**1. INTRODUCTION**

**1.1 MOTIVATION**

In this project we present an automated sms backup system with real time customer feedback. This system is convenient, effective, and easy and by implementing this project in the present system, the user can overcome the problems faced regarding their sms storage needs in an efficient manner.

**1.2 PROBLEM DEFINITION**

While getting messages through friends or family members the main problem is message backup. To overcome such problem we want to develop a mobile application through which the user can restore the message. The proposed application is to develop a system that offers news services through web application. Once the application is downloaded and installed by the user, he/she should register to it and he can access the service by providing the unique username and password.

User need to provide their username, password so that the message is delivered directly to your Phone will be on database process takes place between the customer and the person who delivers the message to us from other end. The user gets a pop-up message while getting new message where the messages are available. These pop-up messages will inform the user about the existence of this application in the mobile device. This application provides more comfort for the users storing message and to avoid the internet.

**1.3 LIMITATIONS OF PROJECT**

This Project is a type Mobile Application site because here the process of storing and delivering takes place where service provider will provide the list of messages that are available in the database selected by the user so that he/she can select the message items provided by that particular database. Here in this application the user is provided with a wide range of items to store with respective to the mobile and their time. The android application on user’s mobile will have all the menu details. The received details from customer’s mobile are wirelessly updated in central database and subsequently sent to application.

**1.4 OBJECTIVE OF PROJECT**

* Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.
* It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.
* When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in maize of instant. Thus the objective of input design is to create an input layout that is easy to follow.

**2. LITERATURE SURVEY**

**2.1 INTRODUCTION**

Literature [survey](http://www.blurtit.com/q876299.html) is the most important step in software development process. Before developing the tool it is necessary to determine the time factor, economy n company strength. Once these things are satisfied, ten next steps are to determine which operating system and language can be used for developing the tool. Once the [programmers](http://www.blurtit.com/q876299.html) start building the tool the programmers need lot of external support. This support can be obtained from senior programmers, from [book](http://www.blurtit.com/q876299.html) or from websites. Before building the system the above consideration are taken into account for developing the proposed system.

**2.2 EXISTING SYSTEM**

User needs to use the default application which seems to be not not restoring lost message data. Message which received not correctly packed. Existing system consists of present application in mobile device which is not safe if it is deleted by someone. This is the main drawback of the existing system. Existing system does not consists of storing the message from the current application by using mobile.

**2.3 PROPOSED SYSTEM**

In this project, we use the Database of an Android-based Smartphone to record and analyze various messages. It gives alert to user, then it gets stored in Application Database. The proposed application is to develop a system that offers news services through offline. Once the application is downloaded and installed by the user, he/she should register to it and he can access the service by providing the unique username and password.

**3. ANALYSIS**

**3.1 INTRODUCTION**

Requirements Analysis is done in order to understand the problem for which the software system is to solve. For example, the problem could be automating an existing manual process, or developing a completely new automated system, or a combination of the two. For large systems which have a large number of features, and that need to perform many different tasks, understanding the requirements of the system is a major task. The emphasis in requirements Analysis is on identifying what is needed from the system and not how the system will achieve its goals. This task is complicated by the fact that there are often at least two parties involved in software development - a client and a developer. The developer usually does not understand the client's problem domain, and the client often does not understand the issues involved in software systems. This causes a communication gap, which has to be adequately bridged during requirements analysis.

In most software projects, the requirement phase ends with a document describing all the requirements. In other words, the goal of the requirement specification phase is to produce the software requirement specification document. The person responsible for the requirement analysis is often called the analyst. There are two major activities in this phase - problem understanding or analysis and requirement specification in problem analysis; the analyst has to understand the problem and its context. Such analysis typically requires a thorough understanding of the existing system, the parts of which must be automated.

Once the problem is analyzed and the essentials understood, the requirements must be specified in the requirement specification document. For requirement specification in the form of document, some specification language has to be selected. The requirements documents must specify all functional and performance requirements, the formats of inputs, outputs and any required standards, and all design constraints that exits due to political, economic environmental, and security reasons. The phase ends with validation of requirements specified in the document. The basic purpose of validation is to make sure that the requirements specified in the document, actually reflect the actual requirements or needs, and that all requirements are specified. Validation is often done through requirement review, in which a group of people including representatives of the client critically review the requirements specification.

### Role of Software Requirement Specification (SRS)

IEEE (Institute of Electrical and Electronics Engineering) defines as,

* A condition of capability needed by a user to solve a problem or achieve an objective;
* A condition or capability that must be met or possessed by a system to satisfy a contract, standard, specification, or other formally imposed document.

Note that in software requirements we are dealing with the requirements of the proposed system, that is, the capabilities that system, which is yet to be developed, should have. It is because we are dealing with specifying a system that does not exist in any form that the problem of requirements becomes complicated. Regardless of how the requirements phase proceeds, the Software Requirement Specification (SRS) is a document that completely describes what the proposed software should do without describing how the system will do it? The basic goal of the requirement phase is to produce the Software Requirement Specification (SRS), which describes the complete external behavior of the proposed software.

**3.2 FEASIBLE STUDY**

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

* ECONOMICAL FEASIBILITY
* TECHNICAL FEASIBILITY
* SOCIAL FEASIBILITY

**ECONOMICAL FEASIBILITY**

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

### TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

**SOCIAL FEASIBILITY**

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

**3.3 REQUIREMENT SPECIFICATION**

**3.3.1 Functional Requirements**

Functional requirements specify which output file should be produced from the given file they describe the relationship between the input and output of the system, for each functional requirement a detailed description of all data inputs and their source and the range of valid inputs must be specified.

**3.3.2 Nonfunctional Requirements**

Describe user-visible aspects of the system that are not directly related with the functional behavior of the system. Non-Functional requirements include quantitative constraints, such as response time (i.e. how fast the system reacts to user commands.) or accuracy.

**3.3.3 Hardware and Software Requirements**

**Hardware Requirements**

* + Processor : Intel Core 2 Duo or above
  + Hard Disk : 40 GB
  + RAM : 1 GB or Above
  + Mobile : ANDROID

**Software Requirements**

* + Operating System : Windows
  + Tool : ANDROID SDK
  + Mobile OS : ANDROID
  + ANDROID Version : 2.3 or above

**4. DESIGN**

**4.1 DESCRIPTION**

The purpose of the design phase is to plan a solution of the problem specified by the requirement document. This phase is the first step in moving from problem domain to the solution domain. The design of a system is perhaps the most critical factor affecting the quality of the software, and has a major impact on the later phases, particularly testing and maintenance. The output of this phase is the design document. This document is similar to a blue print or plan for the solution, and is used later during implementation, testing and maintenance.

The design activity is often divided into two separate phase-system design and detailed design. System design, which is sometimes also called top-level design, aims to identify the modules that should be in the system, the specifications of these modules, and how they interact with each other to produce the desired results. At the end of system design all the major data structures, file formats, output formats, as well as the major modules in the system and their specifications are decided.

During detailed design the internal logic of each of the modules specified in system design is decided. During this phase further details of the data structures and algorithmic design of each of the modules is specified. The logic of a module is usually specified in a high-level design description language, which is independent of the target language in which the software will eventually be implemented. In system design the focus is on identifying the modules, whereas during detailed design the focus is on designing the logic for each of the modules.

During the design phase, often two separate documents are produced. One for the system design and one for the detailed design. Together, these documents completely specify the design of the system. That is they specify the different modules in the system and internal logic of each of the modules.

A design methodology is a systematic approach to creating a design by application of set of techniques and guidelines. Most methodologies focus on system design. The two basic principles used in any design methodology are problem partitioning and abstraction. A large system cannot be handled as a whole, and so for design it is partitioned into smaller systems. Abstraction is a concept related to problem partitioning. When partitioning is used during design, the design activity focuses on one part of the system at a time. Since the part being designed interacts with other parts of the system, a clear understanding of the interaction is essential for properly designing the part. For this, abstraction is used. An abstraction of a system or a part defines the overall behavior of the system at an abstract level without giving the internal details.

While working with the part of a system, a designer needs to understand only the abstractions of the other parts with which the part being designed interacts. The use of abstraction allows the designer to practice the "divide and conquer" technique effectively by focusing one part at a time, without worrying about the details of other parts.

Like every other phase, the design phase ends with verification of the design. If the design is not specified in some executable language, the verification has to be done by evaluating the design documents. One way of doing this is thorough reviews. Typically, at least two design reviews are held-one for the system design and one for the detailed and one for the detailed design.

**Software Development Life Cycle**

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.

The trends of increasing technical complexity of the systems, coupled with the need for repeatable and predictable process methodologies, have driven System Developers to establish system development models or software development life cycle models.

Nearly three decades ago the operations in an organization used to be limited and so it was possible to maintain those using manual procedures. But with the growing operations of organizations, the need to automate the various activities increased, since for manual procedures it was becoming very difficult, slow and complicated.

Like maintaining records for a thousand plus employees company on papers is definitely a cumbersome job. So, at that time more and more companies started going for automation. Since there were a lot of organizations, which were opting for automation, it was felt that some standard and structural procedure or methodology be introduced in the industry so that the transition from manual to automated system became easy. The concept of system life cycle came into existence then.

Life cycle model emphasized on the need to follow some structured approach towards building new or improved system. There were many models suggested. A waterfall model was among the very first models that came into existence. Later on many other models like prototype, rapid application development model, etc were also introduced.

System development begins with the recognition of user needs. Then there is a preliminary investigation stage. It includes evaluation of present system, information gathering, feasibility study, and request approval. Feasibility study includes technical, economic, legal and operational feasibility. In economic feasibility cost-benefit analysis is done. After that, there are detailed design, implementation, testing and maintenance stages.

In this session, we'll be learning about various stages that make system's life cycle. In addition, different life cycles models will be discussed. These include Waterfall model, Prototype model, Object-Oriented Model, spiral model and Dynamic Systems Development Method (DSDM).

