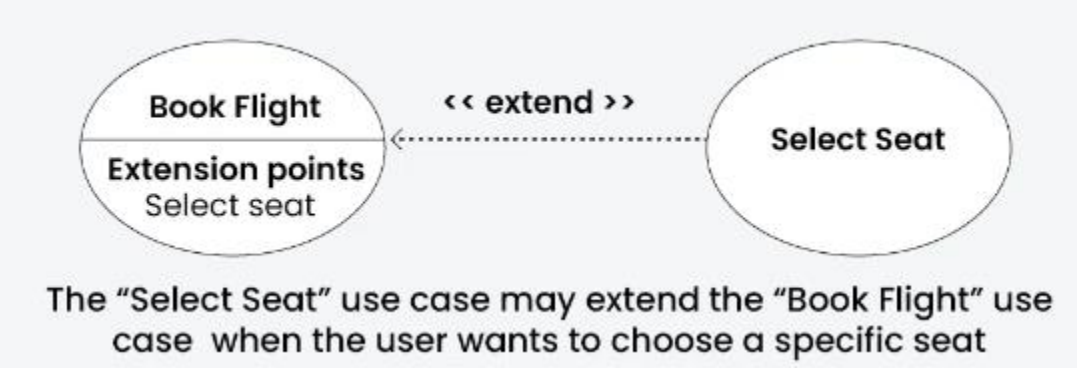
3. Dependency: This relationship shows the dependency between the two use cases



Include Relationship : One use case include the functionality of another use case



Extend Relationship: The Extend Relationship illustrates that a use case can be extended by another use case under specific conditions.



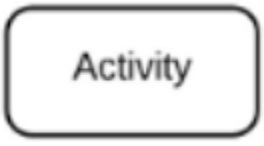
d. *System Boundary:* The system boundary is a visual representation of the scope or limits of the system you are modeling. It defines what is inside the system and what is outside. The boundary helps to establish a clear distinction between the elements that are part of the system and those that are external to it.

Activity Diagram Notations

Some of the most common components of an activity diagram include:

- *Action:*

A step in the activity wherein the users or software perform a given task. In Lucidchart, actions are symbolized with round-edged rectangles.'



Activity symbol

- *Decision node:*

A conditional branch in the flow that is represented by a diamond. It includes a single input and two or more outputs.

Decision symbol

- *Control flows:*

Another name for the connectors that show the flow between steps in the diagram.

Connector symbol

- *Start node:*

Symbolizes the beginning of the activity. The start node is represented by a black circle.

Start symbol

- *End node:*

Represents the final step in the activity. The end node is represented by an outlined black circle.

