PEGASUSOS

PRIVACY AND SECURITY FOCUSED MOBILE OS

PROJECT REPORT

Submitted by

MOHAMMED ALTHAF T

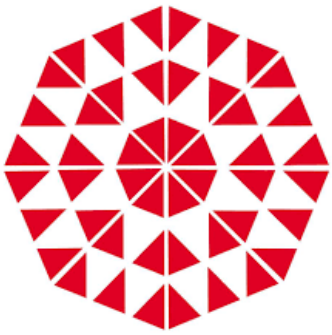
LKMC18MCA026

to

the APJ Abdul Kalam Technological University in partial fulfillment of the requirements for the award of the Degree

of

Master of Computer Applications



Department of Management Studies & Computer Applications

KMCT College of Engineering

Kallanthode, NITC P.O, Kozhikode-673601

JUNE 2021

DECLARATION

I undersigned here by declare that the project report "**PEGASUSOS PRIVACY AND SECURITY FOCUSED MOBILE OS**", submitted for partial fulfillment of the requirements for the award of degree of Master of Computer Applications of the APJ Abdul Kalam Technological University, Kerala is a bonafide work done by me under supervision of **Mrs. Jittumol George** (Assistant Professor, MCA). This submission represents my ideas in my own words and where ideas or words of others have been included, I have adequately and accurately cited and referenced the original sources. I also declare that, I have adhered to ethics of academic honesty and integrity and have not misrepresented or fabricated any data or idea or fact or source in my submission. I understand that any violation of the above will be a cause for disciplinary action by the institute and/or the University and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been obtained. This report has not been previously formed the basis for the award of any degree.

Place: Kallanthode Mohammed Althaf T

Date:

DEPARTMENT OF MANAGEMENT STUDIES & COMPUTER APPLICATIONS

KMCT COLLEGE OF ENGINEERING

Kallanthode, NITC P.O, Kozhikode-673601

CERTIFICATE

This is to certify that the report entitled "**PEGASUSOS PRIVACY AND SECURITY FOCUSED MOBILE OS**" submitted by **MOHAMMED ALTHAF T** (LKMC18MCA026) to the APJ Abdul Kalam Technological University in partial fulfillment of the requirements for the award of the Degree of Master of Computer Applications is a bonafide record of the project work carried out by him under our guidance and supervision. This report in any form has not been submitted to any other University or Institute for any purpose.

Internal supervisor

Project coordinator

External supervisor

Head of the department

ACKNOWLEDGMENT

I would like to take this opportunity to extend my sincere thanks to people who helped me to make this project possible. This project will be incomplete without mentioning all the people who helped me to make it real.

First and foremost I thank **Prof. Dr. M D Sreekumar** (Principal of KMCT College of Engineering) who gave me all support to this project. I thank **Mr. Ajayakumar K K** (Head of Department, MCA), for providing all the facilities and resources for my project. I would also like to express my gratitude towards **Mrs. Sabna T.S** (Assistant Professor, MCA), Project Coordinator for the continuous support, guidance and supervision without which the project wouldn't have been a reality. I thank **Mrs. Jittumol George** (Assistant Professor, MCA), Project Guide for the valuable guidance and inspiration throughout my work. I would also take this opportunity to thank all my friends who took time out of their busy schedule to encourage, support and motivate me which has been the key reason for the successful completion of this project.

Above all I thank God, The almighty for his grace without which it would not have been possible to complete this work in time.

ABSTRACT

Android devices have become a part of our life due to many things. The ease of use, the amount of features it provides, customization makes Android one of the best OSes for our smartphones. OEM or Original Equipment Manufacturer releases a number of Android devices each year. Each device will have different SoC (System on a Chip), Wi-Fi modules, camera setup etc. All devices released by a particular OEM will have almost the same User Interface. The custom User Interface on top of the stock Android has many advantages and disadvantages. Even though the custom UI provides many extra features it will also have many bugs.

This project **PEGASUSOS PRIVACY AND SECURITY FOCUSED MOBILE OS** intends to build a privacy and security focused mobile OS for **Google Pixel XL (marlin)**, Which is sold by google on 2016 with Android Nougat and got updates up to Android 10, then discontinues support. In this project i will be building Android 11 not only for Google Pixel XL, but also giving future support for other older devices from different OEMs like Motorola and Xiaomi.

This OS will not be including any Google specific apps such as Google Play Mobile services (GMS), in benefit of not tracking user activities by google. For this OS to work perfectly we had to sacrifice Google Play services to improve our privacy. Since they are alternatives like F-Droid. Its an install-able catalog of FOSS (Free and Open Source Software) applications for the Android platform. For bringing a new Android Based custom ROM, We need to clone AOSP and modify the components needed to to customized it as per our needs. For this device we need to create a custom device binary tree for a test device in order to create a custom firmware for the device.

The device binary tree will have information about the test device. Like its device name, brand, many modules and SE-Linux policies if needed. This device binary tree might contain some errors which will be fixed once the kernel is brought up for the device. Since we are more focused on security over customization, we will be using security tightened SE-Linux Policies (Security-Enhanced Linux). It is a security architecture for Linux systems.

Contents

List of Figures

List of Tables

Chapter 1

INTRODUCTION

1.1 General Background

When we buy a new phone, we own it, with a few conditions. We can't remove certain apps (Bloatware) and services that preinstalled by the OEMs. Also your phone won't let you access system files. And we can't change aspects of the interface. It's technically our device, but there are many tweaks that we don't have permission to make. Installing a custom ROM puts you in control of our own hardware. We can swap out one operating system for another, tweak more settings, and change the experience until we are as happy. Then our phone is truly ours.

1.2 Objective

This project 'PegasusOS' aims to bring up a privacy and security focused mobile OS. That users can install by themselves. Now users will be empowered to take tweaking their phone to the next level. Android is already very much customization, letting we swap out the launcher, replace icons. But a custom ROM allows us access to more. Smartphones tend to come with more software installed than we need. Unlocked phones may still come with ways more apps which we might have zero interest in. Even Pixel phones come loaded with Google apps that users may not want. This custom ROM will not be including any Google Mobile services (GMS) or Bloatware, With benefit of not tracking activities by anyone.

When we buy a PC, We know that we are getting a piece of hardware that can be used for years. Generally speaking, We don't have to replace/Upgrade the machine until we get tired of it or its get slower for our use. That's not the case with phones. Every device we see in a OEM store comes with an expiration date. Most phones will see updates for two years and some won't even get half. This leaves user stand on old versions of Android, where they're vulnerable to security exploits. In the case of PC users, they can install switch OS after upgrading hardware. Mobile phone users can't replace anything other than battery and memory card. Unlocking bootloader and flashing custom ROMs will allow users extend their phone lifespan and security and less vulnerable on the internet.

Chapter 2

SYSTEM ANALYSIS

2.1 Existing system

Android is a mobile operating system based on Linux kernel and other open source software designed for touch screen mobile devices such as smartphones and tablets. Android is developed by developers known as the Open Handset Alliance and it's commercialized by Google. It is a free open source software, its source code is known as AOSP or Android Open Source Project which is primarily licensed Under Apache License. The version history of Android began with the public release of the Android beta on November 5, 2007. The first commercial version, Android 1.0, was released on September 23, 2008. Android is continually developed by Google and the Open Handset Alliance (OHA), and it has seen several updates to its base operating system since the initial release .The Last release was Android 11.0 on 8 September 2020.

Android has been the best-selling OS worldwide on smartphones since 2011 and on tablets since 2013. As of 2021, it has over 3.8 billion monthly active users, the largest installed base of any operating system, and as of January 2021, the Google Play Store fea-

tures over 3 million apps. Android provides cross-platform means Android application can run on any type screen, size and resolutions including mobile phones, tablets etc. Android provides the unified approach to develop the Android application. It means developers need to develop only for one Android device and then application can run on different devices powered by Android. Each Android device comes with a particular Android version and gets updates for up to 2 to 3 years for flagship devices and 1 to 2 years for mid-range devices, some of them don't even get any updates. And also delay in major Android updates and security updates will be there. The security in older devices will be weak compared to newer devices and Android updates. Security is also depending on the user activities on devices and apps pre-loaded in the phone by OEM (Original Equipment Manufacturer).

Most OEM and users Prefer customization over privacy and security. Heavy skinned stock OEM ROM will be time consuming to rebase into newer Android versions so it will create delay in updates to consumers. Custom skins might also be poor in ram management and might have performance issues. Also due to the presence of custom skins, delay in major Android updates and security updates will be there.

Disadvantages:

- Delay in major Android release updates.
- Issues with performance and compatibility.
- Most often the OEM skin on top are poorly coded and can result in more bugs and issues.
- OEMs can't even provide the monthly security updates released by Google in a timely manner due to their modification to the Android system.

2.2 Proposed system

I will be creating a custom ROM just like how Android is meant to be with only slight modifications in terms of UI and additions. This will ensure that there won't be any performance drops or stability issues. I will be implementing it by modifying the Android source code according to our needs. Google releases the source code of Android for further development and I will be using that to provide a bleeding edge Android build for a device which has not been updated for a long time and still runs an outdated Android version and security patch. Also improve the privacy and security of the OS from the bottom up, security of both the OS and the apps running on it by adding various features like the permission restrictions when the device is locked (like connecting USB peripherals). Along with more complex user-facing privacy and security features. Also features like disabling all primary sensors of the device in one click. I will never include any Google Play Mobile services (GMS) and included OSS alternative MicroG in order to protect user's activity on the phone. It will also improve battery life on phone and less background apps will be running.

Advantages:

- No Google apps, so no tracking.
- Support for older and newer devices.
- More security than normal OEM Stock ROM.
- Reduce performance drops.
- Stock Android experience.
- Amount of bloatware will be less.
- Major Android updates can be provided even before OEMs release it.
- Security updates or patches can be provided the same day Google releases it.

'ROM' stands for 'Read only memory'. A custom ROM replaces your Android operating system. This is the most popular reason for installing a custom ROM. The Manufacturers never updates their older Android phones even if it is compatible to run new versions of Android and other updates. It is done in order to reduce the old device value and to improve their new devices productivity. So custom ROM is the ticket for getting new versions of Android.

This is an AOSP based ROM. Its mission is to offer the maximum possible stability and security, along with essential and useful features for the proper functioning of the mobile device. This ROM extends the functionality and lifespan of mobile devices. The custom ROM is aiming to extend the life of a mobile device by working on enhancing the already existing beauty of Android. The main principle of this Custom OS is to reduce the amount of electronic waste by providing latest update Android and other security packs to old Android phones to improve the performance of the phones. This custom OS provides better performance and battery backup to the mobile devices. Handpicked features beautifully packed in one OS.

Why Custom ROM ?

- To get latest version of Android

This is the most popular reason for installing a custom ROM. The Manufacturers never updates their older Android phones even if it is compatible to run new versions of Android and other updates. It is done in order to reduce the old device value and to improve their new devices productivity. So custom ROM is the ticket for getting new versions of Android.

- To replace Manufacturer skin with a stock version of Android

Manufactures skin or versions of Android can be replaced by stock Android which create a clean look or the user can customize the device to his wishes. The main aim is to provide unique user interface and optimum performance and improved battery capacity.

- Eliminate unnecessary applications (bloatware)

When you purchase a phone, it often comes with unnecessary applications (bloatware) these applications cannot be removed by the user because of the prohibitions of manufacturers or company, these applications waste a lot of disk space which can be used for other purpose. So by installing a custom ROM we can get more disk space or storage.

- Add Additional Features and System Options

Custom ROMs offer features not found in stock Android and many tweaking options you can't get elsewhere. For example, a custom ROM may allow you to, Install skins to customize how your entire Android operating system looks.

Customize the quick settings menu Android includes to add your own most-used settings shortcuts.

- Security

Your data, your rules. With powerful tools such as privacy guard, you are in control of what your apps can do whenever you want. This will help your device and warn about possible threats

Features

These are the basic features which will be included in the ROM.

- Protect USB Port

This options allows users to select different options to enable or disable peripheral connections while device is locked or unlocked. There are three options to deny all USB, allow new devices while unlocked or allow even while locked. This feature can be used to directly disable USB access within kernel. All peripherals including mouse, keyboard or OTG USB will be disabled.

- Permissions Manager

This options allows us to view all the permissions and the apps that are using these permissions and the data access for each apps. We can also allow or deny permissions by clicking them. We can select each app and disable permissions like camera, files, contacts, etc. And also control data usage of app like, WiFi, mobile data.

- Pattern Lock size

This options allows us to select desired pattern lock layout size. The bigger the layout there's more ways to lock user device. We can also hide lock screen pattern dots or hide failure red lines or pattern draw path to avoid pattern visibility to others.

- Scramble pin

This options allows us to scramble the pin layout in lock screen. Which allows us to enter pin in different order each time we unlock phone. Less vulnerable with group of people considering normal layout. Order will be different on each unlock.

- Bubbles

This feature allows us to use Android-11's chat heads. Its similar to Facebook messenger chat heads, but supports more apps and can be used for contacts specific. It is feature introduced in Android-11 for third-party apps. Now its enabled for more apps like WhatsApp, telegram and other SMS apps.

- Data Backup

This feature allow us to backup our data easily with SeedVault. Its a user-friendly encryption using a mnemonic phrase. We can backup the data into cloud, USB flash drive and restore later with the phrases. We can upload data into our cloud storage or USB drive. We can restore the data on initial startup of device or after setup by going to backup option in settings.

- Hidden & Protected apps

This feature to lock and hide apps in launcher. It uses screen lock password as protection, so no need to set separate password for this option. We can hide apps in the drawer and lock the apps from launching from the launcher.

- Icon packs

This feature allow users to install third-party icons packs and customize their launcher icons with ease. We can install third party icon packs from stores provided. Which allows us to change icons in launcher.

- Theme phone

This apps is provided by google, but disabled by default for AOSP device. This app can change device's fonts, accent color, icon shapes, launcher icon shapes,etc.. also change clock faces, styles and wallpapers. There will be different fonts, colors, icons shapes which isn't available in stock. Also different types of clock faces for lockscreen. Users can save their default themes and reuse.

- QS Tiles

Three types of QS tiles which doesn't exist in AOSP. First one caffeine allows to temporary set screen out time and sync allows to sync data from internet connected apps to work properly. And heads up allows to disable notification from status bar to popup. These features allows to control screen timeout temporarily and sets backs to previous state and background data sync of apps like contacts, calendar and cloud apps.

- Partial Screenshot

This feature will allow us to take partial screenshot of the screen by short click and long click for full screen shot. We can use this option to take screenshot of a specific area by selecting. Which is useful in cases which don't want to take full screen shots.

- Link Ringer & Notification volume

This feature will allow us link ringer and notification volume. So we can control both with a single slider. No need to control seperatly. We can toggle it on or off as per user needs. This features allows to control both of the volume slider at the same time.

- Volume Panel

This feature allows users to control the volume of alarm, media and ringer volume with single panel. This was used before Android-9.0 and discontinued in post Android versions. By default there was no expandable panel. Now we can control without going to deep into volume settings.

- Record calls

This feature allows to record all native voice calls and save them to the phone storage. Including the quality of the calls and record type can be adjusted. We can start call recording from dialer when call is ongoing and change record quality in the dialer settings. either AMR/AAC.

- Disable sensors

This tile will allow users to disable sensors on the phone. Mostly it disables cameras and microphones. This will help users to be not monitored by any means or application in background. It doesn't disable system apps like dialer, which is an essential app for calling, so microphone will be working in that app. But rest of the sensors like proximity, orientation will be disabled.

- Panic trigger

”Panic button” that can send it’s trigger message to any app that is a ”panic responder”. Such apps can do things like lock, disguise themselves, delete private data, send an emergency message, and more. This helps to set an action on emergency times.

- Provide updates

This app will allow users to receive updates from the developer. This can improve the device security and future bug fixes will be given through this by the developer. Also get updates and install apps from the store. Which provided, F-Droid and Aurora-store (alternative for Google Play store).

2.3 Module Description

2.3.1 Operating System

- Bring up AOSP
- Add Support for older devices
- Add features
- Improve security
- Create user friendly interface
- Provide guides to build

2.3.2 Kernel

- Bring up kernel from OEM Source
- Add security features
- Add wireguard support
- Add USB Block feature
- Upstream kernel

2.3.3 Device Tree

- Bring up Device Tree on basis of device specifications
- Add security features
- Add additional features to device
- Fix issues with compiling

2.3.4 Vendor

- Bring up Vendor from device's firmware
- Remove unwanted bloats
- Add additional features
- Fix issues with compiling

2.4 Feasibility Study

2.4.1 Operational Feasibility

Proposed system is beneficial only if they can be turned into a running system that will meet the operating requirements like phone and bootloader status. The users from current system must be showing a tendency to change from that. Because current system need more man power and also it is not in a user friendly manner. An estimate should be made to know how strong the reaction of user is likely to have towards the new system. Users must know about operating system and have basic knowledge of how its working. If the user has intention to fiddle with phone and its operations, then the proposed system is operationally feasible.

2.4.2 Technical Feasibility

Evaluating the technical feasibility is the trickiest part of the feasibility study. It is that whether the available resources are enough to carry out the project i.e., both hardware and software configuration and other equipment that are in hand. The proposed system both hardware and software requirements have been specified in system configuration. And it is sure that this project does not need resources that were not available. Thus, the proposed system is technically feasible.

2.4.3 Economic Feasibility

The project is economically feasible as Android, JAVA, C, C++ and python, etc. And other tools like Android SDK and NDK is freely available. Hence the project counts with no extra cost and its benefits outlays the investment. Economic feasibility determines whether the proposed system is capable of generating profit for an organization. It involves cost incurred on the development team, estimated cost of hardware and cost of performing feasibility study and so on, this OS is developing with the available resources and necessary hardware equipment to build. Since cost of input for the system is feasible. The output of the OS is always a profit for the user. This OS doesn't cost any charge from the user who is using it. Hence it is economically feasible.

2.5 System Environment

- Languages used : C++, C, Python, Shell, Java and XML

- Tools used : Android NDK, SDK, Platform and Build tools

- Utility: GNU Make, Soong build system

2.5.1 Minimum Requirements (User)

- Bootloader unlocked mobile phone

- Linux or Windows Operating System

- Android Platform tools Installed on PC

- Stable Internet Access

2.5.2 Minimum Requirements (Developer)

- Processor : Intel i5 CPU or AMD Ryzen 5 or newer

- Memory : 16 GB RAM

- Storage : 500 GB HDD/SSD (SDD Recommended)

- Operating System : Ubuntu 20.04

- Stable Internet Access

2.6 Actors and their roles

2.6.1 User

- Install OS with ease
- Report bugs and issues
- Use OS and features
- Support developer and community

2.6.2 Developer

- Provide Installation guides
- Provide Monthly security patches
- Support device long-term
- Fix bugs and issues
- Give technical support to users

Chapter 3

METHODOLOGY

3.1 Introduction

This project follows Agile methodology. Agile software development comprises various approaches to software development under which requirements and solutions evolve through the collaborative effort of self organizing and cross-sectional teams and their customers/end users. It advocates adaptive planning, evolutionary development, early delivery and continuous improvement and it encourage rapid and flexible response to change.

3.2 UML Diagrams

3.2.1 Activity Diagrams

- User

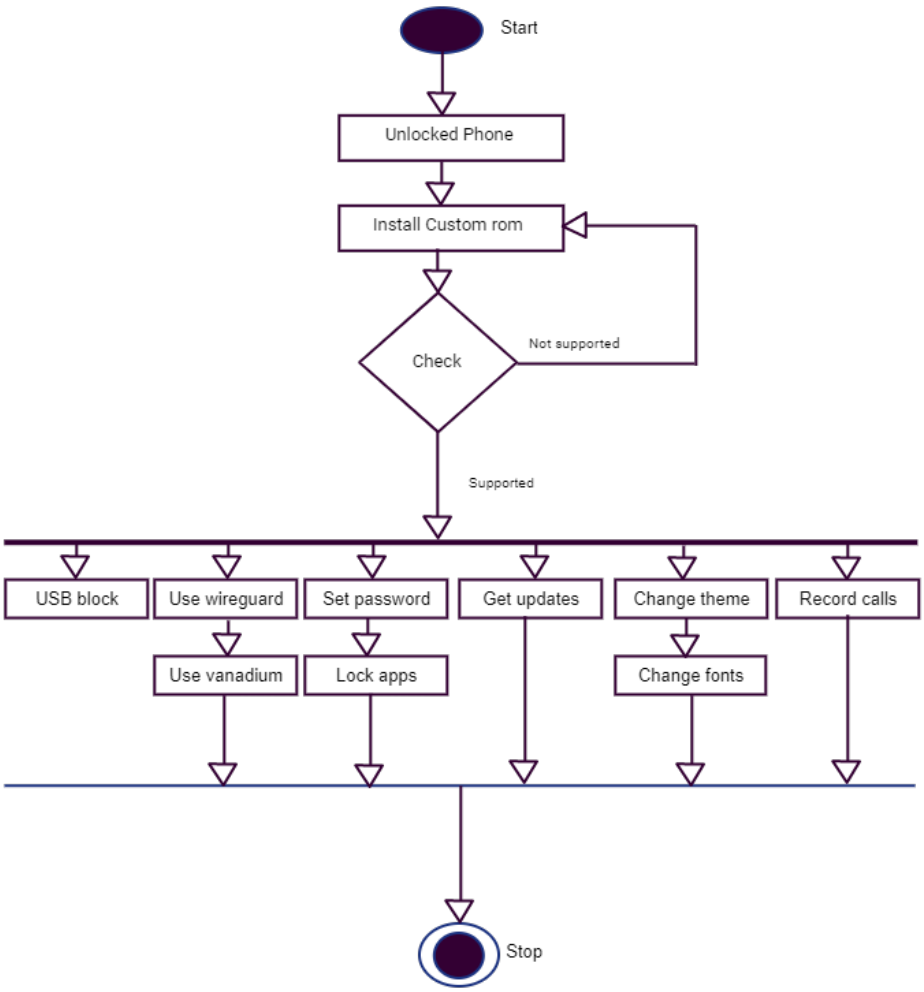


Figure 3.1: User's Activity Diagram

- Developer

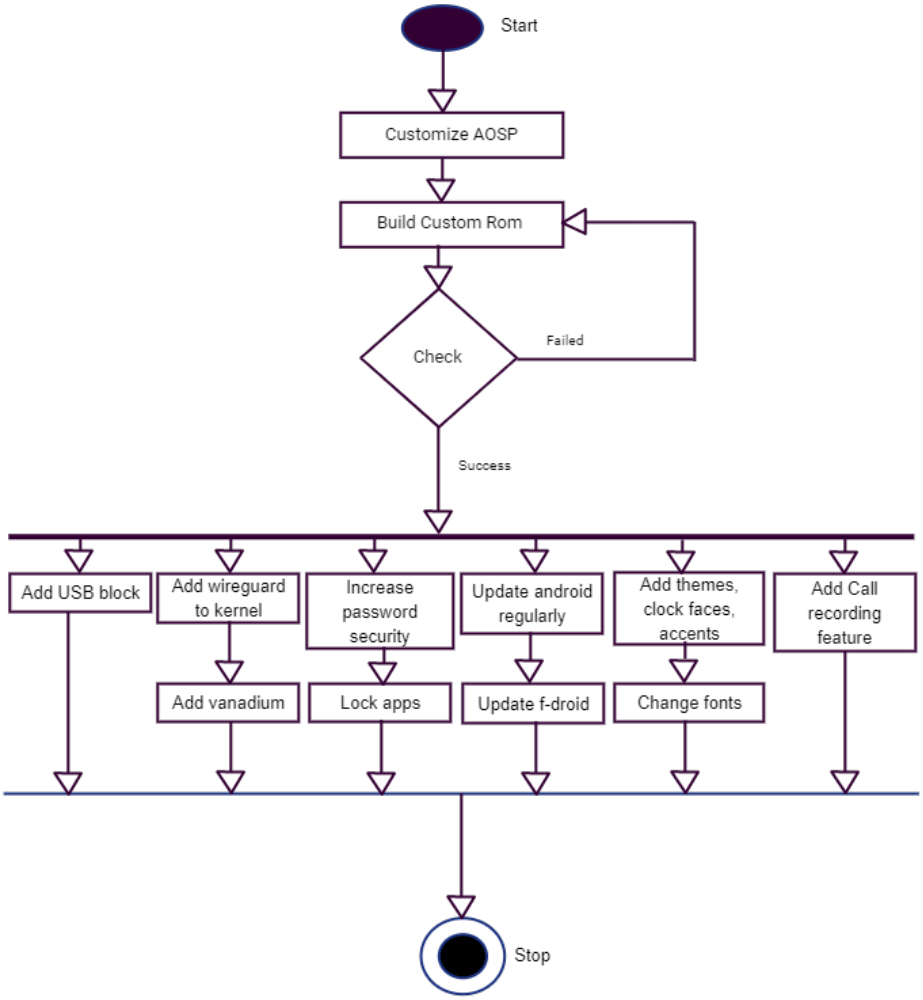
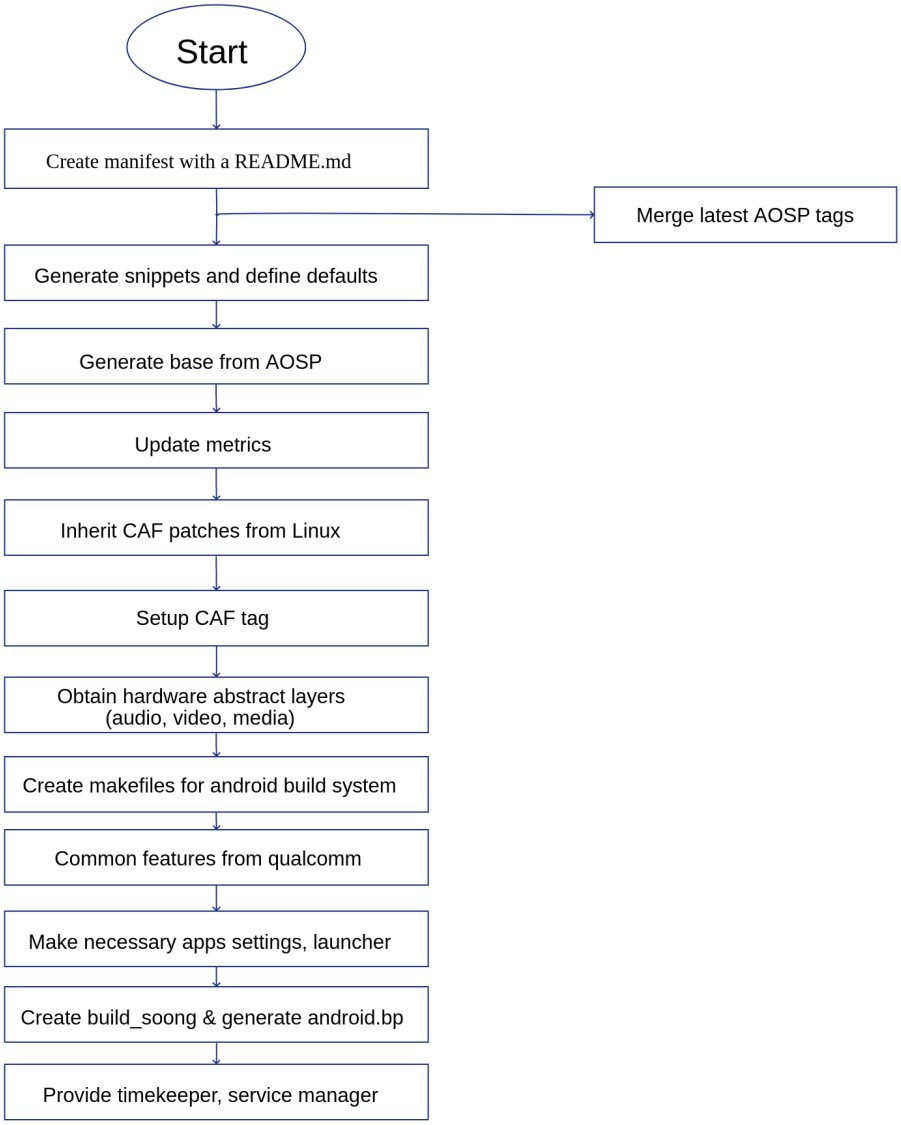


Figure 3.2: Developer's Activity Diagram

3.2.2 Usecase Diagram

Figure 3.3: Usecase Diagram

3.3 Program Flowchart



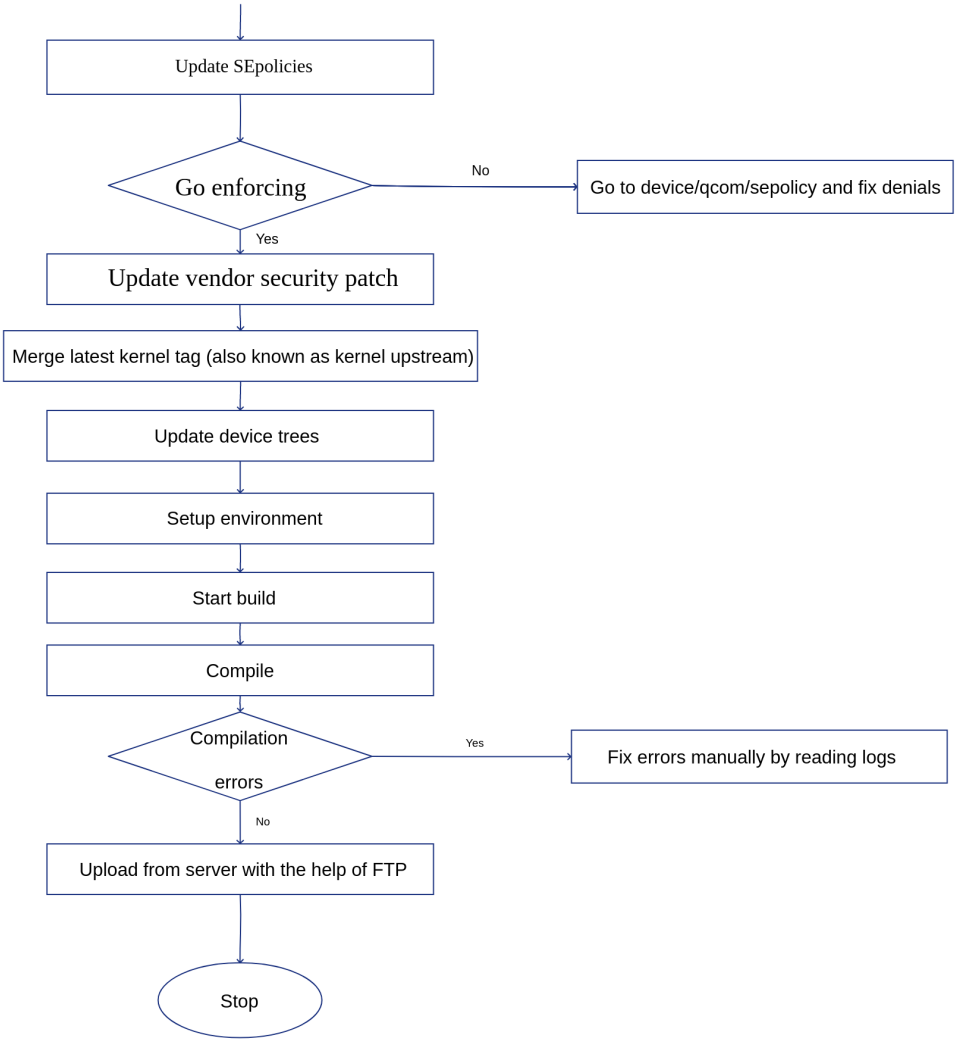


Figure 3.4: Flowchart

The Base source code is obtained from the Android Open Source Project (AOSP) itself. To begin with, few repositories are forked primarily including frameworks_base, package_apps_settings, android_manifest etc.

Defining manifest values is important as we initialize repos as a whole batch for ease. Already built and few basic application repositories are tracked from AOSP itself and except them we track from our source code with our modifications. This is defined in snippets directory as XML files.

Then, open framework_base repository using terminal, open protocol folder, then update metric.proto file. This is done because every fragment, activity has a metrics, for logging purpose and for all the custom activity, for added fragments. For this purpose a ROM specific matrix is used. This is important to identify and fix bugs that are present in the Operating System. After defining the metric value, obtain settings application from AOSP. The manifest stores the essential information about the application to the Android system, so that it can be called at anytime the system wants it. Size of manifest file varies from app to app, and is proportional to size of app. A typical settings application will have 7500 lines in manifest. The 'About us' section is added in Settings application as an XML file and inherit its values from res/values. Next step is adding ROM logo to 'about' section in res/layout/firmware_version.xml file. For selection of accent themes, a theme directory predefined is called into the vendor. To add features, primarily they must be defined in the manifest JavaScript. For example, in-order to set a 6*6 or 5*5 screen lock pattern as part of advanced security, define them as an activity inside manifest file. Then headover to res/drawable folder where lockscreen data base files are located. It is defined in an xml file. So create two more files for 5*5 or 6*6 pattern, named ic_security_pattern_6*6.xml and ic_security_pattern_5*5.xml. After adding necessary useful features, save them. Then tabulate all the added features or extensions into a sub application inside settings. For instance, bring all the navigational customizations into 'navbar settings'.

Next step is to set up the CAF tag and merge the latest tag into kernel head. The tag value is chosen in such away that it matches the SoC used in the device. Use of CAF gives a smooth and lag free experience to the OS. Then obtain hardware abstract layer (HAL). HALs are important for the proper functioning of audio, video and media files of the device. For all these three functions, three specific HALs are used. Next step is to generate build and build_soong repositories from AOSP. And also inherit common features from qualcomm if the device uses a qualcomm chipset.

