FINAL REPORT

Phone Capability Testing Application (Platform: Android)

Date: 8th Dec 2022

Name: Prudhvi Suryadevara

CSU ID: 2822511

e-mail id:
p.suryadevara@vikes.csuohio.edu

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Introduction

- Purpose and Scope

As is well known, there are a plethora of new android devices being launched on a regular basis and tech enthusiasts always look for constructive comparison to understand which phone is better and which is not. The purpose behind the application is to exclusively provide the tech enthusiasts with a portal to explore the different sensors available on the device in terms of their information and the values they produce, explore the functional proficiency of the microphone, explore the functional proficiency of the Bluetooth and, an additional fun function, being a custom music player that changes music on detecting shake.
The project scope is to provide an interest based, enthusiastic approach to understanding a mobile phones capability.

- Product Overview

The application is based on four main functions:

i. Sensor Listing: This function helps in listing out all the sensors on the device, provides specific information about any of the sensor selected and additionally displays the values and specific readings of the said sensor.

ii. Sound Detection: This function helps in understanding the microphone of the device using the application. It plots the sound detected onto a graph to help detect low nodes and high nodes of sound.

iii. Bluetooth Beacon: This function helps in converting the device using the application into a beacon that transmits the Bluetooth signal. This helps in reading the distance and power of the Bluetooth signal.

iv. Shake Music Player: This is a function that can be used as a custom music player as in it changes songs based on shake detection.

The scenarios where the application can be used are as follows:
- Constructive and comparative analysis between multiple android devices.
- Sound plotting to help musicians in terms noting nodal highs and lows through a graph.
- To attain a better understanding of the various sensors incorporated into the device.
- Use the application as a fun custom music player.

The application is very easy to use and read in terms of functionality and usability.

- **Structure of the Document**

This document outlay is very easy to follow and is as such:

- The first segment of the document provides the following information:
  i. The Project management plan detailing the organization, lifecycle model, risk analysis, hardware and software resource requirements, deliverables, and the schedule.
  ii. Requirements specification detailing the stakeholders for the system, use cases, rationale for the use case model, and the non-functional requirements.
  iii. Application architecture detailing the architectural style used, architectural models, technology, software, and hardware used.
  iv. Application design detailing the user interface design, components design, traceability matrix from requirements to detailed design models.

- The second segment of the document provides the following information:
  i. Test management detailing the complete list of system test cases, traceability matrix of test cases to use cases, techniques used for test case generation, test results and assessments.
  ii. Conclusions detailing outcomes of the project, lessons learned and future development.

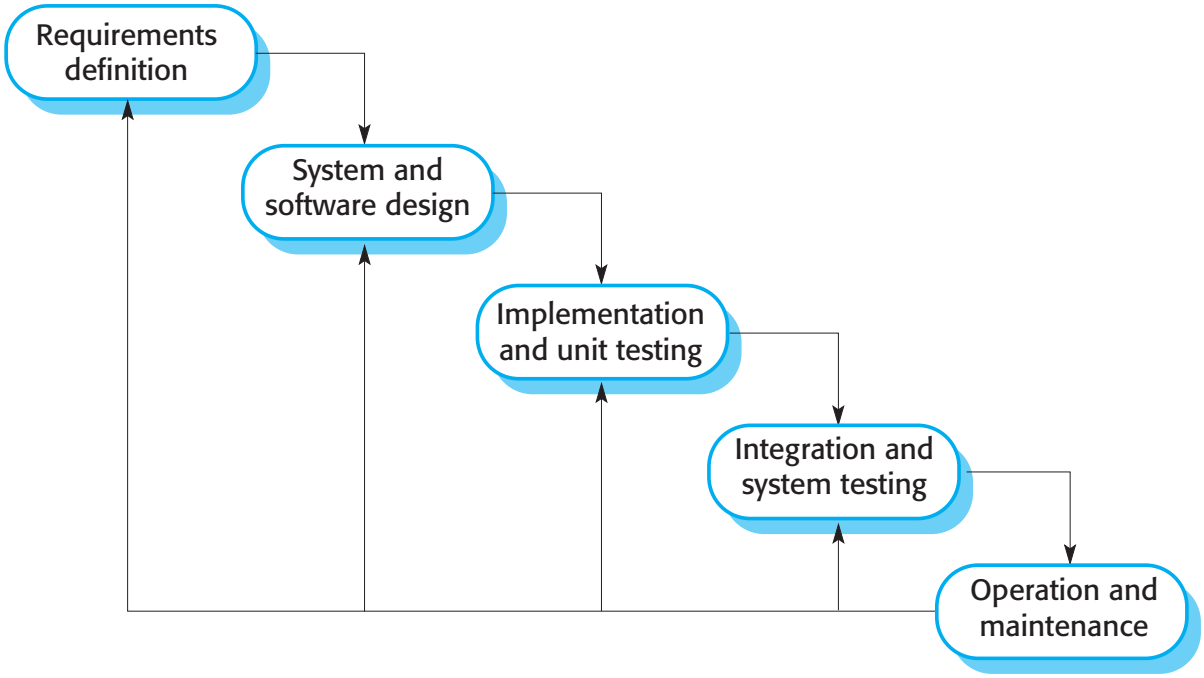
Project Management Plan

- **Project Organization**

As this project was a solo endeavor, I organized the tasks in an orderly fashion as in the first task was to confirm the need of the application following which, layed out the overlay of the functions to be inducted, developing the functions individually, testing each function for any potential errors, confirming the user interface is easily understood and standardized, checked the consistency of integrating the functions together, simulating the output through multiple emulations, installing and launch testing on actual devices, intense debugging to eliminate the crash potency, layout management for resolution matching and finally documenting in detail for a better understanding of the application.

- **Lifecycle Model Used**

The lifecycle model used for the project is the Waterfall Model.



- The first phase involved detailing the requirements for fulling the idea at hand.
- The second phase was to plan and outline the system and the software design.
- The third phase involved developing each proposed function and extensive functional testing of each function down to the unit and component level.
- The fourth phase involved integrating the different functions under one roof and extensively testing the system as a whole, making sure that the application functions as intended.
- The fifth and the final phase was to plan the evolution of the program in terms of both maintenance and operability.

The waterfall model being a plan-driven model involving separate and distinct phases of specification and development supported the conditional requirements of segregating and reaching a conclusion.

- **Risk Analysis**

A full thorough RM3 process was inducted into building the application with focus on set traces of the application along with multiple overseeing methodologies to keep a track on development phase unto the deployment phase. With a clear list of risk drivers for the build using RM3, helped in offsetting any and every minor implication that would have halted the progress.

The following risk table was used to help with the risk analysis.

