****

**COMSATS Institute of Information Technology,**

**Park Road, Chak Shahzad, Islamabad Pakistan**

**GVI**

**(Gesture based Smartphone Application for Visually Impaired)**

***By***

**Ahmad Ali CIIT/FA12-BCS-008/ISB**

**Awais Ahmad CIIT/ FA12-BCS-025/ISB**

***Supervisor***

***Dr. Yasir Faheem***

***Bachelor of Science in Computer Science***

**The candidate confirms that the work submitted is their own and appropriate  
 credit has been given where reference has been made to the work of others**.

****

**COMSATS Institute of Information Technology,**

**Park Road, Chak Shahzad, Islamabad Pakistan**

**GVI**

**(Gesture based Smartphone Application for Visually Impaired)**

**A project presented to**

**COMSATS Institute of Information Technology, Islamabad**

**In partial fulfillment**

**of the requirement for the degree of**

***Bachelors of Science in Computer Science***

**By**

**Ahmad Ali CIIT/FA12-BCS-008/ISB**

**Awais Ahmad CIIT/ FA12-BCS-025/ISB**

**DECLARATION**

We hereby declare that this software, neither whole nor as a part has been copied out from any source. It is further declared that we have developed this software and accompanied report entirely on the basis of our personal efforts. If any part of this project is proved to be copied out from any source or found to be reproduction of some other. We will stand by the consequences. No Portion of the work presented has been submitted of any application for any other degree or qualification of this or any other university or institute of learning.

Ahmad Ali Awais Ahmad

--------------------------- ---------------------------

**CERTIFICATE OF APPROVAL**

It is to certify that the final year project of BS (CS) “**GVI (Gesture based Smartphone Application for Visually Impaired)**” was developed by “**Ahmad Ali (CIIT/FA12-BCS-008)**” and “**Awais Ahmad (CIIT/FA12-BCS-008**)” under the supervision of “Dr. Yasir Faheem” and that in his opinion; it is fully adequate, in scope and quality for the degree of Bachelors of Science in Computer Sciences.

---------------------------------------

**Supervisor**

---------------------------------------

**External Examiner**

---------------------------------------

**Head of Department**

**(Department of Computer Science)**

**Executive Summary**

In public places, there is often a need for monitoring people and different activities going on, which can be referred later for many reasons including security. Appointing humans for this task involves many problems such as increased employee hiring, accuracy problem, trust, no proof for later use, and also the fact that a human can remember things till a certain time limit. Talking about the current security system, they use dumb still cameras with a continuous recording facility ir-respective of the fact that any event may happen or not. Moreover they are usually pointing at a specific user defined locations so more than one cameras are required to cover the entire region.

To prevent all these problems from prevailing, the CSCS is developed. It is a surveillance system, which provides solution to many of these problems. It is a stand-alone application which doesn’t require any computer to operate. It monitors different situations using a camera which is able to rotate intelligently based on sensor messages and captures the scene in the form of video or photos later reference as well.

**C**ustomizable **S**urveillance **C**ontrol **S**ystem **(CSCS)** is a surveillance system that can be assigned a sensor type as in our case a heat sensor is used, it works accordingly, rotates the camera upon event detection and perform user defined actions like capturing video and stores them, for the future use.

It is an embedded system consisting of Linux fox kit with embedded a running server application also a camera, USB storage device and a sensor node base station is attached with fox kit. LAN communication is used by user to download the videos and to operate the system manually.

**Acknowledgement**

All praise is to Almighty Allah who bestowed upon us a minute portion of His boundless knowledge by virtue of which we were able to accomplish this challenging task.

We are greatly indebted to our project supervisor “Dr. Yasir Fahem”. Without their personal supervision, advice and valuable guidance, completion of this project would have been doubtful. We are deeply indebted to them for their encouragement and continual help during this work.

And we are also thankful to our parents and family who have been a constant source of encouragement for us and brought us the values of honesty & hard work.

Ahmad Ali Awais Ahmad

--------------------------- ---------------------------

**Abbreviations**

|  |  |
| --- | --- |
| **SRS** | Software Require Specification |
| **GVI** | Gesture based smartphone application for Visually Impaired |
|  |  |
|  |  |
|  |  |

**Table of Contents**

[1 Introduction 11](#_Toc449452565)

[1.1 Brief 11](#_Toc449452566)

[1.2 Relevance to Course Modules 11](#_Toc449452567)

[1.3 Project Background 11](#_Toc449452568)

[1.4 Literature Review 11](#_Toc449452569)

[1.6 Methodology and Software Lifecycle for This Project 11](#_Toc449452571)

[1.6.1 Rationale behind Selected Methodology 11](#_Toc449452572)

[2 Problem Definition 12](#_Toc449452573)

[2.1 Problem Statement 12](#_Toc449452576)

[2.2 Deliverables and Development Requirements 12](#_Toc449452577)

[2.3 Current System (if applicable to your project) 12](#_Toc449452578)

[3 Requirement Analysis 13](#_Toc449452579)

[3.1 Use CasesDiagram(s) 13](#_Toc449452581)

[3.2 Detailed Use Case 13](#_Toc449452582)

[3.2.1 Use case Name 13](#_Toc449452583)

[3.3 Functional Requirements 15](#_Toc449452584)

[3.4 Non-Functional Requirements 15](#_Toc449452585)

[4 Design and Architecture 16](#_Toc449452586)

[4.1 System Architecture 16](#_Toc449452591)

[4.2 Data Representation [Diagram + Description] 16](#_Toc449452592)

[4.3 Process Flow/Representation 16](#_Toc449452593)

[4.4 Design Models [along with descriptions] 16](#_Toc449452594)

[5 Implementation 18](#_Toc449452595)

[5.1 Algorithm 18](#_Toc449452597)

[5.2 External APIs 18](#_Toc449452598)

[5.3 User Interface 18](#_Toc449452599)

[6 Testing and Evaluation 19](#_Toc449452600)

[6.1 Selected testing methodology 19](#_Toc449452602)

[6.2 Requirement/Use-case based testing [Compulsory] 19](#_Toc449452603)

[6.2.1 Functional Requirement/ Usecase Name 19](#_Toc449452604)

[6.3 Tools used [Optional] 20](#_Toc449452605)

[7 Conclusion and Future Work 21](#_Toc449452606)

[7.1 Conclusion 21](#_Toc449452608)

[7.2 Future Work 21](#_Toc449452609)

[8 References 22](#_Toc449452610)

**List of Figures**

Fig 1.1 Block Diagram 8

Fig 2.1 Use Case Diagram 9

Fig 3.1 Flow Diagram……………………………………………………………………………..

Fig 4.1 Class Diagram……………………………………………………………………………..

Fig 5.1 Sequence Diagram………………………………………………………………………...

Fig 6.1 Activity Diagram………………………………………………………………………….

# Introduction

This system is an android application which will operate using tap clicks, tilt gestures and phone buttons. The product captures and manipulate three gestures (left tilt, right tilt and forward tilt) and define a control/functionality for every specific gesture. This project use of some existing API’s which will provide the system with features of speech to text, haptic response and voice assistance. The product provides control over three applications to the user which include reading unread messages, making a call and calling existing contacts. Furthermore an additional module is added in the system which will read out the current date and time to the Visually Impaired user. It is facilitating the user with all the basic features that are used daily. However, the visually impaired user cannot perform complex operations like playing games, editing a pdf document using this product. There is a learning phase in the system, which guides the user with the input of gestures.

## Brief

GVI Application is an android application which provides usage of Call, Message and Contacts using motion gestures, screen taps and volume keys. Furthermore, a need base additional functionality of Organizer (speaking date and time) is added for the ease of the user. GVI Application works smoothly providing voice feedback on every stage to guide user of the current screen. There is a separate training section which will train the user how to use the system. The methodology used to design this system is Pair Programing and Agile methodology. The Project was developed on Android studio and all three modules mentioned in the previous sections have been implemented and tested correctly.

## Relevance to Course Modules

A brief explanation of how your project is related to various courses studied during BCS. To design this system, our knowledge of Object Oriented Programming helped a lot in creation of classes and the relation between them. We used knowledge from Data Structures to implement Binary search in module of Contacts. Knowledge of Data Base Systems helped us in the handling of SQL lite databases which are used in modules of Contacts and Messaging. Two courses of Software Engineering helped in designing the methodology, design of flow and testing of our Application. Furthermore, the basic knowledge of programming flow and syntax knowledge was provided by the course of ICP.

