Software Design Document

# BluT Scan

**Version 1.2**

**Prepared by**

**Akshay A P – 6**

**Mohammed M V – 44**

**Rahees Ahammed V P – 53**

**Sreerag P S – 64**

**TKM College of Engineering**

**5 April 2023**

# Table of Contents

# 1. Introduction

This Software Design Specification document outlines the design and development of a Bluetooth scanner application for Android devices. The application will scan for nearby Bluetooth devices and display their timestamp, RSSI (Received Signal Strength Indicator), and MAC address. The application will also run in the background and scan every minute to continuously update the list of nearby devices.

## 1.1 Scope

The scope of this software design document is to provide a comprehensive and detailed overview of the design of the Bluetooth scanner application for smartphone, including its architecture, user interface design, component design, algorithm design, data design, security design, testing approach, maintenance design, and deployment design. This document will serve as a reference for ensuring that the application is designed and implemented in a consistent and high-quality manner. The document will cover the functional and non-functional requirements of the application, as well as the design decisions and trade-offs that were made during the development process.

## 1.2 Purpose

The purpose of this Software Design Specification (SDS) document is to provide a detailed description of the design and architecture of the Bluetooth scanner application developed for the Android platform. The document serves as a guide for developers, testers, and stakeholders involved in the development process, outlining the application's features, requirements, user interface design, performance, testing approach, deployment strategy, and maintenance plan.

## 1.3 Intended Audience

The intended audience for this Software Design Specification (SDS) document includes developers, testers, project managers, and stakeholders involved in the development process of the Bluetooth scanner application for the Android platform. The document provides a comprehensive guide to the design and architecture of the application, outlining its features, requirements, testing approach, and maintenance plan, which can be used by the development team as a reference throughout the project's lifecycle.

## 1.4 References

Android Developer Documentation: Bluetooth and BLE Scanning - Official documentation from Android Developer website: https://developer.android.com/guide/topics/connectivity/bluetooth

Bluetooth Core Specification - Official specification document from Bluetooth SIG: https://www.bluetooth.com/specifications/bluetooth-core-specification/

Android Bluetooth API Guide - Guide on Bluetooth API in Android: https://developer.android.com/guide/topics/connectivity/bluetooth

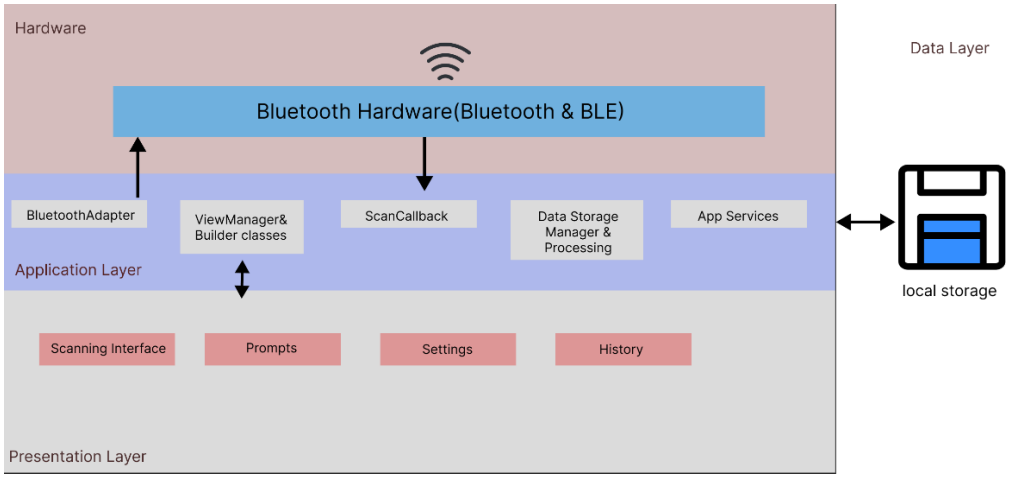
figma – for the ui design:https://www.figma.com/

