**MOBILE BASED ANTI-THEFT ANDROID PROJECT**

A Mini Project Report Submitted   
In partial fulfillment of the requirements for the award of the degree of

## Bachelor of Technology

## in

## Computer Science and Engineering

by

## CHILUVURI SATHVIK 16N31A0537

## AKKALA JAGATH CHANDRA 16N31A0505

## EADARA MARUTHI KUMAR 16N31A0553

Under the esteemed guidance of

**Mr. G. Ravi**

**Associate Professor**



**Department of Computer Science and Engineering**

**Malla Reddy College of Engineering & Technology**

**(Autonomous Institution- UGC, Govt. of India)**

**(Affiliated to JNTUH, Hyderabad, Approved by AICTE, NBA &NAAC with ‘A’ Grade)**

**Maisammaguda, Kompally, Dhulapally, Secunderabad – 500100  
website: www.mrcet.ac.in**

**2019-2020**



**Malla Reddy College of Engineering & Technology**

**(Autonomous Institution- UGC, Govt. of India)**

**(Affiliated to JNTUH, Hyderabad, Approved by AICTE, NBA &NAAC with ‘A’ Grade)**

**Maisammaguda, Kompally, Dhulapally, Secunderabad – 500100  
website: www.mrcet.ac.in**

**CERTIFICATE**

This is to certify that this is the bonafide record of the project entitled “ Mobile based Anti-Theft Android Project ”, submitted by Chiluvuri Sathvik (16N31A0537), Akkala Jagath Chandra (16N31A0505) and Eadara Maruthi Kumar(16N31A0553) of B.Tech in the partial fulfillment of the requirements for the degree of Bachelor of Technology in Computer Science and Engineering, Department of CSE during the year 2019-2020. The results embodied in this project report have not been submitted to any other university or institute for the award of any degree or diploma.

**Internal Guide Head of the Department**

**Mr. G. Ravi Dr. D. Sujatha**

**Associate Professor Professor**

**R**

**External Examiner**

**DECLARATION**

## We hereby declare that the project titled “Mobile based Anti-Theft Android project” submitted to Malla Reddy College of Engineering and Technology (UGC Autonomous), affiliated to Jawaharlal Nehru Technological University Hyderabad (JNTUH) for the award of the degree of Bachelor of Technology in Computer Science and Engineering is a result of original research carried-out in this thesis. It is further declared that the project report or any part thereof has not been previously submitted to any University or Institute for the award of degree or diploma.

**Chiluvuri Sathvik - 16N31A0537**

**Akkala Jagath Chandra - 16N31A0505**

**Eadara Maruthi Kumar - 16N31A0553**

**ACKNOWLEDGMENT**

We feel ourselves honored and privileged to place our warm salutation to our college Malla Reddy College of Engineering and Technology (UGC-Autonomous) and our principal Dr. VSK Reddy who gave us the opportunity to have experience in engineering and profound technical knowledge.

We express our heartiest thanks to our Head of the Department Dr. D. Sujatha for encouraging us in every aspect of our project and helping us realize our full potential.

We would like to thank our internal guide and Project Coordinator G. Ravi for his regular guidance and constant encouragement. We are extremely grateful to his valuable suggestions and unflinching co-operation throughout project work.

We would like to thank our class in charge Mr. M.V. Kamal who in spite of being busy with his duties took time to guide and keep us on the correct path.

We would also like to thank all the supporting staff of the Department of CSE and all other departments who have been helpful directly or indirectly in making our project a success.

We are extremely grateful to our parents for their blessings and prayers for the completion of our project that gave us strength to do our project.

with regards and gratitude

**Chiluvuri Sathvik - 16N31A0537**

**Akkala Jagath Chandra - 16N31A0505**

**Eadara Maruthi Kumar - 16N31A0553**

**ABSTRACT**

The aim of this project is to build a system that will enable the real time tracking of a mobile phone and provide added location aware features.

The mobile phone has quickly become ubiquitous in modern society and everyone feels the need of having one with advanced features. This mobile phone while being a portal to access the whole world can also be used to make mundane tasks simpler. This system will enable the tracking of the device using its Global Positioning System (GPS) or Global Navigation Satellite System (GLONASS) functionality. Over the past decade more and more phones have begun integrating either the GPS or GLONASS or both systems on board to provide navigation assistance. These systems are able to lock on to satellites and provide accurate location details within seconds. The Mobile Phone Tracking system will be able to activate this feature in the phone and locate the user on a map. The system will be built with a XML front end and can be accessed after providing authorized user credentials

**CONTENTS :**

**1. INTRODUCTION**

**1.1. PROBLEM DEFINITION**

**1.2 EXISTING SYSTEM**

**1.3 PROPOSED SYSTEM**

**1.3.1 Modules**

**1.4 SCOPE OF PROJECT**

**2. LITERATURE SURVEY**

**2.1. JAVA**

**2.2. ANDROID**

**2.2.1 About Native code**

**2.2.2 Creating an android project**

**2.2.3 To create an AVD with the AVD manager**

**2.2.4 To Create or Modify a Launch Configuration**

**2.3. SQLITE**

**2.3.1 Design**

**2.3.2 Features**

**3. SYSTEM ANALYSIS**

**3.1 Software Requirements**

**3.2 Hardware Requirements**

### 

### 3.3 Feasibility Study

* + 1. **Economic Feasibility**

**3.3.2 Operational Feasibility**

**3.3.3** **Technical Feasibility**

**4. SYSTEM DESIGN**

**4.1 Architectural Diagram**

**4.2 Unified Modeling Language (UML)**

**5. SYSTEM CODING & IMPLEMENTATION**

**5.1 Sample Code**

**5.2 SCREEN SHOTS**

**6. TESTING AND VALIDATION**

**6.1 Unit Testing**

**6.2 Integration Testing**

**6.3 Validation Testing**

**6.4 System Testing**

**6.5 Test cases**

**7. CONCLUSIONS**

**8. REFERENCES**

**Chapter 1**

1. **INTRODUCTION TO PROJECT**

The aim of this project is to build a system that will enable the real time tracking of a mobile phone and provide added location aware features.

The mobile phone has quickly become ubiquitous in modern society and everyone feels the need of having one with advanced features. This mobile phone while being a portal to access the whole world can also be used to make mundane tasks simpler. This system will enable the tracking of the device using its Global Positioning System (GPS) or Global Navigation Satellite System (GLONASS) functionality. Over the past decade more and more phones have begun integrating either the GPS or GLONASS or both systems on board to provide navigation assistance. These systems are able to lock on to satellites and provide accurate location details within seconds. The Mobile Phone Tracking system will be able to activate this feature in the phone and locate the user on a map. The system will be built with a XML front end and can be accessed after providing authorized user credentials

**1.1 Document Conventions**

In this SRS we used Font Face: ‘Times New Roman’ and Font size: 14 and Font Style: bold for Headings .Font size 16 is used for main headings. Font size 12 is used for the content. Underlining is done for technically important words.

**1.2 Intended Audience and Reading Suggestions**

This document is for better understanding of Service Provider. Mainly intended for Head of the Dept., Internal guide, External guide, Staff members and Users. These details given below guides every normal user how to go through this document for better understanding. The sequence to follow for better understanding is here Purpose, Scope, Features of the Service Provider software, Operating requirements, Modules present in the project, Advantages, References etc.

**Chapter 2**

**2.1 Project Perspective**

The aim of this project is to build a system that will enable the real time tracking of a mobile phone and provide added location aware features.

**2.1.1 Existing System :**

The Existing System has some of the Drawbacks like the exact position of the user cannot be retrieved. The application mainly used by only owners and administrator. The moment of the user is also not visible.

**2.1.2 Proposed System :**

The proposed system provides the user to find exact location of the user from where they are.

1. It provides exact position in Google map.
2. This will enable a person to know the location of the another users.

**2.1.3 Project Features**

The track log application for mobile devices!

1. Extensive statistical information
2. Configurable time span between two positions
3. Elevation can be tracked

**2.1.4 User Classes and Characteristics**

This project consists of only one end user.

**2.1.5 Operating Environment**

Software Requirements are Windows as Operating System. HTML, Cascading Style Sheets and XML as Front-End designing tools. Sqlite is used as Database Server. Default layout is relative layout. Hardware Requirements are pentiumIV 2.4 ghz processor, 1GB of main memory (RAM) and 100MB hard disk and data base memory.