## Project Background

The idea behind the Project was to use the motion sensor of the Mobile phone device and voice assistance. We worked on the accelerometer of the mobile phone and used Text to Speech API for providing voice feedback. Accelerometers are devices that can measure acceleration (the rate of change in velocity), but in smartphones, they're able to detect changes in orientation and tell the screen to rotate. An accelerator looks like a simple circuit for some larger electronic device. Despite its humble appearance, the accelerometer consists of many different parts and works in many ways, two of which are the piezoelectric effect and the capacitance sensor. Typical accelerometers are made up of multiple axes, two to determine most two-dimensional movement with the option of a third for 3D positioning. Most smartphones typically make use of three-axis models, whereas cars simply use only a two-axis to determine the moment of impact. The sensitivity of these devices is quite high as they’re intended to measure even very minute shifts in acceleration. The more sensitive the accelerometer, the more easily it can measure acceleration. We captured the readings, applied algorithms on it and manipulated it according to our usage. TTS (Text-To-Speech) is a form of speech synthesis that converts text into voice output. Text-To-Speech software basically takes the text you write and turns it into speech files that you can use. We used this according to our need in different modules.

## Literature Review

Gestures, such as a wave or a head nod, form a natural part of our face-to-face communication1. While gestures are most often just supporting our verbal communication, they can also by themselves be a simple and very effective way of communication. Although ubiquitous in communication between people, the gesture modality has been mostly ignored in human-computer interaction. The idea of a computer, that responds to hand gestures instead of to speech, a keyboard or a mouse or, was popularised through the \_lm Minority Report (2002) and its famous scene where the protagonist (Tom Cruise) makes objects y around with just a sweep of his arm (c.f. Sa\_er, 2008). However, in academic research, the concept of gestures as a communication channel has been around for decades. In the mid 1970s, the artist and trained computer scientist Myron Krueger, created with VIDEOPLACE one of the earliest interactive systems that responds to gestures (Kalawsky, 1993). The advances in microelectronics in recent years have reduced the costs of numerous small and precise sensors. This has inspired a lot of research e\_ort around multimodal interfaces. The resulting prototypes usually remain con\_ned to special applications like sign language interpretation (c.f. Vogler and Metaxas, 2001). These small sensors are also \_nding their way into an increasing number of consumer electronic devices, such as mobile phones and game consoles. With the release of its game console Wii in 2006, Nintendo made the concept of controlling games via (more or less) natural gestures available to a broad public. The Wii controller device contains an embedded accelerometer sensor, allowing the detection of the controller's movements. In June 2009, Microsoft announced with Project Natal a multimodal interaction system that supports speech recognition and full body gesture recognition and motion tracking. Mobile devices, such as mobile phones, mobile gaming devices or wearable computers, provide new possibilities for communication and computing on the go, but they also introduce new problems due to small screens and input facilities. Noticing this and tying in with the research on multimodal interfaces, this project studies accelerometer-based gesture recognition with the iPhone. A hand gesture recognition

application was implemented and experimented with, in order to serve as a possible supplementary or alternative interaction modality for mobile phones. A gesture, such as a hand movement describing a circle, can not likely be repeated in exactly the same way. On top of that, the e\_ect of normal sensor inaccuracy inevitably leads to dissimilar data samples, even if the former was possible. Therefore, `intelligent' software algorithms, known as pattern recognition

algorithms need to be employed. There is a wide variety of possible methods to choose from, including hybrid methods that combine several approaches. In this paper, so-called discrete hidden Markov models and the corresponding algorithms for training and recognition of gesture models are examined. Accelerometer-based gesture recognition has been discussed in a number of publications (e.g. Kela et al., 2006; Prekopcs\_ak et al., 2008) and with various input devices, such as the Wii remote2 (Schlomer et al., 2008). One major difference to most of the existing papers is { in the herewith presented project, sensing and capture of the data as well as gesture recognition and training, is all done on the mobile device itself. Most other approaches transmit the sensor data to a nearby computer, where the training and recognition software is running, or perform o\_-line analysis of the data. A short summary about differrent gesture recognition approaches is given in chapter 2 and relevant academic publications as well as commercial products are mentioned. Thereafter, various pattern recognition methods, which are relevant in the context of gesture recognition, are presented. Finally, hidden Markov models, the pattern recognition method utilised in this project, are introduced and the gesture recognition and training apparatus, developed in this project is outlined.

Automated gesture recognition has been investigated in various academic research projects, which yielded a number of practical applications and commercial products. This section provides a short summary about several sensor techniques that have been examined in the context of gesture recognition, thereafter, the narrower \_eld of accelerometer-based gesture recognition is illuminated and a selection of relevant publications and commercial non-commercial products are cited. Prior to the actual recognition of gestures, a solution must be found to automatically register the position or movements of the human body or body parts, such as arms and hands. This is sometimes also referred to as motion tracking and a variety of different techniques have been investigated concerning this task. The optical approach uses one or more video cameras to track coloured markers or skin coloured areas (e.g. Elmezain et al., 2008) in the video images. Sometimes self-organising methods, which extract motion trajectories from consecutive sequences of video images are applied in order to register body movement (e.g. Yang et al., 2002). This vision-based approach represents a vast of research, the further exploration of which, however, lies beyond the scope of this paper. Gesture recognition can also be realised based on magnetic motion tracking or mechanical, exoskeleton based tracking (Bergamasco et al., 2007), but these methods are less common. More recently also acoustic methods have been explored. Based on the sounds generated by a pen Seniuk and Blostein (2009) examine the recognition of simple gestures, such as circling or scratch-out, but also the recognition of some handwritten characters and words. In the context of this project, accelerometer-based gesture recognition is investigated. An accelerometer is small sensor, which can measure the acceleration of itself, or the device it is built-in respectively. Based on the acceleration pro\_le, which originates from the movement of the device, the classi\_cation into previously de\_ned gestures is possible. This gesture recognition technique has been discussed in many publications and various input devices have been tried. Accelerometer-based gesture recognition with a Nintendo Wii controller is explored by Schlomer et al. (2008). In their work, the sensor data is transmitted via Bluetooth from the controller device to a nearby PC, where the signal is processed. They also released the open-source Java library wiigee1, which facilitates training and recognition of hand gestures, performed with a Wii controller. LiveMove by AiLive2 is a commercial software product, which provides comparable functionality. It is aimed for game developers and provides gesture recognition methods for several accelerometer-equipped game controllers. Similar to the work by Schlomer et al. (2008) for the Wii controller, in the paper by Prekopcs\_ak (2008) a mobile phone is used to capture accelerometer data, transmit it via Bluetooth to a nearby computer, where the data is analysed and gesture classi\_cation is performed in quasi real-time. The paper compares two distinct pattern recognition methods, support vector machines and hidden Markov models. The results showed that both methods performed almost equally well. Support vector machines achieved an average recognition rate of 96 percent and hidden Markov models reached 97.6 precent. While most recent approaches process the signal data on a separate computer, for the OpenMoko mobile phone family, an open-source software library3 exists, which allows gesture recognition that runs on the device itself. Kela et al. (2006) study accelerometer-based gesture recognition for controlling a television, a video recorder and lighting in a design environment. For a sensing device, they utilise a so-called SoapBox (Sensing, Operating and Activating Peripheral Box). This is a sensor device developed for research activities in the context of ubiquitous computing (c.f. Tuulari and Ylisaukko-oja, 2002). The sample data is analysed with discrete hidden Markov models and the corresponding training and classi\_cation methods were implemented in MATLAB. One of the dynamic gesture recognition systems was

developed by Hofmann et al. (1998) for the recognition of 10 gestures of the German sign language. The data samples were processed o\_-line and it took several hours.

## Methodology and Software Lifecycle for This Project

This project uses incremental model in which the requirements will be divided into various milestones. These milestones are further divided into smaller, more easily managed modules.  Each module passes through the requirements, design, implementation and testing phases. As it is an android application, classes are involved so we use OOP methodology. We have made classes and called the functions by creating objects. Furthermore, this system follows event driven programming. In this methodology, an action is performed of a specific event and the even listener is constantly waiting for the event to occur. The events include a touch, gesture, button press etc.

### Rationale behind Selected Methodology

The benefit of agile software development is that it is capable of significantly reducing the overall risk associated with software development.

In particular, agile development accelerates the delivery of deliverables, and through a process of continuous planning and feedback, is able to ensure that value is continuing to be maximized throughout the development process. As a result of this iterative planning and feedback loop, teams are able to continuously align the delivered software with desired needs, easily adapting to changing requirements throughout the process. By measuring and evaluating status based on the undeniable truth of working, testing software, much more accurate visibility into the actual progress of projects is available. Finally, as a result of following an agile process, at the conclusion of a project is a software system that much better addresses the customer needs. So we chose this methodology.