# 2. System Architecture Design

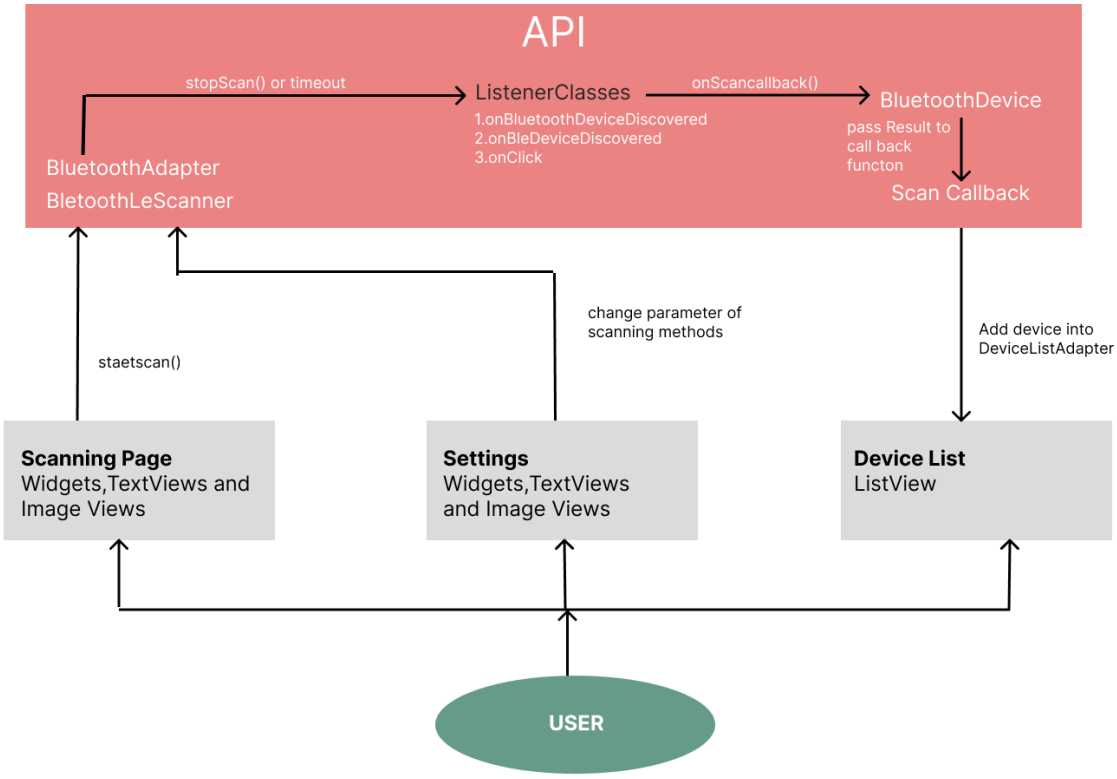
## 2.1 Description

1. Presentation Layer:
    - User interface to display the scanning status and detected devices
    - User input to configure the scanning interval
    - User feedback through notifications and alerts
2. Application Layer:
    - Controllers to manage the scanning process and device discovery
    - Services to handle background scanning using both BLE and classical Bluetooth protocols
    - Logic to process and filter device information
    - Data flow coordination between the scanning process and the data layer
3. Data Layer:
    - API to store and retrieve device information
    - Storage system to manage device data
    - Encryption and access control mechanisms to ensure data security

## 2.2 Architecture




# 3. Application Architecture Design




The architecture of the Bluetooth scanner application can be divided into four main components:

1. User Interface Layer:

The User Interface Layer is responsible for presenting the application's user interface to the user. It should contain all of the UI components such as buttons, text fields, and list views. In

the Bluetooth scanner application, the UI layer should display a list of nearby Bluetooth devices and allow the user to interact with the list by selecting a device to view its details.

## 2. Business Logic Layer:

The Business Logic Layer is responsible for handling the application's core functionality, such as scanning for Bluetooth devices, filtering the results, and updating the UI layer with the results. In the Bluetooth scanner application, the Business Logic Layer should include the Bluetooth Scanning Algorithm, Filtering Algorithm, and User Interface Algorithm.