**2.1.6 Design and Implementation Constraints:**

This project will be developed using the technologies like Android Frontend technologies are xml,backend is sqlite for all which I need to fetch knowledge on all of them in order to code for the project. Also I’ll be learning clearly about Software Development Life Cycle.

**2.2 Non Functional requirements:**

* This provides good security.
* This can be accessed easily.
* Maintainability.
* Portability.
* Quality.

**2.3 Modules Description**

Human tracking with mobile:

1. This amazing invention in software which is also loaded in the cell.
2. This will enable a person to know the location of the another users
3. Notification of the area on the caller’s cell.

**2.4 Feasibility Study:**

The next step in analysis is to verify the feasibility of the proposed system. “All projects are feasible given unlimited resources and infinite time“. But in reality both resources and time are scarce. Project should confirm to time bounce and should be optimal in their consumption of resources. This place a constant is approval of any project.

* Technical feasibility
* Operational feasibility
* Economic feasibility

**2.4.1 Technical Feasibility:**

To determine whether the proposed system is technically feasible, we should take into consideration the technical issues involved behind the system.

This Application uses the web technologies, which is rampantly employed these days worldwide. The world without the web is incomprehensible today. That goes to proposed system is technically feasible.

**2.4.2 Operational Feasibility:**

To determine the operational feasibility of the system we should take into consideration the awareness level of the users. This system is operational feasible and friendly .

**2.4.3 Economic Feasibility:**

To decide whether a project is economically feasible, we have to consider various factors as:

* + - * Cost benefit analysis
      * Long-term returns
      * Maintenance costs

It requires average computing capabilities and access to internet, which are very basic requirements and can be afforded by any organization hence it doesn’t incur additional economic overheads, which renders the system economically feasible.

**2.5 SDLC (SOFTWARE DEVELOPMENT LIFE CYCLE)**

**2.5.1 SDLC Methodologies**

This document play a vital role in the development of life cycle (SDLC) as it describes the complete requirement of the system. It means for use by developers and will be the basic during testing phase. Any changes made to the requirements in the future will have to go through formal change approval process.

**2.5.2 SPIRAL Model**

It was defined by Barry Boehm in his 1988 article, “A spiral Model of Software Development and Enhancement. This model was not the first model to discuss iterative development. The spiral model is similar to the [incremental model](http://istqbexamcertification.com/what-is-incremental-model-advantages-disadvantages-and-when-to-use-it/), with more emphasis placed on risk analysis. The spiral model has four phases: Planning, Risk Analysis, Engineering and Evaluation. A software project repeatedly passes through these phases in iterations (called Spirals in this model). The baseline spirals, starting in the planning phase, requirements are gathered and risk is assessed. Each subsequent spiral builds on the baseline spiral. As originally envisioned, the iterations were typically 6 months to 2 years long. Each phase starts with a design goal and ends with a client reviewing the progress thus far. Analysis and engineering efforts are applied at each phase of the project, with an eye toward the end goal of the project.

The steps for Spiral Model can be generalized as follows:

* The new system requirements are defined in as much details as possible. This usually involves interviewing a number of users representing all the external or internal users and other aspects of the existing system.
* A preliminary design is created for the new system.
* A first prototype of the new system is constructed from the preliminary design. This is usually a scaled-down system, and represents an approximation of the characteristics of the final product.
* A second prototype is evolved by a fourfold procedure:

1. Evaluating the first prototype in terms of its strengths, weakness, and risks.
2. Defining the requirements of the second prototype.
3. Planning an designing the second prototype.
4. Constructing and testing the second prototype.

* At the customer option, the entire project can be aborted if the risk is deemed too great. Risk factors might involved development cost overruns, operating-cost miscalculation, or any other factor that could, in the customer’s judgment, result in a less-than-satisfactory final product.
* The existing prototype is evaluated in the same manner as was the previous prototype, and if necessary, another prototype is developed from it according to the fourfold procedure outlined above.
* The preceding steps are iterated until the customer is satisfied that the refined prototype represents the final product desired.
* The final system is constructed, based on the refined prototype.
* The final system is thoroughly evaluated and tested. Routine maintenance is carried on a continuing basis to prevent large scale failures and to minimize down time.



**Fig-Spiral Model**

* **Planning Phase:**Requirements are gathered during the planning phase. Requirements like ‘BRS’ that is ‘Business Requirement Specifications’ and ‘SRS’ that is ‘System Requirement specifications’.
* **Risk Analysis:** In the**risk analysis phase**, a [process](http://istqbexamcertification.com/what-is-spiral-model-advantages-disadvantages-and-when-to-use-it/) is undertaken to identify risk and alternate solutions.  A prototype is produced at the end of the risk analysis phase. If any risk is found during the risk analysis then alternate solutions are suggested and implemented.
* **Engineering Phase:** In this phase software is **developed**, along with [testing](http://istqbexamcertification.com/what-is-a-software-testing/) at the end of the phase. Hence in this phase the development and testing is done.
* **Evaluation phase:**This phase allows the customer to evaluate the output of the project to [date](http://istqbexamcertification.com/what-is-spiral-model-advantages-disadvantages-and-when-to-use-it/) before the project continues to the next spiral.

**Chapter 3**

**SYSTEM DESIGN**

System design is transition from a user oriented document to programmers or data base personnel. The design is a solution, how to approach to the creation of a new system. This is composed of several steps. It provides the understanding and procedural details necessary for implementing the system recommended in the feasibility study. Designing goes through logical and physical stages of development, logical design reviews the present physical system, prepare input and output specification, details of implementation plan and prepare a logical design walkthrough.

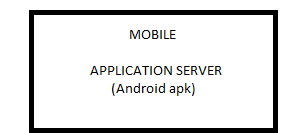
The database tables are designed by analyzing functions involved in the system and format of the fields is also designed. The fields in the database tables should define their role in the system. The unnecessary fields should be avoided because it affects the storage areas of the system. Then in the input and output screen design, the design should be made user friendly. The menu should be precise and compact.

**3.1 SOFTWARE DESIGN**

In designing the software following principles are followed:

* **Modularity and partitioning**: software is designed such that, each system should consists of hierarchy of modules and serve to partition into separate function.
* **Coupling:** modules should have little dependence on other modules of a system.
* **Cohesion:** modules should carry out in a single processing function.
* **Shared use:** avoid duplication by allowing a single module be called by other that need the function it provides

**3.1.1 SYSTEM ARCHITECTURE DESIGN**

****

SQLite

Database

**3.1.2**  **APPLICATION ARCHITECTURE DESIGN**

DATABASE

BUSINESS ACCESS LAYER

DATA ACCESS LAYER

PRESENTATION LAYER

**3.2INPUT/OUTPUT DESIGN**

**3.2.1 Input design:**

Considering the requirements, procedures to collect the necessary input data in most efficiently designed. The input design has been done keeping in view that, the interaction of the user with the system being the most effective and simplified way.

Also the measures are taken for the following

* + - Controlling the amount of input
    - Avoid unauthorized access to the classroom.
    - Eliminating extra steps
    - Keeping the process simple
    - At this stage the input forms and screens are designed.

**3.2.2 Output design:**

All the screens of the system are designed with a view to provide the user with easy operations in simpler and efficient way, minimum key strokes possible. Instructions and important information is emphasized on the screen. Almost every screen is provided with no error and important messages and option selection facilitates. Emphasis is given for speedy processing and speedy transaction between the screens. Each screen assigned to make it as much user friendly as possible by using interactive procedures. So to say user can operate the system without much help from the operating manual.

**3.3 SOFTWARE REQUIREMENT SPECIFICATION**

**3.3.1What is SRS?**

Software Requirement Specification (SRS) is the starting point of the software developing activity. As system grew more complex it became evident that the goal of the entire system cannot be easily comprehended. Hence the need for the requirement phase arose. The software project is initiated by the client needs. The SRS is the means of translating the ideas of the minds of clients (the input) into a formal document (the output of the requirement phase.)

The SRS phase consists of two basic activities:

**3.3.1.1Problem/Requirement Analysis:**

The process is order and more nebulous of the two, deals with understand the problem, the goal and constraints.

**3.3.1.2 Requirement Specification:**

Here, the focus is on specifying what has been found giving analysis such as representation, specification languages and tools, and checking the specifications are addressed during this activity.