Then clone necessary applications from AOSP. Including settings, launcher, calendar, calculator, etc. Then generate android.bp in build_soong and setup the proper build tags and script for the OS. Then service manager is initiated and timekeeper service is set up. Next step is updating SE-policies with the help of security enhancement Linux. This will enhance the security level at its peak. Then upstream the kernel to its latest and update the device tree, common tree, vendor tree and kernel tree of the device to the latest Android version. So there will be denial of service. Each and every denial that comes have to be fixed when SE-Linux is enforcing. If the source code and device specific source code is completely setup and error free, then the environment is setup and start build. Make sure that there is sufficient space for building and this building takes different times depending upon the specifications of cloud server. Now perform compilation. If error occurs, fix it manually by reading logs. After successful compilation, upload the output to a cloud-server host with the help of FTP access. (eg: Sourceforge, Androidfilehost)

3.4 User Story

User story ID	**As a <Type of Users>**	**I want to <Perform some task>**	**So that I can <Achieve some goal >**
1	User	Protect USB Port	Restrict or allow new USB gadgets
2	User	Manage permissions	Grant or revoke app permissions and data restrict
3	User	Pattern lock size	Change size of pattern, hide pattern dots, hide errors
4	User	Scramble pin	Scramble pin numbers in lockscreen
5	User	Floating messenger	Access messenging application with bubbles
6	User	Backup data	Encrypted data backup with seedvault
7	User	Lock apps	Protect apps with password or fingerprint
8	User	Change apps icons	3rd party app icon support
9	User	Theme phone	Change clock,accent color and fonts
10	User	Set screen timeout, sync, heads up toggle	Stay active while reading, sync data, change notification heads up

User story ID	**As a <Type of Users>**	**I want to <Perform some task>**	**So that I can <Achieve some goal >**
11	User	Take Screenshot	Take partial screenshot
12	User	Link ringtone and notification	Combine ringtone and notification volume control
13	User	Control volume	Volume panel with alarm, ringtone, message alert and location to show
14	User	Record calls	Save call recordings to storage
15	User	Disable sensors	Protect camera and mic from being misused
16	User	VPN	Wireguard VPN Kernel support
17	User	Panic trigger	Action for panic trigger
18	Developer	Provide updates	Update check, Network to download

Table 3.1: User Story

3.5 Product Backlog

USER STORY ID	PRIORITY (LOW,HIGH, MEDIUM)	SIZE	SPRINT	STATUS (PLANNED, PROGRESSED, COMPLETED)	RELEASE DATE	RELEASE GOAL
1	HIGH	8	1	Completed	3/4/2021	Grant and revoke USB gadgets from accessing device also while locked
2	HIGH	6		Completed	5/4/2021	Manage each app permissions and data usage
3	HIGH	5		Completed	8/4/2021	Change pattern lock size accordingly and hide dots
4	MEDIUM	9		Completed	10/4/2021	Scramble pin enter layout for security
5	HIGH	5		Completed	12/4/2021	Floating messenger support for messaging apps

USER STORY ID	PRIORITY (LOW,HIGH, MEDIUM)	SIZE	SPRINT	STATUS (PLANNED, PROGRESSED, COMPLETED)	RELEASE DATE	RELEASE GOAL
6	HIGH	7	2	Completed	14/4/2021	Backup data
7	HIGH	7		Completed	17/4/2021	Protect apps with password or pass-code
8	MEDIUM	7		Completed	19/4/2021	Third party launcher icons support
9	HIGH	6		Completed	23/4/2021	Theme device
10	HIGH	4	3	Completed	30/4/2021	Stay active while reading, sync data, Toggle heads up notifica-tions
11	HIGH	6		Completed	2/5/2021	Take partial screenshot
12	HIGH	5		Completed	5/5/2021	Combine volume control for notifi-cation & ringtone
13	MEDIUM	9		Completed	10/5/2021	Expanded vol-ume panel with all controls

USER STORY ID	PRIORITY (LOW,HIGH, MEDIUM)	SIZE	SPRINT	STATUS (PLANNED, PROGRESSED, COMPLETED)	RELEASE DATE	RELEASE GOAL
14	HIGH	5	4	Completed	13/5/2021	Save call recordings
15	HIGH	7		Completed	17/5/2021	Disable camera and mic
16	HIGH	6		Completed	20/5/2021	Wireguard VPN Support
17	HIGH	6		Completed	22/5/2021	Panic trigger
18	HIGH	6		Completed	24/5/2021	Updater application

Table 3.2: Product Backlog

3.6 Project Plan

USER STORY ID	TASK NAME	START DATE	END DATE	DAYS	STATUS (TO BE FILLED BY SCRUM MASTER)
1	SPRINT 1	01/04/2021	03/04/2021	3	Completed
2		04/04/2021	05/04/2021	2	Completed
3		06/04/2021	08/04/2021	3	Completed
4		09/04/2021	10/04/2021	2	Completed
5		11/04/2021	12/04/2021	2	Completed
6	SPRINT 2	13/04/2021	14/04/2021	2	Completed
7		15/04/2021	17/04/2021	3	Completed
8		18/04/2021	19/04/2021	2	Completed
9		20/04/2021	23/04/2021	4	Completed
10	SPRINT 3	24/04/2021	30/04/2021	7	Completed
11		01/04/2021	02/05/2021	2	Completed
12		03/05/2021	05/05/2021	3	Completed
13		06/05/2021	10/05/2021	5	Completed
14	SPRINT 4	11/05/2021	13/05/2021	3	Completed
15		14/05/2021	17/05/2021	4	Completed
16		18/05/2021	20/05/2021	3	Completed
17		21/05/2021	22/05/2021	2	Completed
18		23/05/2021	24/05/2021	2	Completed

Table 3.3: Project Plan

3.7 Sprint Backlog Planned

3.7.1 Sprint 1

Backlog items	Completion date	Original Estimated hours	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7	Day 8	Day 9	Day 10	Day 11	Day 12
			01/4/2021	02/4/2021	03/4/2021	04/4/2021	05/4/2021	06/4/2021	07/4/2021	08/4/2021	09/4/2021	10/4/2021	11/4/2021	12/4/2021
User Story 1			Hours	Hours	Hours	Hours	Hours	Hours	Hours	Hours	Hours	Hours	Hours	Hours
UI Design	01/4/2021	3	3											
Coding	02/4/2021	5		5										
Testing	03/4/2021	1			1									
User Story 2														
UI Design	04/4/2021	7				3	4							
Coding	05/4/2021	7				5	2							
Testing	05/4/2021	2					2							
User Story 3														
UI Design	06/4/2021	4						4						
Coding	07/4/2021	6						2	4					
Testing	08/4/2021	2								2				
User Story 4														
UI Design	09/4/2021	6									4	2		
Coding	10/4/2021	7									2	5		
Testing	10/4/2021	2									1	1		
User Story 5														
UI Design	11/4/2021	7											4	3
Coding	11/4/2021	5											1	4
Testing	12/4/2021	2												2

Table 3.4: Sprint 1 (Planned)

3.7.2 Sprint 2

Backlog items	Completion date	Original Estimated hours	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7	Day 8	Day 9	Day 10	Day 11
			13/4/2021	14/4/2021	15/4/2021	16/4/2021	17/4/2021	18/4/2021	19/4/2021	20/4/2021	21/4/2021	22/4/2021	23/4/2021
User Story 6			Hours	Hours	Hours	Hours	Hours	Hours	Hours	Hours	Hours	Hours	Hours
UI Design	14/4/2021	4	2	2									
Coding	14/4/2021	5	3	2									
Testing	14/4/2021	2		2									
User Story 7													
UI Design	15/4/2021	3			3								
Coding	16/4/2021	5				5							
Testing	17/4/2021	1					1						
User Story 8													
UI Design	19/4/2021	7						3	4				
Coding	19/4/2021	7						5	2				
Testing	19/4/2021	2							2				
User Story 9													
UI Design	22/4/2021	4										4	
Coding	22/4/2021	6								2	2	2	
Testing	23/4/2021	2									1		1

Table 3.5: Sprint 2 (Planned)

3.7.3 Sprint 3

Backlog items	Completion date	Original Estimated hours	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7	Day 8	Day 9	Day 10	Day 11	Day 12
			24/4/2021	25/4/2021	26/4/2021	27/4/2021	28/4/2021	29/4/2021	30/4/2021	01/05/2021	02/05/2021	03/5/2021	04/05/2021	05/05/2021
User Story 10			Hours	Hours	Hours	Hours	Hours	Hours	Hours	Hours	Hours	Hours	Hours	Hours
UI Design	30/4/2021	5	1	1				2	1					
Coding	27/4/2021	7		2	3	2								
Testing	30/4/2021	2		1					1					
User Story 11														
UI Design	1/5/2021	3								3				
Coding	2/5/2021	5								3	2			
Testing	2/5/2021	1									1			
User Story 12														
UI Design	4/5/2021	7										3	3	
Coding	4/5/2021	7										5	2	
Testing	5/5/2021	2												2

Backlog items	Completion date	Original Estimated hours	Day 13	Day 14	Day 15	Day 16	Day 17
			06/05/2021	07/05/2021	08/05/2021	09/05/2021	10/05/2021
User Story 13			Hours	Hours	Hours	Hours	Hours
UI Design	09/5/2021	4			2	2	
Coding	08/5/2021	6	2	2	2		
Testing	10/5/2021	2		1			1

Table 3.6: Sprint 3 (Planned)

3.7.4 Sprint 4

Backlog items	Completion date	Original Estimated hours	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7	Day 8	Day 9	Day 10	Day 11	Day 12	Day 13	Day 14
			11/5/2021	12/5/2021	13/5/2021	14/5/2021	15/5/2021	16/5/2021	17/5/2021	18/05/2021	19/05/2021	20/5/2021	21/05/2021	22/05/2021	23/05/2021	24/05/2021
User Story 14			Hours	Hours	Hours	Hours	Hours	Hours	Hours	Hours	Hours	Hours	Hours	Hours	Hours	Hours
UI Design	12/5/2021	5	2	3												
Coding	13/5/2021	7	2	2	3											
Testing	13/5/2021	2		1	1											
User Story 15																
UI Design	14/5/2021	5				3	2									
Coding	15/5/2021	8				2	3	3								
Testing	17/5/2021	2						1	1							
User Story 16																
UI Design	19/5/2021	5								2	3					
Coding	20/5/2021	2									1	1				
Testing	20/5/2021	2										2				
User Story 17																
UI Design	22/5/2021	5											3	2		
UI Design	22/5/2021	4											2	2		
UI Design	22/5/2021	2												2		
User Story 18																
UI Design	24/5/2021	5													3	2
UI Design	24/5/2021	6													3	1
UI Design	24/5/2021	3													2	1

Table 3.7: Sprint 4 (Planned)

3.8 Sprint Backlog Actual

3.8.1 Sprint 1

Backlog items	Completion date	Original Estimated hours	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7	Day 8	Day 9	Day 10	Day 11	Day 12
			01/4/2021	02/4/2021	03/4/2021	04/4/2021	05/4/2021	06/4/2021	07/4/2021	08/4/2021	09/4/2021	10/4/2021	11/4/2021	12/4/2021
User Story 1			Hours	Hours	Hours	Hours	Hours	Hours	Hours	Hours	Hours	Hours	Hours	Hours
UI Design	01/4/2021	3	4	2										
Coding	03/4/2021	5		2	2									
Testing	03/4/2021	1		1	2									
User Story 2														
UI Design	05/4/2021	7				2	3							
Coding	05/4/2021	7				3	2							
Testing	05/4/2021	2				1	2							
User Story 3														
UI Design	07/4/2021	4						4	1					
Coding	07/4/2021	6						2	3					
Testing	08/4/2021	2							1	2				
User Story 4														
UI Design	09/4/2021	6									4	2		
Coding	10/4/2021	7									3	5		
Testing	10/4/2021	2										2		
User Story 5														
UI Design	11/4/2021	7											3	3
Coding	11/4/2021	5											2	4
Testing	12/4/2021	2											1	2

Table 3.8: Sprint 1 (Actual)

3.8.2 Sprint 2

Backlog items	Completion date	Original Estimated hours	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7	Day 8	Day 9	Day 10	Day 11
			13/4/2021	14/4/2021	15/4/2021	16/4/2021	17/4/2021	18/4/2021	19/4/2021	20/4/2021	21/4/2021	22/4/2021	23/4/2021
User Story 6			Hours	Hours	Hours	Hours	Hours	Hours	Hours	Hours	Hours	Hours	Hours
UI Design	14/4/2021	4	2	3									
Coding	14/4/2021	5	3	3									
Testing	14/4/2021	2	1	2									
User Story 7													
UI Design	16/4/2021	3			3	1							
Coding	17/4/2021	5				3	2						
Testing	17/4/2021	1				1	1						
User Story 8													
UI Design	19/4/2021	7						4	3				
Coding	19/4/2021	7						3	4				
Testing	19/4/2021	2						2	1				
User Story 9													
UI Design	23/4/2021	4								2	1	1	1
Coding	23/4/2021	6									2	2	2
Testing	23/4/2021	2										1	1

Table 3.9: Sprint 2 (Actual)

3.8.3 Sprint 3

Backlog items	Completion date	Original Estimated hours	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7	Day 8	Day 9	Day 10		
			24/4/2021	25/4/2021	26/4/2021	27/4/2021	28/4/2021	29/4/2021	30/4/2021	01/05/2021	02/05/2021	03/5/2021		
User Story 10			Hours	Hours	Hours	Hours	Hours	Hours	Hours	Hours	Hours	Hours		
UI Design	28/4/2021	5		2	2		2							
Coding	27/4/2021	7	2	2	1	2								
Testing	28/4/2021	2			2	1	1							
User Story 11														
UI Design	30/4/2021	3						2	2					
Coding	30/4/2021	5						3	3					
Testing	30/4/2021	1						1	1					
User Story 12														
UI Design	3/5/2021	7								2	2	1		
Coding	3/5/2021	7								1	2	3		
Testing	3/5/2021	2									2	1		

Backlog items	Completion date	Original Estimated hours	Day 11	Day 12	Day 13	Day 14								
			04/05/2021	05/05/2021	06/05/2021	07/05/2021								
User Story 13			Hours	Hours	Hours	Hours								
UI Design	6/5/2021	4	2	2	1									
Coding	7/5/2021	6		3	1	3								
Testing	7/5/2021	2			2	1								

Table 3.10: Sprint 3 (Actual)

3.8.4 Sprint 4

Backlog items	Completion date	Original Estimated hours	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7	Day 8	Day 9	Day 10	Day 11	Day 12	Day 13	Day 14	Day 15	Day 16	Day 17
			08/5/2021	09/5/2021	10/5/2021	11/5/2021	12/5/2021	13/5/2021	14/5/2021	15/05/2021	16/05/2021	17/5/2021	18/05/2021	19/05/2021	20/05/2021	21/05/2021	22/05/2021	23/05/2021	24/05/2021
User Story 14			Hours	Hours	Hours	Hours	Hours	Hours	Hours	Hours	Hours	Hours	Hours	Hours	Hours	Hours	Hours	Hours	Hours
UI Design	10/5/2021	5	2	1	2														
Coding	11/5/2021	7		3	2	2													
Testing	11/5/2021	2			1	1													
User Story 15																			
UI Design	13/5/2021	5					2	3											
Coding	14/5/2021	8					2	2	2										
Testing	14/5/2021	2						1	1										
User Story 16																			
UI Design	16/5/2021	5								2	3								
Coding	16/5/2021	2									2								
Testing	16/5/2021	2								1	1								
User Story 17																			
UI Design	20/5/2021	5										3	3	1					
Coding	20/5/2021	4										2	2	3					
Testing	20/5/2021	2												1	2				
User Story 18																			
UI Design	23/05/2021	5														2	1	2	
Coding	24/05/2021	6															1	1	1
Testing	24/05/2021	3														1		1	2

Table 3.11: Sprint 4 (Actual)

3.9 Sprint Review

3.9.1 Sprint 1

User story ID	**Comments from scrum master, if any**	**Comments from product owner, if any**
1	Satisfied	Test properly
2	Satisfied	Satisfied
3	Satisfied	Improve UI
4	Satisfied	Satisfied
5	Satisfied	Satisfied

Table 3.12: Sprint 1 (Review)

3.9.2 Sprint 2

User story ID	**Comments from scrum master, if any**	**Comments from product owner, if any**
6	Satisfied	Satisfied
7	Satisfied	Satisfied
8	Satisfied	Satisfied
9	Satisfied	Satisfied

Table 3.13: Sprint 2 (Review)

3.9.3 Sprint 3

User story ID	**Comments from scrum master, if any**	**Comments from product owner, if any**
10	Satisfied	Satisfied
11	Satisfied	Satisfied
12	Satisfied	Satisfied
13	Satisfied	Satisfied

Table 3.14: Sprint 3 (Review)

3.9.4 Sprint 4

User story ID	**Comments from scrum master, if any**	**Comments from product owner, if any**
14	Satisfied	Satisfied
15	Satisfied	Satisfied
16	Satisfied	Satisfied
17	Satisfied	Satisfied
18	Satisfied	Satisfied

Table 3.15: Sprint 4 (Review)

3.10 User Interface Design

3.10.1 Protect USB Port

This options allows users to select different options to enable or disable peripheral connections while device is locked or unlocked. There are three options to deny all USB, allow new devices while unlocked or allow even while locked.

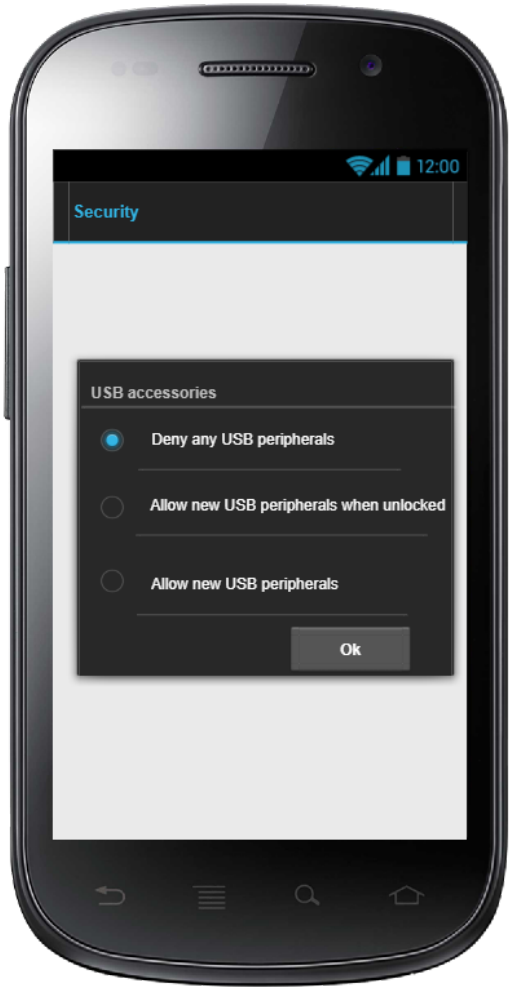


Figure 3.5: Protect USB Port

3.10.2 Permissions Manager

This options allows us to view all the permissions and the apps that are using these permissions. We can also allow or deny permissions by clicking them.

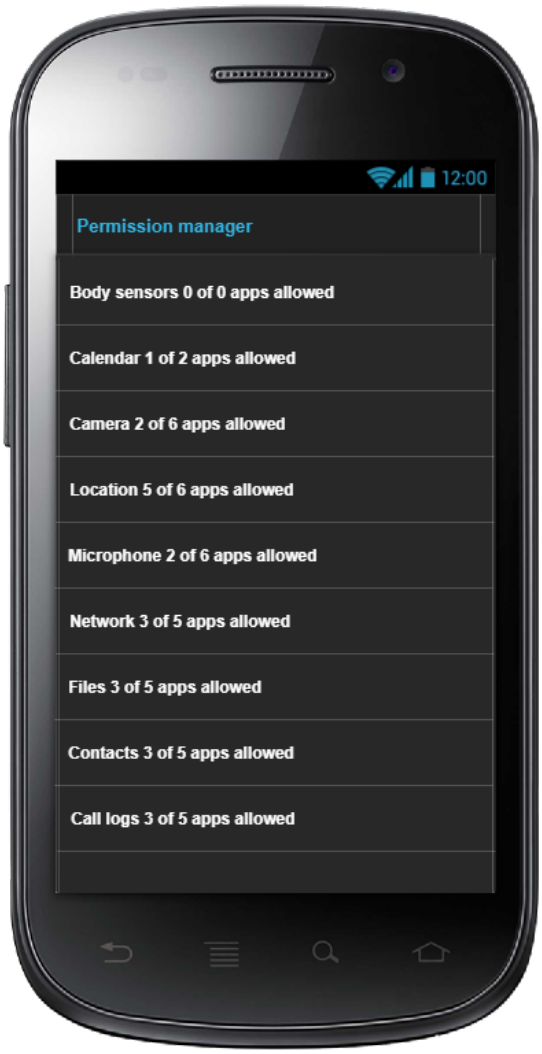


Figure 3.6: Permissions Manager 1

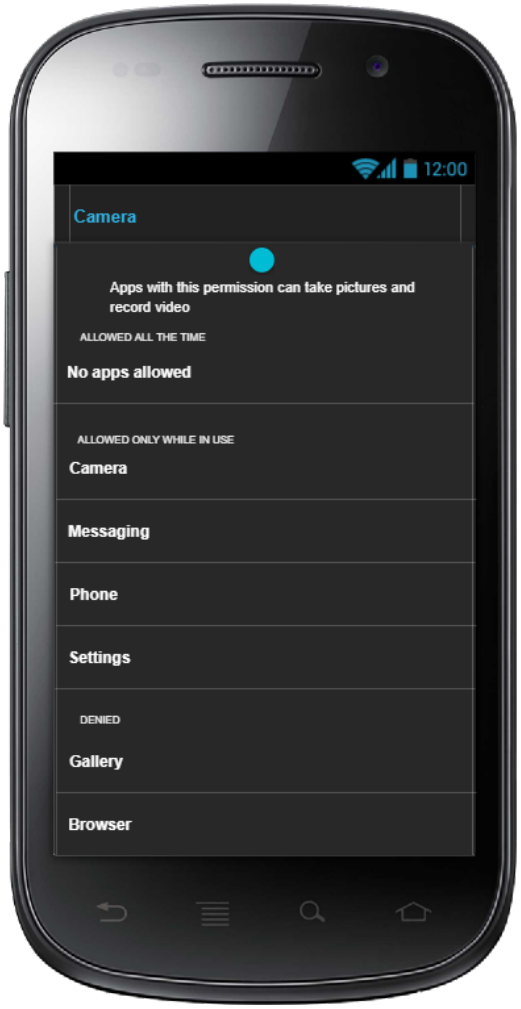


Figure 3.7: Permissions Manager 2

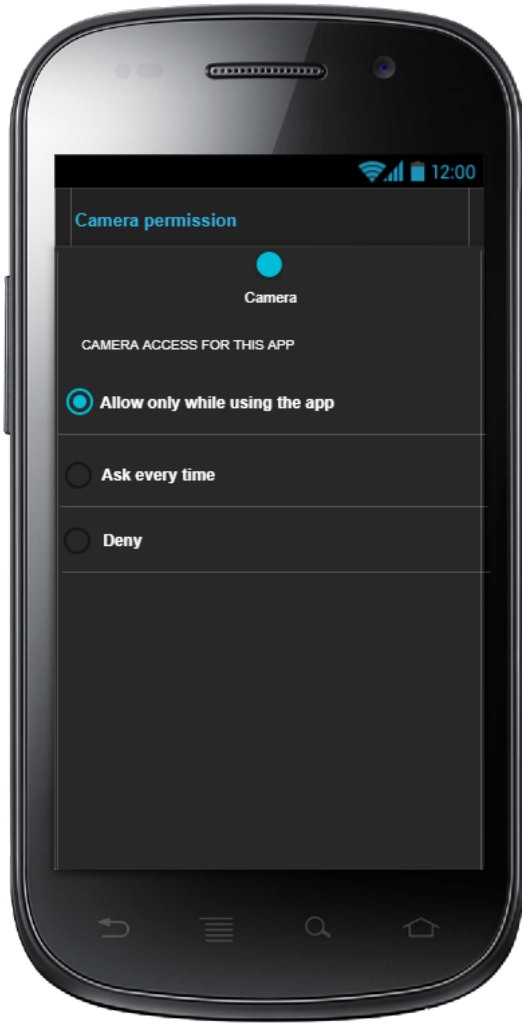


Figure 3.8: Permissions Manager 3

3.10.3 Pattern Lock size

This options allows us to select desired pattern lock layout size. The bigger the layout theres more ways to lock user device.

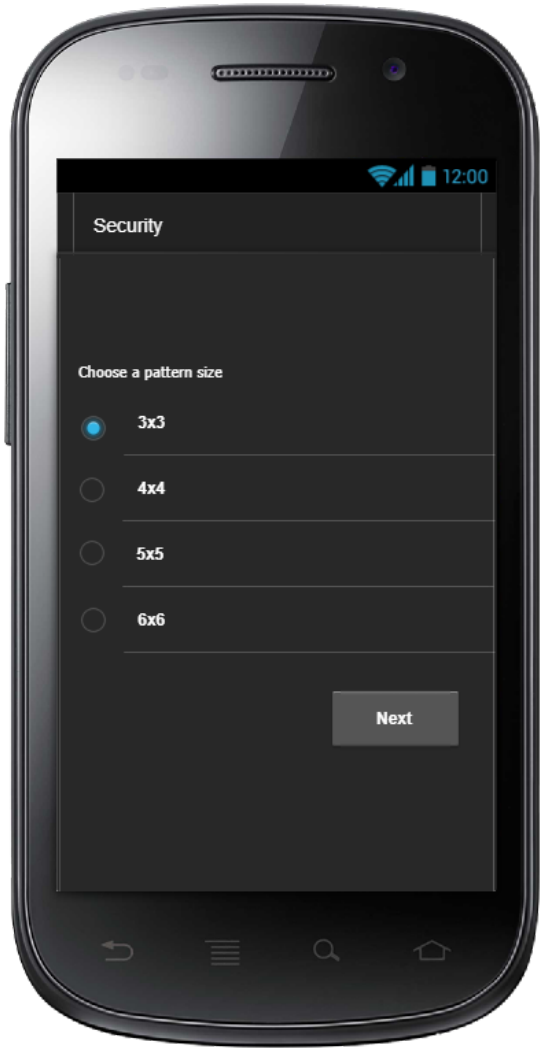


Figure 3.9: Pattern lock size 1

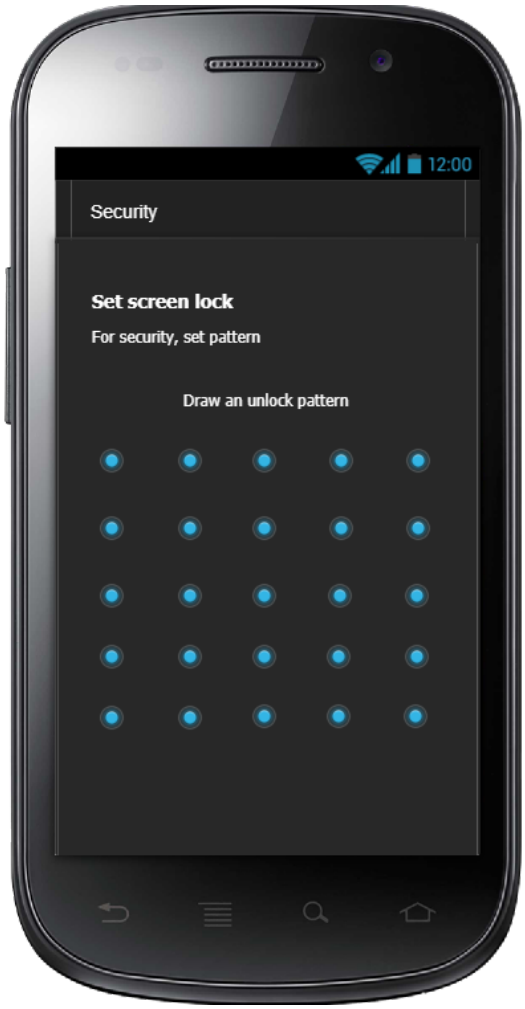


Figure 3.10: Pattern lock size 2

3.10.4 Scramble pin

This options allows us to scramble the pin layout in lock screen. Which allows us to enter pin each time we unlock phone. Less vulnerable with group of people considering normal layout.

Figure 3.11: Scramble pin

3.10.5 Bubbles

This feature allows us to use Android-11's chat heads. Its similar to Facebook messenger chat heads, but supports more apps and can be used for contacts specific.

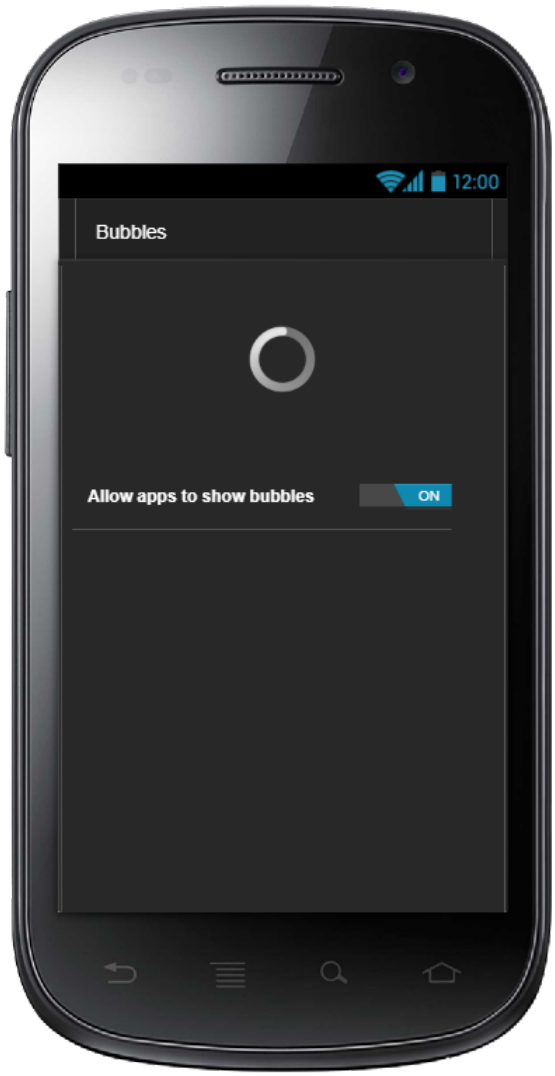


Figure 3.12: Bubbles

3.10.6 Data Backup

This feature allow us to backup our data easily with Seed-vault. Its a user-friendly encryption using a mnemonic phrase. We can backup the data into cloud, USB flash drive and restore later with the phrases.

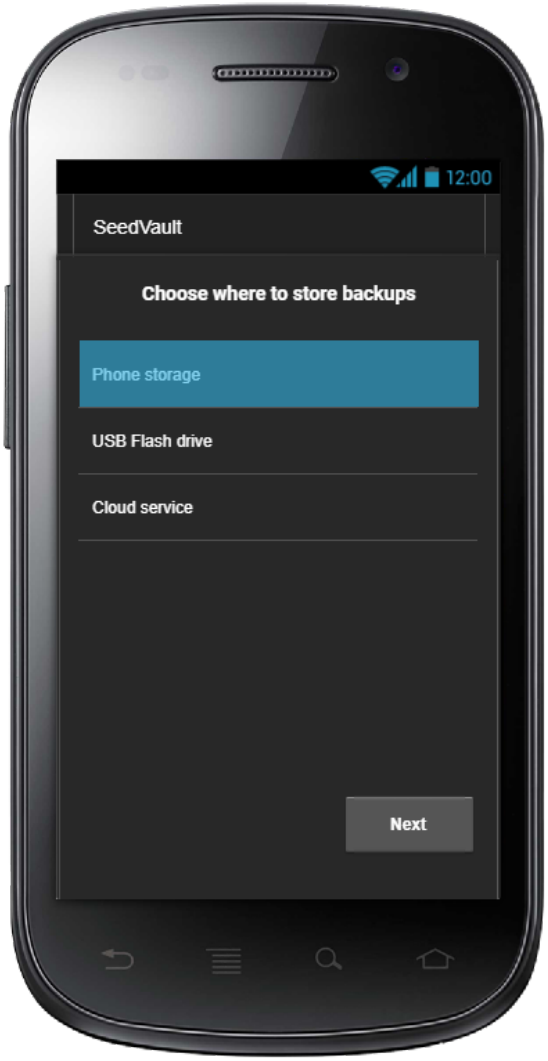


Figure 3.13: Data Backup

3.10.7 Hidden & Protected apps

This feature to lock and hide apps in launcher. It uses screen lock password as protection, so no need to set separate password for this option.

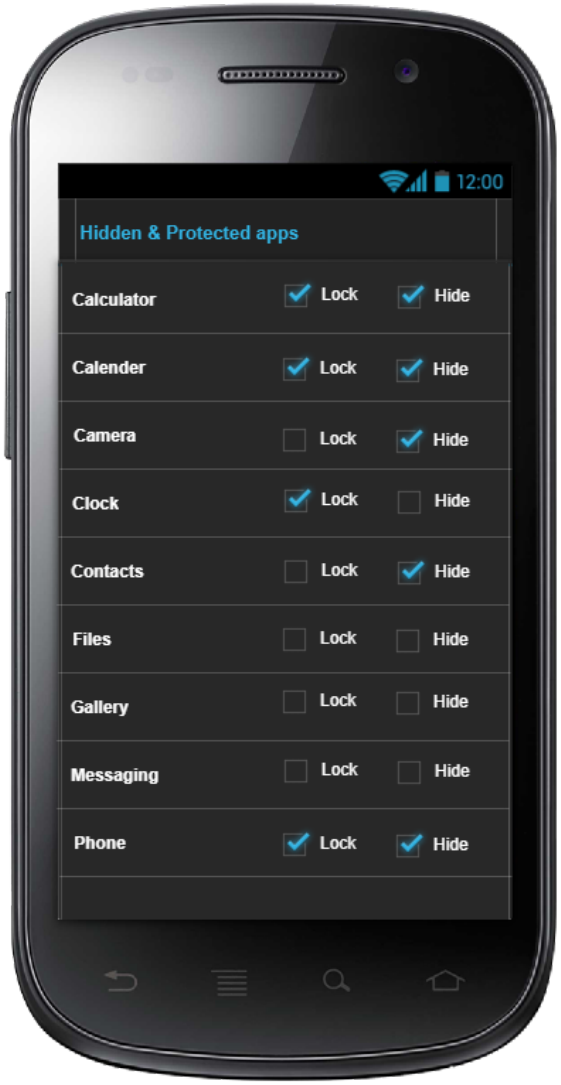


Figure 3.14: Hidden & Protected apps

3.10.8 Icon packs

This feature allow users to install third party icons packs and customize their launcher icons with ease.