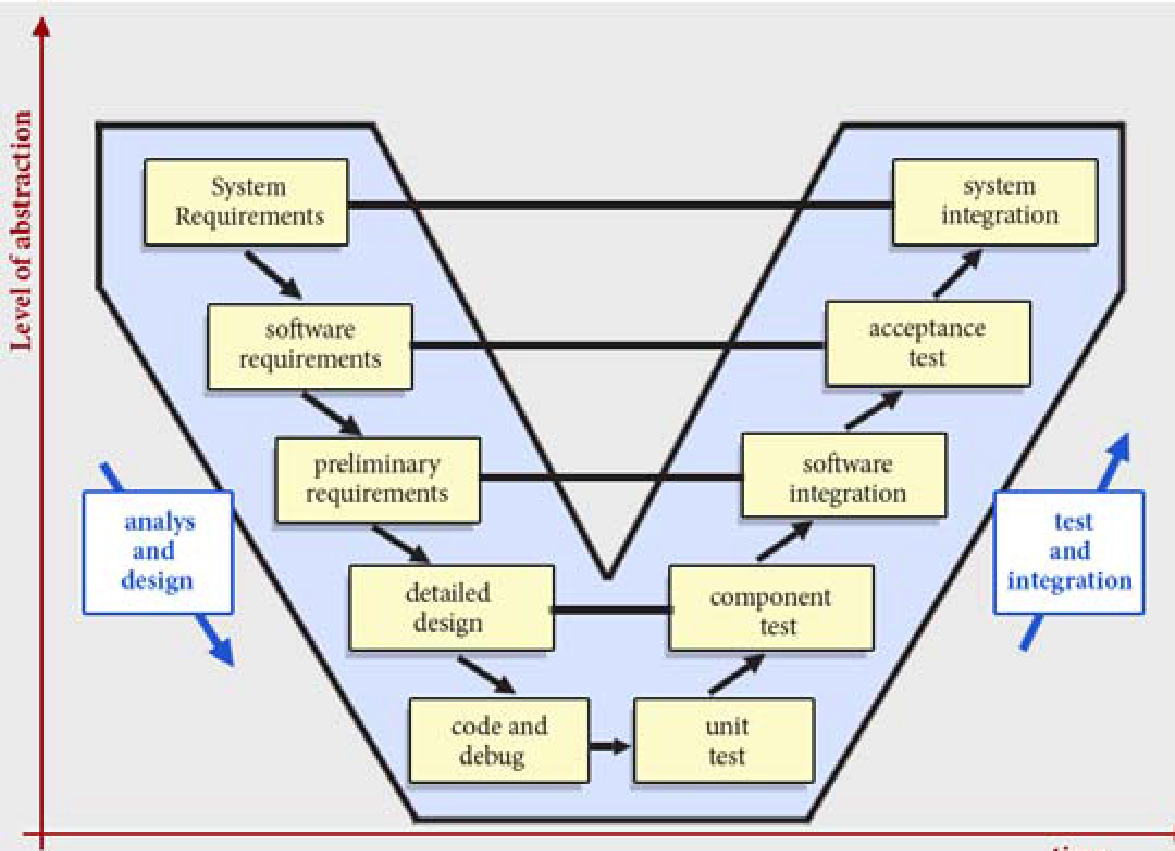
**4.2 V-SHAPED MODEL**

Just like the waterfall model, the V-Shaped life cycle is a sequential path of execution of processes. Each phase must be completed before the next phase begins. Testing is emphasized in this model more than the waterfall model. The testing procedures are developed early in the life cycle before any coding is done, during each of the phases preceding implementation. Requirements begin the life cycle model just like the waterfall model. Before development is started, a system test plan is created. The test plan focuses on meeting the functionality specified in requirements gathering. The high-level design phase focuses on system architecture and design. An integration test plan is created in this phase in order to test the pieces of the software systems ability to work together. However, the low-level design phase lies where the actual software components are designed, and unit tests are created in this phase as well. The implementation phase is, again, where all coding takes place. Once coding is complete, the path of execution continues up the right side of the V where the test plans developed earlier are now put to use.

**Advantages**

* Simple and easy to use.
* Each phase has specific deliverables.
* Higher chance of success over the waterfall model due to the early development of test plans during the life cycle.
* Works well for small projects where requirements are easily understood.

**The following diagram shows how a V-Shaped model acts like:**

****

**V-Shaped Model**

**4.3 MODULE DESCRIPTION**

**MODULES:**

1. Device Background Module

2. Registration Module

3. Message Retrieving

4. Viewing Message

**1. Device Background Module**

* Our work reveals to identify that it is always running in background with the help of receiver class
* We also utilized a single measuring device rather than expensive external sensors placed in numerous places around, which ultimately increases infrastructure costs.
* Our device, which is a mobile Smartphone, contains Received, offering flexibility in methodology and user implementation.

**2. Registration Module**

* User Register his user name and password
* Using Shared preferences logic to store it
* Once registered then it’s for life time

**3. Message Retrieving**

* Here we are retrieve message from sender that is displaying in alert message
* Here concept is done through Broadcast receiver

**4. Viewing Message**

* Here we see all the messages which are retrieved from sender
* Here we use database base concept for storing messages

**4.4 UML DIAGRAMS**

**Object Oriented Analysis**

An object-oriented system is composed of objects. The behavior of the system is achieved through collaboration between these objects, and the state of the system is the combined state of all the objects in it. Collaboration between objects involves those sending messages to each other. The exact semantics of message sending between objects varies depending on what kind of system is being modeled. In some systems, "sending a message" is the same as "invoking a method".

Object Oriented Analysis aims to model the problem domain, the problem we want to solve by developing an object-oriented (OO) System. The source of the analysis is a written requirement statement, and/or written use cases, UML diagrams can be used to illustrate the statements.

An analysis model will not take into account implementation constraints, such as concurrency, distribution, persistence, or inheritance, nor how the system will be built The model of a system can be divided into multiple domains each of which are separately analyzed, and represent separate business, technological, or conceptual areas of interest The result of object-oriented analysis is a description of what is to be built, using concepts and relationships between concepts, often expressed as a conceptual model. Any other documentation that is needed to describe what is to be built is also included in the result of the analysis. That can include a detailed user interface mock-up document. The implementation constraints are decided during the object-oriented design (OOD) process.

### Object Oriented Design

Object-Oriented Design (OOD) is an activity where the designers are looking for logical solutions to solve a problem, using Objects Object-oriented design takes the conceptual model that is the result of object-oriented analysis, and adds implementation constraints imposed by the environment, the programming language and the chosen tools, as well as architectural assumptions chosen as basis of Design.

The concepts in the conceptual model are mapped to concrete classes, to abstract interfaces in APIs and to roles that the objects take in various situations. The interfaces and their implementations for stable concepts can be made available as reusable services. Concepts identified as unstable in object-oriented analysis will form basis for policy classes that make decisions, implement environment-specific or situation specific logic or algorithms.

The result of the object-oriented design is a detail description how the system can be built; using objects .Object-oriented software engineering (OOSE) is an object modeling language and Methodology.

OOSE was developed by Ivar Jacobson in 1992 while at Objectory AB. It is the first object-oriented design methodology to employ use cases to drive software design. It also uses other design products similar to those used by OMT.

The tool Objectory was created by the team at Objectory AB to implement the OOSE methodology. After success in the marketplace, other tool vendors also supported OOSE after Rational bought Objectory AB, the OOSE notation, methodology, and tools became superseded

* As one of the primary sources of the Unified Modeling Language (UML), concepts and notation from OOSE have been incorporated into UML
* The methodology part of OOSE has since evolved into the Rational Unified Process (RUP)
* The OOSE tools have been replaced by tools supporting UML and RUP
* OOSE has been largely replaced by the UML notation and by the RUP methodology

UML is the international standard notation for object-oriented analysis and design. The Object Management Group defines it. The heart of object-oriented problem solving is the construction of the model. The model abstracts the essential details of underline problem from its usually complicated real world. Several modelling tools are wrapped under the heading of the UML, which stands for Unified Modelling Language.

* Use case diagram
* Class diagram
* Object diagram
* Sequence diagram
* Collaboration diagram
* State chart diagram
* Activity diagram
* Component diagram
* Deployment diagram

**Why is UML important?**

Let's look at this question from the point of view of the construction trade. Architects design buildings. Builders use the designs to create buildings. The more complicated the building, the more critical the communication between architect and builder. Blueprints are the standard graphical language that both architects and builders must learn as part of their trade.

Writing software is not unlike constructing a building. The more complicated the underlying system, the more critical the communication among everyone involved in creating and deploying the software. In the past decade, the UML has emerged as the software blueprint language for analysts, designers, and programmers alike. It is now part of the software trade. The UML gives everyone from business analyst to designer to programmer a common vocabulary to talk about software design. The UML is applicable to object-oriented problem solving. Anyone interested in learning UML must be familiar with the underlying tenet of object-oriented problem solving it all begins with the construction of a model. A model is an abstraction of the underlying problem. The domain is the actual world from which the problem comes.

Models consist of objects that interact by sending each other message. Think of an object as "alive." Objects have things they know (attributes) and things they can do (behaviors or operations). The values of an object's attributes determine its state.

Classes are the "blueprints" for objects. A class wraps attributes (data) and behaviors (methods or functions) into a single distinct entity. Objects are instances of classes.

## Actors

## Actors are the users of the system being modeled. Each Actor will have a well-defined role, and in the context of that role have useful interactions with the system. A person may perform the role of more than one Actor, although they will only assume one role during one use case interaction. An Actor role may be performed by a non-human system, such as another computer program.

**Use Cases**

Use case Diagrams represent the functionality of the system from a user’s point of view. Use cases are used during requirements elicitation and analysis to represent the functionality of the system. Use cases focus on the behavior of the system from external point of view. Actors are external entities that interact with the system. Examples of actors include users like administrator, bank customer etc., or another system like central database.

**Sequence diagram**

Is an interaction diagram that details how operations are carried out -- what messages are sent and when Sequence diagrams are organized according to time? The time progresses as you go down the page. The objects involved in the operation are listed from left to right according to when they take part in the message sequence.

**Class Diagram**

A Class diagram gives an overview of a system by showing its classes and the   
relationships among them. Class diagrams are static -- they display what interacts but not what happens when they do interact. Our class diagram has three kinds of relationships.

**Association**

A relationship between instances of the two classes. There is an association between two classes if an instance of one class must know about the other in order to perform its work. In a diagram, an association is a link connecting two classes.

**Aggregation**

An association in which one class belongs to a collection. An aggregation has a diamond end pointing to the part containing the whole.

**Generalization**

An inheritance link indicating one class is a super class of the other. A generalization has a triangle pointing to the super class.

**Activity Diagram**

An activity diagram is essentially a fancy flowchart. Activity diagrams and state chart diagrams are related. While a state chart diagram focuses attention on an object undergoing a process (or on a process as an object), an activity diagram focuses on the flow of activities involved in a single process.

**State chart Diagram**

AState chart diagram describes the flow of control from one state to another state. States are defined as a condition in which an object exists and it changes when some event is triggered. So the most important purpose of State chart diagram is to model life time of an object from creation to termination.

|  |  |  |
| --- | --- | --- |
| **Group** | **Term** | **Definition** |
| Business | Accounting Periods | A defined period of time whereby performance reports may be extracted. (normally 4 week periods). |
| Technical | Association | A relationship between two or more entities. Implies a connection of some type - for example one entity uses the services of another or one entity is connected to another over a network link. |
| Technical | Class | A logical entity encapsulating data and behavior. A class is a template for an object - the class is the design, the object the runtime instance. |
| Technical | Component Model | The component model provides a detailed view of the various hardware and software components that make up the proposed system. It shows both where these components reside and how they inter-relate with other components. Component requirements detail what responsibilities a component has to supply functionality or behavior within the system. |
| Business | Customer | A person or a company that requests An entity to transport goods on their behalf. |
| Technical | Deployment Architecture | A view of the proposed hardware that will make up the new system, together with the physical components that will execute on that hardware. Includes specifications for machine, operating system, network links, backup units &etc. |
| Technical | Deployment Model | A model of the system as it will be physically deployed |
| Technical | Extends Relationship | A relationship between two use cases in which one use case 'extends' the behavior of another. Typically this represents optional behavior in a use case scenario - for example a user may optionally request a list or report at some point in a performing a business use case. |
| Technical | Includes Relationship | A relationship between two use cases in which one use case 'includes' the behavior. This is indicated where there a specific business use cases which are used from many other places - for example updating a train record may be part of many larger business processes. |
| Technical | Use Case | A Use Case represents a discrete unit of interaction between a user (human or machine) and the system. A Use Case is a single unit of meaningful work; for example creating a train, modifying a train and creating orders are all Use Cases.  Each Use Case has a description which describes the functionality that will be built in the proposed system. A Use Case may 'include' another Use Case's functionality or 'extend' another Use Case with its own behavior. |