The Requirement phase terminates with the production of the validate SRS document. Producing the SRS document is the basic goal of this phase.

**3.4Document Conventions:**

We have used Times New Roman (text size 12).Bold Font is used for Main Headings (text size of 16). Normal font is used for sub headings (text size of 14).

**Font:** Times New Roman

**Main Heading:** Bold Font

**3.5Intended Audience and Reading Suggestions:**

This document is for better understanding for Remote desktop control. Mainly intended for Head of the Dept., Internal guide, External guide, Staff members, Users and colleagues. This detail given below guides every normal user to how to go through this document for better understanding. The sequence to follow for better understanding is here Purpose, Scope, Features, Operating requirements, Modules present in the project, Advantages, References etc.

**3.5.1 ROLE OF SRS**

The purpose of the Software Requirement Specification is to reduce the communication gap between the clients and the developers. Software Requirement Specification is the medium though which the client and user needs are accurately specified. It forms the basis of software development. A good SRS should satisfy all the parties involved in the system.

**3.6 UML Concepts:**

The Unified Modeling Language (UML) is a standard language for writing software blue prints. The UML is a language for

* Visualizing
* Specifying
* Constructing
* Documenting the artifacts of a software intensive system.

The UML is a language which provides vocabulary and the rules for combining words in that vocabulary for the purpose of communication. A modeling language is a language whose vocabulary and the rules focus on the conceptual and physical representation of a system. Modeling yields an understanding of a system.

**3.6.1 Building Blocks of the UML:**

The vocabulary of the UML encompasses three kinds of building blocks:

* Things
* Relationships
* Diagrams

Things are the abstractions that are first-class citizens in a model; relationships tie these things together; diagrams group interesting collections of things.

**3.6.1.1Things in the UML:**

There are four kinds of things in the UML:

* Structural things
* Behavioral things
* Grouping things
* Annotational things
* **Structural things** are the nouns of UML models. The structural things used in the project design are first, a **class** is a description of a set of objects that share the same attributes, operations, relationships and semantics.

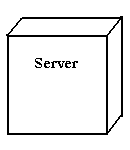
|  |
| --- |
| Window |
| Origin  Size |
| open()  close()  move()  display() |

Second, a **use case** is a description of set of sequence of actions that a system performs that yields an observable result of value to particular actor.



**Fig: Use Cases**

Third, a node is a physical element that exists at runtime and represents a computational resource, generally having at least some memory and often processing capability.



**Fig: Nodes**

* **Behavioral things** are the dynamic parts of UML models. The behavioral thing used is:

**Interaction:**

An interaction is a behavior that comprises a set of messages exchanged among a set of objects within a particular context to accomplish a specific purpose. An interaction involves a number of other elements, including messages, action sequences (the behavior invoked by a message, and links (the connection between objects).



**Fig: Messages**

**3.6.1.2 Relationships in the UML:**

There are four kinds of relationships in the UML:

* Dependency
* Association
* Generalization
* Realization

A **dependency** is a semantic relationship between two things in which a change to one thing may affect the semantics of the other thing (the dependent thing).

**Fig: Dependencies**

An **association** is a structural relationship that describes a set links, a link being a connection among objects. Aggregation is a special kind of association, representing a structural relationship between a whole and its parts.



**Fig: Association**

A **generalization** is a specialization/ generalization relationship in which objects of the specialized element (the child) are substitutable for objects of the generalized element(the parent).



**Fig: Generalization**

A **realization** is a semantic relationship between classifiers, where in one classifier specifies a contract that another classifier guarantees to carry out.



**Fig: Realization**

**3.6.1.3 Sequence Diagrams:**

UML sequence diagrams are used to represent the flow of messages, events and actions between the objects or components of a system. Time is represented in the vertical direction showing the sequence of interactions of the header elements, which are displayed horizontally at the top of the diagram.

Sequence Diagrams are used primarily to design, document and validate the architecture, interfaces and logic of the system by describing the sequence of actions that need to be performed to complete a task or scenario. UML sequence diagrams are useful design tools because they provide a dynamic view of the system behavior which can be difficult to extract from static diagrams or specifications.

* **Actor**

Represents an external person or entity that interacts with the system

**Sequence diagram actor element**

* **Object**

Represents an object in the system or one of its components

**Sequence diagram object element**

* **Unit**

Represents a subsystem, component, unit, or other logical entity in the system (may or may not be implemented by objects)

**Sequence diagram unit element**

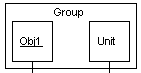
* **Separator**

Represents an interface or boundary between subsystems, components or units (e.g., air interface, Internet, network)

**Sequence diagram separator element**

* **Group**

Groups related header elements into subsystems or components

****

**3.6.1.4Sequence Diagram Body Elements**

* **Action**

Represents an action taken by an actor, object or unit

**Sequence diagram action element**

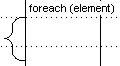
* **Asynchronous Message**

An asynchronous message between header elements

**Sequence diagram asynchronous message element**

* **Block**

A block representing a loop or conditional for a particular header element

****

* **Call Message**

A call (procedure) message between header elements

**Sequence diagram call message element**

* **Create Message**

A "create" message that creates a header element (represented by lifeline going from dashed to solid pattern)

**Sequence diagram create message element**

* **Diagram Link**

Represents a portion of a diagram being treated as a functional block. Similar to a procedure or function call that abstracts functionality or details not shown at this level. Can optionally be linked to another diagram for elaboration.

**Sequence diagram diagram link element**

Else Block Represents an "else" block portion of a diagram block

**Sequence diagram else block element**

* **Message**

A simple message between header elements

**Sequence diagram message element**

* **Return Message**

A return message between header elements

**Sequence diagram return message element**

**Class diagram:**

****

**Use case diagram:**



**State chart:**



**Sequence diagram:**



**Collaboration:**



**3.16 HARDWARE AND SOFTWARE REQUIREMENTS:**

**3.16.1 Software Interfaces:**

We use Java, JSP and Servlets as Programming language for writing the code for the project. and JavaScript for creating the web pages SQLite database server is used for creating the local and global database (server). ECLIPSE IDE for writing the programs. Operating system will be Windows XP.

**3.16.2 Software Requirements**

* Java
* Android SDK
* Eclipse Ganymede/Helios IDE
* Operating System can be Windows XP.

**3.16.3 Hardware Requirements**

* Ram : 1GB Ram and above
* Hard Disk : 50GB and above
* Processor : Dual core and above

Note: For installing this Application in to mobile device, we need android OS Mobile phones.2.2 version android mobiles are preferable.

**Chapter 4**

**IMPLEMENTATION**

**4.7 ANDROID TECHNOLOGY**

**4.7.1 Android:**

Android is a [mobile operating system](http://en.wikipedia.org/wiki/Mobile_operating_system) initially developed by Android Inc. Android was purchased by [Google](http://en.wikipedia.org/wiki/Google) in [2005](http://en.wikipedia.org/wiki/2005). Android is based upon a modified version of the [Linux kernel](http://en.wikipedia.org/wiki/Linux_kernel). Google and other members of the [Open Handset Alliance](http://en.wikipedia.org/wiki/Open_Handset_Alliance) collaborated to develop and release Android to the world. The Android Open Source Project (AOSP) is tasked with the maintenance and further development of Android. Unit sales for Android OS Smartphone ranked first among all [Smartphone](http://en.wikipedia.org/wiki/Smartphone) OS handsets sold in the U.S. in the second and third quarters of 2010, with a third quarter market share of 43.6%.

**4.7.1 The History of Google Android**

The fact that hitherto dedicated devices such as mobile phones can now count themselves among the venerable general-computing platforms is great news for programmers. This new trend makes mobile devices accessible through general-purpose computing languages, which increases the range and market share for mobile applications.  
[**The Android Platform**](http://code.google.com/android/)  embraces the idea of general-purpose computing for handheld devices. It is a comprehensive platform that features a Linux-based operating system stack for managing devices, memory, and processes. Android’s libraries cover telephony, video, graphics, UI programming, and a number of other aspects of the device.