Risk Driver	Probability in %	Impact (In level 1 to 5)	RMMM (Risk Mitigation, Monitoring and Management)
In Planning Phase			
Misunderstanding on project vision	20	2	M1: Build a document to better explain the usability M2: Keep a regular track of the usability requirement M3: Deploy the document before launch to better explain the need
Misunderstanding on project requirements	20	2	M1: Re-iterate through the exact requirements and the usage methodology M2: Progressing through the build plan as to how and what sources to refer to. M3: pre-assertions and planning might help with management
In Testing Phase			
Inconsistent design	50	5	M1: Changing the design through multiple inputs can help mitigate by bypassing code iterations M2: Following a slow place carefully checking the design pattern and a regular flow of code structure by the book can help manage this step M3: Taking regular insights and seeking assistance when needed and inconsistencies in design purged and rebuilding can help manage
In Implementation Phase			
New code logics to learn	30	3	M1: Use the existing code M2: Keep a track of the code being used and if in case the new code is efficient thorough understanding and documenting can help in monitoring. M3: If the new code is being used it is better to be trained thoroughly so that so that the impact can be reduced substantially.
In Delivery Phase			

Lack of time management	50	5	M1: Use time scheduling and create an action plan M2: Build a chart to keep track of the amount of work and period of time passed form the beginning until the project is complete. Compare the trend with the schedule planned and work around it accordingly. M3: If the project is delayed, finding the root cause, and planning to work on solution to cover the lost time should help in management.
Platform Management	40	4	M1: List out new releases and collect the patch notes M2: Understand the platform working on and make sure to have the patch notes handy to work through if in case of any distortions and also keep a regular track on the new releases. M3: Keep a regular trace on future releases and micromanaging the different platforms being considered can help in efficient management.

- **Hardware and Software Resource Requirements**

As for the hardware, any and every android device would certainly help support the application for helping in real time testing. In terms of hardware, nothing in particular is required to help build the application. In terms of software resource requirement, Android studio has been heavily used to construct and deploy the application and was also utilized in terms of debugging and testing. Android studio is the only resource requirement that has been identified to help support the build of the application.

- **Deliverables and Schedule**

The following timeline chart indicates the schedule followed and adhered.

#	Project Task	Start	End	September		October		November		December
				14	27	11	18	1	8	12
1	Project Plan	14-Sep	27-Sep							
2	Software Requirement Specification	27-Sep	11-Oct							
3	Software Design Specification	11-Oct	18-Oct							
4	Initial Version of the Software	18-Oct	1-Nov							
5	Test Plan	1-Nov	8-Nov							
6	Final Software Product	8-Nov	12-Dec							

The deliverables intended and delivered are as follows:

- Project Plan (Document): Delivered on time
- Software Requirement Specification (Document): Delivered on time
- Software Design Specification (Document): Delivered on time
- Initial version of the software (Presentation): Delivered on time
- Test Plan (Document): Delivered on time
- Application Demonstration (Presentation and Live Demo): Delivered on time
- Final Report (Document): Delivered on time
- Final Software product (Application): Scheduled to be delivered on the 9th of Dec/2022
- User Manual (Document): Scheduled to be delivered on the 9th of Dec/2022
- Developer Guide (Document): Scheduled to be delivered on the 9th of Dec/2022

Requirement Specifications

- Stakeholders for the system

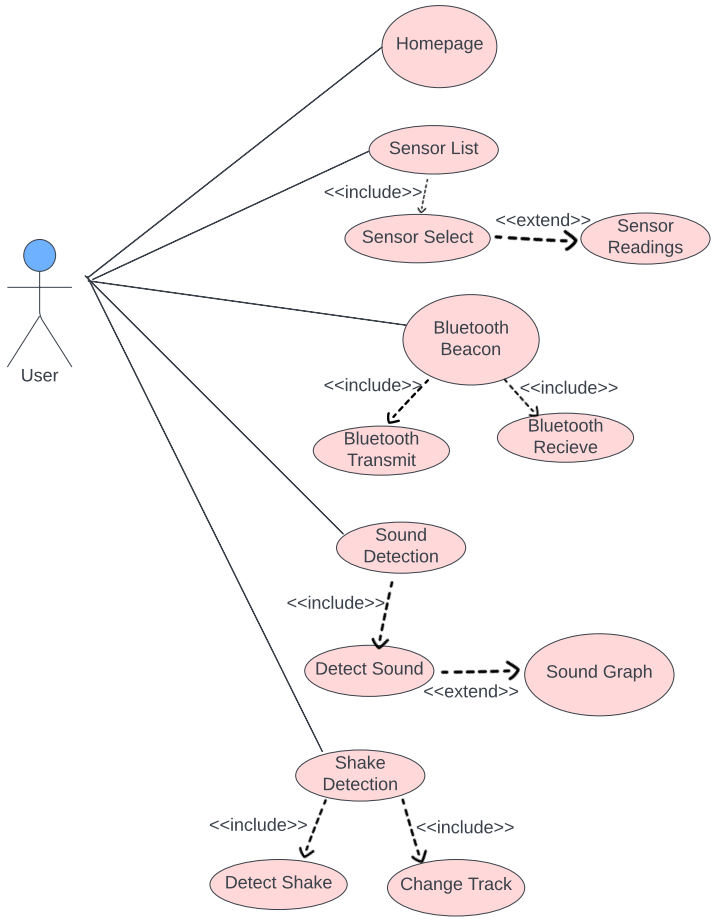
The only true stakeholder for the system is the user as the application is utilized generally by the user with no separate distinction as to who the user can be or is affiliated to.

The following are the usability standards for the user as a stakeholder:

- Can access the Sensor Listing function.
- Can access the Sound Detection function.
- Can access the Bluetooth function.
- Can access the Shake Music Player.

- **Use Cases**

i. Graphical use case model



ii. Textual Description of each use case

- Use Case: Homepage

PCT-App: Homepage	
Actors	User
Description	This allows the user to navigate through the different functions available on the application through buttons.
Data	Function navigation buttons
Stimulus	User command issued by the user
Response	Confirmation of the change of page
Comments	The user must tap the button to help navigate through different functions

- Use Case: Sensor List

PCT-App: Sensor List	
Actors	User
Description	This allows the user to navigate to the list of sensors detected on the mobile device. This includes a function to display the names of all sensors as buttons which then extends to display the readings of the sensor selected
Data	Function navigation buttons, ListView data
Stimulus	User command issued by the user
Response	Prints the list of sensors and on further selection lists the readings
Comments	The user must tap the button to help navigate through different functions of the use case

- Use Case: Bluetooth Beacon

PCT-App: Bluetooth Beacon	
Actors	User
Description	This allows the user to transmit the Bluetooth beacon and receive the Bluetooth transmission to help list out the Bluetooth functionality reading.
Data	Function navigation buttons, Active displacement data.
Stimulus	User command issued by the user.
Response	Prints the Bluetooth readings.
Comments	The user must tap the buttons available to transmit and receive the Bluetooth signal.