**Use Case Diagram**

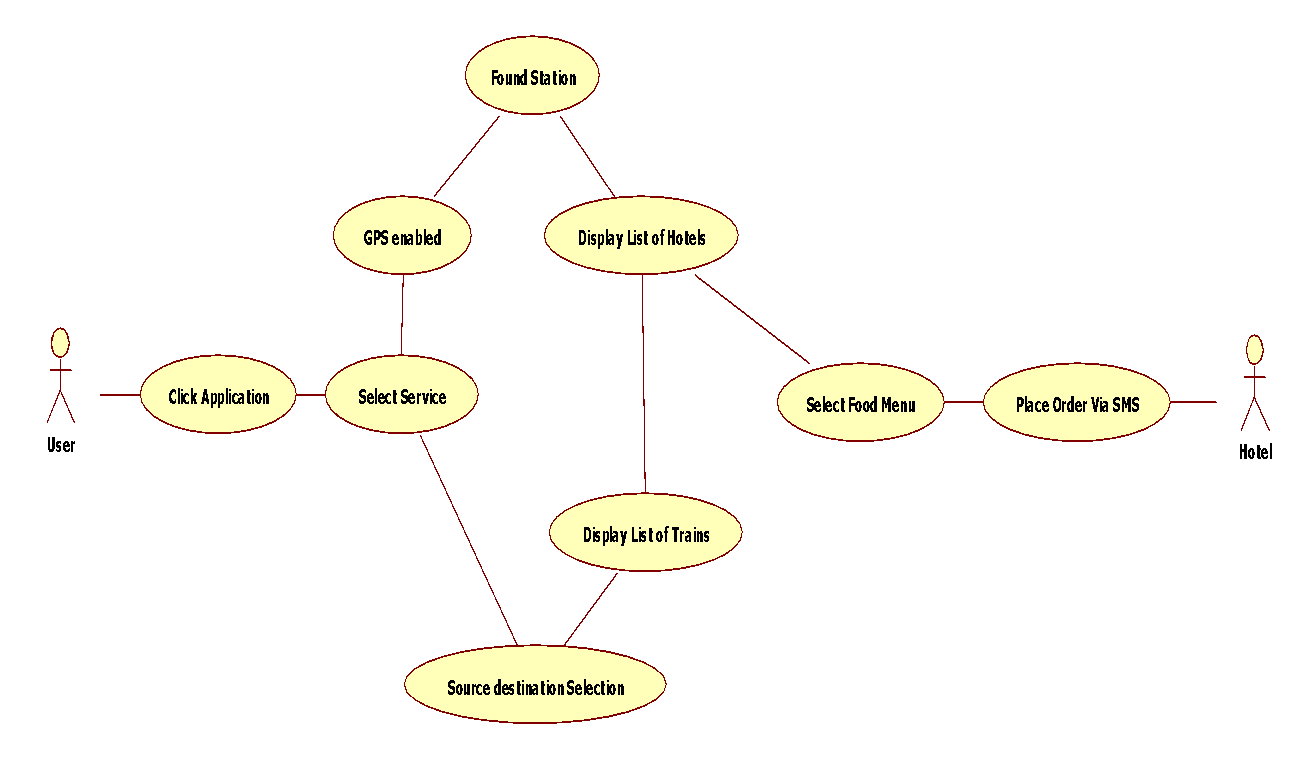


Figure 4.4.1:Use Case Diagram

**Class Diagram**

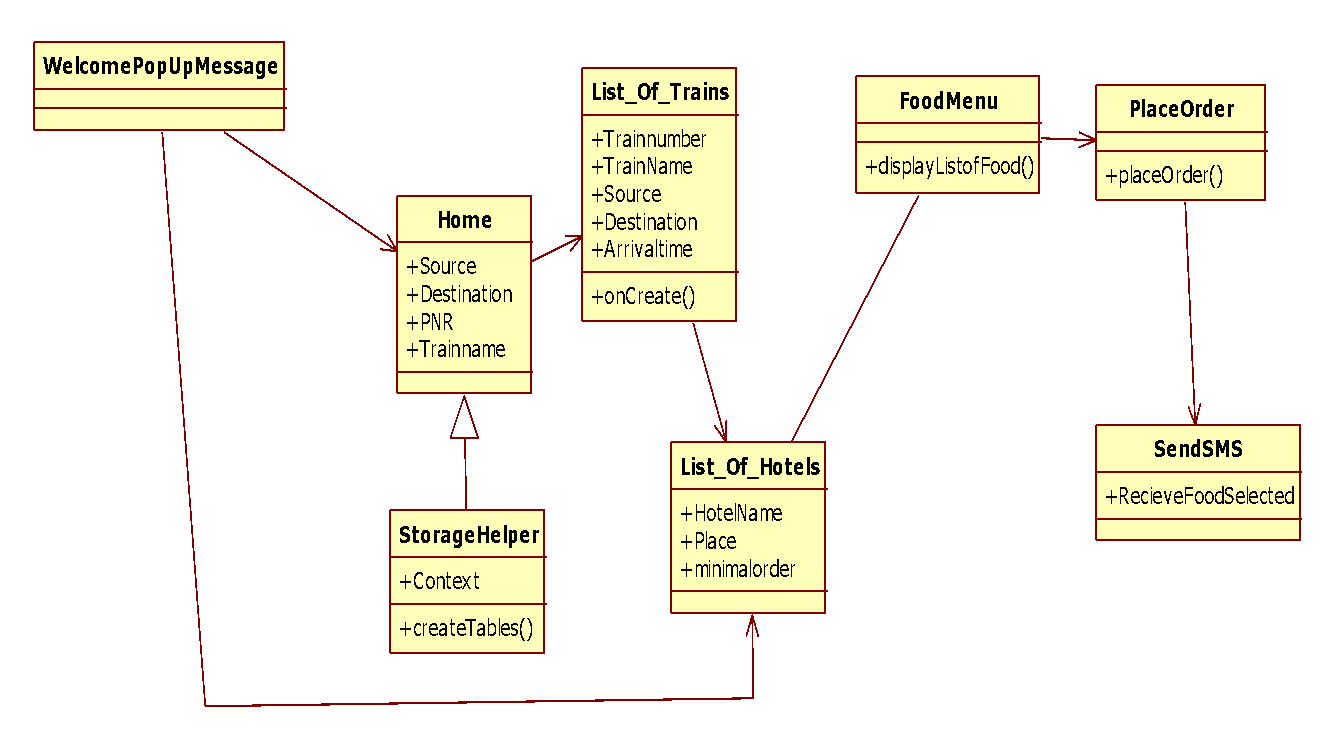


Figure 4.4.2:Class Diagram

**Sequence Diagram**

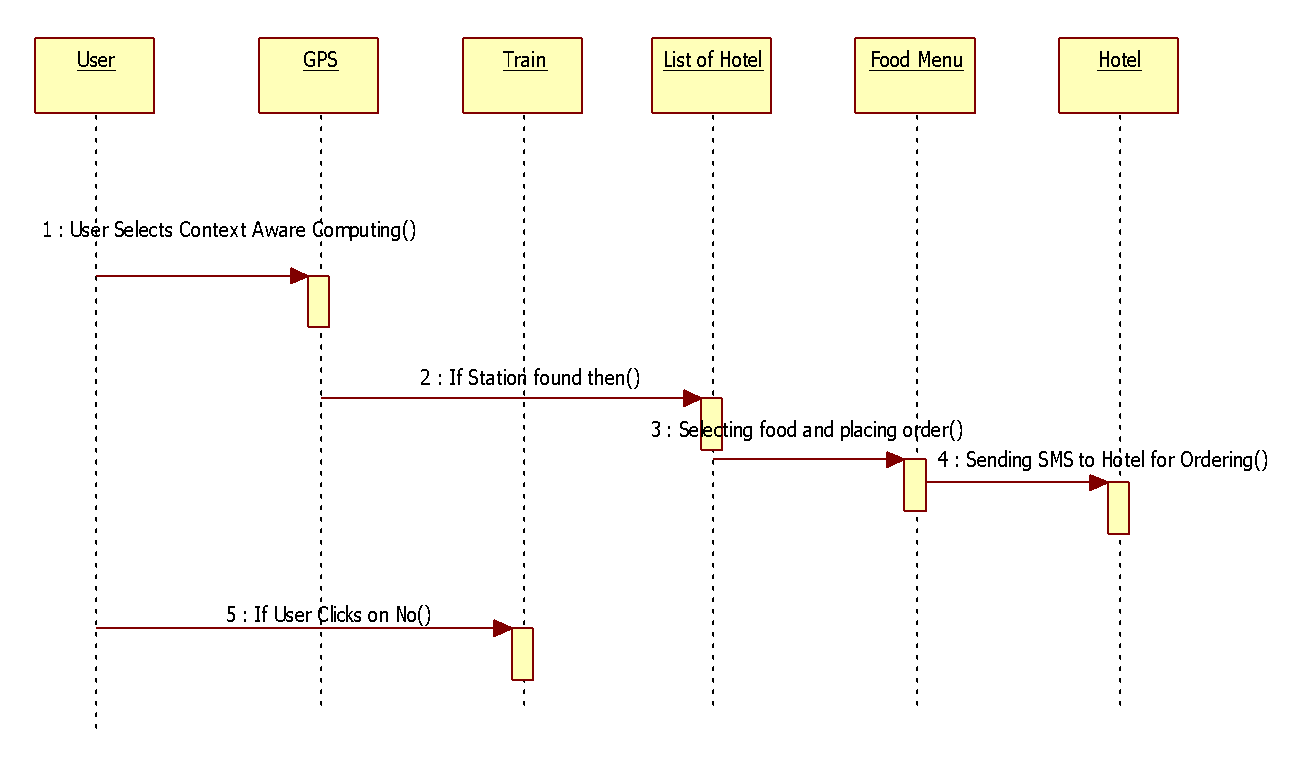
****

Figure 4.4.3:Sequence Diagram

**Activity Diagram**

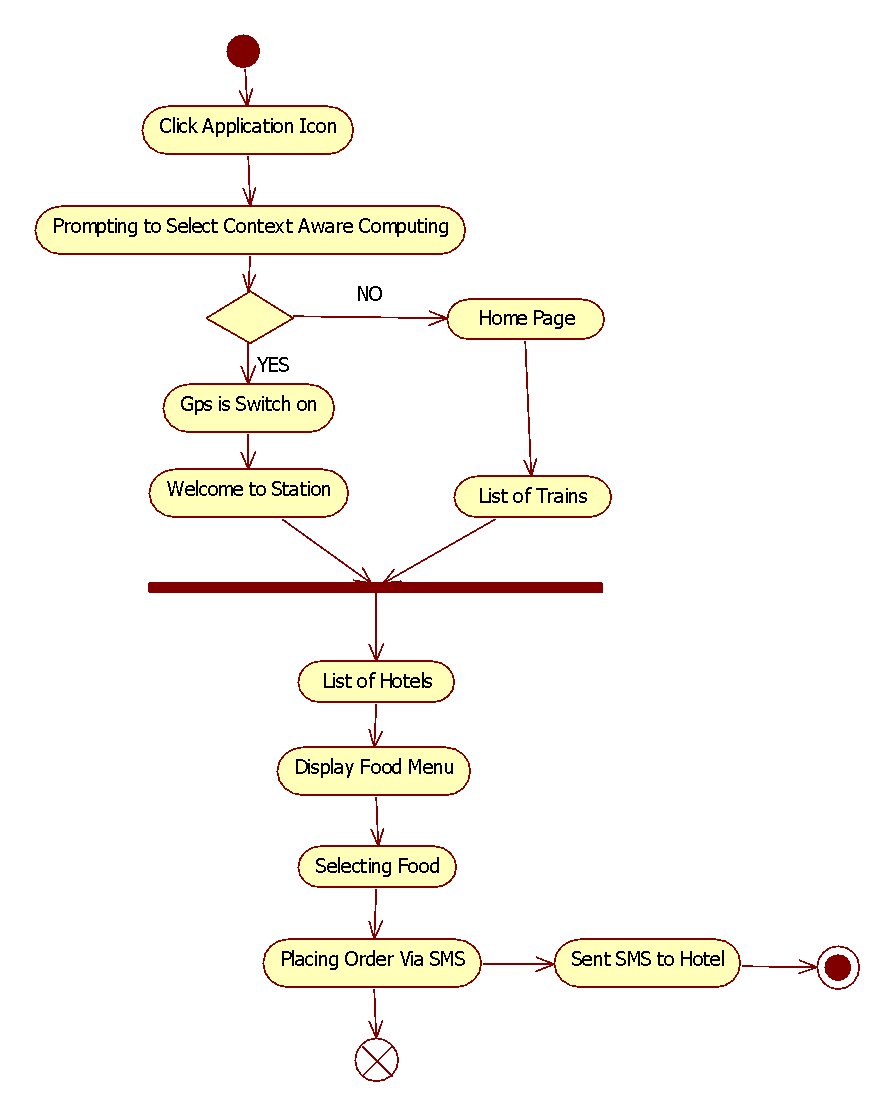
****

Figure 4.4.4:Activity Diagram

**Collaboration Diagram**

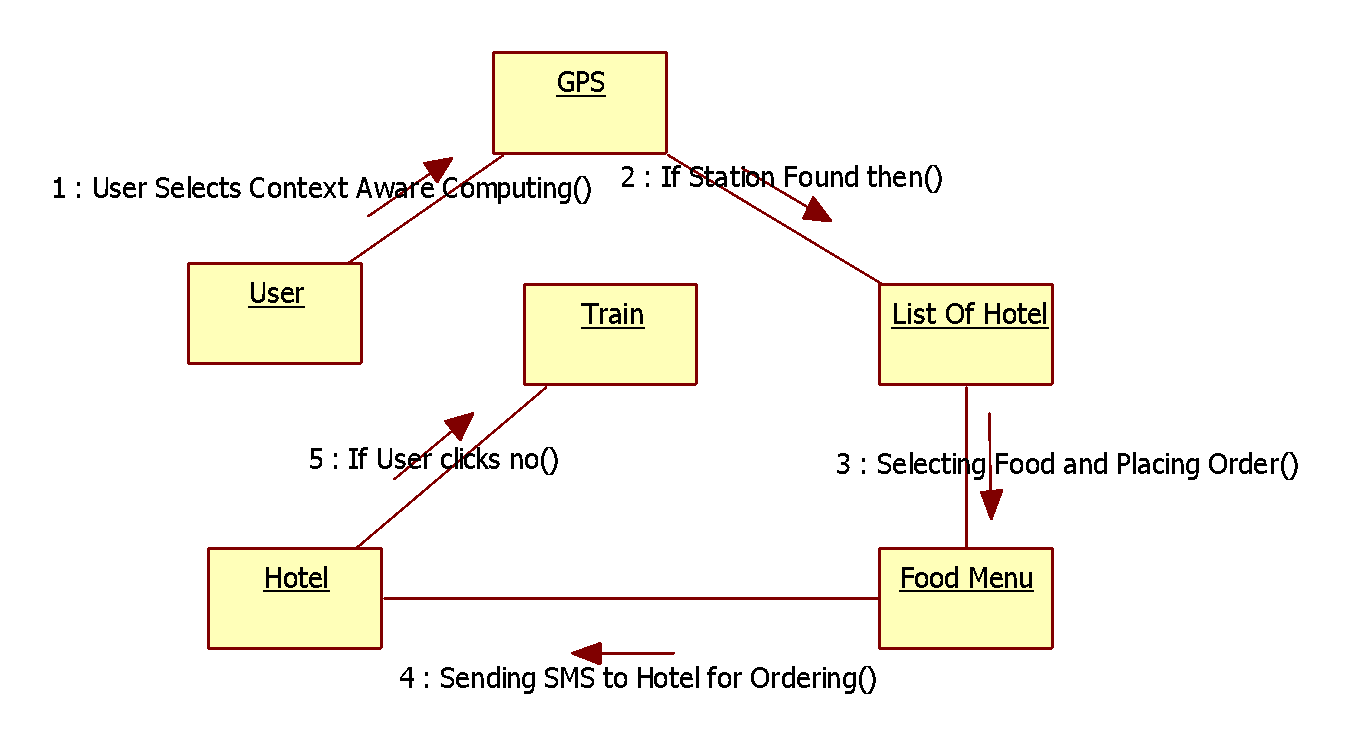
****

Figure 4.4.5:Collaboration Diagram

**4.5 PROCESS SPECIFICATION**

**INPUT DESIGN**

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

* What data should be given as input?
* How the data should be arranged or coded?
* The dialog to guide the operating personnel in providing input.
* Methods for preparing input validations and steps to follow when error occur.

**OBJECTIVES**

1. Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.

2. It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.

3. When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in maize of instant. Thus the objective of input design is to create an input layout that is easy to follow

**OUTPUT DESIGN**

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system’s relationship to help user decision-making.

Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.

The output form of an information system should accomplish one or more of the following objectives.

* Convey information about past activities, current status or projections of the
* Future.
* Signal important events, opportunities, problems, or warnings.
* Trigger an action.
* Confirm an action

**4.6 SYSTEM ARCHITECTURE**

Eclipse employs plug-ins in order to provide all of its functionality on top of (and including) the runtime system, in contrast to some other applications where functionality is typically [hard coded](http://en.wikipedia.org/wiki/Hard_code). The runtime system of Eclipse is based on [Equinox](http://en.wikipedia.org/wiki/Equinox_%28OSGi%29), an [OSGi](http://en.wikipedia.org/wiki/OSGi) standard compliant implementation.

This plug-in mechanism is a lightweight [software componentry](http://en.wikipedia.org/wiki/Software_componentry) framework. In addition to allowing Eclipse to be extended using other [programming languages](http://en.wikipedia.org/wiki/Programming_language) such as [C](http://en.wikipedia.org/wiki/C_%28programming_language%29) and [Python](http://en.wikipedia.org/wiki/Python_%28programming_language%29), the plug-in framework allows Eclipse to work with typesetting languages like [LaTeX](http://en.wikipedia.org/wiki/LaTeX), networking applications such as [telnet](http://en.wikipedia.org/wiki/Telnet), and [database management systems](http://en.wikipedia.org/wiki/Database_management_system). The plug-in architecture supports writing any desired extension to the environment, such as for [configuration management](http://en.wikipedia.org/wiki/Configuration_management). Java and [CVS](http://en.wikipedia.org/wiki/Concurrent_Versions_System) support is provided in the Eclipse [SDK](http://en.wikipedia.org/wiki/Software_development_kit), with [Subversion](http://en.wikipedia.org/wiki/Subversion_%28software%29) support provided by third-party plug-ins.