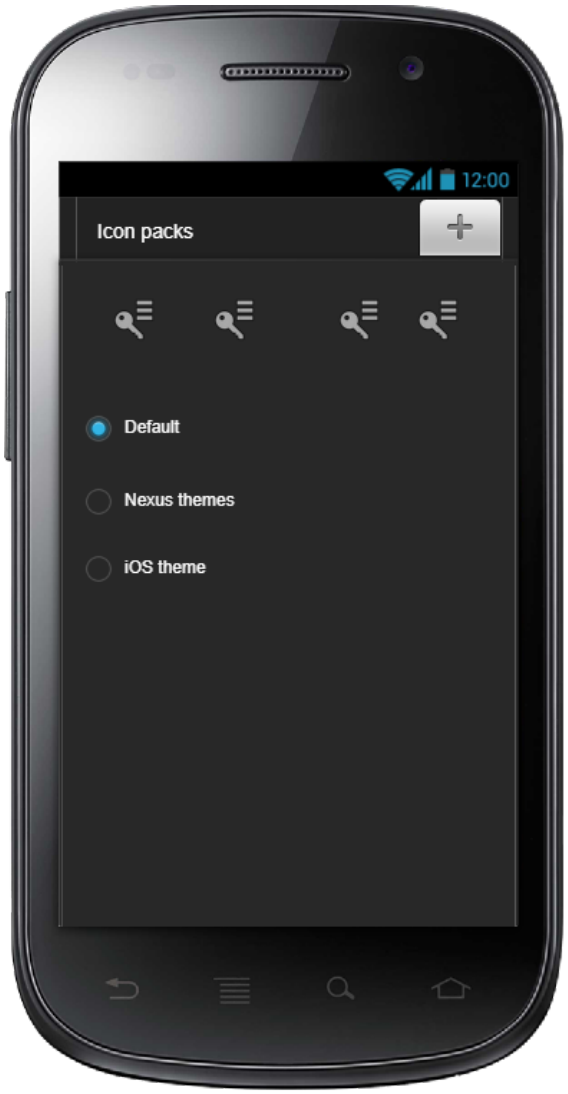


Figure 3.15: Icon Packs

3.10.9 Theme phone

This apps is provided by Google, But disabled by default for AOSP device. This app can change device's fonts, accent color, icon shapes, launcher icon shapes,etc. Also change clock faces, styles and wallpapers.

Figure 3.16: Theme phone 1

Figure 3.17: Theme phone 2

Figure 3.18: Theme phone 3

Figure 3.19: Theme phone 4

3.10.10 QS Tiles

Added three types of QS tiles which doesn't exist in AOSP. First one caffeine allows to temporary set screen out time. And sync allows to sync data from internet connected apps to work properly. And heads up allows to disable notification from status bar to popup.

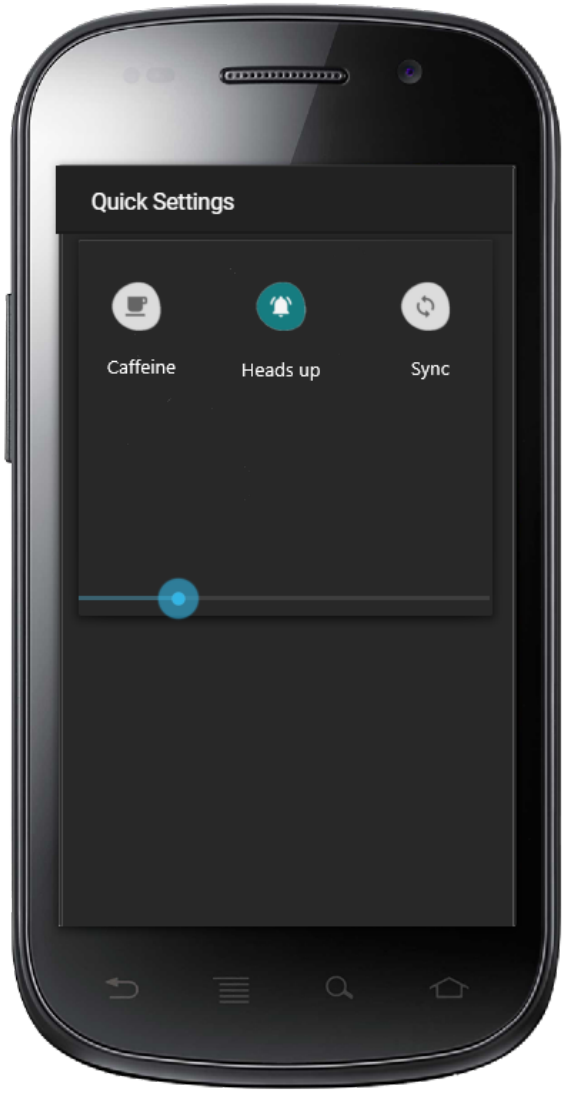


Figure 3.20: QS Tiles

3.10.11 Partial Screenshot

This feature will allow us to take partial screenshot of the screen by short click and long click for full screen shot.

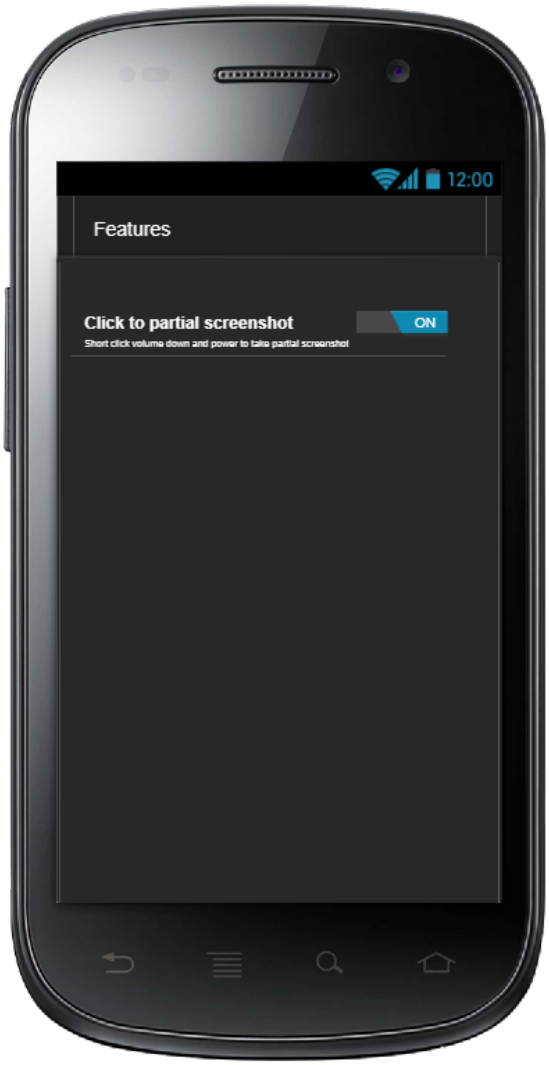


Figure 3.21: Partial Screenshot

3.10.12 Link Ringer & Notification volume

This feature will allow us link ringer and notification volume. So we can control both with a single slider. No need to control separately. We can toggle it on or off as per user needs.

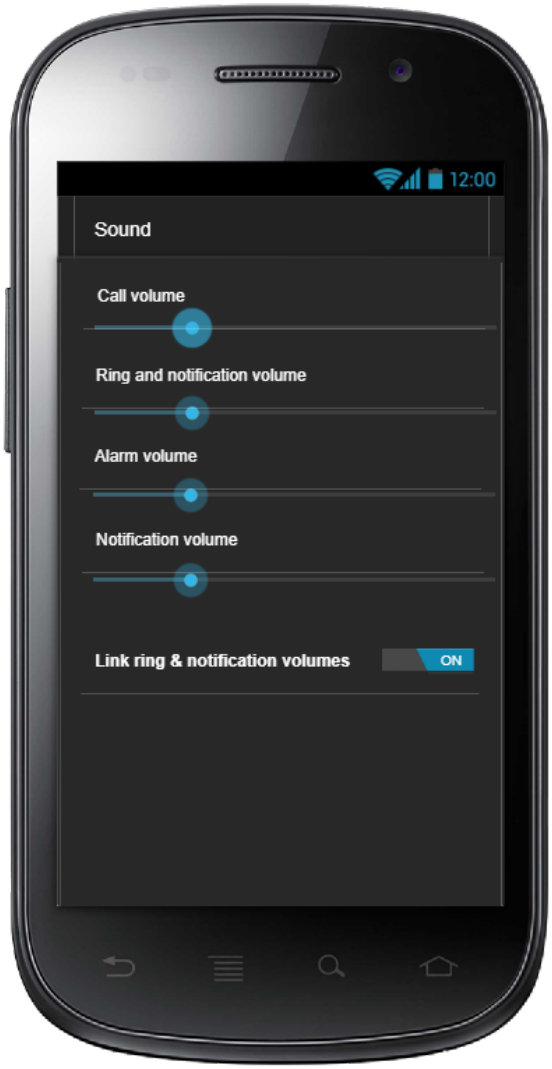


Figure 3.22: Link Ringer & Notification volume

3.10.13 Volume Panel

This feature allows users to control the volume of alarm, media and ringer volume with single panel. This was used before Android-9.0 and discontinued in post Android versions.

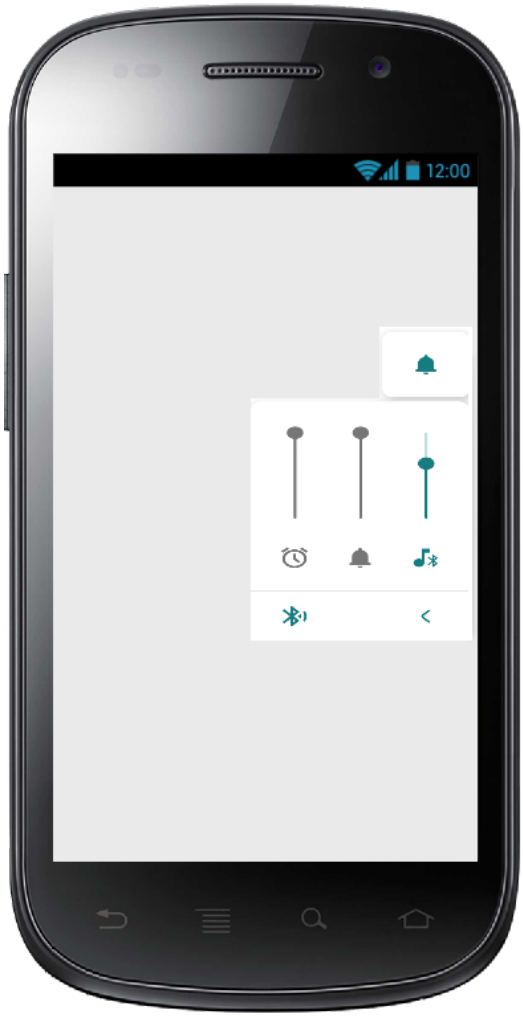


Figure 3.23: Volume Panel 1

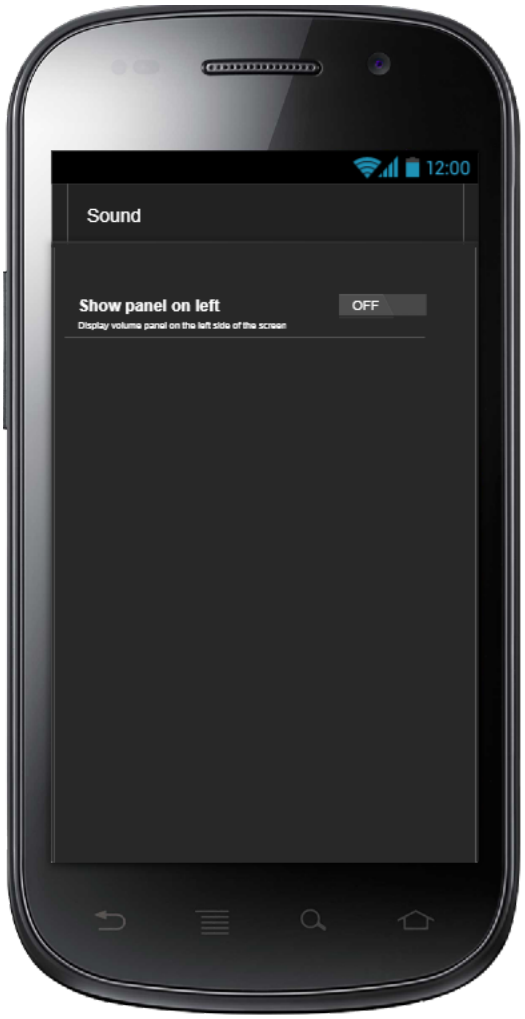


Figure 3.24: Volume Panel 2

3.10.14 Record calls

This feature allows to record all native voice calls and save them to the phone storage. Including the quality of the calls and record type can be adjusted.

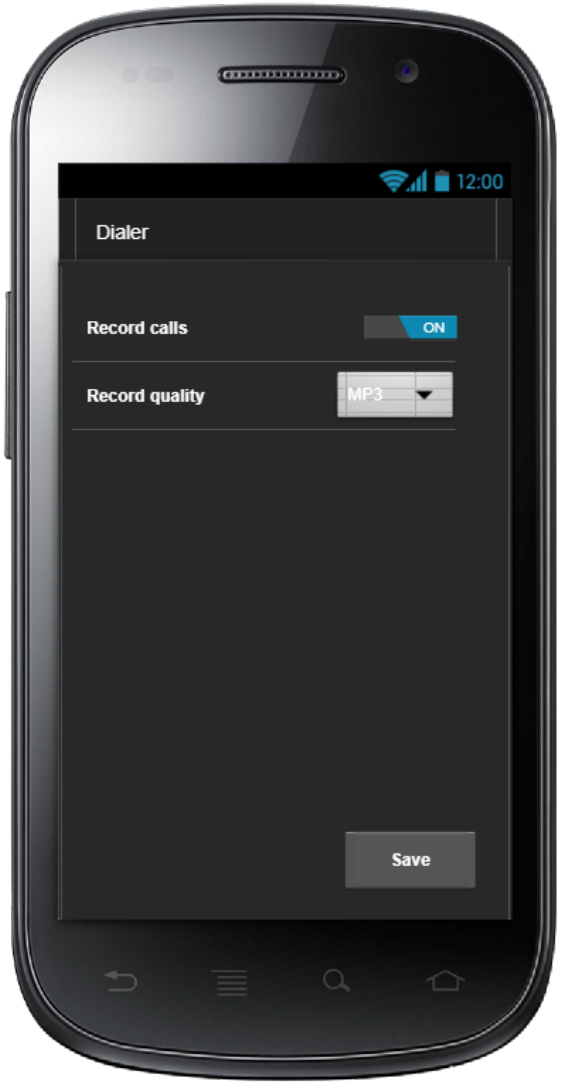


Figure 3.25: Record calls

3.10.15 Disable sensors

This tile will allow users to disable sensors on the phone. Mostly it disables cameras and microphones. This will help users to be not monitored by any means or application in background.

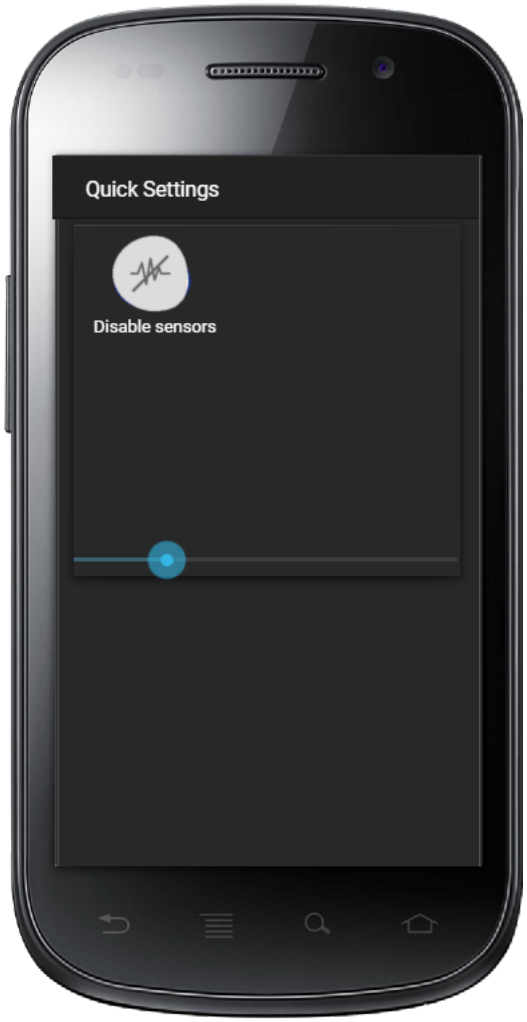


Figure 3.26: Disable sensors

3.10.16 Panic trigger

”Panic button” that can send it’s trigger message to any app that is a ”Panic responder”. Such apps can do things like lock, disguise themselves, delete private data, send an emergency message, and more.

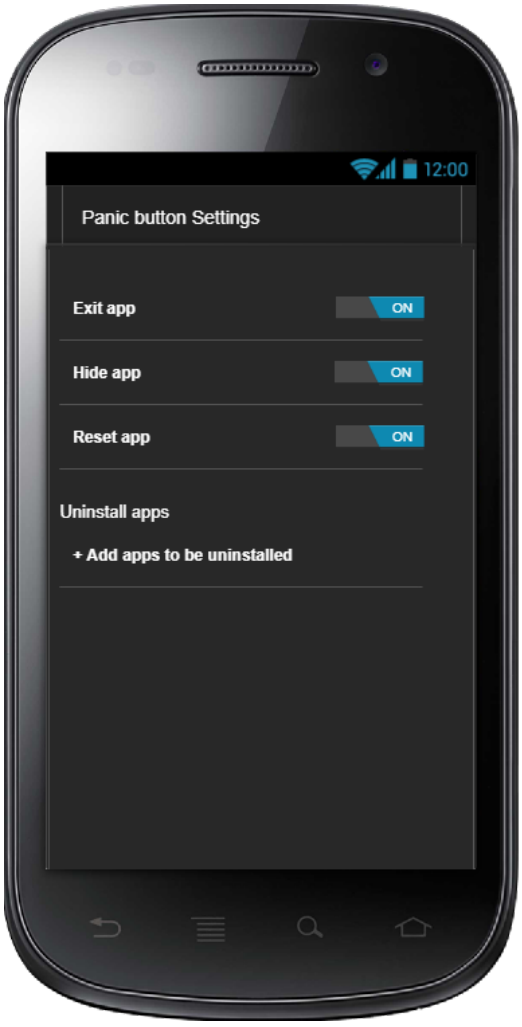


Figure 3.27: Disable sensors

3.10.17 Provide updates

This app will allow users to receive updates from the developer. This can improve the device security and future bug fixes will be given through this by the developer.

Figure 3.28: Provide updates 1

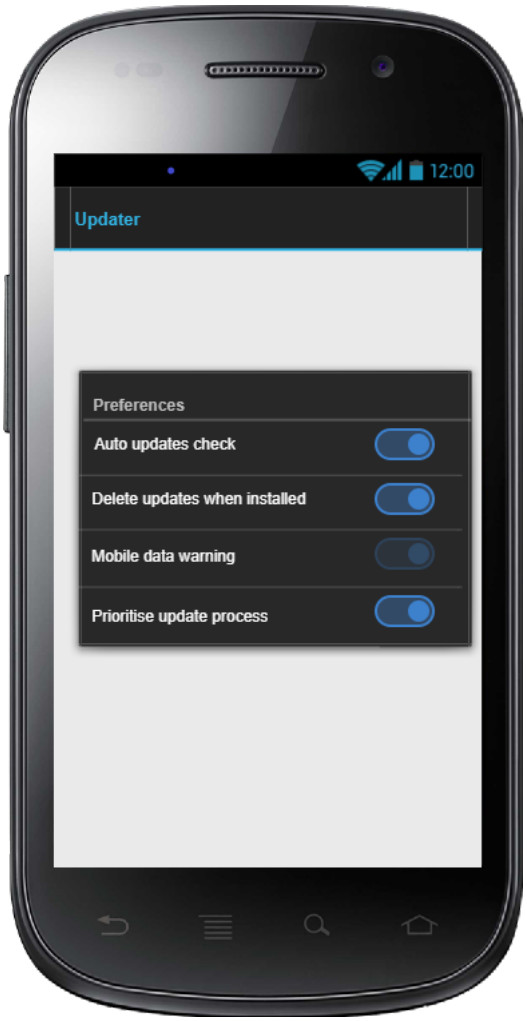


Figure 3.29: Provide updates 2

3.11 Testing and Implementation

3.11.1 Testing

Test Case - 1

No.	Date	Action	Expected Result	Actual Result	Pass?
1	03-04-2021	Protect USB Port	Should be able to grant or revoke USB from accessing device also while locked	Is able to revoke USB from accessing device also while locked	Yes
2	05-04-2021	Manage permissions	Should be able to manage each app permissions	Able to manage app permissions and data usage	Yes
3	08-04-2021	Pattern lock size	Should be able to change size of pattern, hide pattern, hide errors	Able to change size of pattern, hide pattern, hide errors	Yes
4	10-04-2021	Scramble pin	Should be able to scramble pin numbers in lock screen	Able to change scramble pin numbers in lock screen	Yes
5	12-04-2021	Floating messenger	Should be able to access messaging application with bubble	Able to change access messaging application with bubble	Yes

Table 3.16: Test Case - 1

Test Case - 2

No.	Date	Action	Expected Result	Actual Result	Pass?
6	14-04-2021	Backup data	Should be able to encrypted data backup with seed vault	Able to encrypted data backup with seed vault	Yes
7	17-04-2021	Lock apps	Should be able to protect apps with password or fingerprint	Able to protect apps with password or fingerprint	Yes
8	19-04-2021	Change apps icons	Should be able to change 3rd party app icon support	Able to change 3rd party app icon support	Yes
9	23-04-2021	Theme phone	Should be able to change clock,accent color and fonts	Able to change clock,accent color and fonts	Yes

Table 3.17: Test Case - 2

Test Case - 3

No.	Date	Action	Expected Result	Actual Result	Pass?
10	28-04-2021	Easy access to sync, screen timeout, notifications	Should be able to toggle for stay active screen, heads up notification, sync	Able to change notification heads up, screen time-out and sync	Yes
11	30-04-2021	Take partial screenshot	Should be able to take partial screenshot	Able to take partial screenshot and full screen	Yes
12	03-05-2021	Link ringtone and notification	Should be able to link ringtone and notification	Link notification and ringtone volume and control at once	Yes
13	07-05-2021	Control volume	Should be able to change volume with volume panel of alarm, ringtone, message alert and location to show	Able to control volume with single volume dialog	Yes

Table 3.18: Test Case - 3

Test Case - 4

No.	Date	Action	Expected Result	Actual Result	Pass?
14	11-05-2021	Record calls	Should be able save call recordings to storage	Able to record calls and change format	Yes
15	14-05-2021	Disable sensors	Should be able to protect camera and mic from being misused	Able to disable all sensors like camera, mic and accelerometer	Yes
16	16-05-2021	VPN	Should be able to use Wire Guard VPN Kernel support	Able to connect to VPN	Yes
17	20-05-2021	Panic trigger	Should be able to trigger actions on panic button is pressed	Able to trigger actions on panic button is pressed	Yes
18	22-05-2021	Provide updates	Should be able to update check, network to download	Able to see new updates	Yes

Table 3.19: Test Case - 4

3.11.2 Implementation

After testing, the proposed system is ready for the implementation. Implementation is the stage of the project when the theoretical design is turned in to a working system. Implementation is the process of bringing a newly developed system or revised into operational one. The new system and its components are to be tested in a structured and planned manner. The implementation stage of a project is often very complex and time consuming and many more people are involved in the earlier stages. This involves careful planning, investigation of the current system and constraints of implementation, installing hardware, training the operating users in the changeover procedures before the system is setup and running. So, proposed system is easy to implement. It would be very easy to run also.

While implementing this system we only have few challenges. First challenge is to unlock bootloader of the Android phone. For that we need to go turn on OEM Unlocking in developer settings. Since this is possible in any carrier unlocked smartphone, it will be easy to do. The user must have a device with good internet access to download the custom ROM components. Rest of the challenges are depended on the user, knowledge about using Android device and installing with the guide is as follows, for downloading ROM (OTA zip, factory image zip and boot image) visit :
https://pegasusos.github.io (Instructions available in the site)

Installation: For normal builds

1. - Make sure platform tools is installed (ADB and fastboot).
2. - Download ROM ota zip & boot image file from above link.
3. - Boot your device into bootloader.
4. - On your PC, use command - ’ fastboot flash boot boot.img ’

5. - Boot into recovery

6. - Click Apply Update - Apply from ADB (option available in recovery)

7. - Open cmd or terminal in your pc and enter this command - ’ adb sideload rom.zip ’

8. - Reboot

Installation: For factory image builds (You can lock bootloader)

1. - Make sure platform tools is installed (ADB and fastboot).

2. - Download factory image zip from above link.

3. - Extract the zip with archive tools.

4. - Connect pixel in boot-loader to PC (volume up + down combo)

5. - Run the flash-all* file to start flashing (.sh for linux and .bat for windows)

6. - Reboot

7. - Optional : Enter this to lock bootloader (fastboot oem lock)

Important notes:

- Required firmware version must be based on latest Android Q-based builds.
- Formatting data (all user data is wiped, including internal storage) is a must if stock ROM was previously installed and device was encrypted.

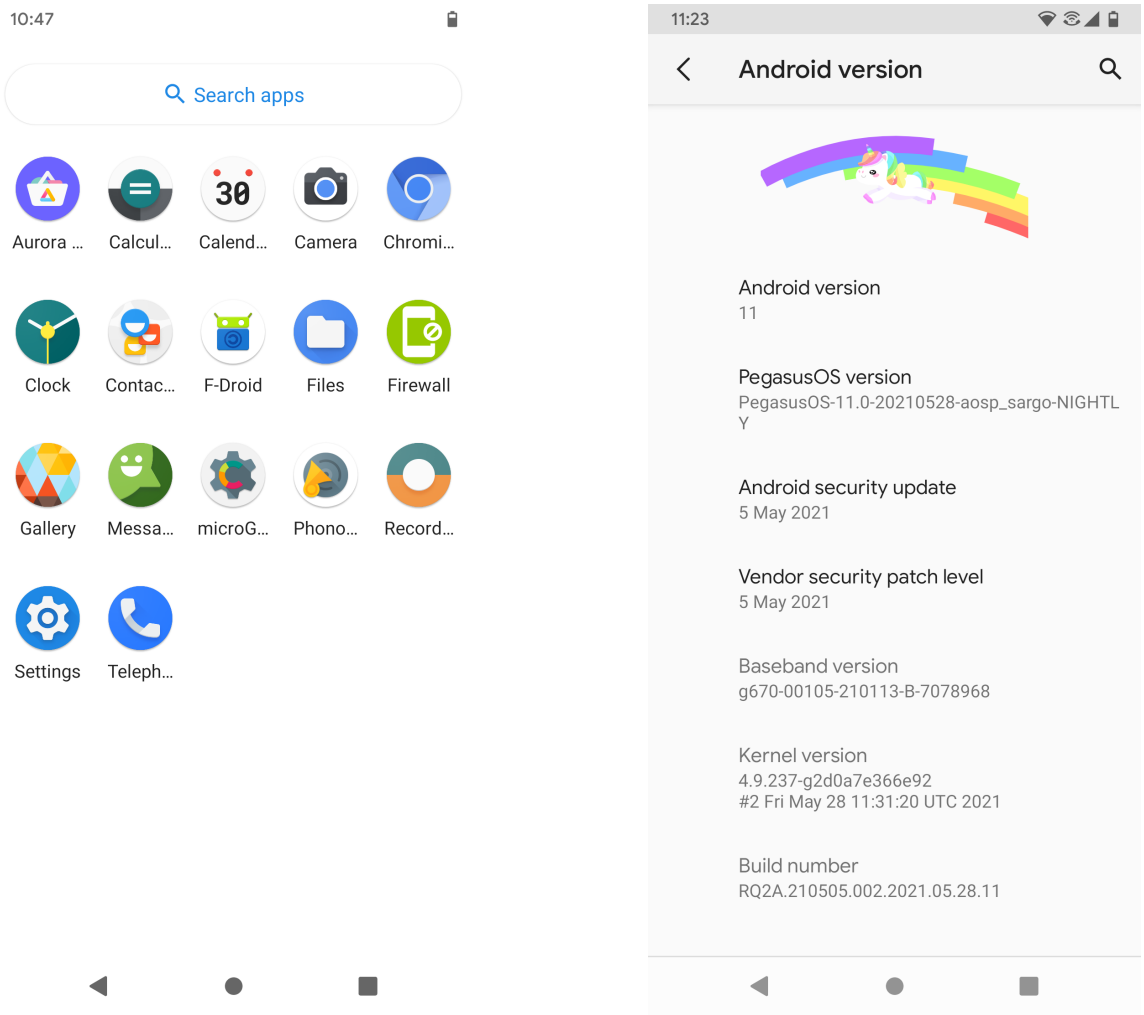
Chapter 4

RESULTS AND DISCUSSION

The project PegasusOS was developed with proper planning and guidance. Agile methodology is used during the development of this project. Planning at each stage was done properly. Each sprint has been conducted as per protocol. Testing was performed at each stage of development. The project is meant to provide a clean Android experience for those who are focused on privacy. PegasusOS is a AOSP based custom ROM for smartphones, which can be ported to other Android phones.

In this project I provide light customization for smoothing user experience and ease use. This ROMs doesn't contain google's proprietary codes or apps such as, google play service and play store, but I'm providing an open source alternatives for these apps like f-droid store for full open source apps and aurora store to download apps available in google play store without adding any google apps or APIs. Basic apps like dialer, messaging, browser, music player is included in this build. This custom os is more focused on privacy and security, so a password/pass code is must to use this operating system, it will asked on initial setup of the OS. Users can backup apps and data into cloud or USB drive using Seed-vault backup, which is using a encrypted backup solution.

4.1 Initial View



(a) App drawer

(b) About section

Figure 4.1: Initial View

Chapter 5

CONCLUSION

PegasusOS improves the privacy and security of the OS from the bottom up. It deploys technologies to mitigate whole classes of vulnerabilities and make exploiting the most common sources of vulnerabilities substantially more difficult.It ships with open source applications which source code is freely available on GitHub of PegasusOS or appropriate app developer's repo. It improves the security of both the OS and the apps running on it. The app sandbox and other security boundaries are fortified. PegasusOS tries to avoid impacting the user experience with the privacy and security features. Ideally, the features can be designed so that they're always enabled with no impact on the user experience and no additional complexity like configuration options. It's not always feasible, and PegasusOS does add various toggles for features like the Network permission, Sensors permission, Pattern lock size, restrictions when the device is locked (USB peripherals, camera, quick tiles), etc. Also theme features like accent colors, lock screen clock, along with more complex user-facing privacy and security features with their own UX.