- Use Case: Sound Detection

PCT-App: Sound Detection	
Actors	User
Description	This allows the user to access the detect sound function which extends to display the sound detected on the graph
Data	Function navigation buttons, GraphView data
Stimulus	User command issued by the user
Response	Prints the sound node data
Comments	The user must tap the button to detect sound and then relays the data onto displaying a graph

- Use Case: Shake Detection

PCT-App: Shake Detection	
Actors	User
Description	This allows the user to access the shake detection function that extends to change an mp3 file.
Data	Function navigation buttons, Media player
Stimulus	User command issued by the user
Response	Change of mp3 player on shake
Comments	The user must physically shake the mobile device to change the mp3 file

- ## Rationale for the use case model

The rationale for the use case model is to illustrate and define the logical basis for the course of different functions and their functionalities.

The set of reasons followed to support the rationale are as follows:

i. Outline each of the functions utilized by the application.
ii. Outline each of the function's functionalities.
iii. Outline appropriately the actor's accessibility to each defined use case.
iv. Design the use case diagram with a single point agenda of making sure it is easily understood and is appropriated in the sense of readability.

- ## Non-Functional Requirements

The following are the non-functional requirements for the application:

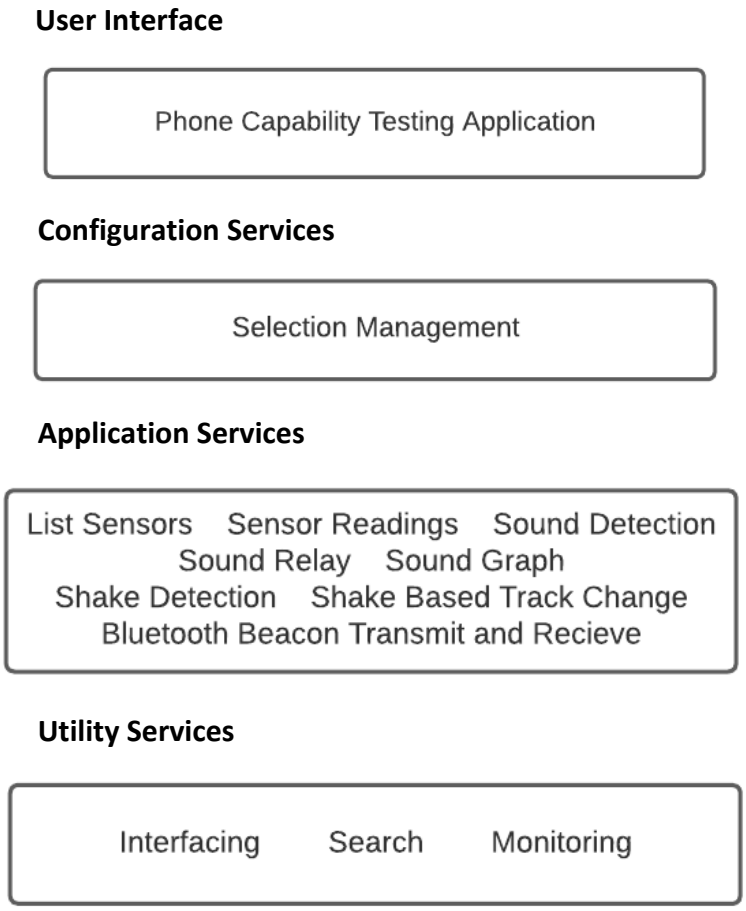
- The minimum android SDK required to run the application is 23 and the targeted android SDK is 32.
- The device intended to use the application requires a minimum of 73 Mb to support installation.
- The device intended to use the application needs to have sensors inbuilt to list the sensors.
- The device intended to use the application needs to be a Bluetooth capable device.
- The user must provide relevant permissions on the device intended to use the application to support different functions.
- The device intended to use the application should have mem block constituency available for the application to read into the device.

Architecture

- Architectural Style used

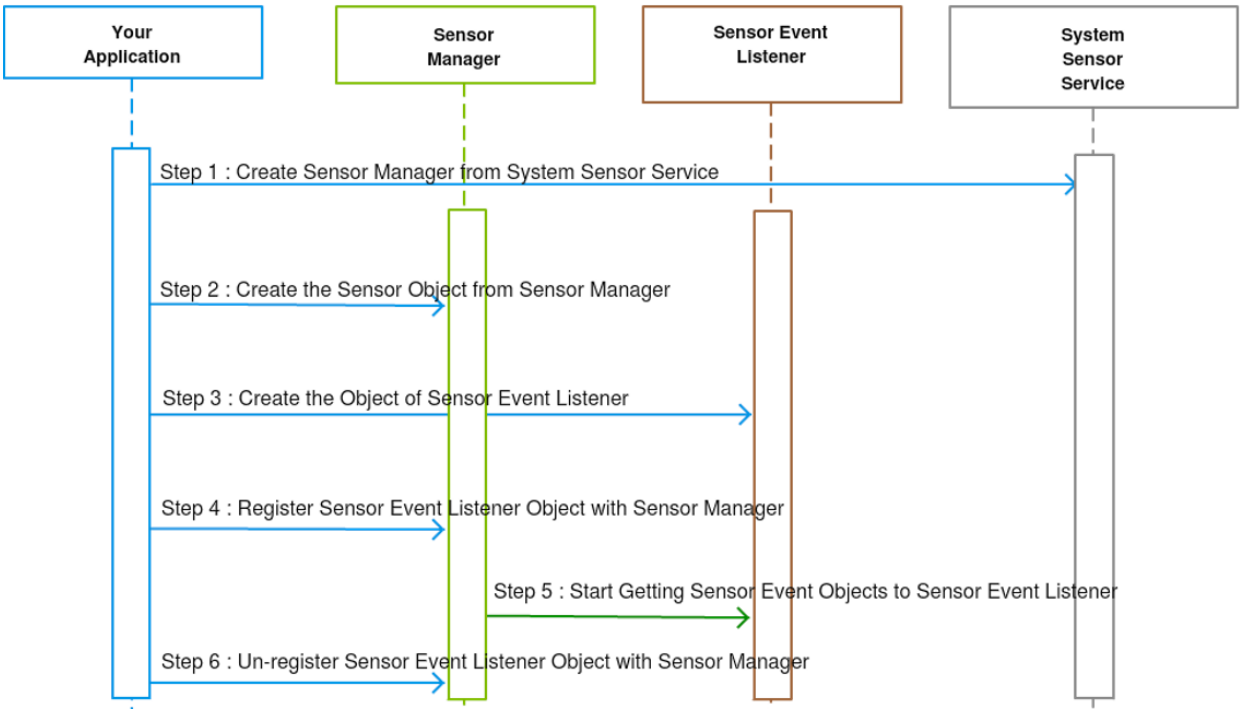
The architectural style used is a Layered Architecture. The reason behind using this specific style is that it allows to organize the system into layers with related functionality associated with each layer wherein, a layer provides services to the layer above it, so the lowest-level layers represent core services that are used throughout the system. This type of architecture suits the nature and theory behind the application also as it supports incremental development of sub systems in different layers of the application.

