Volume Manager

**GitHub Username**: o4wcoder

Contents

[Description 1](#_Toc442800197)

[Intended User 2](#_Toc442800198)

[Features 2](#_Toc442800199)

[User Interface Mocks 2](#_Toc442800200)

[Screen 1 3](#_Toc442800201)

[Screen 2 3](#_Toc442800202)

[Key Considerations 4](#_Toc442800203)

[How will your app handle data persistence? 4](#_Toc442800204)

[Describe any corner cases in the UX. 4](#_Toc442800205)

[Describe any libraries you’ll be using and share your reasoning for including them. 4](#_Toc442800206)

[Next Steps: Required Tasks 4](#_Toc442800207)

[Task 1: Project Setup 4](#_Toc442800208)

[Task 2: Implement UI for Main Activity and Time Profile Detail Fragment 5](#_Toc442800209)

[Task 3: Create Volume Control Backend Services and Receivers 5](#_Toc442800210)

[Task 4: Implement Location UI 5](#_Toc442800211)

[Task 5: Implement Maps and Geofencing Backend Services 6](#_Toc442800212)

[Task 6: Implement Google Places API 6](#_Toc442800213)

[Task 7: Create Custom Library 6](#_Toc442800214)

[Task 8: Create Custom Widget 6](#_Toc442800215)

[Task 9: Final Features and Miscellaneous Items 6](#_Toc442800216)

[Task 10: Get App Ready for Release 7](#_Toc442800217)

[Optional Features 7](#_Toc442800218)

# Description

Don’t you hate it when that one person has their phone loudly go off during a meeting or at the movies? You don’t want to be that guy, right? Well, with the Volume Manager App you won’t! Volume Manager lets you set up time duration when your system volume settings will automatically change when you entire this time period. So say you have a recurring meeting from 3 pm to 4 pm every Wednesday at work; you can create a profile to repeatedly turn your volume settings to vibrate or off during that period every Wednesday so you won’t disturbed you coworkers if you get a call or any other type of contact. How about having a volume profile where you phone volume turns off every night at a specific time and then comes back on when you wake up? No more annoying email buzzes or beeps in the middle of the night!

With Volume Manager you can also create a volume profile around a specific location. By selecting your location on a map, you can create a fence around your location, that when entered will change you volume settings to your desired settings when you enter that location. The volume settings will return to the previous setting when you leave the location. This feature can make sure your phone is quite every time you enter your church or school for instance. You can also search for any location on the map and create your volume settings at locations even if you are not currently there.

The convenient widget will allow you to fully control your volume profiles from your home screen, or simply see if you are currently under one of your pre-defined volume profile’s control.

# Intended User

Volume Manager could be used by any one, but it greatly benefits people that routinely go to places where they should silence their phone like office workers and students.

# Features

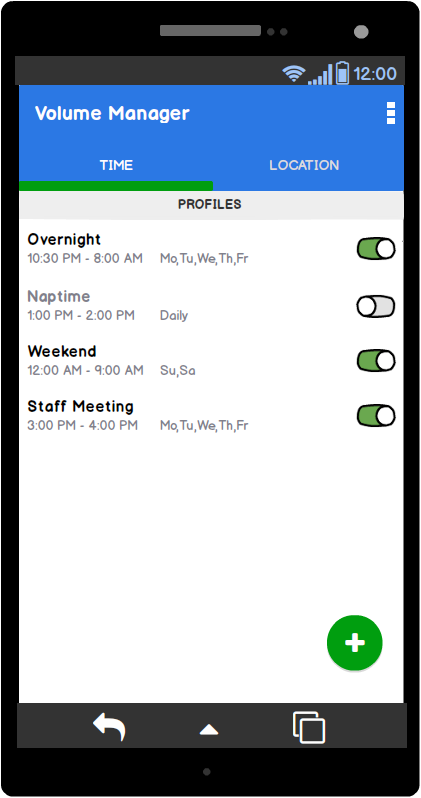
List the main features of your app. For example:

* Saves information
* Takes pictures
* Other features like that

# User Interface Mocks

These can be created by hand (take a photo of your drawings and insert them in this flow), or using a program like Photoshop or Balsamiq.

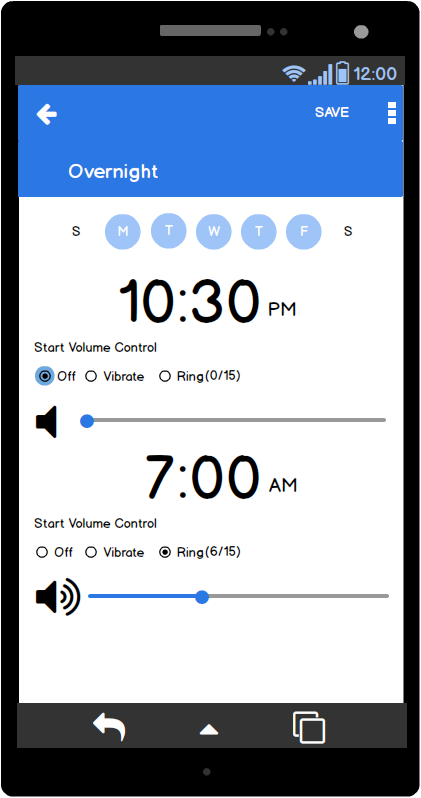
## Time Profile List



Replace the above image with your own mock [ click on the above image, then navigate to Insert → Image… ]

Provide descriptive text for each screen

## Time Profile Detail



# Key Considerations

## How will your app handle data persistence?

Data for each volume profile will be handled with a Content Provider. There will be one table for the main volume control settings. And a second table containing Location information.

