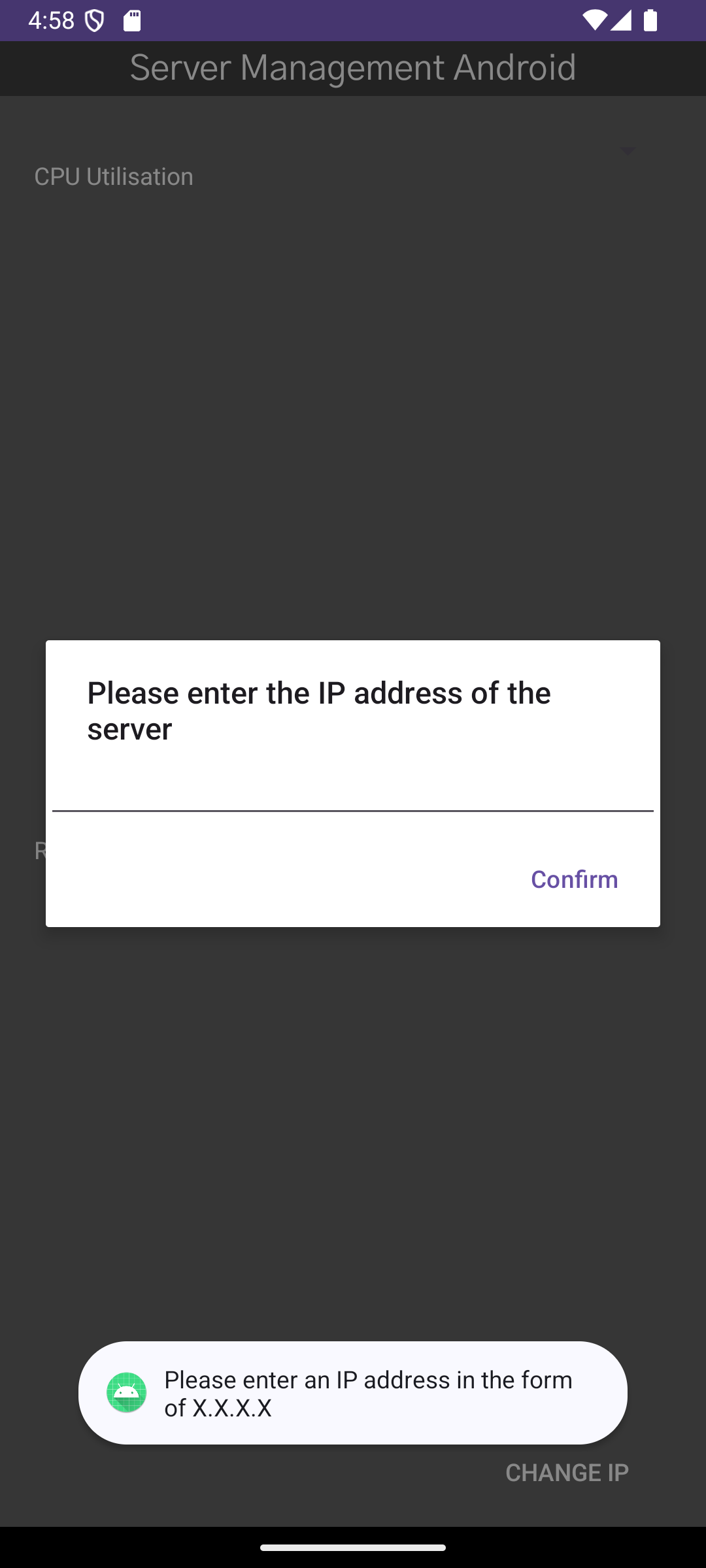
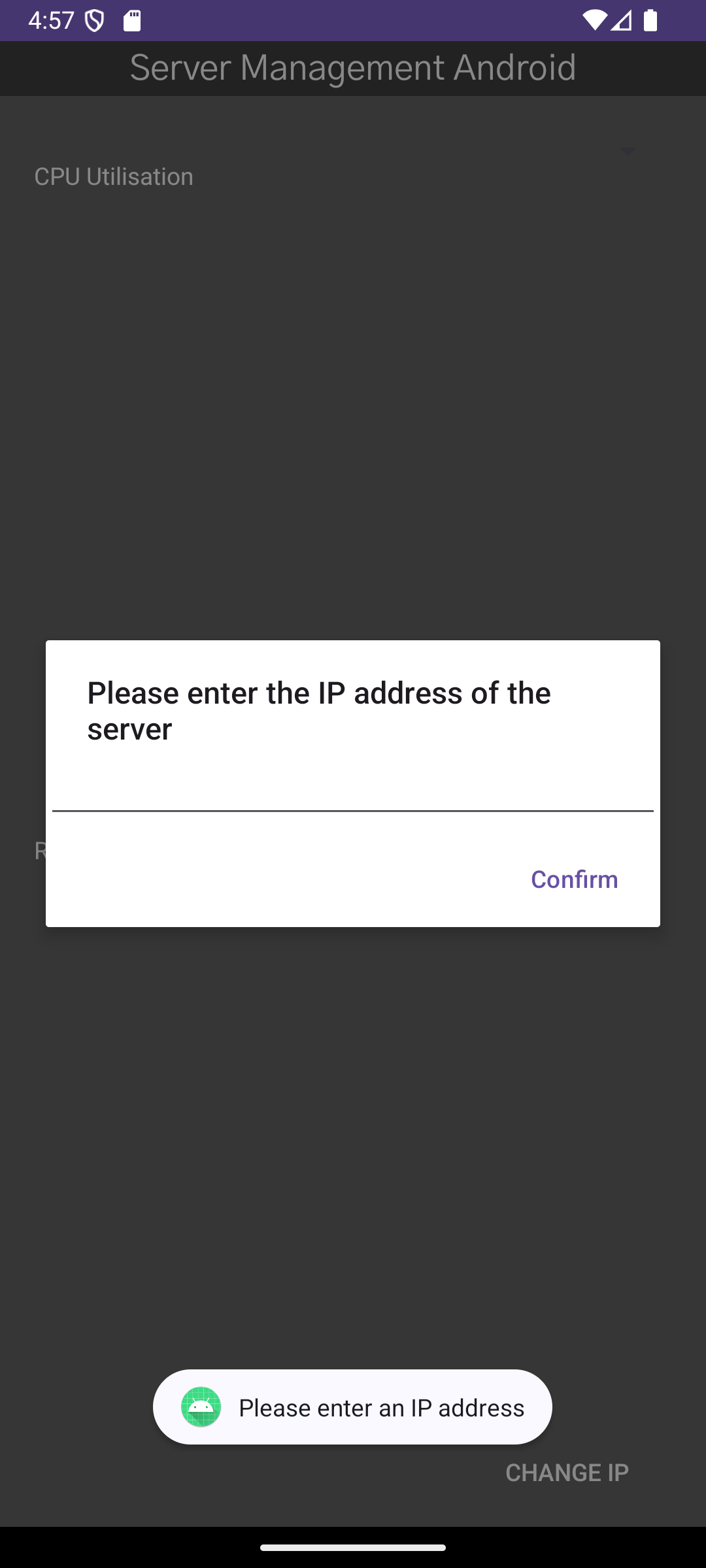
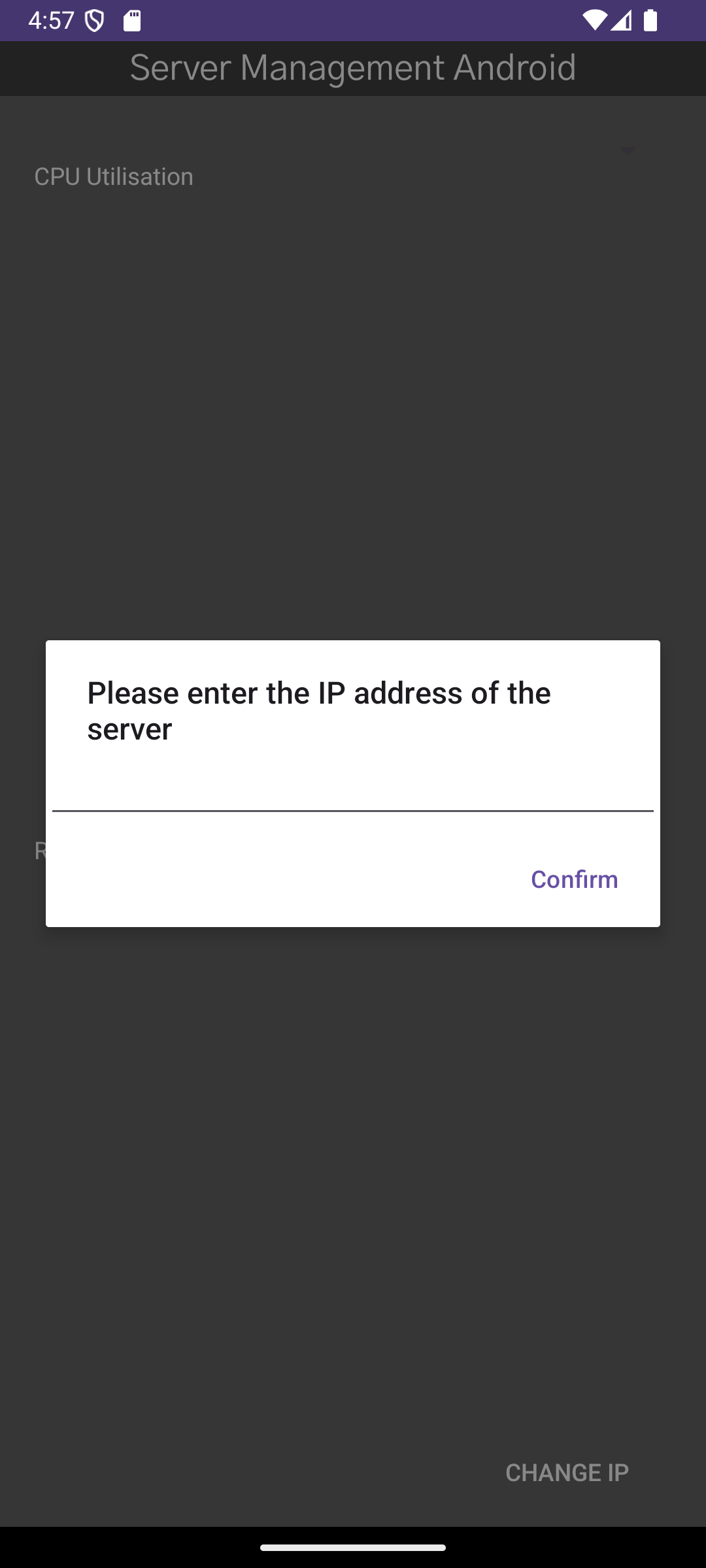
Server Management Android App

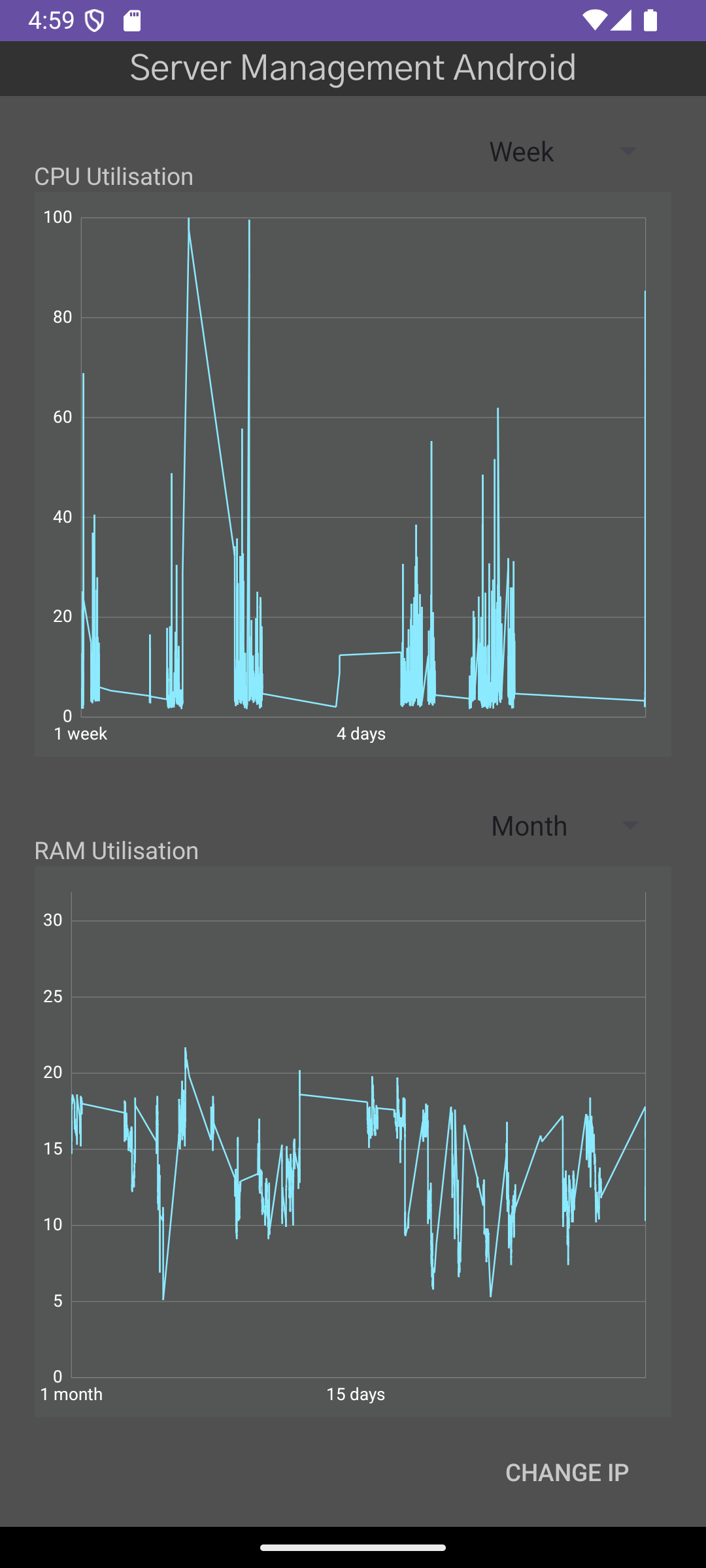
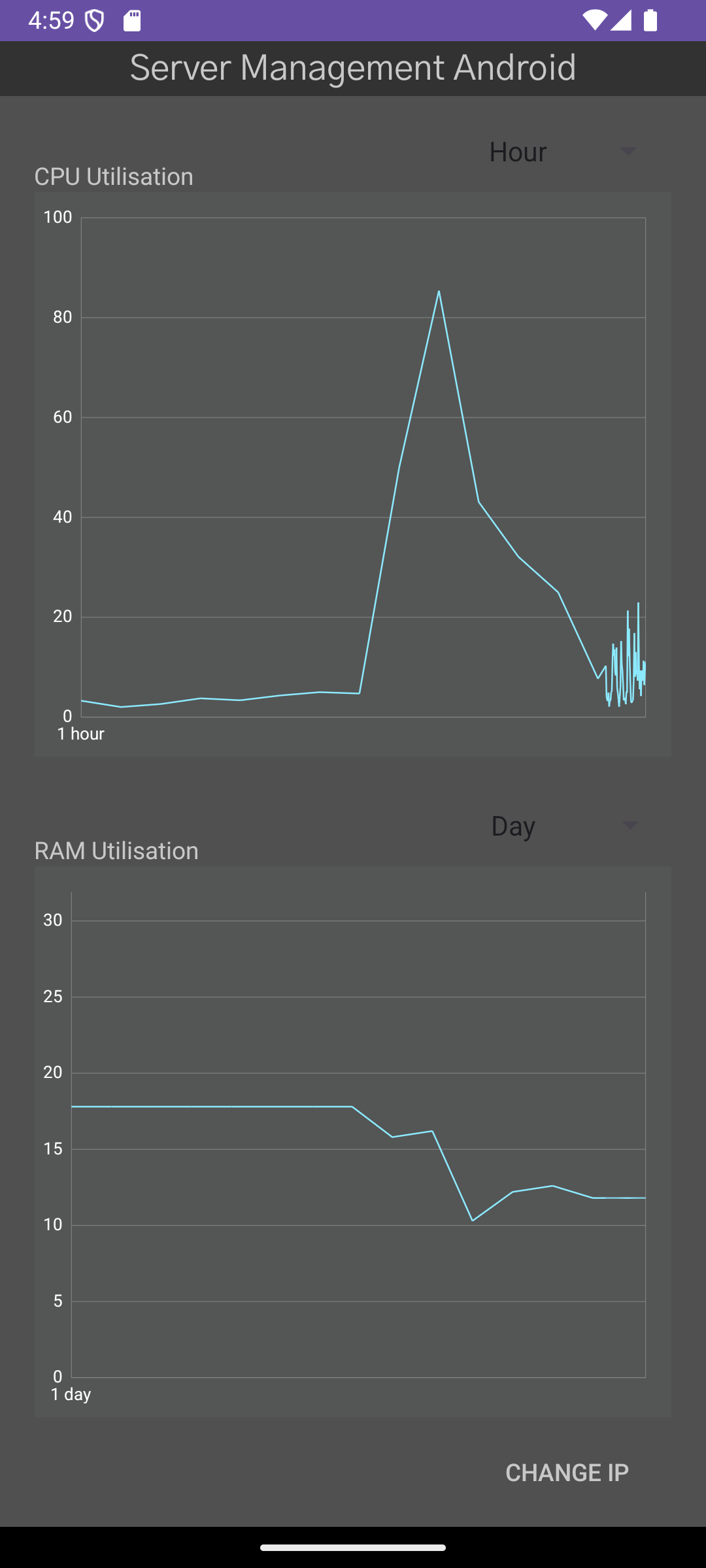
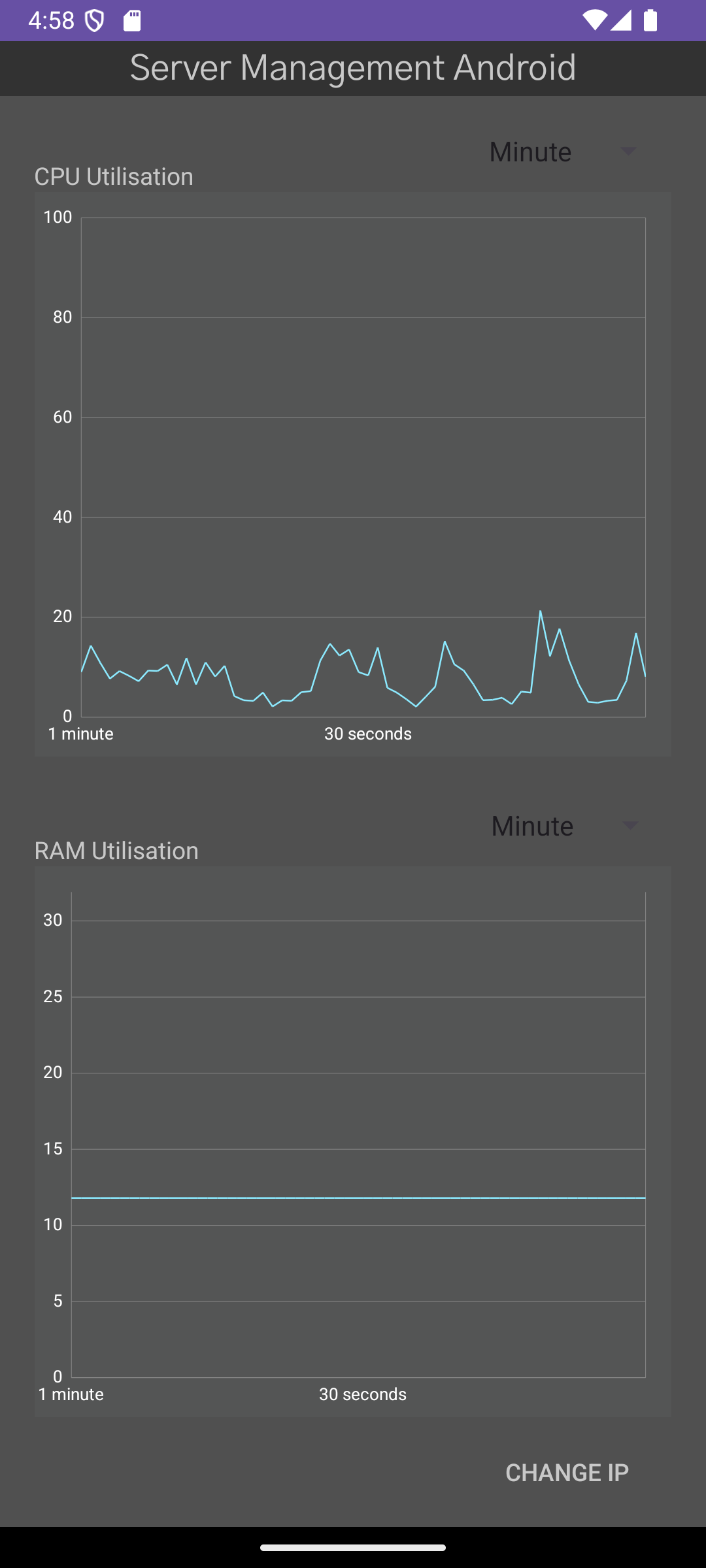
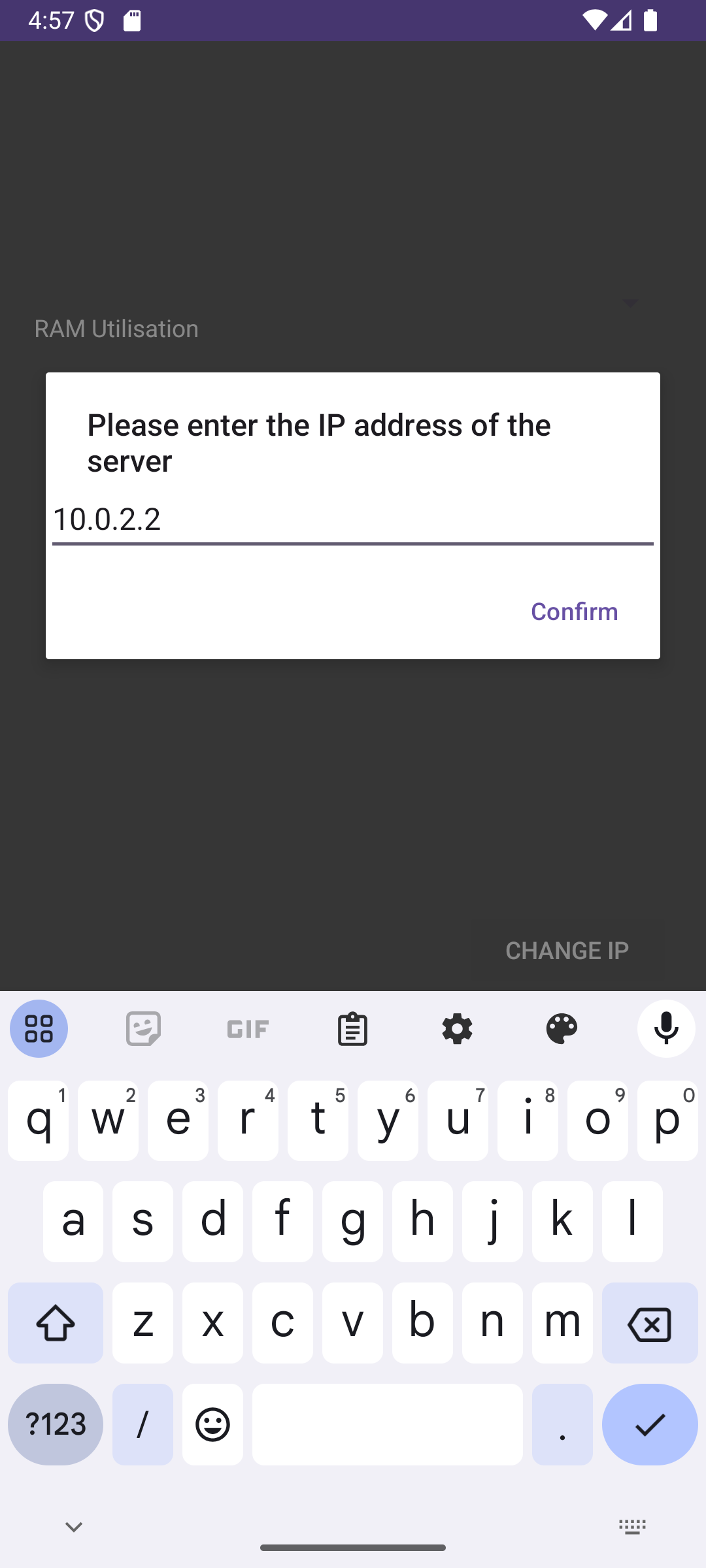
This app is designed to make use of the Server Management REST API, this being an application that checks the utilisation of a computers CPU and RAM. This is similar to services seen on servers that track how the hardware is being used over time, however, as the API is a simple Java app, it can be used on any machine.