- Architectural model



- **Logical View Architecture model**

The logical view architecture model has been designed basis the internal working of sensors that support the various functions as the functions heavily rely on the sensors to operate as intended.



i. **Component SensorManager Description:** The SensorManager component is an entry point, which allows the application to request sensor information and register to receive sensor data.

- Interface Description: Sensor Manager is the class that makes it possible for the application to get access to the sensors. It creates the instance of the system sensor service, which provides various APIs to access sensor information on the device. It exposes the methods that list the available and default sensors on the device. This class also provides several sensor constants that are used to report sensor accuracy, sampling period, and calibrate sensors. One of the important tasks of this class is to register and unregister sensor event listeners for accessing a particular sensor.

ii. **Component Sensor Description:** The Sensor component is a combination of data values of each sensor present on the mobile device.

- Interface Description: Sensor is the class that is used to create an instance of a specific sensor. This class provides various methods that let us determine a

sensor's capabilities: Maximum Range, Minimum Delay, Name, Power, Resolution, Reporting Mode, Type, Vendor, Version, isWakeUp Sensor.

iii. **Component SensorEventListener Description:** This component helps with interfacing the sensor data values sent to it to form the SensorEvent class.

- Interface Description: It provides two callbacks to receive the sensor notification. OnSensorChanged() is the first method of the interface, which is called whenever there is any change in the sensor values. The change in sensor value is communicated through the SensorEvent object, passed as a parameter to this method. OnAccuracyChanged() is the second method, which is called whenever there is a change in the accuracy of the sensor values. The sensor object and newly reported accuracy in integers are sent as parameters to this method. There are 4 accuracy integer constants supported by the SensorManager which are: SENSOR_STATUS_ACCURACY_HIGH, SENSOR_STATUS_ACCURACY_MEDIUM, SENSOR_ACCURACY_STATUS_LOW, and SENSOR_ACCURACY_STATUS_UNRELIAB

iv. **Component SensorEvent Description:** This component contains all the information produced from a selected sensor.

- Interface Description: Sensor event is a special kind of class that is used by the operating system to report changes in the sensor values to the listeners. The SensorEvent object contains the following 4 elements: values[] a multidimensional array that holds the sensor values, timestamp refers to time in nanoseconds at which the event happened, accuracy is one of the four accuracy integer constants, sensor is the sensor type that generated the data.

- Technology, Software and Hardware used

The technology used to build the architectural models is Lucid. It is a web-based software application that can be run on any OS such as Windows, Mac, Android etc. with access to internet.

- Rationale for the architectural style and model

The rationale for the architectural style and model is to illustrate and define the logical basis for the course of different layers of the architecture and the relationality among each of the layers.

The set of reasons followed to support the rationale are as follows:

i. This style of the architecture allows organize the system into layers with related functionality associated with each layer wherein, a layer provides services to the

layer above it, so the lowest-level layers represent core services that are used throughout the system.

ii. Clear depiction of the build requirement segregated into different layers to help incremental build.

iii. Allows better integration of different functions in an orderly fashion.

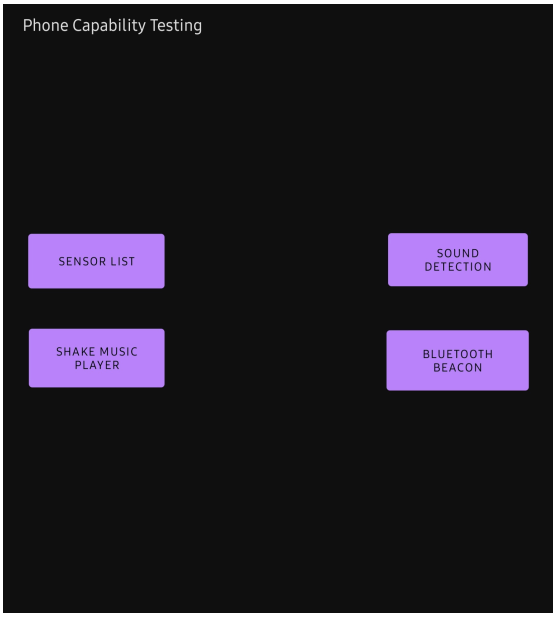
iv. Redundant facilities can be provided in each layer to increase the dependability of the system.

Design

- ## User Interface design

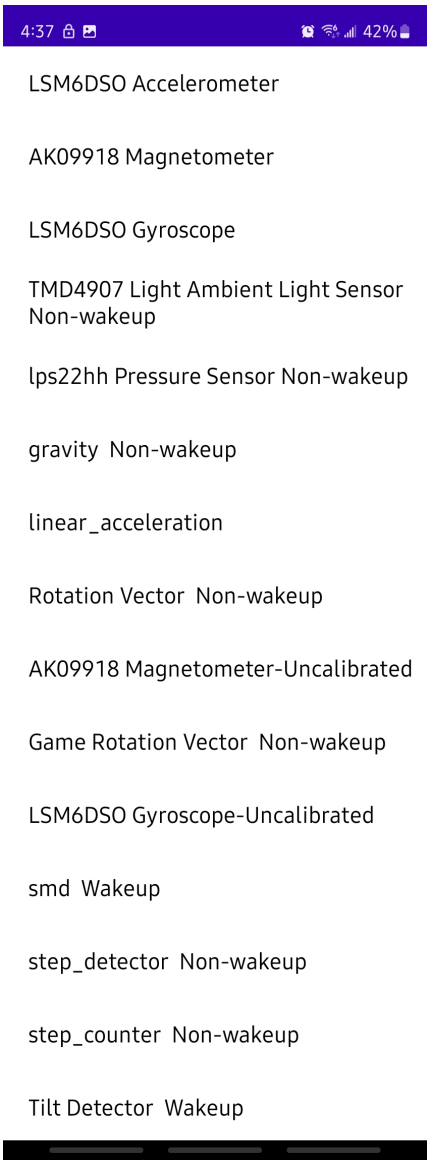
i. Homepage Interface:

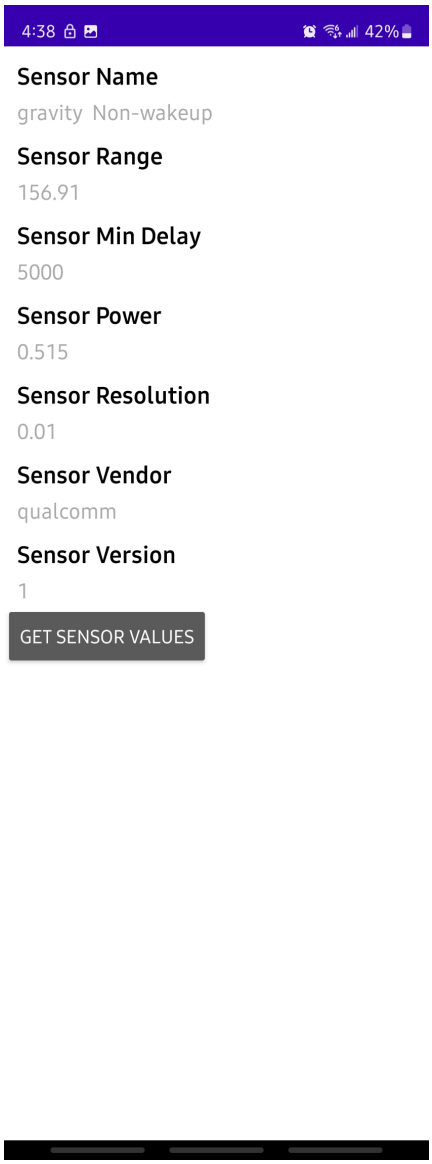
The homepage interface is very simple and easy to use. It allows the user to navigate to through different functions of the application through a simple touch of the button.