Chapter 6

REFERENCES

1. - https://developer.android.com
2. - https://source.android.com
3. - https://source.android.com/setup/build/building
4. - https://gerrit.googlesource.com
5. - https://github.com/seedvault
6. - https://en.wikipedia.org/wiki/Rooting_(Android)
7. - https://en.wikipedia.org/wiki/Custom_firmware
8. - https://developers.google.com/android/images
9. - https://grapheneos.org/build
10. - https://github.com/lineageos
11. - https://github.com/grapheneos

Chapter 7

APPENDIX

7.1 Source code

7.1.1 Sensors Off Tile

```
package com.android.systemui.qs.tiles;

import android.content.Intent;
import android.hardware.SensorPrivacyManager;
import android.service.quicksettings.Tile;
import android.widget.Switch;

import com.android.internal.logging.MetricsLogger;
import com.android.internal.logging.nano.MetricsProto.MetricsEvent;
import com.android.systemui.R;
import com.android.systemui.plugins.ActivityStarter;
import com.android.systemui.plugins.qs.QSTile.BooleanState;
import com.android.systemui.qs.QSHost;
import com.android.systemui.qs.tileimpl.QSTileImpl;

import javax.inject.Inject;

/** Quick settings tile: SensorPrivacy mode **/
public class SensorPrivacyTile extends QSTileImpl<BooleanState> implements
SensorPrivacyManager.OnSensorPrivacyChangedListener {
```

```
private static final String TAG = "SensorPrivacy";
// define the icon for the sensor off tile
private final Icon mIcon =
ResourceIcon.get(R.drawable.ic_signal_sensors);
// create sensor manager for disabling sensors
private final SensorPrivacyManager mSensorPrivacyManager;
private final ActivityStarter mActivityStarter;


@Inject
public SensorPrivacyTile(QSHost host, SensorPrivacyManager sensorPrivacyManager,
ActivityStarter activityStarter) {
        super(host);

        mSensorPrivacyManager = sensorPrivacyManager;
        mActivityStarter = activityStarter;
}

// Return when the values in toggle changes true/false
@Override
public BooleanState newTileState() {
        return new BooleanState();
}

@Override
public void handleClick() {
        // change value on click
        final boolean wasEnabled = mState.value;
        // Don't allow disabling from the lockscreen.
        if (wasEnabled) {
                mActivityStarter.postQSRunnableDismissingKeyguard(() -> {
                        MetricsLogger.action(mContext, getMetricsCategory(), !wasEnabled);
                        setEnabled(!wasEnabled);
                });
                return;
        }

        MetricsLogger.action(mContext, getMetricsCategory(), !wasEnabled);
        setEnabled(!wasEnabled);
}
```

```
// disable sensors on sensor tile change
private void setEnabled(boolean enabled) {
        mSensorPrivacyManager.setSensorPrivacy(enabled);
}

// set sensor tile title
@Override
public CharSequence getTileLabel() {
        return mContext.getString(R.string.sensor_privacy_mode);
}

@Override
public Intent getLongClickIntent() {
        return new Intent();
}

@Override
protected void handleUpdateState(BooleanState state, Object arg) {
        // set boolean on sensors tile change
        final boolean enabled = arg instanceof Boolean ? (Boolean) arg
        : mSensorPrivacyManager.isSensorPrivacyEnabled();
        state.value = enabled;
        // set title
        state.label = mContext.getString(R.string.sensor_privacy_mode);
        // set icons
        state.icon = mIcon;
        state.state = enabled ? Tile.STATE_ACTIVE : Tile.STATE_INACTIVE;
        state.contentDescription = state.label;
        state.expandedAccessibilityClassName = Switch.class.getName();
}

@Override
public int getMetricsCategory() {
        // set metrices for reporting logs
        return MetricsEvent.QS_SENSOR_PRIVACY;
}

@Override
protected String composeChangeAnnouncement() {
```

```
        if (mState.value) {
                return mContext
                .getString(R.string.accessibility_quick_settings_sensor_privacy_changed_on);
        } else {
                return mContext
                .getString(R.string.accessibility_quick_settings_sensor_privacy_changed_off);
        }
        }

        @Override
        protected void handleSetListening(boolean listening) {
                if (listening) {
                        mSensorPrivacyManager.addSensorPrivacyListener(this);
                } else {
                        mSensorPrivacyManager.removeSensorPrivacyListener(this);
                }
        }

        @Override
        public void onSensorPrivacyChanged(boolean enabled) {
                # return on the tile behaviour change
                refreshState(enabled);
        }
}
```

7.1.2 PegasusOS version

```
package com.android.settings.deviceinfo.firmwareversion;

import android.content.Context;
import android.os.SystemProperties;

import androidx.annotation.VisibleForTesting;

import com.android.settings.R;
import com.android.settings.Utils;
import com.android.settings.core.BasePreferenceController;

public class PegasusOSVersionPreferenceController extends BasePreferenceController {

        // Define the property which we need to pick value for pegasusos version
        @VisibleForTesting
        static final String PEGASUSOS_VERSION_PROPERTY = "ro.pegasusos.version";

        public PegasusOSVersionPreferenceController(Context context, String preferenceKey) {
                super(context, preferenceKey);
        }

        // Return availability status
        @Override
        public int getAvailabilityStatus() {
                return AVAILABLE;
        }

        // Return the property to summary to show the version
        @Override
        public CharSequence getSummary() {
                return SystemProperties.get(PEGASUSOS_VERSION_PROPERTY,
                mContext.getString(R.string.unknown));
        }
}
```

7.1.3 Permission Manager

```
package com.android.settings.privacy;

import static android.Manifest.permission_group.CAMERA;
import static android.Manifest.permission_group.LOCATION;
import static android.Manifest.permission_group.MICROPHONE;

import static java.util.concurrent.TimeUnit.DAYS;

import android.content.Context;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.graphics.drawable.Drawable;
import android.os.Bundle;
import android.permission.PermissionControllerManager;
import android.permission.RuntimePermissionUsageInfo;
import android.provider.DeviceConfig;
import android.util.Log;
import android.view.View;

import androidx.annotation.NonNull;
import androidx.annotation.VisibleForTesting;
import androidx.preference.PreferenceScreen;

import com.android.settings.R;
import com.android.settings.core.BasePreferenceController;
import com.android.settingslib.Utils;
import com.android.settingslib.core.lifecycle.LifecycleObserver;
import com.android.settingslib.core.lifecycle.events.OnCreate;
import com.android.settingslib.core.lifecycle.events.OnSaveInstanceState;
import com.android.settingslib.core.lifecycle.events.OnStart;
import com.android.settingslib.widget.BarChartInfo;
import com.android.settingslib.widget.BarChartPreference;
import com.android.settingslib.widget.BarViewInfo;

import java.util.ArrayList;
import java.util.List;


public class PermissionBarChartPreferenceController extends BasePreferenceController implements
```

```
PermissionControllerManager.OnPermissionUsageResultCallback, LifecycleObserver, OnCreate,
OnStart, OnSaveInstanceState {

        private static final String TAG = "BarChartPreferenceCtl";
        private static final String KEY_PERMISSION_USAGE = "usage_infos";

        @VisibleForTesting
        List<RuntimePermissionUsageInfo> mOldUsageInfos;
        private PackageManager mPackageManager;
        private PrivacyDashboardFragment mParent;
        private BarChartPreference mBarChartPreference;

        public PermissionBarChartPreferenceController(Context context, String preferenceKey) {
                super(context, preferenceKey);
                mOldUsageInfos = new ArrayList<>();
                mPackageManager = context.getPackageManager();
        }

        public void setFragment(PrivacyDashboardFragment fragment) {
                mParent = fragment;
        }

        @Override
        public void onCreate(Bundle savedInstanceState) {
        if (savedInstanceState != null) {
                mOldUsageInfos = savedInstanceState.getParcelableArrayList(KEY_PERMISSION_USAGE);
        }
        }

        @Override
        public void onSaveInstanceState(Bundle outState) {
                outState.putParcelableList(KEY_PERMISSION_USAGE, mOldUsageInfos);
        }

        @Override
        public int getAvailabilityStatus() {
                return Boolean.parseBoolean(
                DeviceConfig.getProperty(DeviceConfig.NAMESPACE_PRIVACY,
                com.android.settings.Utils.PROPERTY_PERMISSIONS_HUB_ENABLED)) ?
                AVAILABLE_UNSEARCHABLE : UNSUPPORTED_ON_DEVICE;
```

```
}

@Override
public void displayPreference(PreferenceScreen screen) {
        super.displayPreference(screen);
        mBarChartPreference = screen.findPreference(getPreferenceKey());

        // set default values for title
        final BarChartInfo info = new BarChartInfo.Builder()
        .setTitle(R.string.permission_bar_chart_title)
        .setDetails(R.string.permission_bar_chart_details)
        .setEmptyText(R.string.permission_bar_chart_empty_text)
        .setDetailsOnClickListener((View v) -> {
                final Intent intent = new Intent(Intent.ACTION_REVIEW_PERMISSION_USAGE);
                intent.putExtra(Intent.EXTRA_DURATION_MILLIS, DAYS.toMillis(1));
                mContext.startActivity(intent);
        })
        .build();

        mBarChartPreference.initializeBarChart(info);
        if (!mOldUsageInfos.isEmpty()) {
                mBarChartPreference.setBarViewInfos(createBarViews(mOldUsageInfos));
        }
}

@Override
public void onStart() {
if (!isAvailable()) {
        return;
}

// Add a shadow animation to action bar scroll only when the chart is available.
com.android.settings.Utils.setActionBarShadowAnimation(mParent.getActivity(),
mParent.getSettingsLifecycle(), mParent.getListView());
// We don't hide chart when we have existing data.
mBarChartPreference.updateLoadingState(mOldUsageInfos.isEmpty() /* isLoading */);
// But we still need to hint user with progress bar that we are updating new usage data.
mParent.showPinnedHeader(true);
retrievePermissionUsageData();
}
```

```
@Override
public void onPermissionUsageResult(@NonNull List<RuntimePermissionUsageInfo> usageInfos) {
        // return values with respect to the xname and yname
        usageInfos.sort((x, y) -> {
                int usageDiff = y.getAppAccessCount() - x.getAppAccessCount();
                if (usageDiff != 0) {
                        return usageDiff;
                }
                String xName = x.getName();
                String yName = y.getName();
                if (xName.equals(LOCATION)) {
                        return -1;
                } else if (yName.equals(LOCATION)) {
                        return 1;
                } else if (xName.equals(MICROPHONE)) {
                        return -1;
                } else if (yName.equals(MICROPHONE)) {
                        return 1;
                } else if (xName.equals(CAMERA)) {
                        return -1;
                } else if (yName.equals(CAMERA)) {
                        return 1;
                }
                return x.getName().compareTo(y.getName());
        });

        // If the result is different, we need to update bar views.
        if (!areSamePermissionGroups(usageInfos)) {
                mBarChartPreference.setBarViewInfos(createBarViews(usageInfos));
                mOldUsageInfos = usageInfos;
        }

        mBarChartPreference.updateLoadingState(false /* isLoading */);
        mParent.showPinnedHeader(false);
}

private void retrievePermissionUsageData() {
        mContext.getSystemService(PermissionControllerManager.class).getPermissionUsages(
        false /* countSystem */, (int) DAYS.toMillis(1),
```

```
        mContext.getMainExecutor() /* executor */, this /* callback */);
}

private BarViewInfo[] createBarViews(List<RuntimePermissionUsageInfo> usageInfos) {
        if (usageInfos.isEmpty()) {
                return null;
        }

final BarViewInfo[] barViewInfos = new BarViewInfo[
Math.min(BarChartPreference.MAXIMUM_BAR_VIEWS, usageInfos.size())];

for (int index = 0; index < barViewInfos.length; index++) {
final RuntimePermissionUsageInfo permissionGroupInfo = usageInfos.get(index);
final int count = permissionGroupInfo.getAppAccessCount();
final CharSequence permLabel = getPermissionGroupLabel(permissionGroupInfo.getName());

barViewInfos[index] = new BarViewInfo(
getPermissionGroupIcon(permissionGroupInfo.getName()), count, permLabel,
mContext.getResources().getQuantityString(R.plurals.permission_bar_chart_label,
count, count), permLabel);

// Set the click listener for each bar view.
// The listener will navigate user to permission usage app.
barViewInfos[index].setClickListener((View v) -> {
final Intent intent = new Intent(Intent.ACTION_REVIEW_PERMISSION_USAGE);
intent.putExtra(Intent.EXTRA_PERMISSION_GROUP_NAME, permissionGroupInfo.getName());
intent.putExtra(Intent.EXTRA_DURATION_MILLIS, DAYS.toMillis(1));
mContext.startActivity(intent);
});
}

return barViewInfos;
}

// set default icon changes like accent color
private Drawable getPermissionGroupIcon(String permissionGroup) {
        Drawable icon = null;
        try {
        icon = mPackageManager.getPermissionGroupInfo(permissionGroup, 0)
        .loadIcon(mPackageManager);
```

```
                icon.setTintList(Utils.getColorAttr(mContext, android.R.attr.textColorSecondary));
                } catch (PackageManager.NameNotFoundException e) {
                Log.w(TAG, "Cannot find group icon for " + permissionGroup, e);
                }

                return icon;
        }

        private CharSequence getPermissionGroupLabel(String permissionGroup) {
                CharSequence label = null;
                try {
                        label = mPackageManager.getPermissionGroupInfo(permissionGroup, 0)
                        .loadLabel(mPackageManager);
                } catch (PackageManager.NameNotFoundException e) {
                        Log.w(TAG, "Cannot find group label for " + permissionGroup, e);
                }

                return label;
        }

        # returns if its in same permission group
        private boolean areSamePermissionGroups(List<RuntimePermissionUsageInfo> newUsageInfos) {
                if (newUsageInfos.size() != mOldUsageInfos.size()) {
                        return false;
                }

                for (int index = 0; index < newUsageInfos.size(); index++) {
                        final RuntimePermissionUsageInfo newInfo = newUsageInfos.get(index);
                        final RuntimePermissionUsageInfo oldInfo = mOldUsageInfos.get(index);

                        if (!newInfo.getName().equals(oldInfo.getName()) ||
                        newInfo.getAppAccessCount() != oldInfo.getAppAccessCount()) {
                                return false;
                        }
                }
                return true;
        }
}
```

7.1.4 Protect USB

```
package com.android.settings.security;

import android.content.Context;
import android.os.UserHandle;
import android.os.UserManager;
import android.os.SystemProperties;
import android.provider.Settings;
import androidx.preference.ListPreference;
import androidx.preference.Preference;
import androidx.preference.PreferenceCategory;
import androidx.preference.PreferenceGroup;
import androidx.preference.PreferenceScreen;
import com.android.internal.widget.LockPatternUtils;
import com.android.settings.core.PreferenceControllerMixin;
import com.android.settingslib.core.AbstractPreferenceController;
import com.android.settingslib.core.lifecycle.events.OnResume;

public class DenyNewUsbPreferenceController extends AbstractPreferenceController
implements PreferenceControllerMixin, OnResume, Preference.OnPreferenceChangeListener {

        // default properties for usb deny values
        private static final String KEY_DENY_NEW_USB = "deny_new_usb";
        private static final String DENY_NEW_USB_PROP = "security.deny_new_usb";
        private static final String DENY_NEW_USB_PERSIST_PROP = "persist.security.deny_new_usb";
        private static final String PREF_KEY_SECURITY_CATEGORY = "security_category";

        private PreferenceCategory mSecurityCategory;
        private ListPreference mDenyNewUsb;
        private boolean mIsAdmin;
        private UserManager mUm;

        public DenyNewUsbPreferenceController(Context context) {
                super(context);
                mUm = UserManager.get(context);
        }

        @Override
        public void displayPreference(PreferenceScreen screen) {
                super.displayPreference(screen);
```

```
        mSecurityCategory = screen.findPreference(PREF_KEY_SECURITY_CATEGORY);
        updatePreferenceState();
}

// return if feature is supported (if main user)
@Override
public boolean isAvailable() {
        mIsAdmin = mUm.isAdminUser();
        return mIsAdmin;
}

// return default key value
@Override
public String getPreferenceKey() {
        return KEY_DENY_NEW_USB;
}

private void updatePreferenceState() {
        if (mSecurityCategory == null) {
                return;
        }

// add or remove option for admin and guest
if (mIsAdmin) {
mDenyNewUsb = (ListPreference) mSecurityCategory.findPreference(KEY_DENY_NEW_USB);
mDenyNewUsb.setValue(SystemProperties.get(DENY_NEW_USB_PERSIST_PROP, "disabled"));
} else {
mSecurityCategory.removePreference(mSecurityCategory.findPreference(KEY_DENY_NEW_USB));
}
}

@Override
public void onResume() {
        updatePreferenceState();

        if (mDenyNewUsb != null) {

                // default values  on the property changes
                String mode = mDenyNewUsb.getValue();
                if (mode.equals("dynamic") || mode.equals("disabled")) {
```

```
                        SystemProperties.set(DENY_NEW_USB_PROP, "0");
                } else {
                        SystemProperties.set(DENY_NEW_USB_PROP, "1");
                }
        }
    }

    @Override
    public boolean onPreferenceChange(Preference preference, Object value) {
    final String key = preference.getKey();
    // on key change, change default values  on the property
    if (KEY_DENY_NEW_USB.equals(key)) {
            String mode = (String) value;
            SystemProperties.set(DENY_NEW_USB_PERSIST_PROP, mode);
            // The dynamic mode defaults to the disabled state
            if (mode.equals("dynamic") || mode.equals("disabled")) {
                    SystemProperties.set(DENY_NEW_USB_PROP, "0");
            } else {
                    SystemProperties.set(DENY_NEW_USB_PROP, "1");
            }
    }
    return true;
    }
}
```

7.2 Screenshots

7.2.1 Website homepage

https://pegasusos.github.io This website contains basic information about the operating system and features. Also for more info provided a PDF of the report. It also includes download instructions and contact details for the help.

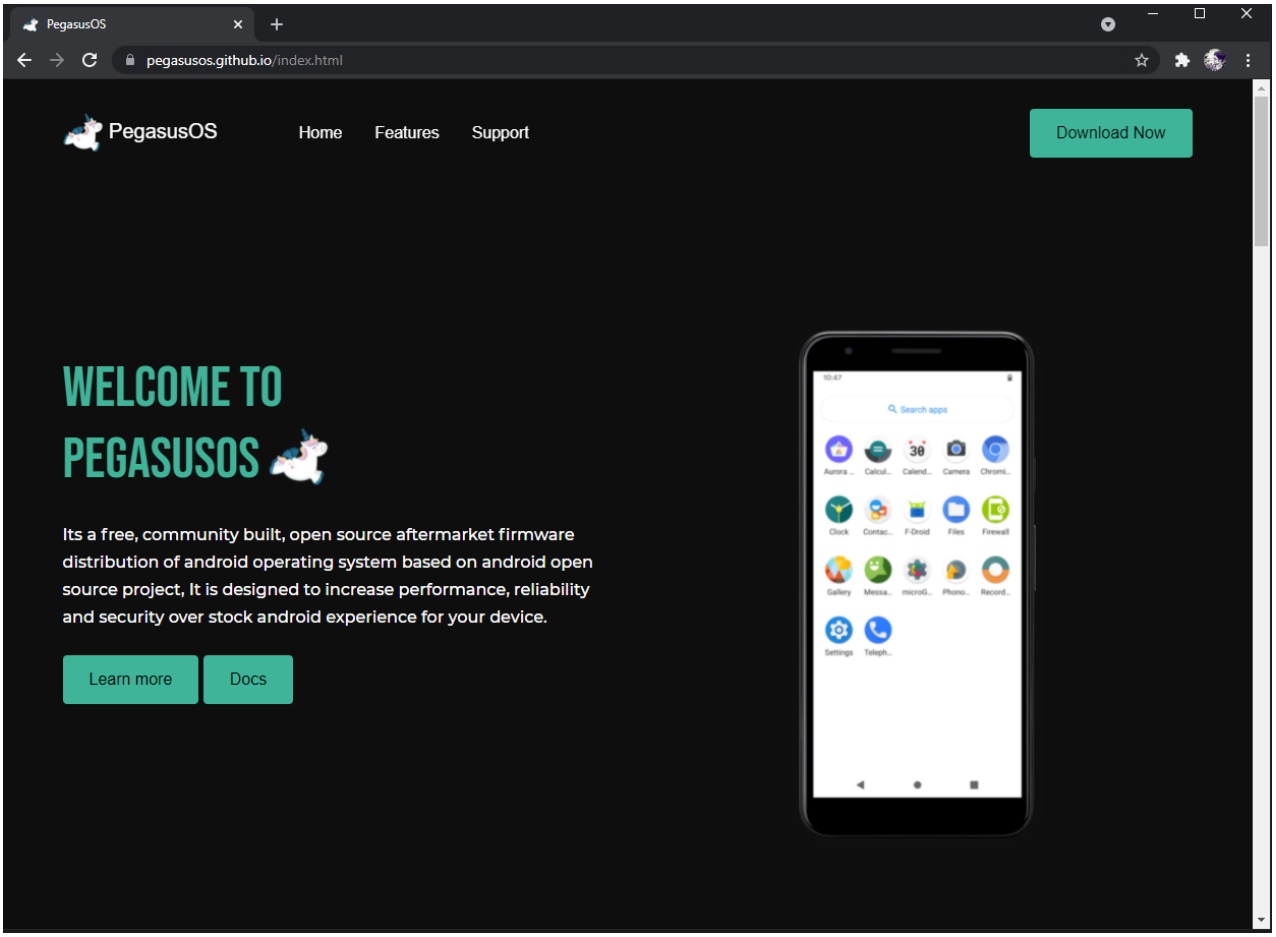


Figure 7.1: Homepage 1

This page contains why you need PegasusOS.

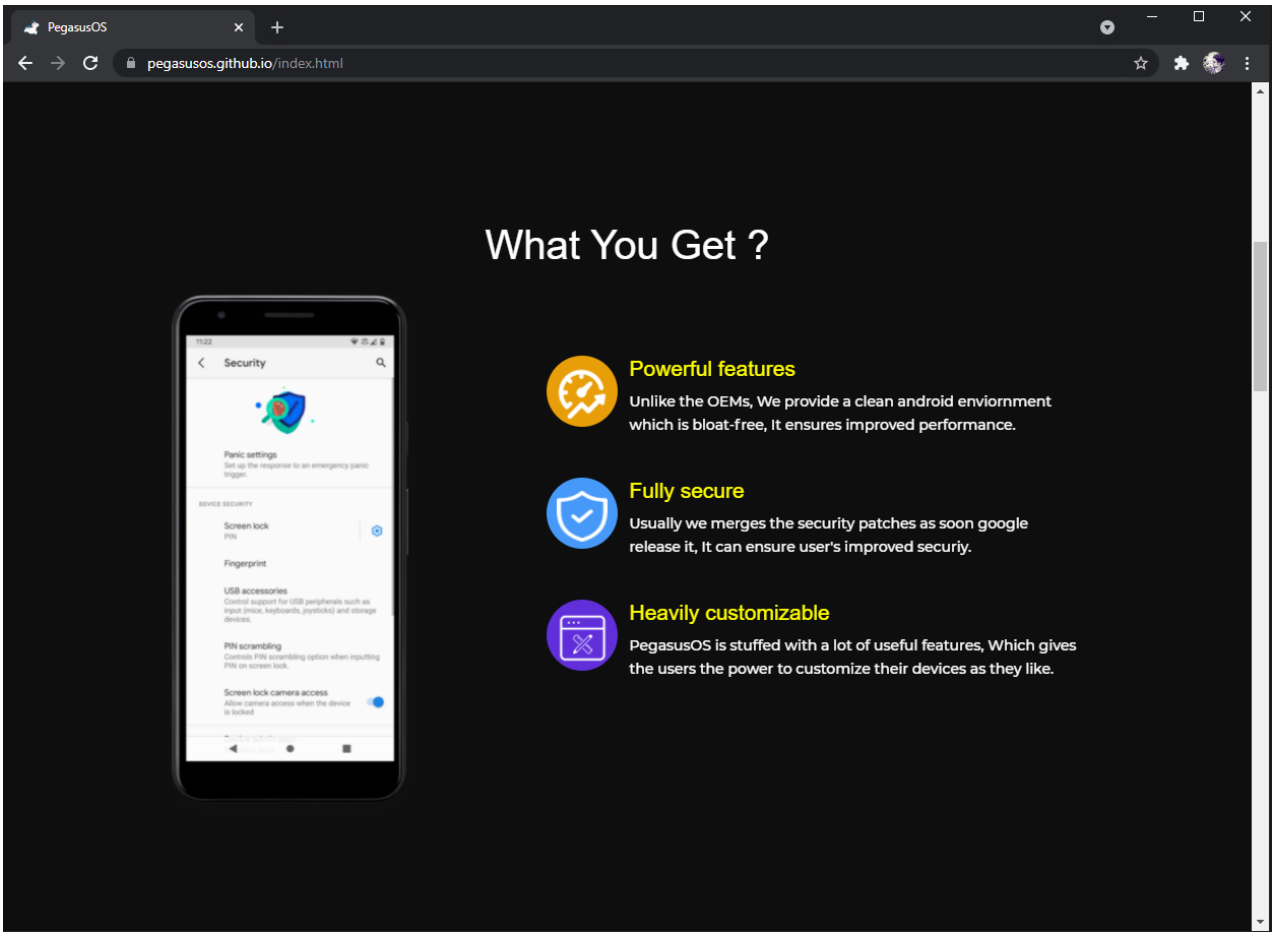


Figure 7.2: Homepage 2

This page contains the features containing in PegasusOS.

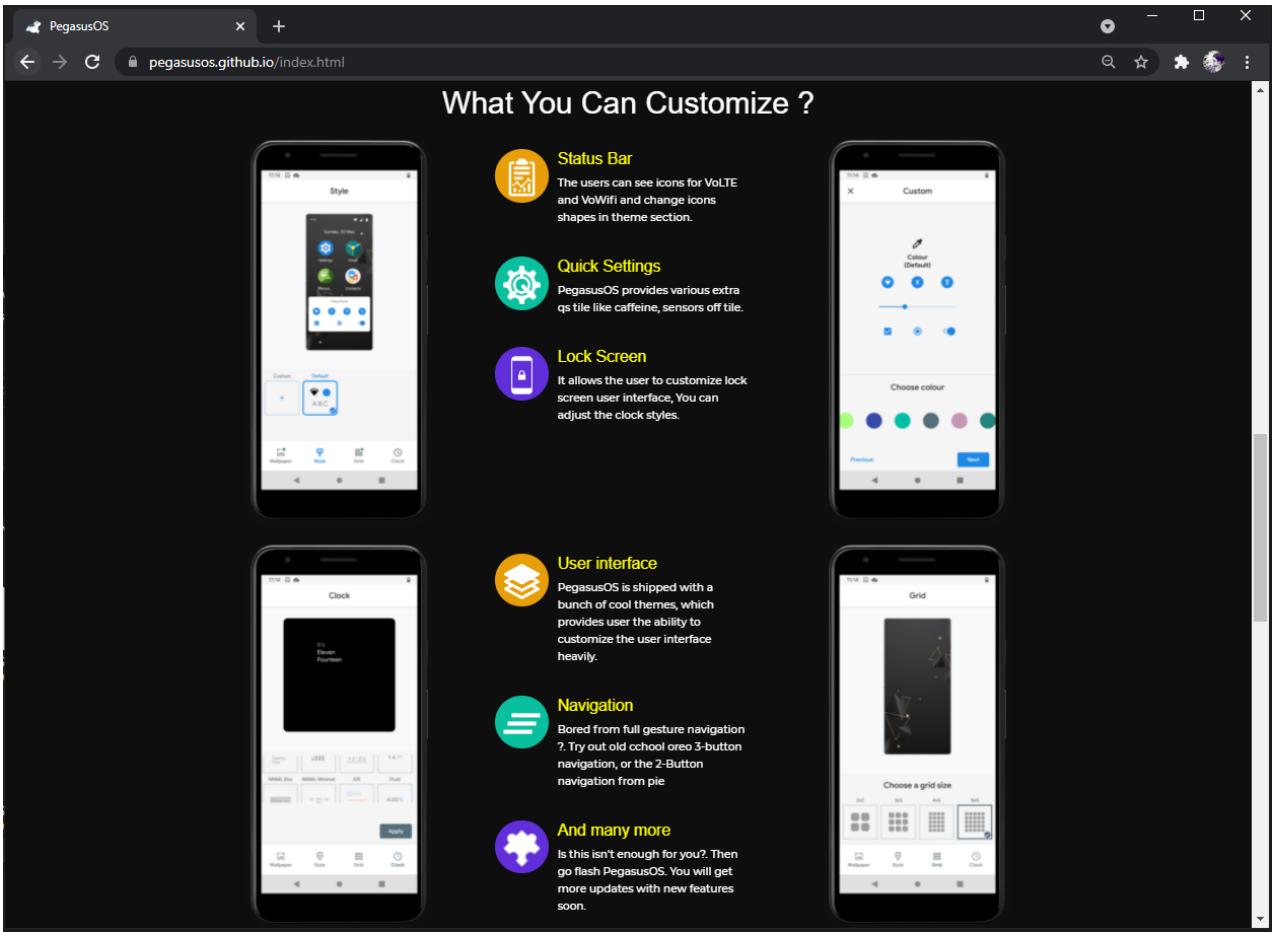


Figure 7.3: Homepage 3

This page contains the download page link, user and developer support links and contact details.

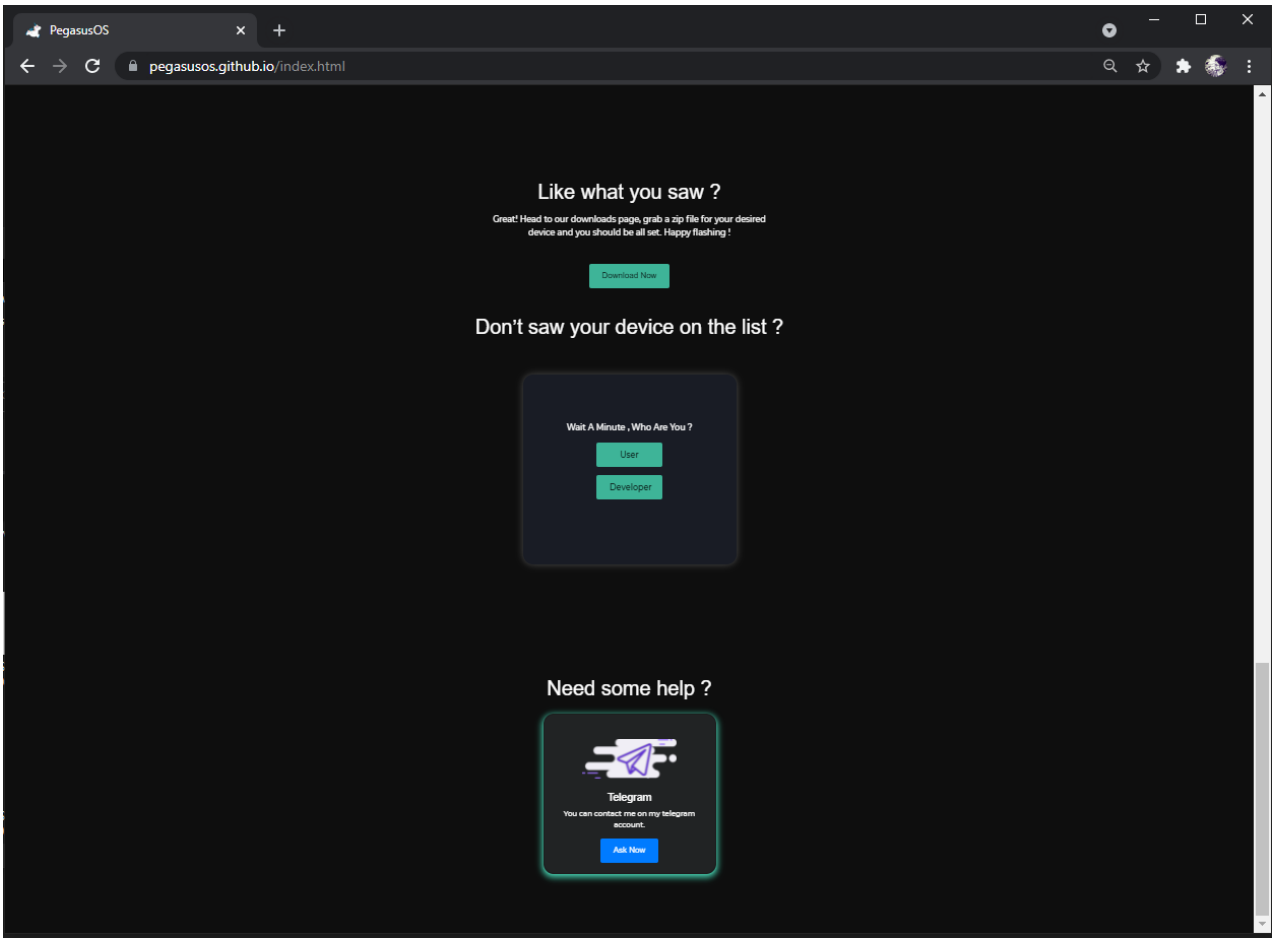


Figure 7.4: Homepage 4

This page contains PegasusOS downloads for supported devices.

