Practical no:-01
Study and implementation of Class Diagram

A UML class diagram is a visual tool that represents the structure of a system by showing its classes, attributes, methods, and the relationships between them.

The class diagram below models a student information management system. The central class is the student. Associated with it are the Student Marks, Student Fee and Student Placement.

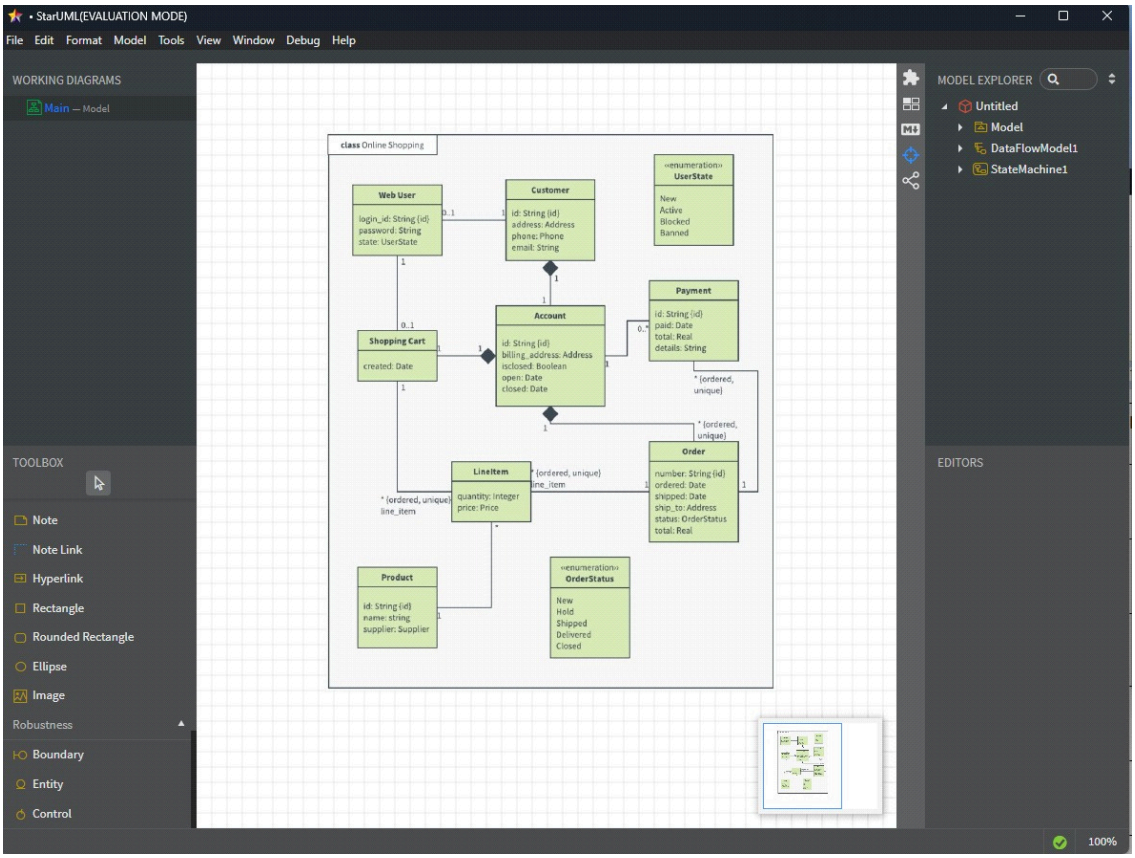
Our class diagram has three kinds of relationships:

Composition: Composition is depicted by a filled diamond shape at the "whole" end of the association line connecting it to the "part" classes.

Association: There is an association between two classes if an instance of one class must know about the other in order to perform its work. In a diagram, an association is a link connecting two classes. In the model below Student fee class is associated with Student class.

Generalization: An inheritance link indicating one class is a superclass of the other. A generalization has a triangle pointing to the superclass. Student is a superclass of Student Placement.

Diagram:



Practical no:-02
Study and implementation of Use Case Diagram

A use case diagram is a graphical representation of a system's functionality and how users interact with it.

The Use case diagram below models Online Shopping.

Elements:

Actors (Stick Figures): Represent users interacting with the system.

* Use Cases (Ovals): Define specific functionalities.