This data is not displayed in an easy-to-understand way however and requires another application to do something with the data. This app is designed to query this API to receive the computers utilisation statistics and display it in a more useful graph format. This app is also particularly useful as it allows users to check how much load their computer is under from anywhere as long as they are on the same Local Area Network.

To use the android app, the user must first deploy the Server Management API, either running the Java files or with Kubernetes. They must also set their computers firewall to allow connections from port 9000. For extra security, they can specify which devices are allowed to connect on port 9000.



When the app is first launched, the user will first see a screen asking them to enter the IP address of the server that the Server Management API is running on. Without the correct IP, the app will not display anything. There is also some input validation involved as, if the user does not input anything, the app will display a toast asking them to input an IP. If the user inputs a string that is not in the pattern of an IPv4 address, i.e. X.X.X.X, the app will ask the user to enter a valid IP address. This piece of rejex prevents the user from accidentally entering an invalid input and causing errors. It also helps prevent any kind of injection attacks by ensuring that the input that is used for the request is nothing more than an IP.



After the user enters a valid IP address and the app receives a response from the API, the user will first see two graphs displaying the CPU and RAM utilisations from the past minute. The data points used to create this graph have been taken every second, granting a greater level of detail. Fragments are used for these graphs, allowing the code used to create the graphs to be reused as well as making it easier to implement new graphs if new utilisation details become available from the API. Above the graphs is a spinner, that allows the user to select from a set number of time periods, therefore allowing them to choose over what time they want to see the utilisation. It is important to note that for every time period above “Minute” the data points only use readings taken every minute. The reason for this is that, when, for example, the month time period is selected, 8100 records need to be sent along the network and this can take around a second to complete. If a reading for every second from the past month were to be sent, there would be 2,678,400 records. This would cause a large delay as all of these were sent over the network and have a serious performance impact on the device as it tried to render a graph with 2.7 million data points.

Even with only 8100 records, the graphs can be quite compacted and hard to read, therefore the graphs can be pinched and dragged to focus in on the area the user is most interested in. Custom labels are also used, as the default ones would display the number of seconds since 1970, not useful for most users. Instead, the number of days ago is displayed. Additionally, if the user wishes to see a larger graph, they can rotate their phone into a landscape view and the graphs will display in a much larger format and the user can scroll between the two.

Finally, if the user wishes to change the IP perhaps because they have multiple computers, they can press the Change IP button, bringing up the previously seen IP entry popup.