Assignment no 7

Aim:
Prepare and implement state model

Problem Statement:
- Prepare a State Model.
- Identify States and events for your system.
- Study state transitions and identify Guard conditions.
- Draw State chart diagram with advanced UML 2 notations.
- Implement the state model with a suitable OO language

Objective:
- To Identify States Transitions, events in the system flow.
- Draw State Diagram and Implement Model.

Theory:

State Machine Diagram:
A state machine diagram models the behavior of a single object, specifying the sequence of events that an object goes through during its lifetime in response to events.
The name of the diagram itself clarifies the purpose of the diagram and other details. It describes different states of a component in a system. The states are specific to a component/object of a system.
A State chart diagram describes a state machine. Now to clarify it state machine can be defined as a machine which defines different states of an object and these states are controlled by external or internal events.
As the State chart diagram defines states it is used to model the lifetime of an object.

Purpose:
State chart diagram is one of the five UML diagrams used to model the dynamic nature of a system. They define different states of an object during its lifetime. And these states are changed by events.

States

A state is denoted by a round-cornered rectangle with the name of the state written inside it.

Initial and Final States

The initial state is denoted by a filled black circle and may be labeled with a name. The final state is denoted by a circle with a dot inside and may also be labeled with a name.

Transitions

Transitions from one state to the next are denoted by lines with arrowheads. A transition may have a trigger, a guard and an effect, as below.

"Trigger" is the cause of the transition, which could be a signal, an event, a change in some condition, or the passage of time.

"Guard" is a condition which must be true in order for the trigger to cause the transition. "Effect" is an action which will be invoked directly on the object that owns the state machine as a result of the transition.

State Actions

In the transition example above, an effect was associated with the transition. If the target state had many transitions arriving at it, and each transition had the same effect associated with it, it would be better to associate the effect with the target state rather than the transitions.

This can be done by defining an entry action for the state. The diagram below shows a state with an entry action and an exit action.

Self-Transitions

A state can have a transition that returns to itself, as in the following diagram.

This is most useful when an effect is associated with the transition.

Entry Point

Sometimes you won't want to enter a sub-machine at the normal initial state. For example, in the following sub-machine it would be normal to begin in the "Initializing" state, but if for some reason it wasn't necessary to perform the initialization, it would be possible to begin in the "Ready" state by transitioning to the named entry point.

Exit Point
In a similar manner to entry points, it is possible to have named alternative exit points. The following diagram gives an example where the state executed after the main processing state depends on which route is used to transition out of the state.

Choice Pseudo-State
A choice pseudo-state is shown as a diamond with one transition arriving and two or more transitions leaving. The following diagram shows that whichever state is arrived at, after the choice pseudo-state, is dependent on the message format selected during execution of the previous state.

Junction Pseudo-State
Junction pseudo-states are used to chain together multiple transitions. A single junction can have one or more incoming, and one or more outgoing, transitions; a guard can be applied to each transition. Junctions are semantic-free.
A junction which splits an incoming transition into multiple outgoing transitions realizes a static conditional branch, as opposed to a choice pseudo-state which realizes a dynamic conditional branch.

Terminate Pseudo-State
Entering a terminated pseudo-state indicates that the lifeline of the state machine has ended. A terminated pseudo-state is notated as a cross.

History States
A history state is used to remember the previous state of a state machine when it was interrupted. The following diagram illustrates the use of history states. The example is a state machine belonging to a washing machine.

Concurrent Regions
A state may be divided into regions containing sub-states that exist and execute concurrently. The example below shows that within the state "Applying Brakes", the front and rear brakes will be operating simultaneously and independently. Notice the use of fork and join pseudo-states, rather than choice and merge pseudo-states. These symbols are used to synchronize the concurrent threads.

How to draw a State chart Diagram?

State chart diagram is used to describe the states of different objects in its life cycle. So the emphasis is given on the state changes upon some internal or external events. These states of objects are important to analyze and implement them accurately.

