

SAVITRIBAI PHULE PUNE UNIVESITY

A PROJECT REPORT ON

ALERTO

**SUBMITTED TOWARDS THE PARTIAL FULFILLMENT OF THE
REQUIREMENTS OF**

BACHELOR OF ENGINEERING(Computer Engineering)

BY

Mihika Deshpande 403005

Tarini Parihar 403010

Ruchi Jain 403014

Zindagi 403029

SPONSORED BY

RAJA SOFTWARE LABS PVT.LTD.

UNDER THE GUIDANCE OF

Prof. S. R. VIJ(Internal Guide)

Mr. PRAKHAR AJABE (External Guide)



**Department Of Computer Engineering
MAEERs MAHARASHTRA INSTITUTE OF TECHNOLOGY
Kothrud, Pune 411 038
2015-2016**

MAHARASHTRA ACADEMY OF ENGINEERING AND EDUCATIONAL
RESEARCH'S
MAHARASHTRA INSTITUTE OF TECHNOLOGY
PUNE
DEPARTMENT OF COMPUTER ENGINEERING

C E R T I F I C A T E

This is to certify that

ALERTO

Submitted by

Mihika Deshpande 403005

Tarini Parihar 403010

Ruchi Jain 403014

Zindagi 403029

is a bonafide work carried out by Students under the supervision of Prof. S. R. Vij
and it is submitted towards the partial fulfillment of the requirement of Bachelor of
Engineering (Computer Engineering).

Prof. S. R. Vij

Dr. V. Y. Kulkarni

Internal Guide

H.O.D.

Dept. of Computer Engg.

Dept. of Computer Engg.

Dr. L. K. Kshirsagar

Principal

Maharashtra Institute of Technology

Signature of Internal Examiner

Signature of External Examiner

PROJECT APPROVAL SHEET

A Project title

ALERTO

Is successfully completed by

Mihika Deshpande 403005

Tarini Parihar 403010

Ruchi Jain 403014

Zindagi 403029

at

DEPARTMENT OF COMPUTER ENGINEERING

MAHARASHTRA INSTITUTE OF TECHNOLOGY

SAVITRIBAI PHULE PUNE UNIVERSITY,PUNE

ACADEMIC YEAR 2015-2016

Prof. S. R. Vij

Dr. V. Y. Kulkarni

Internal Guide

H.O.D.

Dept. of Computer Engg.

Dept. of Computer Engg.

Abstract

The proposed application targets at implementing a system which will enable the end-users to make their life easy. We are using technologies like wearable device, Bluetooth LE and Android SDK. The system will help the users to get rid of the constant urge to check their phone instead it will notify them whenever an important call or message is coming on their phone

We are using Bluetooth low energy instead of bluetooth to save the battery which is an important issue in Android phones. We have a device called "ALERTO" which needs to be synced to our application so that it can alert the user.

Keywords:

Bluetooth LE, Wearable Device, Alerto.

ACKNOWLEDGEMENT

It gives us great pleasure in presenting the preliminary project report on ‘ALERTO’.

We would like to take this opportunity to thank our internal guide Prof. S. R. Vij for giving us all the help and guidance we needed. We are really grateful for her kind support. Her valuable suggestions were very helpful.

We are also grateful to Dr. V. Y. Kulkarni, Head of Computer Engineering Department, Maharashtra Institute of Technology, Pune for her indispensable support, suggestions.

In the end our special thanks to Mr. Prakhar Ajabe, our external guide for providing constant support and guidance for our Project. We are grateful to him for giving his valuable time and knowledge.

Mihika Deshpande

Tarini Parihar

Ruchi Jain

Zindagi

(B.E. Computer Engineering)

Contents

1	SYNOPSIS	1
1.1	PROJECT TITLE	2
1.2	PROJECT OPTION	2
1.3	INTERNAL GUIDE	2
1.4	SPONSORSHIP AND EXTERNAL GUIDE	2
1.5	TECHNICAL KEYWORDS	2
1.6	PROBLEM STATEMENT	3
1.7	ABSTRACT	3
1.8	GOALS AND OBJECTIVES	3
1.9	RELEVANT MATHEMATICS ASSOCIATED WITH THE PROJECT	4
1.9.1	MATHEMATICAL MODEL	4
1.10	NAMES OF CONFERENCES / JOURNALS WHERE PAPERS CAN BE PUBLISHED	6
1.11	PLAN OF PROJECT EXECUTION	6
2	TECHNICAL KEYWORDS	8
2.1	AREA OF PROJECT	9
2.2	TECHNICAL KEYWORDS	9
3	INTRODUCTION	10
3.1	PROJECT IDEA	11
3.2	MOTIVATION OF THE PROJECT	11

3.3	LITERATURE SURVEY	11
4	PROBLEM DEFINITION AND SCOPE	15
4.1	PROBLEM STATEMENT	16
4.1.1	Goals and Objectives	16
4.1.2	Statement of Scope	16
4.2	MAJOR CONSTRAINTS	17
4.3	METHODOLOGIES OF PROBLEM SOLVING AND EFFICIENCY ISSUES	17
4.4	OUTCOME	18
4.5	HARDWARE RESOURCES REQUIRED	18
4.6	SOFTWARE RESOURCES REQUIRED	18
5	PROJECT PLAN	19
5.1	PROJECT ESTIMATES	20
5.1.1	Reconciled Estimates	22
5.1.2	Project Resources	22
5.2	RISK MANAGEMENT W.R.T. NP HARD ANALYSIS	23
5.2.1	Risk Identification	23
5.2.2	Risk Analysis	24
5.2.3	Overview of Risk Mitigation, Monitoring, Management	25
5.3	PROJECT SCHEDULE	26
5.3.1	Project task set	26
5.3.2	Task network	27
5.3.3	Timeline Chart	28
5.4	Team Organization	29
5.4.1	Team structure	29
5.4.2	Management reporting and communication	29

6	SOFTWARE REQUIREMENT SPECIFICATION	30
6.1	INTRODUCTION	31
6.1.1	Purpose and Scope of Document	33
6.1.2	Overview of responsibilities of Developer	33
6.2	USAGE SCENARIOS	33
6.2.1	User profiles	34
6.2.2	Use cases	34
6.2.3	Use case view	35
6.3	FUNCTIONAL MODEL AND DESCRIPTION	36
6.3.1	Data Flow Diagram	37
6.3.2	Activity Diagram	38
6.3.3	Non Functional Requirements	39
6.3.4	State Diagram	43
6.3.5	Design Constraints	43
6.3.6	Software Interface Description	43
7	DETAILED DESIGN DOCUMENT USING APPENDIX A AND	
B		44
7.1	INTRODUCTION	45
7.1.1	Purpose	45
7.1.2	Scope	45
7.1.3	Definitions, Acronyms, Abbreviations	45
7.2	ARCHITECTURAL DESIGN	46
7.2.1	Bluetooth Low Energy:	46
7.2.2	Bluetooth GATT(Software Model):	49
7.2.3	Services:	52
7.2.4	Activities :	53

7.2.5	Telephony Manager :	54
7.2.6	Broadcast Manager:	54
7.3	COMPONENT DESIGN	55
7.3.1	Class Diagram	55
8	PROJECT IMPLEMENTATION	56
8.1	INTRODUCTION	57
8.2	TOOLS AND TECHNOLOGIES USED	57
8.2.1	HARDWARE RESOURCES	57
8.2.2	SOFTWARE RESOURCES	57
8.2.3	TECHNOLOGY	57
8.3	METHOLOGIES AND ALGORITHMS DETAILS	58
8.3.1	Algorithm For Bluetooth LE Connection Establishment	58
8.3.2	Algorithm For Execution of the Android App	58
8.4	VERIFICATION AND VALIDATION FOR ACCEPTANCE	58
9	SOFTWARE TESTING	60
9.1	TYPES OF TESTING USED	61
9.2	TEST CASES AND TEST RESULTS	62
10	RESULTS	66
10.1	SCREEN SHOTS	67
11	DEPLOYMENT AND MAINTENANCE	68
11.1	INSTALLATION AND UN-INSTALLATION	69
11.1.1	Installation	69
11.1.2	Un-installation	69
11.2	USER HELP	69

12 FUTURE ENHANCEMENT AND CONCLUSION	70
12.1 FUTURE SCOPE	71
12.2 CONCLUSION	71
13 REFERENCES	72
BIBLIOGRAPHY	73
Annexure A LABORATORY ASSIGNMENTS ON PROJECT ANALYSIS OF ALGORITHM DESIGN	74
Annexure B LABORATORY ASSIGNMENTS ON PROJECT QUALITY AND RELIABILITY TESTING OF PROJECT DESIGN	78
Annexure C PROJECT PLANNER	84
Annexure D TERM-II PROJECT LABORATORY ASSIGNMENTS	86
Annexure E INFORMATION OF PROJECT GROUP MEMBERS	90

List of Figures

1.1	Mathematical model	6
5.1	Waterfall model	20
5.2	Task network	27
6.1	Use Case diagram-Connection	35
6.2	Use case diagram-Setting and Favouritism	35
6.3	Use Case diagram-Working Device	35
6.4	Level 0 Data Flow Diagram	37
6.5	Level 1 Data Flow Diagram	38
6.6	Activity diagram-Alerto Application	38
6.7	Activity diagram-User	39
6.8	State Diagram	43
7.1	Communication module using bluetooth LE	46
7.2	Architecture Diagram	52

7.3	Class Diagram	55
B.1	Divide and Conquer Strategy	79
B.2	Functional Dependency Graph	80

List of Tables

1.1	Plan of Project Execution	7
5.1	Risk Table	25
5.2	Risk Probability definitions	25
5.3	Risk Impact definitions	25
5.4	Timeline chart	28
6.1	Use Cases	35
7.1	Bluetooth Vs Bluetooth Low Energy	48
A.1	IDEA Matrix	75
C.1	Plan of Project	85

Chapter 1

SYNOPSIS

1.1 PROJECT TITLE

The title of our project is 'ALERTO'

1.2 PROJECT OPTION

The option of our project is Industry sponsored

1.3 INTERNAL GUIDE

The internal guide of our project is Prof. S. R. Vij

1.4 SPONSORSHIP AND EXTERNAL GUIDE

Our project is sponsored by Raja Software Pvt. Ltd. under the guidance of Mr. Prakhar Ajabe.