Relationships:

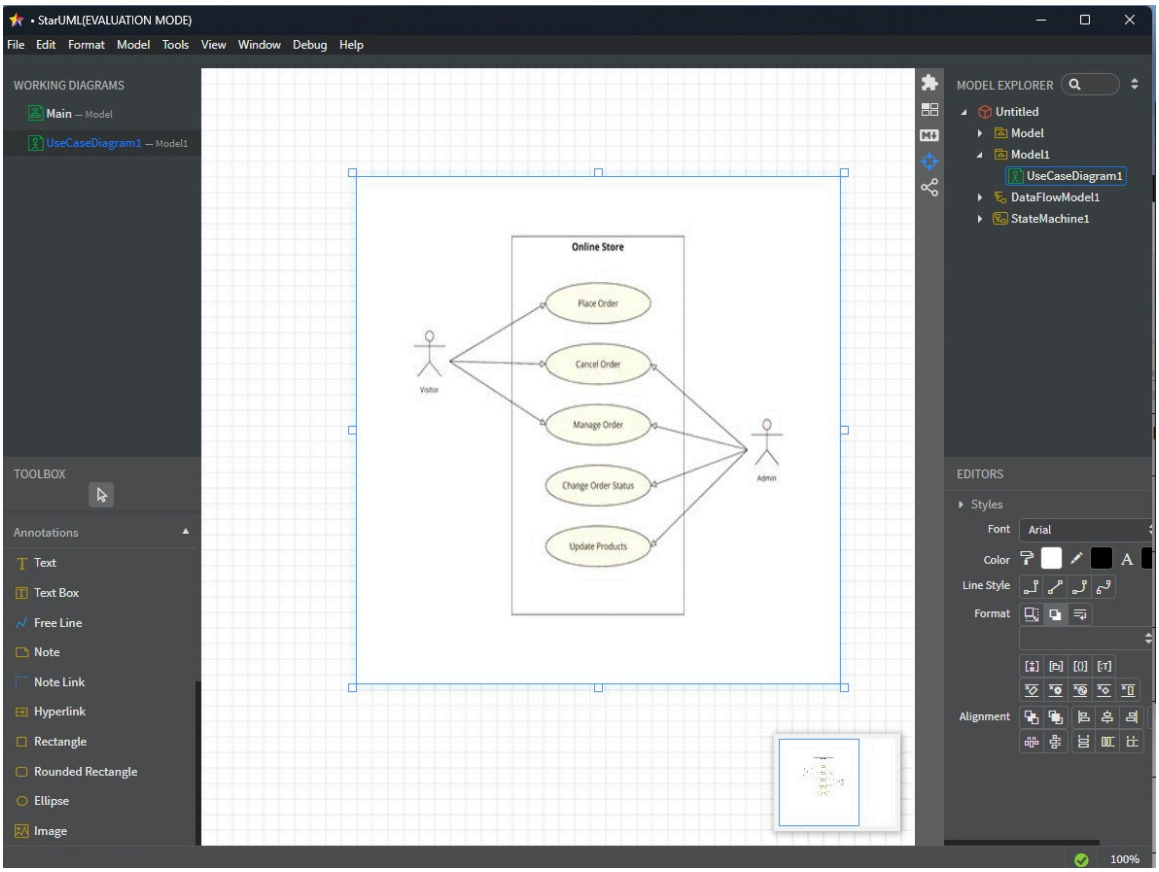
Association (Solid Line): Connects actors to use cases.

* Generalization (Triangle Arrow): Inheritance of behaviour. Customer is a general entity with two subtypes: New Customer and Registered Customer.

* Include(<<include>>): One use case is always part of another. Login includes Verify Username and Password (as it is a mandatory step).

* Extend(<<extend>>): An optional extension of a use case. Purchase Item extends Add Item, Update Item, and Cancel Item (optional functionalities). Payment extends multiple payment methods.

Diagram:



Practical no:-03
Study and implementation of Entity Relationship Diagram

An Entity Relationship (ER) Diagram is a type of flowchart that illustrates how "entities" such as people, objects or concepts relate to each other within a system.

The ER (Entity-Relationship) Diagram below represents the Employee Management System.

Entities and Their Attributes:

Entities (Boxes): Represent tables (Employee, Department, Salary, Leave, Project).

Attributes (Inside Boxes): Represent table columns.

Primary Key (PK): Uniquely identifies each record.

Foreign Key (FK): Creates a relationship between tables.

