Capstone Stage 1

GitHub Username: DallinWilcox

Turn it Down!

Description

Constantly asking your kids to turn down the volume on their devices? Now you can manage volume on their devices remotely. Turn it Down allows you to have peace of mind and your kids to be less interrupted, plus, you know they won't be blowing out their eardrums.

Intended Users

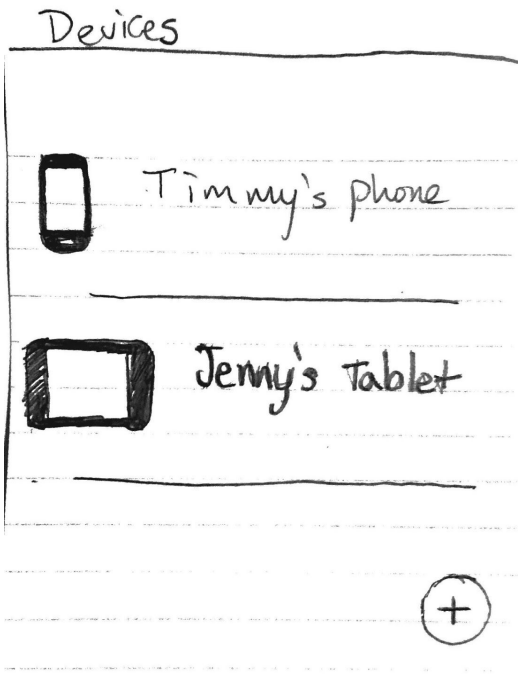
- Parents
- Owners of multiple devices
- Digital classroom instructors

Features

- Enroll a new device
- Set the volume remotely for devices you've configured
- Check the current volume settings for a remote device

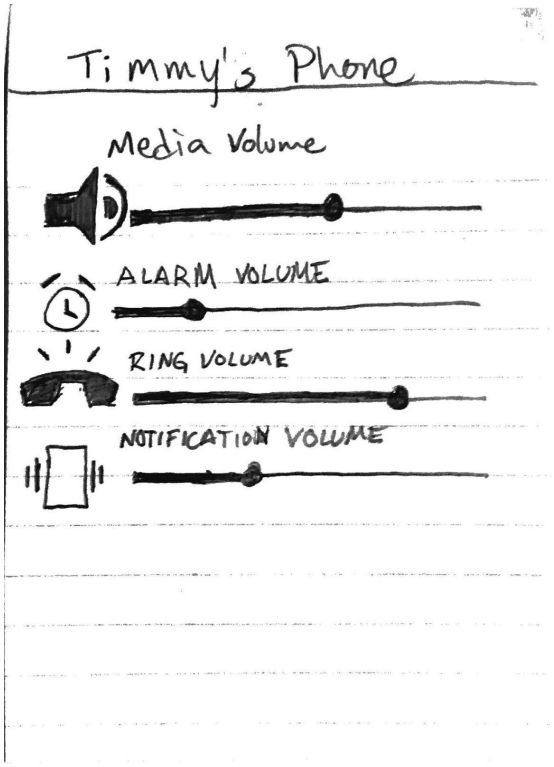
User Interface Mocks

Device List Activity



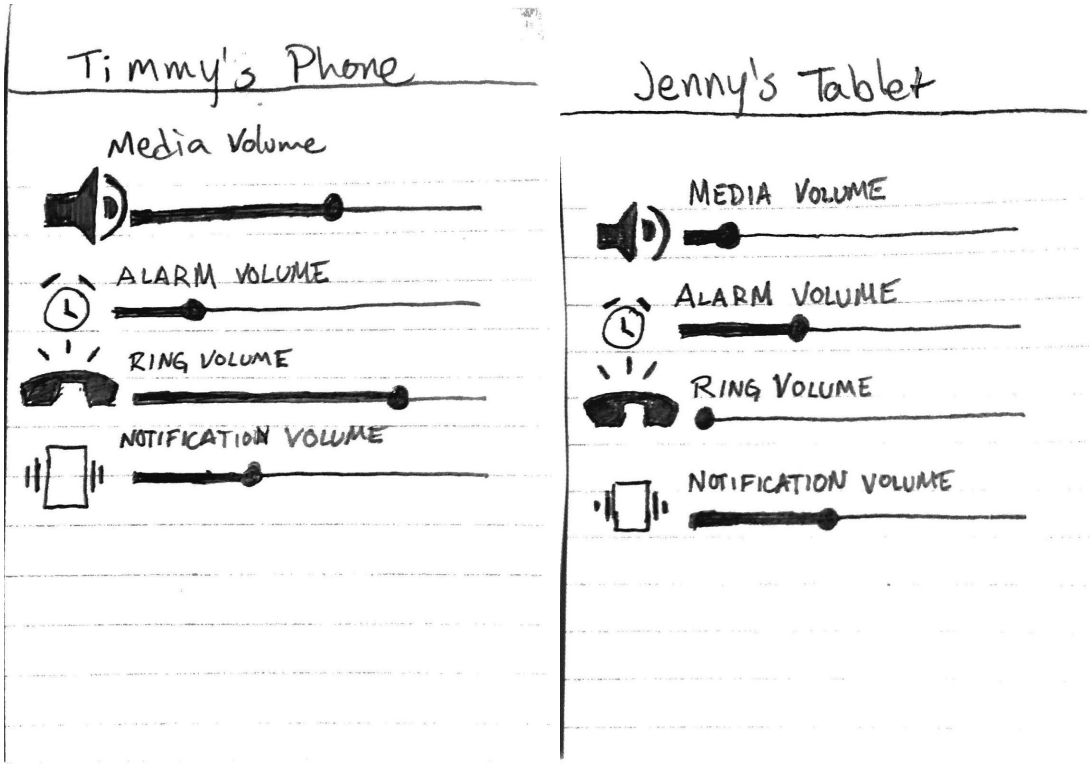
Default view on phones. List shows all enrolled devices. Tapping on a row takes you to the Device Detail for that device.