State chart diagrams are very important for describing the states. States can be identified as the condition of objects when a particular event occurs.

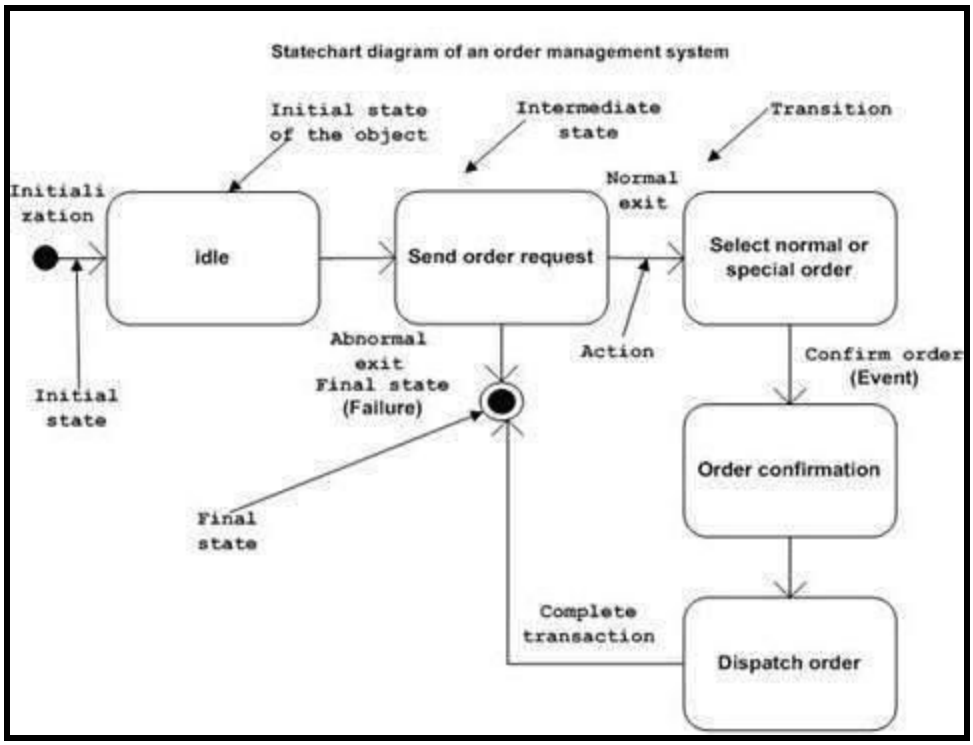
Before drawing a State chart diagram we must have clarified the following points:

- Identify important objects to be analyzed.
- Identify the states.
- Identify the events.

The following is an example of a State chart diagram where the state of Order object is analyzed.

The first state is an idle state from where the process starts. The next states are arrived for events like send request, confirm request, and dispatch order. These events are responsible for state changes of order.

During the life cycle of an object (here order object) it goes through the following states and there may be some abnormalities also. This abnormal exit may occur due to some problem in the system.



Where to use State chart Diagrams?

From the above discussion we can define the practical applications of a State chart diagram. State chart diagrams are used to model the dynamic aspect of a system like the other four diagrams discussed in this tutorial. But it has some distinguishing characteristics for modeling dynamic nature.
State chart diagrams define the states of a component and these state changes are dynamic in nature.

So its specific purpose is to define state changes triggered by events. Events are internal or external factors influencing the system.
State chart diagrams are used to model states and also events operating on the system.

When implementing a system it is very important to clarify different states of an object during its lifetime and state chart diagrams are used for this purpose. When these states and events are identified they are used to model it and these models are used during implementation of the system.