## 3. Data Access Layer:

The Data Access Layer is responsible for accessing and storing data within the application. In the Bluetooth scanner application, there is no need for a separate Data Access Layer since the data is retrieved and displayed in real-time from the Bluetooth adapter.

## 4. Background Service Layer:

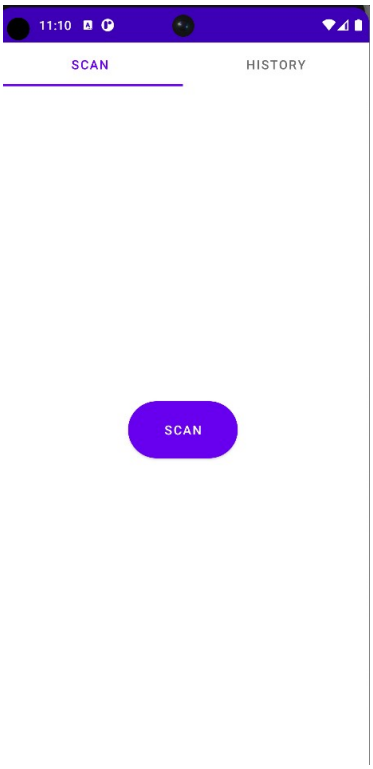
The Background Service Layer is responsible for running the Bluetooth Scanning Algorithm in the background at specified intervals, even when the application is not in the foreground. This layer should use the Android AlarmManager component to schedule the scan at the specified interval and should run as a service to ensure that it continues to run even if the application is closed.

# 4. GUI Design

## 4.1 User Interface Design

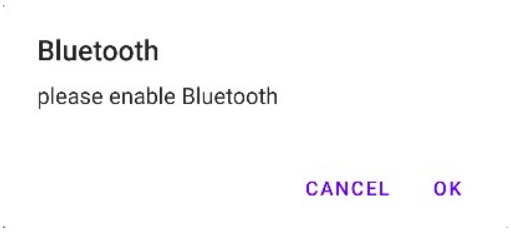
The user interface of the Bluetooth scanner application will consist of two main screens - the scan screen and the device list screen.

**Scan Screen:**



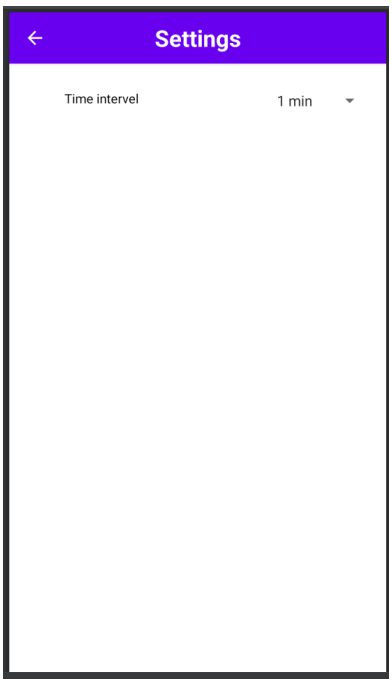

The scan screen will be the default screen of the application. It will have a simple and minimalistic design with a SCAN button in the center of the screen. The user can click on the SCAN button to initiate the scanning process.

## Permission Screen:

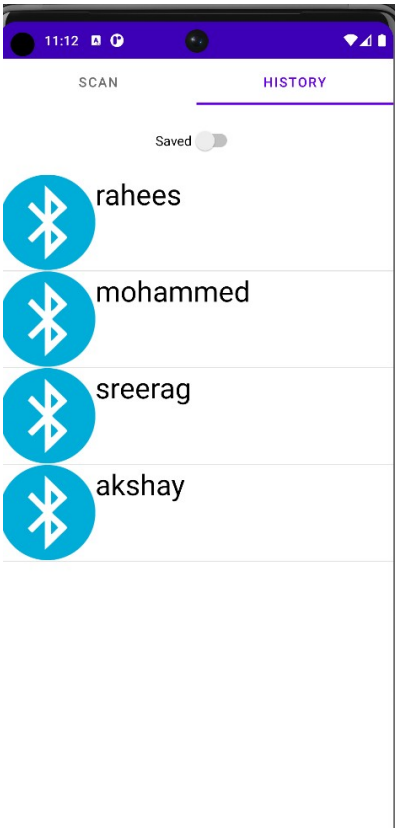



The app prompts the user to enable the Bluetooth in the device for scanning when he presses SCAN button.