The Android SDK supports most of the Java Platform, Standard Edition (Java SE) except for the Abstract Window Toolkit (AWT) and Swing. In place of AWT and Swing, Android SDK has its own extensive modern UI framework. Because you’re programming your applications in Java, you could expect that you need a Java Virtual Machine (JVM) that is responsible for interpreting the runtime Java byte code. A JVM typically provides the necessary optimization to help Java reach performance levels comparable to compiled languages such as C and C++. Android offers its own optimized JVM to run the compiled Java class files in order to counter the handheld device limitations such as memory, processor speed, and power. This virtual machine is called **the** [**Dalvik virtual machine.**](http://www.dalvikvm.com/)

The familiarity and simplicity of the Java programming language coupled with Android’s extensive class library makes Android a compelling platform to write programs for.

Let us look at how Android arrived on the Mobile OS landscape. Mobile phones use a variety of operating systems such as Symbian OS, Microsoft’s Windows Mobile, Mobile Linux, iPhone operating system (based on Mac OS X), Moblin (from Intel), and many other proprietary operating systems. So far no single operating system has become the de facto standard. The available APIs and environments for developing mobile applications are too restrictive and seem to fall behind when compared to desktop frameworks. This is where Google comes in. The Android platform promised openness, affordability, open source code, and a high-end development framework.

Google [**acquired the startup company Android Inc**](http://www.businessweek.com/technology/content/aug2005/tc20050817_0949_tc024.htm). in 2005 to start the development of the Android Platform The key players at Android Inc. included Andy Rubin, Rich Miner, Nick Sears, and Chris White.

In late 2008 Google released a handheld device called Android Dev Phone 1 that was capable of running Android applications without being tied to any cell phone provider network. The goal of this device (at an approximate cost of $400.00) was to allow developers to experiment with a real device that could run the Android OS without any contracts. At around the same time, Google also released a bug fix, version 1.1 of the OS that is solely based on version 1.0. In releases 1.0 and 1.1 Android did not support soft keyboards, requiring the devices to carry physical keys. Android fixed this issue by releasing the 1.5 SDK in April 2009, along with a number of other features, such as advanced media-recording capabilities, widgets, and live folders.

**4.8 Features:**

**4.8.1 Handset layouts :** The platform is adaptable to larger, [VGA](http://en.wikipedia.org/wiki/Video_Graphics_Array), [2D graphics](http://en.wikipedia.org/wiki/2D_computer_graphics) library, [3D graphics](http://en.wikipedia.org/wiki/3D_computer_graphics) library based on [OpenGL ES](http://en.wikipedia.org/wiki/OpenGL_ES) 2.0 specifications, and traditional Smartphone layouts.

**4.8.2 Storage:**[SQLite](http://en.wikipedia.org/wiki/SQLite), a lightweight [relational database](http://en.wikipedia.org/wiki/Relational_database), is used for data storage purposes

**4.8.3 Connectivity** : Android supports connectivity technologies including [GSM](http://en.wikipedia.org/wiki/GSM)/[EDGE](http://en.wikipedia.org/wiki/Enhanced_Data_Rates_for_GSM_Evolution), [IDEN](http://en.wikipedia.org/wiki/Integrated_Digital_Enhanced_Network), [CDMA](http://en.wikipedia.org/wiki/Code_division_multiple_access), [EV-DO](http://en.wikipedia.org/wiki/Evolution-Data_Optimized), [UMTS](http://en.wikipedia.org/wiki/Universal_Mobile_Telecommunications_System), [Bluetooth](http://en.wikipedia.org/wiki/Bluetooth), [Wi-Fi](http://en.wikipedia.org/wiki/Wi-Fi), [LTE](http://en.wikipedia.org/wiki/LTE_Advanced), and [WiMAX](http://en.wikipedia.org/wiki/WiMAX).

**4.8.4 Messaging** : [SMS](http://en.wikipedia.org/wiki/SMS) and [MMS](http://en.wikipedia.org/wiki/Multimedia_Messaging_Service) are available forms of messaging, including threaded [text messaging](http://en.wikipedia.org/wiki/Text_messaging) and now Android Cloud to Device Messaging Framework ([C2DM](http://en.wikipedia.org/w/index.php?title=C2DM&action=edit&redlink=1)) is also a part of Android Push Messaging service.

**4.8.5 Web browser**: The web browser available in Android is based on the open-source WebKit layout engine, coupled with [Chrome](http://en.wikipedia.org/wiki/Google_Chrome)'s [V8](http://en.wikipedia.org/wiki/V8_%28JavaScript_engine%29) JavaScript engine. The browser scores a 93/100 on the [Acid3](http://en.wikipedia.org/wiki/Acid3) Test.

**4.8.6 Java support:**While Android applications are written in [Java](http://en.wikipedia.org/wiki/Java_%28programming_language%29), there's no [Java Virtual Machine](http://en.wikipedia.org/wiki/Java_Virtual_Machine) in the platform and Java byte code is not executed. Java classes get recompiled into [Dalvik executable](http://en.wikipedia.org/w/index.php?title=Dalvik_executable&action=edit&redlink=1) and run on [Dalvik virtual machine](http://en.wikipedia.org/wiki/Dalvik_virtual_machine). Dalvik is a specialized virtual machine designed specifically for Android and optimized for battery-powered mobile devices with limited memory and CPU. J2ME support can be provided via third-party-application such as the J2ME MIDP Runner.[[67]](http://en.wikipedia.org/wiki/Android_%28operating_system%29#cite_note-66)

**4.8.7 Media support Android supports** : the following audio/video/still media formats: [WebM](http://en.wikipedia.org/wiki/WebM), [H.263](http://en.wikipedia.org/wiki/H.263), [H.264](http://en.wikipedia.org/wiki/H.264) (in [3GP](http://en.wikipedia.org/wiki/3GP) or [MP4](http://en.wikipedia.org/wiki/MP4)[container](http://en.wikipedia.org/wiki/Container_format_%28digital%29)), [MPEG-4 SP](http://en.wikipedia.org/wiki/MPEG-4_Part_2), [AMR](http://en.wikipedia.org/wiki/Adaptive_multi-rate_compression), [AMR-WB](http://en.wikipedia.org/wiki/AMR-WB) (in 3GP container).

**4.8.8 Streaming media support:** RTP/RTSP streaming (3GPP PSS, ISMA), HTML progressive download (HTML5 <video> tag). Adobe Flash Streaming (RTMP) is supported through Adobe Flash Player plugin. Apple HTTP Live Streaming is supported through stagefright media player by default since 2.3(Gingerbread). Microsoft Smooth Streaming is planned to be supported through the awaited port of Silverlight plugin to Android. Adobe Flash HTTP Dynamic Streaming is planned to be supported through an upgrade of the Flash plugin.

**4.8.9 Additional hardware support** : Android can use video/still cameras, [touch screens](http://en.wikipedia.org/wiki/Touchscreen), [GPS](http://en.wikipedia.org/wiki/Global_Positioning_System), [accelerometers](http://en.wikipedia.org/wiki/Accelerometer), [gyroscopes](http://en.wikipedia.org/wiki/Gyroscope), [magnetometers](http://en.wikipedia.org/wiki/Magnetometer), [proximity](http://en.wikipedia.org/wiki/Proximity_sensor) and [pressure sensors](http://en.wikipedia.org/wiki/Pressure_sensor), [thermometers](http://en.wikipedia.org/wiki/Thermometer), accelerated 2D [bit blits](http://en.wikipedia.org/wiki/Bit_blit) (with hardware orientation, scaling, pixel format conversion) and accelerated 3D graphics.

**4.8.10 Development environment:** Includes a device emulator, tools for [debugging](http://en.wikipedia.org/wiki/Debugging), memory and [performance profiling](http://en.wikipedia.org/wiki/Software_performance_analysis). The [integrated development environment](http://en.wikipedia.org/wiki/Integrated_development_environment) (IDE) is Eclipse (currently 3.4 or 3.5) using the Android Development Tools (ADT) Plugin.

**4.8.11 Market**: Like many phone-based application stores, the [Android Market](http://en.wikipedia.org/wiki/Android_Market) is a catalog of applications that can be downloaded and installed to target hardware over-the-air, without the use of a PC. Originally only free applications were supported. Paid-for applications have been available on the Android Market in the United States since 19 February 2009. The Android Market has been expanding rapidly. As of August 3, 2010, it had over 100,000 Android applications for download. There are other markets, such as SlideME and Getjar. Multi-touch Android has native support for [multi-touch](http://en.wikipedia.org/wiki/Multi-touch) which was initially made available in handsets such as the [HTC Hero](http://en.wikipedia.org/wiki/HTC_Hero). The feature was originally disabled at the kernel level (possibly to avoid infringing Apple's patents on touch-screen technology). Google has since released an update for the [Nexus One](http://en.wikipedia.org/wiki/Nexus_One) and the [Motorola Droid](http://en.wikipedia.org/wiki/Motorola_Droid) which enables multi-touch natively.