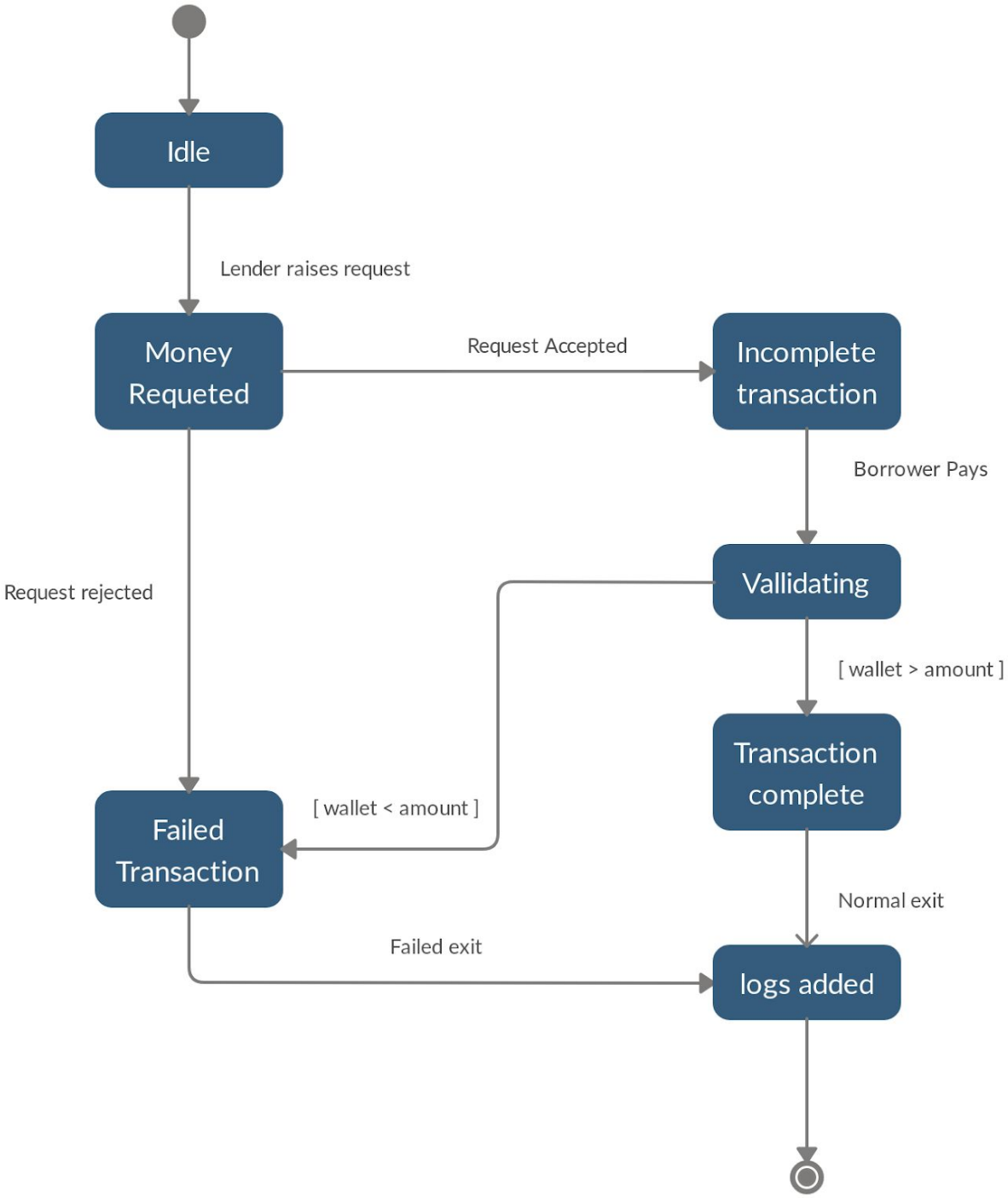
If we look into the practical implementation of State chart diagrams then it is mainly used to analyze the object states influenced by events.