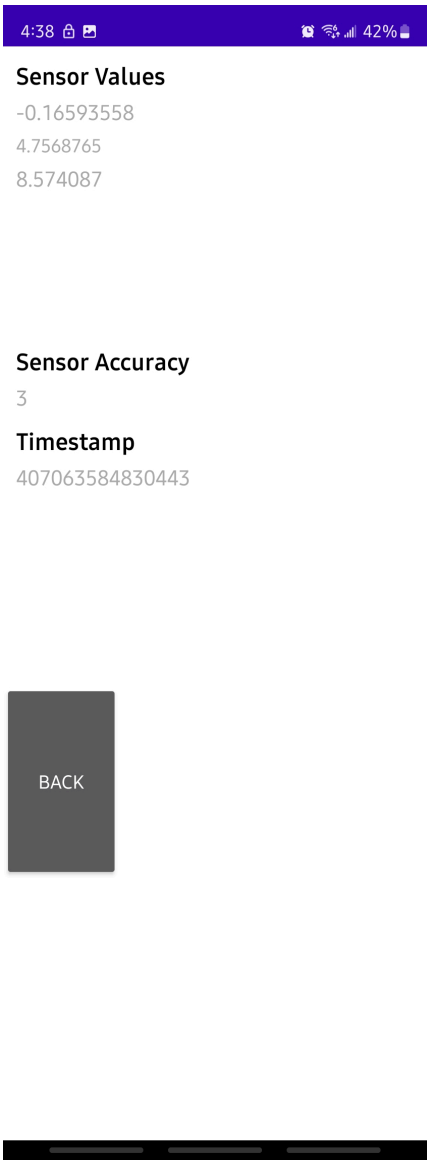


ii. Sensor List

On selecting the Sensor List button, it follows to listing the sensors on the device and then on selecting a specific sensor, it follows to a page with information about the sensor and then to a page depicting the values of the sensor.

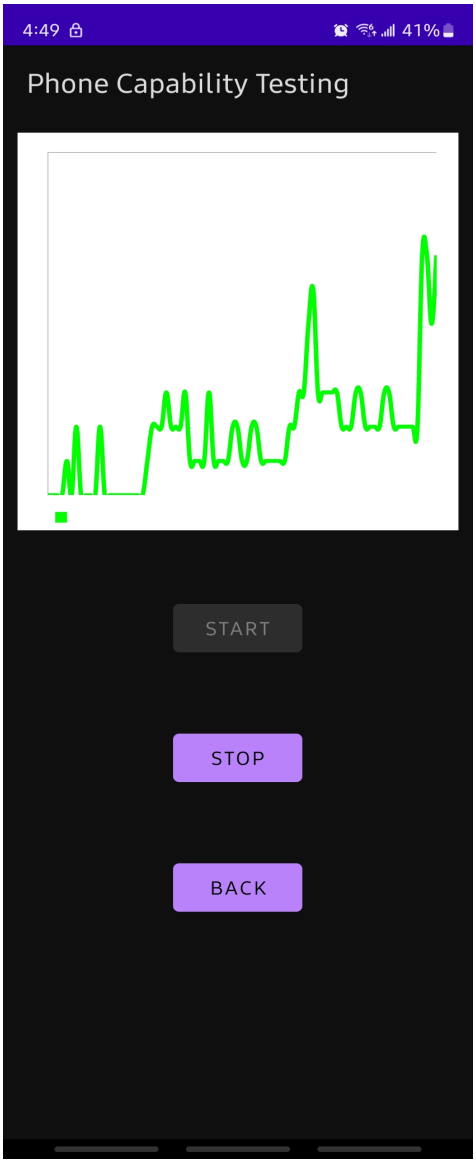






iii. Sound Detection

On selecting the sound detection button, it follows to a page with two buttons Start and Stop, which allows the function to record sound and then project it on a graph.

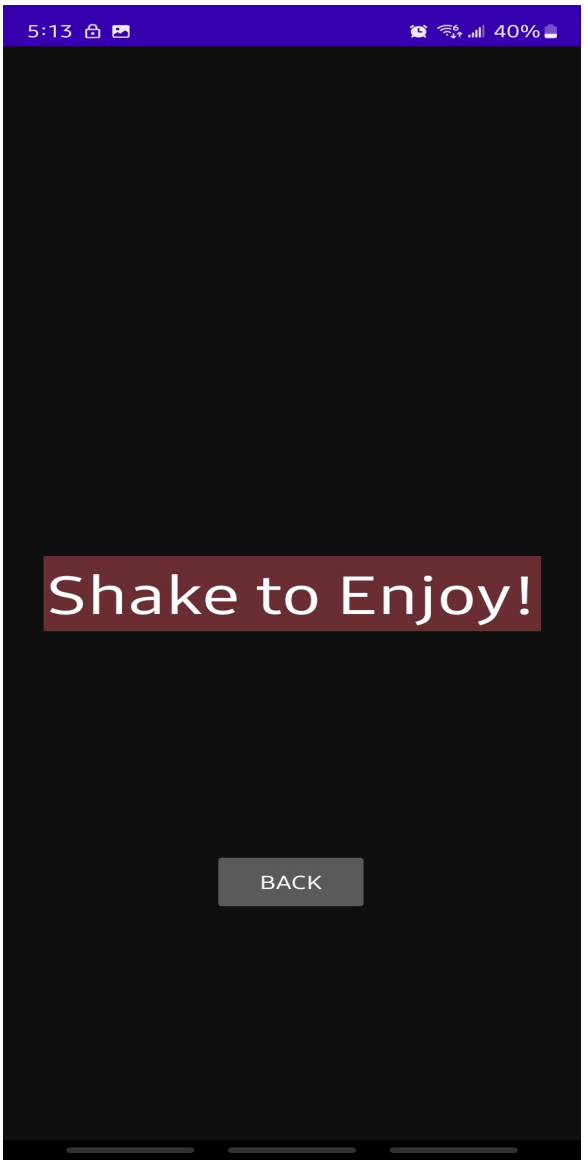


iv. Bluetooth Beacon

On clicking the button, it follows through to a page that has an icon centered on the page and on clicking the button it will transmit the Bluetooth signal following which, on a supported app on a different device can claim the signal and provide the information about Bluetooth signal strength.