**4.8.12 Multi-touch:** Android has native support for [multi-touch](http://en.wikipedia.org/wiki/Multi-touch) which was initially made available in handsets such as the [HTC Hero](http://en.wikipedia.org/wiki/HTC_Hero). The feature was originally disabled at the kernel level (possibly to avoid infringing Apple's patents on touch-screen technology). Google has since released an update for the [Nexus One](http://en.wikipedia.org/wiki/Nexus_One) and the [Motorola Droid](http://en.wikipedia.org/wiki/Motorola_Droid) which enables multi-touch natively.

**4.8.13 Bluetooth:** Support for [A2DP](http://en.wikipedia.org/wiki/A2DP) and [AVRCP](http://en.wikipedia.org/wiki/AVRCP) were added in version 1.5; sending files (OPP) and accessing the phone book (PBAP) were added in version 2.0; and voice dialing and sending contacts between phones were added in version 2.2.

**4.8.14 Video calling** : The mainstream Android version doesn't support video calling, however some handsets could have a customized version of the operating system which supports it, either via [UMTS](http://en.wikipedia.org/wiki/UMTS) network (like the [Samsung i9000 Galaxy S](http://en.wikipedia.org/wiki/Samsung_i9000_Galaxy_S)) or over IP.

**4.8.15 Multitasking:** Multitasking of applications is available.

**Voice based features:** Google search through Voice is available as Search Input since initial release. Also launched Voice actions supported on Android 2.2 onwards.

**4.8.16 Tethering :** Android supports tethering, which allows a phone to be used as a wireless/wired hotspot (All 2.2 Froyo phones, unofficial on phones running 1.6 or higher via applications available in the Android Market, e.g. PdaNet). To allow a laptop to share the 3G connection on an Android phone software may need to be installed on both the phone and the laptop

**4.9 Hardware running Android**

The Android OS can be used as an operating system for cellphones, netbooks and tablet PCs, including the [Dell Streak](http://en.wikipedia.org/wiki/Dell_Streak), [Samsung Galaxy Tab](http://en.wikipedia.org/wiki/Samsung_Galaxy_Tab) and other devices. The world's first TV running Android, called Scandinavia, has also been launched by the company People of Lava.

The first commercially available phone to run the Android operating system was the [HTC Dream](http://en.wikipedia.org/wiki/HTC_Dream), released on 22 October 2008.

**4.10 Software development**

The early feedback on developing applications for the Android platform was mixed. Issues cited include bugs, lack of documentation, inadequate QA infrastructure, and no public issue-tracking system. (Google announced an issue tracker on 18 January 2008.) In December 2007, MergeLab mobile startup founder Adam MacBeth stated, "Functionality is not there, is poorly documented or just doesn't work... It's clearly not ready for prime time." Despite this, Android-targeted applications began to appear the week after the platform was announced. The first publicly available application was the [Snake game](http://en.wikipedia.org/wiki/Snake_%28video_game%29). The [Android Dev Phone](http://en.wikipedia.org/wiki/Android_Dev_Phone) is a [SIM](http://en.wikipedia.org/wiki/Subscriber_Identity_Module)-unlocked and hardware-unlocked device that is designed for advanced developers. While developers can use regular consumer devices purchased at retail to test and use their applications, some developers may choose not to use a retail device, preferring an unlocked or no-contract device.

**4.10.1 Software development kit**

The Android [software development kit](http://en.wikipedia.org/wiki/Software_development_kit) (SDK) includes a comprehensive set of development tools.[[85]](http://en.wikipedia.org/wiki/Android_%28operating_system%29#cite_note-84) These include a [debugger](http://en.wikipedia.org/wiki/Debugger), [libraries](http://en.wikipedia.org/wiki/Software_library), a handset [emulator](http://en.wikipedia.org/wiki/Emulator) (based on [QEMU](http://en.wikipedia.org/wiki/QEMU)), documentation, sample code, and tutorials. Currently supported development platforms include computers running [Linux](http://en.wikipedia.org/wiki/Linux_kernel) (any modern desktop [Linux distribution](http://en.wikipedia.org/wiki/List_of_GNU/Linux_distributions)), [Mac OS X](http://en.wikipedia.org/wiki/Mac_OS_X) 10.4.9 or later, [Windows XP](http://en.wikipedia.org/wiki/Windows_XP) or later. The officially supported [integrated development environment](http://en.wikipedia.org/wiki/Integrated_development_environment) (IDE) is [Eclipse](http://en.wikipedia.org/wiki/Eclipse_%28software%29) (currently 3.4 or 3.5) using the Android Development Tools (ADT) Plugin, though developers may use any text editor to edit Java and XML files then use [command line](http://en.wikipedia.org/wiki/Command_line) tools ([Java Development Kit](http://en.wikipedia.org/wiki/Java_Development_Kit) and [Apache Ant](http://en.wikipedia.org/wiki/Apache_Ant) are required) to create, build and debug Android applications as well as control attached Android devices (e.g., triggering a reboot, installing software package(s) remotely).

**4.1.1 JAVASCRIPT METHODS:**

**4.1.1.1Writeln:**

Document.writeln() is a method, which is used to write some text to the current web page.

**4.1.1.2 onClick:**

Occurs when an element is clicked.

**4.1.1.3 onLoad:**

Occurs when the page loads.

**4.1.1.4 onMouseDown:**

Occurs when a mouse button goes down.

**4.1.1.5 onMouseMove:**

Occurs when the mouse moves.

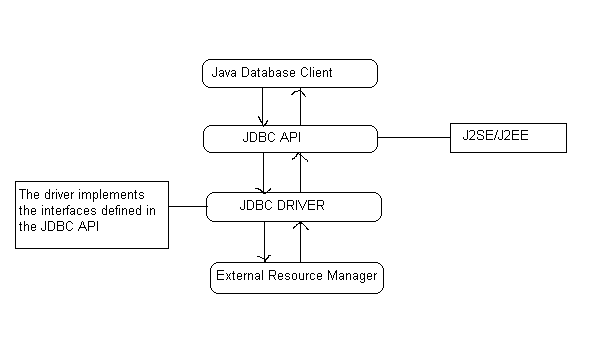
**4.1.1.6 onUnload:**

Occurs when a page is unloaded.

**4.2 JDBC DRIVERS:**

The JDBC API only defines interfaces for objects used for performing various database-related tasks like opening and closing connections, executing SQL commands, and retrieving the results. We all write our programs to interfaces and not implementations. Either the resource manager vendor or a third party provides the implementation classes for the standard JDBC interfaces. These software implementations are called JDBC drivers. JDBC drivers

transform the standard JDBC calls to the external resource manager-specific API calls. The diagram below depicts how a database client written in java accesses an external resource manager using the JDBC API and JDBC driver:



Depending on the mechanism of implementation, JDBC drivers are broadly classified into four types.

**TYPE1:**

Type1 JDBC drivers implement the JDBC API on top of a lower level API like ODBC. These drivers are not generally portable because of the independency on native libraries. These drivers translate the JDBC calls to ODBC calls and ODBC sends the request to external data source using native library calls. The JDBC-ODBC driver that comes with the software distribution for J2SE is an example of a type1 driver.

**TYPE2:**

Type2 drivers are written in mixture of java and native code. Type2 drivers use vendors specific native APIs for accessing the data source. These drivers transform the JDBC calls to vendor specific calls using the vendor’s native library.

These drivers are also not portable like type1 drivers because of the dependency on native code.

**TYPE3:**

Type3 drivers use an intermediate middleware server for accessing the external data sources. The calls to the middleware server are database independent. However, the middleware

server makes vendor specific native calls for accessing the data source. In this case, the driver is purely written in java.

**TYPE4:**

Type4 drivers are written in pure java and implement the JDBC interfaces and translate the JDBC specific calls to vendor specific access calls. They implement the data transfer and network protocol for the target resource manager. Most of the leading database vendors provide type4 drivers for accessing their database servers.