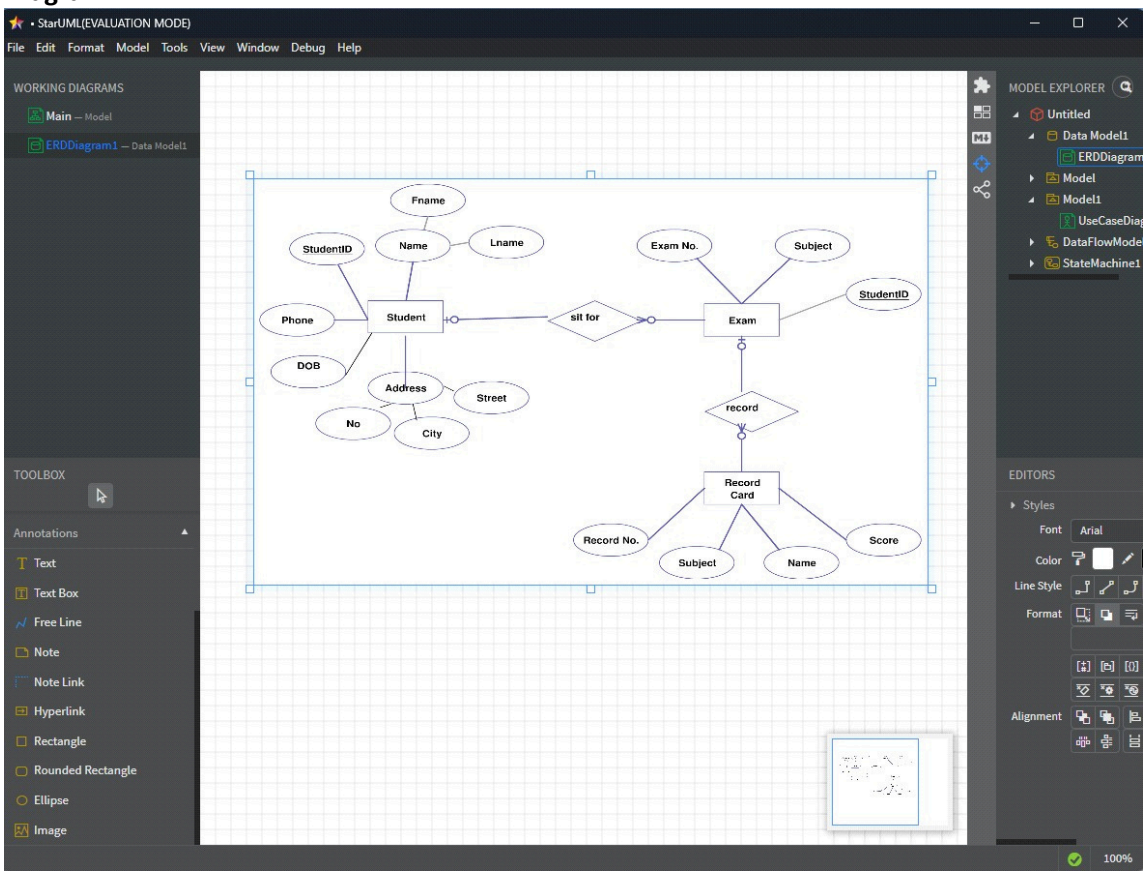
Relationships (Lines with Crow's Foot Notation):

*** One-to-Many (1:0):** An employee works in a department.
An employee receives a salary.

An employee takes leave.

*** Many-to-Many (0:0):** Multiple departments can have multiple projects

Diagram:



Practical no:-04

Study and implementation of Sequence Diagram

A Sequence Diagram represents the interaction between different components in a system over time. It shows how messages are exchanged between objects in a particular order of execution.

Elements Used in the Diagram:

*** Customer (Actor1):** The person initiating the ATM transaction.

ATM Machine: The interface through which the customer interacts.

Bank Server: The backend system that processes the transaction.