This analysis is helpful to understand the system behavior during its execution.
So the main usages can be described as:

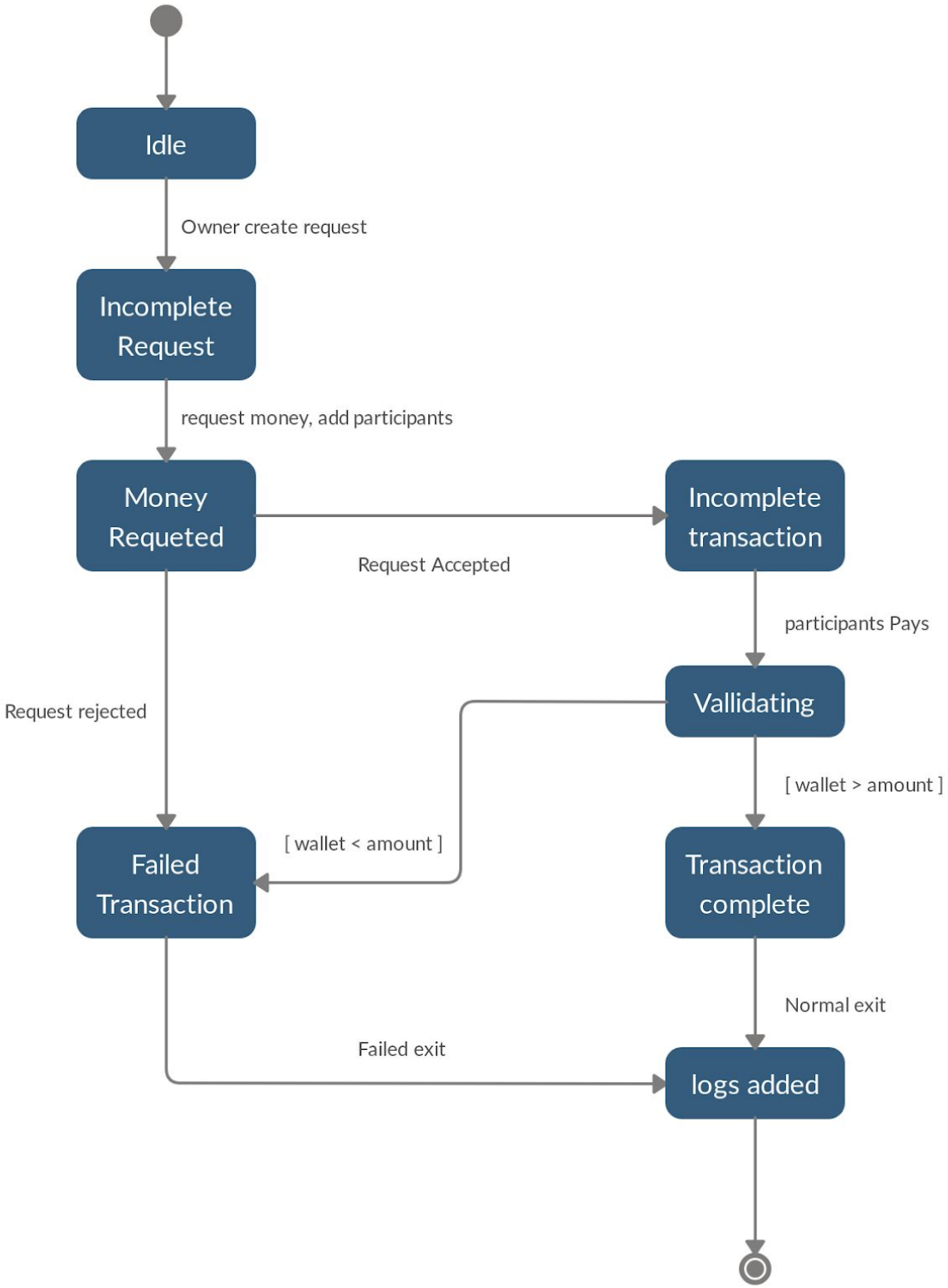
➢ To model object states of a system.
➢ To model a reactive system. Reactive system consists of reactive objects.
➢ To identify events responsible for state changes.
➢ Forward and reverse engineering.

