Problem:

The machine starts in a locked state (Locked). When a coin is detected (Coin), the machine changes

to the unlocked state (UnLocked) and open the turnstyle gate for the person to pass. When the

machine detects that a person has passed (Pass) it turns back to the locked state. If a person attempts

to pass while the machine is locked, an alarm is generated. If a coin is inserted while the machine is

unlocked, a Thankyou message is displayed. When the machine fails to open or close the gate, a

failure event (Failed) is generated and the machine enters the broken state (Broken). When the

repair person fixes the machine, the fixed event (Fixed) is generated and the machine returns to the

locked state.

A diagram of a coin

Description automatically generated with low confidence

FSM Model

A picture containing text, screenshot, receipt, line

Description automatically generated

Programming points ::

1. Create interface class for state
2. Create functions for all actions (transitions) in interface state class.
3. Create class for all the states derive it from interface state class.
4. Override functions in inherited state class only if the action need to care in that state otherwise let the default function get executed from the interface state class
5. Create the context class
6. Pass the context class object to State class functions that helps to change the state.
7. Make sure singleton object created for all the state classes. In case any status or state value is used in interface class ,then define those variables at static. So that all the state classes access to the same variable.