1.5 TECHNICAL KEYWORDS

A.Hardware

- Alerto(The wearable device)
- Android phone

B.Architecture

- Client Server Architecture
- Database Connectivity

C.Networking

- Bluetooth LE

D.Network Protocols

- Broadcast
- Send and Receive

1.6 PROBLEM STATEMENT

Development of an android application for a wearable device called ‘Alerto’, that detects notifications for calls, messages and conversations of any android devices using Bluetooth Low Energy technology and alert users using the device.

1.7 ABSTRACT

The proposed application targets at implementing a system which will enable the end-users to make their life easy. We are using technologies like wearable device, Bluetooth LE and Android SDK. The system will help the users to get rid of the constant urge to check their phone instead it will notify them whenever an important call or message is coming on their phone.

We are using Bluetooth low energy instead of bluetooth to save the battery which is an important issue in Android phones. We have a device called ”ALERTO” which needs to be synced to our application so that it can alert the user.

1.8 GOALS AND OBJECTIVES

1. To free the user from the constant urge to check the phone.
2. To provide the user with a user-friendly environment so that user becomes technologically accustomed.
3. To inform the user when the phone is out of reach.
4. To let the user know when a notification from an important contact has arrived.

1.9 RELEVANT MATHEMATICS ASSOCIATED WITH THE PROJECT

1.9.1 MATHEMATICAL MODEL

A mathematical model is a description of system using mathematical concepts and language. A model may help to explain a system and to study the effects of different components, and to make predictions about behaviour.

Mathematical models can take many forms, including but not limited to dynamical systems, statistical models, differential equations, or game theoretic models. These and other types of models can overlap, with a given model involving a variety of abstract structures. In general, mathematical models may include logical models.

In many cases, the quality of a scientific field depends on how well the mathematical models developed on the theoretical side agree with results of repeatable experiments. Lack of agreement between theoretical mathematical models and experimental measurements often leads to important advances as better theories are developed.

Mathematical Model for Android Application ‘Alerto’:

- I: Set of Inputs
- O: Set of outputs
- F: Functions
- Sc: Success cases
- Fc: Failure cases
- $I = \{I1, I2\}$ where,
 - I1: Installation of the Android application Alerto

- I2: Execution of the application
- O: {O1, O2} where,
 - O1: List of available Alertos
 - O2: Connection established with a particular Alerto.
- F: {F1, F2, F3, F4, F5} where,
 - F1: Get started with the application.
 - F2: Turn on Bluetooth.
 - F3: Scan for available Alerto devices in range.
 - F4: Connect with specific Alerto.
 - F5: Once desired notification arrives on smart phone, the particular Alerto vibrates.
- Sc: {Sc1, Sc2} where,
 - Sc1: Connection established.
 - Sc2: Alerto vibrates.
- Fc: {Fc1, Fc2, Fc3, Fc4, Fc5} where,
 - Fc1: Application not working.
 - Fc2: Installation error.
 - Fc3: Alerto devices invisible.
 - Fc4: Connection establishment error.
 - Fc5: Vibration doesn't occur

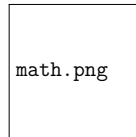


Figure 1.1: Mathematical model

1.10 NAMES OF CONFERENCES / JOURNALS WHERE PAPERS CAN BE PUBLISHED

- IEEE/ACM Conference/Journal 1
- Conferences/workshops in IITs
- Central Universities or SPPU Conferences
- IEEE/ACM Conference/Journal 2

1.11 PLAN OF PROJECT EXECUTION

Table 1.1: Plan of Project Execution

From	To	Task	Status
27-06-2015	30-06-2015	Group Formation and finalization	Done
01-07-2015	15-07-2015	Topic Search	Done
16-07-2015	20-07-2015	Preliminary Information Gathering	Done
21-07-2015	28-07-2015	Project Discussion with Project Coordinator and topic finalization	Done
01-08-2015	04-08-2015	Synopsis preparation and submission	Done
25-08-2015	30-08-2015	Detailed Literature Survey	Done
19-09-2015	24-09-2015	.	.
25-09-2015	06-10-2015	Preparing Interim report	Done
20-12-2015	02-01-2016	Language Study	Done
03-01-2016	20-01-2016	Android and Bluetooth LE Study	Done
21-01-2016	04-03-2016	Coding and Implementation	Done
05-03-2016	21-04-2016	Testing	Done
13-04-2016	21-04-2016	Final Documentation	Done
25-04-2016	20-05-2016	Final Project Report	Done

Chapter 2

TECHNICAL KEYWORDS

2.1 AREA OF PROJECT

The area of our project is Mobile Application.

2.2 TECHNICAL KEYWORDS

A.Hardware

- Alerto(The wearable device)
- Android phone

B.Architecture

- Client Server Architecture
- Database Connectivity

C.Networking

- Bluetooth LE

D.Network Protocols

- Broadcast
- Send and Receive

Chapter 3

INTRODUCTION

3.1 PROJECT IDEA

The proposed application targets at implementing a system which will enable the end-users to make their life easy. We are using technologies like wearable device, Bluetooth LE and Android SDK. The system will help the users to get rid of the constant urge to check their phone instead it will notify them whenever an important call or message is coming on their phone.

3.2 MOTIVATION OF THE PROJECT

Many times a situation has occurred where a call or message was missed while doing some important work and we were unable to check it at that time. Hence we decided to create an application which will help us to avoid this inconvenience.

3.3 LITERATURE SURVEY

1. Modeling Neighbor Discovery in Bluetooth Low Energy Networks

Jia Liu, Member, IEEE, Canfeng Chen, Member, IEEE, and Yan Ma, Member, IEEE COMMUNICATIONS LETTERS, VOL. 16, NO. 9, SEPTEMBER 2012

An analytical model is proposed for 3-channel-based neighbor discovery in Bluetooth Low Energy (BLE) networks. The model can be used to determine some important performance metrics, such as average latency or average energy consumption during the course of discovering neighbors. Since intermittent connections are frequently encountered in practical scenarios of BLE, the modeling results can provide a beneficial guidance to customize advertising or scanning behavior towards user desired performance.

2. Energy Analysis of Device Discovery for Bluetooth Low Energy

Jia Liu, Member, IEEE, Canfeng Chen, Member, IEEE, and Yan Ma,

Member, IEEE

Bluetooth Low Energy (BLE) is drawing more and more attention due to its recent appearance in consumer electronic products. As a low-power wireless solution, BLE provides attractive energy performance that makes it particularly suitable for portable, battery-driven electronic devices. Although there are some prior arts focusing on BLE energy performance, it still lacks a thorough study on the important aspect of device discovery. Such energy cost, introduced by intermittent scanning or connection setup, could seriously affect the battery endurance ability of the devices. In this paper, quantitative analysis on the neighbor discovery energy for BLE is presented. The modeling results that built upon measurement of CC2541 Mini-Development Kit have been validated quite accurate via extensive experiments. In addition, several interesting conclusions are found while investigating the achieved energy model, which may provide precious guidelines to the design of energy-efficient applications for BLE.

3. Advertising semantically described physical items with Bluetooth Low Energy beacons 2nd Mediterranean Conference on Embedded Computing „/” MECD - 2013

Enabling smart applications to access information about the physical world in a machine interpretable format is a high priority in the Internet of Things (IoT) related research. In this paper, we present a novel approach for advertising information related to physical objects in the user vicinity. There are three distinctive features in the approach. First, ubiquitous codes (ucodes) are used for providing globally unique identifiers for the physical objects. Second, using Bluetooth Low Energy beacons for broadcasting the identifiers of the physical objects.

Third, the information related to the physical object is represented with semantic technologies.

4.Wapplet: A Media Access Framework for Wearable Applications

Takeshi Iwamoto¹, Nobuhiko Nishio¹, and Hideyuki Tokuda^{1,2}

1. Graduate School of Media and Governance, Keio University, Japan

iwaiwa, vino, hxt@ht.sfc.keio.ac.jp

2.Faculty of Environmental Information, Keio University, Japan

This paper presents a new framework for constructing applications for wearable computers. We do not consider wearable computing merely as a single self-confined system. Rather, wearable computing can be treated as a cooperative application with information surrounding a person and various devices attached to him/her. We have developed a framework of such wearable computing called Wapplet. Wapplet provides adaptation to the availability of devices as well as a systematic scheme of constructing applications for wearable computing. In this paper, we describe the design and implementation of Wapplet, and show its prototype system.

5.Wearable Electronics Sensors: Current Status and Future Opportunities
Anindya Nag and Subhas Chandra Mukhopadhyay
Massey University, Palmerston North, New Zealand

The technological advancement in the past three decades has impacted our lives and wellbeing significantly. Different aspects of monitoring our physiological parameters are considered. Wearable sensors are one of its most important areas that have an ongoing trend and have a huge tendency to rise in the future. The wearable sensors are the externally used devices attached to any individual to measure physiological parameters of interest. The range of wearable sensors varies from minuscule to large scaled devices physically fitted to the user operating on wired or wireless terms. Many common diseases affecting large number of people notably gait abnormalities, Parkinsons disease are analysed by the wearable sensors. The use of wearable sensors has got a better prospect with improved technical qualities and a better understanding of the currently used research methodologies. This chapter deals with the overview of

the current and past means of wearable sensors with its associated protocols used for communication. It concludes with the ways the currently dealt wearable sensors can be improved in future.