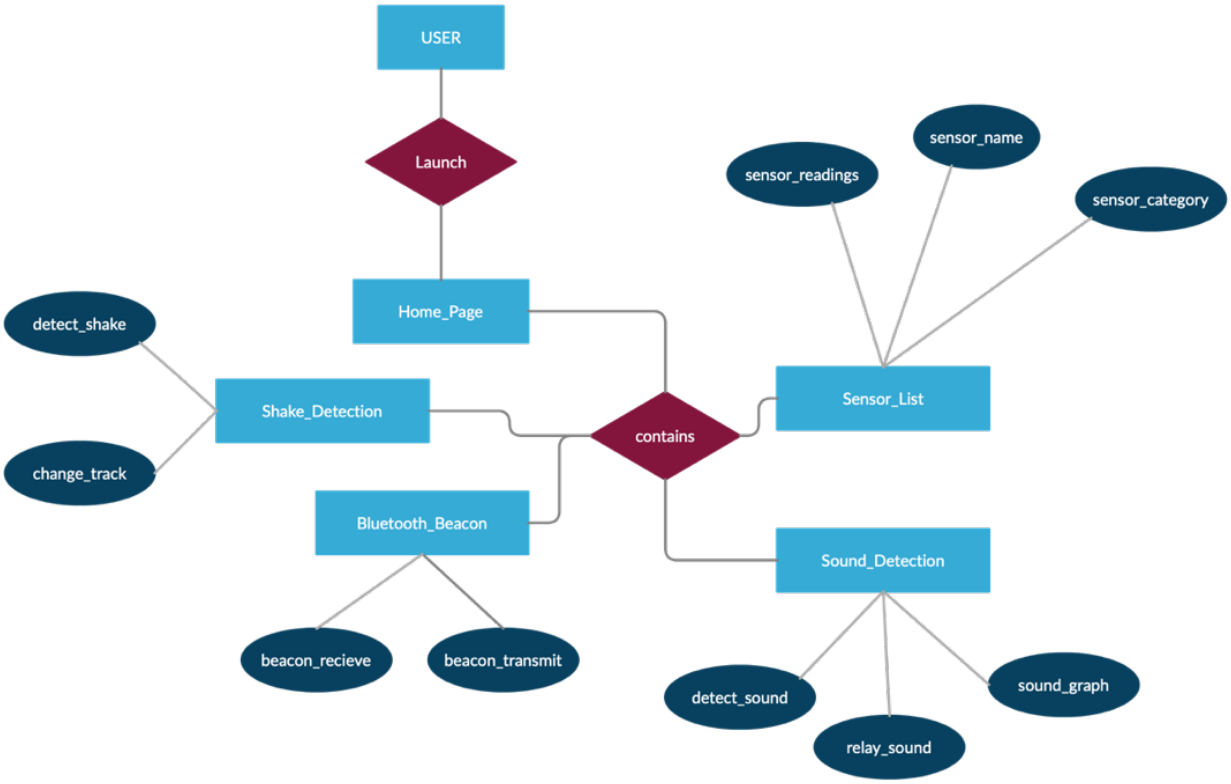
v. Shake Music Player

For this function, on clicking the button it follows to a page which does not eventually display anything as such except the name of the function, but on shake changes the mp3 track inducted into the application directory.



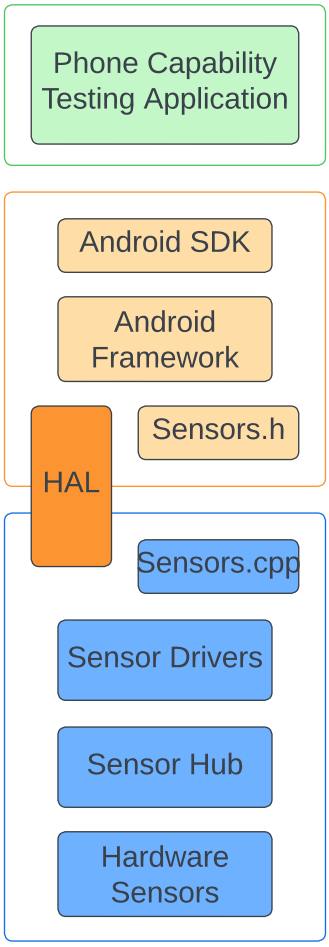
- **Components design**

The diagram below outlines the design structure of different components. It details the flow of the application in terms of navigational context to different components that support the various functionalities of the application.



- **Data Design**

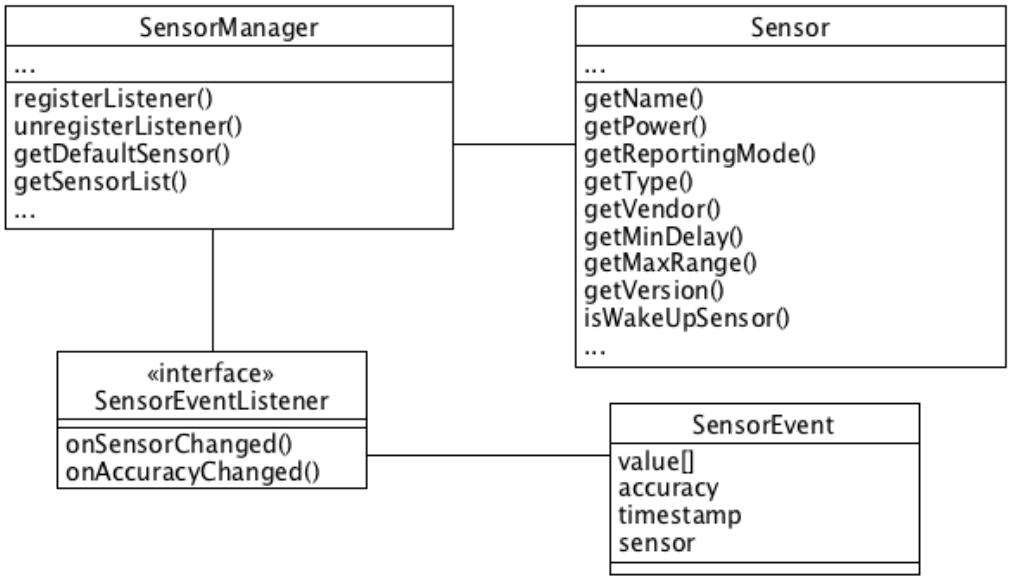
Since the application is heavily reliant on the sensor accessibility for the different functions to operate, the application is built on a stack data structure.



i. **Phone Capability Testing Application:** Consumer of the data from the sensors.
ii. **Android SDK:** This is a layer through which the application can access the sensors.
iii. **Android Framework:** Links the application to a single HAL client.
iv. **HAL (Hardware Abstraction Layer):** This provides the interface between the hardware drivers and the Android Framework.
v. **Sensor:** This is responsible for interacting with the physical devices.
vi. **Sensor Hub:** This is responsible for sensor batching and adding hardware FIFO queue.
vii. **Hardware Sensors:** This is the physical hardware sensors.