With the exception of a small run-time kernel, everything in Eclipse is a plug-in. This means that every plug-in developed integrates with Eclipse in exactly the same way as other plug-ins; in this respect, all features are "created equal". Eclipse provides plug-ins for a wide variety of features, some of which are through third parties using both free and commercial models. Examples of plug-ins include a [UML](http://en.wikipedia.org/wiki/Unified_Modeling_Language) plug-in for Sequence and other UML diagrams, a plug-in for DB Explorer, and many others.

The Eclipse SDK includes the Eclipse Java Development Tools (JDT), offering an IDE with a built-in [incremental](http://en.wikipedia.org/wiki/Incremental_compiler) Java compiler and a full model of the Java source files. This allows for advanced [refactoring](http://en.wikipedia.org/wiki/Refactor) techniques and code analysis. The IDE also makes use of a workspace, in this case a set of [metadata](http://en.wikipedia.org/wiki/Metadata) over a flat file space allowing external file modifications as long as the corresponding workspace "resource" is refreshed afterwards.

Eclipse implements [widgets](http://en.wikipedia.org/wiki/GUI_widget) through a widget toolkit for Java called [SWT](http://en.wikipedia.org/wiki/Standard_Widget_Toolkit), unlike most Java applications, which use the Java standard [Abstract Window Toolkit](http://en.wikipedia.org/wiki/Abstract_Window_Toolkit) (AWT) or [Swing](http://en.wikipedia.org/wiki/Swing_%28Java%29). Eclipse's user interface also uses an intermediate [GUI](http://en.wikipedia.org/wiki/GUI) layer called [JFace](http://en.wikipedia.org/wiki/JFace), which simplifies the construction of applications based on SWT.

## Rich Client Platform

* [Equinox OSGi](http://en.wikipedia.org/wiki/Equinox_%28OSGi%29) – a standard bundling framework
* Core platform – boot Eclipse, run [plug-ins](http://en.wikipedia.org/wiki/Plug-in_%28computing%29)
* [Standard Widget Toolkit](http://en.wikipedia.org/wiki/Standard_Widget_Toolkit) (SWT) – a portable [widget toolkit](http://en.wikipedia.org/wiki/Widget_toolkit)
* [JFace](http://en.wikipedia.org/wiki/JFace) – viewer classes to bring [model view controller](http://en.wikipedia.org/wiki/Model_view_controller) programming to SWT, file buffers, text handling, text editors
* Eclipse Workbench – views, editors, perspectives, wizards

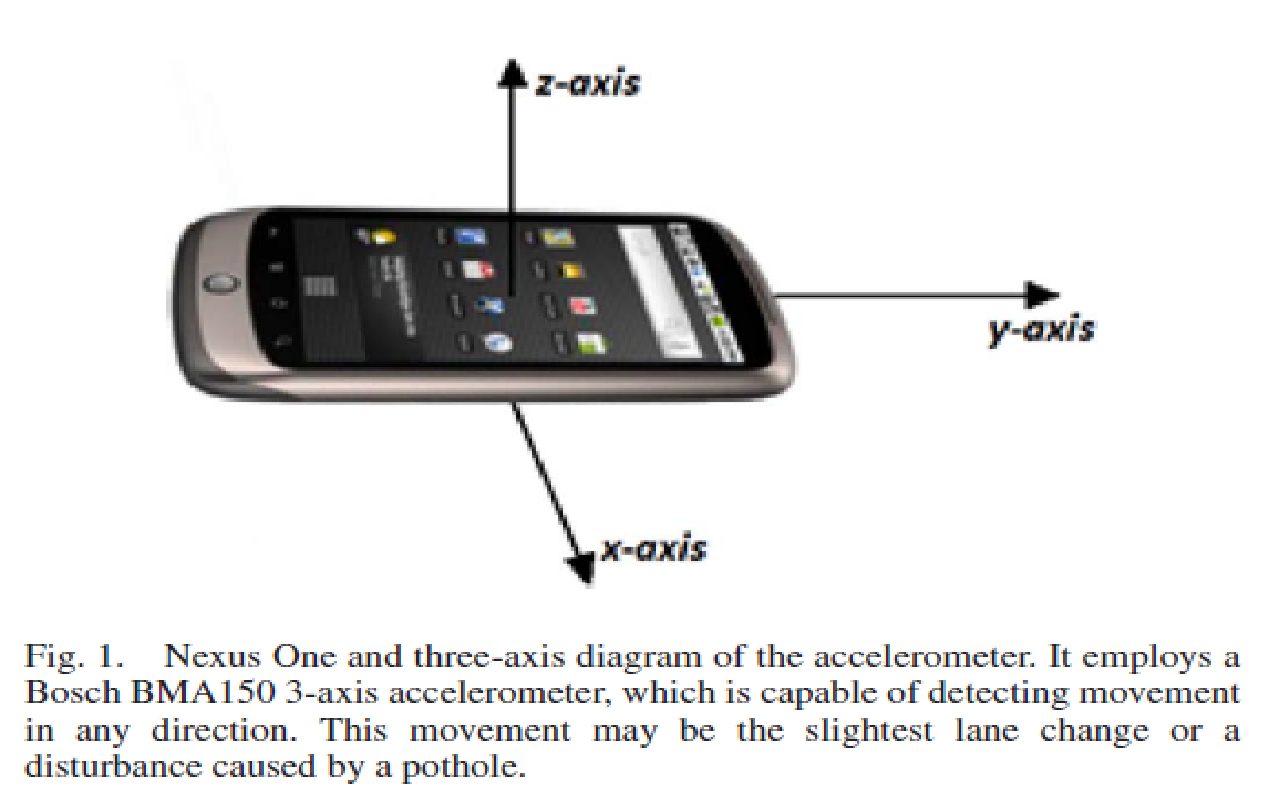
****

Figure 4.6.1:System Architecture

**5. IMPLEMENTATION AND RESULTS**

**5.1 INTRODUCTION**

Implementation literally means to put into effect or to carry out. In the system implementation phase, the software deals with translation of the design specifications into source code. The ultimate goal of the implementation is to write the source code and the internal documentation so that it can be verified easily. The code and documentation should be written in a manner that eases debugging, testing and modification. System flow charts, simple run on packages, sample output etc., is part of the implementation.

