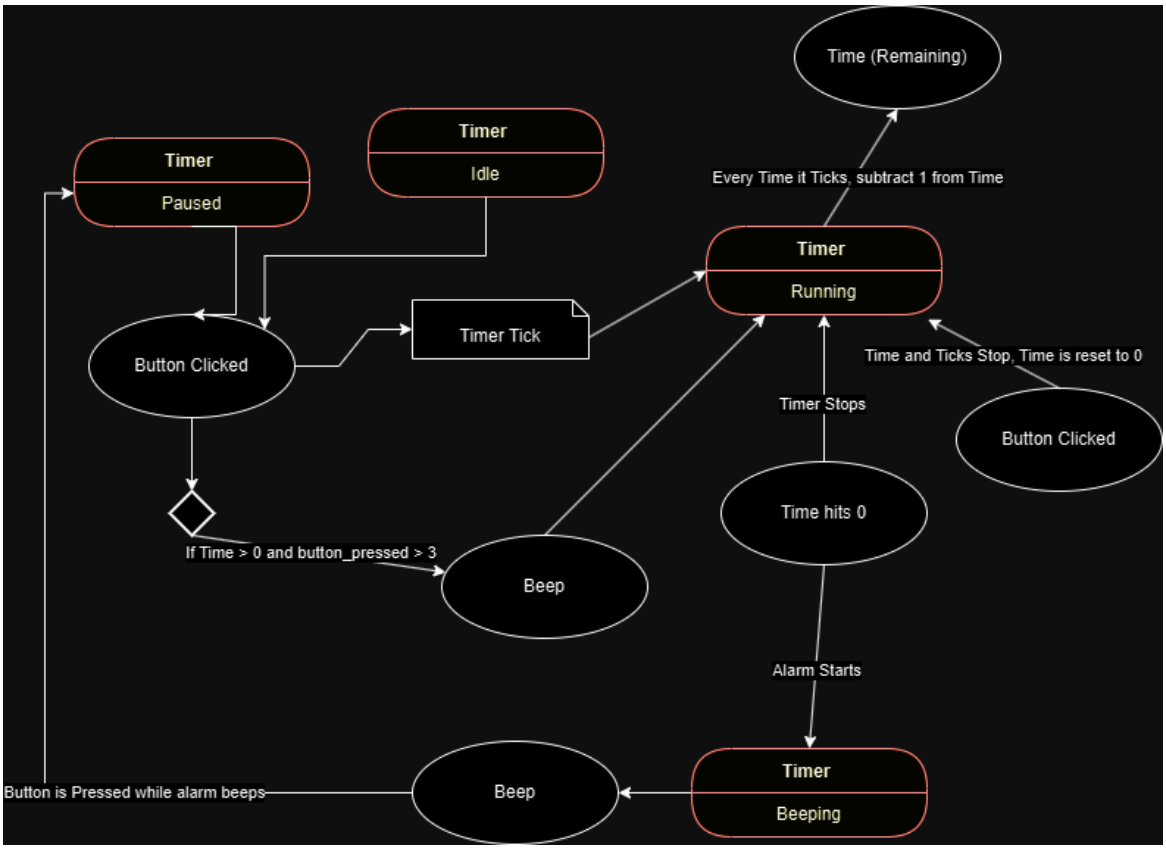
Extended State Diagram:



Group Development Journey:

Our group development journey was a process in learning how the State design pattern works in a practical application. We started with a code overview to learn how the different abstract classes and interfaces work together, and then studied the concrete classes and state machines.

In our development journey, we worked together as a unit of four throughout the development of the project. Each team member took a turn leading a section of the project. Having done the UML diagramming of the stopwatch idea prior to starting development, we as a team knew the general direction we would take when programming and problem solving.

The first thing we added was the ability to change states in the stopwatch machine to get the core of the machine working. This allowed us to start, stop, add a lap, and reset the stopwatch. With this, we needed to alter some of the different equations set in place for calculating time in

order to determine the time down to milliseconds through the Java timer package.

Afterwards, we switched our direction of development towards the timer. This is where we made edits and cuts to the stopwatch to streamline timer code. One struggle we found ourselves facing was finding a way to track when the timer had hit 99, and implementing a getter method to be able to track that inside the state machine.

One thing that came up with the timer that was unique was the question of how to handle the case when time was >99, which is the max digit output for a 2 digit clock. Additionally, on the timer we needed to implement beeping noises, and thus we took inspiration from clickcounter's creation and implementation of the noise.

The relationship between our extended state machine, and our code, was key in allowing everyone on the team to understand the vision of the project, along with the splitting of tasks. Throughout the development process, this gave everyone the underlying confidence to contribute, as we knew as a whole where the team wanted to get to. At the time of making the state machine diagram, this wasn't apparent to us. However, developing with a larger team of four individuals made us realize the importance of proper planning in order to coordinate our programming appropriately.

The state machine lent to our understanding of the operation of the timer at different junctures. Without it, we may have forgotten that once the button is clicked, we must wait three seconds for it to start and subsequently beep. The diagram was definitely an oversimplification of what was required for the completion of our program, as the coding specific components required a lot of detail that was necessary to fulfill requirements. However, the model gave us a macroscopic lens into the viewing of our project, allowing us to gain a holistic view of our tasks before beginning.

As a team, we have agreed that it is more effective to model first before coding. Now that we have our code, we would have thought of the Beeping (ALARM) state and the decrement functionality.