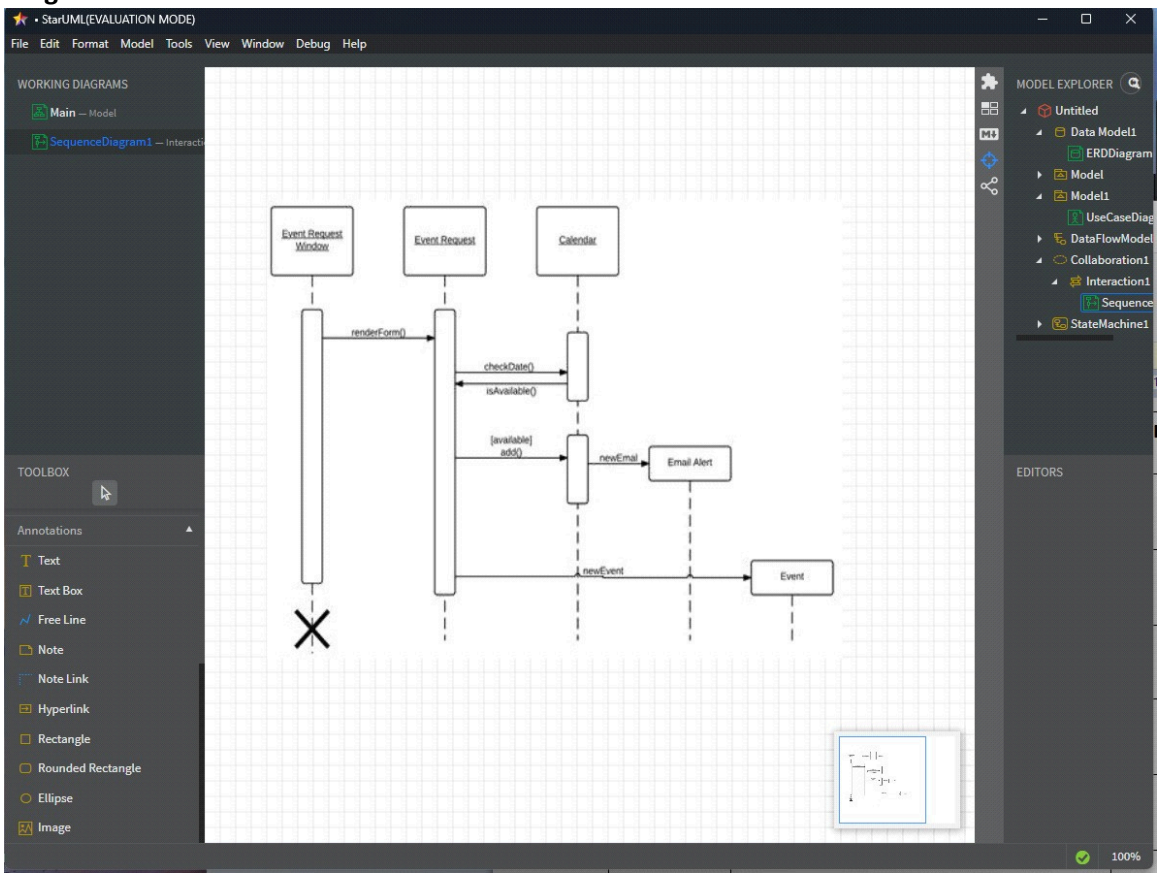
Account: The account of the customer, which gets updated.

Lifelines: Represented by dashed vertical lines extending from each participant.

They indicate the duration during which an object is active.

Messages (Arrows): Solid arrows indicate communication between the actors and system components. Numbered steps show the sequence of interactions.

Diagram:



Practical no:-05
Study and implementation of State Transition Diagram

A State Transition Diagram shows how an object transition between different states based on specific conditions or events.

The below State Transition diagram example is of an ATM System.

Elements used in the diagram:

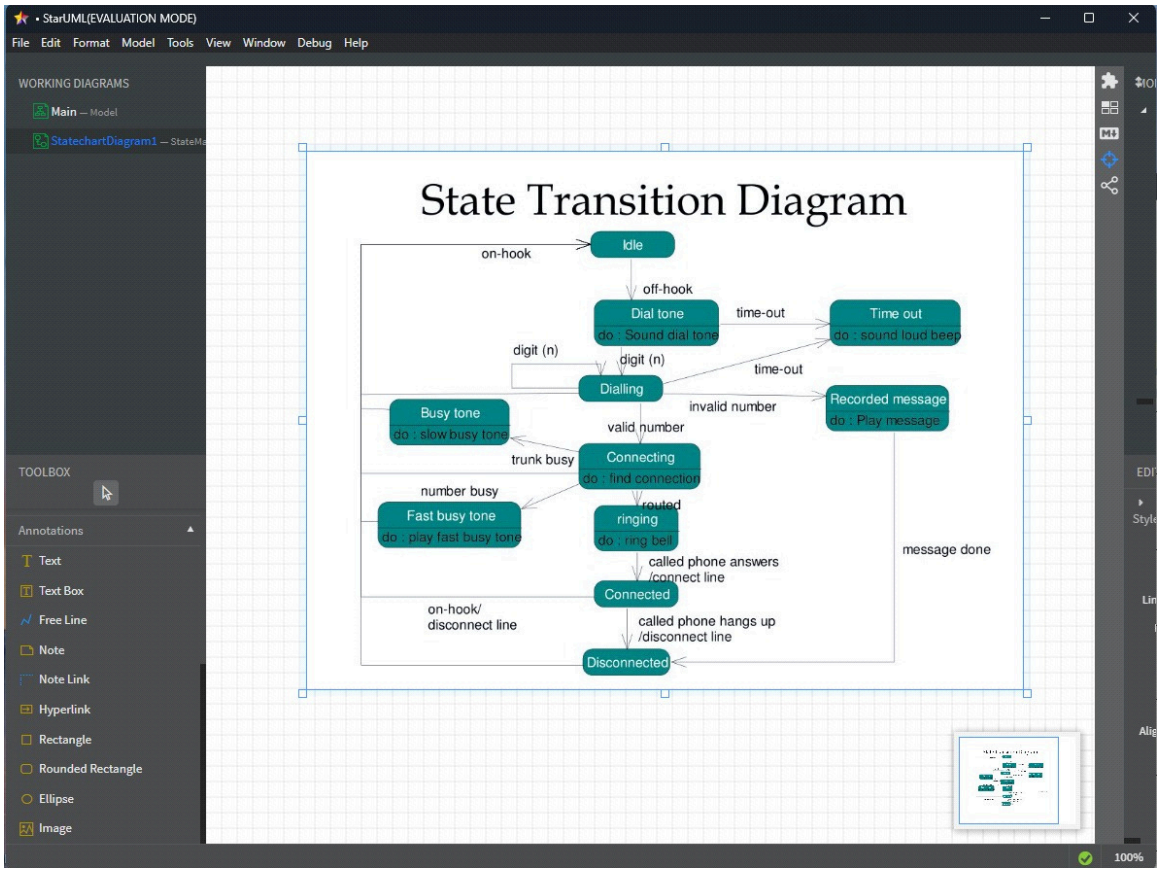
States (Rounded Rectangles): Used to describe the different conditions or modes an object or system can be in at any given time.

Transitions (Arrows): Represent state changes based on conditions or user actions.

Decision Points (Diamonds): Used to check conditions such as "Is PIN valid?" or "Is the balance sufficient?".

*** Final State (Black Circle):** Represents the end of the transaction. The ATM goes back to the Idle state after completion.

Diagram:



Practical no:-06
Study and implementation of Data Flow Diagram

The flow of data in a system or process is represented by a Data Flow Diagram (DFD). It also gives insight into the inputs and outputs of each entity and the process itself.

Elements used in the diagram:

* **Processes (Rectangles with rounded edges):** Represent actions or tasks performed (e.g., login, add/drop course).

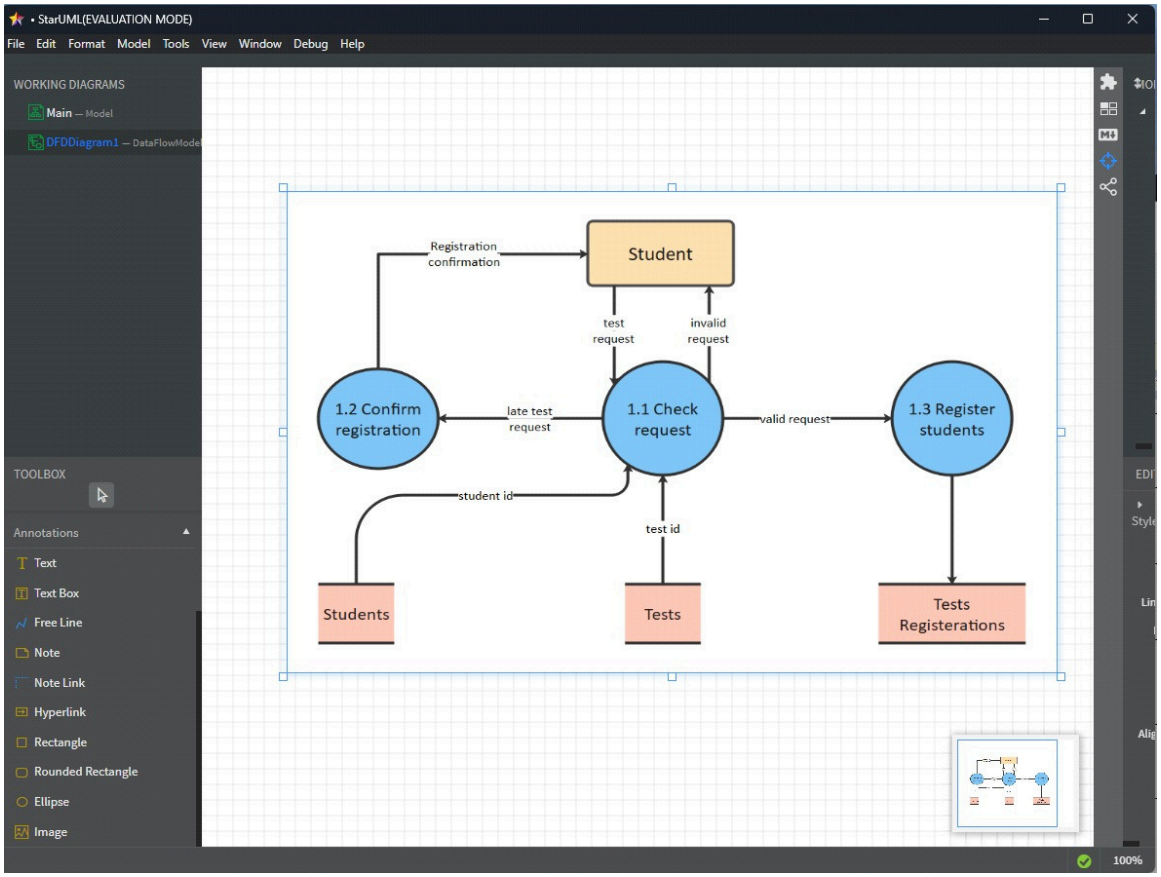
Data Stores (Open-ended rectangles): Indicate where data is stored temporarily (e.g., offered courses).