# Problem Definition

In this chapter we will discuss the precise problem to be solved and how our system provides a solution.



## Problem Statement

It is the right of every individual to get benefitted from the latest technology and ease provided today in cell phones. However, visually impaired are still deprived from using their smartphones because of the complexities they cannot handle. Although there are many related applications present today, like Google Talkback and Speech Recognition which benefit the users with disabilities. Google Talkback, for example, targets users who have poor or no visibility and provides voice assistance for usage. Whereas, Speech recognition facilitates usage on voice commands. But, there is no single application (one app) which provides full control over the above mentioned applications to them.

## Deliverables and Development Requirements

For development, we require knowledge of JAVA (Programming language), along with knowledge of Android framework and a system on which Android Studio needed to run. Furthermore an Android device or an emulator for execution and testing purposes.

## Current System (if applicable to your project)

There is no system, which exists, providing similar functionality like our system.

# Requirement Analysis

This section contains Software Requirements for our system.



## Use Case Diagram

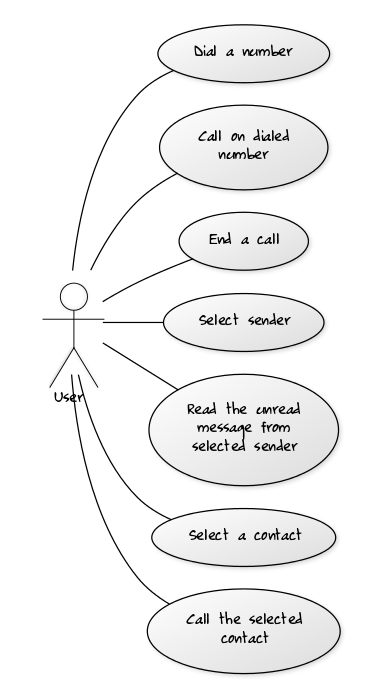


Fig 3.1 Use Case Diagram GVI

## Detailed Use Case

Use cases are a widely used and highly regarded format for capturing requirements. Before writing functional requirement use cases can help you to understand the requirements in the way user expect. Following table presents you not only the template to write use case(s) as well as guides you to write each section with example.

### Launch GVI Application

|  |  |
| --- | --- |
| **Use Case ID:** | UC-1 |
| **Use Case Name:** | Launch GVI application |
| **Actors:** | User |
| **Description:** | User wants to start the application and the result will be launch of application. |
| **Trigger:** | User shakes the device. |
| **Preconditions:** | The device should be in unlock screen |
| **Postconditions:** | 1. Application is launched 2. Voice feedback is given |
| **Normal Flow:** | 1. User unlocks the device 2. User shakes the device 3. Application starts |
| **Alternative Flows:** | User locks the device |
| **Exceptions:** | The device is locked |
| **Includes:** | - |
| **Special Requirements:** | The accelerometer of the device should be working properly. |
| **Assumptions:** | The user should know basic usage of phone (knowledge of buttons) |
| **Notes and Issues:** | * Does the user know how to shake up to the defined threshold value? |
|  |  |

### Open Call

|  |  |
| --- | --- |
| **Use Case ID:** | UC-2.1 |
| **Use Case Name:** | Open Call |
| **Actors:** | User |
| **Description:** | User wants to open Call section and the screen appears |
| **Trigger:** | Single tap on home screen |
| **Preconditions:** | Application is running and home screen is displayed on the device. |
| **Postconditions:** | 1. Call section is opened 2. Call screen appears 3. Voice feedback is given |
| **Normal Flow:** | 1. User unlocks the device 2. User shakes the device 3. Application starts 4. Home screen is tapped once 5. Call section is opened |
| **Alternative Flows:** | 2.1a. If screen tap is greater than one shake the device again.   1. Shake the device again 2. Tap 1 time on home screen 3. Call Section is opened |
| **Exceptions:** | 1. The device is locked 2. The user is not on home screen 3. Voice feedback is provided |
| **Includes:** | UC-1 |
| **Special Requirements:** | Touch sensor of the screen should be working properly |
| **Assumptions:** | User should know how to tap the screen and user understands English language |
| **Notes and Issues:** | - |
|  |  |

### Open Contacts

|  |  |
| --- | --- |
| **Use Case ID:** | UC-2.2 |
| **Use Case Name:** | Open Contacts |
| **Actors:** | User |
| **Description:** | User wants to open Contact section and the screen appears |
| **Trigger:** | Two taps on home screen |
| **Preconditions:** | Application is running and home screen is displayed on the device. |
| **Postconditions:** | 1. Contact section is opened 2. Contact screen appears 3. Voice feedback is given |
| **Normal Flow:** | 1. User unlocks the device 2. User shakes the device 3. Application starts 4. Home screen is tapped twice 5. Call section is opened |
| **Alternative Flows:** | 2.2a. If screen tap is not equal to 2, shake the device again.   1. Shake the device again 2. Tap 2 times on home screen 3. Contact Section is opened |
| **Exceptions:** | 1. The device is locked 2. The user is not on home screen 3. Voice feedback is provided |
| **Includes:** | UC-1 |
| **Special Requirements:** | Touch sensor of the screen should be working properly |
| **Assumptions:** | User should know how to tap the screen and user understands English. |
| **Notes and Issues:** | * Tap two times quickly |
|  |  |

### Open Messaging

|  |  |
| --- | --- |
| **Use Case ID:** | UC-2.3 |
| **Use Case Name:** | Open Messaging |
| **Actors:** | User |
| **Description:** | User wants to Messaging section and the screen appears |
| **Trigger:** | Three taps on home screen |
| **Preconditions:** | Application is running and home screen is displayed on the device. |
| **Postconditions:** | 1. Messaging section is opened 2. Message screen appears 3. Voice feedback is given |
| **Normal Flow:** | 1. User unlocks the device 2. User shakes the device 3. Application starts 4. Home screen is tapped 3 times 5. Message section is opened |
| **Alternative Flows:** | 2.3a. If screen tap is greater than one shake the device again.   1. Shake the device again 2. Tap 3 time on home screen 3. Message Section is opened |
| **Exceptions:** | 1. The device is locked 2. The user is not on home screen 3. Voice feedback is provided |
| **Includes:** | UC-1 |
| **Special Requirements:** | Touch sensor of the screen should be working properly |
| **Assumptions:** | User should know how to tap the screen and have knowledge of English language |
| **Notes and Issues:** | * Tap 3 times quickly |
|  |  |
|  |  |

### Open Organiser

|  |  |
| --- | --- |
| **Use Case ID:** | UC-2.4 |
| **Use Case Name:** | Open Organiser |
| **Actors:** | User |
| **Description:** | User wants to open Organiser section and the screen appears |
| **Trigger:** | Four taps on home screen |
| **Preconditions:** | Application is running and home screen is displayed on the device. |
| **Postconditions:** | 1. Organiser section is opened 2. Organiser screen appears 3. Voice feedback is given |
| **Normal Flow:** | 1. User unlocks the device 2. User shakes the device 3. Application starts 4. Home screen is tapped 4 times 5. Organiser section is opened |
| **Alternative Flows:** | 2.4a. If screen tap is not equal to 2, shake the device again.   1. Shake the device again 2. Tap 2 times on home screen 3. Organiser Section is opened |
| **Exceptions:** | 1. The device is locked 2. The user is not on home screen 3. Voice feedback is provided |
| **Includes:** | UC-1 |
| **Special Requirements:** | Touch sensor of the screen should be working properly |
| **Assumptions:** | User should know how to tap the screen and user understands English. |
| **Notes and Issues:** | * Tap four times quickly |
|  |  |

### Open Training

|  |  |
| --- | --- |
| **Use Case ID:** | UC-2.5 |
| **Use Case Name:** | Open Training |
| **Actors:** | User |
| **Description:** | User wants to open training section and the screen appears |
| **Trigger:** | Five taps on home screen |
| **Preconditions:** | Application is running and home screen is displayed on the device. |
| **Postconditions:** | 1. Training section is opened 2. Training screen appears 3. Voice feedback is given |
| **Normal Flow:** | 1. User unlocks the device 2. User shakes the device 3. Application starts 4. Home screen is tapped 5 times 5. Training section is opened |
| **Alternative Flows:** | 2.5a. If screen tap is not equal to 2, shake the device again.   1. Shake the device again 2. Tap 2 times on home screen 3. Contact Section is opened |
| **Exceptions:** | 1. The device is locked 2. The user is not on home screen 3. Voice feedback is provided |
| **Includes:** | UC-1 |
| **Special Requirements:** | Touch sensor of the screen should be working properly |
| **Assumptions:** | User should know how to tap the screen and user understands English. |
| **Notes and Issues:** | * Tap five times quickly |
|  |  |

### Dial a Call

|  |  |
| --- | --- |
| **Use Case ID:** | UC-3 |
| **Use Case Name:** | Dial a call |
| **Actors:** | User |
| **Description:** | User wants to dial a call and the screen appears |
| **Trigger:** | Number entered using number button taps and call button is long pressed. |
| **Preconditions:** | Application is running and user has tapped once correctly. |
| **Postconditions:** | 1. Called the entered number. |
| **Normal Flow:** | 1. User unlocks the device 2. User shakes the device 3. Application starts 4. Home screen is tapped once 5. Call section is opened 6. Dials a number 7. Number is Called |
| **Alternative Flows:** | 3a. If a number is not entered.   1. Voice feedback to enter number 2. User enters a number 3. Call is performed on entered number |
| **Exceptions:** | 1. The device is locked 2. The user is not in Call section 3. Voice feedback is provided |
| **Includes:** | UC-1 and UC-2.1 |
| **Special Requirements:** | Touch sensor of the screen should be working properly |
| **Assumptions:** | User should know how to tap the screen and user understands English. |
| **Notes and Issues:** | - |
|  |  |

### Select a contact using volume keys

|  |  |
| --- | --- |
| **Use Case ID:** | UC-4.1 |
| **Use Case Name:** | Select a contact using volume keys |
| **Actors:** | User |
| **Description:** | User wants to open select a contact and the screen appears |
| **Trigger:** | Volume key is pressed in Contact section. |
| **Preconditions:** | Application is running and Contact screen is displayed on the device. |
| **Postconditions:** | 1. Contact is selected 2. Voice feedback is given |
| **Normal Flow:** | 1. User unlocks the device 2. User shakes the device 3. Application starts 4. Home screen is tapped twice 5. Call section is opened 6. Volume key is pressed 7. Contact is selected |
| **Alternative Flows:** | 4.1a. Contact is not selected  1. Volume key is not pressed  2. Forward gesture is performed  3. Voice feedback given  4. Volume key is pressed  5. Voice feedback is provided |
| **Exceptions:** | 1. Forward gesture is performed before volume key press 2. Voice feedback is provided. |
| **Includes:** | UC-1, UC-2.2 |
| **Special Requirements:** | Touch sensor of the screen should be working properly |
| **Assumptions:** | User should know how to tap the screen and user understands English. |
| **Notes and Issues:** | Knowledge about button press and Forward gesture. |
|  |  |

### Dial a number

|  |  |
| --- | --- |
| **Use Case ID:** | UC-4.2 |
| **Use Case Name:** | Dial a number |
| **Actors:** | User |
| **Description:** | User wants to dial a call and call is performed |
| **Trigger:** | Forward gesture is performed with long press |
| **Preconditions:** | 1. Application is running 2. Home screen is displayed on the device. 3. Contact is selected |
| **Postconditions:** | Call is performed. |
| **Normal Flow:** | 1. User unlocks the device 2. User shakes the device 3. Application starts 4. Home screen is tapped twice 5. Call section is opened 6. Volume key is pressed 7. Contact is selected 8. Number is dialed |
| **Alternative Flows:** | 4.2a. Forward gesture is not performed  1. Contact is selected   1. Shake the device again 2. Tap 2 times on home screen 3. Contact Section is opened 4. Select contact 5. Give forward gesture 6. Number is called |
| **Exceptions:** | 1. The device is locked 2. The user is not on contact section 3. Voice feedback is provided |
| **Includes:** | UC-1, UC-2.2, UC-4.1 |
| **Special Requirements:** | Touch sensor of the screen should be working properly |
| **Assumptions:** | User should know how to tap the screen and user understands English. |
| **Notes and Issues:** | Knowledge of performing forward gesture |

### Read unread message

|  |  |
| --- | --- |
| **Use Case ID:** | UC-5 |
| **Use Case Name:** | Read unread message |
| **Actors:** | User |
| **Description:** | User wants to read unread message and message is read out by the system |
| **Trigger:** | Forward gesture is performed with long press |
| **Preconditions:** | 1. Application is running 2. Home screen is displayed on the device. 3. 3 taps performed correctly 4. Sender is selected using volume keys |
| **Postconditions:** | Message is read |
| **Normal Flow:** | 1. User unlocks the device 2. User shakes the device 3. Application starts 4. Home screen is tapped thrice 5. Message section is opened 6. Volume key is pressed 7. Sender is selected 8. Message is read |
| **Alternative Flows:** | 5a. Sender not selected   1. Forward gesture is performed 2. Error feedback (voice) 3. User presses volume keys 4. Forward gesture performed 5. Message is read |
| **Exceptions:** | 1. No unread message 2. Voice feedback is provided |
| **Includes:** | UC-1, UC-2.2, UC-2.3 |
| **Special Requirements:** | Touch sensor of the screen should be working properly |
| **Assumptions:** | User should know how to tap the screen and user understands English. |
| **Notes and Issues:** | Knowledge of performing forward gesture |

### Speak date and time

|  |  |
| --- | --- |
| **Use Case ID:** | UC-6 |
| **Use Case Name:** | Speak date and time |
| **Actors:** | User |
| **Description:** | User wants to listen the date and time and system speaks it (voice) |
| **Trigger:** | Tap the screen |
| **Preconditions:** | 1. Application is running 2. Home screen is displayed on the device. 3. 4 taps performed correctly |
| **Postconditions:** | System reads out the date and time |
| **Normal Flow:** | 1. User unlocks the device 2. User shakes the device 3. Application starts 4. Home screen is tapped 4 times 5. Organiser section is opened 6. System speaks current date and time |
| **Alternative Flows:**  **[Alternative Flow 1 – Not in Network]** | 6a. Device is shaked   1. Voice feedback provide on home screen 2. Screen is tapped 4 times 3. Organiser section is opened 4. Screen is tapped 5. Current date and time is read out |
| **Exceptions:** | Device is locked. |
| **Includes:** | UC-1, UC-2.4 |
| **Special Requirements:** | Touch sensor of the screen should be working properly |
| **Assumptions:** | User should know how to tap the screen and user understands English. |
| **Notes and Issues:** |  |
|  |  |

## Functional Requirements

In this section all the functional requirements for the system are discussed. Every requirement is discussed individually in tabular format.

Table 1: Requirements for Open Application

|  |  |
| --- | --- |
| ID | R-1 |
| Title | Open Application |
| Requirement | When the user shakes the device (when screen is unlocked) the app gets triggered. |
| Source | Interviews and observations. |
| Rationale | Providing the user with an easy method to open app |
| Restrictions and Risk | The screen of the device should be in unlock state to perform shake operation. Secondly, there is a chance that this action is not performed if the user does not shake to the set criteria |
| Dependencies | None |
| Priority | High |

Table 2: Requirements for Setting Application Priority

|  |  |
| --- | --- |
| ID | R-2 |
| Title | Set Application Priority |
| Requirement | The user can set priority for most frequently used application |
| Source | Interviews and observations. |
| Rationale | To make the system adaptable. |
| Restrictions and Risk | None |
| Dependencies | R-1 |
| Priority | Medium |

Table 3: Requirements for Listening for Input

|  |  |
| --- | --- |
| ID | R-3 |
| Title | Listen for input |
| Requirement | Before performing an action, the system should be able to get a gesture for performing that action. So the system should be able to listen for this input (gesture) when required. |
| Source | Discussions |
| Rationale | For correct working of the system |
| Restrictions and Risk | Working accelerometer in the device |
| Dependencies | R-1 |
| Priority | High |

Table 4: Requirements for Recognizing Input Gesture

|  |  |
| --- | --- |
| ID | R-4 |
| Title | Recognize the input gesture |
| Requirement | After getting the gesture (as an input), the system should be able to perform recognition on it to identify if it is a valid gesture or not. |
| Source | Internet and Discussions |
| Rationale | For correct working of the system |
| Restrictions and Risk | The device should be in running state. Secondly, there is a chance that a gesture might not get recognized if the values of the accelerometer does not fall in the recognized gestures category. |
| Dependencies | R-3 |
| Priority | High |

Table 5: Requirements for Performing Action on Gesture

|  |  |
| --- | --- |
| ID | R-5 |
| Title | Perform action on recognized gesture |
| Requirement | After recognizing the gesture, the system should be able to perform an action against it. |
| Source | Discussions |
| Rationale | Allow user to perform the desired task |
| Restrictions and Risk | A specific action can only be performed if it is associated with a recognized gesture. |
| Dependencies | R-4 |
| Priority | High |

Table 6: Requirements for Feedback of Action performed by user

|  |  |
| --- | --- |
| ID | R-6 |
| Title | Feedback of Action performed by the user |
| Requirement | On every input, the system should provide a feedback in terms of a small vibration so that the user is notified that some instruction is received by the device. |
| Source | Discussions and Interviews |
| Rationale | For the system to be user friendly. |
| Restrictions and Risk | The device should support vibration mechanism (hardware). |
| Dependencies | R-1 |
| Priority | Medium |

Table 7: Requirements for Voice Assistance

|  |  |
| --- | --- |
| ID | R-7 |
| Title | Voice Assistance |
| Requirement | At every step, the system should assist the user with voice feedback for better understanding of the flow of events occurring in the system. |
| Source | Observation and discussions |
| Rationale | To make the system user-friendly |
| Restrictions and Risk | All the sound components of the device should be in working state. |
| Dependencies | R-1 |
| Priority | Medium |

Table 8: Requirements for Closing Application

|  |  |
| --- | --- |
| ID | R-8 |
| Title | Close Application |
| Requirement | When the user locks the device (when screen is locked) the app should close. |
| Source | Discussions |
| Rationale | Providing the user with an easy method to close app |
| Restrictions and Risk | Separate button to lock the screen (on device) should be working. Secondly, the auto lock (set on time of inactivity) should be disabled else it will perform the same task. |
| Dependencies | R-1 |
| Priority | Medium |

## Non-Functional Requirements

* Performance:

This includes the response time, processing time and reporting time of a system. In our system this time depends on the hardware (processors) present in the device. However for our test device, the response time is between 0.5-1.5 seconds.

* Capacity and Scalability

As android applications have different versions which make it compatible for different devices, this system is designed on version 19 which makes it compatible for over 95 percent of the devices

* Availability

This application has no timing limitations and complications such as hours of operation or location of operation.

### 

### Maintainability

### This System follows standard coding convention which are easily understood by a technical person.

### Recovery

The chances of failure of an android application are very low if it does not contain any bugs or issues. However if the application crashes or stops for no visible reason it restarts automatically within one second.

# Design and Architecture

This chapter will discuss the design and architecture of our system. We will first start with the architecture.



## System Architecture

The system architecture followed will be the MVC (Model View Controller). We place Activities in the View part of MVC, as it forms the UIs of Android applications and updates them throughout the lifecycle of the Android application. The Services and Content Provider or data are hidden components and monitoring the behavior of the application, they are placed in Model part of MVC. Widget components at kernel are acting as Controller of MVC and interacts with Models and Views. Therefore, we can separate presentation (View) from the logic of application (Model). Frontend activities rely on the XML and Java files and their interactions in Android app programming. Java files/libraries are falling in Model part of MVC, whereas XML belongs to the View part.

## Data Representation

This project is an android application and uses XML Schema format to represent data of the system.

<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">

<xs:element name="manifest">

<xs:complexType>

<xs:sequence>

<xs:element name="uses-sdk">

<xs:complexType>

<xs:simpleContent>

<xs:extension base="xs:string">

<xs:attribute ref="and:minSdkVersion" xmlns:and="http://schemas.android.com/apk/res/android"/>

<xs:attribute ref="and:targetSdkVersion" xmlns:and="http://schemas.android.com/apk/res/android"/>

</xs:extension>

</xs:simpleContent>

</xs:complexType>

</xs:element>

<xs:element name="uses-permission">

<xs:complexType>

<xs:simpleContent>

<xs:extension base="xs:string">

<xs:attribute ref="and:name" xmlns:and="http://schemas.android.com/apk/res/android"/>

</xs:extension>

</xs:simpleContent>

</xs:complexType>

</xs:element>

<xs:element name="application">

<xs:complexType>

<xs:sequence>

<xs:element name="activity">

<xs:complexType>

<xs:sequence>

<xs:element name="intent-filter">

<xs:complexType>

<xs:sequence>

<xs:element name="action">

<xs:complexType>

<xs:simpleContent>

<xs:extension base="xs:string">

<xs:attribute ref="and:name" xmlns:and="http://schemas.android.com/apk/res/android"/>

</xs:extension>

</xs:simpleContent>

</xs:complexType>

</xs:element>

<xs:element name="category">

<xs:complexType>

<xs:simpleContent>

<xs:extension base="xs:string">

<xs:attribute ref="and:name" xmlns:and="http://schemas.android.com/apk/res/android"/>

</xs:extension>

</xs:simpleContent>

</xs:complexType>

</xs:element>

</xs:sequence>

</xs:complexType>

</xs:element>

</xs:sequence>

<xs:attribute ref="and:name" xmlns:and="http://schemas.android.com/apk/res/android"/>

<xs:attribute ref="and:configChanges" xmlns:and="http://schemas.android.com/apk/res/android"/>

<xs:attribute ref="and:label" xmlns:and="http://schemas.android.com/apk/res/android"/>

<xs:attribute ref="and:theme" xmlns:and="http://schemas.android.com/apk/res/android"/>

<xs:attribute ref="and:screenOrientation" xmlns:and="http://schemas.android.com/apk/res/android"/>

</xs:complexType>

</xs:element>

</xs:sequence>

<xs:attribute ref="and:allowBackup" xmlns:and="http://schemas.android.com/apk/res/android"/>

<xs:attribute ref="and:icon" xmlns:and="http://schemas.android.com/apk/res/android"/>

<xs:attribute ref="and:label" xmlns:and="http://schemas.android.com/apk/res/android"/>

<xs:attribute ref="and:supportsRtl" xmlns:and="http://schemas.android.com/apk/res/android"/>

<xs:attribute ref="and:theme" xmlns:and="http://schemas.android.com/apk/res/android"/>

</xs:complexType>

</xs:element>

</xs:sequence>

<xs:attribute type="xs:string" name="package"/>

</xs:complexType>

</xs:element>

</xs:schema>

XML SCHEMA for GVI application

## Process Flow/Representation

The Flow of our system starts from a shake of device (when phone is unlocked and is in wake screen) which launches the application. A menu is spoken out to the user via voice assistance and user is taken to a specific screen once the user enters the desired option after listening the instructions. A sample is provided for the first three modules of the system.

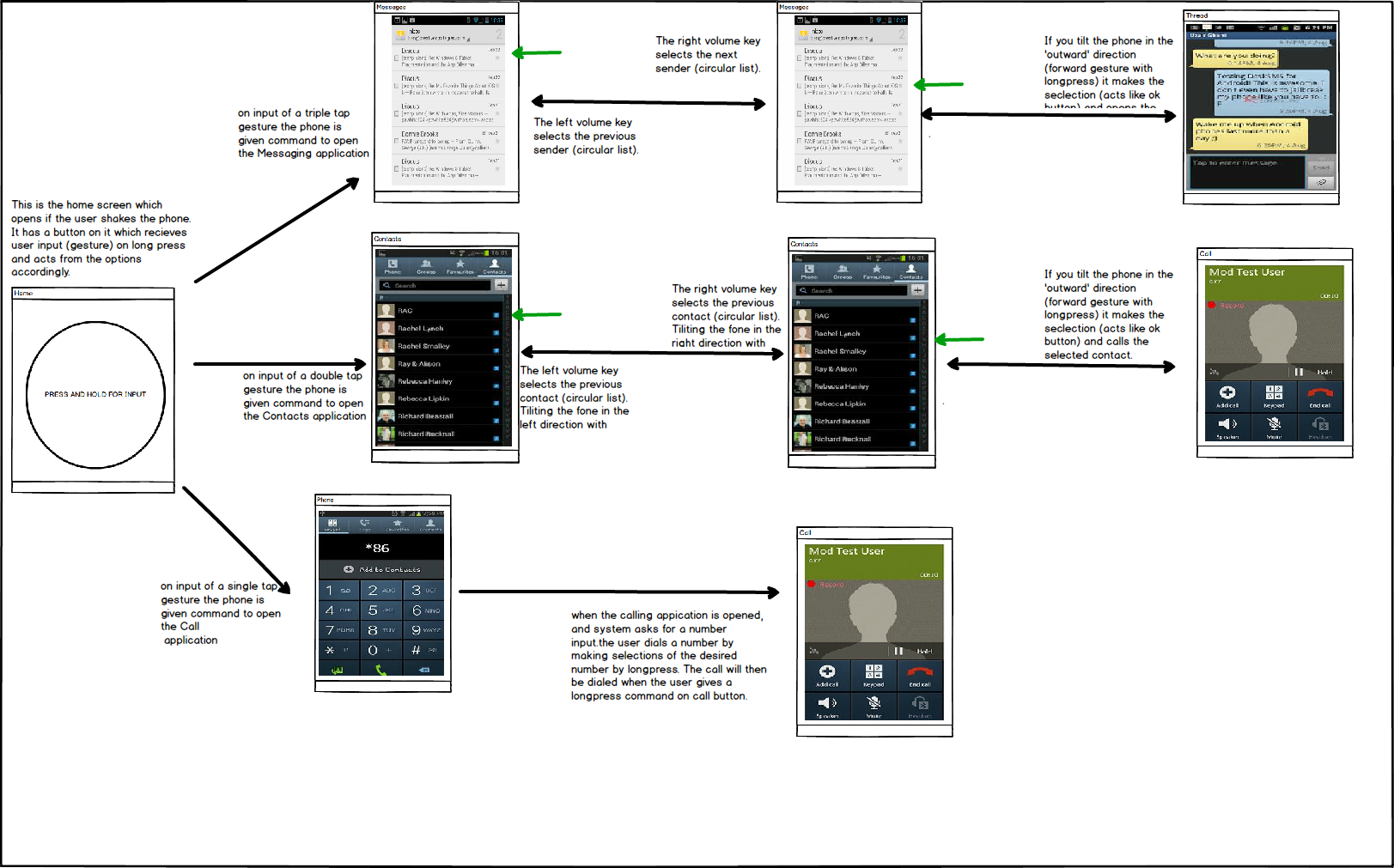


Fig 4.1 Flow Diagram for GVI

## Design Models

The Design models help to understand the states and flow of the system.

**Class Diagram:**

A class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.

This system comprises of 8 classes in total in which the main class (shake device) is public. These classes are Autostart, MyService, Organiser, Taining, Messaging Contacts, Call and ShakeDevice.

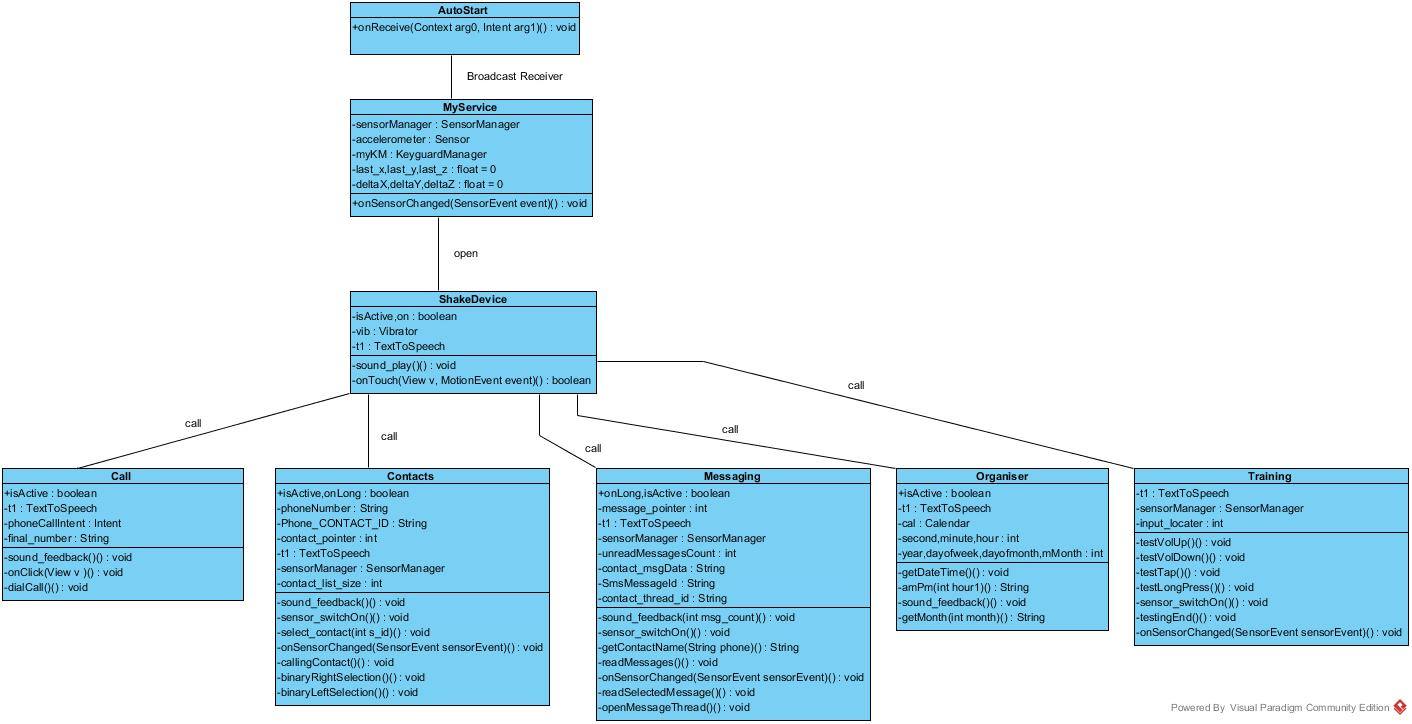


Fig 4.1 Class Diagram for GVI

**Sequence Diagram:**

A Sequence diagram is an interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. A sequence diagram shows object interactions arranged in time sequence. The sequence diagram of GVI is shown below.

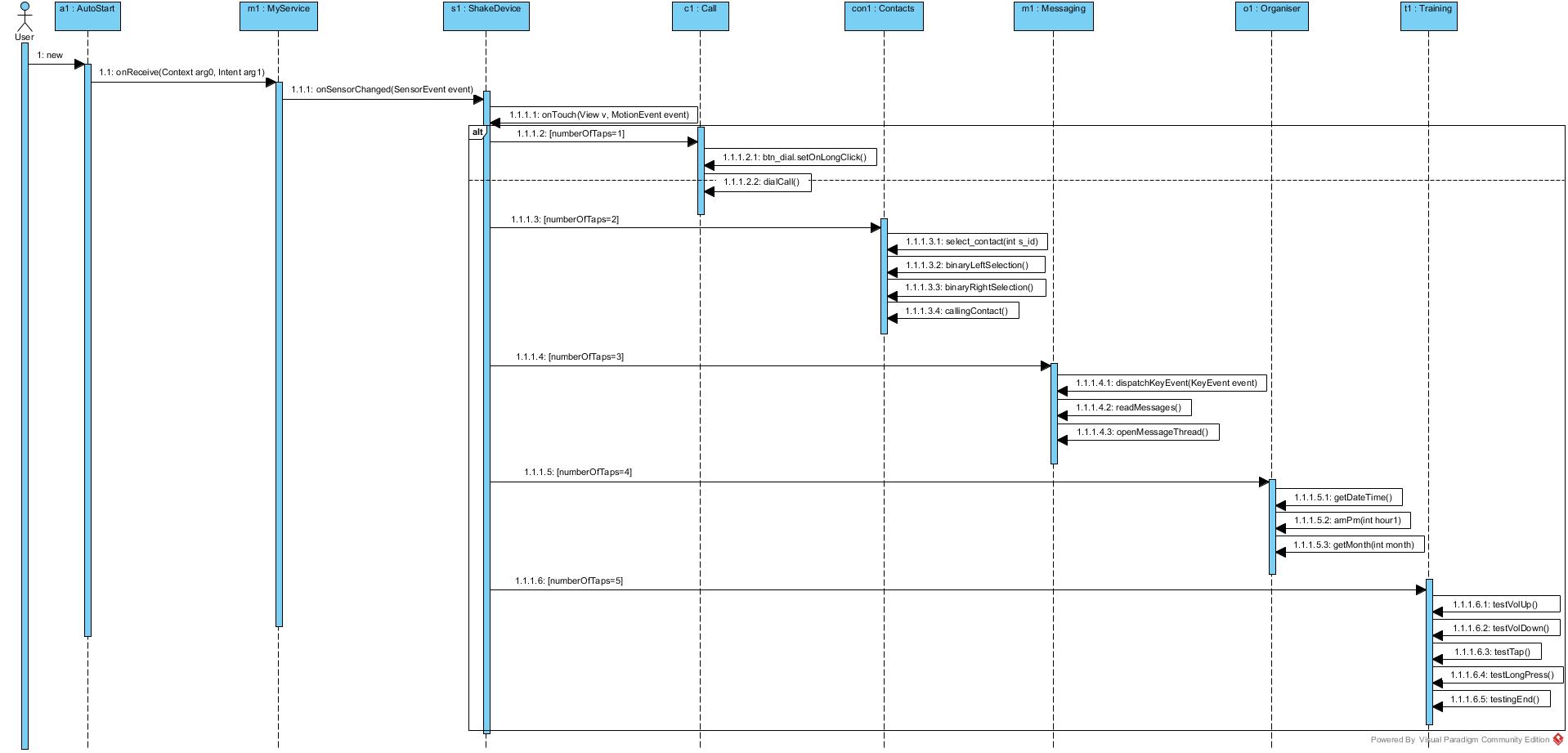


Fig 5.1 Sequence Diagram for GVI

**Activity Diagram:**

Activity diagram is an important diagram in UML to describe dynamic aspects of the system. Activity diagram is basically a flow chart to represent the flow form one activity to another activity. The activity can be described as an operation of the system. The activity diagram explains the flow of event as follows. When the screen is open (unlocked) and you shake the device (up to the defined threshold) the app starts. From this state the user can shift to 5 states which are shown below.

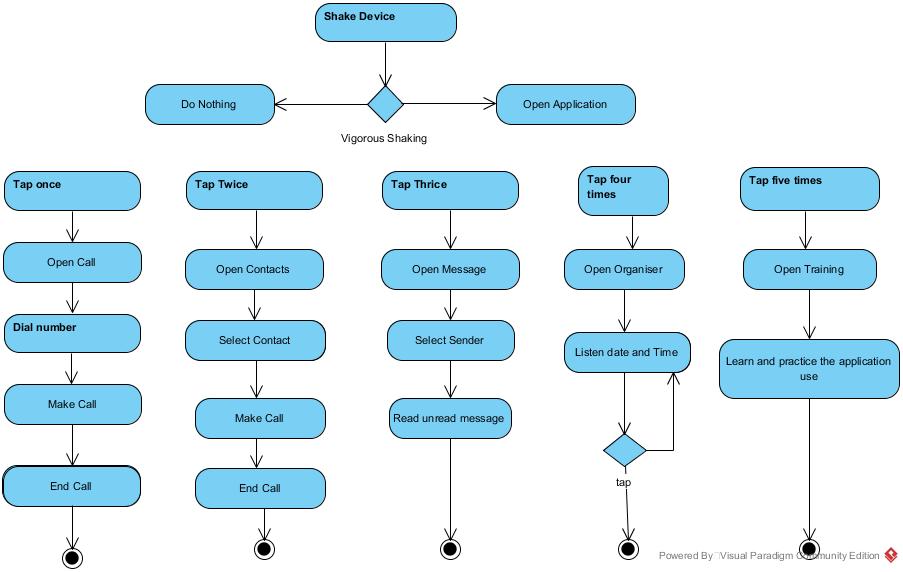


Fig 6.1 Flow Diagram for GVI

# Implementation

This chapter will discuss implementation details of GVI application.



## Algorithm

The APIs used currently in the early stage of project implementation includes four main APIs of android which are Sensor API, Vibrator API, Widgets API and TTS (Text-to-Speech) API. Whereas for the algorithms to be used in the project implementation, a sensing algorithm is used. This algorithm serves the purpose of noise removal by using a low pas filter and a high pass filter, and a threshold value against noise.

When device (in unlocked state) is shaken to a level where it matches the threshold values defined in the system, the GVI application is turned on and main screen appears on the device. The user is then provided with a set of instructions for using the basic features (Call, Contacts and Messages) on tap input. According to the number of taps given by the user, the system opens the set module (One tap for Call, two taps for Contacts and three taps for Messages) and its related screen appears on device. The user is given a voice and vibration feedback for the action performed on the system and when new screen appears on device its set of instructions (voice assistance) are spoken to the user. User has options to navigate cursor (for selection) using volume keys in both Contacts and Messaging. The current cursor position is given to the user in form of voice feedback. In Contacts, if the user gives a button long press input and gives a forward gesture (tilt the device in forward direction), call is dialed to the selected contact. However in Messaging, if the user gives a button long press input and gives a forward gesture (tilt the device in forward direction), the system reads the unread message of that selected sender.

On any stage if the device is shaken vigorously, the user is taken to main screen of GVI application. To improve battery stats, the sensors are being registered on long press of the button. This improves the efficiency of the system and avoids taking unnecessary accelerometer (when not required) of the device to save battery.

## External APIs

The major APIs used in the system are in the following table.

|  |  |  |  |
| --- | --- | --- | --- |
| **Name of API** | **Description of API** | **Purpose of usage** | **List down the function/class name in which it is used** |
| SensorManager | Accelerometer readings | To determine the motion of mobile phone. | MyService, ShakeDevice, Contacts, Messaging, Training. |
| TextToSpeech | Converts given text into Speech | To provide voice assistance | All classes except AutoStart and MyService. |
| Vibrator | Vibrates the device for specific (set) time | To provide haptic feedback | All classes except AutoStart and Organiser. |

## User Interface

GVI Application has six screens in total which are shown below. The first screen (Home) appears when the application launches on user shake of the device in unlock state.

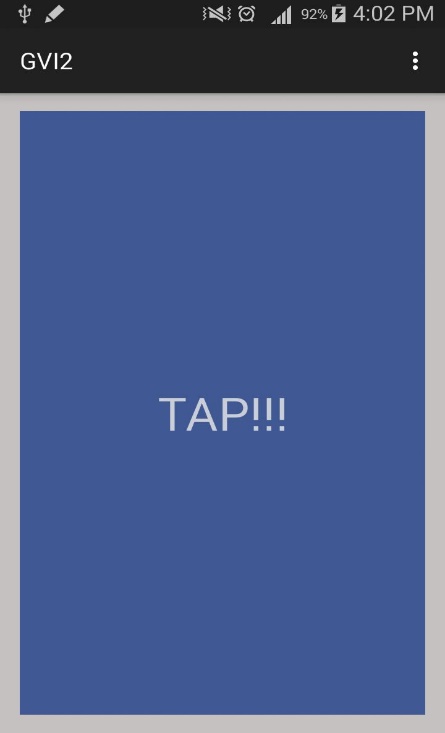


Fig 5.3.1 Home Screen of GVI Application

The next screen is the interface for Call which appears when the user taps one times(s) on the Home screen. It has large buttons spread out on screen which provide a sound feedback on tap (telling which digit is about to be selected) and selection on long press of that digit.

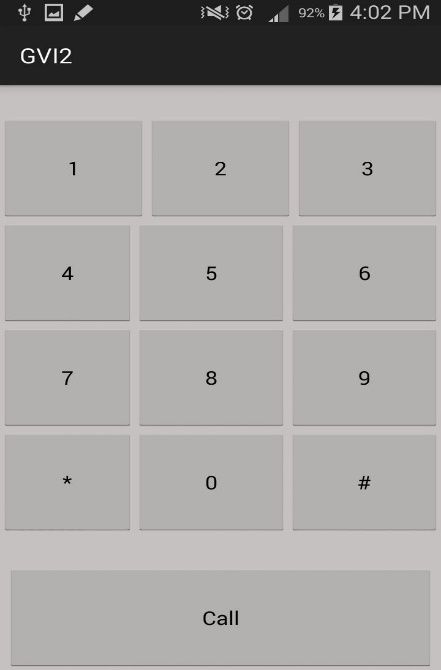


Fig 5.3.2 Call Screen of GVI Application

The next screen is the interface for Contacts which appears when the user taps two times(s) on the Home screen. It has a large button for user ease.

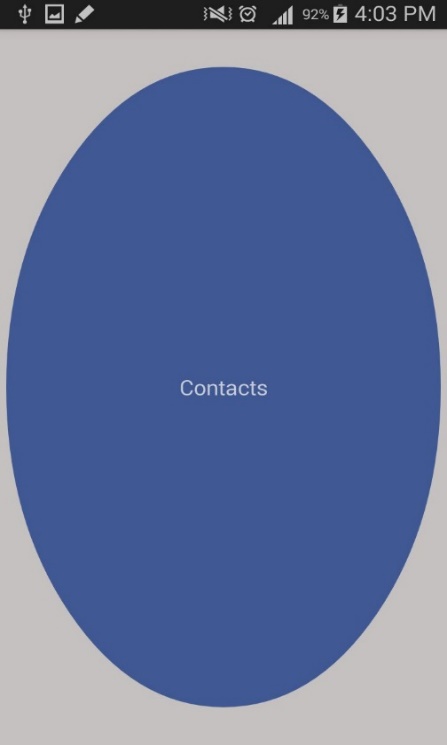


Fig 5.3.3 Contacts Screen of GVI Application

The next screen is the interface for Messages which appears when the user taps three times(s) on the Home screen. It has a large button for user ease.

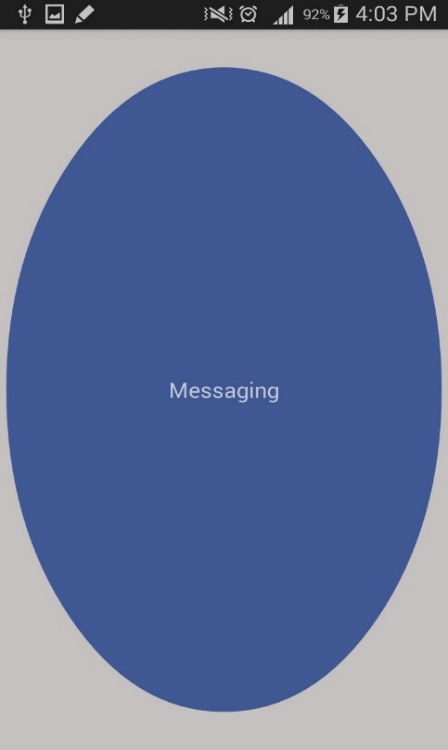


Fig 5.3.4 Home Screen of GVI Application

The next screen is the interface for Organiser which appears when the user taps four times(s) on the Home screen. It has a large button for user ease.

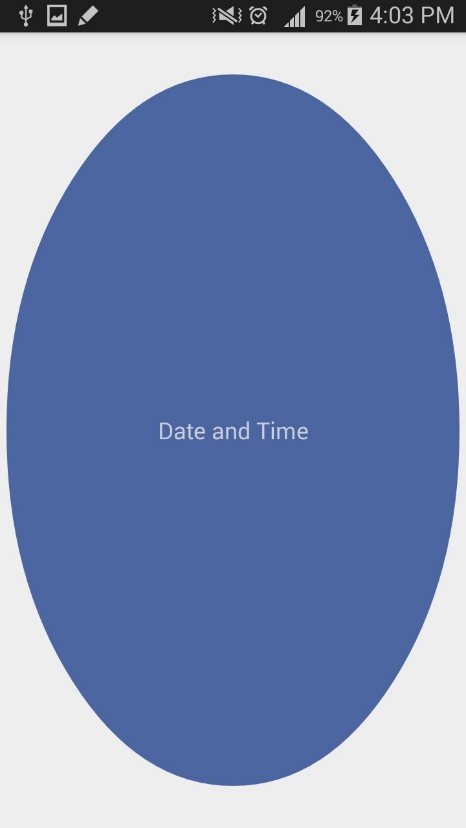


Fig 5.3.5 Organiser Screen of GVI Application

The last screen is the interface for Training section which appears when the user taps five times(s) on the Home screen. It has a large button for user ease.



Fig 5.3.6 Training Screen of GVI Application

# Testing and Evaluation



## Selected testing methodology

The selected testing methodology used is manual use-case based and requirement based testing as required by our user (visually impaired). Unit testing for every module was carried out and then testing of the whole system was performed to check the correctness of the module after integration. Thorough testing was carried out after the completion of the system.

## Requirement/Use-case based testing

This system involves heavy user interaction which is done on the bases of use cases.

### TC- 1.2.1: Start the Application

|  |  |  |
| --- | --- | --- |
| Test Case ID | TC-1.2.1 | |
| Test Case Name | Start the Application (GVI) | |
| Test Data | Vigorous device shake. | |
| Pre-Condition | Application exists in the mobile phone. | |
| Actions | | System Response |
| User shakes the mobile device in unlocked screen. | | The application starts when the default threshold values are matched with the user input gesture. |
| Result | | **Pass.** Main screen appears on the device and a voice and vibration feedback is given. |

**Test verification:** This test verifies that the user can start the application by shaking the device to a certain threshold.

**Testing Environment:** Android studio and Mobile phone.

**Tested by:** Dr. Yasir Faheem, Ahmad Ali and Awais Ahmad.

**Date:** March 29th, 2016

### TC- 1.2.2: Tap Input

|  |  |  |
| --- | --- | --- |
| Test Case ID | TC-1.2.2 | |
| Test Case Name | Tap Input | |
| Test Data | Tapping screen button with one finger. | |
| Pre-Condition | The application should be successfully turned on and should currently open the main screen. | |
| Actions | | System Response |
| The user taps on main screen to shift to the 3 available options of Dialing Call, Opening Contacts and Opening Messages. | | The system responses by shifting to the set screen and gives haptic feedback and voice feedback. Tapping once opens call, tapping twice opens Contacts and tapping three times takes you to Messages. |
| Result | | **Pass.** The system is shifted to the desired screen if the user taps are one two or three. On more than three taps the system shifts to Messages. |

**Test verification:** This test verifies that the user can shift to the desired module on number of taps.

**Testing Environment:** Android studio and Mobile phone.

**Tested by:** Ahmad Ali and Awais Ahmad.

**Date:** March 29th, 2016

### TC- 1.2.3: Button Long press

|  |  |  |
| --- | --- | --- |
| Test Case ID | TC-1.2.3 | |
| Test Case Name | Button Long press | |
| Test Data | Press and hold screen button with one finger. | |
| Pre-Condition | The user has tapped twice and the system has currently opened the Contacts screen on device. | |
| Actions | | System Response |
| When Contacts are opened, If the user long presses the Button, sensors are registered. | | The system will toast a message telling the tester that button has been long pressed so that the tester is notified that sensors are registered and are ready to take accelerometer readings. |
| Result | | **Pass.** The Accelerometer readings (x, y and z values) are stored in the declared variables for comparison with defined threshold values for gesture recognition. |

**Test verification:** This test verifies that the Mobile Sensors are being registered correctly and storing the x, y and z values of the accelerometer.

**Testing Environment:** Android studio and Mobile phone.

**Tested by:** Ahmad Ali and Awais Ahmad.

**Date:** March 29th, 2016

### TC- 1.2.4: Giving Forward Gesture

|  |  |  |
| --- | --- | --- |
| Test Case ID | TC-1.2.4 | |
| Test Case Name | Giving Forward Gesture. | |
| Test Data | Tilting the device in the forward direction (moving device top downwards). | |
| Pre-Condition | 1. The desired screen (Messaging or Contacts currently) is opened. 2. The Button on screen is long pressed. | |
| Actions | | System Response |
| Forward Gesture acts as a selection button. In case of Contacts, forward gesture is used to dial a call and in messaging, it is used to read the new unread message from the sender. | | In case of Contacts, forward gesture will dial a call to the selected contact. However in messaging, it will read the new unread message from the sender. Voice feedback will be provided for assistance |
| Result | | **Pass.** Selected number will be dialed or unread message will be spoken out to the user. |

**Test verification:** This test verifies that the Visually Impaired user can make selection easily.

**Testing Environment:** Android studio and Mobile phone.

**Tested by:** Ahmad Ali and Awais Ahmad.

**Date:** March 29th, 2016

### TC- 1.2.5: Volume key pressed (up and down)

|  |  |  |
| --- | --- | --- |
| Test Case ID | TC-1.2.5 | |
| Test Case Name | Volume key pressed (up and down) | |
| Test Data | Volume key pressed (up or down) | |
| Pre-Condition | 1. The desired screen (call or message is opened) 2. Input is given when the system requests (voice assistance). | |
| Actions | | System Response |
| Pressing the volume buttons on the device helps user to navigate between contacts or unread messages. | | The system will provide voice feedback to tell the user where the cursor is currently pointing. |
| Result | | **Pass.** The user can call the selected number or read the new message. |

**Test verification:** This test verifies that each user can traverse between the created list easily.

**Testing Environment:** Android studio and Mobile phone.

**Tested by:** Ahmad Ali and Awais Ahmad.

**Date:** March 29th, 2016

# Conclusion and Future Work

This chapter concludes the project and highlights future work.



## Conclusion

GVI application is a need base application which is developed to facilitate the Visually Impaired users to use the functionality of their phones with ease, which was lacking previously. This system provides a complete robust basic functionality of Call (new number), Message (reading), Contacts (calling) and Organiser (speaking date and time) with a training section. There is no complicated hardware involvement or constraints like internet availability to use the system. It is a free application which can be used easily.

## Future Work

For Future we plan to add further modules on user feedback from the current system. Furthermore we will write a research paper and place the application on Google Playstore.

# References

* 1. Marco Klingmann, (2009), Accelerometer-Based Gesture Recognition with the iPhone, Goldsmiths University of London.
  2. Creative Commons Attributions, <http://developer.android.com/develop/index.html>, Index, April 22,2016.