An effort was made to satisfy the following goals in order:

* Clarity and simplicity of the code
* Minimization of Hard Coding
* Minimization of the amount of memory used
* Thorough phased implementation has been done so that we can use our proposed system correctly.

**5.2 TECHNOLOGIES USED**

Android is a [software stack](http://en.wikipedia.org/wiki/Solution_stack) for [mobile devices](http://en.wikipedia.org/wiki/Mobile_devices) that includes an [operating system](http://en.wikipedia.org/wiki/Operating_system), [middleware](http://en.wikipedia.org/wiki/Middleware) and key [applications](http://en.wikipedia.org/wiki/Application_software). [Google Inc.](http://en.wikipedia.org/wiki/Google) purchased the initial developer of the software; Android Inc. in 2005.Android's [mobile operating system](http://en.wikipedia.org/wiki/Mobile_operating_system) is based on 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 on Android's development and release. The Android Open Source Project (AOSP) is tasked with the maintenance and further development of Android. The Android operating system is the world's best-selling [Smartphone](http://en.wikipedia.org/wiki/Smartphone) platform. The [Android SDK](http://developer.android.com/sdk/index.html) provides the tools and APIs necessary to begin developing applications Android platform using the Java programming language. Android has a large community of developers writing [applications](http://en.wikipedia.org/wiki/Application_software) ("apps") that extend the functionality of the devices. There are currently over 250,000 apps available for Android.

**Features**

* Application framework enabling reuse and replacement of components
* Dalvik virtual machine optimized for mobile devices
* Integrated browser based on the open source [WebKit](http://webkit.org/) engine
* Optimized graphics powered by a custom 2D graphics library; 3D graphics based on the OpenGL ES 1.0 specification (hardware acceleration optional)
* SQLitefor structured data storage
* Media support for common audio, video, and still image formats (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF)
* GSM Telephony (hardware dependent)
* Bluetooth, EDGE, 3G, and Wi-Fi (hardware dependent)
* Camera, GPS, compass, and accelerometer (hardware dependent)
* Rich development environment including a device emulator, tools for debugging, memory and performance profiling, and a plug-in for the Eclipse IDE.

## Android Architecture

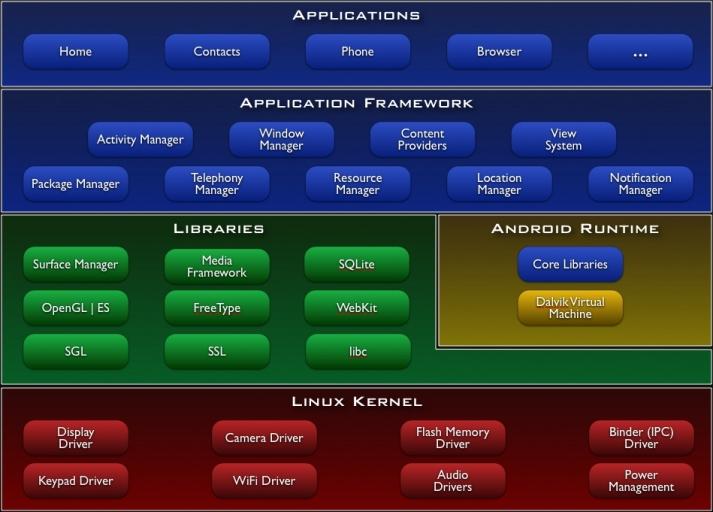


Figure 5.2.1:Android Architecture

## Libraries

Android includes a set of C/C++ libraries used by various components of the Android system. These capabilities are exposed to developers through the Android application framework. Some of the core libraries are listed below:

* System C library - a BSD-derived implementation of the standard C system library (libc), tuned for embedded Linux-based devices
* Media Libraries - based on Packet Video's Open CORE; the libraries support playback and recording of many popular audio and video formats, as well as static image files, including MPEG4, H.264, MP3, AAC, AMR, JPG, and PNG
* Surface Manager- manages access to the display subsystem and seamlessly composites 2D and 3D graphic layers from multiple applications
* LibWebCore - a modern web browser engine which powers both the Android browser and an embeddable web view
* SGL - the underlying 2D graphics engine
* 3D libraries- an implementation based on OpenGL ES 1.0 APIs; the libraries use either hardware 3D acceleration (where available) or the included, highly optimized 3D software pasteurizer
* Free Type- bitmap and vector font rendering
* SQLite - a powerful and lightweight relational database engine available to all applications

## Android Runtime

Android includes a set of core libraries that provides most of the functionality available in the core libraries of the Java programming language. Every Android application runs in its own process, with its own instance of the Dalvik virtual machine. Dalvik has been written so that a device can run multiple VMs efficiently. The Dalvik VM executes files in the Dalvik Executable (.dex) format which is optimized for minimal memory footprint. The VM is register-based, and runs classes compiled by a Java language compiler that have been transformed into the .dex format by the included "dx" tool. The Dalvik VM relies on the Linux kernel for underlying functionality such as threading and low-level memory management.

### Android Operation System

Android is an operating system based on [Linux](http://www.vogella.de/articles/Ubuntu/article.html) with a [Java](http://www.vogella.de/articles/JavaIntroduction/article.html) programming interface. It provides tools, e.g. a compiler, debugger and a device emulator as well as its own Java Virtual machine (Dalvik Virtual Machine - DVM). [Android](http://www.vogella.de/articles/Android/article.html) is created by the Open Handset Alliance which is lead by Google. Android uses a special virtual machine, e.g. the Dalvik Virtual Machine. Dalvik uses special byte code. Therefore you cannot run standard Java byte code on Android. Android provides a tool "dx" which allows converting Java Class files into "dex" (Dalvik Executable) files.

Android applications are packed into an .apk (Android Package) file by the program "aapt" (Android Asset Packaging Tool) to simplify development Google provides the Android Development Tools (ADT) for [Eclipse](http://www.vogella.de/articles/Eclipse/article.html). The ADT performs automatically the conversion from class to dex files and creates the apk during deployment. Android supports 2-D and 3-D graphics using the OpenGL libraries and supports data storage in a [SQLite](http://www.vogella.de/articles/AndroidSQLite/article.html) database. Every [Android](http://www.vogella.de/articles/Android/article.html) applications runs in its own process and under its own userid which is generated automatically by the Android system during deployment. Therefore the application is isolated from other running applications and a misbehaving application cannot easily harm other Android applications.

### Important Android components

An Android application consists out of the following parts:

* Activity - Represents the presentation layer of an Android application, e.g. a screen which the user sees. An Android application can have several activities and it can be switched between them during runtime of the application.
* Views - The User interface of Activities is build with widgets classes which inherent from "android.view.View". The layout of the views is managed by "android.view.ViewGroups".
* [Services](http://www.vogella.de/articles/AndroidServices/article.html) - perform background tasks without providing an UI. They can notify the user via the notification framework in Android.
* [Content Provider](http://www.vogella.de/articles/Android/article.html#contentprovider) - provides data to applications, via a content provider your application can share data with other applications. Android contains a SQLite DB which can serve as data provider
* [Intents](http://www.vogella.de/articles/AndroidIntent/article.html) are asynchronous messages which allow the application to request functionality from other services or activities. An application can call directly a service or activity (explicit intent) or asked the Android system for registered services and applications for intent (implicit intents). For example the application could ask via intent for a contact application. Application registers them to an intent via an IntentFilter. Intents are a powerful concept as they allow creating loosely coupled applications.
* Broadcast Receiver - receives system messages and implicit intents, can be used to react to changed conditions in the system. An application can register as a broadcast receiver for certain events and can be started if such an event occurs.
* A Java Virtual Machine (JVM) enables a set of computer software programs and data structures to use a [virtual machine](http://en.wikipedia.org/wiki/Virtual_machine) model for the execution of other computer programs and [scripts](http://en.wikipedia.org/wiki/Scripting_language). The model used by a JVM accepts a form of computer [intermediate language](http://en.wikipedia.org/wiki/Intermediate_language) commonly referred to as [Java byte code](http://en.wikipedia.org/wiki/Java_bytecode). This language conceptually represents the instruction set of a [stack-oriented](http://en.wikipedia.org/wiki/Stack-oriented_programming_language), [capability architecture](http://en.wikipedia.org/wiki/Capability_architecture). [Sun Microsystems](http://en.wikipedia.org/wiki/Sun_Microsystems) states there are over 4.5 billion JVM-enabled devices
* A JVM can also execute byte code compiled from programming languages other than Java. For example, [Ada](http://en.wikipedia.org/wiki/Ada_%28programming_language%29) source code can be compiled to execute on a JVM. JVMs can also be released by other companies besides Oracle (the developer of Java) — JVMs using the "Java" trademark may be developed by other companies as long as they adhere to the JVM specification published by Oracle and to related contractual obligations.
* Java was conceived with the concept of WORA: "[write once, run anywhere](http://en.wikipedia.org/wiki/Write_once,_run_anywhere)". This is done using the Java Virtual Machine. The JVM is the environment in which java programs execute. It is software that is implemented on non-virtual hardware and on standard [operating systems](http://en.wikipedia.org/wiki/Operating_system).
* JVM is a crucial component of the [Java platform](http://en.wikipedia.org/wiki/Java_%28software_platform%29), and because JVMs are available for many hardware and software [platforms](http://en.wikipedia.org/wiki/Platform_%28computing%29), Java can be both [middleware](http://en.wikipedia.org/wiki/Middleware) and a platform in its own right, hence the trademark [write once, run anywhere](http://en.wikipedia.org/wiki/Write_once,_run_anywhere). The use of the same byte code for all platforms allows Java to be described as "compile once, run anywhere", as opposed to "write once, compile anywhere", which describes cross-platform [compiled languages](http://en.wikipedia.org/wiki/Compiled_language). A JVM also enables such features as [automated exception handling](http://en.wikipedia.org/wiki/Automated_exception_handling), which provides "root-cause" debugging information for every software error ([exception](http://en.wikipedia.org/wiki/Exception_handling)), independent of the source code.
* A JVM is distributed along with a [set of standard class libraries](http://en.wikipedia.org/wiki/Java_Class_Library) that implement the Java [application programming interface](http://en.wikipedia.org/wiki/Application_programming_interface) (API). Appropriate APIs bundled together form the Java Runtime Environment (JRE).
* Java's execution environment is termed the Java Runtime Environment, or JRE.
* Programs intended to run on a JVM must be compiled into a standardized portable binary format, which typically comes in the form of [.class](http://en.wikipedia.org/wiki/Class_%28file_format%29) files. A program may consist of many classes in different files. For easier distribution of large programs, multiple class files may be packaged together in a [.jar](http://en.wikipedia.org/wiki/Jar_%28file_format%29) file (short for Java archive).
* The Java application launcher, java, offers a standard way of executing Java code. Compare javaw.
* The JVM [runtime](http://en.wikipedia.org/wiki/Run-time_system) executes .class or .jar files, [emulating](http://en.wikipedia.org/wiki/Emulator) the JVM [instruction set](http://en.wikipedia.org/wiki/Instruction_set) by [interpreting](http://en.wikipedia.org/wiki/Interpreter_%28computing%29) it, or using a [just-in-time compiler](http://en.wikipedia.org/wiki/Just-in-time_compilation) (JIT) such as Oracle's [HotSpot](http://en.wikipedia.org/wiki/HotSpot_%28Java%29). JIT compiling, not interpreting, is used in most JVMs today to achieve greater speed. There are also [ahead-of-time compilers](http://en.wikipedia.org/wiki/AOT_compiler) that enable developers to pre compile class files into native code for particular platforms.
* Like most virtual machines, the Java Virtual Machine has a [stack](http://en.wikipedia.org/wiki/Stack_machine)-based architecture akin to a microcontroller/microprocessor. However, the JVM also has low-level support for Java-like classes and methods, which amounts to a highly idiosyncratic [memory model](http://en.wikipedia.org/wiki/Java_Memory_Model) and capability-based architecture.