Chapter 4

PROBLEM DEFINITION AND SCOPE

4.1 PROBLEM STATEMENT

Development of an android application for a wearable device called Alerto, that detects notifications for calls, messages and conversations of your android devices using Bluetooth Low Energy technology and alert users using the device.

4.1.1 Goals and Objectives

1. To free the user from the constant urge to check the phone.
2. To provide the user with a user-friendly environment so that user becomes technologically accustomed.
3. To inform the user when the phone is out of reach.
4. To let the user know when a notification from an important contact has arrived.

4.1.2 Statement of Scope

We describe what features are in the scope of the software and what are not in scope of the software to be developed.

1. Notifications for Incoming Calls

The user will feel it vibrate when someone they have chosen as important is trying to reach them.

2. Notifications for Incoming Texts

The user will feel it vibrate when someone they have chosen as important is trying to text them.

3. Notifications for When The User Leave Their Phone Behind

Alerto has a built-in wireless tether. When the user is about to leave their phone behind, Alerto will vibrate to warn them.

4. No Cables, No Buttons, No Recharging

Since Alerto never needs charging, just leave it with other things. Alerto will remind the user when it needs a new battery.

4.2 MAJOR CONSTRAINTS

1. The android version used in the app should be higher than Jelly_bean_MR2(4.3+)
2. The distance between the Alerto device and Smartphone should be within Bluetooth LE range.
3. App requires access to contact details, Bluetooth information, Device ID and call information

4.3 METHODOLOGIES OF PROBLEM SOLVING AND EFFICIENCY ISSUES

A problem can be solved by many different solutions. The methodologies of problem solving considers the performance parameters for each approach. Thus considering the efficiency issues.

1. Our project is based on Android operating system but it can be implemented on other operating systems as well. However we have selected Android because it is much more popular in comparison to other OS which increases the usability of our application.
2. Bluetooth Low energy uses less amount of battery power as compared to classic Bluetooth devices. Hence using this technology helps to reduce power consumption and increase battery life. Thus we have used this technology for our application

4.4 OUTCOME

We have implemented an application in Android with a wearable device which alerts the users about incoming notifications.

4.5 HARDWARE RESOURCES REQUIRED

1. 400MB Hard disk+1GB for Android SDK, emulator system images and caches
2. Alerto device(Wearable)
3. 2GB RAM minimum,4GB RAM recommended
4. Intel Processor with support for Intel VT-x, Intel EM64T(Intel 64)
5. Computer with windows Vista/7/8.

4.6 SOFTWARE RESOURCES REQUIRED

- Platform: Android SDK
- Operating System: Android
- Programming Language: Java, XML.

Chapter 5

PROJECT PLAN

5.1 PROJECT ESTIMATES

In "The Waterfall" approach, the whole process of software development is divided into separate phases. In Waterfall model, typically, the outcome of one phase acts as the input for the next phase sequentially.

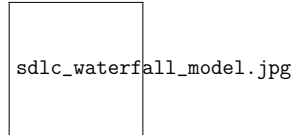


Figure 5.1: Waterfall model

The sequential phases in Waterfall model are:

- **Requirement Gathering and analysis:**

All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification document.

- **Hardware Requirements:**

- * 400MB Hard disk+1GB for Android SDK, emulator system images and caches
 - * Alerto device(Wearable) 2GB RAM minimum,4GB RAM recommended
 - * Intel Processor with support for Intel VT-x, Intel EM64T(Intel 64)
 - * Computer with windows Vista/7/8.

- **Software Requirements:**

- * Platform: Android SDK
 - * Operating System: Android
 - * Programming Language: Java, XML.

- **System Design:** The requirement specifications from first phase are studied in this phase and system design is prepared. System Design helps in specifying

hardware and system requirements and also helps in defining overall system architecture.

- The App will be developed using Android SDK
 - The app is created using Android/Java
- **Implementation:** With inputs from system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality which is referred to as Unit Testing. The following units were serially developed:
 1. Connection module
 2. Vibration module
 3. Telephony Manager
 4. Broadcast Receiver
- **Integration and Testing:** All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.
- **Deployment of system:** Once the functional and non functional testing is done, the product is deployed in the customer environment or released into the market. User can have the app installed on their android phones.
- **Maintenance:** There are some issues which come up in the client environment. To fix those issues patches are released. Also to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment.

Here, if needed by the developer, he can extend the application usage to third party apps.

Waterfall Model Application

Every software developed is different and requires a suitable SDLC approach to be followed based on the internal and external factors. Some situations where the use of Waterfall model is most appropriate are:

- Requirements are very well documented, clear and fixed.
- Product definition is stable.
- Technology is understood and is not dynamic.
- There are no ambiguous requirements.
- Ample resources with required expertise are available to support the product.
- The project is short

5.1.1 Reconciled Estimates

- **Cost Estimate**

Cost Estimate: Free of cost.

- **Time Estimate**

Time Estimates: 8 months

5.1.2 Project Resources

- **Hardware Resources:**

- 400MB Hard disk+1GB for Android SDK, emulator system images and caches
- Alerto device(Wearable)
- 2GB RAM minimum,4GB RAM recommended
- Intel Processor with support for Intel VT-x, Intel EM64T(Intel 64)

- Computer with windows Vista/7/8

- **Software Resources:**

- Platform: Android SDK
- Operating System: Android
- Programming Language: Java, XML.

- **Human Resources:**

The team includes four members who are involved in making the project. Our external and internal guide helped to solve our doubts and remove the errors in the project.

5.2 RISK MANAGEMENT W.R.T. NP HARD ANALYSIS

5.2.1 Risk Identification

Risk identification is the process of determining risks that could potentially prevent the program, enterprise, or investment from achieving its objectives. It includes documenting and communicating the concern.

For risks identification, review of scope document, requirements specifications and schedule is done. Answers to questionnaire revealed some risks. Each risk is categorized as per the categories.

The following questions are asked for identification of various categorical risks:

1. Have top software and customer managers formally committed to support the project?
2. Are end-users enthusiastically committed to the project and the system/product to be built?

3. Are requirements fully understood by the software engineering team and its customers?
4. Have customers been involved fully in the definition of requirements?
5. Do end-users have realistic expectations?
6. Does the software engineering team have the right mix of skills?
7. Are project requirements stable?
8. Is the number of people on the project team adequate to do the job?
9. Do all customer/user constituencies agree on the importance of the project and on the requirements for the system/product to be built?

5.2.2 Risk Analysis

The risks for the Project can be analyzed within the constraints of time and quality

ID	Risk Description	Probability	Impact	
			Quality	Overall
1	Top software and customer managers formally committed to support the project	High	Low	Low
2	End-users enthusiastically committed to the project and the system/product to be built	High	Low	Low
3	Requirements fully understood by the software engineering team and its customers	High	Low	Low
4	Customers been involved fully in the definition of requirements	High	Low	Low
5	End-users' realistic expectations	High	Low	Low
6	The software engineering team have the right mix of skills	High	Low	Low
7	Project requirements stability	High	Low	Low
8	Number of people on the project team adequate to do the job	High	Low	Low
9	All customer/user constituencies agree on the importance of the project and on the requirements for the system/product to be built	High	Low	Low

Table 5.1: Risk Table

Probability	Value	Description
High	Probability of occurrence is	> 75%
Medium	Probability of occurrence is	26 – 75%
Low	Probability of occurrence is	< 25%

Table 5.2: Risk Probability definitions

Impact	Value	Description
Very high	> 10%	Schedule impact or Unacceptable quality
High	5 – 10%	Schedule impact or Some parts of the project have low quality
Medium	< 5%	Schedule impact or Barely noticeable degradation in quality Low Impact on schedule or Quality can be incorporated

Table 5.3: Risk Impact definitions

5.2.3 Overview of Risk Mitigation, Monitoring, Management

Following are the details for each risk.

Risk ID	1
Risk Description	Top software and customer managers formally committed to support the project
Category	Development Environment.
Source	Platform: Android SDK, Operating System: Android, Programming Language: Java, XML
Probability	High
Impact	Low
Response	Aggravate
Strategy	Discussion of plan & steps for implementation & improvement
Risk Status	Not occurred

Risk ID	3
Risk Description	Requirements fully understood by the software engineering team and its customers
Category	Requirements
Source	Platform: Android SDK, Operating System: Android, Programming Language: Java, XML
Probability	High
Impact	Low
Response	Aggravate
Strategy	Proceed with the implementation
Risk Status	Identified

Risk ID	5
Risk Description	End-users' realistic expectations
Category	Customers' Requirements
Source	This was identified during early development and testing.
Probability	High
Impact	Low
Response	Accept
Strategy	Advance as per users' need & make necessary changes
Risk Status	Identified

5.3 PROJECT SCHEDULE

5.3.1 Project task set

Major Tasks in the Project stages are:

- Task 1: Selecting the project domain & the project.

- Task 2: Checking for duplicity & research work.
- Task 3: Getting the company's sponsorship.
- Task 4: Communication with the internal & external guide.
- Task 5: Project Planning.
- Task 6: Study of tools & technology- Android Studio, Bluetooth LE.
- Task 7: Design Phase.
- Task 8: Implementation & Coding.
- Task 9: Testing.
- Task 10: Deployment.
- Task 11: Final Presentation.