7.2.2 Downloads

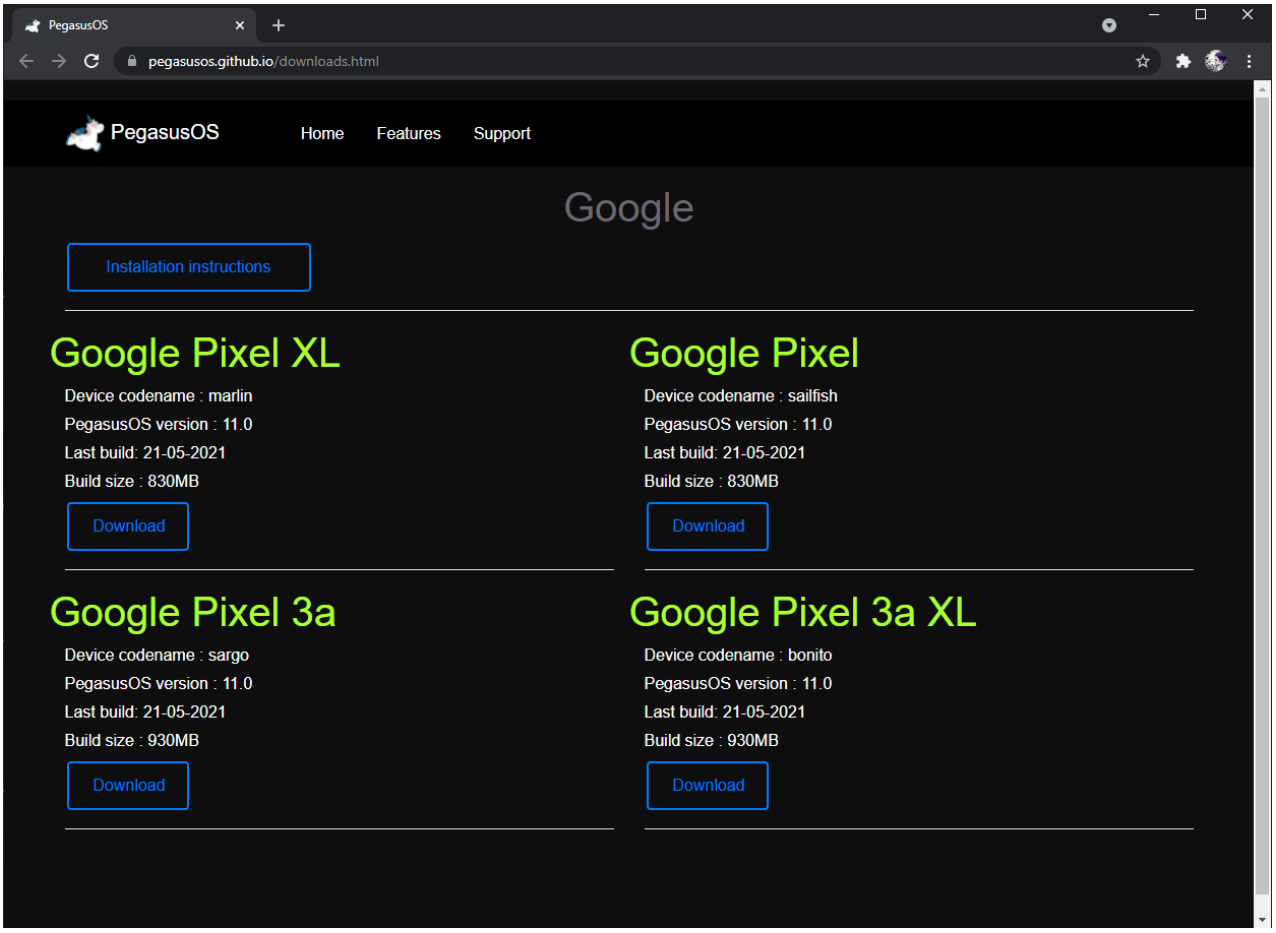


Figure 7.5: Downloads

7.2.3 Installation instructions

This page contains installation instructions for PegasusOS

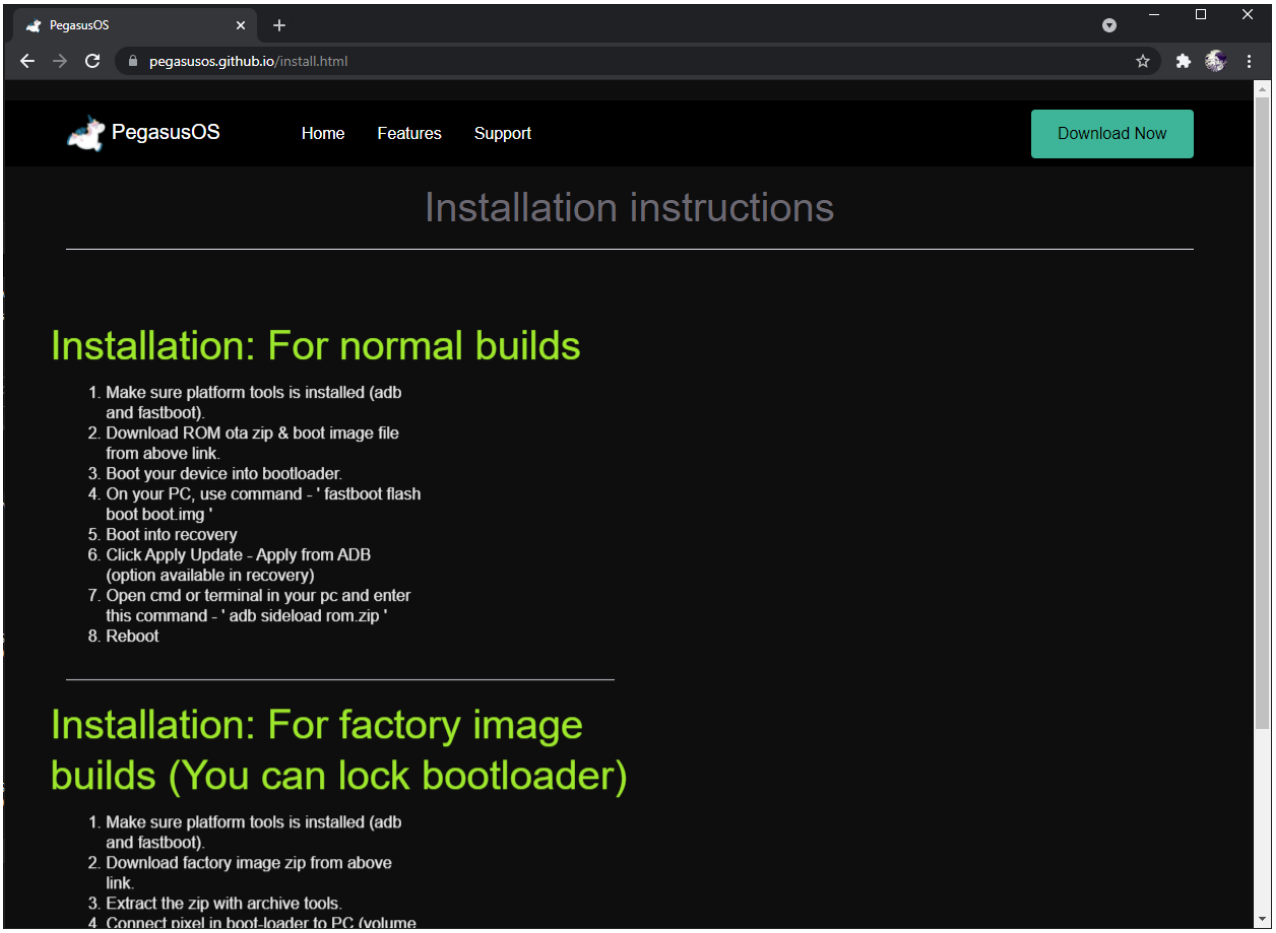


Figure 7.6: Installation instructions

7.2.4 Documentation

This page contains the documents of the project and download option.

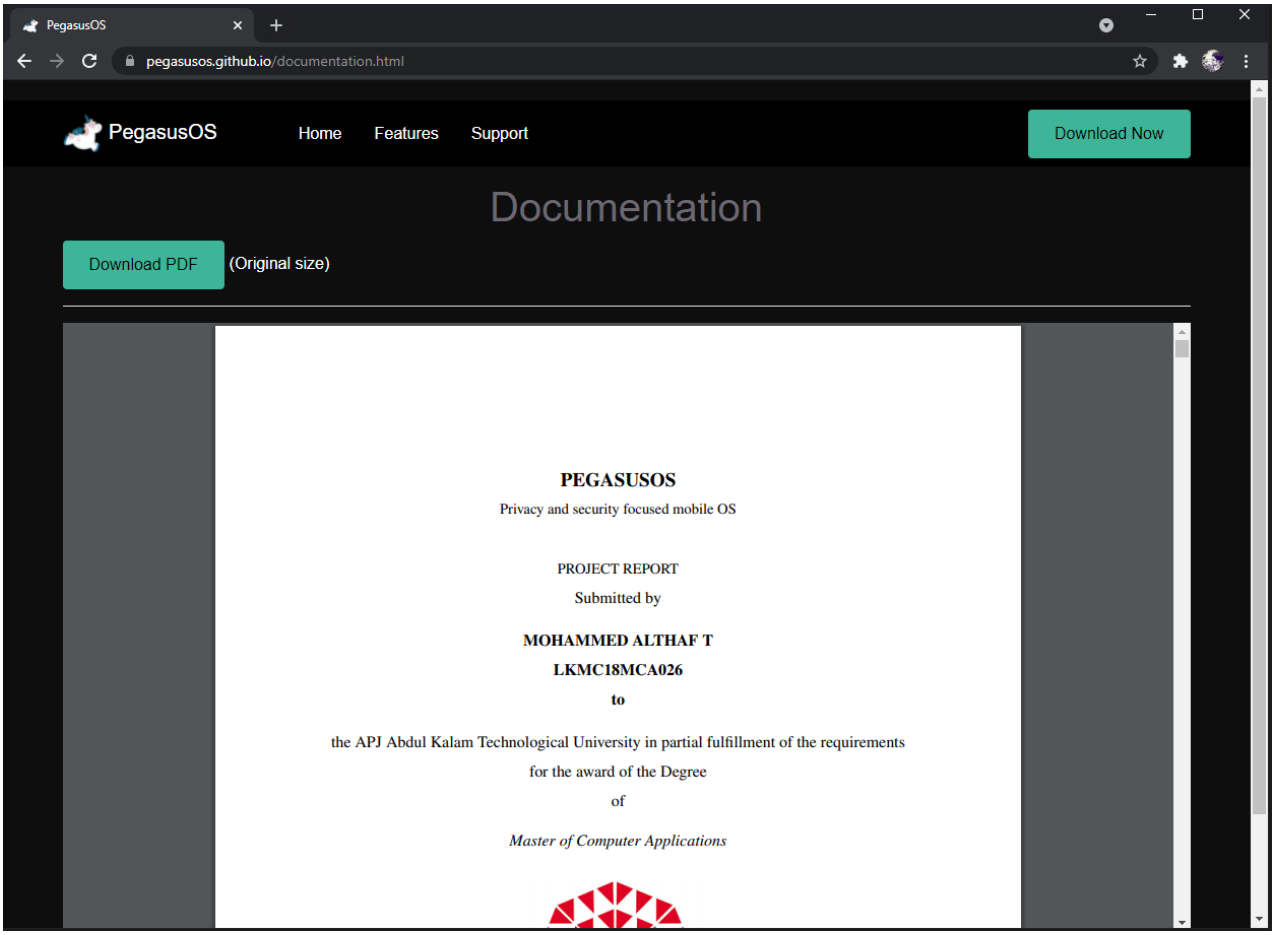


Figure 7.7: Documentation

7.2.5 Bootloader (fastboot mode)

This section in phone allows us to unlock/lock bootloader, access recovery and flash image files for this ROM. We can control with volume up/down and power button.

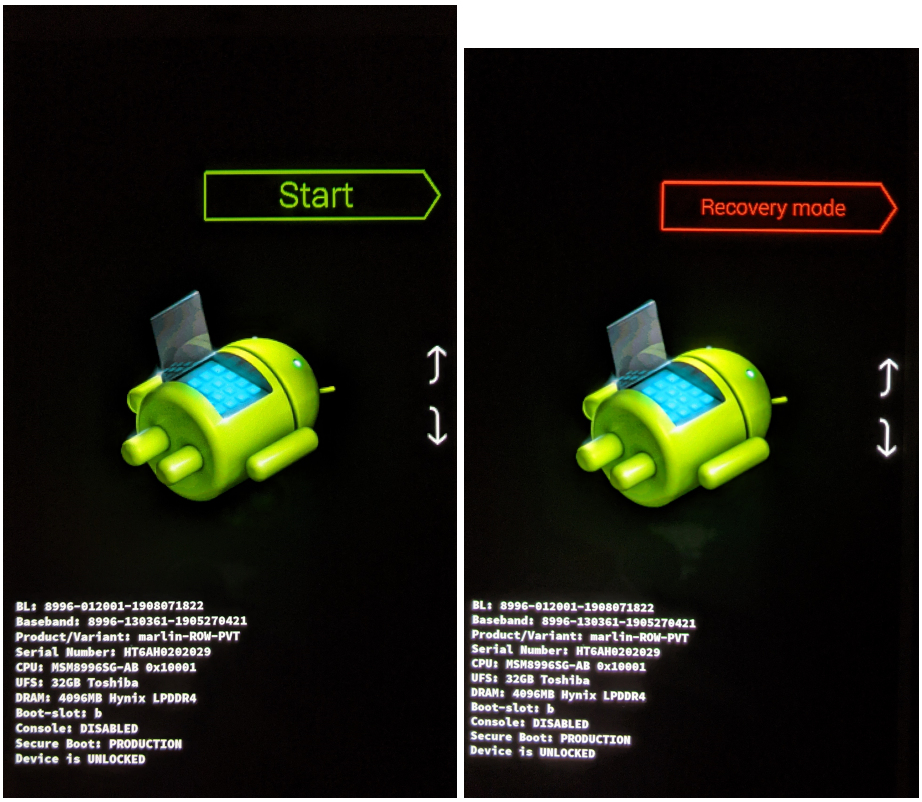


Figure 7.8: Bootloader

7.2.6 Recovery

This section in phone allows us to flashing ROM zip and wiping device. We can control with volume up/down and power button or touchscreen.

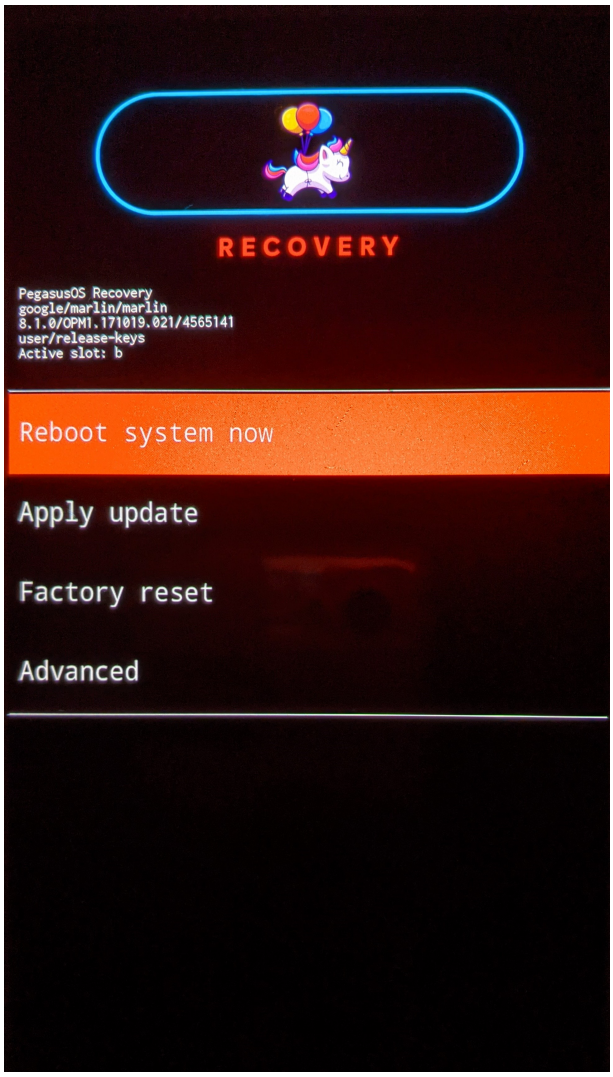


Figure 7.9: Recovery

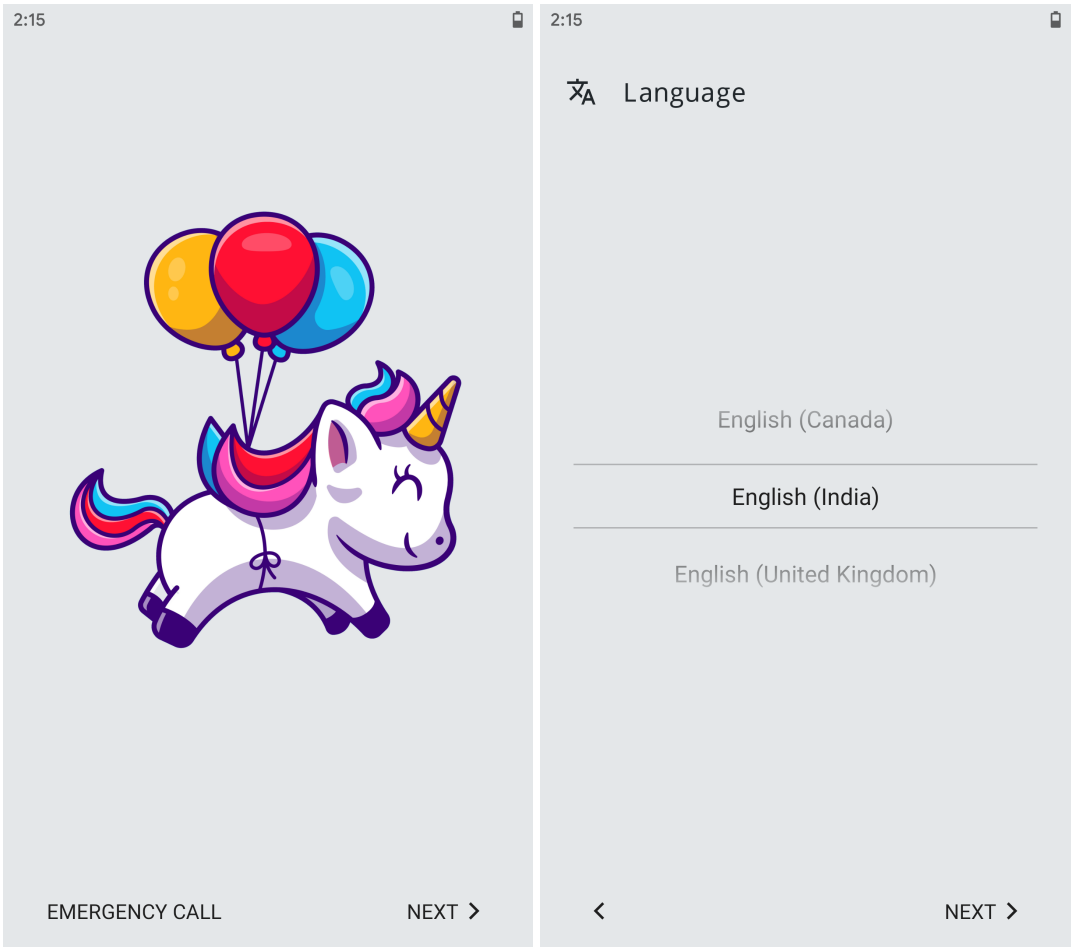
7.2.7 Bootanimation logo

Shown on booting of the OS

Figure 7.10: Bootanimation logo

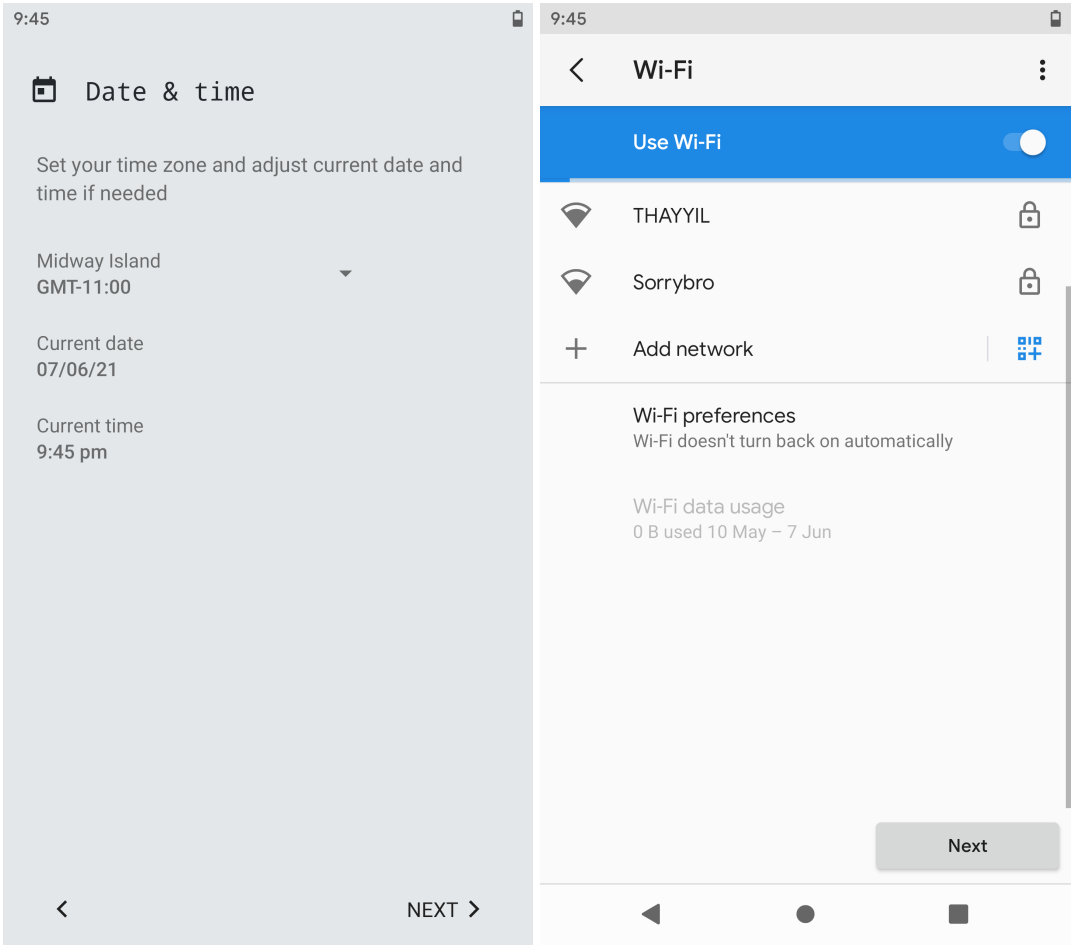
7.2.8 SetupWizard

Initial user setup of device. User can set default language, set time,date and region, give permission for location, enable mobile data, setup password/pin code/pattern lock, restore Seed-vault backup.



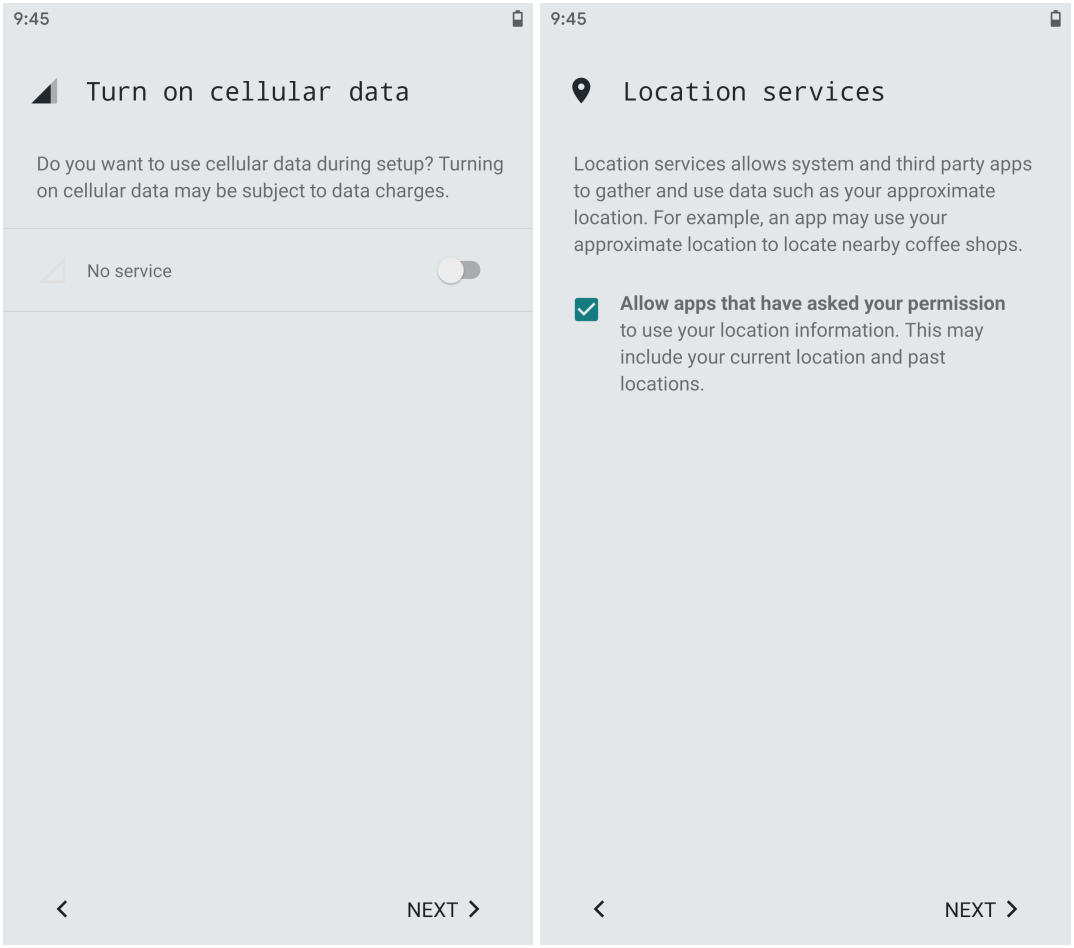
(a) (b)

Figure 7.11: SetupWizard 1



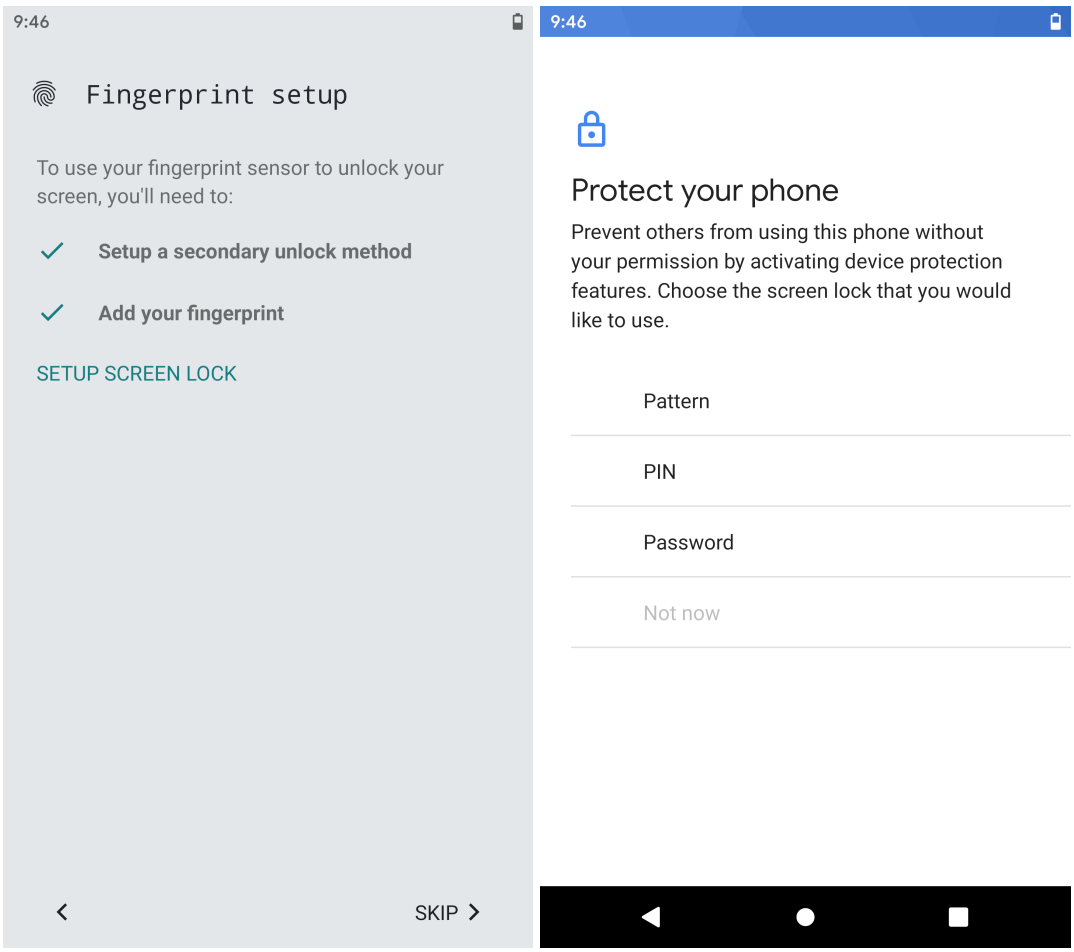
(a) (b)

Figure 7.12: SetupWizard 2



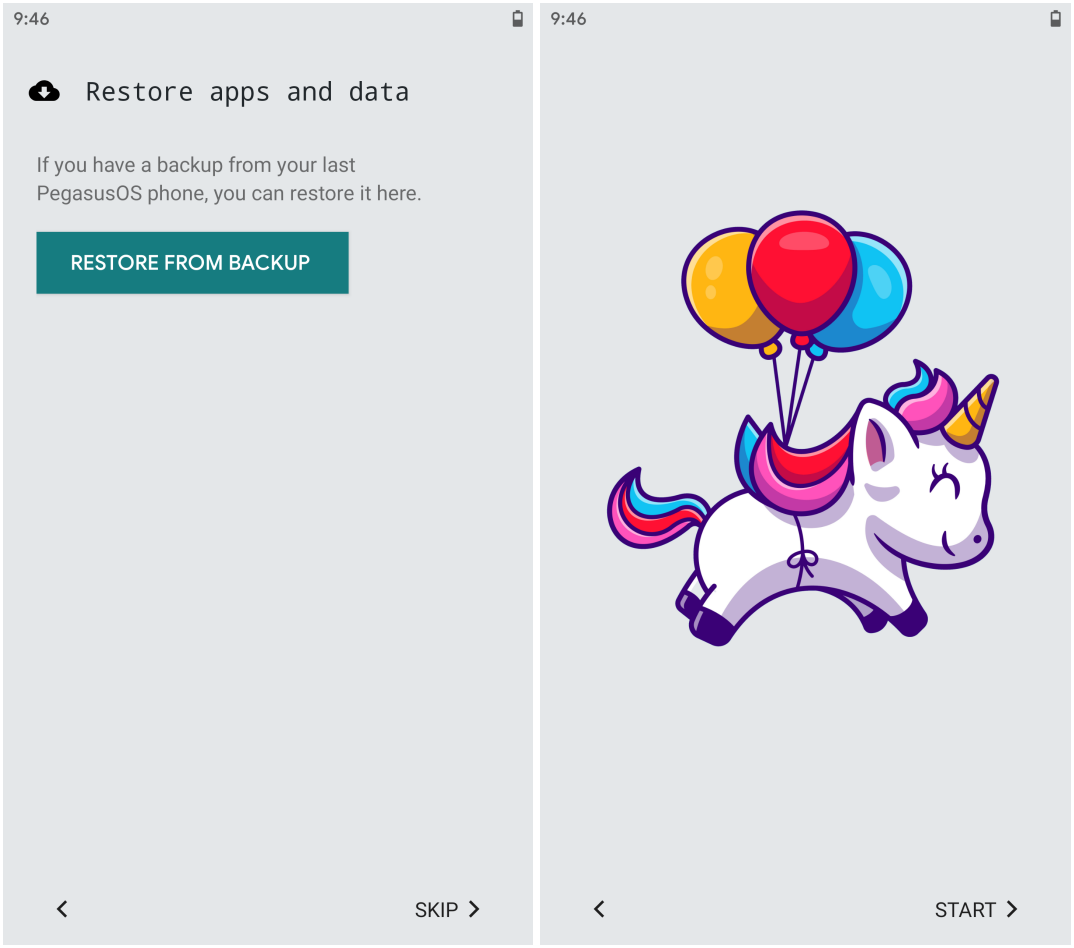
(a) (b)

Figure 7.13: SetupWizard 3



(a) (b)

Figure 7.14: SetupWizard 4

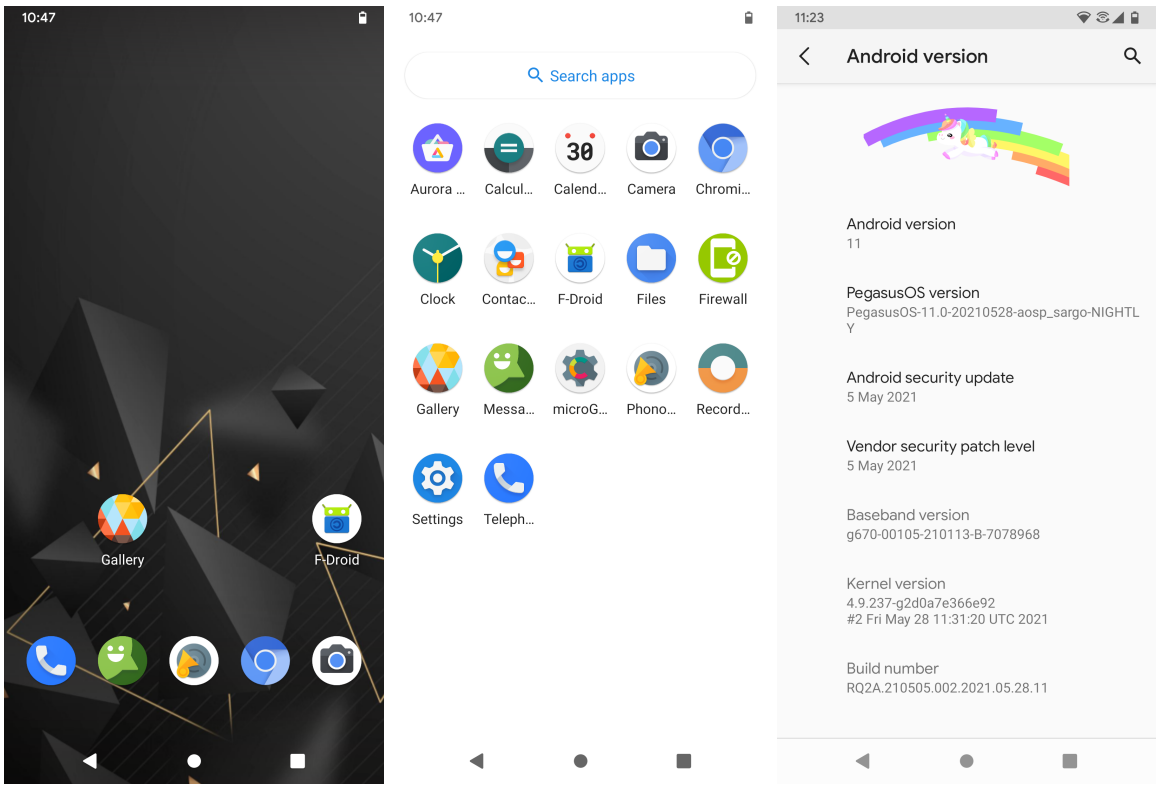


(a) (b)

Figure 7.15: SetupWizard 5

7.2.9 Homescreen & About

First look after phone setupwizard



(a) Homescreen (b) App drawer (c) About section

Figure 7.16: Homescreen & About

7.2.10 Protect USB Port

This options allows users to select difference option to enable or disable peripheral connections while device is locked or unlocked. There are three options to deny all USB, allow new devices while unlocked or allow even while locked.