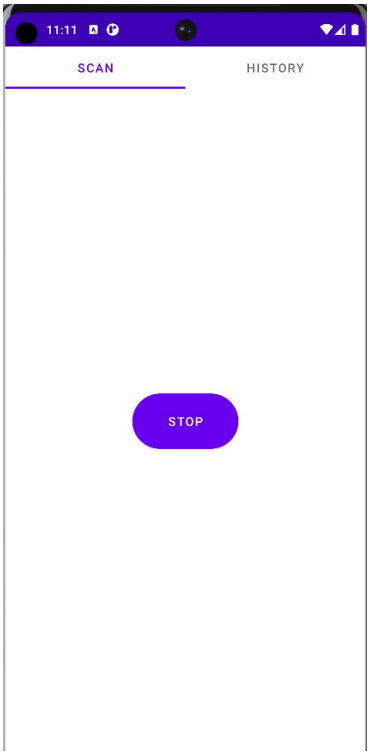
## Settings:



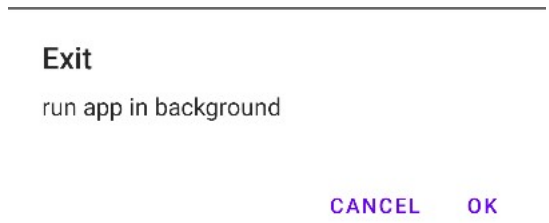

## Device List Screen:


11:12
SCAN
HISTORY
Saved
rahees
mohammed
sreerag
akshay

## Stop Scan:


11:11
SCAN
HISTORY
STOP

### Exit prompt:




Once the scanning process is initiated, the application will display the list of nearby Bluetooth devices on the device list screen. The device list screen will have a clean and simple design with a scrollable list of devices. Each device will be displayed in a card view, showing the device's name, MAC address, and RSSI value. The list will be sorted based on the device's RSSI value, with the device having the strongest signal displayed at the top. The user can click on each device card to expand it, showing additional details such as timestamp, device type, and device services.

## 4.2 Visual Design

The visual design of the Bluetooth scanner application should be clean, simple, and easy to navigate. It should focus on providing essential information to the user in a clear and organized way. The following design principles should be considered while designing the application:

### Color Scheme:

The color scheme of the application should be minimalistic and not distract the user from the content. The primary colors should be shades of blue or gray, which are commonly associated with Bluetooth and technology.

### Typography:

The font selection should be legible and easy to read. The font size should be appropriate, with important information displayed in a larger font size than other content.

### Iconography:

The use of icons can help to convey information quickly and efficiently. The icons should be simple, clear, and easily recognizable.

### Consistency:

The visual design of the application should be consistent throughout the application. The use of consistent design elements, such as font, color, and iconography, will help the user to navigate the application more easily.

Overall, the visual design of the Bluetooth scanner application should focus on providing a simple and intuitive user experience, making it easy for the user to navigate and understand the application's functionality.

## 4.3 Navigation Design

The application will have a simple and intuitive navigation system. The user can switch between the scan screen and the device list screen by swiping left or right on the screen, or through the navigation bar at the top. Additionally, the user can access the settings screen by clicking on the settings icon on the top right corner of the screen. The settings screen will allow the user to configure the scanning interval.

Overall, the user interface of the Bluetooth scanner application will be simple, intuitive, and easy to use. The application's design will focus on providing the user with essential information about nearby Bluetooth devices in a clean and organized way.