External Entity (Rectangle): Represents an external source or sink of data (e.g., student information).

Data Flow (Arrows): Show the movement of data between processes, data stores, and entities.

* **Decision Note:** A textual note attached to clarify a decision point, such as checking whether the student is authorized.

Diagram:



Practical no:-07
Study and implementation of Collaboration Diagram

A collaboration diagram, also known as a communication diagram, is an illustration of the relationships and interactions among software objects in the Unified Modeling Language (UML).

The below Collaboration diagram example is of a Course Registration System.

Elements used in the diagram:

* **Actors**: Represent the entities interacting with the system, such as the student or external systems.

* **Objects**: Represent the components of the system (e.g., Course Registration System, Registered Courses, Legacy System).

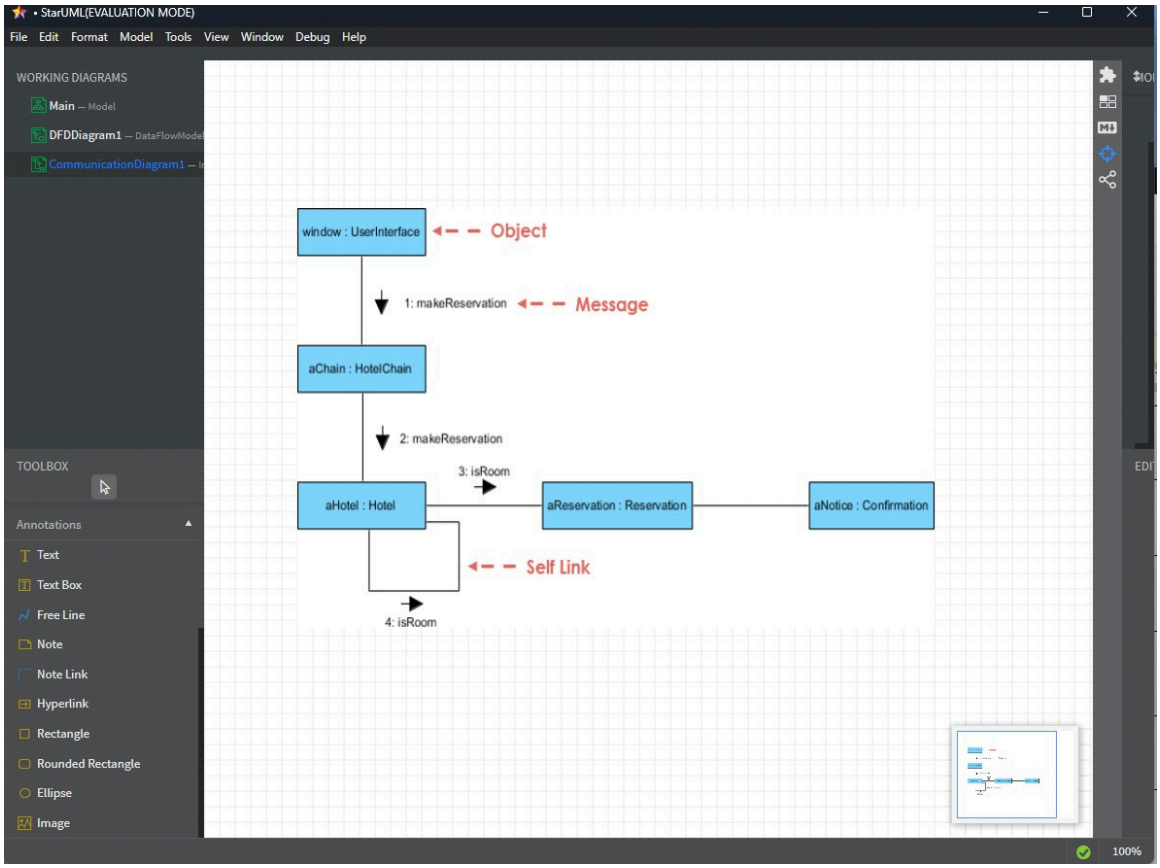
Lifelines: Indicate the life span of an object or actor during the interaction.