State Diagrams:

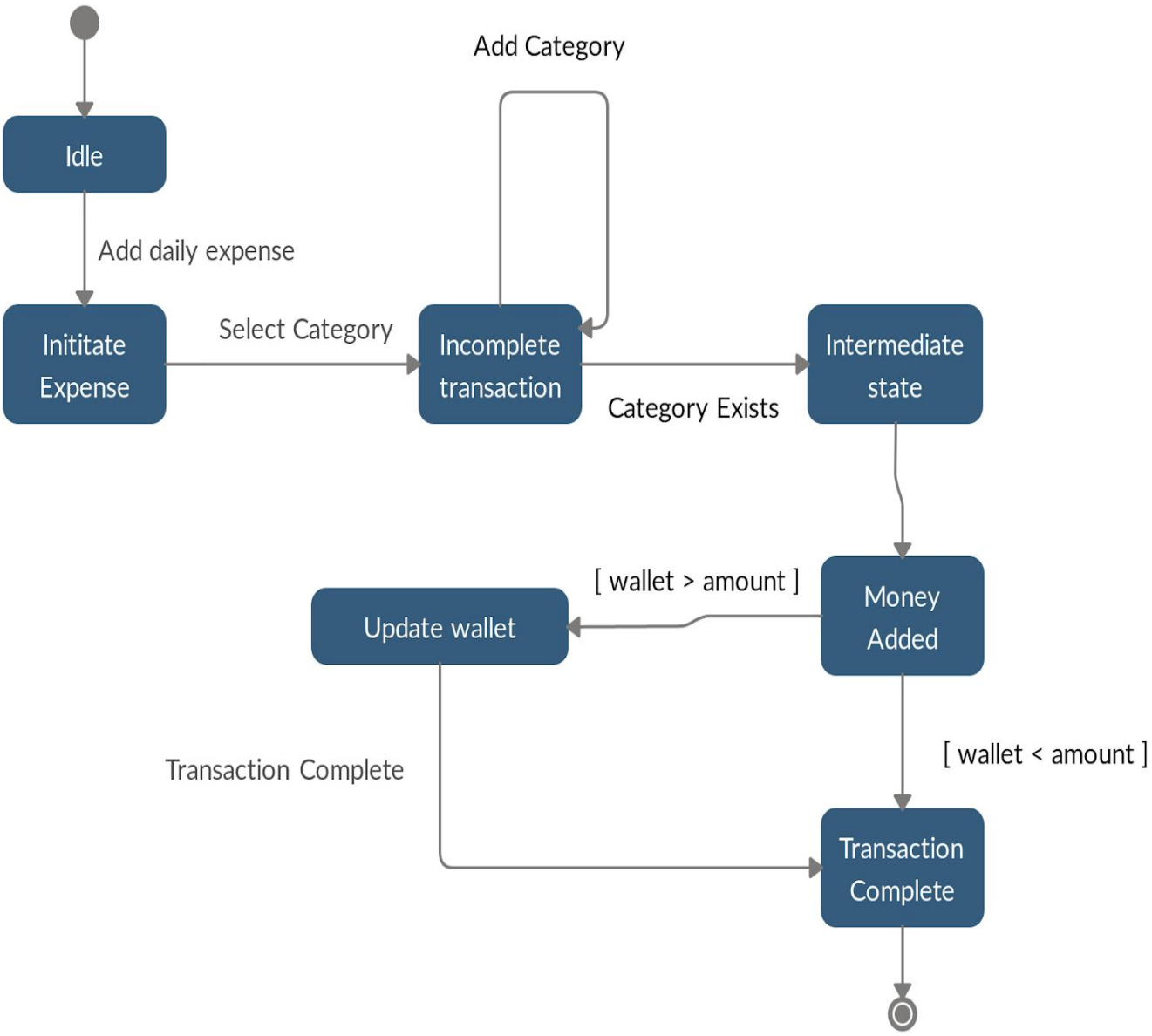
Personal Expense



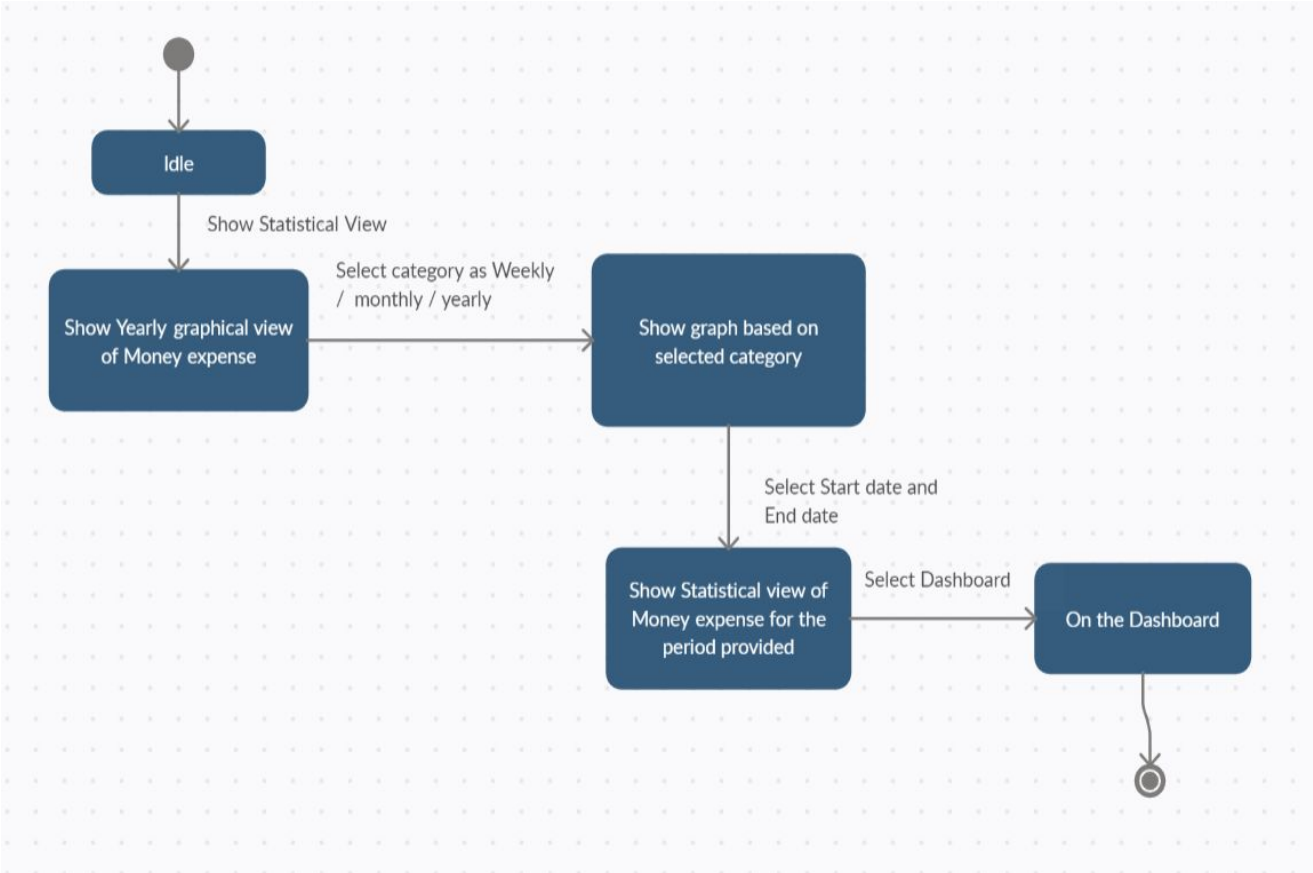
Group Expense



Daily Expense



Dashboard



<u>Conclusion</u>:

We prepared and implemented a State diagram successfully.