Device Detail Fragment



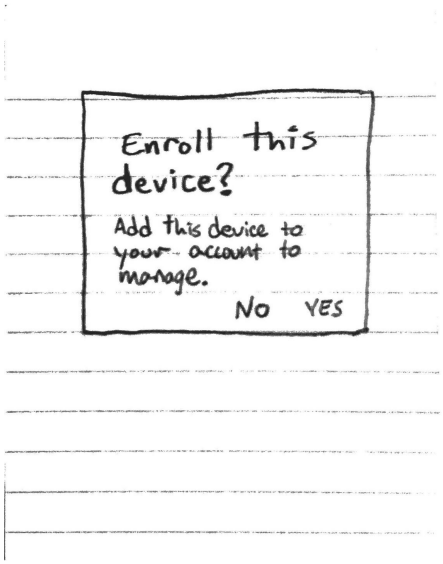
Shows the details of the volume settings for the selected device with icons and headings for each type of volume accompanied by a slider to adjust volume.

Tablet View



Default view for a tablet. Shows a grid of multiple Detail View Fragments of the volume settings for enrolled devices.

Enroll Device Dialog



Prompt to enroll a device to the user's account.

Key Considerations

How will your app handle data persistence?

The app will use a Content Provider to front Firebase Data Storage to store data about enrolled devices and provide that data to both a loader for the UI as well as a to the widget.

Describe any corner cases in the UX.

Fairly standard Master-detail design, such that invoking either back, or up from the device detail view will return the user to the device list. Any other dialogs such as authentication or permissions requests will be dismissed/finished by the back button.

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

Firebase SDK - to interface with Firebase Google Play Services.

Describe how you will implement Google Play Services.

Firebase Cloud Messaging - used to send messages to devices to trigger them to update their volume settings.
Firebase Authentication - used to authenticate the user and to ensure they are controlling only their registered devices
Firebase Storage - used to store info about registered devices

Next Steps: Required Tasks

Task 1: Project Setup

- Create a new Android Studio Project
- Set up a signing key
- Install the Firebase SDK.
- Create a Firebase Project and add the app to it in the Firebase console

Task 2: Implement UI for Each Activity and Fragment

- Build UI for Device List Activity
- Build UI for Tablet Grid Activity

- Build UI for Device Detail Fragment
- Build UI for Authentication Activity, see Firebase docs for reference implementation
- Take care to keep accessibility in mind, including implementing appropriate D-pad navigation and content descriptions

Task 3: Implement Firebase Authentication

- Refer to https://firebase.google.com/docs/auth/android/start/

Task 4: Configure Firebase Data Storage and Build a Content Provider to front it

Setup and configure Firebase Data Storage

- Follow the instructions to get started at https://firebase.google.com/docs/database/android/start/
- Define the data structure in json

Build a content provider to make data available from Firebase to the content loader and widget.

Task 5: Implement App Engine/Firebase Backend

The backend will need to handle calls to push a notification to a remote device to retrieve.

- Refer to this Google I/O 2013 Demo for an example implementation using the FCM's predecessor, GCM.
- Implement an endpoint to send a notification to retrieve current settings for the remote device. Input would be notification ID for the target device.
- Implement an endpoint to send a notification to pass settings to a remote device. Input would be notification ID for the target device and a data object representing the settings to pass, including an indication of whether the settings are being returned from another device or sent to be set on the target device.
- Wire up UI to call the service to retrieve and send settings from/to a remote device.

Task 6: Build a Service to handle volume settings in the background

- Add manifest entries to declare your service
- Build functionality into the service to retrieve and set the appropriate volume settings
- Ensure the appropriate permissions for controlling the volume settings have also been added to the manifest

Task 7: Build out the client to handle Firebase Cloud Messaging notifications received

- Implement FCM device registration and token handling
- Build the receiver service to handle messages
- Wire the receiver service to invoke the volume settings service created previously

Task 8: Build a Widget

- Create a widget layout
- Create an AppWidgetProvider and tie it to the content provider for data about the user's devices.
- Add the appropriate entries to your manifest for the AppWidget provider.

Suggestions to make your project stand out!

- Implement volume thresholds feature to allow thresholds to be set for a device. This could consist of a service with listeners/broadcast receivers to be invoked whenever volume settings are changed locally on the device and check them against the threshold. When settings outside of the allowed threshold are detected, reset them to within the threshold.
- To improve performance, particularly in cases where there are a large number of controlled devices, show a grid of icons for devices in the tablet view that expand into the full Detail View Fragment when selected.