David Zhai
State Management Document
On my platform of choice, Android, the following states are possible.

- Created state. This is the state which fires when the system first creates the activity
- Started state - when an activity enters the started state, the system invokes onStart().
- Resumed state - once the callback for onStart() finishes, the activity enters the resumed state, and comes to the foreground. Another way this can happen is reentering from the paused state. The user typically interacts with the app in this state, and the app stays in this state until an interruption occurs.
- Paused state - when an interruption occurs, the activity will enter the paused state.
- Stopped state - when the activity is no longer visible to the user
- Destroyed state - when the activity is finished due to user dismission or finish() being called.

My food recommendation app, Nyoom, will need to use all these states as explained below.
Created state - This matters since all apps need to start with onCreate(), which will be in charge of essential setups and startup logic. According to the documentation, this logic happens only once for the entire life of the activity. My app must initialize UI components, location services (assuming permission is granted) and prepare the app to receive preferences (if they are a first time user)
Started state- Why it matters: The app will become visible here, which is important for the activity to enter the foreground and become interactive. According to the documentation, the code which maintains the UI is initialized when onStart() is called. This is exactly what must happen with my app in this state.
Resumed state - Why it matters: The app comes to the foreground in this state, and allows the user to interact with the app until the state is changed. For this state, my app must ensure that any lifecycle-aware component tied to the activity lifecycle receives the ON_RESUME event. My app should actively be monitoring location data in this state, and be ready for users to enter preferences into text fields.
Paused state - Why it matters: Although my app is made to be quick and take only a minute or so to give a recommendation, users will inevitably pause the app on certain occasions such as realizing they don't actually want to use it. This state can occur often; an instance of this happening in my app would be the user switching to google maps to navigate to a recommended restaurant. My app needs to suspend activities that aren't necessary for the background. It should also save changes or states.
Stopped state - Why it matters: Even though a user cannot see my app, it could still be in the background running (such as if google maps is open). The stopped state should handle stopping the functionality that doesn't need to run while the app is not visible on the screen. My app in particular should release features such as location services until the app is restarted or returns to the foreground.

Destroyed state - Why it matters: All apps must eventually be destroyed. The proper handling of this state is vital to ensure system resources are released, and that the app can restart smoothly without issues next time. My app must correctly release resources and destroy the activity for this stage.