## Linux Kernel

Android relies on Linux version 2.6 for core system services such as security, memory management, process management, network stack, and driver model. The kernel also acts as an abstraction layer between the hardware and the rest of the software stack. The Linux kernel is an operating system [kernel](http://en.wikipedia.org/wiki/Kernel_%28computing%29) used by the [Linux](http://en.wikipedia.org/wiki/Linux) family of [Unix-like](http://en.wikipedia.org/wiki/Unix-like) [operating systems](http://en.wikipedia.org/wiki/Operating_system). It is one of the most prominent examples of [free and open source software](http://en.wikipedia.org/wiki/Free_and_open_source_software).

The Linux kernel is released under the [GNU General Public License](http://en.wikipedia.org/wiki/GNU_General_Public_License) version 2 (GPLv2),(plus some [firmware images](http://en.wikipedia.org/wiki/Firmware) with various licenses), and is developed by contributors worldwide. Day-to-day development takes place on the [Linux kernel mailing list](http://en.wikipedia.org/wiki/Linux_kernel_mailing_list).

The Linux kernel was initially conceived and created by [Finnish](http://en.wikipedia.org/wiki/Finns) [computer science](http://en.wikipedia.org/wiki/Computer_science) student[Linus Torvalds](http://en.wikipedia.org/wiki/Linus_Torvalds) in 1991. Linux rapidly accumulated developers and users who adapted code from other [free software](http://en.wikipedia.org/wiki/Free_software) projects for use with the new operating system. The Linux kernel has received contributions from thousands of programmers. Many [Linux distributions](http://en.wikipedia.org/wiki/Linux_distribution) have been released based upon the Linux kernel.

The Linux kernel has extensive support for and runs on many [virtual machine](http://en.wikipedia.org/wiki/Virtual_machine) architectures both as the host operating system and as a guest operating system. The virtual machines usually emulate [Intel x86](http://en.wikipedia.org/wiki/Intel_x86) family of processors, though in a few cases [PowerPC](http://en.wikipedia.org/wiki/PowerPC) or [ARM](http://en.wikipedia.org/wiki/ARM_architecture) processors are also emulated. At Google, the team led by Rubin developed a mobile device platform powered by the [Linux kernel](http://en.wikipedia.org/wiki/Linux_kernel). Google marketed the platform to handset makers and [carriers](http://en.wikipedia.org/wiki/Mobile_network_operator) on the premise of providing a flexible, upgradable system. Google had lined up a series of hardware component and software partners and signaled to carriers that it was open to various degrees of cooperation on their part.

Speculation about Google's intention to enter the mobile communications market continued to build through December 2006.Reports from the [BBC](http://en.wikipedia.org/wiki/BBC) and [The Wall Street Journal](http://en.wikipedia.org/wiki/The_Wall_Street_Journal) noted that Google wanted its search and applications on mobile phones and it was working hard to deliver that. Print and online media outlets soon reported rumors that Google was developing a Google-branded [handset](http://en.wikipedia.org/wiki/Handset#Telephony). Some speculated that as Google was defining technical specifications, it was showing prototypes to cell phone manufacturers and network operators.

**Hardware Running Android**

The main supported platform for Android is the [ARM architecture](http://en.wikipedia.org/wiki/ARM_architecture).

The Android OS can be used as an operating system for cell phones, net books and [tablets](http://en.wikipedia.org/wiki/Tablet_personal_computer), including the [Dell Streak](http://en.wikipedia.org/wiki/Dell_Streak), [Samsung Galaxy Tab](http://en.wikipedia.org/wiki/Samsung_Galaxy_Tab), TV and other devices. 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. In early 2010 Google collaborated with [HTC](http://en.wikipedia.org/wiki/HTC) to launch its flagship Android device, the [Nexus One](http://en.wikipedia.org/wiki/Nexus_One). This was followed later in 2010 with the [Samsung](http://en.wikipedia.org/wiki/Samsung)-made [Nexus S](http://en.wikipedia.org/wiki/Nexus_S).

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.

The Android [software development kit](http://en.wikipedia.org/wiki/Software_development_kit) (SDK) includes a comprehensive set of development tools. 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. The SDK is downloadable on the [android developer website](http://developer.android.com/sdk/index.html). 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 3.5 or 3.6) 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).

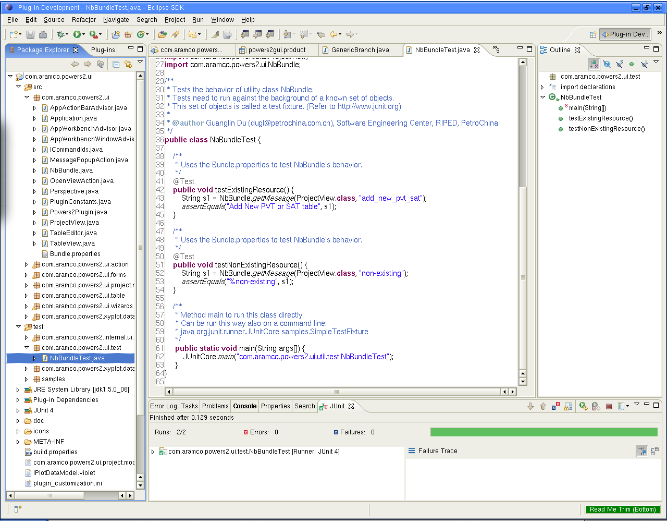
Android applications are packaged in [.apk](http://en.wikipedia.org/wiki/APK_%28file_format%29) format and stored under /data/app folder on the Android OS (the folder is accessible to root user only for security reasons). APK package contains .dex files(compiled byte code files called [Dalvik](http://en.wikipedia.org/wiki/Dalvik_Virtual_Machine) executables), resource files, etc.

# Eclipse (SDK)

Eclipse Software Development Kit (SDK) is a Java based open-source integrated development environment (IDE) which combines a number of different Eclipse projects including Platform, Java Development Tools (JDT) and the Plug-in Development Environment (PDE).

Eclipse can be used to create a large array of software applications using languages ranging from PHP, C++ programs, to Java. It is one of the most popular development tools in both the open-source and commercial worlds. It provides Java editing with validation, incremental compilation, cross-referencing, code assist; an XML Editor; Mylyn; and much more.

Eclipse is released under the Eclipse Foundation, a commercially friendly license that allows organizations to include Eclipse software in their commercial products, while at the same time asking those who create derivative works of EPL code to contribute back to the community.



Eclipse IDE

**Eclipse Platform**

The Eclipse Platform provides the core frameworks and services upon which all plug-in extensions are created. It also provides the runtime in which plug-ins are loaded, integrated, and executed. The primary purpose of the Platform is to enable other tool developers to easily build and deliver integrated tools.

**Features include**

* Supports the construction of a variety of tools for application development
* Supports an unrestricted set of tool providers, including independent software vendors (ISVs)
* Supports tools to manipulate arbitrary content types (e.g., HTML, Java, C, JSP, EJB, XML, and GIF)
* Facilitates seamless integration of tools within and across different content types and tool providers
* Supports both GUI and non-GUI-based application development environments

**Java Development Tools (JDT)**

The JDT project provides the tool plug-ins that implements a Java IDE supporting the development of any Java application, including Eclipse plug-ins. It adds a Java project nature and Java perspective to the Eclipse Workbench as well as a number of views, editors, wizards, builders, and code merging and refactoring tools. The JDT project allows Eclipse to be a development environment for itself.

**Features include**

* Java projects with source files arranged in package directories
* Editing with keyword and syntax coloring, outline showing declaration structure
* Code formatter
* Refactoring
* Search
* Compare
* Compile - JCK-compliant Java compiler
* Run Java programs in a separate target Java virtual machine
* Debug programs with JPDA-compliant Java virtual machine

### Android Source Code

The following step is optional.

During Android development it is very useful to have the Android source code available as Android uses a lot of defaults. [Haris Peco](http://twitter.com/snpe60) maintains plug-in with provides access to the Android Source code. Use the Eclipse update manager to install two of his plug-in.

**5.3 INSTALLATION OF ANDROID**

**Download the Android SDK**

Welcome Developers! If you are new to the Android SDK, please read the steps below, for an overview of how to set up the SDK. If you're already using the Android SDK, you should update to the latest tools or platform using the Android SDK and AVD Manager, rather than downloading a new SDK starter package. See [Adding SDK Components](http://developer.android.com/sdk/adding-components.html).

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |
|  |  |  |  |
|  |  |  |  |

Here an overview of the steps you must follow to set up the Android SDK:

1. Prepare your development computer and ensure it meets the system requirements.
2. Install the SDK starter package from the table above. (If you're on Windows, download the installer for help with the initial setup.)
3. Install the ADT Plug-in for Eclipse (if you'll be developing in Eclipse).
4. Add Android platforms and other components to your SDK.
5. Explore the contents of the Android SDK (optional).

To get started, download the appropriate package from the table above, and then read the guide to [Installing the SDK](http://developer.android.com/sdk/installing.html).

# Installing the SDK

# Step 1: Preparing Your Development Computer

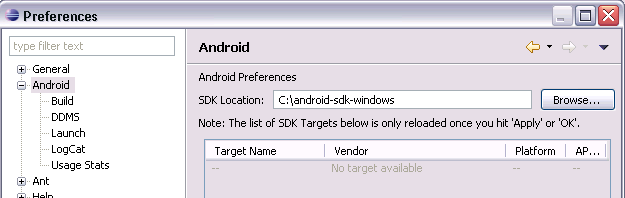
Before getting started with the Android SDK, take a moment to confirm that your development computer meets the [System Requirements](http://developer.android.com/sdk/requirements.html). In particular, you might need to install the [JDK](http://java.sun.com/javase/downloads/index.jsp), if you don't have it already.

If you will be developing in Eclipse with the Android Development Tools (ADT) Plug-in the recommended path if you are new to Android make sure that you have a suitable version of Eclipse installed on your computer as described in the [System Requirements](http://developer.android.com/sdk/requirements.html) document. If you need to install Eclipse, you can download it from this location:

The "Eclipse Classic" version is recommended. Otherwise, a Java or RCP version of Eclipse is recommended. Use the [Eclipse update manager](http://www.vogella.de/articles/Eclipse/article.html#updatemanager) to install all available plug-in for the Android Development Tools (ADT) from the URL <https://dl-ssl.google.com/android/eclipse/> .

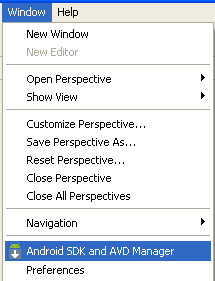
### Configuration

In Eclipse open the Preferences dialog via Windows -> Preferences. Select Android and maintain the installation path of the Android SDK.



