How it was Built:

We started with the stopwatch code and turned it into a countdown timer. First, we looked through the original state machine to figure out what to keep, what to tweak, and what to toss. We ditched the lap feature right away since the timer doesn't need it.

Then we built out the three main states: Stopped, Running, and Alarm. Using the State pattern helped a lot here, we could work on each state one at a time without breaking everything else.

The trickiest bit was that 3 second auto start in the Stopped state. We had to track how much time passed without the user pressing the button and reset the countdown if they did. Took a couple tries to get it working smoothly.

Adding sound wasn't too bad once we got the hang of MediaPlayer. We made it beep once when the timer starts on its own, and repeat when time's up. Pressing the button shuts the alarm off right away.

We cleaned up the UI by taking out the extra button and swapping the minutes: seconds display for just two digits. Much simpler.

We used Git to share code and split up the work, and tested regularly in the Android emulator so we didn't end up with big surprises at the end.

The Model compared to the Code

Our state diagram and the actual code match up pretty well. The three states in the diagram line up with the three Java classes we wrote, and all the transitions in the model became method calls in the code. Both follow the State pattern, which keeps things organized and easy to follow.

But there are a couple differences. In the diagram, transitions have little conditions (like "time > 0"). In the code, those are just if statements inside the states. Also, we had to add some extra variables to track timeouts and counters that didn't show up in the diagram. Modeling first helped us understand what we were building before jumping into code. After writing it, though, we realized not every little coding detail needs to be in the model, it's better to keep the diagram simple and high-level.