5.3.2 Task network

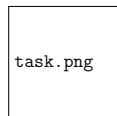


Figure 5.2: Task network

5.3.3 Timeline Chart

See Table below

Table 5.4: Timeline chart

From	To	Task	Status
27-06-2015	30-06-2015	Group Formation and finalization	Done
01-07-2015	15-07-2015	Topic Search	Done
16-07-2015	20-07-2015	Preliminary Information Gathering	Done
21-07-2015	28-07-2015	Project Discussion with Project Coordinator and topic finalization	Done
01-08-2015	04-08-2015	Synopsis preparation and submission	Done
25-08-2015	30-08-2015	Detailed Literature Survey	Done
19-09-2015	24-09-2015	.	.
25-09-2015	06-10-2015	Preparing Interim report	Done
20-12-2015	02-01-2016	Language Study	Done
03-01-2016	20-01-2016	Android and Bluetooth LE Study	Done
21-01-2016	04-03-2016	Coding and Implementation	Done
05-03-2016	21-04-2016	Testing	Done
13-04-2016	21-04-2016	Final Documentation	Done
25-04-2016	20-05-2016	Final Project Report	Done

5.4 Team Organization

The team structure is a newer, less hierarchical organizational structure in which individuals are grouped into teams.

A team should be a group of workers, with complementary skills and synergistic efforts, all working toward a common goal. An organization may have several teams that can change over time. Teams that include members from different functions are known as cross-functional teams.

5.4.1 Team structure

We have completed the journal, report and partial implementation of our project. The project was equally divided and distributed among the team members. Under the guidance of our external guide, we have implemented various adaptations in our project.

5.4.2 Management reporting and communication

Emails, phone calls and messages were used to communicate within the team. The team met in person as well to do different tasks of the project. We also communicated with our external guide via emails and messages. In cases where we required assistance in our project. We met our guide who helped to remove our difficulties and solve our doubts.

Chapter 6

SOFTWARE REQUIREMENT SPECIFICATION

6.1 INTRODUCTION

Alerto is a kind of anti-gadget, something to free people from worrying about their smart phones and to be more present in life. We are developing an android application that will be interacting with these Alertos. The application would be doing the following activities:

- Notifications for Incoming Calls :

The user will feel it vibrate when someone they have chosen as important is trying to reach them.

- Notifications for Incoming Texts :

The user will feel it vibrate when someone they have chosen as important is trying to text them.

- Notifications for Third Party Apps :

Use a third party chat service? Want to stay on top of your social media notifications? You can customize Alerto to alert you about notifications on a number of popular apps including WhatsApp, Hangouts.

People often do not check the messages and sometimes while they cannot access their phone, they cannot know if some important calls or messages have arrived on their phone if their phone is not nearby, so with our application we are trying to solve this problem.

We already have something complex as smartphones. There are times when we are not nearby our phones or we cannot receive them and we miss some calls and messages. That's where when our application and Alerto comes into existence. Too much technology leads to complicated and expensive products. Hence, we kept a simple approach for such a development. The idea is that you can wear Alerto discretely clipped to your waistband, perhaps and get alerts for things that really matter. Our

app lets you choose which apps or contacts make Alerto buzz, and assign a distinct vibration pattern to each one. Alerto can also remind you that you have left your phone behind, buzzing as it falls out of Bluetooth range.

It has a clever clipping mechanism built in, with a slight raised edge that you press on to open the clip. The battery is a standard coin shape that needs replacement every four to six months.

We are using Bluetooth Low energy Technology for communication in our application . We chose Bluetooth LE over Bluetooth because it has certain advantages ,which are:

- Bluetooth can handle a lot of data, but consumes battery life quickly and costs a lot more. BLE is used for applications that do not need to exchange large amounts of data, and can therefore run on battery power for years at a cheaper cost.
- Since in our project we have minimum amount of data exchange and we need more battery power because it consumes battery while running in the background,we chose Bluetooth Low Energy technology.

Our application is able to run on android devices (in particular), but here we are implementing it on an emulator.The application can be installed by any android phone but is fully functional only when a Alerto device is discovered by the application.

Getting started with the application,access to Bluetooth is asked. Once enabled list of Alerto devices are made available as per the requirement,certain connection between the device and emulator is established. Herewith, the application runs in the background and as and when,any desired notification arrives,the user wearing the

Alerto gets the vibration.

It can function as a tracking device. But in doing such, it doesnt interfere with our privacy boundaries. We value privacy, denying access to their location. We dont even work with any sort of data, so theres no issue of any valuable information leakage.

Alerto is completely intuitive to use no learning required. The app is just used for setting. Cool tools should make life easier, not more difficult. Alertos companion app lets you choose which apps and contacts will trigger a buzz. Our app is straight-forward enough, letting you tap on each listed app or contact to set up their vibration patterns.

6.1.1 Purpose and Scope of Document

- The reader can understand basic functionality of the project
- Also he can understand the working and architectural setup of the app so that he can use the app efficiently

6.1.2 Overview of responsibilities of Developer

- Coding
- Implementation
- Testing

6.2 USAGE SCENARIOS

A usage scenario, or scenario for short, describes a real-world example of how one or more people or organizations interact with a system. They describe the steps, events, and/or actions which occur during the interaction. Usage scenarios can be

very detailed, indicating exactly how someone works with the user interface, or reasonably high-level describing the critical business actions but not indicating how they're performed. The basic strategy is to identify a path through a use case, or through a portion of a use case, and then write the scenario as an instance of that path.

Scenario:

1. Go to the app.
2. Connect.
3. Keep the Alerto with you.
4. Whenever a call or message arrives, the device with you will vibrate.

6.2.1 User profiles

- Developer :
 - They create the app and analyse customer feedbacks.
 - They should be knowledgeable about end users.
- User :
 - User should have the device with them at all times
 - They should be knowledgeable about the app.

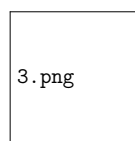
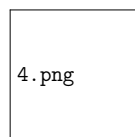
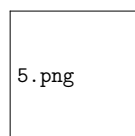
6.2.2 Use cases

Sr. No.	Use Cases	Description	Actors	Assumptions
1	Use case 1	Connecting the Alerto with the android device	User, Alerto	Bluetooth is switched on
2	Use Case 2	Setting and Favouritism	User, Alerto	App functioning properly
3	Use Case 3	Working of Device	User, Android Device, Alerto	Successful connection ios established between the app & Alerto

Table 6.1: Use Cases

6.2.3 Use case view

Use case diagram shows the overall functionality of the system. In our system there are three use case diagrams.

**Figure 6.1:** Use Case diagram-Connection**Figure 6.2:** Use case diagram-Setting and Favouritism**Figure 6.3:** Use Case diagram-Working Device

6.3 FUNCTIONAL MODEL AND DESCRIPTION

1. **BluetoothAdapter**

The BluetoothAdapter is required for any and all Bluetooth activity. The BluetoothAdapter represents the device's own Bluetooth adapter (the Bluetooth radio). There's one Bluetooth adapter for the entire system, and your application can interact with it using this object.

The first step in interacting with a BLE device is connecting to it more specifically, connecting to the GATT server on the device. To connect to a GATT server on a BLE device, you use the `connectGatt()` method. This method takes three parameters: a Context object, `autoConnect` (boolean indicating whether to automatically connect to the BLE device as soon as it becomes available), and a reference to a `BluetoothGattCallback`

2. **onServicesDiscovered**

This function is called when the connection is established between the android phone and the Alerto device so as to know what further step is to be taken after connection.

3. **onCharacteristicRead**

The Alerto provides a ton of services which are read by the phone and services have various characteristics. The above function is used when the service and the characteristics of those services are explored and further step is to be decided.

4. Telephony manager

Provides access to information about the telephony services on the device. Applications can use the methods in this class to determine telephony services and states, as well as to access some types of subscriber information. Applications can also register a listener to receive notification of telephony state changes. You

do not instantiate this class directly; instead, you retrieve a reference to an instance through `Context.getSystemService(Context.TELEPHONY_SERVICE)`.

6.3.1 Data Flow Diagram

- Level 0 Data Flow Diagram

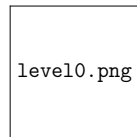


Figure 6.4: Level 0 Data Flow Diagram

- **Level 1 Data Flow Diagram**

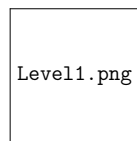


Figure 6.5: Level 1 Data Flow Diagram

6.3.2 Activity Diagram

In activity diagram the overall flow of the system is shown. It contains the following sequence of action.

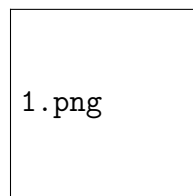


Figure 6.6: Activity diagram-Alerto Application

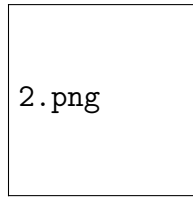


Figure 6.7: Activity diagram-User

6.3.3 Non Functional Requirements

- **Interface Requirements:**

Your app's user interface is everything that the user can see and interact with. Android provides a variety of pre-built UI components such as structured layout objects and UI controls that allow you to build the graphical user interface for your app. Android also provides other UI modules for special interfaces such as dialogs, notifications, and menus.

- **Performance Requirements:**

- Choosing the right algorithms and data structures should always be the priority.
- do not allocate memory if you can avoid it
- To ensure your app performs well across a wide variety of devices, ensure your code is efficient at all levels and aggressively optimize your performance.

- **Avoid Creating Unnecessary Objects**

Object creation is never free. A generational garbage collector with per-thread allocation pools for temporary objects can make allocation cheaper, but allocating memory is always more expensive than not allocating memory.