**Preference Setting**

Select Window -> Android SDK and AVD Manager from the menu.



**Check Latest Version**

Select available packages and select the latest version of the SDK.

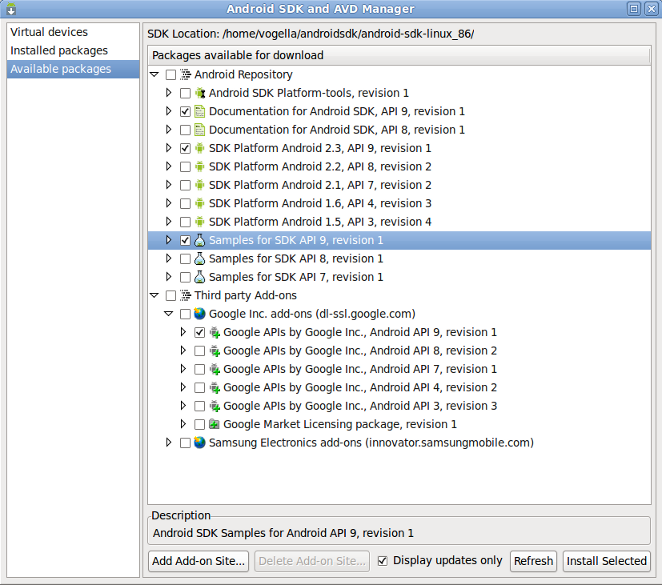
**Step 2: Downloading the SDK Starter Package**

The SDK starter package is not a full development environment it includes only the core SDK Tools, which you can use to download the rest of the SDK components (such as the latest Android platform).

If you haven't already, get the latest version of the SDK starter package from the [SDK download page](http://developer.android.com/sdk/index.html). If you downloaded a .zip or .tgz package (instead of the SDK installer), unpack it to a safe location on your machine. By default, the SDK files are unpacked into a directory named android-sdk-<machine-platform>.

If you downloaded the Windows installer (.exe file), run it now and it will check whether the proper Java SE Development Kit (JDK) is installed (installing it, if necessary), then install the SDK Tools into a default location (which you can modify).

Make a note of the name and location of the SDK directory on your system—you will need to refer to the SDK directory later, when setting up the ADT plugin and when using the SDK tools from the command line.



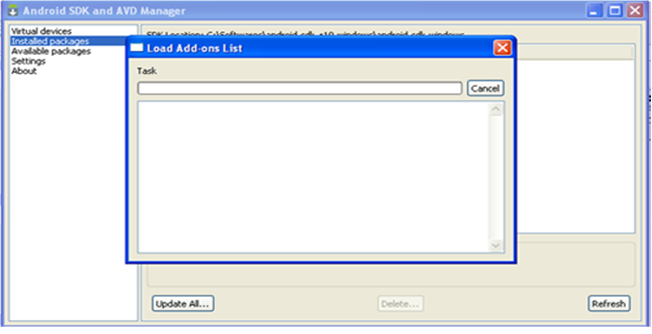
**Selecting Versions of Android**

## Step 3: Installing the ADT Plug-in for Eclipse

Android offers a custom plug-in for the Eclipse IDE, called Android Development Tools (ADT) that is designed to give you a powerful, integrated environment in which to build Android applications. It extends the capabilities of Eclipse to let you quickly set up new Android projects, create an application UI, debug your applications using the Android SDK tools, and even export signed (or unsigned) APKs in order to distribute your application. In general, developing in Eclipse with ADT is a highly recommended approach and is the fastest way to get started with Android.

If you'd like to use ADT for developing Android applications, install it now. Read [Installing the ADT Plug-in](http://developer.android.com/sdk/eclipse-adt.html#installing) for step-by-step installation instructions, then return here to continue the last step in setting up your Android SDK.

If you prefer to work in a different IDE, you do not need to install Eclipse or ADT. Instead, you can directly use the SDK tools to build and debug your application. The [Introduction](http://developer.android.com/guide/developing/index.html) to Android application development outlines the major steps that you need to complete when developing in Eclipse or other IDEs.

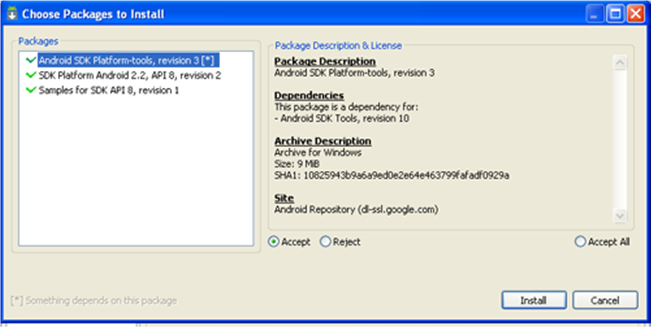


**Loading from Android Server**

## Step 4: Adding Platforms and Other Components

The last step in setting up your SDK is using the Android SDK and AVD Manager (a tool included in the SDK starter package) to download essential SDK components into your development environment.

The SDK uses a modular structure that separates the major parts of the SDK—Android platform versions, add-ons, tools, samples, and documentation—into a set of separately installable components. The SDK starter package, which you've already downloaded, includes only a single component: the latest version of the SDK Tools. To develop an Android application, you also need to download at least one Android platform and the associated platform tools. You can add other components and platforms as well, which is highly recommended.



**Select API**

If you used the Windows installer, when you complete the installation wizard, it will launch the Android SDK and AVD Manager with a default set of platforms and other components selected for you to install. Simply click Install to accept the recommended set of components and install them. You can then skip to [Step 5](http://developer.android.com/sdk/installing.html#sdkContents), but we recommend you first read the section about the [Available Components](http://developer.android.com/sdk/installing.html#components) to better understand the components available from the Android SDK and AVD Manager.

You can launch the Android SDK and AVD Manager in one of the following ways:

* From within Eclipse, select Window > Android SDK and AVD Manager.
* On Windows, double-click the SDK Manager.exe file at the root of the Android SDK directory.
* On Mac or Linux, open a terminal and navigate to the tools/ directory in the Android SDK, then execute.

To download components, use the graphical UI of the Android SDK and AVD Manager to browse the SDK repository and select new or updated components. The Android SDK and AVD Manager installs the selected components in your SDK environment. For information about which components you should download, see [Recommended Components](http://developer.android.com/sdk/installing.html#which).

The Android Repository offers these types of components:

* SDK Tools: Contains tools for debugging and testing your application and other utility tools. These tools are installed with the Android SDK starter package and receive periodic updates. You can access these tools in the <sdk>/tools/ directory of your SDK. To learn more about them, see [SDK Tools](http://developer.android.com/guide/developing/tools/index.html#tools-sdk) in the developer guide.
* SDK Platform-tools: Contains platform-dependent tools for developing and debugging your application. These tools support the latest features of the Android platform and are typically updated only when a new platform becomes available. You can access these tools in the <sdk>/platform-tools/ directory. To learn more about them, see [Platform Tools](http://developer.android.com/guide/developing/tools/index.html#tools-platform) in the developer guide.
* Android platforms: An SDK platform is available for every production Android platform deployable to Android-powered devices. Each SDK platform component includes a fully compliant Android library, system image, sample code, and emulator skins. To learn more about a specific platform, see the list of platforms that appears under the section "Downloadable SDK Components" on the left part of this page.
* USB Driver for Windows (Windows only): Contains driver files that you can install on your Windows computer, so that you can run and debug your applications on an actual device. You do not need the USB driver unless you plan to debug your application on an actual Android-powered device. If you develop on Mac OS X or Linux, you do not need a special driver to debug your application on an Android-powered device. See [Using Hardware Devices](http://developer.android.com/guide/developing/device.html) for more information about developing on a real device.
* Samples: Contains the sample code and apps available for each Android development platform. If you are just getting started with Android development, make sure to download the samples to your SDK.
* Documentation: Contains a local copy of the latest multi version documentation for the Android framework API.
* The Third party Add-ons provide components that allow you to create a development environment using a specific Android external library (such as the Google Maps library) or a customized (but fully compliant) Android system image. You can add additional Add-on repositories by clicking Add Add-on Site.

**ECLIPSE**

Eclipse is an open source community, whose projects are focused on building an extensible development platform, runtimes and application frameworks for building, deploying and managing software across the entire software lifecycle. Many people know us, and hopefully love us, as a Java IDE but Eclipse is much more than a Java IDE.

1. Enterprise Development
2. Embedded and Device Development
3. Rich Client Platform
4. Rich Internet Applications
5. Application Frameworks
6. Application Lifecycle Management (ALM)
7. Service Oriented Architecture (SOA)

**5.4 CREATING VIRTUAL CONSOLE FOR ANDROID**

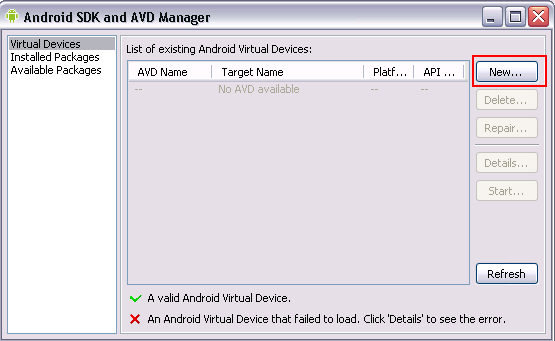
## Create an Android Emulator Device

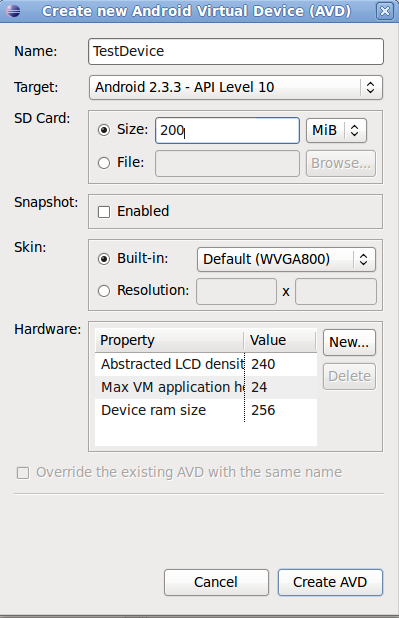
The Android tools include an emulator. This emulator behaves like a real Android device in most cases and allows you to test your application without having a real device. You can emulate one or several devices with different configurations. Each configuration is defined via an "Android Virtual Device" (AVD).

To define an AVD press the device manager button, press "New" and maintain the following.



**AVD Manager**

**Creation of New AVD**



**Setting Properties to AVD**

Press "Create AVD". This will create the device and display it under the "Virtual devices". To test if your setup is correct, select your device and press "Start".



**AVD Launched**

## Error handling

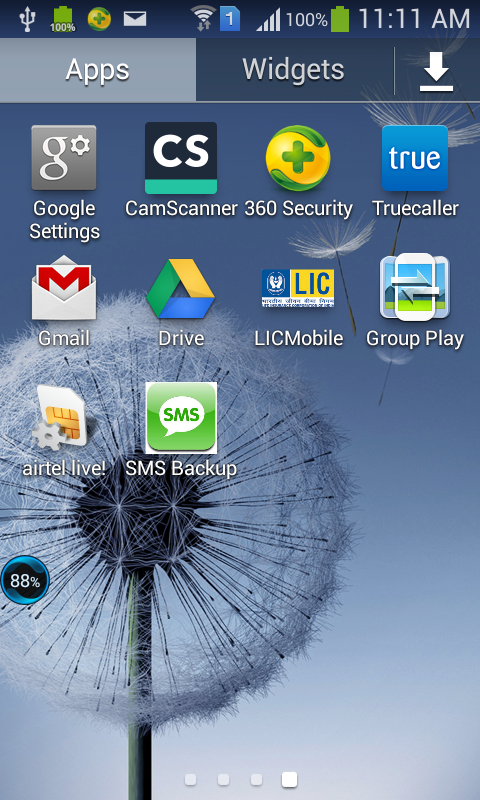
Things are not always working as they should be. Several users’ report that get the following errors:

1. Project is missing required source folder: 'gen'
2. The project could not be built until build path errors are resolved.
3. Unable to open class file R.java.