End symbol

- *Fork: Splits a single activity into flow into two concurrent activities.*

Fork symbol

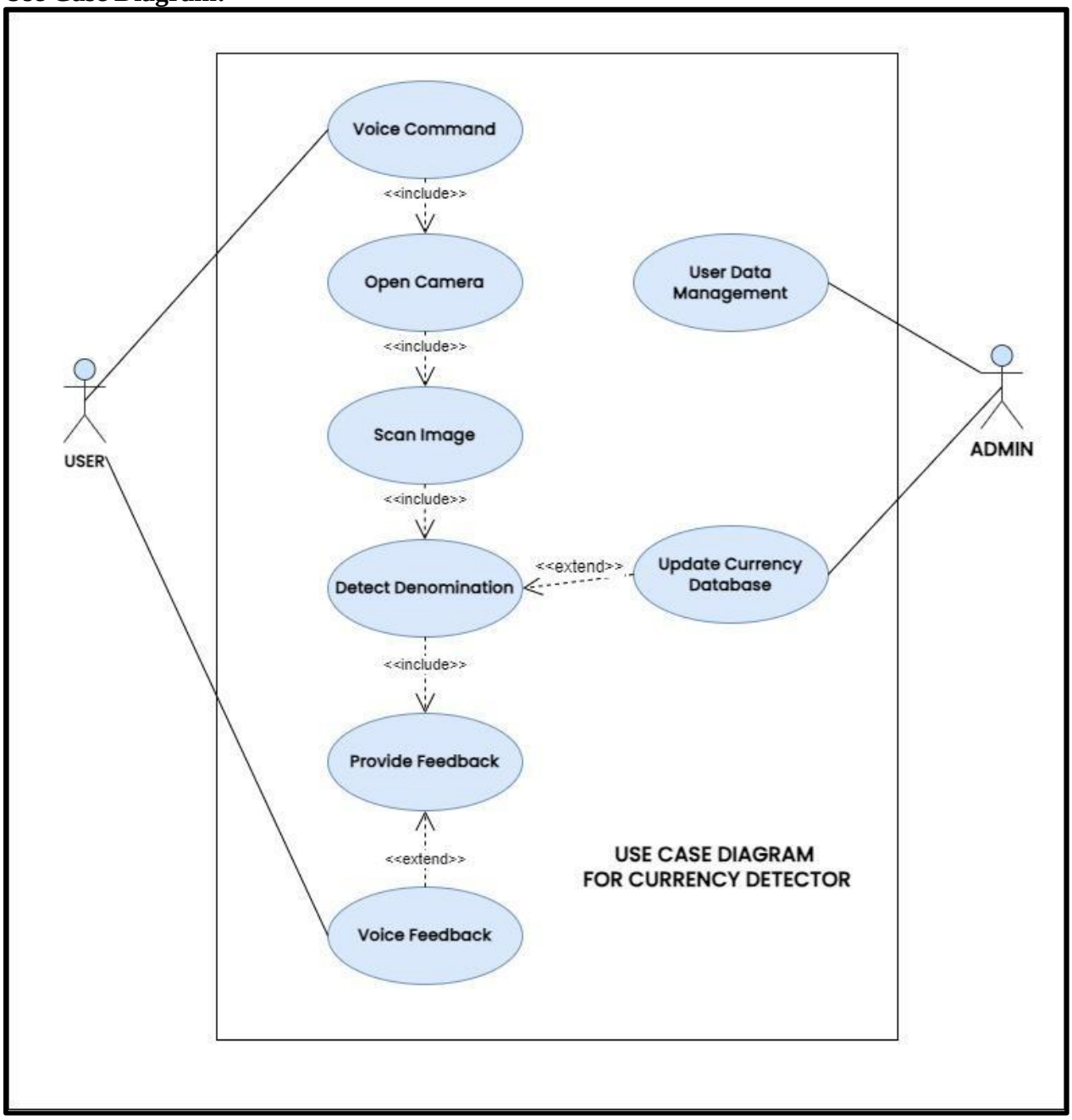
,

- *Join : Combines two concurrent activities into a single flow*

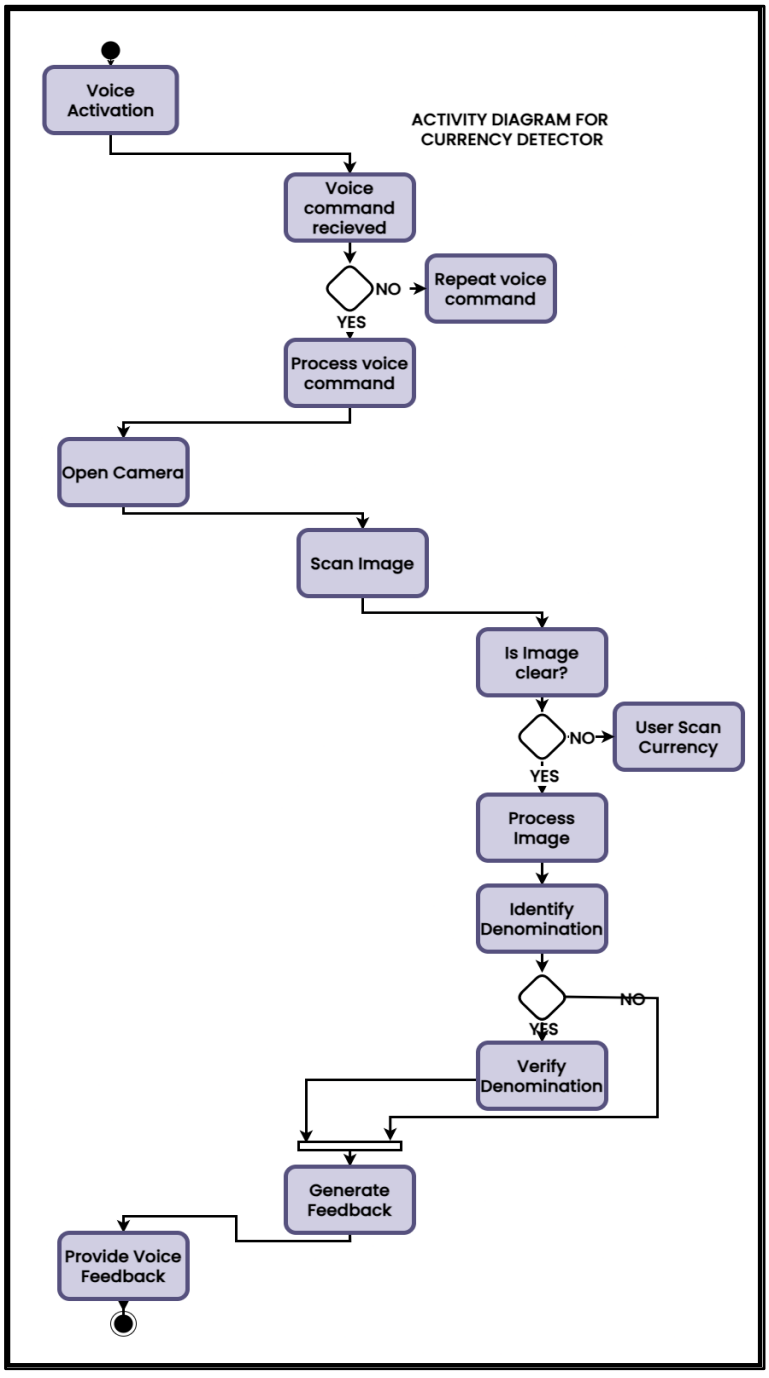
Joint symbol/ Synchronization bar

Solution:

Use Case Diagram:



Activity Diagram:



Conclusion: In this experiment, we successfully created Use Case and Activity diagrams for the Currency Detector designed for the visually impaired. Through this process, we gained valuable insights into how the system functions and how users interact with it. The Use Case diagram clearly outlines the essential functionalities, while the Activity diagram illustrates the step-by-step flow of actions involved in using the device. This work not only enhanced our understanding of UML but also reinforced the importance of designing accessible technology that meets the needs of diverse users. We look forward to leveraging these skills in future projects to create impactful solutions.