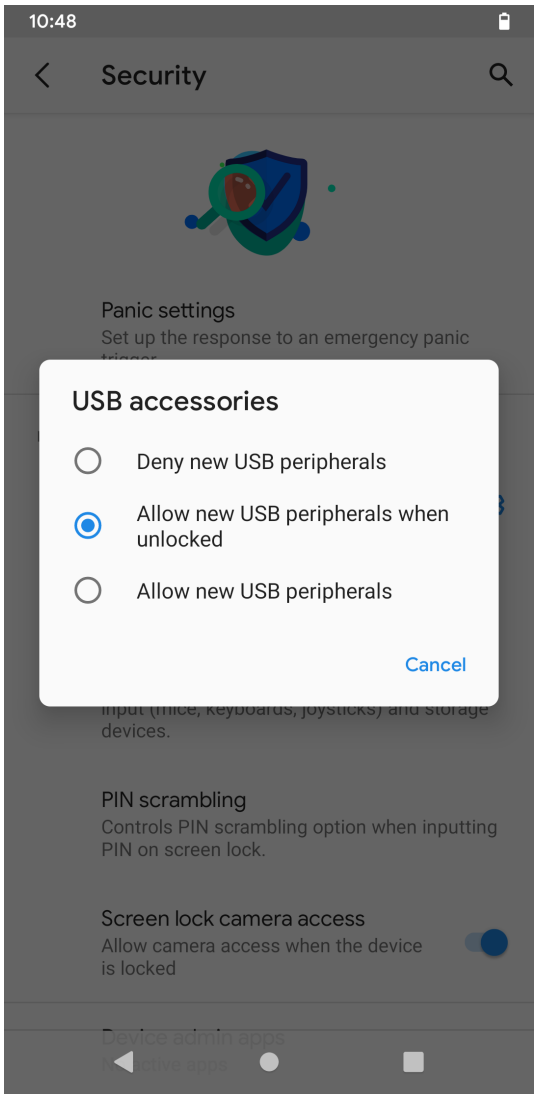
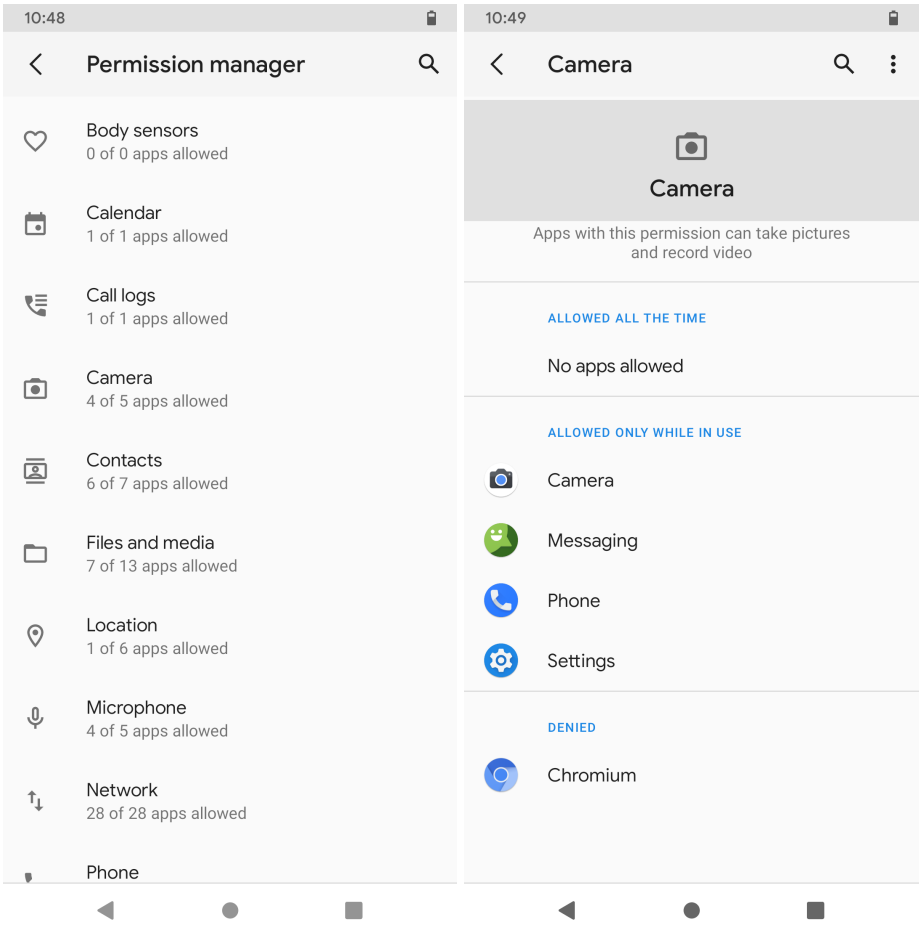


Figure 7.17: Protect USB Port

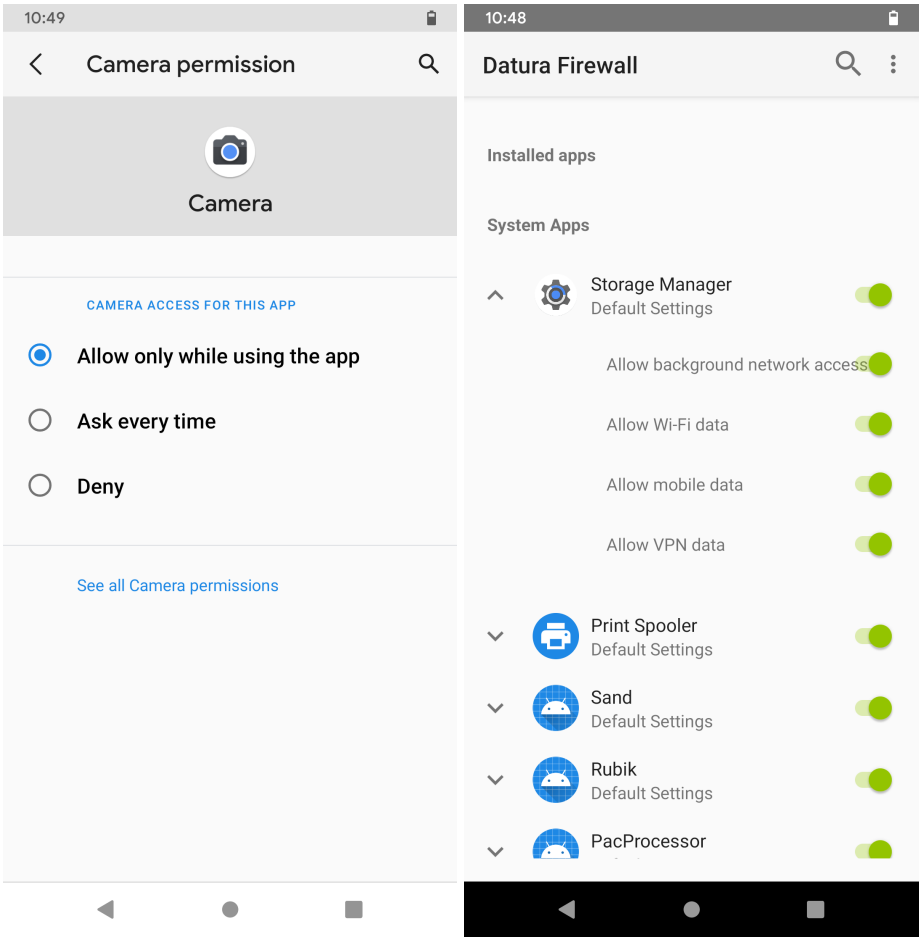
7.2.11 Permissions Manager

This options allows us to view all the permissions and apps that are using these permission. We can also allow or deny permissions by clicking them. Also change data usage for apps and restrict them.



(a) (b)

Figure 7.18: Permissions Manager 1

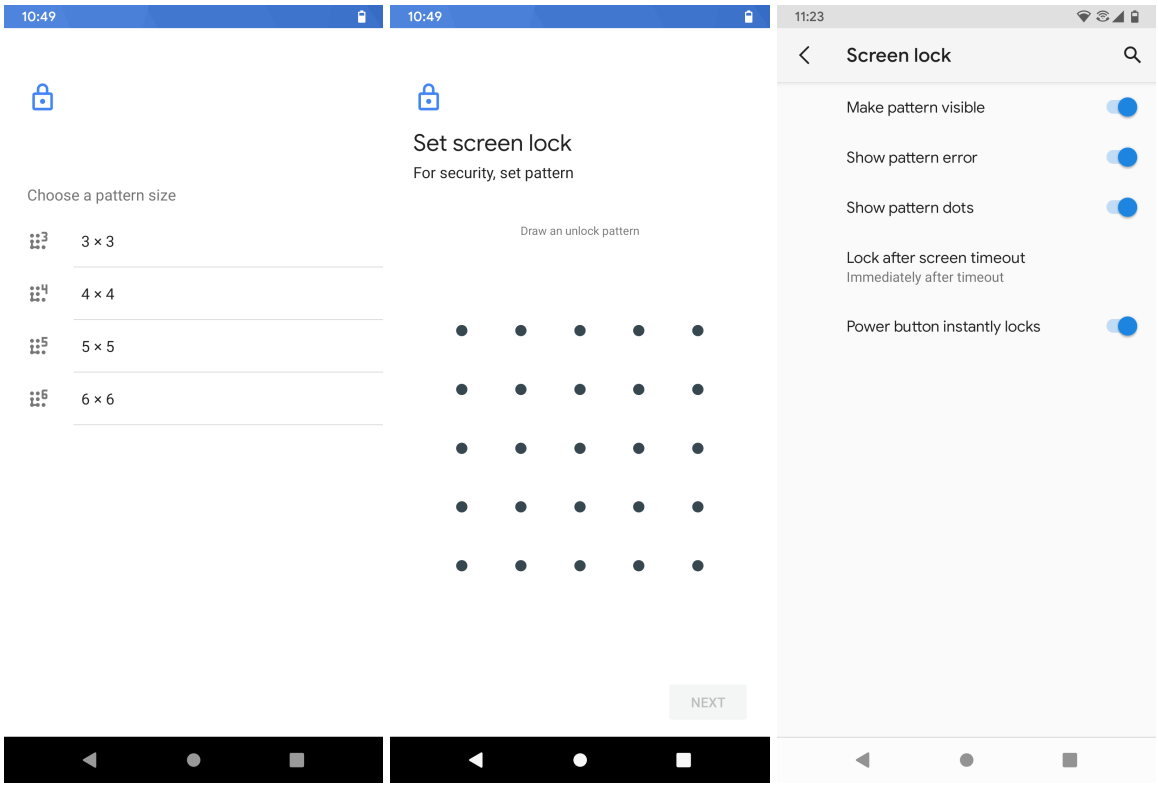


(a) (b)

Figure 7.19: Permissions Manager 2

7.2.12 Pattern Lock size

This options allows us to select desired pattern lock layout size. The bigger the layout theres more ways to lock user device.

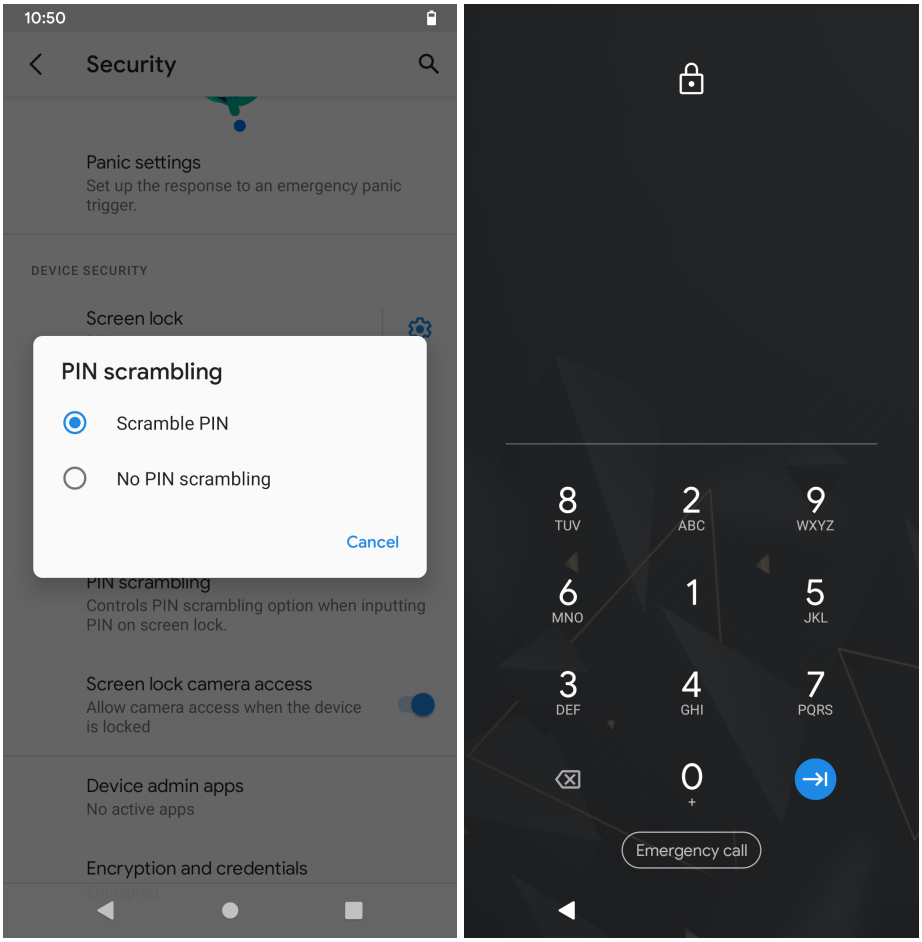


(a) (b) (c)

Figure 7.20: Pattern Lock size

7.2.13 Scramble pin

This options allows us to scramble the pin layout in lock screen. Which allows us to enter pin each time we unlock phone. Less vulnerable with group of people considering normal layout.

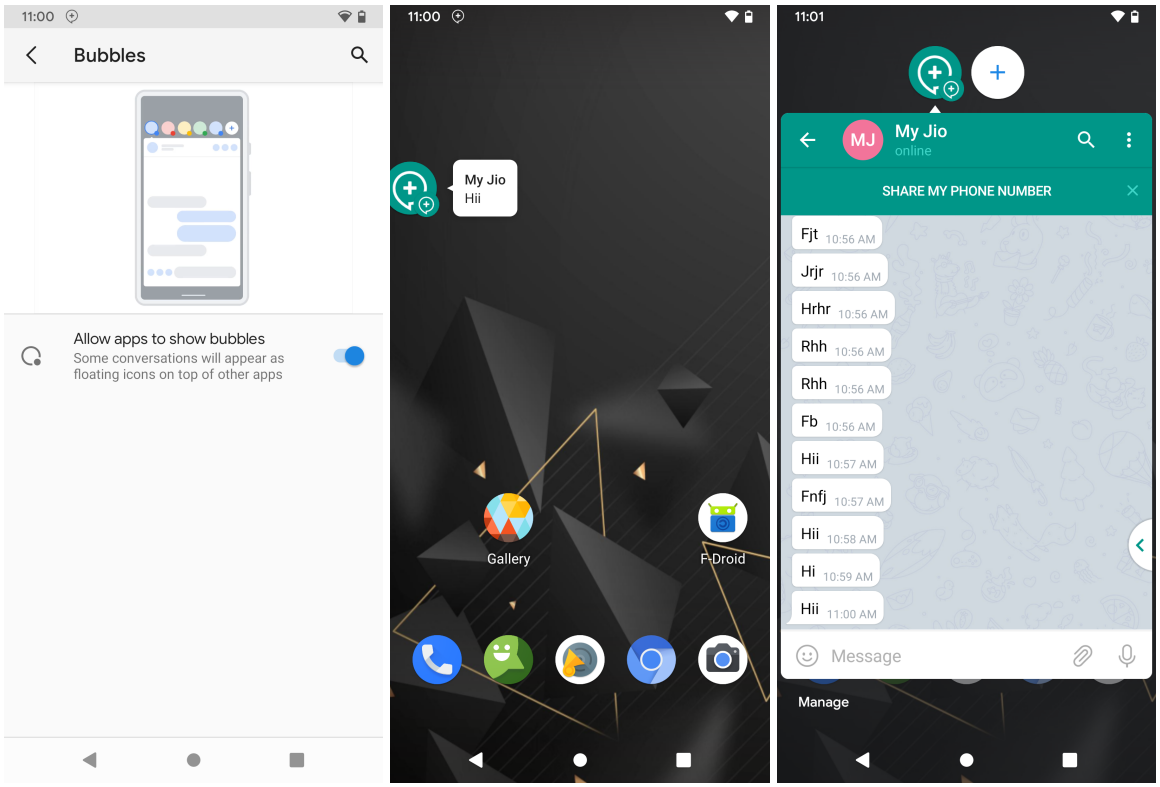


(a) (b)

Figure 7.21: Scramble pin

7.2.14 Bubbles

This feature allows us to use Android-11's chat heads. Its similar to Facebook messenger chat heads, but supports more apps and can be used for contacts specific.

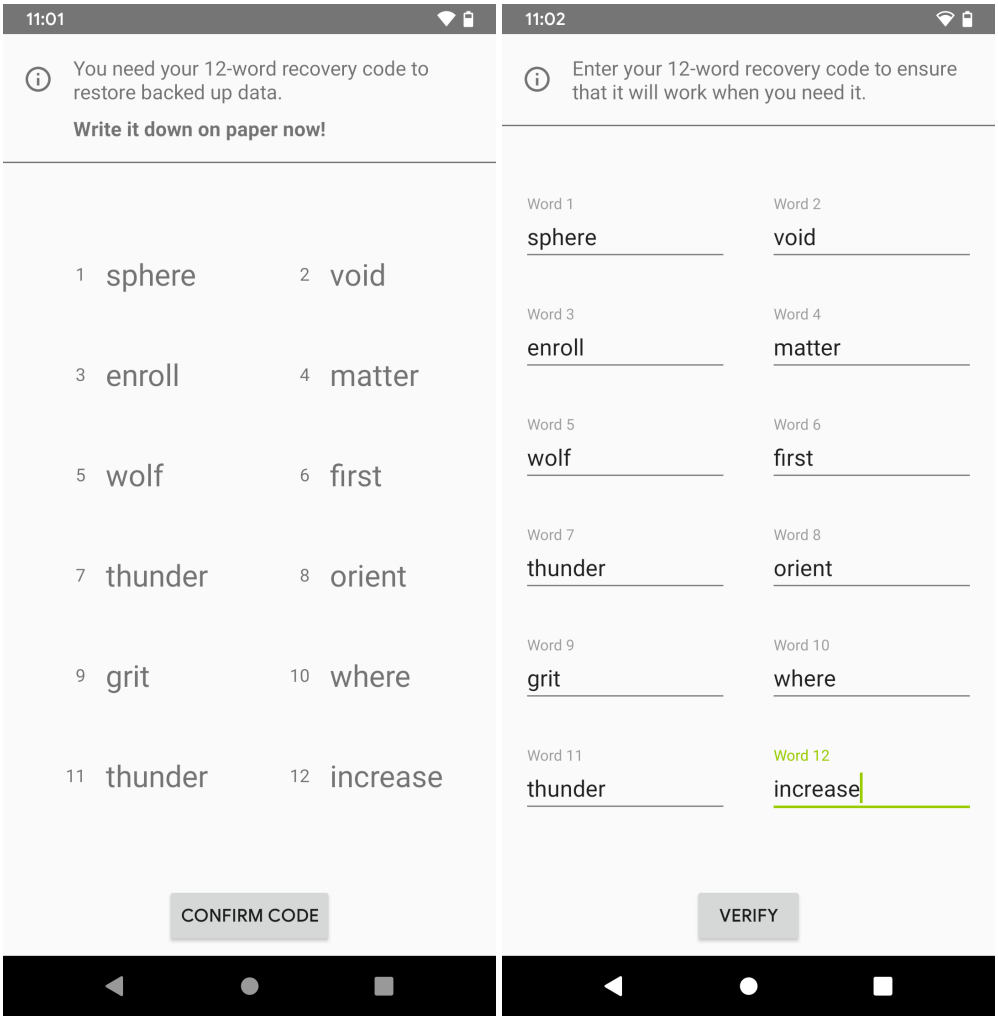


(a) (b) (c)

Figure 7.22: Bubbles

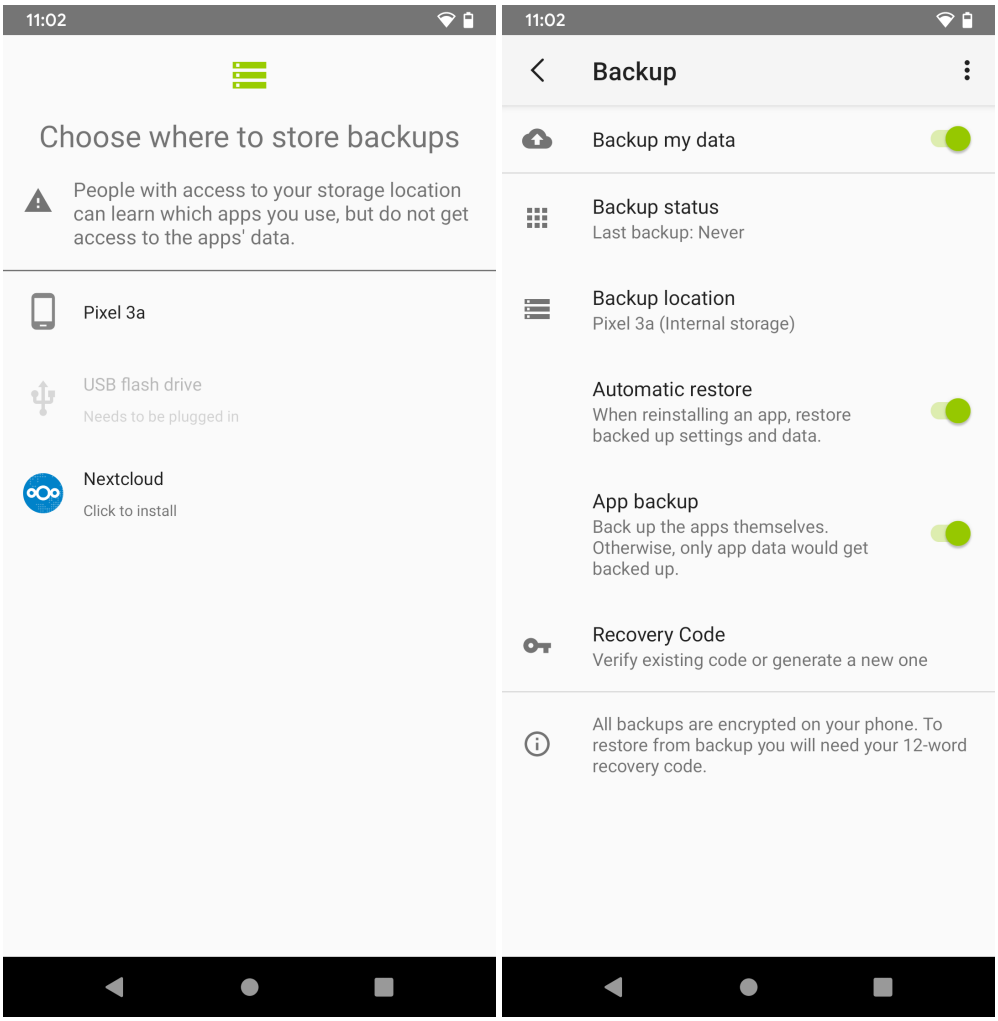
7.2.15 Data Backup

This feature allow us to backup our data easily with Seed-vault. Its a user-friendly encryption using a mnemonic phrase. We can backup the data into cloud, USB flash drive and restore later with the phrases.



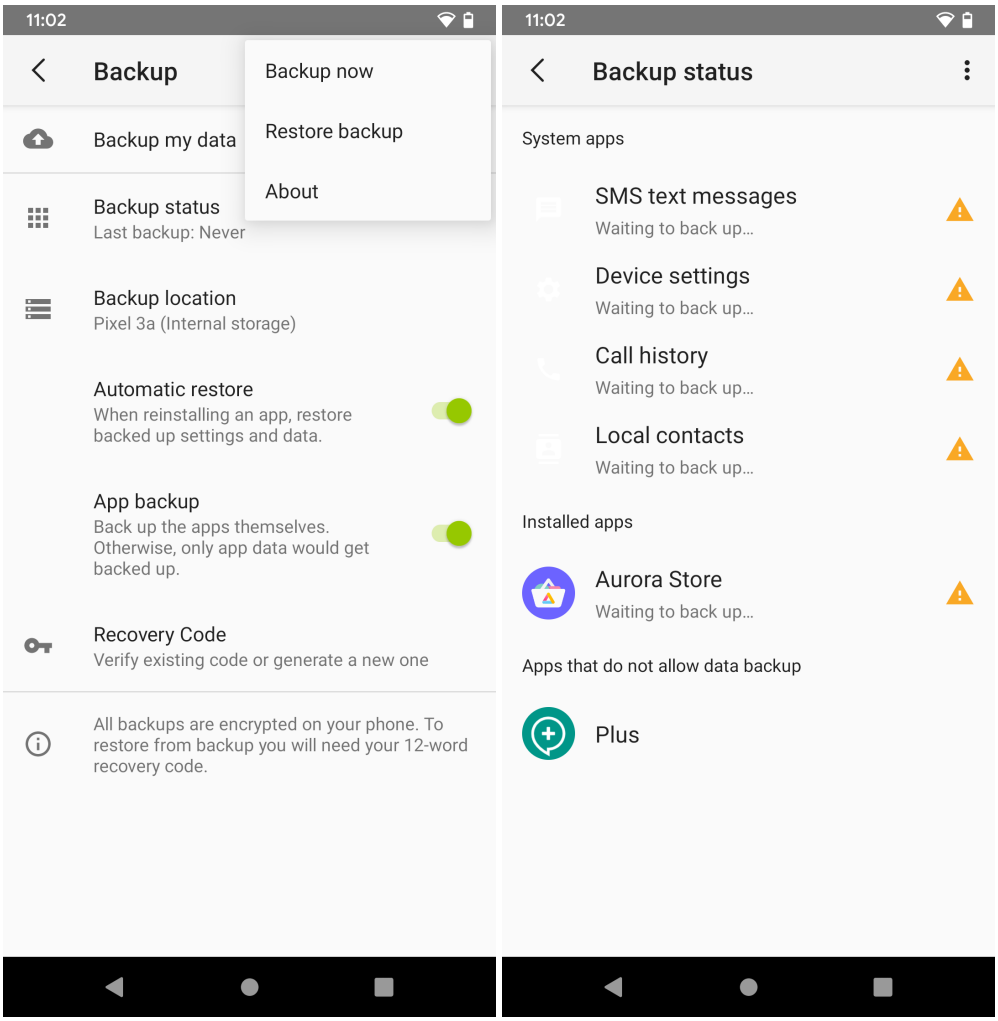
(a) (b)

Figure 7.23: Data Backup 1



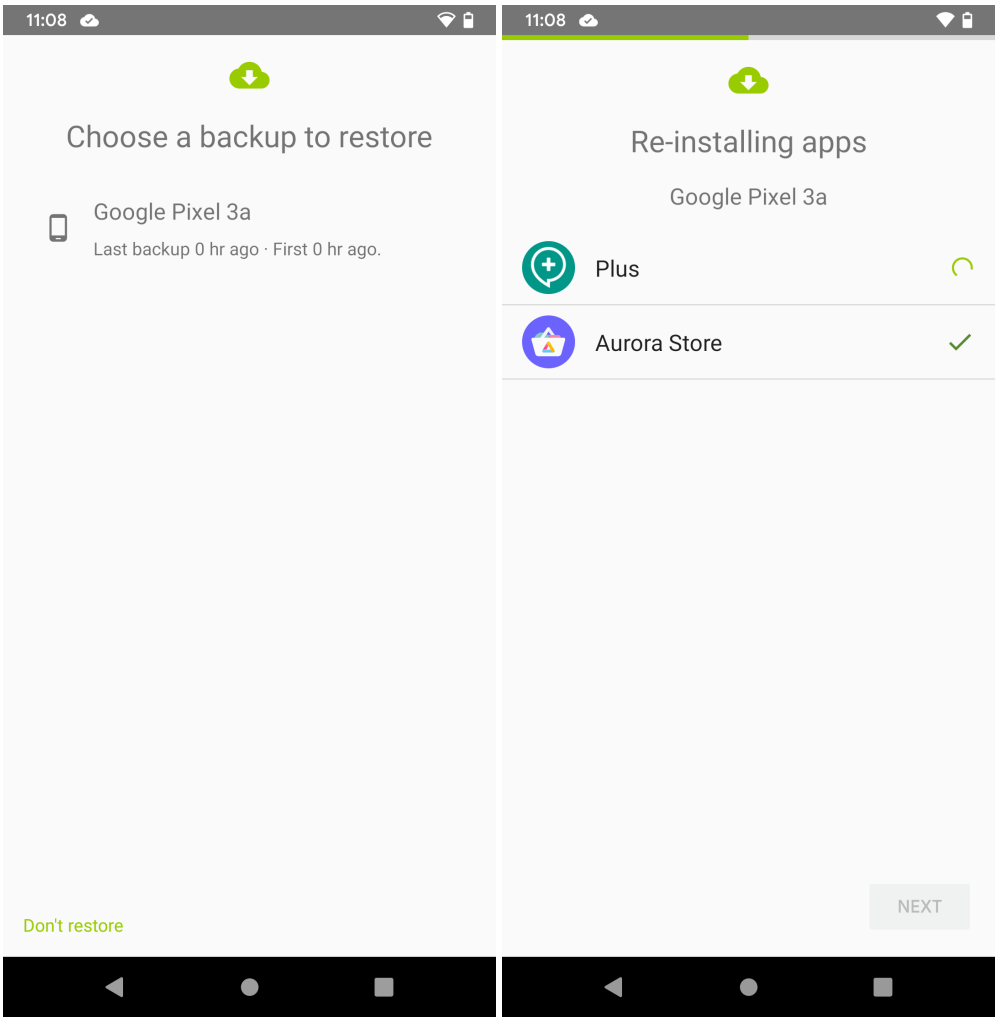
(a) (b)

Figure 7.24: Data Backup 2



(a) (b)

Figure 7.25: Data Backup 3

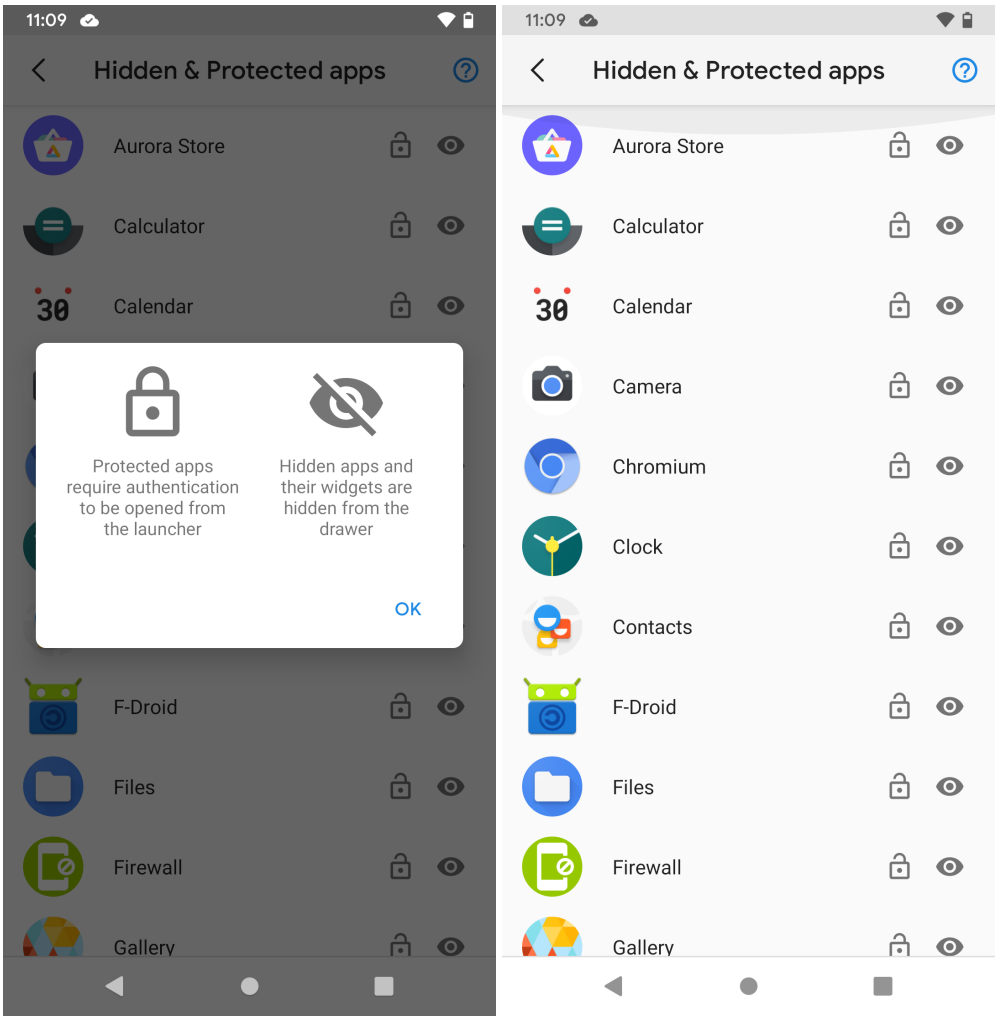


(a) (b)

Figure 7.26: Data Backup 4

7.2.16 Hidden & Protected apps

This feature to lock and hide apps in launcher. It uses screen lock password as protection, so no need to set separate password for this option.