# 5. API Design

### i. Android API 27+

### ii. BluetoothScanManager Class:

This class provides methods to manage the Bluetooth and BLE scanning functionality. It could have the following methods:

**startBluetoothScan():** Starts scanning for nearby Bluetooth devices using the BluetoothAdapter and BluetoothLeScanner APIs. This method could take parameters such as scan duration, scan mode, and a callback interface for receiving scan results.

**stopBluetoothScan():** Stops an ongoing Bluetooth scan.

**startBleScan():** Starts scanning for nearby BLE devices using the BluetoothLeScanner API. This method could also take parameters such as scan duration, scan filters, and a callback interface for receiving scan results.

**stopBleScan():** Stops an ongoing BLE scan.

**setBluetoothScanListener():** Sets a listener for receiving Bluetooth scan results, which could include information about discovered Bluetooth devices such as their name, address, and signal strength.

**setBleScanListener():** Sets a listener for receiving BLE scan results, which could include information about discovered BLE devices such as their MAC address, advertisement data, and signal strength.

## iii. BluetoothScanListener Interface:

This interface defines callback methods that can be implemented by the app to receive Bluetooth and BLE scan results. It could include methods such as:

**onBluetoothDeviceDiscovered(device: BluetoothDevice):** Called when a Bluetooth device is discovered during a scan. The device parameter could provide information about the discovered device, such as its name, address, and signal strength.

**onBleDeviceDiscovered(device: BluetoothDevice, advertisementData: ByteArray):** Called when a BLE device is discovered during a scan. The device parameter could provide information about the discovered device, and the advertisementData parameter could provide raw data from the device's advertisement packets, which could be parsed to extract additional information.

## iv. bluetoothscanapp Class:

This is the main class of the app that utilizes the BluetoothScanManager and BluetoothScanListener to perform Bluetooth and BLE scanning. It could have methods such as:

**initBluetoothScan():** Initializes the BluetoothScanManager and sets the BluetoothScanListener for receiving Bluetooth scan results.

**initBleScan():** Initializes the BluetoothScanManager and sets the BluetoothScanListener for receiving BLE scan results.

**startScan():** Starts both Bluetooth and BLE scans using the BluetoothScanManager's methods.

**stopScan():** Stops both Bluetooth and BLE scans using the BluetoothScanManager's methods.

**onBluetoothDeviceDiscovered(device: BluetoothDevice):** Implements the callback method from the BluetoothScanListener to handle discovered Bluetooth devices.

**onBleDeviceDiscovered(device: BluetoothDevice, advertisementData: ByteArray):** Implements the callback method from the BluetoothScanListener to handle discovered BLE devices and parse the advertisement data.

# 6. Technological Stack

The technological stack of the Android Bluetooth scanning API includes a combination of software libraries, frameworks, and protocols that enable communication between Android devices and Bluetooth devices.

## Android Studio IDE

Android studio is used as the IDE to Develop, Debug, Testing and build the app

## Android Java API(Bluetooth & BLE)

This API is developed on top of Java Language, which enables the device to access the Bluetooth

and BLE .

1. **BluetoothAdapter:** This is an Android system class that provides methods to control the Bluetooth functionality of a device. It allows you to enable or disable Bluetooth, retrieve the current status of Bluetooth, and discover nearby Bluetooth devices.
2. **BluetoothDevice:** This class represents a remote Bluetooth device, such as a Bluetooth headset or a Bluetooth printer. It provides methods to retrieve information about the device, such as its name, address, and supported services.
3. **BluetoothGatt:** This class provides methods for Bluetooth Low Energy (BLE) communication. It allows you to connect to a BLE device, discover its services and characteristics, and read/write data to/from the device.
4. **BluetoothGattCallback:** This is a callback interface that allows you to receive notifications about events related to Bluetooth Low Energy communication, such as connection status changes, service discovery, and data notifications.
5. **BluetoothProfile:** This is an Android system class that provides constants for identifying different Bluetooth profiles, such as A2DP (Advanced Audio Distribution Profile), HFP (Hands-Free Profile), and GATT (Generic Attribute Profile).
6. **BluetoothLeScanner:** This class provides methods for scanning nearby Bluetooth Low Energy devices. It allows you to start and stop scanning, and receive callbacks with scan results, which include information about discovered devices, such as their MAC address, signal strength, and advertisement data.
7. **BluetoothAdapter.LeScanCallback:** This is a callback interface that allows you to receive notifications about Bluetooth Low Energy devices discovered during a scan.
8. **Android Bluetooth Stack:** This is the underlying Bluetooth stack provided by the Android operating system, which includes the Bluetooth protocol stack, Bluetooth profiles, and Bluetooth hardware drivers. It is responsible for handling the low-level Bluetooth communication between the Android device and Bluetooth devices.