This application is a real-time use and destroy model. It does not use any particular database and as such does not require a database design at any point of time.

- **Product Class design**



- **Rationale for design models**

The rationale for the design model is to illustrate and define the logical basis for the course of different components working in tandem and different classes being inducted.

The set of reasons followed to support the rationale are as follows:

i. Clean design helps during implementation.
ii. Clear and concise design detail can help understand each of the components effectively.
iii. Class design helps in effectively implementing different classes without causing any confusion.
iv. Data design helps in organizing data per the structure decided upon for better and organized implementation.

- **Traceability Matrix from Requirements to Design**

Description	Need/Justification	Project Objective	Department	Status
Homepage	Starting point for User	Users should be able to access different functions of the project	Design	Completed
Sensor List	User function access	Users should be able to access the Sensor List function	Design	Completed
Sound Detection	User function access	Users should be able to access the Sound Detection function	Design	Completed
Shake Detection	User function access	Users should be able to access the Shake Detection function	Design	Completed
Bluetooth Beacon	User function access	Users should be able to access the Bluetooth beacon function	Design	Completed

Test Management

- **Complete list of system test cases**

- **Case ID 1: List Sensors Button**

ID	**1.1**
Test Input	Select 'List Sensors' button on emulated device
Expected Output	Navigate to the page listing the sensors
Description	On clicking the List Sensors button on the homepage, the application should transition from the homepage to the page listing the sensors

ID	**1.2**
Test Input	Select ‘List Sensors’ button on Samsung Galaxy S10+
Expected Output	Navigate to the page listing the sensors
Description	On clicking the List Sensors button on the homepage, the application should transition from the homepage to the page listing the sensors

ID	**1.3**
Test Input	Select ‘List Sensors’ button on Samsung Galaxy Z Fold 3
Expected Output	Navigate to the page listing the sensors
Description	On clicking the List Sensors button on the homepage, the application should transition from the homepage to the page listing the sensors

- **Case ID 2: Sensor Description**

ID	**2.1**
Test Input	Select a sensor from the list
Expected Output	Display sensor information on transition
Description	On selecting a sensor, the application should transition from the listing page to the page with description of the sensor selected

- **Case ID 3: Get Sensor Values Button**

ID	**3.1**
Test Input	Select ‘Get Sensor Values’ button on emulated device
Expected Output	Navigate to Sensor Values page
Description	On clicking the ‘Get Sensor Values’ button on the homepage, the application should transition from the homepage to the sensor values page

ID	**3.2**
Test Input	Select ‘Get Sensor Values’ button on Samsung Galaxy S10+
Expected Output	Navigate to Sensor Values page
Description	On clicking the ‘Get Sensor Values’ button on the homepage, the application should transition from the homepage to the sensor values page

ID	**3.3**
Test Input	Select ‘Get Sensor Values’ button on Samsung Galaxy Z Fold 3
Expected Output	Navigate to Sensor Values page
Description	On clicking the ‘Get Sensor Values’ button on the homepage, the application should transition from the homepage to the sensor values page

- **Case ID 4: Sound Detection Button**

ID	**4.1**
Test Input	Select ‘Sound Detection’ button on emulated device
Expected Output	Navigate to Sound Detection functional page
Description	On clicking the ‘Sound Detection” button on the homepage, the application should transition from the homepage to the sound detection functional page

ID	**4.2**
Test Input	Select ‘Sound Detection’ button on Samsung Galaxy S 10+
Expected Output	Navigate to Sound Detection functional page
Description	On clicking the ‘Sound Detection’ button on the homepage, the application should transition from the homepage to the sound detection functional page

ID	**4.3**
Test Input	Select ‘Sound Detection’ button on Samsung Galaxy Z Fold 3
Expected Output	Navigate to Sound Detection functional page
Description	On clicking the ‘Sound Detection’ button on the homepage, the application should transition from the homepage to the sound detection functional page

- **Case ID 5: Low Sound Plotting**

ID	**5.1**
Test Input	Low Sound
Expected Output	Graph plotting on the lower end basis the sound input
Description	A sound is pronounced to the application so that the sound can be recognized in terms of its relational noise value and be plotted on a graph to predict devices microphone accuracy

- **Case ID 6: Medium Sound Plotting**

ID	**6.1**
Test Input	Medium Sound
Expected Output	Graph plotting on the mid end basis the sound input
Description	A sound is pronounced to the application so that the sound can be recognized in terms of its relational noise value and be plotted on a graph to predict devices microphone accuracy

- **Case ID 7: High Sound Plotting**

ID	**7.1**
Test Input	High Sound
Expected Output	Graph plotting on the higher end basis the sound input
Description	A sound is pronounced to the application so that the sound can be recognized in terms of its relational noise value and be plotted on a graph to predict devices microphone accuracy

- **Case ID 8: Shake Detection Button**

ID	**8.1**
Test Input	Select ‘Shake Detection’ button on emulated device
Expected Output	Navigate to Sound Detection functional page
Description	On clicking the ‘Shake Detection” button on the homepage, the application should transition from the homepage to the shake detection functional page

ID	**8.2**
Test Input	Select ‘Shake Detection’ button on Samsung Galaxy S 10+
Expected Output	Navigate to Sound Detection functional page
Description	On clicking the ‘Shake Detection’ button on the homepage, the application should transition from the homepage to the shake detection functional page

ID	**8.3**
Test Input	Select ‘Shake Detection’ button on Samsung Galaxy Z Fold 3
Expected Output	Navigate to Shake Detection functional page
Description	On clicking the ‘Shake Detection button on the homepage, the application should transition from the homepage to the shake detection functional page

- **Case ID 9: Shake Detection**

ID	**9.1**
Test Input	Physical device shake
Expected Output	Start playing the music file onboard on detection motion
Description	On detecting the physical motion, the function should start playing the music file onboard

ID	**9.1.1**
Test Input	Physical device shake
Expected Output	Change track
Description	On detecting the physical motion, the function should then change the track to a different one

- **Case ID 10: Bluetooth Beacon Button**

ID	**10.1**
Test Input	Select ‘Bluetooth Beacon’ button on emulated device
Expected Output	Navigate to Bluetooth beacon functional page
Description	On clicking the ‘Bluetooth Beacon” button on the homepage, the application should transition from the homepage to the Bluetooth beacon functional page