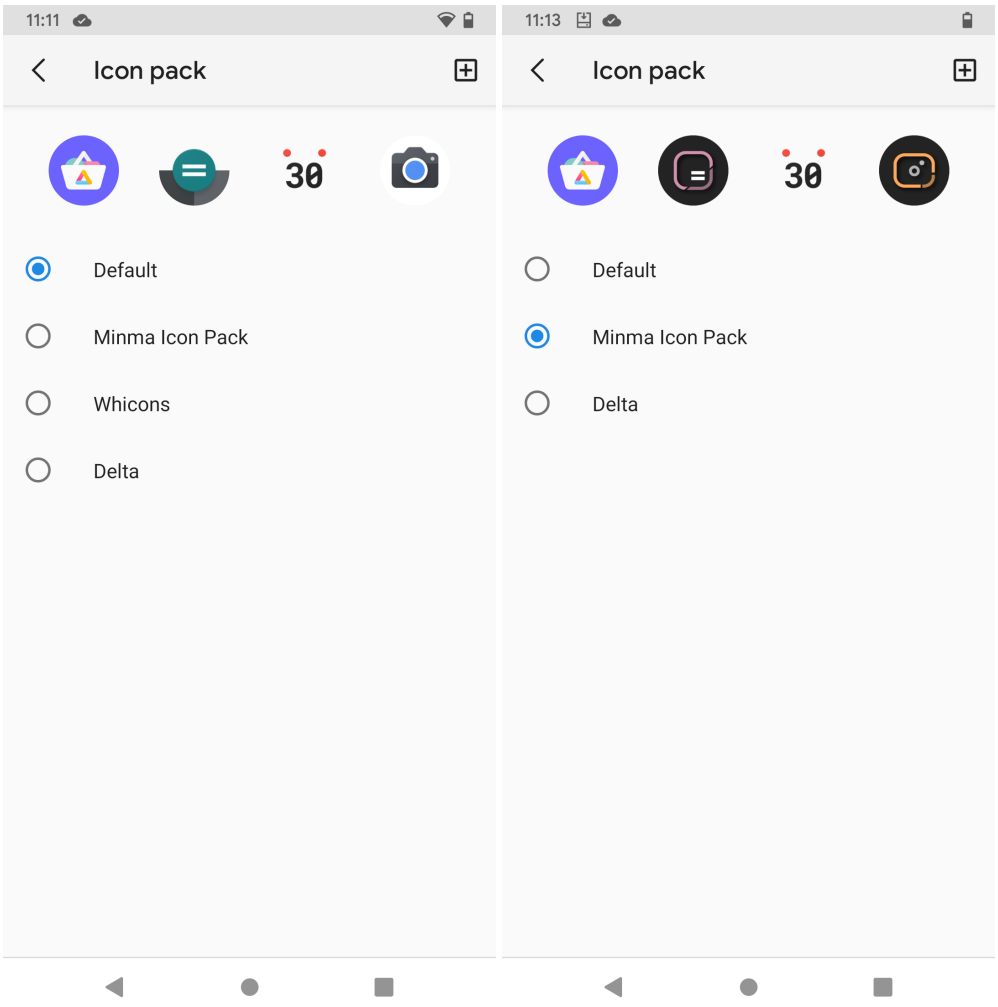


(a) (b)

Figure 7.27: Hidden & Protected apps

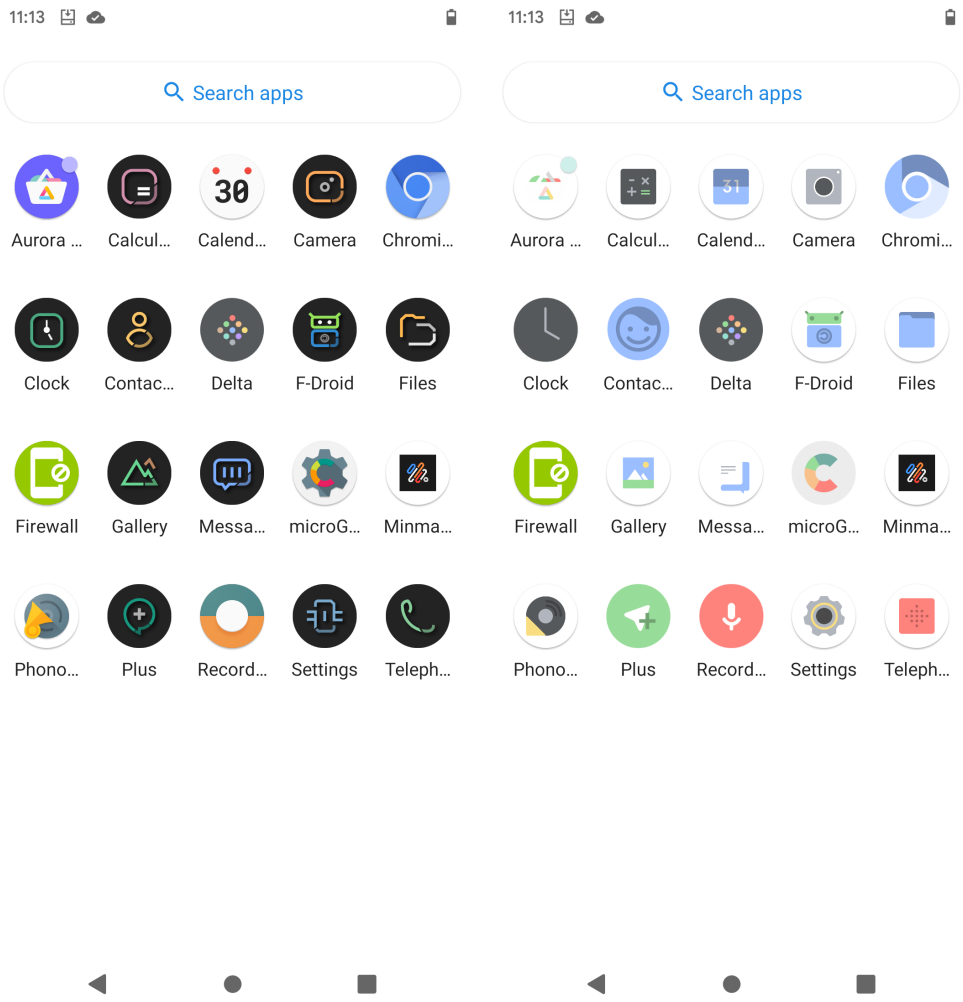
7.2.17 Icon packs

This feature allow users to install third party icons packs and customize their launcher icons with ease.



(a) (b)

Figure 7.28: Icon packs 1

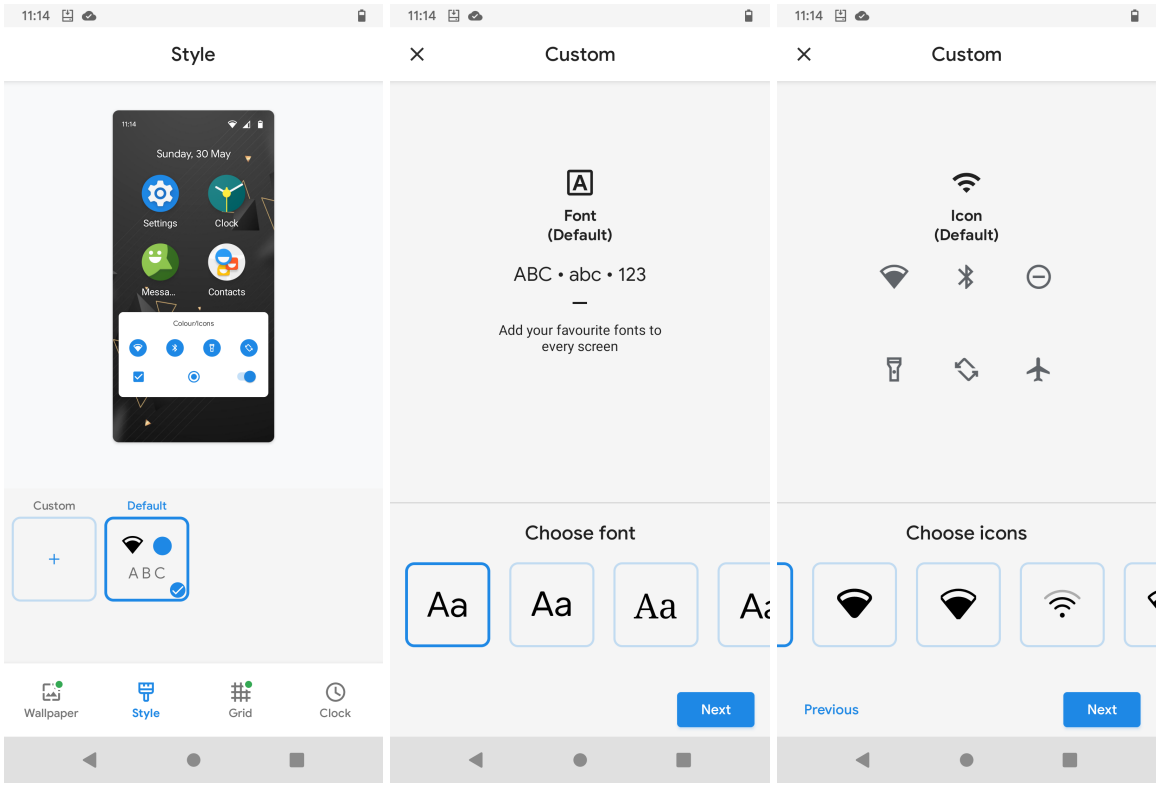


(a) (b)

Figure 7.29: Icon packs 2

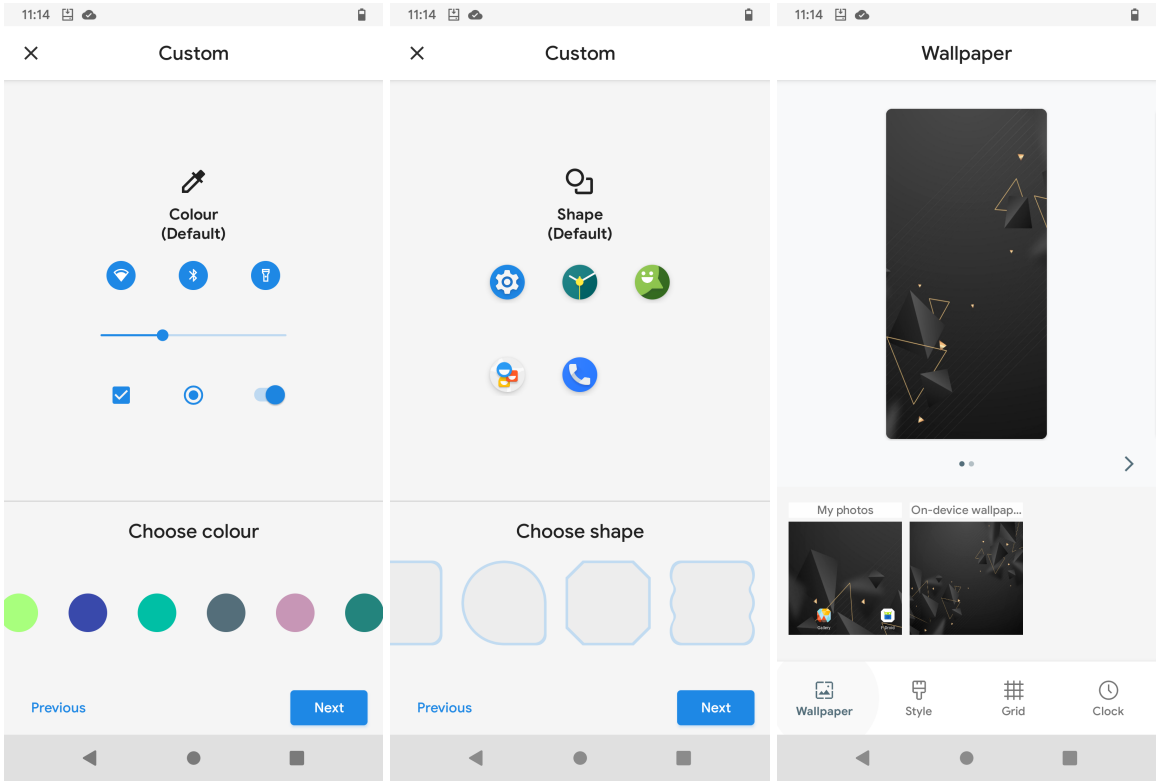
7.2.18 Theme phone

This apps is provided by google, but disabled by default for AOSP device. This app can change device's fonts, accent color, icon shapes, launcher icon shapes,etc. Also change clock faces, styles and wallpapers.



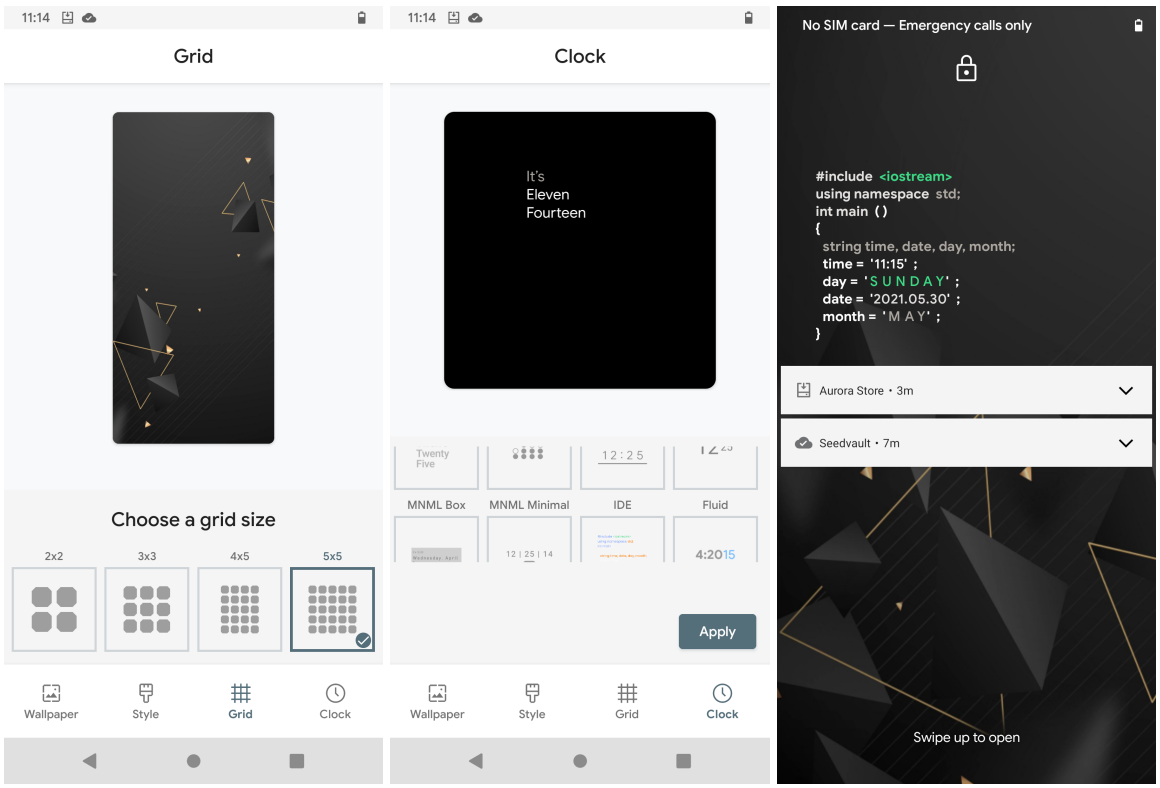
(a) (b) (c)

Figure 7.30: Theme phone 1



(a) (b) (c)

Figure 7.31: Theme phone 2

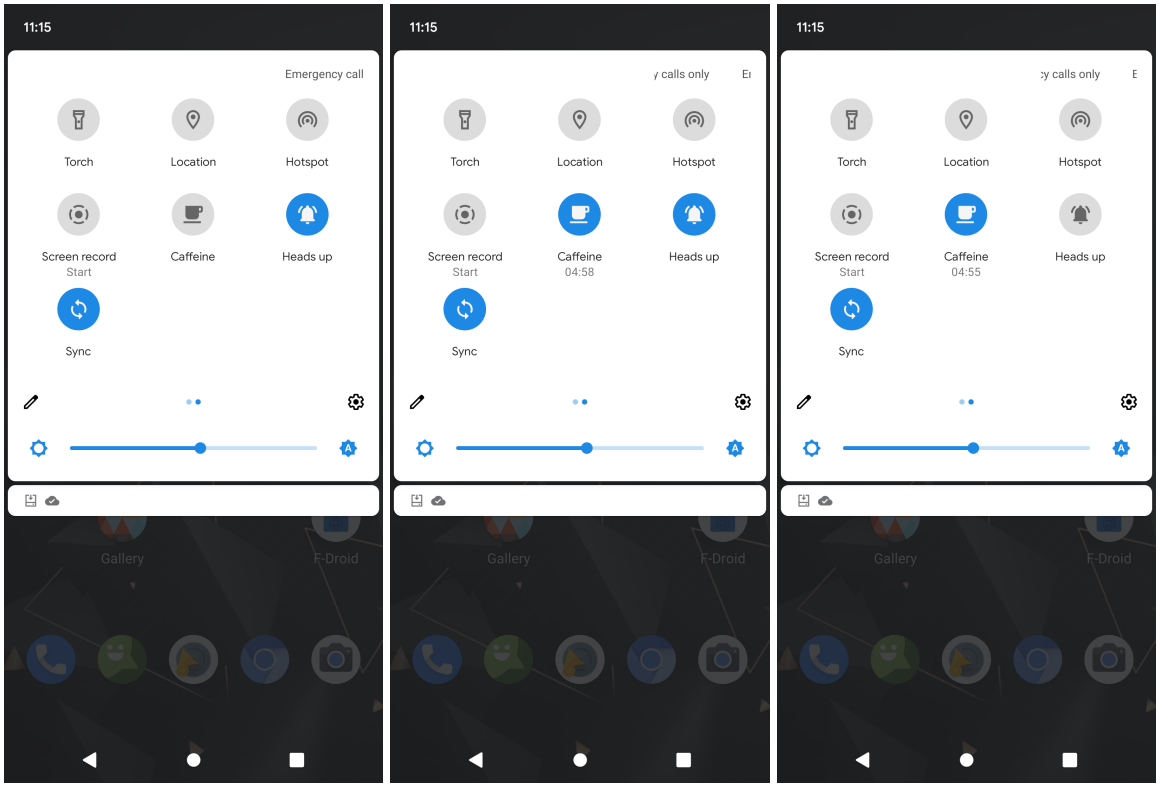


(a) (b) (c)

Figure 7.32: Theme phone 3

7.2.19 QS Tiles

Added three types of QS tiles which doesn't exist in AOSP. First one caffeine allows to temporary set screen out time. And sync allows to sync data from internet connected apps to work properly. And heads up allows to disable notification from status bar to popup.

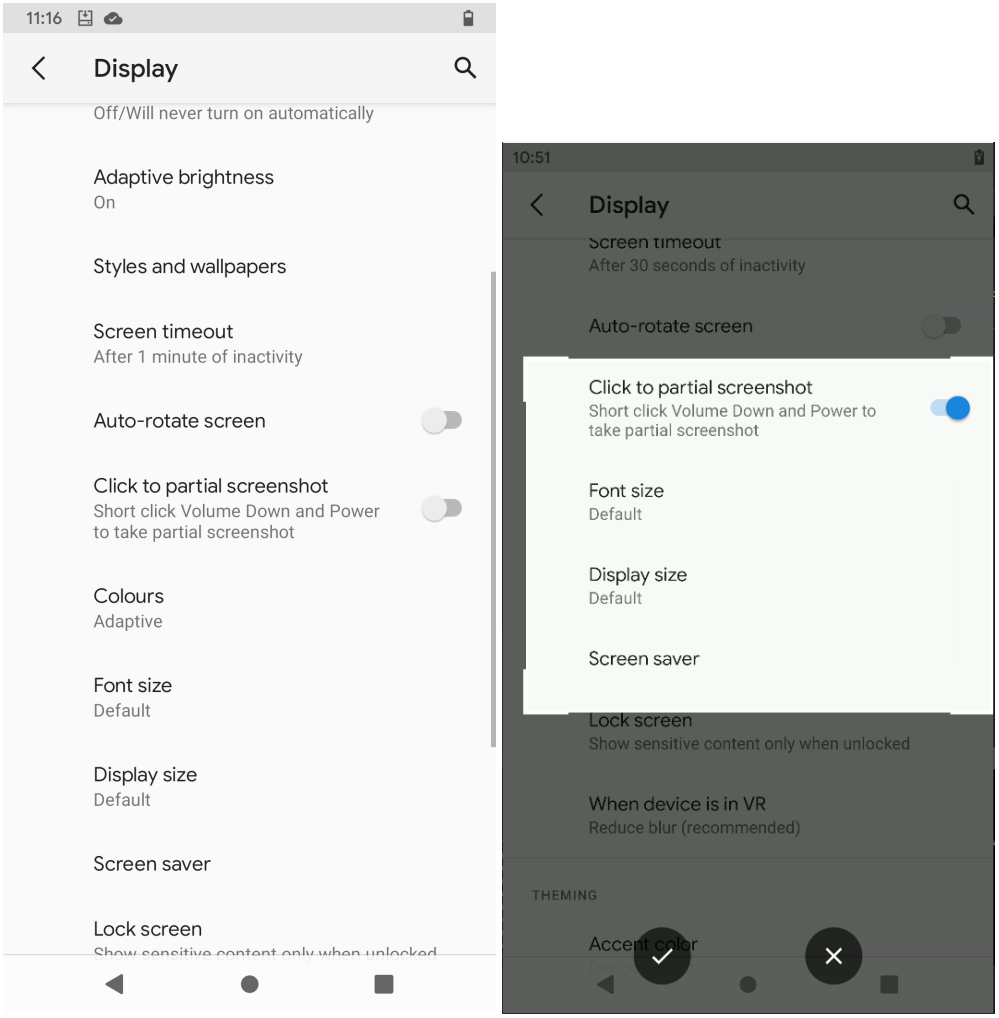


(a) (b) (c)

Figure 7.33: QS Tiles

7.2.20 Partial Screenshot

This feature will allow us to take partial screenshot of the screen by short click and long click for full screen shot with ease.

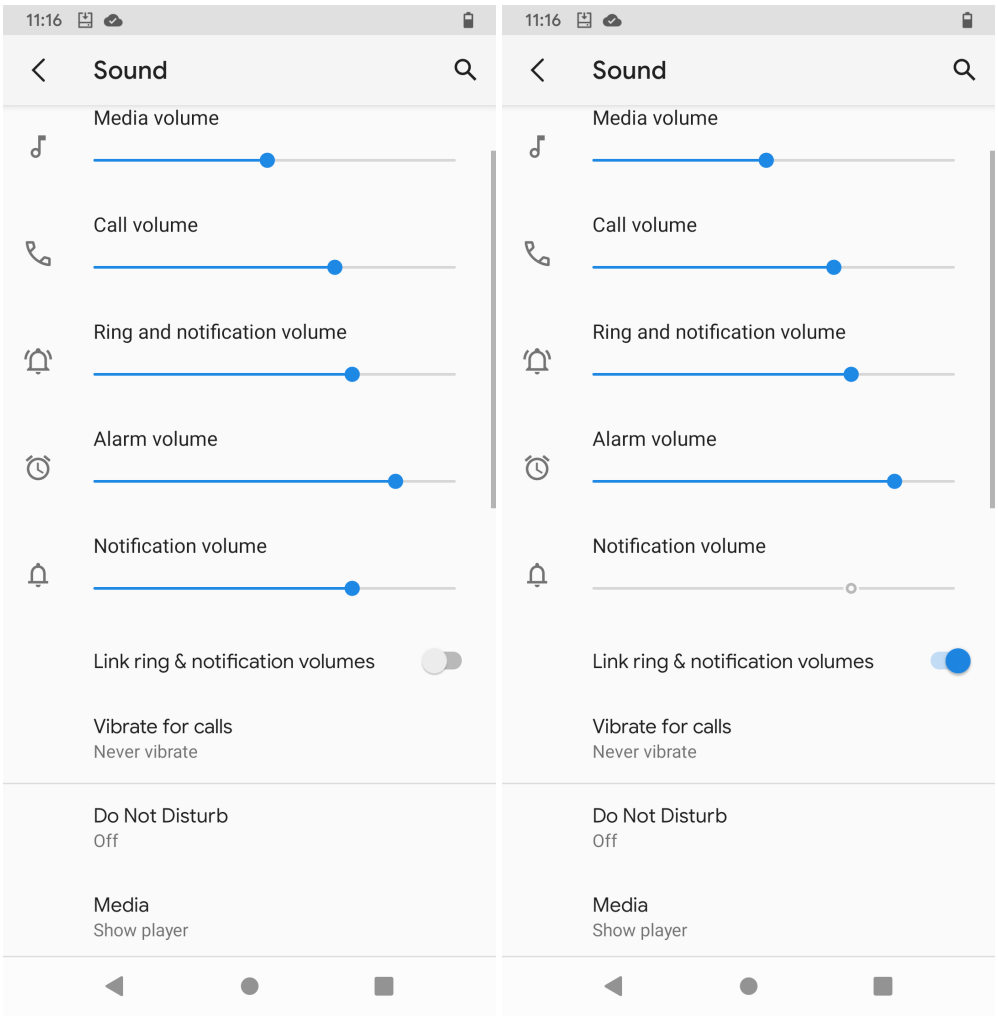


(a) (b)

Figure 7.34: Partial Screenshot

7.2.21 Link Ringer & Notification volume

This feature will allow us link ringer and notification volume. So we can control both with a single slider. No need to control separately. We can toggle it on or off as per user needs.

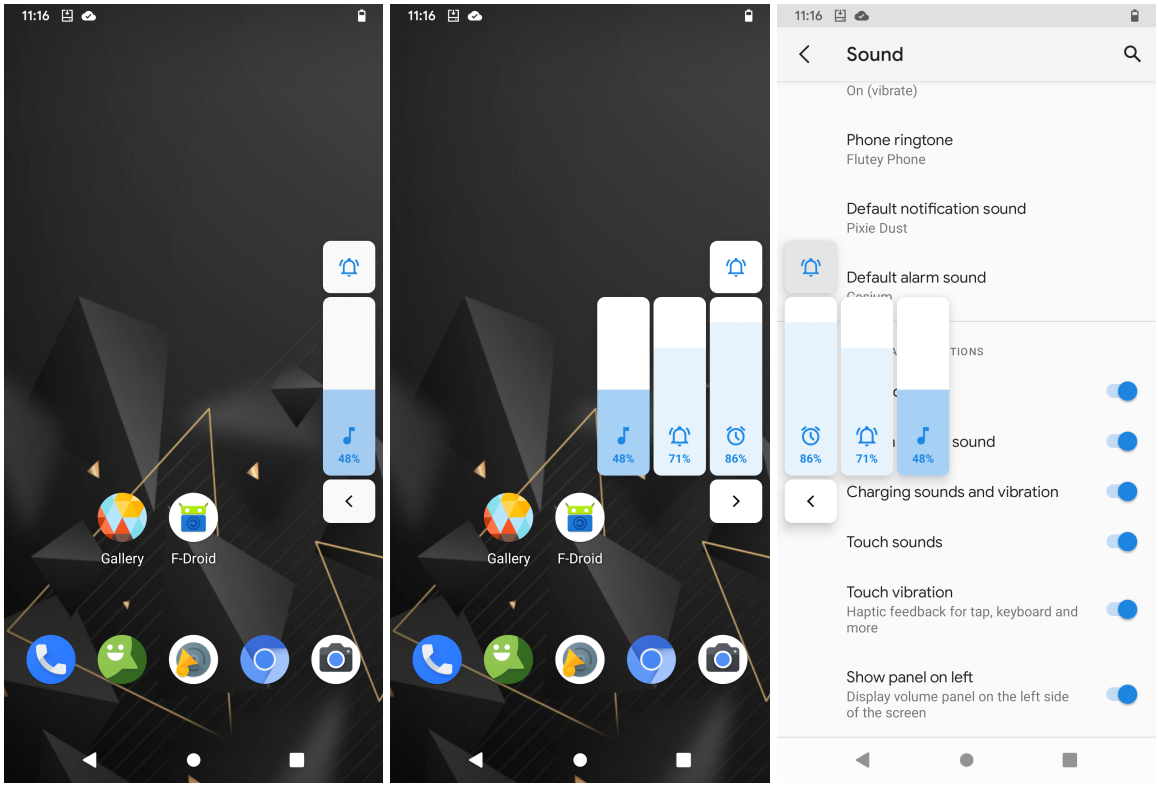


(a) (b)

Figure 7.35: Link Ringer & Notification volume

7.2.22 Volume Panel

This feature allows users to control the volume of alarm, media and ringer volume with single panel. This was used before android-9.0 and discontinued in post android versions. We can also change location of volume panel.

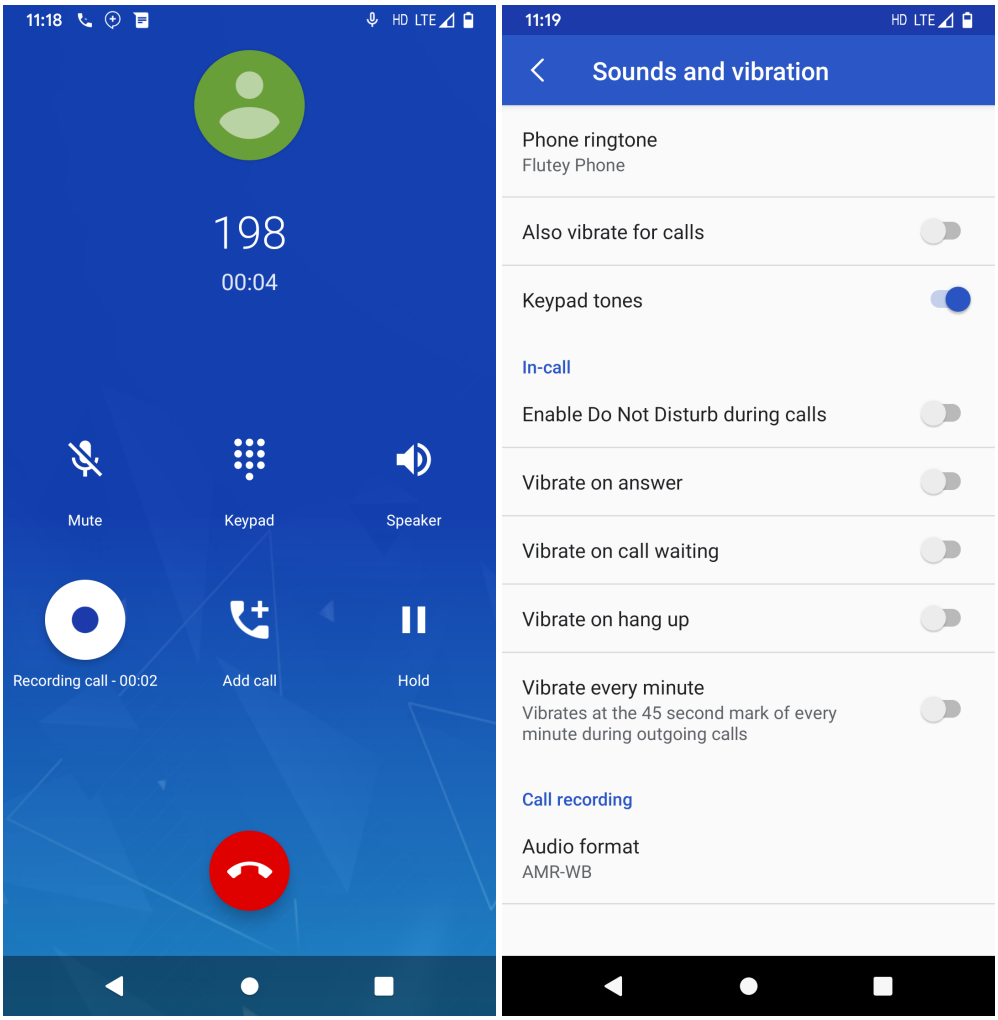


(a) (b) (c)

Figure 7.36: Volume Panel

7.2.23 Record calls

This feature allows to record all native voice calls and save them to the phone storage. Including the quality of the calls and record type can be adjusted.

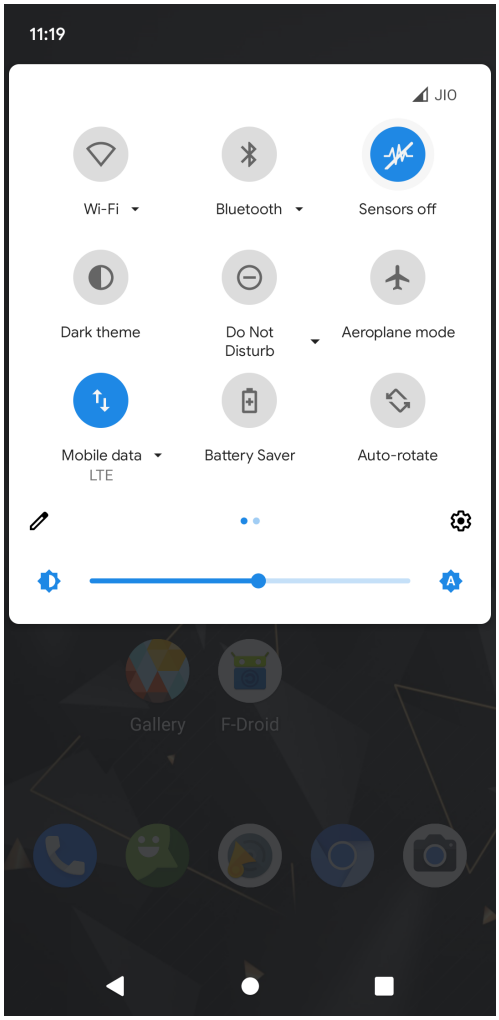


(a) (b)

Figure 7.37: Record calls

7.2.24 Disable sensors

This tile will allow users to disable sensors on the phone. Mostly it disables cameras and microphones. This will help users to be not monitored by any means or application in background.

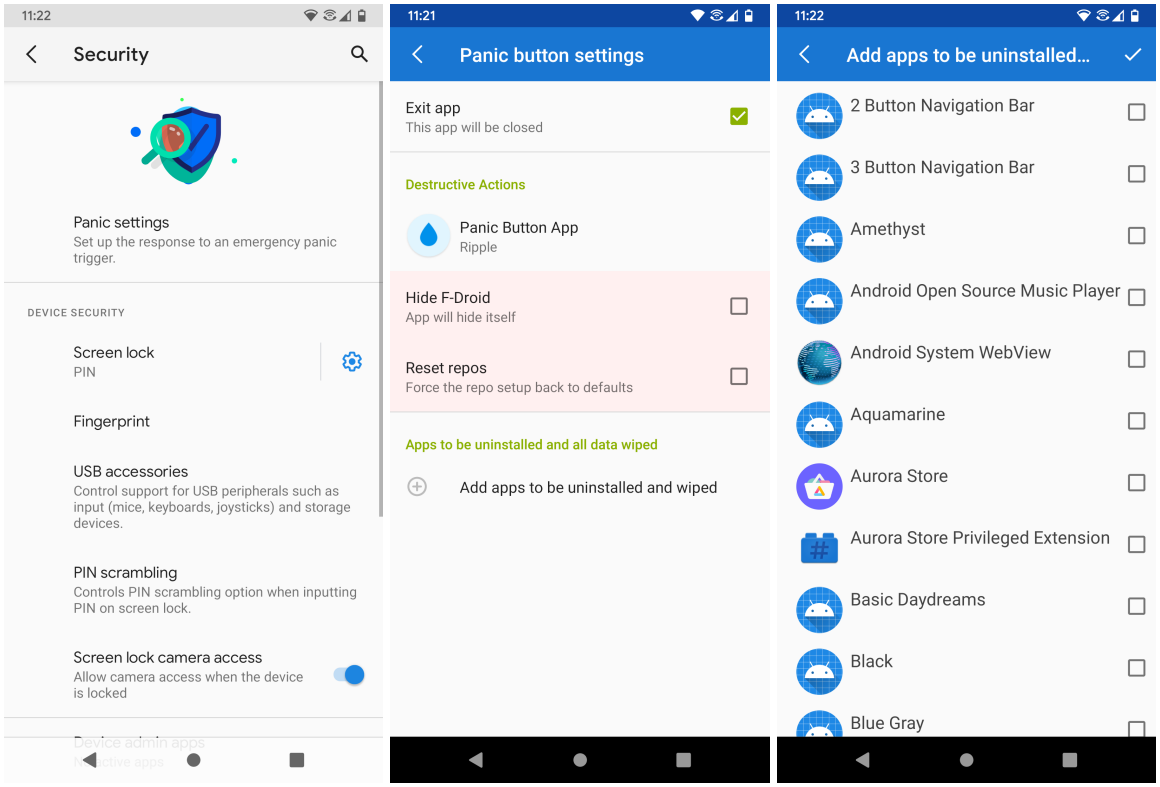


(a)

Figure 7.38: Disable sensors

7.2.25 Panic trigger

”Panic button” that can send it’s trigger message to any app that is a ”panic responder”. Such apps can do things like lock, disguise themselves, delete private data, send an emergency message, and more.