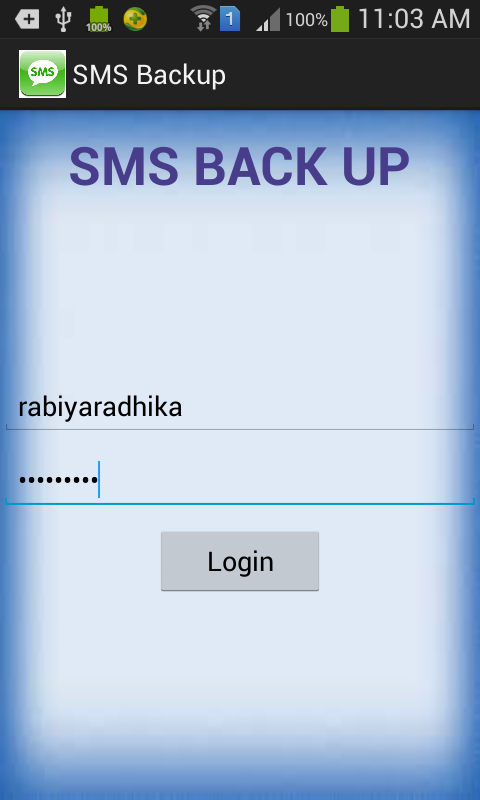
**5.5 SQLite DATABASE**

* SQLite is an Open Source database.
* It supports standard relational database features like SQL syntax, transactions and prepared statements.
* The database requires limited memory at runtime (approx. 250 KByte) which makes it a good candidate from being embedded into other runtimes.
* SQLite supports the data types TEXT, INTEGER and REAL (similar to string, long & double in Java respectively).
* All other types must be converted into one of these fields before getting saved in the database.
* SQLite is embedded into every Android device.
* Using a SQLite database in Android does not require a setup procedure or administration of the database.
* You only have to define the SQL statements for creating and updating the database.
* Afterwards the database is automatically managed for you by the Android platform.
* If the application creates a database, this database is by default saved in the directory DATA/data/APP\_NAME/databases/FILENAME.

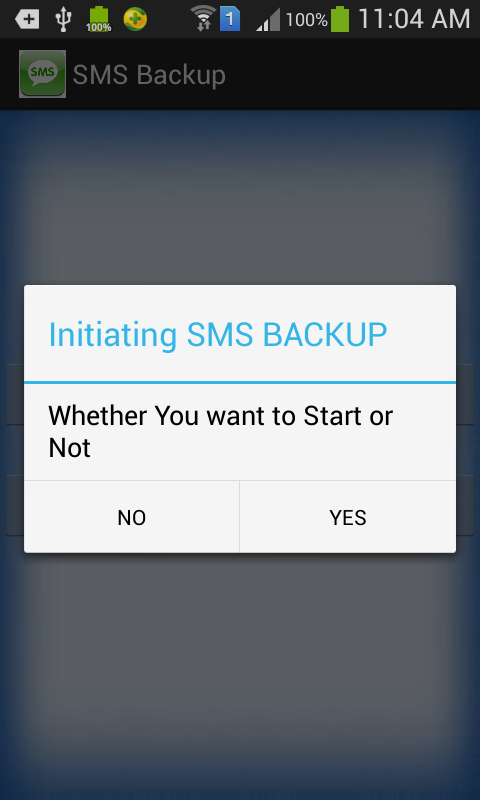
**5.6 OUTPUT SCREENS**

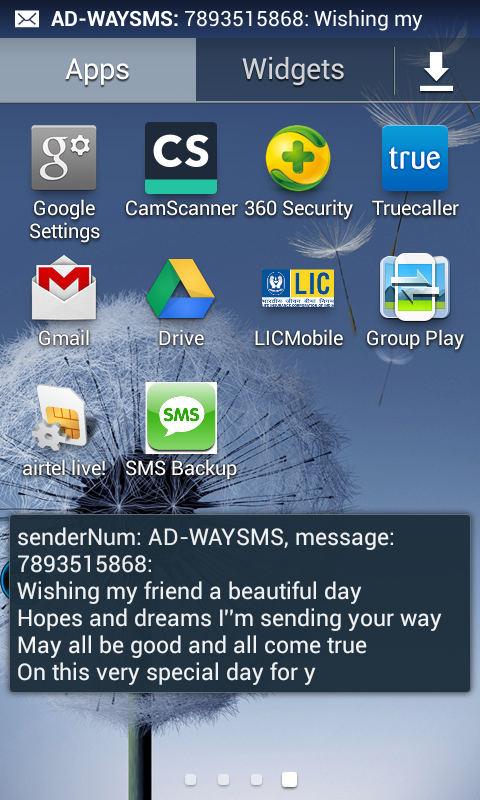
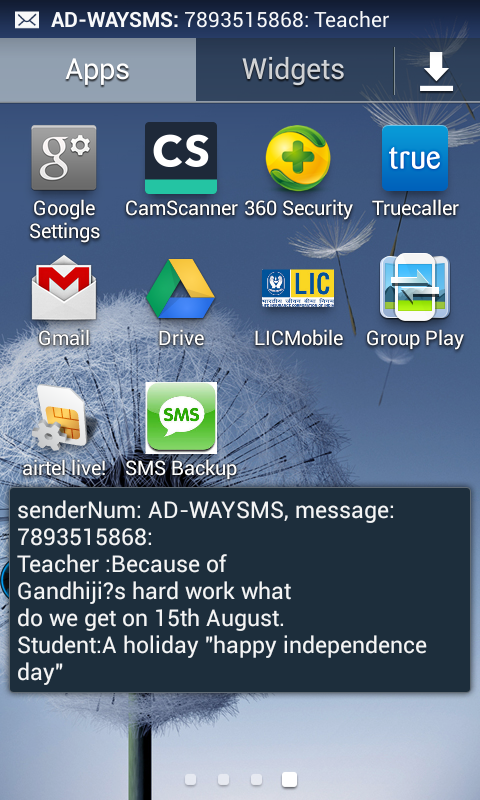
****

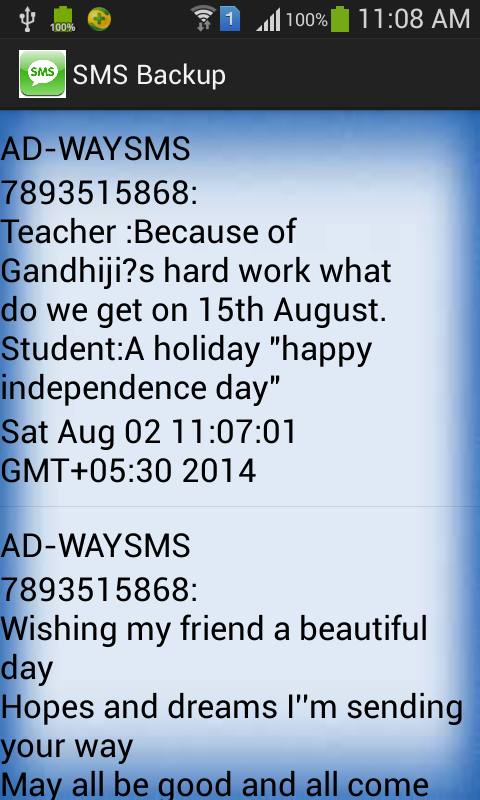
**Figure 5.3.1 Application Installed**

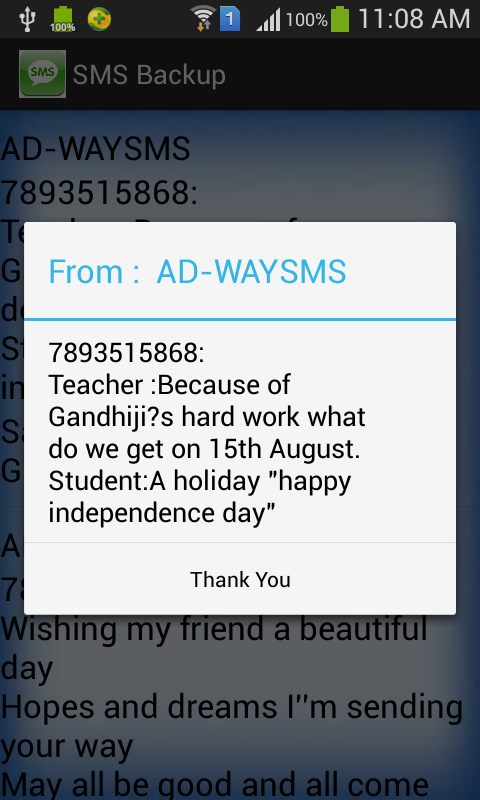
****

****

****

** **

****

****

**6. TESTING**

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the

Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

**TYPES OF TESTS**

**Unit testing**

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

**Integration testing**

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

**Functional test**

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems : interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

**System Test**

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

**White Box Testing**

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

**Black Box Testing**

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

**6.1 UNIT TESTING**

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

**Test objectives**

* All field entries must work properly.
* Pages must be activated from the identified link.
* The entry screen, messages and responses must not be delayed.

**Features to be tested**

* Verify that the entries are of the correct format
* No duplicate entries should be allowed
* All links should take the user to the correct page.

# 6.2 INTEGRATION TESTING

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects. The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

**6.3 ACCEPTANCE TESTING**

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

**7. CONCLUSION**

Finally we conclude that SMS BACK UP is still more to be embedded with new features and it should be done using wireless service also

1. **BIBLIOGRAPHY**

1. Software Engineering, A practitioner’s Approach- Roger S. Pressman, 6th edition McGrawHill International Edition.

2. Software Engineering- Sommerville, 7th edition, Pearson education.

3. Software Project Management, Walker Royce: Pearson Education, 2005.

4. Software Project Management, Bob Hughes and Mike Cotterell: Tata McGraw-Hill Edition.

5. Software Project Management, Joel Henry, Pearson Education.

6. Software Project Management in practice, Pankaj Jalote, Pearson Education.2005.

7. Software Testing techniques - Baris Beizer, Dreamtech, second edition.

8. Software Testing Tools – Dr.K.V.K.K.Prasad, Dreamtech.

9. Java; the complete reference, 7th editon, Herbert schildt, TMH.

10. Understanding OOP with Java, updated edition, T. Budd, pearson eduction.

11. Grady Booch, James Rumbaugh, Ivar Jacobson : The Unified Modeling  
Language User Guide, Pearson Education.

12. Hans-Erik Eriksson, Magnus Penker, Brian Lyons, David Fado: UML 2  
Toolkit, WILEY-Dreamtech India Pvt. Ltd.

**9. REFERENCE**

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

2. <http://www.androidhive.info/>

3. <http://www.vogella.com/android.html>

4. <http://www.tutorialspoint.com/android/>

5. <http://www.mkyong.com/tutorials/android-tutorial/>

6. <http://www.learn-android.com/category/beginner/>

7. <http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6171850>

8. <http://www.lynda.com/Android-training-tutorials/947-0.html>

9. <http://androidprogrammingstuff.blogspot.in/>

10. <http://mylifewithandroid.blogspot.in/>

11. <http://www.samsung.com/in/#latest-home>

12. <http://www.androidauthority.com/best-android-apps-for-writers-note-taking-56006/>