**4.2.1 DRIVER MANAGER AND DRIVER:**

The java.sql package defines an interface called Java.sql.Driver that makes to be implemented by all the JDBC drivers and a class called java.sql.DriverManager that acts as the interface to the database clients for performing tasks like connecting to external resource managers, and setting log streams. When a JDBC client requests the DriverManager to make a connection to an external resource manager, it delegates the task to an approate driver class implemented by the JDBC driver provided either by the resource manager vendor or a third party.

**4.2.2 JAVA.SQL.DRIVERMANAGER:**

The primary task of the class driver manager is to manage the various JDBC drivers register. It also provides methods for:

* Getting connections to the databases.
* Managing JDBC logs.
* Setting login timeout.

**4.2.3 MANAGING DRIVERS:**

JDBC clients specify the JDBC URL when they request a connection. The driver manager can find a driver that matches the request URL from the list of register drivers and delegate the connection request to that driver if it finds a match JDBC URLs normally take the following format:

**<Protocol>:<sub-protocol>:<resource>**

The protocol is always jdbc and the sub-protocol and resource depend on the type of resource manager. The URL for postgreSQL is in the format:

**Jdbc: postgres ://< host> :< port>/<database>**

Here host is the host address on which post master is running and database is the name of the database to which the client wishes to connect.

**4.3MANAGING CONNECTION:**

DriverManager class is responsible for managing connections to the databases:

Public static Connection getConnection (String url,Properties info) throws SQLException

This method gets a connection to the database by the specified JDBC URL using the specified username and password. This method throws an instance of SQLException if a database access error occurs.

**4.3.1 CONNECTIONS:**

The interface java.sql.Connection defines the methods required for a persistent Connection to the database. The JDBC driver vendor implements this interface. A database ‘vendor-neutral’ client never uses the implementation class and will always use only the interface. This interface defines methods for the following tasks:

* Statements, prepared statements, and callable statements are the different types of statements for issuing sql statements to the database by the JDBC clients.
* For getting and setting auto-commit mode.

**4.3.2 CREATING STATEMENTS:**

The interface java.sql.Connection defines a set of methods for creating database statements. Database statements are used for sending SQL statements to the database:

Public Statement createStatement () throws SQLExceptionThis method is used for creating instances of the interface java.sql.Statement. This interface can be used for sending SQL statements to the database. The interface java.sql.Statement is normally used for sending SQL statements that don’t take any arguments. This method throws an instance of SQLException if a database access error occurs: Public Statement createStatement (intresType, intresConcurrency) throws SQLException

**4.4 JDBC RESULT SETS:**

A JDBC resultset represents a two dimentional array of data produced as a result of executing SQL SELECT statements against databases using JDBC statements. JDBC resultsets are represented by the interface java.sql.ResultSet. The JDBC vendor provider provides the implementation class for this interface.

**4.4.2 ACCESSING RESULTSET DATA:**

#### Method name and Purpose

public Boolean getBoolean (int i)

Gets the data in the specified column as a boolean.

public Boolean getBoolean (String col)

Gets the data in the specified column as a boolean.

public int getInt(int I)

Gets the data in the specied column as an int.

**4.4.3 STATEMENT:**

The interface java.sql.Stament is normally used for sending SQL statements that do not have IN or OUT parameters. The JDBC driver vendor provides the implementation class for this interface. The common methods required by the different JDBC statements are defined in this interface. The methods defined by java.sql. Statement can be broadly categorized as follows:

* Executing SQL statements
* Querying results and resultsets
* Handling SQL batches
* Other miscellaneous methods

The interface java.sql.statements defines methods for executing different SQL statements like SELECT, UPDATE, INSERT, DELETE, and CREATE.

Public Resultset execute Query (string sql) throws SQLException

The following figure shows how the DriverManager, Driver, Connection, Statement, ResultSet classes are connected.

DriverManager

Driver

Driver

Layer

Application

Layer

Connection

Prepared Statement

Statement

Callable Statement

Result Set

Result Set

Result Set

**4.5 JAVA SERVER PAGES (JSP)**

**4.5.1 INTRODUCTION:**

Java Server Pages (JSP) technology enables you to mix regular, static HTML with dynamically generated content. You simply write the regular HTML in the normal manner, using familiar Web-page-building tools. You then enclose the code for the dynamic parts in special tags, most of which start with <% and end with %>.

**4.5.5 TYPES OF JSP SCRIPTING ELEMENTS:**

JSP scripting elements allow you to insert Java code into the servlet that will be generated from the JSP page. There are three forms:

* **Expressions** of the form <%=Java Expression %>, which are evaluated and inserted into the servlet’s output.
* **Sciptlets**of the form <%Java code %>, which are inserted into the servlet’s\_jspService method (called by service).
* **Declarations** of the form<%! Field/Method Declaration %>, which are inserted into the body of the servlet class, outside any existing methods.

**4.5.6 USING JSP EXPRESSIONS:**

A JSP element is used to insert values directly into the output. It has the following form:

<%= Java Expression %>

The expression is evaluated, converted to a string, and inserted in the page. This evaluation is performed at runtime (when the page is requested) and thus has full access to the information about the request. For example, the following shows the date/time that the page was requested.

Current time: <%=new java.util.Date () %>

**4.5.7 PREDEFINED VARIABLES:**

To simplify expressions we can use a number of predefined variables (or “implicit objects”). The specialty of these variables is that, the system simple tells what names it will use for the local variables in \_jspService.The most important ones of these are:

* **request**, the HttpServletRequest.
* **response**, the HttpServletResponse.
* **session,** the HttpSession associated with the request
* **out,** the writer used to send output to clients.
* **application,** the ServletContext. This is a data structure shared by all servlets and JSP pages in the web application and is good for storing shared data.

Here is an example:

Your hostname: <%= **request.**getRemoteHost () %>

**4.5.8 COMPARING SERVLETS TO JSP PAGES**

JSP works best when the structure of pages is fixed but the values at various places need to be computed dynamically. If the structure of the page is dynamic, JSP is less beneficial. Some times servlets are better in such a case. If the page consists of binary data or has little static content, servlets are clearly superior. Sometimes the answer is neither servlets nor JSP alone, but rather a combination of both.

**4.5.9 WRITING SCRIPTLETS**

If you want to do something more complex than output the value of a simple expression .JSP scriptlets let you insert arbitrary code into the servlet’s jspService method. Scriptlets have the following form:

<% Java code %>

Scriptlets have access to the same automatically defined variables as do expressions (request, response, session, out , etc ) .So for example you want to explicitly send output of the resultant page , you could use the out variable , as in the following example:

<% String queryData = request.getQueryString ();

out.println (“Attached GET data: “+ queryData);

%>

**4.5.10 SCRIPTLET EXAMPLE:**

As an example of code that is too complex for a JSP expression alone, a JSP page that uses the bgColor request parameter to set the background color of the page .Simply using

<BODY BGCOLOR=”<%= request.getParameter (“bgcolor”) %> “>

would violate the cardinal rule of reading form data.

**4.5.11 USING DECLARATIONS**

A JSP declaration lets you define methods or fields that get inserted into the main body of the servlet class .A declaration has the following form:

<%! Field or Method Definition %>

Since declarations do not generate output, they are normally used in conjunction with JSP expressions or scriptlets. In principle, JSP declarations can contain field (instance variable) definitions, method definitions, inner class definitions, or even static initializer blocks: anything that is legal to put inside a class definition but outside any existing methods. In practice declarations almost always contain field or method definitions.

We should not use JSP declarations to override the standard servlet life cycle methods. The servlet into which the JSP page gets translated already makes use of these methods. There is no need for declarations to gain access to service, doget, or dopost, since calls to service are automatically dispatched to \_jspService , which is where code resulting from expressions and scriptlets is put. However for initialization and cleanup, we can use jspInit and jspDestroy.

**4.6JAKARTA TOMCAT**

Tomcat is the Servlet/JSP container. Tomcat implements the Servlets and Java Server Pages 2.0 specification. It also includes many additional features that make it a useful platform for developing and deploying web applications and web services.

**4.6.2 DIRECTORIES AND FILES:**

**/bin** – Startup, shutdown, and other scripts. The \*.sh files (for Unix systems) are functional duplicates of the \*.bat files (for Windows systems). Since the Win32 command-line lacks certain functionality, there are some additional files in here.

**/conf –** Configuration files and related DTDs. The most important file in here is server.xml. It is the main configuration file for the container.