Preference Menu settings will be stored in shared preferences. These will be the required settings.

* Enable Notifications
* Geofence Unit Measurements (meters or feet)

## Describe any corner cases in the UX.

For example, how does the user return to a Now Playing screen in a media player if they hit the back button?

## Describe any libraries you’ll be using and share your reasoning for including them.

For example, Picasso or Glide to handle the loading and caching of images.

# Next Steps: Required Tasks

## Task 1: Project Setup

* Install libraries for Google Play Services in build.gradle file and AndroidManifest.xml
* Get and setup required Google API Keys
* Create Profile Model class. This will contain all the settings for a single volume control profile.
* Create Content Provider
* Create AndroidTest functions to test the Content Provider. Verify it’s working before proceeding.

## Task 2: Implement UI for Main Activity and Time Profile Detail Fragment

The main Activity will consist of two tabs. One for Time profiles and the other for Location profiles. Each tab will contain a ListView that will display the list of profiles that have been created. A PagerView will connect the two tabs

Tasks:

* Create MainActivity UI that contains two tabs and a PagerView
* Add ListView for the Basic and Location Profiles. They will share the same ListView, even though they are on different tabs. However, they will require separate adapters.
* Create the UI for the Profile Detail Fragment. This is where you will set the start and end times of a profile along with the volume settings at each point
* Link Content Provider to Main Activity and Detail Fragment. The Main Activity will have the ability to add a new Profile Row to the Content Provider and the Delete a Profile Row. From the Detail Fragment you can add, delete and update a Profile row.
* Adapt UI for Tablets. Create Main/Detail UI for Landscape Mode using Material Design Elements

## Task 3: Create Volume Control Backend Services and Receivers

The Backend of the app will handle the changes in the volume settings. Each volume profile will have a start and stop event for changes in the volume. This will be handled by the AlarmManager class. Each start and stop times will be registered with the AlarmManager to be fired off when the times are reached. When they are fired off, they will be handled in the background by an IntentService. It is hear in the IntentService’s onHandleIntent() method that the volume settings will be changed. Also here is wehre notifications will be sent if enabled.

Tasks:

* Create IntentService to handle the start and stop times of the volume profile
* Create AlarmManager that will register all the volume profiles start and stop times and register with the IntentService.
* Change volume settings, using the AudidoManager when IntentService’s onHandleIntent() is called.
* Send out notification of change in volume. This setting will be on always until the Preference Settings Menu is created to be able to turn it off.
* Use a BroadcastReciever to reset all the stored profile’s start/end times in the AlarmManager.

## Task 4: Implement Location UI

Tasks:

* Create Google Maps Activity
* Adapt UI for Tablets. Create Main/Detail UI for Landscape Mode using Material Design Elements

## Task 5: Implement Maps and Geofencing Backend Services

Tasks:

* Add Map Markers to Map when the user presses on a location. Have the title of the profile, given by the user, displayed above the marker.
* Create and IntentService to handle enter/exit of Geofence.
* Use BroadcastReceiver to turn back on Geofences when the phone reboots.

## Task 6: Implement Google Places API

The user will not only be able to set up Location Profiles by selecting a spot on a map, they will also be able to search for any address using Google Places API

Tasks:

* Create search bar at the top of the map for entering and address
* Implement backend of Google Places API

## Task 7: Create Custom Library

Tasks:

## Task 8: Create Custom Widget

There will be two widgets for this app. One small widget to just show if the user is currently under volume control, and a large widget that will contain the list of time and location profiles.

Tasks:

* Create small widget to show if the user is currently under volume control.
* Create large widget that will contain the list of time and location profiles.

## Task 9: Final Features and Miscellaneous Items

This section includes and final tasks to make the app production ready.

Tasks

* Create Preference Settings menu
* Verify all strings are located in the strings.xml file and uses RTL
* Create Content Descriptions on relative components on the UI

## Task 10: Get App Ready for Release

All tasks that should be done in the final stage for release to the Google Play store.

Tasks:

* Create Free and Paid Versions. Add Ad to the bottom of the MainActivity under the Profile ListView
* Create App Icon
* Create Signed APK for App
* Release Free and Paid versions and release each to the Google Play Store.

# Optional Features

These are features that are optional and may be implemented on this first phase if time permits. If not, they will be moved to Phase II development.

Tasks:

* Extend to Wearable
* Create Intro Tutorial Screens
* Create a third type of profile called “Calendar”. This will link to your Google Calendar and perhaps other, so you can set volume controls for your meetings and events.