ID	**10.2**
Test Input	Select ‘Bluetooth Beacon’ button on Samsung Galaxy S 10+
Expected Output	Navigate to Bluetooth Beacon functional page
Description	On clicking the ‘Bluetooth Beacon’ button on the homepage, the application should transition from the homepage to the shake detection functional page

ID	**10.3**
Test Input	Select ‘Bluetooth Beacon’ button on Samsung Galaxy Z Fold 3
Expected Output	Navigate to Bluetooth Beacon functional page
Description	On clicking the ‘Bluetooth beacon’ button on the homepage, the application should transition from the homepage to the Bluetooth beacon functional page

- **Case ID 11: Bluetooth Transmit**

ID	**11.1**
Test Input	Tap the Bluetooth icon
Expected Output	Transmit the Bluetooth signal
Description	On clicking the Bluetooth icon, the device should start transmitting the Bluetooth signal

- **Case ID 12: Bluetooth Receive**

ID	**12.1**
Test Input	Tap the receive button
Expected Output	Display signal strength
Description	On clicking the receive button, the device would then display the signal strength

- **Software Tested**

Module Testing
The following modules of the application have been tested:
- Sensor listing
- Sound detection
- Shake detection
- Bluetooth transmission
- Bluetooth receiver

Functional Testing of individual functions of each module
The following components of each module have been tested:
- List device sensors
- Sensor description
- Sensor values
- Sound input detection
- Sound output
- Sound graph plotting
- Motion detection
- Bluetooth transmit
- Bluetooth receive

Interface Testing
The following components of the software will be tested:
- Layout matching (To help match the layout relevant to individual android devices utilizing the application)
- Resolution matching (To help match the resolution fit of each individual android device utilizing the application)

Integration Testing
The integration testing involves the testing of the application on completing the integration of all the base functions under one roof and making sure that no individual base function interrupts the other which may result in a crash of the application or a mismatch in the results of the application.

- **Traceability Matrix of test cases to use cases**

Description	Need/Justification	Project Objective	Department	Status
Homepage	Starting point for User	Users should be able to access different functions of the project	Testing	Completed
Sensor List	User function access	Users should be able to access the Sensor List function	Testing	In Progress
Sound Detection	User function access	Users should be able to access the Sound Detection function	Testing	Completed
Shake Detection	User function access	Users should be able to access the Shake Detection function	Testing	Completed
Bluetooth Beacon	User function access	Users should be able to access the Bluetooth beacon function	Testing	In Progress

- **Techniques used for test case generation**

i. Specification-based Test Case Generation: This method is used to generate a set of test cases from the previously built documents such as Requirement Specification document and the Software Design Specification document.

ii. Sketch Diagram-based Test Case Generation: This method is used to generate test cases from model diagrams like use case diagram and design diagrams.

iii. Source code-based Test Case Generation: This method generally uses source code analysis the help with generating test cases to the trace the code functionality through output.

- **Test Results and Assessments**

Test ID	Test Result
1.1	Pass
1.2	Pass
1.3	Pass
2.1	Pass
3.1	Pass
3.2	Pass
3.3	Pass
4.1	Pass
4.2	Pass
4.3	Pass
5.1	Pass
6.1	Pass
7.1	Pass
8.1	Pass
8.2	Pass
8.2	Pass
9.1	Pass
9.1.1	Pass
10.1	Pass
10.2	Pass
10.3	Pass
11.1	Pass
12.1	Pass

On rigorous testing and working through the noted test cases, all the tests have passed, no errors were noted, the application executes and operates appropriately as intended with each function functioning efficiently. With this, I was able to conclude that the test cases were built effectively and helped with verifying the integrity of the application.

With the following assessment, it can be clearly stated that the software is of the assumed operability and is good in terms of intended quality.

- **Defects Reports**

On successful execution, no defects were observed. There are no pending defects observed as well. Due to on the go debugging, as and when a function was built any noted code defects observed were immediately corrected. Initially there were design constraints as in resolution matching and layout matching in terms of different screen sizes for different devices which has also been resolved during the final test phases.

Conclusions

- ## Outcomes of the project

  All the relevant goals intended have been achieved. The project functions and operates as expected. No conditional halts or crashes are noted. The operability of all the different functions is on point and no crashes have been noted. All the design and architectural models have been considered and the implementation detail and functional proficiency was thoroughly analyzed, executed, and validated basis the said models. All the deliverables have been achieved on time and henceforth not halting the progress. The testing schedule was thoroughly monitored and adhered to ensuring that the testing was constructive and on point ensuring a bug free application.

- ## Lessons Learned

  All through the development period of the project,

- I was able to concretely learn about the waterfall model, the different phases it entails, the purpose and the need of each phase, the lifecycle induction, and relevant backdrops also of the said model.
- I have learned about documentation and the necessity it serves and the importance it holds both in terms of relaying the idea and implementing the idea inclusive of the patterns and standards to follow.
- I have learned about the different design and architectural diagrammatic representations in terms of better understanding the interactions between the components, classes, relationships, and the system.
- I have learned about layered architecture design in detail as to what benefits it entails and the purpose it serves.
- I have learned about test phases and test planning in detail and learned about how to build and utilize the test cases.
- I have learned about integrating different functions onto one consequential bulk.
- Additionally, I have been able to learn in depth, the utilization of Android studio, usage of various sensors on an android device, utilization of supportive functions concerning the inner workings of the android devices.
- I have learned about purposeful utilization of references to support the build idea of the application.
- I have also learned about opinion seeking to formulate the ideology behind the users need and ask for future developments.

  Overall, the project has helped me with a better understanding of project organization and project handling which has been a great learning experience.

- **Future Development**

I plan to implement and develop the following in terms of improving the functional standards of the application built:

- Graph addition to plot the values of the sensor selected from the sensor list function.
- Value declaration on the graph to help read the microphone reading even better on the sound detection function.
- Ability to draft the sound plot onto a custom chart and download as a separate file.
- Ability to add soundtracks per the user choice to enhance the shake music player function.
- Overall functional enhancement of the application.
- Beautification of the user interface to enhance the user experience.

- **References**

i. https://developer.android.com/guide/topics/sensors/sensors_overview
ii. https://developer.android.com/guide/topics/sensors/sensors_motion
iii. https://developer.android.com/guide/topics/sensors/sensors_position
iv. https://developer.android.com/docs/quality-guidelines/large-screen-app-quality
v. https://developer.android.com/guide/topics/manifest/manifest-intro
vi. https://www.tutorialspoint.com/sdlc/sdlc_waterfall_model.htm
vii. https://www.lucidchart.com/blog/types-of-UML-diagrams
viii. https://www.tutorialspoint.com/android/index.htm
ix. https://developer.android.com/training/testing/fundamentals
x. https://medium.com/mindful-engineering/getting-started-with-writing-functional-test-cases-in-android-d293382f7f93
xi. https://www.softwaretestinghelp.com/android-app-testing/