**/logs –** Log files are here by default.

**/webapps –** This is where webapps go\

**4.6.3 INSTALLATION:**

Tomcat will operate under any Java Development Kit (JDK) environment that provides a JDK 1.2 (also known as Java2 Standard Edition, or J2SE) or later platform. JDK is needed so that servlets, other classes, and JSP pages can be compiled.

**4.11 Coding:**

**FIRSTACTIVITY.JAVA**

**package com.anwar.antitheft;**

**import android.os.Bundle;**

**import android.content.SharedPreferences;**

**import android.content.Context;**

**import android.app.Activity;**

**import android.telephony.TelephonyManager;**

**import android.view.Menu;**

**import android.view.View;**

**import android.view.View.OnClickListener;**

**import android.view.Window;**

**import android.widget.Button;**

**import android.widget.EditText;**

**import android.widget.Toast;**

**public class FirstActivity extends Activity {**

**EditText word,number;**

**Button ok,cancel;**

**public static final String MyPREFERENCES = "MyPrefs1" ;**

**public static final String Word = "wordKey";**

**public static final String Phone = "phoneKey";**

**SharedPreferences sharedpreferences;**

**@Override**

**protected void onCreate(Bundle savedInstanceState) {**

**super.onCreate(savedInstanceState);**

**this.requestWindowFeature(Window.FEATURE\_NO\_TITLE);**

**setContentView(R.layout.activity\_first);**

**word=(EditText)findViewById(R.id.magicWord);**

**number=(EditText)findViewById(R.id.phNumber);**

**ok=(Button)findViewById(R.id.button);**

**sharedpreferences = getSharedPreferences(MyPREFERENCES, Context.MODE\_PRIVATE);**

**word.setText(sharedpreferences.getString("wordKey", ""));**

**number.setText(sharedpreferences.getString("phoneKey", ""));**

**ok.setOnClickListener(new OnClickListener() {**

**@Override**

**public void onClick(View arg0) {**

**// TODO Auto-generated method stub**

**String wd = word.getText().toString();**

**String ph = number.getText().toString();**

**TelephonyManager tMgr=(TelephonyManager)getSystemService(Context.TELEPHONY\_SERVICE);**

**String mPhoneNumber = tMgr.getLine1Number();**

**SharedPreferences.Editor editor = sharedpreferences.edit();**

**editor.putString(Word, wd);**

**editor.putString(Phone, ph);**

**editor.putString("phonenumber", mPhoneNumber);**

**editor.commit();**

**Toast.makeText(FirstActivity.this,"Successfully Stored",Toast.LENGTH\_LONG).show();**

**finish();**

**}**

**});**

**}**

**@Override**

**public boolean onCreateOptionsMenu(Menu menu) {**

**// Inflate the menu; this adds items to the action bar if it is present.**

**getMenuInflater().inflate(R.menu.activity\_first, menu);**

**return true;**

**}}**

**MAINACTVITY.JAVA**

**package com.anwar.antitheft;**

**import android.os.Bundle;**

**import android.content.SharedPreferences;**

**import android.content.Context;**

**import android.app.Activity;**

**import android.telephony.TelephonyManager;**

**import android.view.Menu;**

**import android.view.View;**

**import android.view.View.OnClickListener;**

**import android.view.Window;**

**import android.widget.Button;**

**import android.widget.EditText;**

**import android.widget.Toast;**

**public class FirstActivity extends Activity {**

**EditText word,number;**

**Button ok,cancel;**

**public static final String MyPREFERENCES = "MyPrefs1" ;**

**public static final String Word = "wordKey";**

**public static final String Phone = "phoneKey";**

**SharedPreferences sharedpreferences;**

**@Override**

**protected void onCreate(Bundle savedInstanceState) {**

**super.onCreate(savedInstanceState);**

**this.requestWindowFeature(Window.FEATURE\_NO\_TITLE);**

**setContentView(R.layout.activity\_first);**

**word=(EditText)findViewById(R.id.magicWord);**

**number=(EditText)findViewById(R.id.phNumber);**

**ok=(Button)findViewById(R.id.button);**

**sharedpreferences = getSharedPreferences(MyPREFERENCES, Context.MODE\_PRIVATE);**

**word.setText(sharedpreferences.getString("wordKey", ""));**

**number.setText(sharedpreferences.getString("phoneKey", ""));**

**ok.setOnClickListener(new OnClickListener() {**

**@Override**

**public void onClick(View arg0) {**

**// TODO Auto-generated method stub**

**String wd = word.getText().toString();**

**String ph = number.getText().toString();**

**TelephonyManager tMgr=(TelephonyManager)getSystemService(Context.TELEPHONY\_SERVICE);**

**String mPhoneNumber = tMgr.getLine1Number();**

**SharedPreferences.Editor editor = sharedpreferences.edit();**

**editor.putString(Word, wd);**

**editor.putString(Phone, ph);**

**editor.putString("phonenumber", mPhoneNumber);**

**editor.commit();**

**Toast.makeText(FirstActivity.this,"Successfully Stored",Toast.LENGTH\_LONG).show();**

**finish();**

**}**

**});**

**}**

**@Override**

**public boolean onCreateOptionsMenu(Menu menu) {**

**// Inflate the menu; this adds items to the action bar if it is present.**

**getMenuInflater().inflate(R.menu.activity\_first, menu);**

**return true;**

**}}**

**MENUACTIVITY.JAVA**

**package com.anwar.antitheft;**

**import android.app.Activity;**

**import android.content.Intent;**

**import android.os.Bundle;**

**import android.view.Menu;**

**import android.view.View;**

**import android.view.View.OnClickListener;**

**import android.widget.Button;**

**public class MenuActivity extends Activity {**

**@Override**

**protected void onCreate(Bundle savedInstanceState) {**

**super.onCreate(savedInstanceState);**

**setContentView(R.layout.activity\_menu);**

**Button bt = (Button) findViewById(R.id.button1);**

**Button bt2 = (Button) findViewById(R.id.button2);**

**bt.setOnClickListener(new OnClickListener() {**

**@Override**

**public void onClick(View arg0) {**

**// TODO Auto-generated method stub**

**Intent it = new Intent(MenuActivity.this, FirstActivity.class);**

**startActivity(it);**

**}**

**});**

**bt2.setOnClickListener(new OnClickListener() {**

**@Override**

**public void onClick(View arg0) {**

**// TODO Auto-generated method stub**

**Intent it = new Intent(MenuActivity.this, SimActivity.class);**

**startActivity(it);**

**}**

**});**

**}**

**@Override**

**public boolean onCreateOptionsMenu(Menu menu) {**

**// Inflate the menu; this adds items to the action bar if it is present.