## Figma

This Platform is used to develop the UI design

Overall, the Android Bluetooth scanning API is built on top of the Bluetooth stack provided by the Android operating system, and it provides a high-level abstraction for developers to interact with Bluetooth devices in their Android applications.

# 7. Component Design

The component design of the Bluetooth scanner application will depend on the architecture chosen for the project. However, in general, the following components will be required:

**Bluetooth Adapter:**

The Bluetooth Adapter component will be responsible for enabling Bluetooth on the device, scanning for Bluetooth devices, and retrieving information such as MAC address and RSSI signal strength.

**Background Service:**

The Background Service component will be responsible for scanning for Bluetooth devices in the background. It should use minimal system resources to avoid draining the device's battery.

**List View:**

The List View component will be responsible for displaying the list of Bluetooth devices found during the scan. It should display the device name, MAC address, and RSSI signal strength.

**Top Navigation Bar:**

The Top Navigation Bar component will be responsible for providing access to the different screens of the application. It should display the selected screen.

**Settings:**

The Settings component will be responsible for providing access to the application settings. It should allow the user to change the scan interval and other settings related to Bluetooth scanning.

Overall, the component design of the Bluetooth scanner application should be modular and flexible. The components should be designed to work together seamlessly to provide a smooth and intuitive user experience. The use of common design patterns such as the Model-View-Controller (MVC) architecture can help to simplify the component design and make the application easier to maintain and update.

# 8. Data Design

The data design for the Bluetooth scanner application includes the following components:

Bluetooth Device Data: The Bluetooth device data includes the device's MAC address, name, and RSSI (Received Signal Strength Indicator). This data is collected by the client application using the device's Bluetooth adapter and is sent to the application for storage.

# 9. Error Handling Design

**User notifications:** The app should provide clear and concise error messages to the user when errors occur. The messages should be informative and provide guidance on how to resolve the issue. For example, if the app fails to connect to a Bluetooth device, it should display a message informing the user of the failure and suggest steps to resolve the issue.

**Retry mechanisms:** The app should include retry mechanisms for operations that fail due to network or other transient issues. For example, if the app fails to connect to a Bluetooth device, it should provide an option to retry the connection.

**Error reporting:** The app should include a mechanism for users to report errors and issues to the developer team. This will help the team identify and prioritize issues to be fixed in future updates.

# 10. Security Design

**Authentication:** The app should authenticate the devices it connects to, ensuring that they are legitimate and authorized to communicate with the app. This can help prevent attacks from rogue devices.

**Permissions:** The app should request only the necessary permissions to access device resources, such as the Bluetooth adapter, and should not request unnecessary permissions that could compromise user privacy.

# 11. Testing design

## 11.1 Testing Approach

To ensure the Bluetooth scanner application is reliable and operates as intended, a comprehensive testing approach should be taken. The following testing types and methodologies should be considered:

**Unit Testing:** This involves testing individual code units or modules to ensure that they work as expected. Unit testing should be carried out during development, and a comprehensive suite of tests should be created to cover all possible scenarios.

**Integration Testing:** This tests the interaction between different components of the application to ensure that they work together seamlessly. Integration testing should be carried out once development is complete, and before moving on to system testing.

**System Testing:** This involves testing the entire system as a whole, to ensure that it meets the requirements and functions correctly. System testing should be carried out using real-world scenarios and test data.