* **Messages**: Show the communication between actors and objects, usually numbered for sequence.

* **Conditions**: Represent decision-making logic (e.g., "if students < 10" in step 8).

Sequence: Messages are executed in order, showing how interactions flow between actors and system components.

Diagram:



Practical no:-08
Study and implementation of Activity Diagram

An activity diagram is a type of Unified Modeling Language (UML) flowchart that shows the flow from one activity to another in a system or process.

Elements used in the diagram:

. Black Circle (Initial Node): This represents the start of the activity flow.

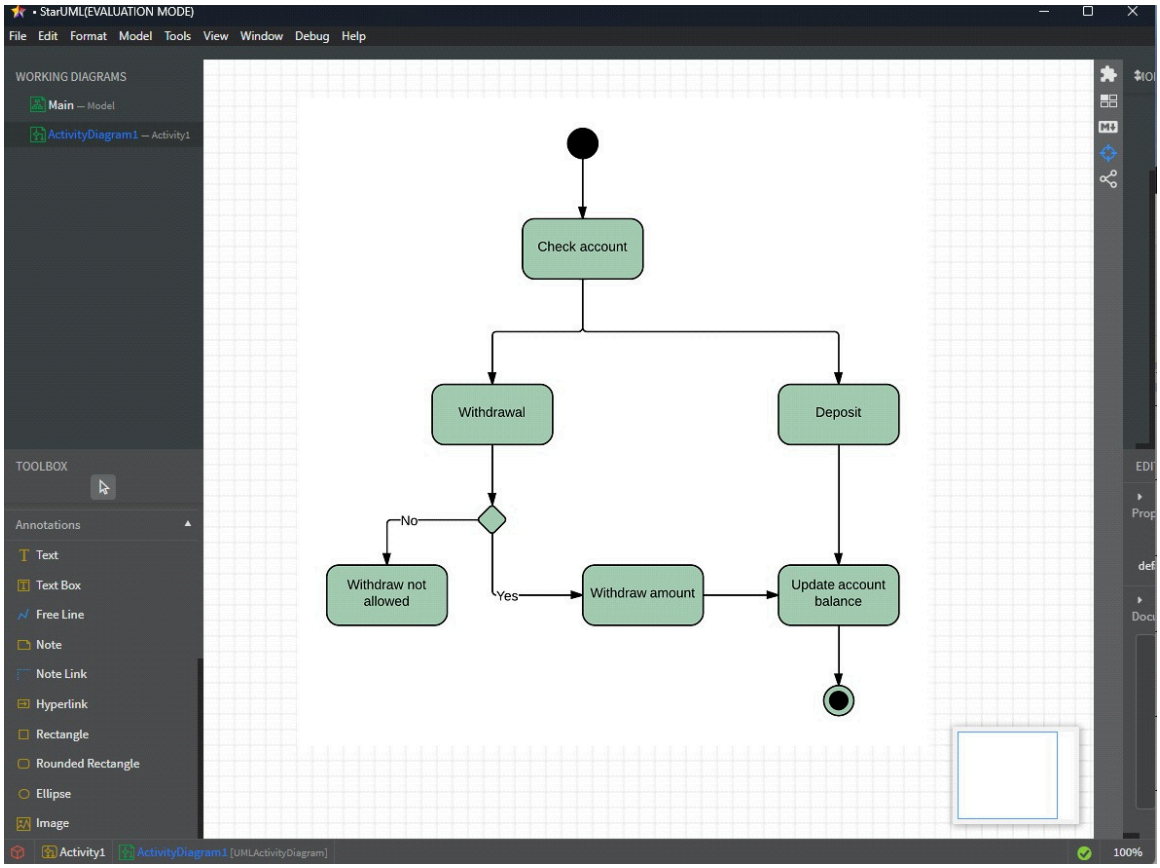
*** Activity/Action Nodes (Rectangles):** These represent the specific actions or activities performed by the user