**

**getMenuInflater().inflate(R.menu.activity\_main, menu);**

**return true;**

**}**

**}**

**SIMACTIVITY.JAVA**

**package com.anwar.antitheft;**

**import java.io.IOException;**

**import java.util.ArrayList;**

**import android.app.Activity;**

**import android.content.Context;**

**import android.database.SQLException;**

**import android.database.sqlite.SQLiteDatabase;**

**import android.os.Bundle;**

**import android.telephony.TelephonyManager;**

**import android.view.Menu;**

**import android.view.View;**

**import android.view.View.OnClickListener;**

**import android.widget.Button;**

**import android.widget.EditText;**

**import android.widget.Toast;**

**public class SimActivity extends Activity {**

**Dbhandler myDbHelper;**

**SQLiteDatabase Mydatabase;**

**ArrayList<String> aa;**

**@Override**

**protected void onCreate(Bundle savedInstanceState) {**

**super.onCreate(savedInstanceState);**

**setContentView(R.layout.activity\_sim);**

**Button bt = (Button) findViewById(R.id.button1);**

**final EditText et = (EditText) findViewById(R.id.editText1);**

**this.myDbHelper = new Dbhandler(this);**

**FetchingData();**

**Mydatabase = myDbHelper.getReadableDatabase();**

**aa = this.myDbHelper.getModes(Mydatabase);**

**System.out.println("values of Db " + aa);**

**String s1 = aa.toString();**

**s1 = s1.substring(1, s1.length() - 1).trim();**

**et.setText(s1);**

**bt.setOnClickListener(new OnClickListener() {**

**@Override**

**public void onClick(View arg0) {**

**// TODO Auto-generated method stub**

**String stxt = et.getText().toString();**

**TelephonyManager telephoneMgr = (TelephonyManager) getSystemService(Context.TELEPHONY\_SERVICE);**

**String phoneNumber = telephoneMgr.getLine1Number();**

**myDbHelper = new Dbhandler(SimActivity.this);**

**FetchingData();**

**Mydatabase = myDbHelper.getReadableDatabase();**

**Mydatabase.execSQL("update SimNumber set SimNo='" + phoneNumber**

**+ "' where SNO='1'");**

**Mydatabase.execSQL("update PhoneNumber set PhoneNumberr='"**

**+ stxt + "' where SNO='1'");**

**Toast.makeText(getApplicationContext(), "Updated Successfully",**

**Toast.LENGTH\_LONG).show();**

**}**

**});**

**}**

**private void FetchingData() {**

**// TODO Auto-generated method stub**

**try {**

**myDbHelper.onCreateDataBase();**

**}**

**catch (IOException ioe) {**

**throw new Error("Unable to create database");**

**}**

**try**

**{**

**myDbHelper.openDataBase();**

**Mydatabase = myDbHelper.getWritableDatabase();**

**System.out.println("executed");**

**} catch (SQLException sqle) {**

**throw sqle;**

**}**

**// TODO Auto-generated method stub**

**}**

**}**

**5 TESTING**

**5.1 SOFTWARE TESTING**

Software testing is a critical element of software quality assurance and represents the ultimate review of specification, design and code generation**.**

**5.1.1 TESTING OBJECTIVES**

* + To ensure that during operation the system will perform as per specification.
  + TO make sure that system meets the user requirements during operation
  + To make sure that during the operation, incorrect input, processing and output will be detected
  + To see that when correct inputs are fed to the system the outputs are correct
  + To verify that the controls incorporated in the same system as intended
  + Testing is a process of executing a program with the intent of finding an error
  + A good test case is one that has a high probability of finding an as yet undiscovered error.

The software developed has been tested successfully using the following testing strategies and any errors that are encountered are corrected and again the part of the program.

**5.2 TEST CASE DESIGN:**

**5.2.1 White box testing**

White box testing is a testing case design method that uses the control structure of the procedure design to derive test cases. All independents path in a module are exercised at least once, all logical decisions are exercised at once, execute all loops at boundaries and within their operational bounds exercise internal data structure to ensure their validity. Here the customer is given three chances to enter a valid choice out of the given menu. After which the control exits the current menu.

**5.2.2 Black Box Testing**

Black Box Testing attempts to find errors in following areas or categories, incorrect or missing functions, interface error, errors in data structures, performance error and initialization and termination error. Here all the input data must match the data type to become a valid entry.

The following are the different tests at various levels:

**5.2.3 Unit Testing:**

Unit testing is essentially for the verification of the code produced during the coding phase and the goal is test the internal logic of the module/program. In the Generic code project, the unit testing is done during coding phase of data entry forms whether the functions are working properly or not. In this phase all the drivers are tested they are rightly connected or not.

**5.2.4 Integration Testing:**

All the tested modules are combined into sub systems, which are then tested. The goal is to see if the modules are properly integrated, and the emphasis being on the testing interfaces between the modules. In the generic code integration testing is done mainly on table creation module and insertion module.

**5.2.5 Validation Testing**

This testing concentrates on confirming that the software is error-free in all respects. All the specified validations are verified and the software is subjected to hard-core testing. It also aims at determining the degree of deviation that exists in the software designed from the specification; they are listed out and are corrected.

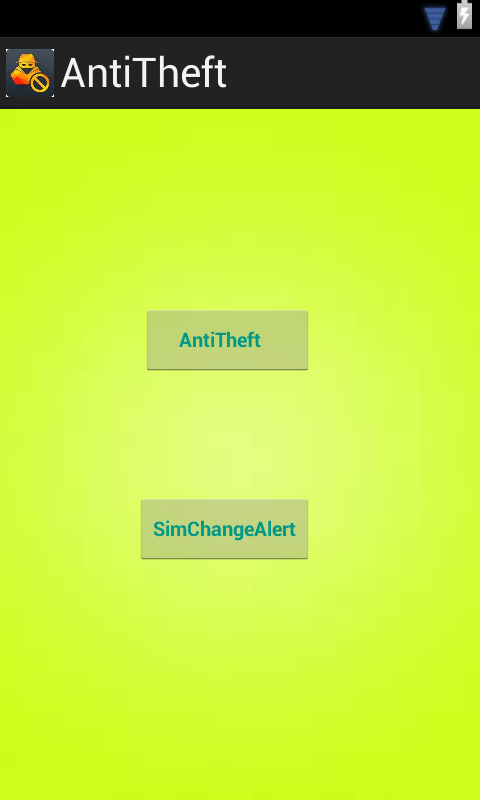
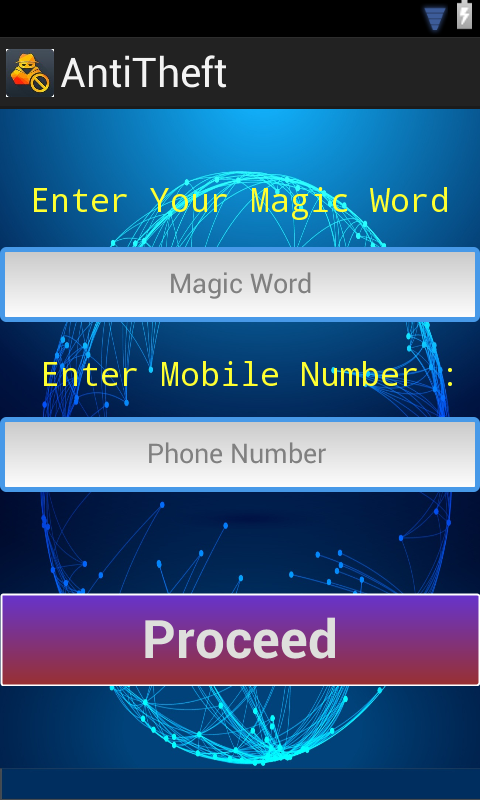
**5.2.6 System Testing**

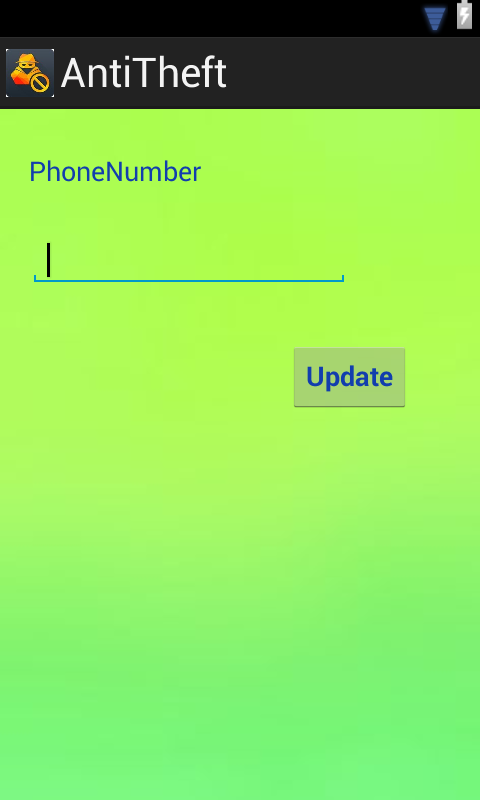
This testing is a series of different tests whose primary is to fully exercise the computer-based system. This involves:

* Implementing the system in a simulated production environment and testing it.
* Introducing errors and testing for error handling.

**Chapter 6**

**SCREEN SHOTS**

**** 

****

**Chapter 7**

**CONCLUSION:**

This system will enable the tracking of the device using its Global Positioning System (GPS) or Global Navigation Satellite System (GLONASS) functionality. Over the past decade more and more phones have begun integrating either the GPS or GLONASS or both systems on board to provide navigation assistance. These systems are able to lock on to satellites and provide accurate location details within seconds. The Mobile Phone Tracking system will be able to activate this feature in the phone and locate the user on a map. The system will be built with a XML front end and can be accessed after providing authorized user credentials.

**BIBLIOGRAPHY**

[www.eci.gov.in](http://www.eci.gov.in)

[www.google.com](http://www.google.com)

[www.apeci.com](http://www.apeci.com)

We will be using the information collected from websites for android

<http://developer.android.com/index.html>

<http://www.android-trainer.com/>

<http://stackoverflow.com/>

<http://www.google.co.in/>