- **Prefer Static Over Virtual**

If you do not need to access an object's fields, make your method static. Invocations will be about 15%-20% faster. It's also good practice, because you can tell from the method signature that calling the method can't alter the object's state.

- **Consider Package Instead of Private Access with Private Inner Classes**

- **Software quality attributes:**

- Availability

- * It is the proportion of time a system is in a functioning condition.
 - * Availability of a system is typically measured as a factor of its reliability as reliability increases, so does availability.
 - * Reliability needs to be evaluated and improved related to both availability and the cost of ownership (due to cost of spare parts, maintenance man-hours, transport costs, storage cost, part obsolete risks etc.).
 - * Fault tree analysis and related software are developed to calculate (analytic or by simulation) availability of a system or a functional failure condition within a system.
 - * In our case, our application is constantly available (running in the background).

- Modifiability

- * Portability

- Portability in high-level computer programming is the usability of the same software in different environments. The prerequisite

for portability is the generalized abstraction between the application logic and system interfaces.

- Our application is not portable as it is supported only on android devices.

* Reusability

- In computer science and software engineering, reusability is the use of existing assets in some form within the software product development process. Assets are products and by-products of the software development life cycle and include code, software components, test suites, designs and documentation.
- Subroutines or functions are the simplest form of reuse. A chunk of code is regularly organized using modules or namespaces into layers.
- The vibration module in our code has been re-used for various types of incoming notifications.

* Scalability

- An algorithm, design, networking protocol, program, or other system is said to scale if it is suitably efficient and practical when applied to large situations (e.g. a large input data set, a large number of outputs or users, or a large number of participating nodes in the case of a distributed system).
- We can scale our application to third party app notifications, such as WhatsApp, Emails, etc.

– Testability

- * Software testability is the degree to which a software artifact (i.e. a software system, software module, requirements- or design document)

supports testing in a given test context.

- * It is an extrinsic property which results from interdependency of the software to be tested and the test goals, test methods used, and test resources (i.e., the test context).
- * Our app is being tested for the following things:
 - Connection of the Alerto with the android device.
 - If the app is running even in the background.
 - Vibration happening for following situations:
 1. Arrival of notifications;
 2. Device going out of BLE range

Note: Detailed test cases have been provided in section 9.2.

– Usability

- * In Software engineering, usability is the degree to which a software can be used by specified consumers to achieve quantified objectives with effectiveness, efficiency, and satisfaction in a quantified context of use.
- * It includes methods of measuring usability, such as needs analysis and the study of the principles behind an object's perceived efficiency or elegance.
- * The primary notion of usability is:
 1. More efficient to use, takes less time to accomplish a particular task
 2. Easier to learn operation can be learned by observing the object
 3. More satisfying to use

6.3.4 State Diagram

State machine diagram capture the behavior of the software system. State machines can be used to model the behavior of a class or an entire application

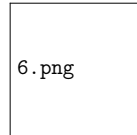


Figure 6.8: State Diagram

6.3.5 Design Constraints

1. The android version used in the app should be higher than Jelly_bean_MR2(4.3+)
2. The distance between the Alerto device and Smartphone should be within Bluetooth LE range.
3. App requires access to contact details, Bluetooth information, Device ID and call information

6.3.6 Software Interface Description

- We are using android to provide an interface to the user.
- The software interface(Android) provides various options and buttons such as connection, selection of device, vibration patterns, disconnect which are used for interacting with the Alerto device.

Chapter 7

DETAILED DESIGN DOCUMENT USING APPENDIX A AND B

7.1 INTRODUCTION

7.1.1 Purpose

The purpose of this document is to describe the implementation of the Alerto Android Application. This app is designed to notify the user of his mobile calls & notifications, when his phone isn't around him. activities.

7.1.2 Scope

This document describes the implementation details of the Alerto Android Application. The app makes a Alerto device(wearable) vibrate, when being used by the user. It consists of major functions such as:

- Connect- connects to the device;
- Send data- Vibration Patterns when any notification arrives.

7.1.3 Definitions, Acronyms, Abbreviations

Abbreviations

- BLE: Bluetooth Low Energy
- Android SDK: Android Software Development Kit

Definitions

- BLE: Compared to Classic Bluetooth, BLE is intended to provide considerably reduced power consumption and cost while maintaining a similar communication range.
- Telephony Manager: A Java class(in Android) that provides access to information about the telephony services on the device.
- Broadcast Receiver: A broadcast receiver (short receiver) is an Android component which allows you to register for system or application events.

- Services: A Service is an application component that can perform long-running operations in the background and does not provide a user interface.
- Alerto Device: A wearable device that vibrates on phone's incoming notifications, when the phone's isn't with the person.

7.2 ARCHITECTURAL DESIGN

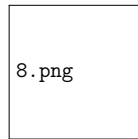


Figure 7.1: Communication module using bluetooth LE

7.2.1 Bluetooth Low Energy:

Compared to Classic Bluetooth, BLE is intended to provide considerably reduced power consumption and cost while maintaining a similar communication range. Mobile operating systems including iOS, Android, Windows Phone and BlackBerry, as well as OS X, Linux, and Windows 8, natively support BLE. BLE uses the same 2.4 GHz radio frequencies as Classic Bluetooth, which allows dual-mode devices to share a single radio antenna. LE does, however, use a simpler modulation system.

Internet Connectivity

- IPSP (Internet Protocol Support Profile)

Generic Sensors

- ESP (Environmental Sensing Profile)
- UDS (User Data Service)

HID Connectivity

- HOGP (HID over GATT Profile)

Proximity sensing:

”Electronic leash” applications are well suited to the long battery life possible for ‘always-on’ devices. Manufacturers of iBeacon devices implement the appropriate specifications for their device to make use of proximity sensing capabilities supported by Apple Inc. compatible iDevices.

Relevant application profiles include:

- FMP the ”find me” profile allows one device to issue an alert on a second misplaced device.
- PXP the proximity profile allows a proximity monitor to detect whether a proximity reporter is within a close range. Physical proximity can be estimated using the radio receiver’s RSSI value, although this does not have absolute calibration of distances. Typically, an alarm may be sounded when the distance between the devices exceeds a set threshold.

Alerts and time profiles

- The phone alert status profile and alert notification profile allow a client device to receive notifications such as incoming call alerts from another device.

BLE has 40 2-MHz channels. Within a channel, data is transmitted using Gaussian frequency shift modulation, similar to Classic Bluetooth’s Basic Rate scheme. The bit rate is 1Mbit/s, and the maximum transmit power is 10 mW. BLE uses frequency hopping to counteract narrow band interference problems. BLE is classified as a system using digital modulation techniques or a direct-sequence spread spectrum.

Table 7.1: Bluetooth Vs Bluetooth Low Energy

Technical Specifications	Classic Bluetooth Technology	Bluetooth Technology
Distance/Range (theoretical max.)	100 m (330 ft)	~100 m (~330 ft)
Over the air data rate	13 Mbit/s	1 Mbit/s
Application throughput	0.72.1 Mbit/s	0.27 Mbit/s
Active slaves	7	Not defined; implementation dependent
Security	56/128-bit and application layer user defined	128-bit AES with Counter Mode CBC-MAC and application layer user defined
Robustness	Adaptive fast frequency hopping, FEC, fast ACK	Adaptive frequency hopping, Lazy Acknowledgement, 24-bit CRC, 32-bit Message Integrity Check
Latency (from a non-connected state)	Typically 100 ms	6 ms
Minimum total time to send data (det.battery life)	100 ms	3 ms
Voice capable	Yes	No
Network topology	Scatternet	Scatternet
Power consumption	1 W as the reference	0.01 to 0.5 W (depending on use case)
Peak current consumption	~30 mA	~15 mA
Primary use cases	Mobile phones, gaming, headsets, stereo audio streaming, smart homes, wearables, automotive, PCs, security, proximity, healthcare, sports and fitness, etc.	Mobile phones, gaming, PCs, watches, sports and fitness, healthcare, security and proximity, automotive, home electronics, automation, Industrial, etc.

7.2.2 Bluetooth GATT(Software Model):

All BLE devices use the Generic Attribute Profile (GATT). GATT has the following terminology:

- Client:

A device that initiates GATT commands and requests, and accepts responses, for example a computer or smartphone.

- Server:

device that receives GATT commands and requests, and returns responses, for example a temperature sensor.

- Characteristic:

A data value transferred between client and server, for example the current battery voltage.

- Service:

A collection of related characteristics, which operate together to perform a particular function. Services may also include other services as sub-functions; the main functions of the device are so-called primaryservices, and the auxiliary functions they refer to are secondary services.

- Descriptor:

A descriptor provides additional information about a characteristic. Descriptors are optional - each characteristic can have any number of descriptors.

- Identifier:

Services, characteristics, and descriptors are collectively referred to as attributes, and identified by UUIDs. Any implementer may pick a random or

pseudorandom UUID for proprietary uses, but the Bluetooth SIG have reserved a range of UUIDs (of the form xxxxxxxx-0000-1000-8000-00805F9B34FB) for standard attributes. For efficiency, these identifiers are represented as 16-bit or 32-bit values in the protocol, rather than the 128 bits required for a full UUID. For example, the Device Information service has the short code 0x180A, rather than 0000180A-1000-.

- GATT Operation:

The GATT protocol provides a number of commands for the client to discover information about the server. These include:

- Discover UUIDs for all primary services
- Find a service with a given UUID
- Find secondary services for a given primary service
- Discover all characteristics for a given service
- Find characteristics matching a given UUID
- Read all descriptors for a particular characteristic.