*** Decision Node (Diamond):** This represents a decision point where a condition is evaluated

*** Arrows (Flow Connectors):** These represent the flow of control between actions.

Final with BoNode (Black Circle rder): This represents the end of the activity flow.

Diagram:



Practical no:-09
Study and implementation of Component Diagram

A component diagram breaks down the actual system under development into various high levels of functionality.

The below Component diagram example is of an ATM System.

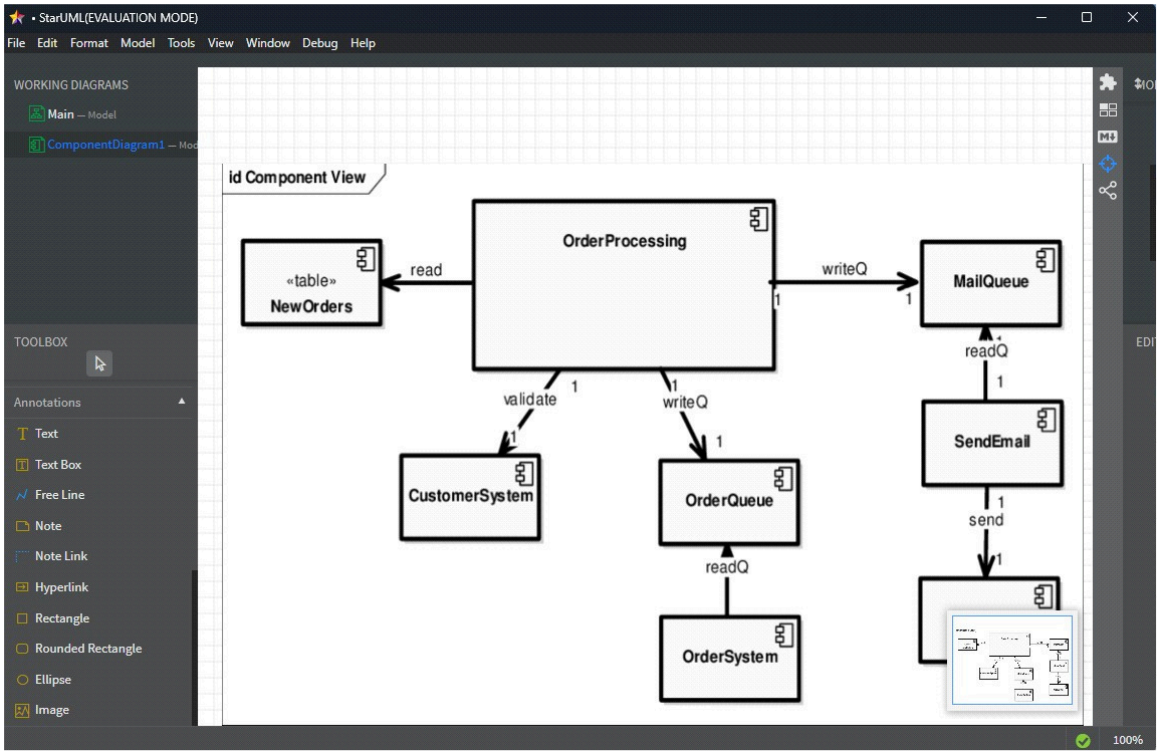
Elements used in the diagram:

Components (Rectangles with puzzle-like connectors): Represent the physical or logical parts of the system that interact with each other.

Interfaces (Puzzle-like connectors): Represent the interfaces or entry/exit points through which components communicate.

Dependencies (Dashed Arrows): Show the relationships or dependencies between components.

Diagram:



Practical no:-10
Study and implementation of Deployment Diagram

A Deployment Diagram shows how the software design turns into the actual physical system where the software will run.

The below Deployment diagram example is of an LAN.

Elements used in the diagram:

Nodes (Rectangles with 3D effects): Represent physical devices or servers involved in the system.

Communication Links (Lines): Represent the connections between nodes for data transfer and communication.

Diagram: