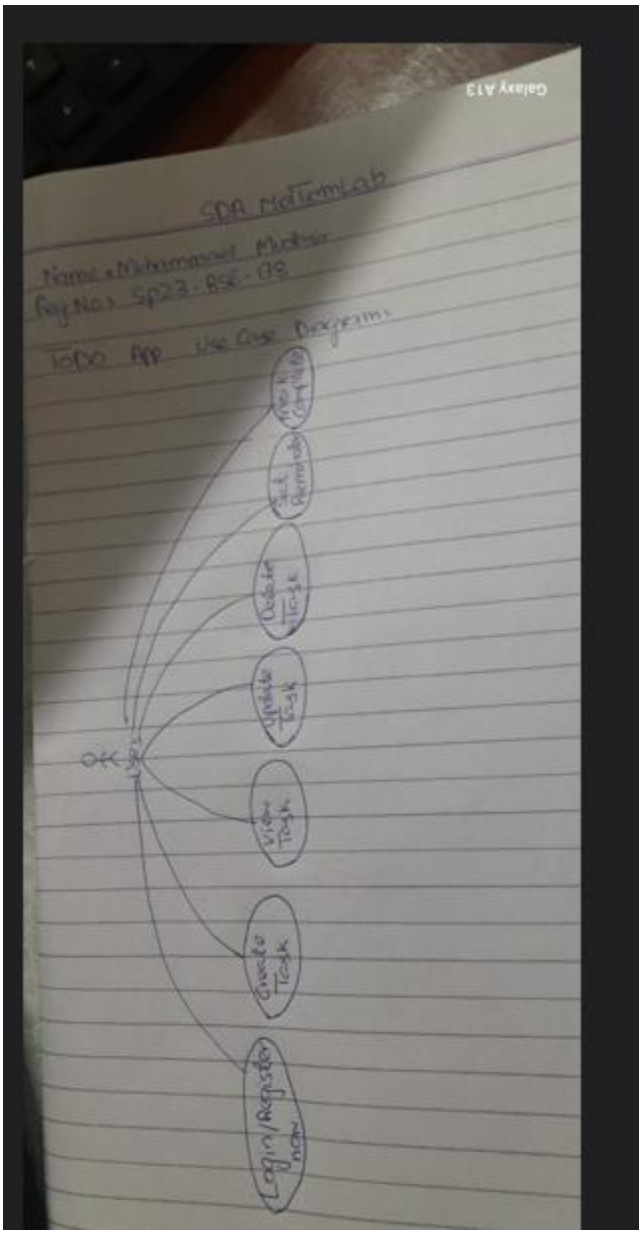
USE CASE DIAGRAM:



The best suited princiapl/pattern that will help in my implementation is Creator principal.

The Creator principle states:

"Assign the responsibility of creating an instance of class A to class B if one or more of the following is true: B contains A, aggregates A, records A, or closely uses A."

Application in my TODO App:

- In my app, the TaskFactory class is responsible for creating different types of tasks such as:
    - NamazTask
    - GymTask
    - GeneralTask
- The factory knows which type of task to instantiate based on the user's input.
- This clearly follows the Creator principle because:
    - TaskFactory is the one that knows the details of each subclass.
    - It encapsulates object creation, which keeps that logic out of the main app (like TaskManager or User).

Focuses on Interactions
Communication diagrams are ideal when user want to show how objects interact to perform a task — in this case, updating a task. It shows the sequence and structure of interactions between:

- User
- UI
- TaskManager
- Task

Clear Responsibility Distribution

- UI handles user interaction.
- TaskManager handles business logic and task retrieval.
- Task is responsible for maintaining and updating its internal data.

Low Coupling, High Cohesion

- Each class does only what it's supposed to — following GRASP principles (like Controller and Creator).
- The update operation is handled by the Task itself, which means high cohesion — perfect for maintainability.

Reflects Real OOP Design:
My app already uses object-oriented design with classes like Task, TaskManager, etc.

This diagram maps directly onto your existing code structure, making it accurate and practical.