Commands are also provided to read (data transfer from server to client) and write (from client to server) the values of characteristics:

- A value may be read either by specifying the characteristic's UUID, or by a handle value (which is returned by the information discovery commands above).
- Write operations always identify the characteristic by handle, but have a choice of whether or not a response from the server is required.
- 'Long read' and 'Long write' operations can be used when the length of the characteristic's data exceeds the MTU of the radio link.

Finally, GATT offers notifications and indications. The client may request a notification for a particular characteristic from the server. The server can then send the value to the client whenever it becomes available. An indication is similar to a notification, except that it requires a response from the client, as confirmation that it has received the message.

- Battery impact:

BLE is designed to enable devices with low power consumption. Devices with peripheral and central roles have different power requirements. A study by beacon software company. This is possible because of power efficiency of BLE protocol which only transmits small packets as compared to Bluetooth Classic which was also suitable for audio and high bandwidth data. In contrast, a continuous scan for the same beacons in central role can consume 1,000 mAh in few hours. With the newer chipsets and advances in software, both Android and iOS phones now have negligible power consumption in real-life BLE use scenarios.

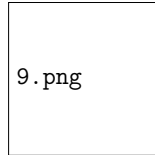


Figure 7.2: Architecture Diagram

7.2.3 Services:

A Service is an application component that can perform long-running operations in the background and does not provide a user interface. Another application component can start a service and it will continue to run in the background even if the user switches to another application. Additionally, a component can bind to a service to interact with it and even perform interprocess communication (IPC). A service can essentially take two forms:

- **Started:**

A service is "started" when an application component (such as an activity) starts it by calling `startService()`. Once started, a service can run in the background indefinitely, even if the component that started it is destroyed. Usually, a started service performs a single operation and does not return a result to the caller. For example, it might download or upload a file over the network. When the operation is done, the service should stop itself.

- **Bound:**

A service is "bound" when an application component binds to it by calling `bindService()`. A bound service offers a client-server interface that allows components to interact with the service, send requests, get results, and even do so across processes with interprocess communication (IPC). A bound service runs only as long as another application component is bound to it. Multiple

components can bind to the service at once, but when all of them unbind, the service is destroyed.

7.2.4 Activities :

An Activity is an application component that provides a screen with which users can interact in order to do something, such as dial the phone, take a photo, send an email, or view a map. Each activity is given a window in which to draw its user interface. The window typically fills the screen, but may be smaller than the screen and float on top of other windows.

An application usually consists of multiple activities that are loosely bound to each other. Typically, one activity in an application is specified as the "main" activity, which is presented to the user when launching the application for the first time. Each activity can then start another activity in order to perform different actions. Each time a new activity starts, the previous activity is stopped, but the system preserves the activity in a stack (the "back stack"). When a new activity starts, it is pushed onto the back stack and takes user focus. The back stack abides to the basic "last in, first out" stack mechanism, so, when the user is done with the current activity and presses the Back button, it is popped from the stack (and destroyed) and the previous activity resumes.

When an activity is stopped because a new activity starts, it is notified of this change in state through the activity's lifecycle callback methods. There are several callback methods that an activity might receive, due to a change in its state whether the system is creating it, stopping it, resuming it, or destroying it and each callback provides you the opportunity to perform specific work that's appropriate to that state change. For instance, when stopped, your activity should release any large objects, such as network or database connections. When the activity resumes, you can reacquire the necessary resources and resume actions that were interrupted. These

state transitions are all part of the activity lifecycle.

7.2.5 Telephony Manager :

Provides access to information about the telephony services on the device. Applications can use the methods in this class to determine telephony services and states, as well as to access some types of subscriber information. Applications can also register a listener to receive notification of telephony state changes. You do not instantiate this class directly; instead, you retrieve a reference to an instance through `Context.getSystemService(Context.TELEPHONY_SERVICE)`.

Note that access to some telephony information is permission-protected. Your application cannot access the protected information unless it has the appropriate permissions declared in its manifest file. Where permissions apply, they are noted in the the methods through which you access the protected information.

7.2.6 Broadcast Manager:

Helper to register for and send broadcasts of Intents to local objects within your process. This is has a number of advantages over sending global broadcasts with `sendBroadcast(Intent)`:

- You know that the data you are broadcasting won't leave your app, so do not need to worry about leaking private data.
- It is not possible for other applications to send these broadcasts to your app, so you do not need to worry about having security holes they can exploit.
- It is more efficient than sending a global broadcast through the system.

7.3 COMPONENT DESIGN

- Component-level design occurs after the first iteration of the architectural design
- It strives to create a design model from the analysis and architectural models
 - The translation can open the door to subtle errors that are difficult to find and correct later
 - "Effective programmers should not waste their time debugging they should not introduce bugs to start with." Edsgar Dijkstra
- A component-level design can be represented using some intermediate representation (e.g. graphical, tabular, or text-based) that can be translated into source code
- The design of data structures, interfaces, and algorithms should conform to well-established guidelines to help us avoid the introduction of errors

7.3.1 Class Diagram

Class diagram are the most common diagram found in modeling object oriented systems. A class diagram shows set of classes interface and collaboration and their relationships.

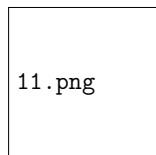


Figure 7.3: Class Diagram

Chapter 8

PROJECT IMPLEMENTATION

8.1 INTRODUCTION

Our project is a simple application. We are using android for implementation of our project. We have implemented on Android SDK which uses java and xml languages. Our app is using the Bluetooth LE technology which is much more cost effective than Bluetooth and has a greater battery life.

While implementing we have used classes and module based implementation is done.

8.2 TOOLS AND TECHNOLOGIES USED

8.2.1 HARDWARE RESOURCES

1. 400MB Hard disk+1GB for Android SDK, emulator system images and caches
2. Alerto device(Wearable)
3. 2GB RAM minimum,4GB RAM recommended
4. Intel Processor with support for Intel VT-x, Intel EM64T(Intel 64)
5. Computer with windows Vista/7/8.

8.2.2 SOFTWARE RESOURCES

- Platform: Android SDK
- Operating System: Android
- Programming Language: Java, XML.

8.2.3 TECHNOLOGY

- Bluetooth LE

8.3 METHOLOGIES AND ALGORITHMS DETAILS

8.3.1 Algorithm For Bluetooth LE Connecttion Establishment

1. Bluetooth is turned on.
2. Beacons are broadcasted for connection.
3. Device requests for connection.
4. User needs to select the device and press connect.
5. Connection completed.

8.3.2 Algorithm For Execution of the Android App

1. Install the app on android phone.
2. Turn on phone's bluetooth and select the device for connection.
3. The app is running in background in the phone.If: Incoming notification received- send vibrate() command to Alerto device.
4. Device vibrates

8.4 VERIFICATION AND VALIDATION FOR ACCEPTANCE

In software project management, software testing, and software engineering, verification and validation (V&V) is the process of checking that a software system meets specifications and that it fulfills its intended purpose. It may also be referred to as software quality control. It is normally the responsibility of software testers as part of the software development lifecycle.

Validation checks that the product design satisfies or fits the intended use (high-level checking), i.e., the software meets the user requirements. This is done through dynamic testing and other forms of review.

Verification and validation are not the same thing, although they are often confused. Boehm succinctly expressed the difference between

- Verification: Are we building the product right?
- Validation: Are we building the right product?

According to the Capability Maturity Model

- Software Verification: The process of evaluating software to determine whether the products of a given development phase satisfy the conditions imposed at the start of that phase.
- Software Validation: The process of evaluating software during or at the end of the development process to determine whether it satisfies specified requirements.

Verification: In our project we have compared and used various technologies which will be best fit for the application

1. Our project is based on Android operating system but it can be implemented on other operating systems as well. However we have selected Android because it is much more popular in comparison to other OSs which increases the usability of our application.
2. Bluetooth Low energy uses less amount of battery power as compared to regular Bluetooth devices. Hence using this technology helps to reduce power consumption and increase battery life. Thus we have used this technology for our application

Chapter 9

SOFTWARE TESTING

9.1 TYPES OF TESTING USED

Since our application is on android, we have done manual testing. There are different stages for manual testing such as unit testing, integration testing, system testing, and user acceptance testing.

1. Unit testing

In computer programming, unit testing is a software testing method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures, are tested to determine whether they are fit for use.

In our program, we have units such as Connection() which is used for connection, Vibrate() to check if the vibration occurs, and different modules for checking if device vibrates when incoming call, Incoming message or notification from third party app comes on the android phone which have been successfully tested unitwise.

2. Integration testing

Integration testing (sometimes called integration and testing, abbreviated I&T) is the phase in software testing in which individual software modules are combined and tested as a group. It occurs after unit testing and before validation testing.

In our program we have integrated the connection and vibration module to be tested as integration testing. Also we have integrated the modules which the functionality of vibration for different incoming notifications.

3. System testing

System testing of software or hardware is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified require-

ments. System testing falls within the scope of black-box testing, and as such, should require no knowledge of the inner design of the code or logic. We have successfully performed the system testing in which the system was tested as a whole.

4. User Acceptance testing

User acceptance testing (UAT) is the last phase of the software testing process. During UAT, actual software users test the software to make sure it can handle required tasks in real-world scenarios, according to specifications. We have successfully completed this testing.

9.2 TEST CASES AND TEST RESULTS

Project Name:Alerto					
Test Case ID:001					
Test Priority(Low/Med/High):High					
Test Designed and Executed by:Student					
Test Designed and Executed date:01/04/2016					
Module Name:Connection					
Test Title:Testcase1					
Description:Checking if the device(Alerto) and Android phone connect					
Pre-conditions: Show the connection UI to user.					
Post condition: The Alerto device and android phone connect over Bluetooth low energy.					
Steps	Test steps	Test data	Expected Result	Actual Result	Status
1	Check whether the application gets started.	Running the application	The app starts when run.	The app starts when run.	pass
2	Check if available devices for connection are shown.	Available devices information	All devices in range are shown	All device in range are shown	pass
3	Check if the selected device gets connected.	Selected device.	Successful connection and device information	Successful connection and device information	pass
4	Check if connection occurs when device is out of range.	Out of range device	Shows error message.	Abruptly disconnects	fail

Project Name:Alerto					
Test Case ID:002					
Test Priority(Low/Med/High):High					
Test Designed and Executed by:Student					
Test Designed and Executed date:01/04/2016					
Module Name:Vibration					
Test Title:Testcase2					
Description:Check if after connection, device vibrates on pressing vibrate button.					
Pre-conditions: Show the vibration UI to user.					
Post condition: : Device vibrates after giving the command.					
Steps	Test steps	Test data	Expected Result	Actual Result	Status
1	Check whether vibrate key is working correctly.	Vibration Byte data.	Key should work correctly.	Key is working correctly.	Pass
2	Check if vibration occurs.	Byte data.	Device should vibrate.	Device is vibrating after giving command.	Pass

Project Name:Alerto					
Test Case ID:003					
Test Priority(Low/Med/High):High					
Test Designed and Executed by:Student					
Test Designed and Executed date:01/04/2016					
Module Name:Call					
Test Title:Testcase3					
Description:: Checking if device vibrates when call comes					
Pre-conditions: App running.					
Post condition: Alerto device vibrates when notification of call comes on phone..					
Steps	Test steps	Test data	Expected Result	Actual Result	Status
1	Check whether device vibrates.	Notification from call.	Device vibrates on notification.	Device vibrates on notification.	Pass
2	Check if device vibrated when out of range.	Notification from call.	Error message.	Device disconnected.	Fail

Project Name:Alerto					
Test Case ID:004					
Test Priority(Low/Med/High):High					
Test Designed and Executed by:Student					
Test Designed and Executed date:01/04/2016					
Module Name:Messages					
Test Title:Testcase4					
Description:: Checking if device vibrates when messages comes					
Pre-conditions: App running.					
Post condition: Alerto device vibrates when message comes on phone..					
Steps	Test steps	Test data	Expected Result	Actual Result	Status
1	Check whether device vibrates.	Notification from messages.	Device vibrates on notification.	Device vibrates on notification.	Pass
2	Check if device vibrated when out of range.	Notification from messages.	Error message.	Device disconnected.	Fail

Project Name:Alerto					
Test Case ID:005					
Test Priority(Low/Med/High):High					
Test Designed and Executed by:Student					
Test Designed and Executed date:01/04/2016					
Module Name:NotificationListenersService					
Test Title:Testcase5					
Description:: Checking if device vibrates when third party apps give notification					
Pre-conditions: App running.					
Post condition: Alerto device vibrates when notification of third party apps comes on phone..					
Steps	Test steps	Test data	Expected Result	Actual Result	Status
1	Check whether device vibrates.	Notification from third party apps.	Device vibrates on notification.	Device vibrates on notification.	Pass
2	Check if device vibrated when out of range.	Notification from third party apps.	Error message.	Device disconnected.	Fail

Chapter 10

RESULTS

10.1 SCREEN SHOTS



Chapter 11

DEPLOYMENT AND MAINTENANCE

11.1 INSTALLATION AND UN-INSTALLATION

11.1.1 Installation

Once we execute the application it gets installed on the phone automatically. The Alerto device is connected to the phone via Bluetooth low energy. When an incoming call or message is received on the phone, the Alerto will vibrate informing the user about the notification. When we stop the execution of the code, the app remains installed on Android phone

11.1.2 Un-installation

The app needs to be manually removed from the phone as it exist even after we stop running the code.

11.2 USER HELP

Our app is quite simple. We have made it very user-friendly. The user can easily navigate through the pages of the app.

Chapter 12

FUTURE ENHANCEMENT AND CONCLUSION

12.1 FUTURE SCOPE

In the future, we want to add the following as enhancement:

1. The favouritism can be extended to emails.
2. The application can be used with other wearable devices also, which are using bluetooth LE technology.
3. Further more options for alarms can be added to the application.

12.2 CONCLUSION

In this project we are developing a mobile based "Alerto" application. This application would help to notify the user whenever a notification arrives on its phone.

The system uses bluetooth low energy technology which is used to sync our mobile devices with a wearable device called "Alerto". Whenever a notification arrives on our mobile phone the Alerto notifies the user with a vibration.

Thus, we successfully solved the problem of being worried for an important phone call because this application separates the messages/calls from the important ones.

In future, more functionalities can be added to use this application with another device with more features. With the advent of smart phones, when developed to its fullest would be able for all to use.

Chapter 13

REFERENCES

Bibliography

- [1] Jia Liu, Member, IEEE, Canfeng Chen, Member, IEEE, and Yan Ma, Member, IEEE
Modeling Neighbor Discovery in Bluetooth Low Energy Networks
IEEE COMMUNICATIONS LETTERS, VOL. 16, NO. 9, SEPTEMBER 2012
- [2] Jia Liu, Member, IEEE, Canfeng Chen, Member, IEEE, and Yan Ma, Member, IEEE
Energy Analysis of Device Discovery for Bluetooth Low Energy, 2013
- [3] Jia Liu, Member, IEEE, Canfeng Chen, Member, IEEE, and Yan Ma, Member, IEEE
Advertising semantically described physical items with Bluetooth Low Energy beacons, 2013
2nd Mediterranean Conference on Embedded Computing „,/” MECD - 2013
- [4] Takeshi Iwamoto¹, Nobuhiko Nishio¹, and Hideyuki Tokuda^{1,2}
Wapplet: A Media Access Framework for Wearable Applications, 2002
1. Graduate School of Media and Governance, Keio University, Japan iwaiwa,
vino, hxt@ht.sfc.keio.ac.jp
2. Faculty of Environmental Information, Keio University, Japan
- [5] Anindya Nag and Subhas Chandra Mukhopadhyay
Wearable Electronics Sensors: Current Status and Future Opportunities, 2015
Massey University, Palmerston North, New Zealand

Annexure A

LABORATORY ASSIGNMENTS ON PROJECT ANALYSIS OF ALGORITHM DESIGN

- To develop the problem under consideration and justify feasibility using concepts of knowledge canvas and IDEA Matrix.

IDEA Matrix is represented in the following form. Knowledge canvas represents about identification of opportunity for product. Feasibility is represented w.r.t. business perspective.

I	D	E	A
Increase	Drive	Educate	Accelerate
Improve	Deliver	Evaluate	Associate
Ignore	Decrease	Eliminate	Avoid

Table A.1: IDEA Matrix

- Project problem statement feasibility assessment using NP-Hard, NP-Complete or satisfy ability issues using modern algebra and/or relevant mathematical models.

Polynomial(P) and Non-Polynomial(NP)

- Polynomial Time:

The class of polynomial-time solvable problems P includes all the sets in which membership may be decided by an algorithm whose running time is bounded by a polynomial.

Example:

- * Linear Time- $O(n)$
- * Quadratic Time- $O(n^2)$
- * Exponential Time- $O(2^n)$
- * Logarithmic Time- $O(\log n)$

- Non-Polynomial Time:

The class non-deterministic polynomially acceptable problems, NP contains all sets in which membership can be verified in polynomial time. All problems from P is also NP. Examples:

- * Decision problem version of the integer factorisation problem
- * Graph isomorphism problem
- * A decision version of travelling salesman problem
- * The Boolean satisfiability problem

Types of NP type problems:

- * NP-hard:

In computational complexity theory, is a class of problems that are, informally, "at least as hard as the hardest problems in NP". More precisely, a problem H is NP-hard when every problem L in NP can be reduced in polynomial time to H. As a consequence, finding a polynomial algorithm to solve any NP-hard problem would give polynomial algorithms for all the problems in NP, which is unlikely as many of them are considered hard.

- * NP-complete:

In computational complexity theory, a decision problem is NP-complete when it is both in NP and NP-hard. The set of NP-complete problems is often denoted by NP-C or NPC. The abbreviation NP refers to "nondeterministic polynomial time".

Thus from the above definition our application of is P (Polynomial) class.

- **Input x, Output y, $y=f(x)$**

– I: Set of Inputs = {I1, I2}

where,

- * I1: Installation of the Android application Alerto

- * I2: Running of the application

- O: Set of Outputs={O1, O2}

where,

- * O1: List of available Alertos

- * O2: Connection established with a particular Alerto.

- F:Set of Functions={F1, F2, F3, F4, F5}

where,

- * F1: Get started with the application.

- * F2: Turn on Bluetooth.

- * F3: Scan for available Alerto devices in range.

- * F4: Connect with specific Alerto.

- * F5: Once desired notification arrives on emulator, the particular Alerto vibrates.

Annexure B

LABORATORY ASSIGNMENTS ON PROJECT QUALITY AND RELIABILITY TESTING OF PROJECT DESIGN

- Use of divide and conquer strategies to exploit distributed/parallel/concurrent processing of the above to identify object, morphisms, overloading in functions (if any), and functional relations and any other dependencies (as per requirements). It can include Venn diagram, state diagram, function relations, i/o relations; use this to derive objects, morphism, overloading

With this strategy, a problem is solved by splitting it into sub-problems, solving them independently, and merging their solutions into a solution for the whole problem. The sub-problems can be solved directly, or they can in turn be solved using the same divide-and-conquer strategy, leading to an overall recursive program structure. The potential concurrency in this strategy is not hard to see: Since the sub-problems are solved independently, their solutions can be computed concurrently, leading to a parallel program that is very similar to its sequential counterpart. The following figure illustrates the strategy and the potential concurrency. **Identify Morphism**

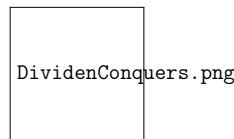


Figure B.1: Divide and Conquer Strategy

Much of the terminology of morphisms, as well as the intuition underlying them, comes from concrete categories, where the objects are simply sets with some additional structure, and morphisms are structure-preserving functions. A category C consists of two classes, one of objects and the other of morphisms. There are two objects that are associated to every morphism, the source and the target. For many common categories, objects are sets (usually with more structure) and morphisms are functions from an object to another object. Therefore the source and the target of a morphism are often called respectively domain and codomain. A morphism f with source X and target Y is writ-

ten $f : X \rightarrow Y$. Thus a morphism is represented by an arrow from its source to its target. In the category of smooth manifolds, morphisms are smooth functions and isomorphisms are called diffeomorphisms.

Functional Dependency

Functional dependency is a relationship that exists when one attribute uniquely determines another attribute.

If R is a relation with attributes X and Y , a functional dependency between the attributes is represented as $X \twoheadrightarrow Y$, which specifies Y is functionally dependent on X . Here X is a determinant set and Y is a dependent attribute. Each value of X is associated precisely with one Y value.

- Use of above to draw functional dependency graphs and relevant Software modeling methods, techniques including UML diagrams or other necessities using appropriate tools.

1. To draw functional dependency graph.

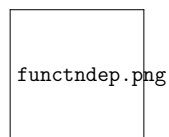


Figure B.2: Functional Dependency Graph

2. UML diagrams or other necessities using appropriate tools. UML diagrams consists of following diagrams:

- (a) Use-Case Diagram
 - (b) Activity Diagram
 - (c) State Diagram
 - (d) Component Diagram
 - (e) Class Diagram
 - (f) Package Component Diagram
 - (g) Deployment Diagram
 - (h) Sequence Diagram
- Testing of project problem statement using generated test data (using mathematical models, GUI, Function testing principles, if any) selection and appropriate use of testing tools, testing of UML diagram's reliability. Write also test cases [Black box testing] for each identified functions. You can use Mathematica or equivalent open source tool for generating test data.

Software testing can be stated as process of validating and verifying that a computer program/application/product:

- Meets the requirements that guided its design and development.
- Works as expected.
- Can be implemented with the same characteristics.
- Satisfies the needs of stake holders.

Software testing, depending on the testing methods employed, can be implemented at any time in the development process. Traditionally, most of the test efforts occur after the requirements have been defined and the coding process has been completed but in the agile approaches most of the test effort

is ongoing. As such the methodology of test is governed by a chosen software development methodology.

Types of testing:

– Unit testing:

Unit testing is a software testing method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures, are tested to determine whether they are fit for use. Intuitively, one can view a unit as the smallest testable part of an application. In procedural programming, a unit could be an entire module, but it is more commonly an individual function or procedure. In object-oriented programming, a unit is often an entire interface, such as a class, but could be an individual method. Unit tests are short code fragments created by programmers or occasionally by white box testers during the development process. It forms the basis for component testing.

– Integration testing:

Integration testing is the phase in software testing in which individual software modules are combined and tested as a group. It occurs after unit testing and before validation testing. Integration testing takes as its input modules that have been unit tested, groups them in larger aggregates, applies tests defined in an integration test plan to those aggregates, and delivers as its output the integrated system ready for system testing.

– Acceptance testing:

Acceptance testing is a test conducted to determine if the requirements of a specification or contract are met. It may involve chemical tests, physical tests, or performance tests. In systems engineering it may involve black-

box testing performed on a system (for example: a piece of software, lots of manufactured mechanical parts, or batches of chemical products) prior to its delivery.

Test Cases

1. List of devices shown properly.
2. Connection established.
3. Configuration of Settings.
4. Notifications alerted.
5. Correct notifications are alerted

GUI testing

GUI test is for testing if the GUI we have made for the application is working in the correct manner as desired. In our application we are going to have many buttons and lists so we will be checking if they are working properly.

Annexure C

PROJECT PLANNER

Table C.1: Plan of Project

From	To	Task	Status
27-06-2015	30-06-2015	Group Formation and finalization	Done
01-07-2015	15-07-2015	Topic Search	Done
16-07-2015	20-07-2015	Preliminary Information Gathering	Done
21-07-2015	28-07-2015	Project Discussion with Project Coordinator and topic finalization	Done
01-08-2015	04-08-2015	Synopsis preparation and submission	Done
25-08-2015	30-08-2015	Detailed Literature Survey	Done
19-09-2015	24-09-2015	.	.
25-09-2015	06-10-2015	Preparing Interim report	Done
20-12-2015	02-01-2016	Language Study	Done
03-01-2016	20-01-2016	Android and Bluetooth LE Study	Done
21-01-2016	04-03-2016	Coding and Implementation	Done
05-03-2016	21-04-2016	Testing	Done
13-04-2016	21-04-2016	Final Documentation	Done
25-04-2016	20-05-2016	Final Project Report	Done

Annexure D

TERM-II PROJECT LABORATORY ASSIGNMENTS

1. Review of design and necessary corrective actions taking into consideration the feedback report of Term I assessment, and other competitions/conferences participated like IIT, Central Universities, University Conferences or equivalent centers of excellence etc.

After presenting the design of our project, a few questions were asked regarding the detailing and feasibility of its working. We were queried whether the device could be made auditory instead of vibrating for alerts. No specific correction were made necessary for our project design.

2. Project workstation selection, installations along with setup and installation report preparations.

Android is used as the base for our project. Android is a mobile operating system (OS) currently developed by Google, based on the Linux kernel and designed primarily for touchscreen mobile devices such as smartphones and tablets. Android's user interface is mainly based on direct manipulation, using touch gestures that loosely correspond to real-world actions, such as swiping, tapping and pinching, to manipulate on-screen objects, along with a virtual keyboard for text input. In addition to touchscreen devices, Google has further developed Android TV for televisions, Android Auto for cars, and Android Wear for wrist watches, each with a specialized user interface. Variants of Android are also used on notebooks, game consoles, digital cameras, and other electronics.

3. Programming of the project functions, interfaces and GUI (if any) as per 1 st Term term-work submission using corrective actions recommended in Term-I assessment of Term-work.

Many functions are implemented in our android code. They include:

- Vibrate: This function is used to make the wearable device vibrate once the app is running.

- Connect: It connect the android app to the wearable device
 - Disconnect: This disconnects the device from the app and no more communication can occur between them.
4. Test tool selection and testing of various test cases for the project performed and generate various testing result charts, graphs etc. including reliability testing.
- Testing is a critical software development activity because it helps you improve the quality of your apps, ensure better user satisfaction, and reduce overall development time spent on fixing defects.

The following sections describe tools that help you test your mobile apps for the Android platform.

(a) **Android Testing Support Library**

This library provides a set of APIs that allow you to quickly build and run test code for your apps, including JUnit 4 and functional user interface (UI) tests. The Android Testing Support Library includes the following test automation tools:

- AndroidJUnitRunner: JUnit 4-compatible test runner for Android
- Espresso: UI testing framework; suitable for functional UI testing within an app
- UI Automator: UI testing framework; suitable for cross-app functional UI testing across system and installed apps

(b) **Monkey**

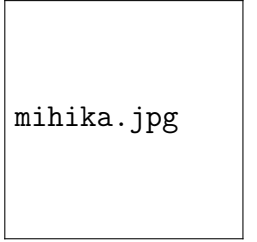
This tool runs on your emulator or device and generates pseudo-random streams of user events such as clicks, touches, or gestures, as well as a number of system-level events. You can use the Monkey tool to stress-test applications that you are developing, in a random yet repeatable manner.

monkeyrunner

This testing system provides an API for writing programs that control an Android device or emulator from outside of Android code.

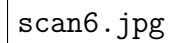
Annexure E

INFORMATION OF PROJECT GROUP MEMBERS

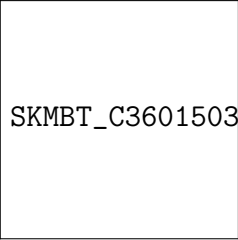


mihika.jpg

1. Name : Mihika Deshpande
2. Date of Birth :18/03/1994
3. Gender : Female
4. Permanent Address: C10, Kapil Malhar, Baner Road, Baner, Pune 411045
5. E-Mail : mihikadd@gmail.com
6. Mobile/Contact No. :9673722106
7. Placement Details :
8. Paper Published : NO


scan6.jpg

1. Name : Tarini Parihar
2. Date of Birth :8/04/1995
3. Gender : Female
4. Permanent Address: Plot No. 51, Vasant Nagar, Jawahar Colony road, Aurangabad-431005
5. E-Mail :tarini.parihar@gmail.com
6. Mobile/Contact No. :9422206944
7. Placement Details :
8. Paper Published : NO



SKMBT_C36015030406051.jpg

1. Name : Ruchi Jain
2. Date of Birth :21/06/1994
3. Gender : Female
4. Permanent Address: 699/2A, Chandrakiran apts; Mukundnagar Pune-411037
5. E-Mail : ruchijain367@gmail.com
6. Mobile/Contact No. :9021645697
7. Placement Details :
8. Paper Published : NO



zindagi.png

1. Name : Zindagi
2. Date of Birth :27/11/1992
3. Gender : Female
4. Permanent Address: D/o Bikask Kumar Sahu, Bhikhanpur, Gumti No.2, Bhatta Road, Bhagalpur, Bihar-812001
5. E-Mail : ginilife@gmail.com
6. Mobile/Contact No. :9657110879
7. Placement Details :
8. Paper Published : NO