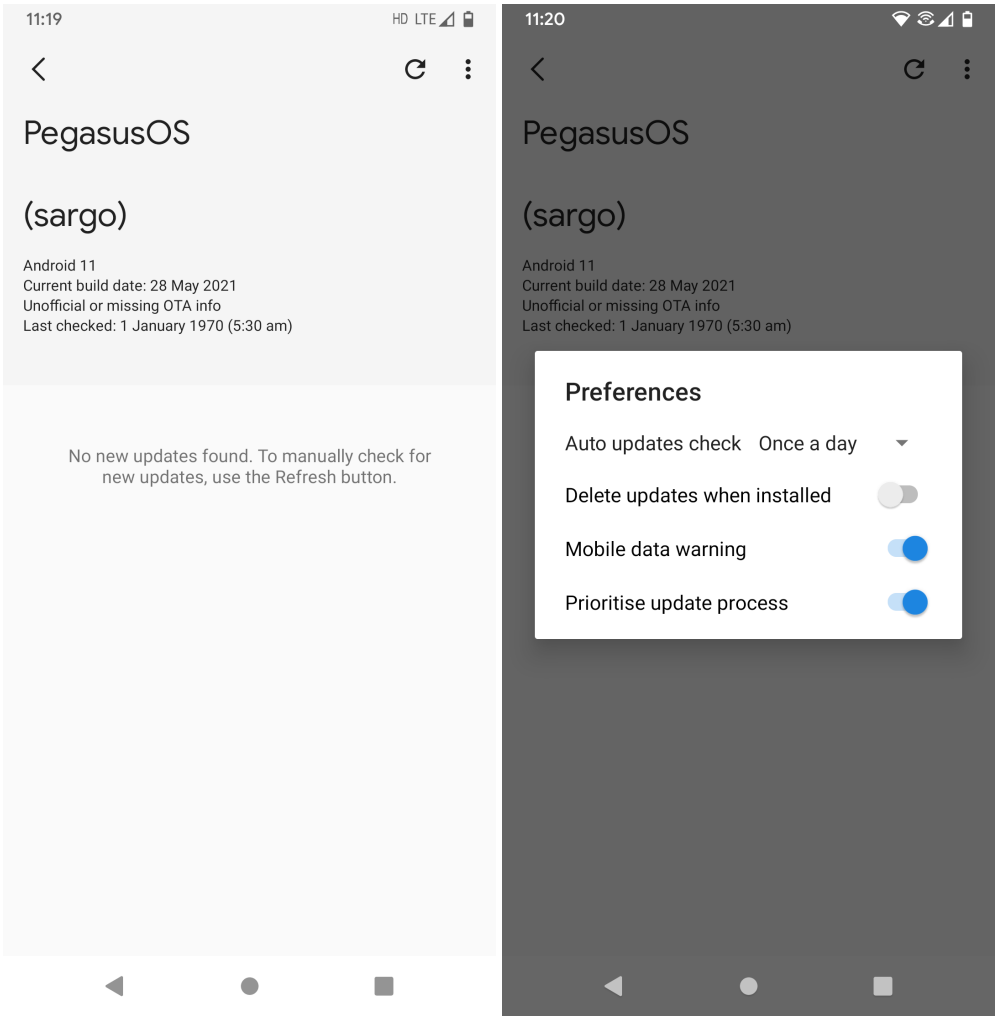


(a) (b) (c)

Figure 7.39: Panic trigger

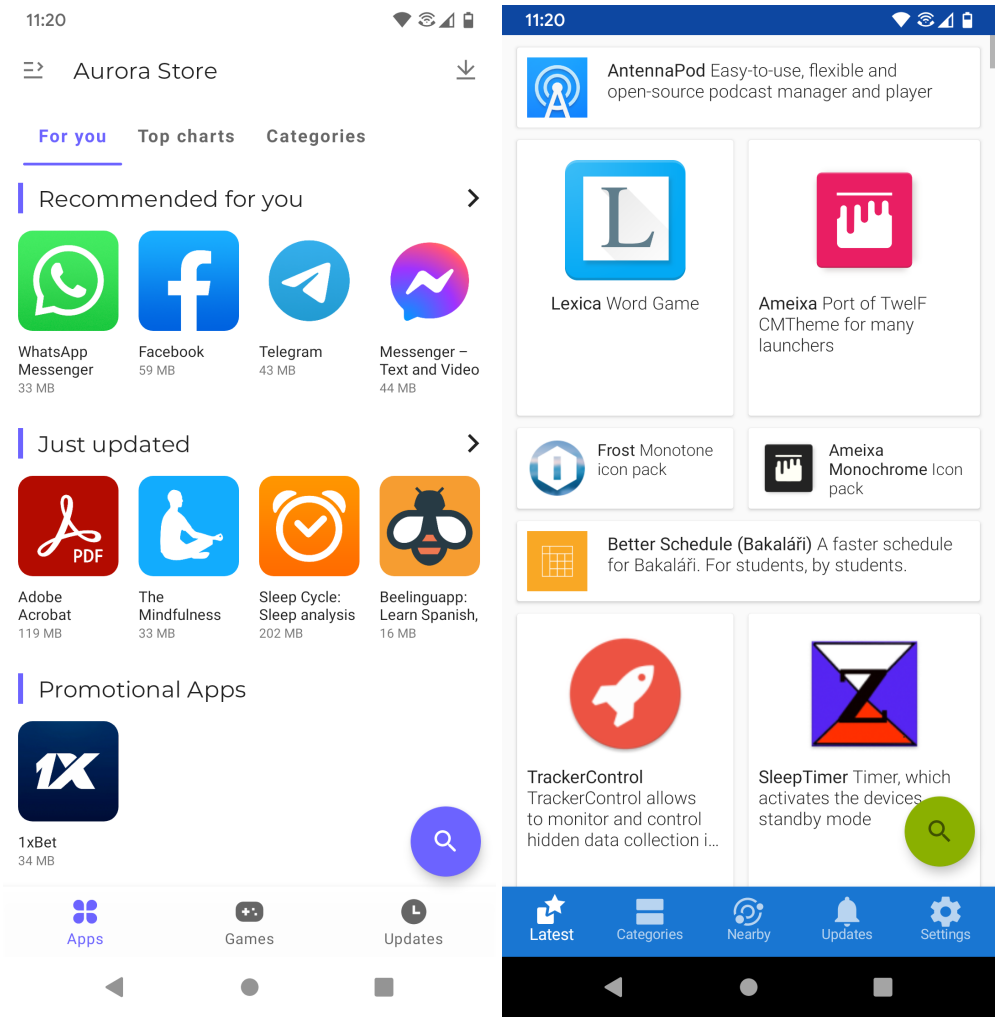
7.2.26 Provide updates

These apps will allow users to receive updates from the developer and update for apps. This can improve the device security and future bug fixes will be given through this by the developer.



(a) (b)

Figure 7.40: Provide updates 1



(a) (b)

Figure 7.41: Provide updates 2

7.3 Git history

Source code: https://github.com/PegasusOS

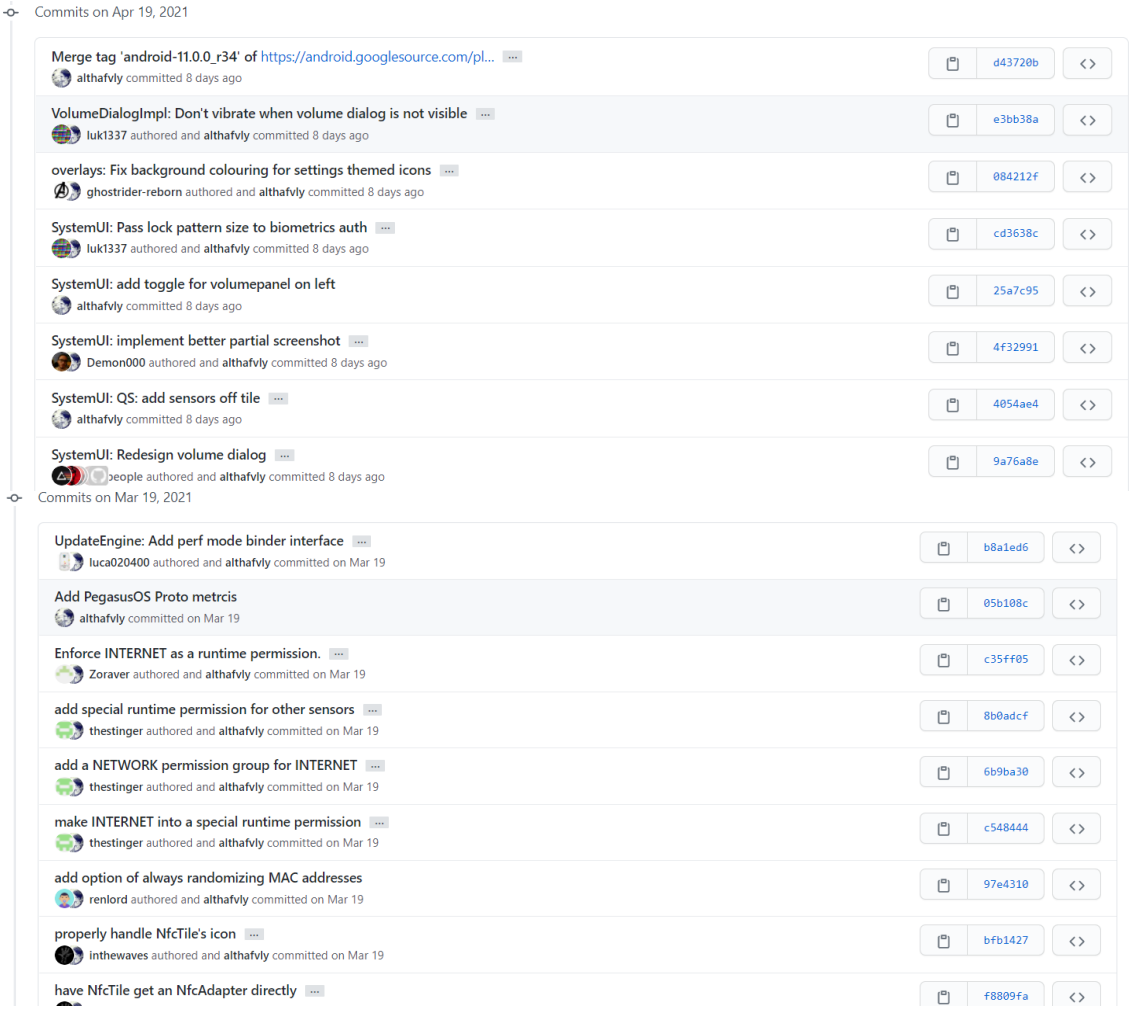


Figure 7.42: Git history 1

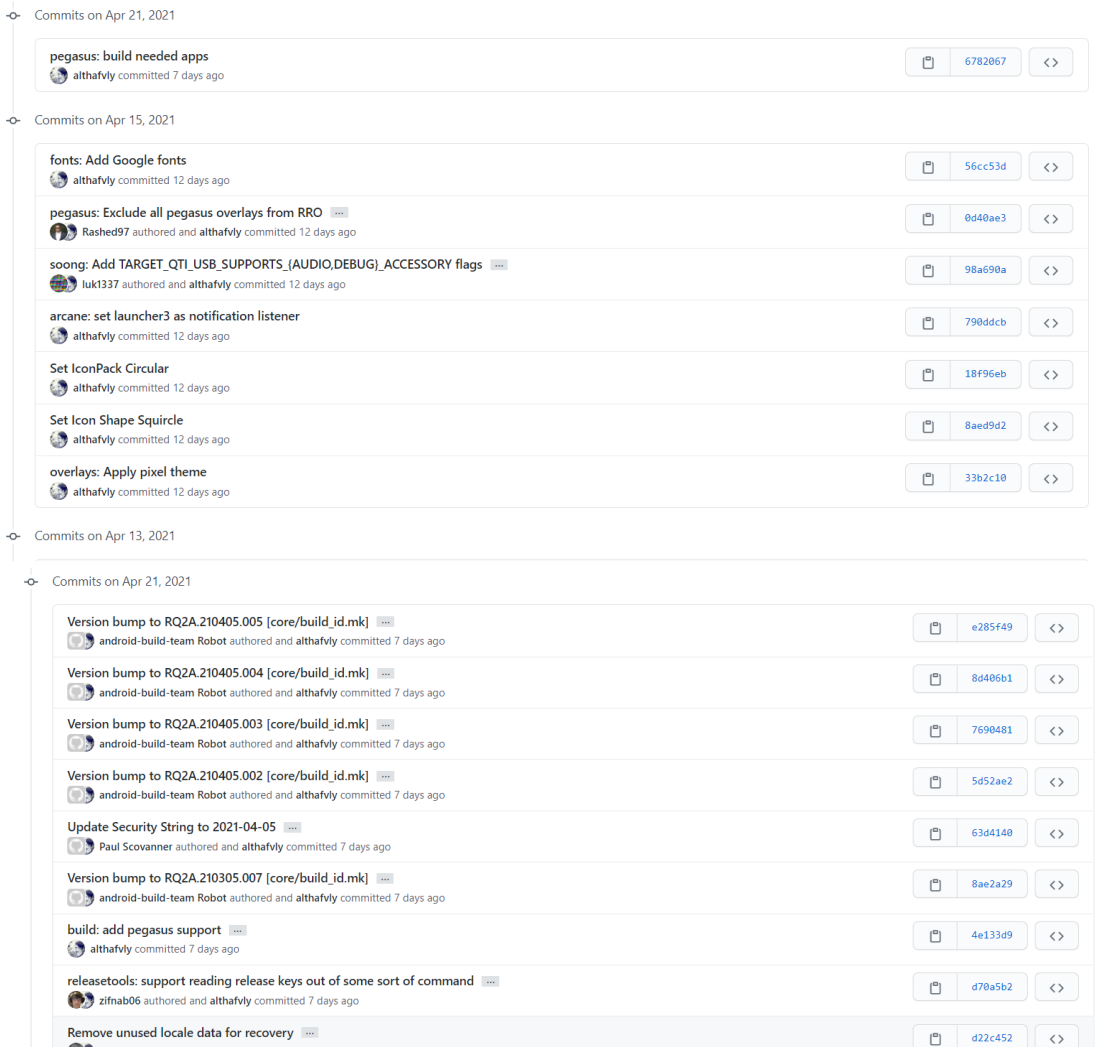


Figure 7.43: Git history 2

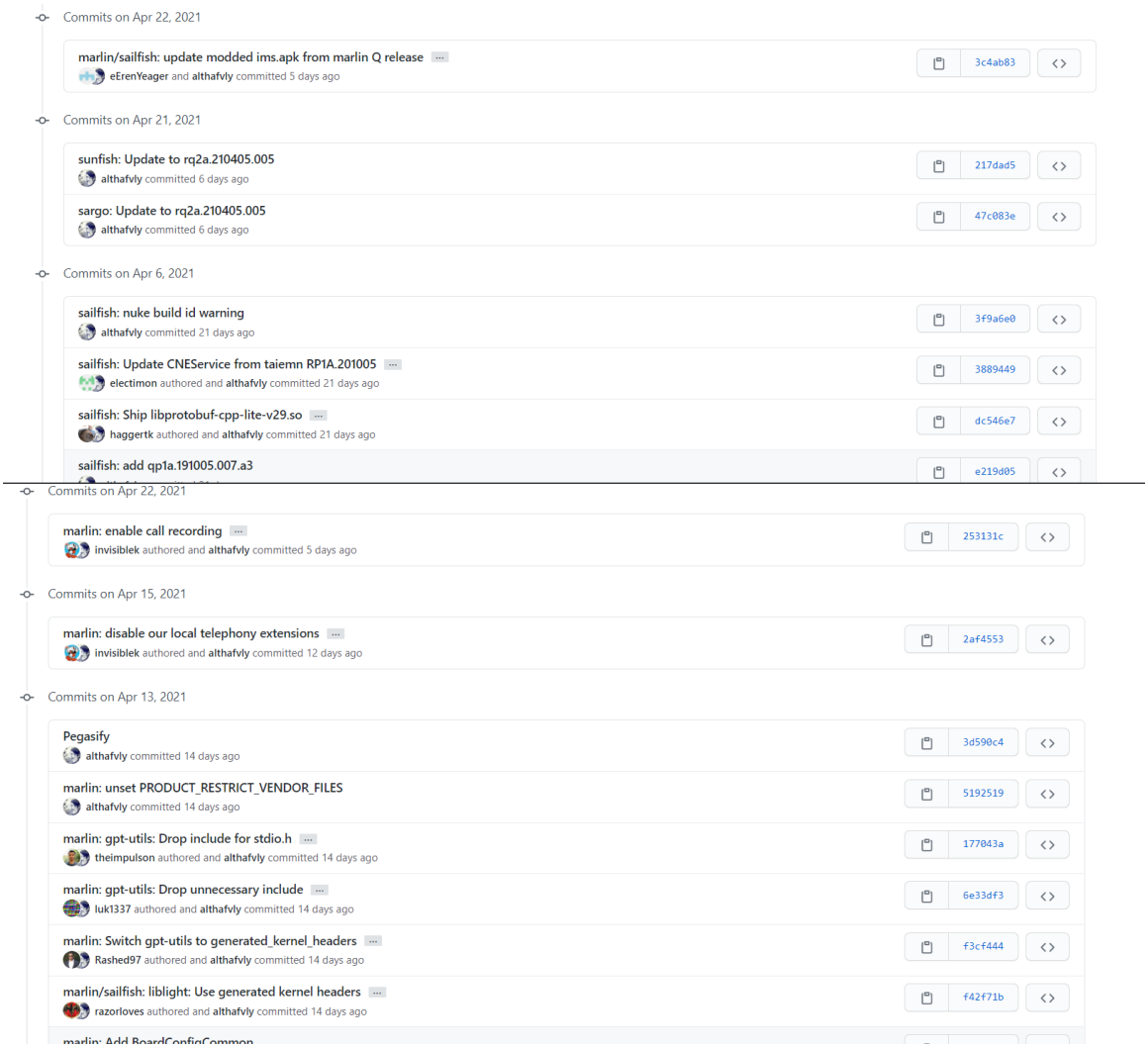


Figure 7.44: Git history 3

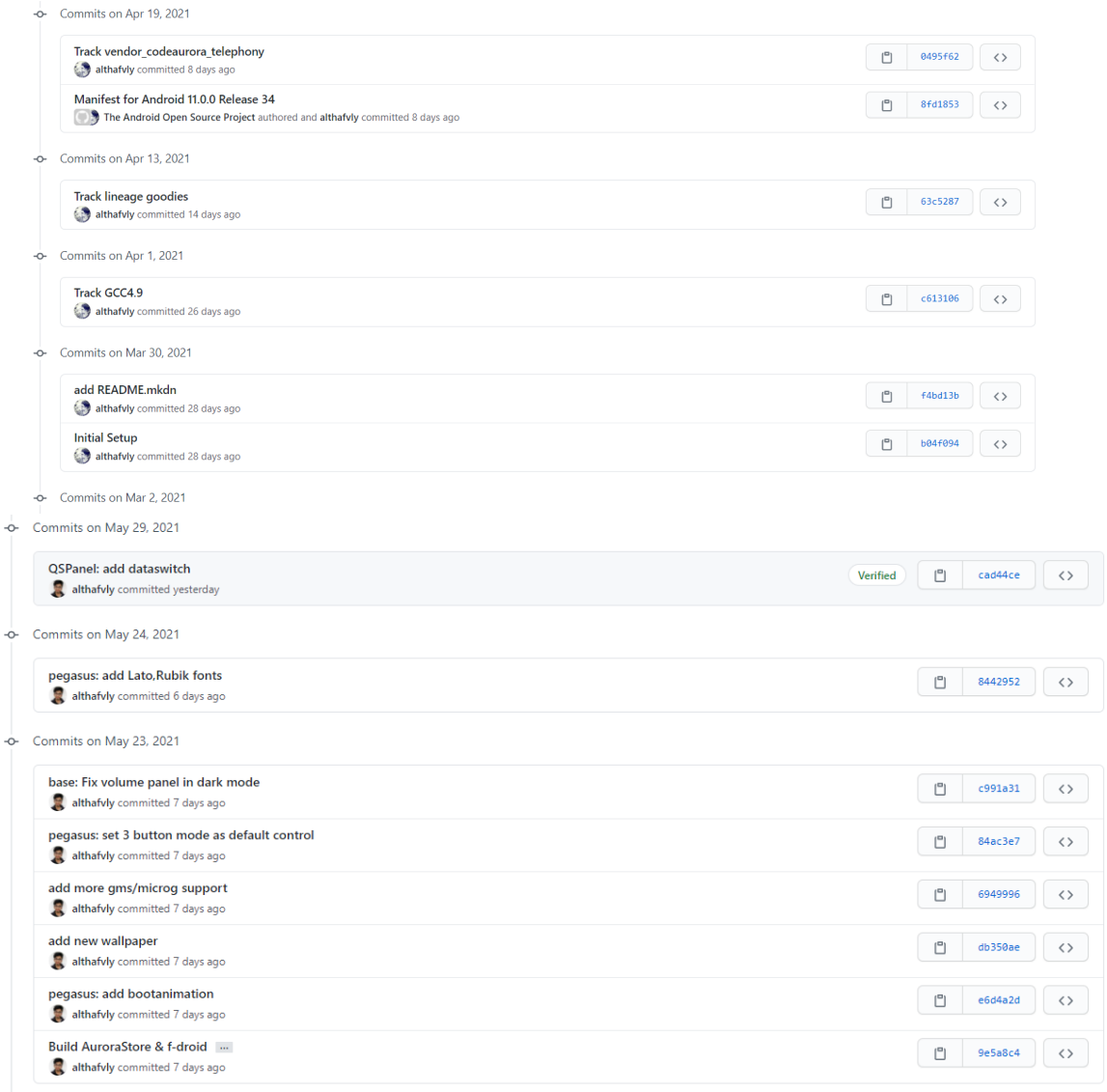


Figure 7.45: Git history 4

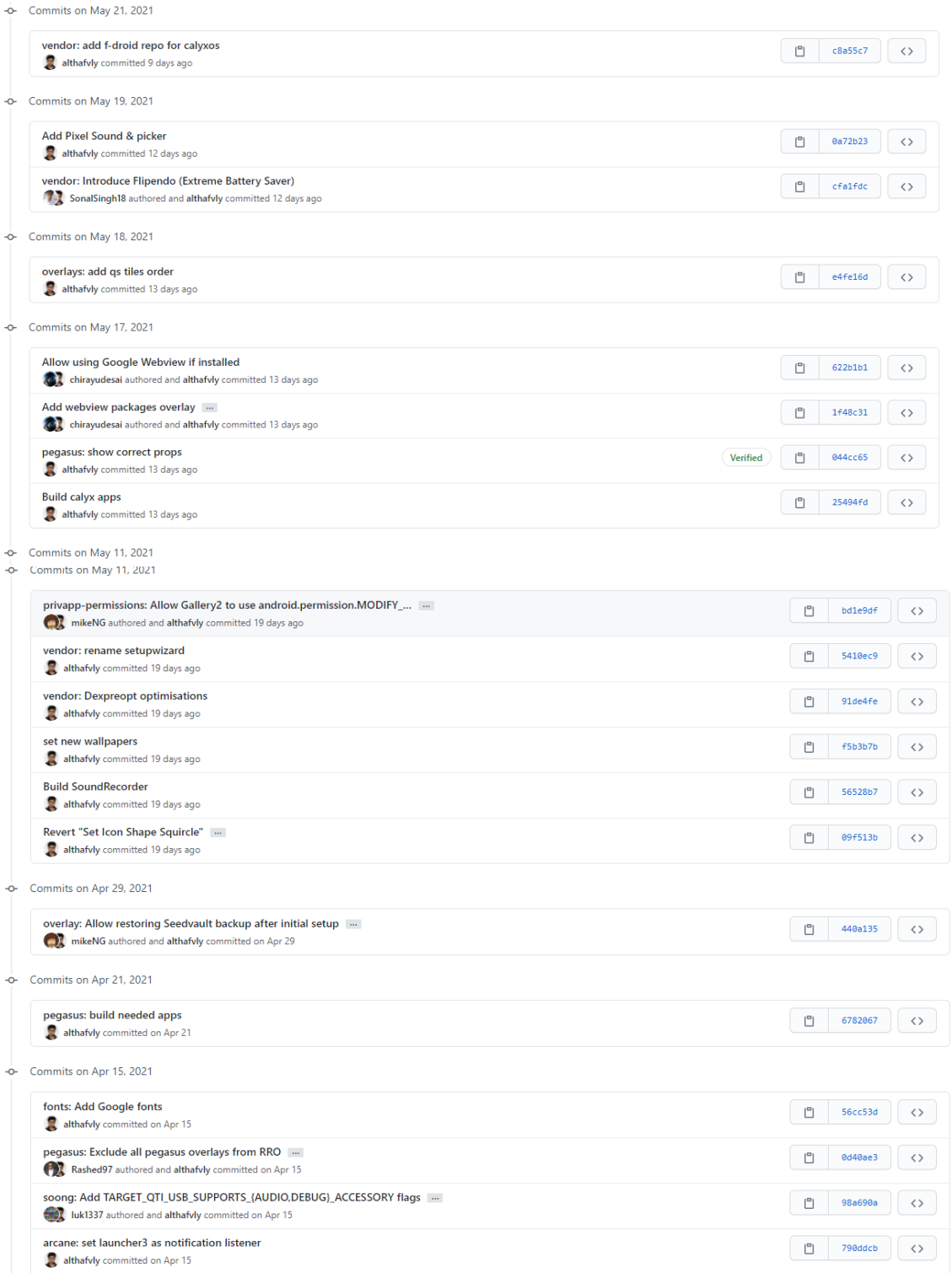


Figure 7.46: Git history 5

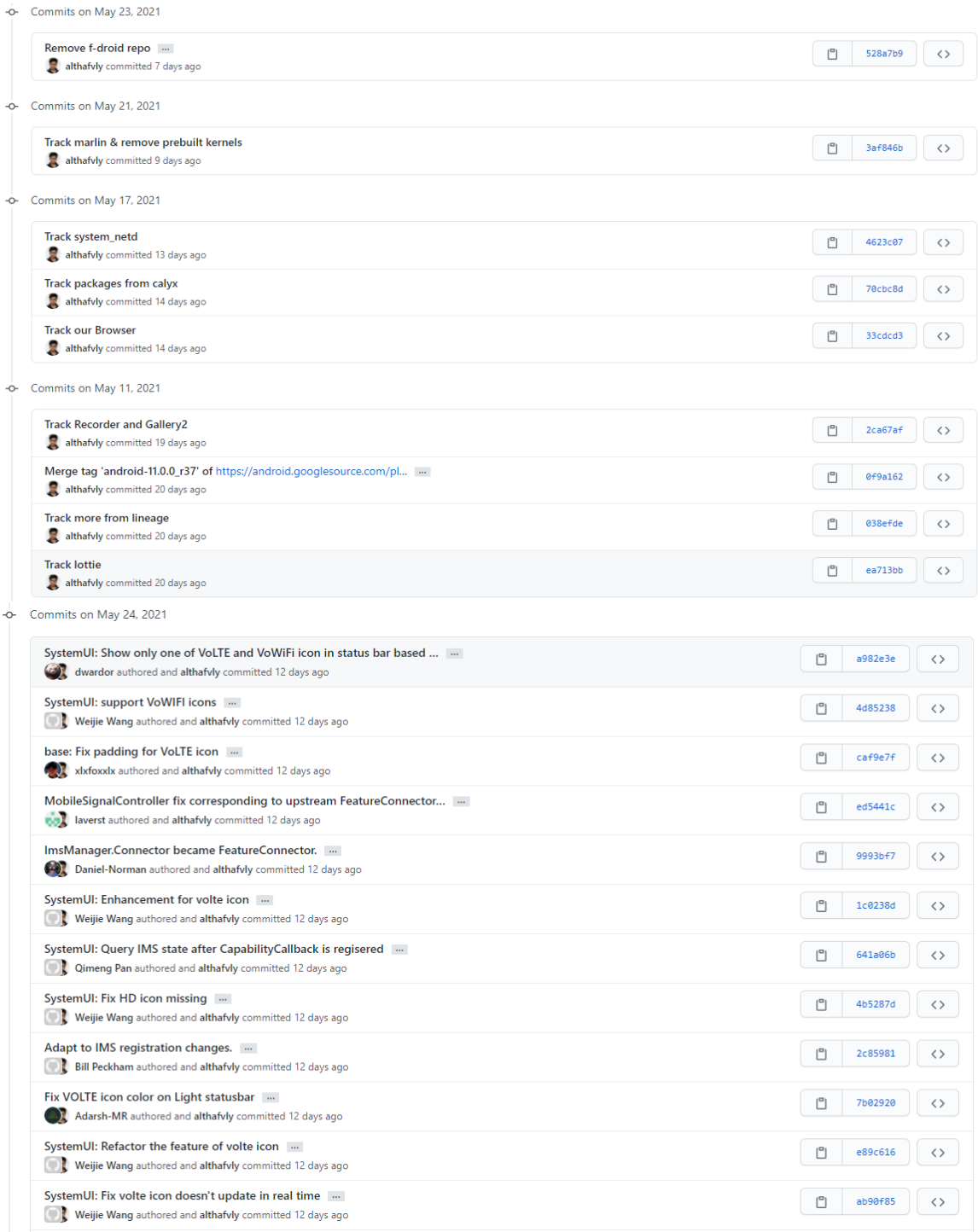


Figure 7.47: Git history 6

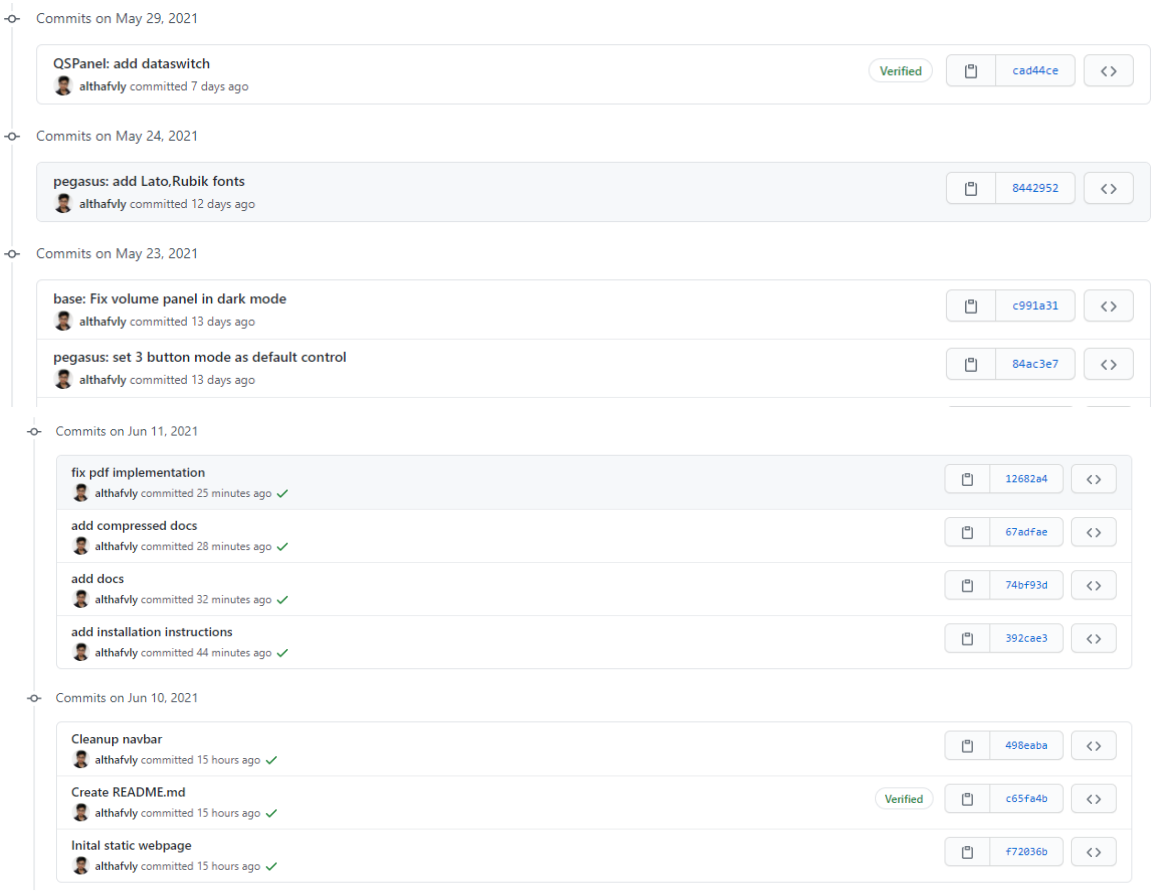


Figure 7.48: Git history 7